MQL5　言語レフェレンス

MetaTrader 5 クライアント端末

**MQL5の自学と
タスク解決に:**

- 様々な複雑度を持つ独自のテクニカル分析指標の作成
- 自動売買の使用 –
取引システムを自動化して様々な金融市場で動作
- 数学的な成果と伝統的な方法に基づいた独自の分析
ツールの開発
- トレーディング、モニタリング、アラートなどの幅広いタス
クを解決するための情報取引システムの作成

© 2000–2025, MetaQuotes Ltd

目次

MQL5 リファレンス 71

1 言語基礎 74

構文 75
コメント 76
識別子 77
予約語 78
データ型 80
整数型 81
Char、Short、Int 及び Long 型 82
文字定数 86
日付時刻型 89
カラー型 90
ブール型 91
列挙型 92
浮動小数点数型（double、float） 94
複素数（complex） 99
文字列型 100
構造体、クラス、インターフェイス 101
動的配列オブジェクト 128
行列とベクトル 129
型キャスト 136
Void 型とNULL 定数 141
ユーザー定義型 142
オブジェクトポインタ 152
参照： 修飾子「 & 」とキーワード「 this 」 156
演算子と式 158
式 159
算術演算子 160
代入演算子 161
関係の演算子 162
ブール演算子 163
ビット演算子 165
その他の演算子 168
優先順位のルール 172
演算子 174
重文演算子 175
式の演算子 176
Return 演算子 177
if-else 条件演算子 178
三項演算子 ?: 179
Switch 演算子 181
while 反復演算子 183
for 反復演算子 184
do while 反復演算子 185
Break 演算子 186
Continue 演算子 187
new オブジェクト作成演算子 188
delete オブジェクト解除演算子 190
関数 191
関数の呼び出し 193
パラメータの引き渡し 194
関数の多重定義（オーバーロード） 197
演算子の多重定義（オーバーロード） 200

© 2000-2025, MetaQuotes Ltd.

外部関数の記述 214
関数のエクスポート 216
イベント処理の関数 217
変数 228
ローカル変数 232
仮引数 234
静的変数 236
グローバル変数 237
Input変数 238
Extern 変数 244
変数の初期化 245
変数のアクセス権の有効範囲とライフタイム 247
オブジェクトの作成と解除 249
プリプロセッサ 252
マクロ代入(#define) 254
プログラムのプロパティ (#property) 257
ファイルの読み込み (#include) 263
関数のインポート(#import) 264
条件付きコンパイル(#ifdef、#ifndef、#else、#endif) 267
オブジェクト指向プログラミング 269
型の封装と拡張 271
継承 273
ポリモーフィズム 278
多重定義(オーバーロード) 282
仮想関数 283
クラスの静的メンバ 287
関数テンプレート 291
クラステンプレート 295
抽象クラス 300
名前空間 302
2 標準的な定数、列挙と構造体 305
チャート定数 306
チャートイベントの種類 307
チャート時間軸 314
チャートプロパティ 316
位置定数 323
チャート表示 324
チャート操作の例 326
オブジェクト定数 384
オブジェクト型 385
OBJ_VLINE 387
OBJ_HLINE 392
OBJ_TREND 397
OBJ_TRENDBYANGLE 404
OBJ_CYCLES 410
OBJ_ARROWED_LINE 416
OBJ_CHANNEL 422
OBJ_STDDEVCHANNEL 429
OBJ_REGRESSION 436
OBJ_PITCHFORK 442
OBJ_GANNLINE 450
OBJ_GANNFAN 457
OBJ_GANNGRID 464
OBJ_FIBO 471
OBJ_FIBOTIMES 478
OBJ_FIBOFAN 485
OBJ_FIBOARC 492
OBJ_FIBOCHANNEL 499

© 2000-2025, MetaQuotes Ltd.

OBJ_EXPANSION 507
OBJ_ELLIOTWAVE5 515
OBJ_ELLIOTWAVE3 523
OBJ_RECTANGLE 530
OBJ_TRIANGLE 536
OBJ_ELLIPSE 543
OBJ_ARROW_THUMB_UP 550
OBJ_ARROW_THUMB_DOWN 556
OBJ_ARROW_UP 563
OBJ_ARROW_DOWN 569
OBJ_ARROW_STOP 575
OBJ_ARROW_CHECK 581
OBJ_ARROW_LEFT_PRICE 587
OBJ_ARROW_RIGHT_PRICE 592
OBJ_ARROW_BUY 597
OBJ_ARROW_SELL 602
OBJ_ARROW 607
OBJ_TEXT 613
OBJ_LABEL 619
OBJ_BUTTON 627
OBJ_CHART 634
OBJ_BITMAP 642
OBJ_BITMAP_LABEL 649
OBJ_EDIT 657
OBJ_EVENT 664
OBJ_RECTANGLE_LABEL 669
オブジェクトプロパティ 675
オブジェクト結合の方法 703
チャートのコーナー 708
オブジェクトの可視性 711
エリオット波動のレベル 714
ギャンオブジェクト 715
ウェブカラー 717
特殊文字 719
指標定数 **720**
価格定数 721
平滑化の方法 724
指標線 725
描画スタイル 727
カスタム指標プロパティ 732
指標の種類 738
データ型識別子 740
環境状態 **741**
クライアント端末プロパティ 742
MQL5 プログラム実行プロパティ 747
シンボルプロパティ 751
口座プロパティ 832
テスト統計 839
取引定数 **843**
履歴データベースプロパティ 844
注文プロパティ 845
ポジションプロパティ 859
約定プロパティ 863
取引操作の種類 867
取引トランザクションの種類 879
DOM での取引注文 881
シグナルプロパティ 882
名前付き定数 **884**

© 2000-2025, MetaQuotes Ltd.

定義済みマクロ代入 885
数学定数 891
数値定数 892
初期化解除の理由のコード 895
オブジェクトポインタのチェック 897
その他の定数 898
データ構造体 902
データ型構造体 903
入力パラメータの構造体 904
履歴データの構造体 905
DOM構造体 906
取引リクエスト構造体 907
小切手リクエスト構造体 921
取引リクエスト結果の構造体 922
取引処理構造体 925
現在価格の構造体 933
経済指標カレンダー構造体 934
エラーコードと警告コード 944
取引サーバのリターンコード 945
コンパイラ警告 948
は、コンパイルエラー 951
ランタイムエラー 961
入出力定数 973
ファイルを開く際のフラグ 974
ファイルプロパティ 976
ファイル内の位置 977
コードページの利用 978
メッセージボックス 979
3 MQL5 プログラム 981
プログラム実行 982
取引許可 989
クライアント端末イベント 993
リソース 996
インポートした関数の呼び出し 1008
ランタイムエラー 1010
取引ストラテジーのテスト 1011
4 定義済み変数 1036
_AppliedTo 1037
_Digits 1039
_Point 1040
_LastError 1041
_Period 1042
_RandomSeed 1043
_StopFlag 1044
_Symbol 1045
_UninitReason 1046
_IsX64 1047
5 共通関数 1048
Alert 1050
CheckPointer 1055
Comment 1057
CryptEncode 1059
CryptDecode 1061
DebugBreak 1063
ExpertRemove 1064
GetPointer 1066
GetTickCount 1070

© 2000-2025, MetaQuotes Ltd.

GetTickCount64 1071
GetMicrosecondCount 1073
MessageBox 1075
PeriodSeconds 1080
PlaySound 1082
Print 1085
PrintFormat 1087
ResetLastError 1092
ResourceCreate 1093
ResourceFree 1097
ResourceReadImage 1100
ResourceSave 1104
SetReturnError 1109
SetUserError 1111
Sleep 1112
TerminalClose 1114
TesterHideIndicators 1116
TesterStatistics 1118
TesterStop 1120
TesterDeposit 1122
TesterWithdrawal 1125
TranslateKey 1128
ZeroMemory 1129

6 配列関数 1132

ArrayBsearch 1134
ArrayCopy 1138
ArrayCompare 1143
ArrayFree 1146
ArrayGetAsSeries 1155
ArrayInitialize 1158
ArrayFill 1160
ArrayIsDynamic 1162
ArrayIsSeries 1164
ArrayMaximum 1166
ArrayMinimum 1178
ArrayPrint 1190
ArrayRange 1193
ArrayResize 1194
ArrayInsert 1197
ArrayRemove 1199
ArrayReverse 1201
ArraySetAsSeries 1203
ArraySize 1206
ArraySort 1208
ArraySwap 1213
ArrayToFP16 1215
ArrayToFP8 1218
ArrayFromFP16 1221
ArrayFromFP8 1224

7 行列とベクトルのメソッド 1227

行列とベクトルの種類 1233
列挙型 1234
初期化 1239
Assign 1242
CopyIndicatorBuffer 1244
CopyRates 1246
CopyTicks 1252
CopyTicksRange 1255

© 2000–2025, MetaQuotes Ltd.

Eye ... 1257
Identity ... 1259
Ones ... 1261
Zeros ... 1262
Full ... 1263
Tri ... 1264
Init ... 1265
Fill ... 1267
行列とベクトルの操作 ... **1268**
HasNan ... 1269
Transpose ... 1270
TransposeConjugate ... 1272
TriL ... 1273
TriU ... 1274
Diag ... 1275
Row ... 1277
Col ... 1279
Copy ... 1281
Compare ... 1283
CompareByDigits ... 1285
CompareEqual ... 1287
Flat ... 1290
Clip ... 1292
Reshape ... 1293
Resize ... 1295
Set ... 1297
SwapRows ... 1299
SwapCols ... 1300
Split ... 1301
Hsplit ... 1303
Vsplit ... 1305
ArgSort ... 1307
Sort ... 1308
解を求める ... **1310**
数学演算 ... 1312
数学関数 ... 1313
行列とベクトルの積 ... **1314**
MatMul ... 1315
GeMM ... 1320
Power ... 1324
Dot ... 1327
Kron ... 1329
Inner ... 1331
Outer ... 1333
CorrCoef ... 1337
Cov ... 1340
Correlate ... 1343
Convolve ... 1346
行列変換 ... **1349**
Cholesky ... 1350
Eig ... 1351
EigVals ... 1354
LU ... 1355
LUP ... 1357
QR ... 1359
SVD ... 1361
統計メソッド ... **1364**
ArgMax ... 1365

© 2000–2025, MetaQuotes Ltd.

ArgMin 1366
Max 1367
Min 1368
Ptp 1369
Sum 1370
Prod 1371
CumSum 1373
CumProd 1375
Percentile 1377
Quantile 1379
Median 1381
Mean 1383
Average 1384
Std 1386
Var 1388
LinearRegression 1390
特徴メソッド **1393**
Rows 1394
Cols 1395
Size 1396
Norm 1397
Cond 1400
Det 1403
SLogDet 1405
Rank 1406
Trace 1409
Spectrum 1410
解 **1411**
Solve 1412
LstSq 1413
Inv 1414
PInv 1416
機械学習 **1417**
Activation 1422
Derivative 1426
Loss 1428
LossGradient 1430
RegressionMetric 1432
ConfusionMatrix 1434
ConfusionMatrixMultilabel 1436
ClassificationMetric 1438
ClassificationScore 1442
PrecisionRecall 1447
ReceiverOperatingCharacteristic 1451
OpenBLAS **1455**
Singular Value Decomposition 1458
SingularValueDecompositionDC 1459
SingularValueDecompositionQR 1461
SingularValueDecompositionQRPivot 1463
SingularValueDecompositionBisect 1466
SingularValueDecompositionJacobiHigh 1469
SingularValueDecompositionJacobiLow 1474
SingularValueDecompositionBidiagDC 1477
SingularValueDecompositionBidiagBisect 1480
Eigen Values 1483
General Matrices 1485
EigenSolver 1486
EigenSolverX 1488

© 2000–2025, MetaQuotes Ltd.

EigenSolverShur 1493
EigenSolver2 1495
EigenSolver2X 1498
EigenSolver2Shur 1502
EigenSolver2Blocked 1504
EigenSolver2ShurBlocked 1507
Symmetric Matrices 1510
EigenSymmetricDC 1511
EigenSymmetricQR 1513
EigenSymmetricRobust 1515
EigenSymmetricBisect 1518
Singular Spectrum Analysis 1521
SingularSpectrumAnalysisSpectrum 1522
SingularSpectrumAnalysisForecast 1523
SingularSpectrumAnalysisReconstructComponents 1524
SingularSpectrumAnalysisReconstructSeries 1525

8 変換関数 1526

CharToString 1527
CharArrayToString 1529
CharArrayToStruct 1531
StructToCharArray 1535
ColorToARGB 1538
ColorToString 1540
DoubleToString 1541
EnumToString 1542
IntegerToString 1544
ShortToString 1546
ShortArrayToString 1548
TimeToString 1550
NormalizeDouble 1553
StringToCharArray 1555
StringToColor 1557
StringToDouble 1558
StringToInteger 1559
StringToShortArray 1560
StringToTime 1562
StringFormat 1565

9 数学関数 1570

MathAbs 1572
MathArccos 1573
MathArcsin 1576
MathArctan 1579
MathArctan2 1582
MathClassify 1586
MathCeil 1588
MathCos 1590
MathExp 1593
MathFloor 1596
MathLog 1598
MathLog10 1602
MathMax 1606
MathMin 1608
MathMod 1610
MathPow 1611
MathRand 1615
MathRound 1616
MathSin 1618
MathSqrt 1621

© 2000–2025, MetaQuotes Ltd.

MathSrand 1625
MathTan 1628
MathIsValidNumber 1631
MathExpm1 1632
MathLog1p 1636
MathArccosh 1640
MathArcsinh 1643
MathArctanh 1646
MathCosh 1649
MathSinh 1652
MathTanh 1655
MathSwap 1658
10 文字列関数 1661
StringAdd 1662
StringBufferLen 1664
StringCompare 1665
StringConcatenate 1667
StringFill 1669
StringFind 1670
StringGetCharacter 1674
StringInit 1676
StringLen 1677
StringSetLength 1678
StringReplace 1680
StringReserve 1681
StringSetCharacter 1683
StringSplit 1685
StringSubstr 1687
StringToLower 1688
StringToUpper 1689
StringTrimLeft 1690
StringTrimRight 1691
11 日付と時刻 1692
TimeCurrent 1693
TimeTradeServer 1695
TimeLocal 1697
TimeGMT 1699
TimeDaylightSavings 1701
TimeGMTOffset 1702
TimeToStruct 1703
StructToTime 1704
12 口座情報 1706
AccountInfoDouble 1707
AccountInfoInteger 1708
AccountInfoString 1710
13 チェックアップ 1711
GetLastError 1712
IsStopped 1713
UninitializeReason 1714
TerminalInfoInteger 1716
TerminalInfoDouble 1717
TerminalInfoString 1718
MQLInfoInteger 1719
MQLInfoString 1720
Symbol 1721
Period 1722
Digits 1723

© 2000-2025, MetaQuotes Ltd.

Point 1724
14 イベント処理 1725
OnStart 1727
OnInit 1730
OnDeinit 1733
OnTick 1736
OnCalculate 1743
OnTimer 1747
OnTrade 1750
OnTradeTransaction 1755
OnBookEvent 1761
OnChartEvent 1765
OnTester 1772
OnTesterInit 1779
OnTesterDeinit 1786
OnTesterPass 1787
15 市場情報 1788
SymbolsTotal 1789
SymbolExist 1790
SymbolName 1792
SymbolSelect 1794
SymbolIsSynchronized 1796
SymbolInfoDouble 1798
SymbolInfoInteger 1800
SymbolInfoString 1802
SymbolInfoMarginRate 1804
SymbolInfoTick 1806
SymbolInfoSessionQuote 1807
SymbolInfoSessionTrade 1809
MarketBookAdd 1811
MarketBookRelease 1813
MarketBookGet 1815
16 経済指標カレンダー 1816
CalendarCountryById 1817
CalendarEventById 1819
CalendarValueById 1822
CalendarCountries 1825
CalendarEventByCountry 1827
CalendarEventByCurrency 1829
CalendarValueHistoryByEvent 1831
CalendarValueHistory 1835
CalendarValueLastByEvent 1838
CalendarValueLast 1844
17 時系列と指標へのアクセス 1849
配列、バッファ及び時系列における索引付けの方向 1853
データアクセスの整理 1856
SeriesInfoInteger 1865
Bars 1867
BarsCalculated 1870
IndicatorCreate 1872
IndicatorParameters 1874
IndicatorRelease 1876
CopyBuffer 1878
CopyRates 1882
CopySeries 1886
CopyTime 1890
CopyOpen 1893

© 2000–2025, MetaQuotes Ltd.

CopyHigh 1896
CopyLow 1900
CopyClose 1903
CopyTickVolume 1906
CopyRealVolume 1910
CopySpread 1913
CopyTicks 1917
CopyTicksRange 1923
iBars 1925
iBarShift 1926
iClose 1929
iHigh 1931
iHighest 1933
iLow 1934
iLowest 1936
iOpen 1937
iTime 1939
iTickVolume 1941
iRealVolume 1943
iVolume 1945
iSpread 1947

18 カスタム銘柄 1949

CustomSymbolCreate 1950
CustomSymbolDelete 1953
CustomSymbolSetInteger 1955
CustomSymbolSetDouble 1959
CustomSymbolSetString 1963
CustomSymbolSetMarginRate 1967
CustomSymbolSetSessionQuote 1972
CustomSymbolSetSessionTrade 1978
CustomRatesDelete 1984
CustomRatesReplace 1989
CustomRatesUpdate 1995
CustomTicksAdd 2001
CustomTicksDelete 2009
CustomTicksReplace 2018
CustomBookAdd 2028

19 チャート操作 2031

ChartApplyTemplate 2033
ChartSaveTemplate 2036
ChartWindowFind 2041
ChartTimePriceToXY 2043
ChartXYToTimePrice 2045
ChartOpen 2047
ChartFirst 2049
ChartNext 2050
ChartClose 2051
ChartSymbol 2053
ChartPeriod 2054
ChartRedraw 2055
ChartSetDouble 2058
ChartSetInteger 2060
ChartSetString 2062
ChartGetDouble 2064
ChartGetInteger 2066
ChartGetString 2068
ChartNavigate 2070
ChartID 2073

© 2000–2025, MetaQuotes Ltd.

ChartIndicatorAdd 2075
ChartIndicatorDelete 2079
ChartIndicatorGet 2082
ChartIndicatorName 2084
ChartIndicatorsTotal 2086
ChartWindowOnDropped 2088
ChartPriceOnDropped 2089
ChartTimeOnDropped 2090
ChartXOnDropped 2091
ChartYOnDropped 2092
ChartSetSymbolPeriod 2093
ChartScreenShot 2095
20 取引関数 2098
OrderCalcMargin 2100
OrderCalcProfit 2104
OrderCheck 2107
OrderSend 2111
OrderSendAsync 2116
PositionsTotal 2127
PositionGetSymbol 2128
PositionSelect 2130
PositionSelectByTicket 2132
PositionGetDouble 2134
PositionGetInteger 2137
PositionGetString 2139
PositionGetTicket 2142
OrdersTotal 2144
OrderGetTicket 2145
OrderSelect 2147
OrderGetDouble 2151
OrderGetInteger 2155
OrderGetString 2161
HistorySelect 2164
HistorySelectByPosition 2166
HistoryOrderSelect 2170
HistoryOrdersTotal 2173
HistoryOrderGetTicket 2174
HistoryOrderGetDouble 2176
HistoryOrderGetInteger 2180
HistoryOrderGetString 2183
HistoryDealSelect 2186
HistoryDealsTotal 2190
HistoryDealGetTicket 2191
HistoryDealGetDouble 2194
HistoryDealGetInteger 2199
HistoryDealGetString 2202
21 売買シグナル 2206
SignalBaseGetDouble 2207
SignalBaseGetInteger 2208
SignalBaseGetString 2209
SignalBaseSelect 2210
SignalBaseTotal 2211
SignalInfoGetDouble 2212
SignalInfoGetInteger 2213
SignalInfoGetString 2214
SignalInfoSetDouble 2215
SignalInfoSetInteger 2216
SignalSubscribe 2217

© 2000–2025, MetaQuotes Ltd.

SignalUnSubscribe 2218
22 ネットワーク関数 2219
SocketCreate 2221
SocketClose 2224
SocketConnect 2227
SocketIsConnected 2230
SocketIsReadable 2234
SocketIsWritable 2237
SocketTimeouts 2241
SocketRead 2243
SocketSend 2247
SocketTlsHandshake 2251
SocketTlsCertificate 2256
SocketTlsRead 2259
SocketTlsReadAvailable 2263
SocketTlsSend 2268
WebRequest 2272
SendFTP 2275
SendMail 2277
SendNotification 2279
23 端末のグローバル変数 2281
GlobalVariableCheck 2282
GlobalVariableTime 2283
GlobalVariableDel 2284
GlobalVariableGet 2285
GlobalVariableName 2286
GlobalVariableSet 2287
GlobalVariablesFlush 2288
GlobalVariableTemp 2289
GlobalVariableSetOnCondition 2290
GlobalVariablesDeleteAll 2291
GlobalVariablesTotal 2292
24 ファイル関数 2293
FileSelectDialog 2296
FileFindFirst 2298
FileFindNext 2300
FileFindClose 2302
FileIsExist 2304
FileOpen 2307
FileClose 2310
FileCopy 2311
FileDelete 2314
FileMove 2316
FileFlush 2319
FileGetInteger 2321
FileIsEnding 2324
FileIsLineEnding 2326
FileReadArray 2331
FileReadBool 2333
FileReadDatetime 2336
FileReadDouble 2339
FileReadFloat 2342
FileReadInteger 2345
FileReadLong 2349
FileReadNumber 2352
FileReadString 2357
FileReadStruct 2359

© 2000–2025, MetaQuotes Ltd.

FileSeek 2363
FileSize 2366
FileTell 2368
FileWrite 2371
FileWriteArray 2374
FileWriteDouble 2377
FileWriteFloat 2380
FileWriteInteger 2382
FileWriteLong 2385
FileWriteString 2387
FileWriteStruct 2390
FileLoad 2394
FileSave 2396
FolderCreate 2398
FolderDelete 2401
FolderClean 2404

25 カスタム指標 2407

指標スタイルの例 2410
DRAW_NONE 2419
DRAW_LINE 2422
DRAW_SECTION 2426
DRAW_HISTOGRAM 2430
DRAW_HISTOGRAM2 2434
DRAW_ARROW 2438
DRAW_ZIGZAG 2443
DRAW_FILLING 2448
DRAW_BARS 2453
DRAW_CANDLES 2459
DRAW_COLOR_LINE 2466
DRAW_COLOR_SECTION 2471
DRAW_COLOR_HISTOGRAM 2476
DRAW_COLOR_HISTOGRAM2 2481
DRAW_COLOR_ARROW 2486
DRAW_COLOR_ZIGZAG 2492
DRAW_COLOR_BARS 2497
DRAW_COLOR_CANDLES 2503
指標プロパティと関数との関係 2510
SetIndexBuffer 2512
IndicatorSetDouble 2515
IndicatorSetInteger 2519
IndicatorSetString 2523
PlotIndexSetDouble 2526
PlotIndexSetInteger 2527
PlotIndexSetString 2531
PlotIndexGetInteger 2532

26 オブジェクト関数 2535

ObjectCreate 2537
ObjectName 2541
ObjectDelete 2542
ObjectsDeleteAll 2543
ObjectFind 2544
ObjectGetTimeByValue 2545
ObjectGetValueByTime 2546
ObjectMove 2547
ObjectsTotal 2548
ObjectSetDouble 2549
ObjectSetInteger 2553
ObjectSetString 2557

© 2000-2025, MetaQuotes Ltd.

ObjectGetDouble 2559
ObjectGetInteger 2561
ObjectGetString 2563
TextSetFont 2565
TextOut 2567
TextGetSize 2571
27 テクニカル指標 2572
iAC 2575
iAD 2580
iADX 2585
iADXWilder 2591
iAlligator 2597
iAMA 2604
iAO 2609
iATR 2614
iBearsPower 2619
iBands 2624
iBullsPower 2630
iCCI 2635
iChaikin 2640
iCustom 2645
iDEMA 2648
iDeMarker 2653
iEnvelopes 2658
iForce 2664
iFractals 2669
iFrAMA 2674
iGator 2679
iIchimoku 2686
iBWMFI 2693
iMomentum 2698
iMFI 2703
iMA 2708
iOsMA 2713
iMACD 2718
iOBV 2724
iSAR 2729
iRSI 2734
iRVI 2739
iStdDev 2744
iStochastic 2749
iTEMA 2755
iTriX 2760
iWPR 2765
iVIDyA 2770
iVolumes 2775
28 最適化の結果の操作 2780
FrameFirst 2782
FrameFilter 2783
FrameNext 2784
FrameInputs 2785
FrameAdd 2786
ParameterGetRange 2787
ParameterSetRange 2790
29 イベント操作 2792
EventSetMillisecondTimer 2793
EventSetTimer 2794

© 2000–2025, MetaQuotes Ltd.

EventKillTimer 2795
EventChartCustom 2796
30 OpenCL 操作 2802
CLHandleType 2804
CLGetInfoInteger 2805
CLGetInfoString 2808
CLContextCreate 2811
CLContextFree 2812
CLGetDeviceInfo 2813
CLProgramCreate 2818
CLProgramFree 2823
CLKernelCreate 2824
CLKernelFree 2825
CLSetKernelArg 2826
CLSetKernelArgMem 2827
CLSetKernelArgMemLocal 2828
CLBufferCreate 2829
CLBufferFree 2830
CLBufferWrite 2831
CLBufferRead 2836
CLExecute 2840
CLExecutionStatus 2842
31 データベースの操作 2843
DatabaseOpen 2846
DatabaseClose 2848
DatabaseImport 2849
DatabaseExport 2852
DatabasePrint 2858
DatabaseTableExists 2863
DatabaseExecute 2864
DatabasePrepare 2878
DatabaseReset 2887
DatabaseBind 2894
DatabaseBindArray 2900
DatabaseRead 2905
DatabaseReadBind 2906
DatabaseFinalize 2910
DatabaseTransactionBegin 2911
DatabaseTransactionCommit 2916
DatabaseTransactionRollback 2917
DatabaseColumnsCount 2918
DatabaseColumnName 2919
DatabaseColumnType 2920
DatabaseColumnSize 2921
DatabaseColumnText 2922
DatabaseColumnInteger 2923
DatabaseColumnLong 2924
DatabaseColumnDouble 2925
DatabaseColumnBlob 2926
32 DirectXの操作 2927
DXContextCreate 2929
DXContextSetSize 2930
DXContextGetSize 2931
DXContextClearColors 2932
DXContextClearDepth 2933
DXContextGetColors 2934
DXContextGetDepth 2935

© 2000–2025, MetaQuotes Ltd.

DXBufferCreate 2936
DXTextureCreate 2937
DXInputCreate 2943
DXInputSet 2944
DXShaderCreate 2945
DXShaderSetLayout 2946
DXShaderInputsSet 2947
DXShaderTexturesSet 2948
DXDraw 2949
DXDrawIndexed 2950
DXPrimiveTopologySet 2951
DXBufferSet 2952
DXShaderSet 2953
DXHandleType 2954
DXRelease 2955
33 Python用MetaTrader 2956
initialize 2962
login 2964
shutdown 2967
version 2968
last_error 2970
account_info 2972
terminal_info 2975
symbols_total 2978
symbols_get 2979
symbol_info 2982
symbol_info_tick 2986
symbol_select 2988
market_book_add 2992
market_book_get 2993
market_book_release 2996
copy_rates_from 2997
copy_rates_from_pos 3001
copy_rates_range 3004
copy_ticks_from 3007
copy_ticks_range 3011
orders_total 3014
orders_get 3015
order_calc_margin 3018
order_calc_profit 3021
order_check 3024
order_send 3028
positions_total 3033
positions_get 3034
history_orders_total 3037
history_orders_get 3039
history_deals_total 3042
history_deals_get 3044
34 ONNXモデル 3048
ONNXサポート 3049
フォーマット変換 3051
データ型の自動変換 3052
モデルの作成 3055
モデルの実行 3062
ストラテジーテスターでの実行 3068
OnnxCreate 3073
OnnxCreateFromBuffer 3074
OnnxRelease 3075

© 2000–2025, MetaQuotes Ltd.

OnnxRun 3076
OnnxGetInputCount 3079
OnnxGetOutputCount 3080
OnnxGetInputName 3081
OnnxGetOutputName 3082
OnnxGetInputTypeInfo 3083
OnnxGetOutputTypeInfo 3084
OnnxSetInputShape 3085
OnnxSetOutputShape 3086
データ構造体 3087
35 標準ライブラリ 3091
数学 3092
統計 3093
統計的特性 3096
MathMean 3097
MathVariance 3098
MathSkewness 3099
MathKurtosis 3100
MathMoments 3101
MathMedian 3102
MathStandardDeviation 3103
MathAverageDeviation 3104
正規分布 3105
MathProbabilityDensityNormal 3109
MathCumulativeDistributionNormal 3111
MathQuantileNormal 3113
MathRandomNormal 3115
MathMomentsNormal 3116
対数正規分布 3117
MathProbabilityDensityLognormal 3121
MathCumulativeDistributionLognormal 3123
MathQuantileLognormal 3125
MathRandomLognormal 3127
MathMomentsLognormal 3128
ベータ分布 3129
MathProbabilityDensityBeta 3133
MathCumulativeDistributionBeta 3135
MathQuantileBeta 3137
MathRandomBeta 3139
MathMomentsBeta 3140
非心ベータ分布 3141
MathProbabilityDensityNoncentralBeta 3145
MathCumulativeDistributionNoncentralBeta 3147
MathQuantileNoncentralBeta 3149
MathRandomNoncentralBeta 3151
MathMomentsNoncentralBeta 3152
ガンマ分布 3153
MathProbabilityDensityGamma 3157
MathCumulativeDistributionGamma 3159
MathQuantileGamma 3161
MathRandomGamma 3163
MathMomentsGamma 3164
カイ2乗分布 3165
MathProbabilityDensityChiSquare 3169
MathCumulativeDistributionChiSquare 3171
MathQuantileChiSquare 3173
MathRandomChiSquare 3175
MathMomentsChiSquare 3176

© 2000–2025, MetaQuotes Ltd.

非心カイ2乗分布 3177
MathProbabilityDensityNoncentralChiSquare 3181
MathCumulativeDistributionNoncentralChiSquare 3183
MathQuantileNoncentralChiSquare 3185
MathRandomNoncentralChiSquare 3187
MathMomentsNoncentralChiSquare 3188
指数分布 3189
MathProbabilityDensityExponential 3193
MathCumulativeDistributionExponential 3195
MathQuantileExponential 3197
MathRandomExponential 3199
MathMomentsExponential 3200
F分布 3201
MathProbabilityDensityF 3205
MathCumulativeDistributionF 3207
MathQuantileF 3209
MathRandomF 3211
MathMomentsF 3212
非心F分布 3213
MathProbabilityDensityNoncentralF 3217
MathCumulativeDistributionNoncentralF 3219
MathQuantileNoncentralF 3221
MathRandomNoncentralF 3223
MathMomentsNoncentralF 3224
t分布 3225
MathProbabilityDensityT 3229
MathCumulativeDistributionT 3231
MathQuantileT 3233
MathRandomT 3235
MathMomentsT 3236
非心t分布 3237
MathProbabilityDensityNoncentralT 3241
MathCumulativeDistributionNoncentralT 3243
MathQuantileNoncentralT 3245
MathRandomNoncentralT 3247
MathMomentsNoncentralT 3248
ロジスティック分布 3249
MathProbabilityDensityLogistic 3253
MathCumulativeDistributionLogistic 3255
MathQuantileLogistic 3257
MathRandomLogistic 3259
MathMomentsLogistic 3260
コーシー分布 3261
MathProbabilityDensityCauchy 3265
MathCumulativeDistributionCauchy 3267
MathQuantileCauchy 3269
MathRandomCauchy 3271
MathMomentsCauchy 3272
一様分布 3273
MathProbabilityDensityUniform 3277
MathCumulativeDistributionUniform 3279
MathQuantileUniform 3281
MathRandomUniform 3283
MathMomentsUniform 3284
ワイブル分布 3285
MathProbabilityDensityWeibull 3289
MathCumulativeDistributionWeibull 3291
MathQuantileWeibull 3293

© 2000-2025, MetaQuotes Ltd.

MathRandomWeibull 3295
MathMomentsWeibull 3296
二項分布 3297
MathProbabilityDensityBinomial 3300
MathCumulativeDistributionBinomial 3302
MathQuantileBinomial 3304
MathRandomBinomial 3306
MathMomentsBinomial 3307
負の二項分布 3308
MathProbabilityDensityNegativeBinomial 3311
MathCumulativeDistributionNegativeBinomial 3313
MathQuantileNegativeBinomial 3315
MathRandomNegativeBinomial 3317
MathMomentsNegativeBinomial 3318
幾何分布 3319
MathProbabilityDensityGeometric 3323
MathCumulativeDistributionGeometric 3325
MathQuantileGeometric 3327
MathRandomGeometric 3329
MathMomentsGeometric 3330
超幾何分布 3331
MathProbabilityDensityHypergeometric 3335
MathCumulativeDistributionHypergeometric 3337
MathQuantileHypergeometric 3339
MathRandomHypergeometric 3341
MathMomentsHypergeometric 3342
ポアソン分布 3343
MathProbabilityDensityPoisson 3347
MathCumulativeDistributionPoisson 3349
MathQuantilePoisson 3351
MathRandomPoisson 3353
MathMomentsPoisson 3354
サブ関数 3355
MathRandomNonZero 3359
MathMoments 3360
MathPowInt 3361
MathFactorial 3362
MathTrunc 3363
MathRound 3364
MathArctan2 3366
MathGamma 3368
MathGammaLog 3369
MathBeta 3370
MathBetaLog 3371
MathBetaIncomplete 3372
MathGammaIncomplete 3373
MathBinomialCoefficient 3374
MathBinomialCoefficientLog 3375
MathHypergeometric2F2 3376
MathSequence 3377
MathSequenceByCount 3378
MathReplicate 3379
MathReverse 3380
MathIdentical 3381
MathUnique 3382
MathQuickSortAscending 3383
MathQuickSortDescending 3384
MathQuickSort 3385

© 2000–2025, MetaQuotes Ltd.

MathOrder 3386
MathBitwiseNot 3387
MathBitwiseAnd 3388
MathBitwiseOr 3389
MathBitwiseXor 3390
MathBitwiseShiftL 3391
MathBitwiseShiftR 3392
MathCumulativeSum 3393
MathCumulativeProduct 3394
MathCumulativeMin 3395
MathCumulativeMax 3396
MathSin 3397
MathCos 3398
MathTan 3399
MathArcsin 3400
MathArccos 3401
MathArctan 3402
MathSinPi 3403
MathCosPi 3404
MathTanPi 3405
MathAbs 3406
MathCeil 3407
MathFloor 3408
MathSqrt 3409
MathExp 3410
MathPow 3411
MathLog 3412
MathLog2 3413
MathLog10 3414
MathLog1p 3415
MathDifference 3416
MathSample 3418
MathTukeySummary 3420
MathRange 3421
MathMin 3422
MathMax 3423
MathSum 3424
MathProduct 3425
MathStandardDeviation 3426
MathAverageDeviation 3427
MathMedian 3428
MathMean 3429
MathVariance 3430
MathSkewness 3431
MathKurtosis 3432
MathExpm1 3433
MathSinh 3434
MathCosh 3435
MathTanh 3436
MathArcsinh 3437
MathArccosh 3438
MathArctanh 3439
MathSignif 3440
MathRank 3442
MathCorrelationPearson 3443
MathCorrelationSpearman 3444
MathCorrelationKendall 3445
MathQuantile 3446

© 2000–2025, MetaQuotes Ltd.

MathProbabilityDensityEmpirical 3447
MathCumulativeDistributionEmpirical 3448
ファジー論理 3449
メンバーシップ関数 3450
CConstantMembershipFunction 3451
GetValue 3453
CCompositeMembershipFunction 3454
CompositionType 3456
MembershipFunctions 3456
GetValue 3456
CDifferencTwoSigmoidalMembershipFunction 3458
A1 3460
A2 3460
C1 3461
C2 3461
GetValue 3462
CGeneralizedBellShapedMembershipFunction 3463
A 3465
B 3465
C 3466
GetValue 3466
CNormalCombinationMembershipFunction 3468
B1 3470
B2 3470
Sigma1 3471
Sigma2 3471
GetValue 3472
CNormalMembershipFunction 3473
B 3475
Sigma 3475
GetValue 3476
CP_ShapedMembershipFunction 3477
A 3479
B 3479
C 3480
D 3480
GetValue 3481
CProductTwoSigmoidalMembershipFunctions 3482
A1 3484
A2 3484
C1 3485
C2 3485
GetValue 3486
CS_ShapedMembershipFunction 3487
A 3489
B 3489
GetValue 3490
CSigmoidalMembershipFunction 3491
A 3493
C 3493
GetValue 3494
CTrapezoidMembershipFunction 3495
X1 3497
X2 3497
X3 3498
X4 3498
GetValue 3499
CTriangularMembershipFunction 3500

© 2000–2025, MetaQuotes Ltd.

X1 3502
X2 3502
X3 3503
ToNormalMF 3503
GetValue 3503
CZ_ShapedMembershipFunction 3505
A 3507
B 3507
GetValue 3508
IMembershipFunction 3509
GetValue 3509
ファジーシステムルール 3510
CMamdaniFuzzyRule 3511
Conclusion 3511
Weight 3512
CSugenoFuzzyRule 3513
Conclusion 3513
CSingleCondition 3515
Not 3515
Term 3516
Var 3516
CConditions 3518
ConditionsList 3518
Not 3519
Op 3519
CGenericFuzzyRule 3520
Conclusion 3520
Condition 3521
CreateCondition 3521
ファジーシステム変数 3523
CFuzzyVariable 3524
AddTerm 3525
GetTermByName 3525
Max 3525
Min 3526
Terms 3526
Values 3526
CSugenoVariable 3528
Functions 3528
GetFuncByName 3529
Values 3529
ファジー用語 3530
MembershipFunction 3531
ファジーシステム 3532
Mamdani system 3533
AggregationMethod 3533
Calculate 3534
DefuzzificationMethod 3534
EmptyRule 3534
ImplicationMethod 3534
Output 3535
OutputByName 3535
ParseRule 3535
Rules 3535
Sugeno system 3537
Calculate 3537
CreateSugenoFunction 3538
EmptyRule 3539

© 2000–2025, MetaQuotes Ltd.

Output ... 3539
OutputByName ... 3539
ParseRule ... 3539
Rules ... 3540
OpenCL ... 3541
BufferCreate ... 3543
BufferFree ... 3544
BufferFromArray ... 3545
BufferRead ... 3546
BufferWrite ... 3547
Execute ... 3548
GetContext ... 3549
GetKernel ... 3550
GetKernelName ... 3551
GetProgram ... 3552
Initialize ... 3553
KernelCreate ... 3554
KernelFree ... 3555
SetArgument ... 3556
SetArgumentBuffer ... 3557
SetArgumentLocalMemory ... 3558
SetBuffersCount ... 3559
SetKernelsCount ... 3560
Shutdown ... 3561
SupportDouble ... 3562
基本クラスCObject ... 3563
Prev ... 3564
Prev ... 3565
Next ... 3566
Next ... 3567
Compare ... 3568
Save ... 3570
Load ... 3572
Type ... 3574
データ収集 ... 3575
CArray ... 3576
Step ... 3578
Step ... 3579
Total ... 3580
Available ... 3581
Max ... 3582
IsSorted ... 3583
SortMode ... 3584
Clear ... 3585
Sort ... 3586
Save ... 3587
Load ... 3588
CArrayChar ... 3589
Reserve ... 3591
Resize ... 3592
Shutdown ... 3593
Add ... 3594
AddArray ... 3595
AddArray ... 3596
Insert ... 3598
InsertArray ... 3599
InsertArray ... 3600
AssignArray ... 3602

© 2000–2025, MetaQuotes Ltd.

AssignArray 3603
Update 3605
Shift 3606
Delete 3607
DeleteRange 3608
At 3609
CompareArray 3611
CompareArray 3612
InsertSort 3613
Search 3614
SearchGreat 3615
SearchLess 3616
SearchGreatOrEqual 3617
SearchLessOrEqual 3618
SearchFirst 3619
SearchLast 3620
SearchLinear 3621
Save 3622
Load 3623
Type 3625
CArrayShort 3626
Reserve 3628
Resize 3629
Shutdown 3630
Add 3631
AddArray 3632
AddArray 3633
Insert 3635
InsertArray 3636
InsertArray 3637
AssignArray 3639
AssignArray 3640
Update 3642
Shift 3643
Delete 3644
DeleteRange 3645
At 3646
CompareArray 3648
CompareArray 3649
InsertSort 3650
Search 3651
SearchGreat 3652
SearchLess 3653
SearchGreatOrEqual 3654
SearchLessOrEqual 3655
SearchFirst 3656
SearchLast 3657
SearchLinear 3658
Save 3659
Load 3661
Type 3663
CArrayInt 3664
Reserve 3666
Resize 3667
Shutdown 3668
Add 3669
AddArray 3670
AddArray 3671

© 2000–2025, MetaQuotes Ltd.

Insert 3673
InsertArray 3674
InsertArray 3675
AssignArray 3677
AssignArray 3678
Update 3680
Shift 3681
Delete 3682
DeleteRange 3683
At 3684
CompareArray 3686
CompareArray 3687
InsertSort 3688
Search 3689
SearchGreat 3690
SearchLess 3691
SearchGreatOrEqual 3692
SearchLessOrEqual 3693
SearchFirst 3694
SearchLast 3695
SearchLinear 3696
Save 3697
Load 3699
Type 3701
CArrayLong 3702
Reserve 3705
Resize 3706
Shutdown 3707
Add 3708
AddArray 3709
AddArray 3710
Insert 3712
InsertArray 3713
InsertArray 3714
AssignArray 3716
AssignArray 3717
Update 3719
Shift 3720
Delete 3721
DeleteRange 3722
At 3723
CompareArray 3725
CompareArray 3726
InsertSort 3727
Search 3728
SearchGreat 3729
SearchLess 3730
SearchGreatOrEqual 3731
SearchLessOrEqual 3732
SearchFirst 3733
SearchLast 3734
SearchLinear 3735
Save 3736
Load 3738
Type 3740
CArrayFloat 3741
Delta 3743
Reserve 3744

© 2000–2025, MetaQuotes Ltd.

Resize 3745
Shutdown 3746
Add 3747
AddArray 3748
AddArray 3749
Insert 3751
InsertArray 3752
InsertArray 3753
AssignArray 3755
AssignArray 3756
Update 3758
Shift 3759
Delete 3760
DeleteRange 3761
At 3762
CompareArray 3764
CompareArray 3765
InsertSort 3766
Search 3767
SearchGreat 3768
SearchLess 3769
SearchGreatOrEqual 3770
SearchLessOrEqual 3771
SearchFirst 3772
SearchLast 3773
SearchLinear 3774
Save 3775
Load 3777
Type 3779
CArrayDouble 3780
Delta 3783
Reserve 3784
Resize 3785
Shutdown 3786
Add 3787
AddArray 3788
AddArray 3789
Insert 3791
InsertArray 3792
InsertArray 3793
AssignArray 3795
AssignArray 3796
Update 3798
Shift 3799
Delete 3800
DeleteRange 3801
At 3802
CompareArray 3804
CompareArray 3805
Minimum 3806
Maximum 3807
InsertSort 3808
Search 3809
SearchGreat 3810
SearchLess 3811
SearchGreatOrEqual 3812
SearchLessOrEqual 3813
SearchFirst 3814

© 2000–2025, MetaQuotes Ltd.

SearchLast 3815
SearchLinear 3816
Save 3817
Load 3819
Type 3821
CArrayString 3822
Reserve 3824
Resize 3825
Shutdown 3826
Add 3827
AddArray 3828
AddArray 3829
Insert 3831
InsertArray 3832
InsertArray 3833
AssignArray 3835
AssignArray 3836
Update 3838
Shift 3839
Delete 3840
DeleteRange 3841
At 3842
CompareArray 3844
CompareArray 3845
InsertSort 3846
Search 3847
SearchGreat 3848
SearchLess 3849
SearchGreatOrEqual 3850
SearchLessOrEqual 3851
SearchFirst 3852
SearchLast 3853
SearchLinear 3854
Save 3855
Load 3857
Type 3859
CArrayObj 3860
FreeMode 3865
FreeMode 3866
Reserve 3868
Resize 3869
Clear 3870
Shutdown 3871
CreateElement 3872
Add 3874
AddArray 3875
Insert 3878
InsertArray 3880
AssignArray 3882
Update 3884
Shift 3885
Detach 3886
Delete 3887
DeleteRange 3888
At 3889
CompareArray 3890
InsertSort 3891
Search 3892

© 2000–2025, MetaQuotes Ltd.

SearchGreat 3893
SearchLess 3894
SearchGreatOrEqual 3895
SearchLessOrEqual 3896
SearchFirst 3897
SearchLast 3898
Save 3899
Load 3900
Type 3902
CList 3903
FreeMode 3905
FreeMode 3906
Total 3908
IsSorted 3909
SortMode 3910
CreateElement 3911
Add 3912
Insert 3913
DetachCurrent 3915
DeleteCurrent 3916
Delete 3917
Clear 3918
IndexOf 3919
GetNodeAtIndex 3920
GetFirstNode 3921
GetPrevNode 3922
GetCurrentNode 3923
GetNextNode 3924
GetLastNode 3925
Sort 3926
MoveToIndex 3927
Exchange 3928
CompareList 3929
Search 3930
Save 3931
Load 3933
Type 3935
CTreeNode 3936
Owner 3941
Left 3942
Right 3943
Balance 3944
BalanceL 3945
BalanceR 3946
CreateSample 3947
RefreshBalance 3948
GetNext 3949
SaveNode 3950
LoadNode 3951
Type 3952
CTree 3953
Root 3959
CreateElement 3960
Insert 3961
Detach 3962
Delete 3963
Clear 3964
Find 3965

© 2000–2025, MetaQuotes Ltd.

Save 3966
Load 3967
Type 3968
ジェネリックデータコレクション 3969
ICollection<T> 3971
Add 3972
Count 3973
Contains 3974
CopyTo 3975
Clear 3976
Remove 3977
IEqualityComparable<T> 3978
Equals 3979
HashCode 3980
IComparable<T> 3981
Compare 3982
IComparer<T> 3983
Compare 3984
IEqualityComparer<T> 3985
Equals 3986
HashCode 3987
IList<T> 3988
TryGetValue 3989
TrySetValue 3990
Insert 3991
IndexOf 3992
LastIndexOf 3993
RemoveAt 3994
IMap<TKey,TValue> 3995
Add 3996
Contains 3997
Remove 3998
TryGetValue 3999
TrySetValue 4000
CopyTo 4001
ISet<T> 4002
ExceptWith 4004
IntersectWith 4005
SymmetricExceptWith 4006
UnionWith 4007
IsProperSubsetOf 4008
IsProperSupersetOf 4009
IsSubsetOf 4010
IsSupersetOf 4011
Overlaps 4012
SetEquals 4013
CDefaultComparer<T> 4014
Compare 4015
CDefaultEqualityComparer<T> 4016
Equals 4017
HashCode 4018
CRedBlackTreeNode<T> 4019
Value 4020
Parent 4021
Left 4022
Right 4023
Color 4024
IsLeaf 4025

© 2000–2025, MetaQuotes Ltd.

CreateEmptyNode 4026
CLinkedListNode<T> 4027
List 4028
Next 4029
Previous 4030
Value 4031
CKeyValuePair<TKey,TValue> 4032
Key 4033
Value 4034
Clone 4035
Compare 4036
Equals 4037
HashCode 4038
CArrayList<T> 4039
Capacity 4041
Count 4042
Contains 4043
TrimExcess 4044
TryGetValue 4045
TrySetValue 4046
Add 4047
AddRange 4048
Insert 4049
InsertRange 4050
CopyTo 4051
BinarySearch 4052
IndexOf 4053
LastIndexOf 4054
Clear 4055
Remove 4056
RemoveAt 4057
RemoveRange 4058
Reverse 4059
Sort 4060
CHashMap<TKey,TValue> 4061
Add 4063
Count 4064
Comparer 4065
Contains 4066
ContainsKey 4067
ContainsValue 4068
CopyTo 4069
Clear 4070
Remove 4071
TryGetValue 4072
TrySetValue 4073
CHashSet<T> 4074
Add 4076
Count 4077
Contains 4078
Comparer 4079
TrimExcess 4080
CopyTo 4081
Clear 4082
Remove 4083
ExceptWith 4084
IntersectWith 4085
SymmetricExceptWith 4086

© 2000–2025, MetaQuotes Ltd.

UnionWith 4087
IsProperSubsetOf 4088
IsProperSupersetOf 4089
IsSubsetOf 4090
IsSupersetOf 4091
Overlaps 4092
SetEquals 4093
CLinkedList<T> 4094
Add 4096
AddAfter 4097
AddBefore 4098
AddFirst 4099
AddLast 4100
Count 4101
Head 4102
First 4103
Last 4104
Contains 4105
CopyTo 4106
Clear 4107
Remove 4108
RemoveFirst 4109
RemoveLast 4110
Find 4111
FindLast 4112
CQueue<T> 4113
Add 4114
Enqueue 4115
Count 4116
Contains 4117
TrimExcess 4118
CopyTo 4119
Clear 4120
Remove 4121
Dequeue 4122
Peek 4123
CRedBlackTree<T> 4124
Add 4126
Count 4127
Root 4128
Contains 4129
Comparer 4130
TryGetMin 4131
TryGetMax 4132
CopyTo 4133
Clear 4134
Remove 4135
RemoveMin 4136
RemoveMax 4137
Find 4138
FindMin 4139
FindMax 4140
CSortedMap<TKey, TValue> 4141
Add 4143
Count 4144
Comparer 4145
Contains 4146
ContainsKey 4147

© 2000–2025, MetaQuotes Ltd.

ContainsValue 4148
CopyTo 4149
Clear 4150
Remove 4151
TryGetValue 4152
TrySetValue 4153
CSortedSet<T> 4154
Add 4156
Count 4157
Contains 4158
Comparer 4159
TryGetMin 4160
TryGetMax 4161
CopyTo 4162
Clear 4163
Remove 4164
ExceptWith 4165
IntersectWith 4166
SymmetricExceptWith 4167
UnionWith 4168
IsProperSubsetOf 4169
IsProperSupersetOf 4170
IsSubsetOf 4171
IsSupersetOf 4172
Overlaps 4173
SetEquals 4174
GetViewBetween 4175
GetReverse 4176
CStack<T> 4177
Add 4178
Count 4179
Contains 4180
TrimExcess 4181
CopyTo 4182
Clear 4183
Remove 4184
Push 4185
Peek 4186
Pop 4187
ArrayBinarySearch<T> 4188
ArrayIndexOf<T> 4189
ArrayLastIndexOf<T> 4190
ArrayReverse<T> 4191
Compare 4192
Equals<T> 4195
GetHashCode 4196
ファイル 4199
CFile 4200
Handle 4202
Filename 4203
Flags 4204
SetUnicode 4205
SetCommon 4206
Open 4207
Close 4208
Delete 4209
IsExist 4210
Copy 4211

© 2000–2025, MetaQuotes Ltd.

Move 4212
Size 4213
Tell 4214
Seek 4215
Flush 4216
IsEnding 4217
IsLineEnding 4218
FolderCreate 4219
FolderDelete 4220
FolderClean 4221
FileFindFirst 4222
FileFindNext 4223
FileFindClose 4224
CFileBin 4225
Open 4227
WriteChar 4228
WriteShort 4229
WriteInteger 4230
WriteLong 4231
WriteFloat 4232
WriteDouble 4233
WriteString 4234
WriteCharArray 4235
WriteShortArray 4236
WriteIntegerArray 4237
WriteLongArray 4238
WriteFloatArray 4239
WriteDoubleArray 4240
WriteObject 4241
ReadChar 4242
ReadShort 4243
ReadInteger 4244
ReadLong 4245
ReadFloat 4246
ReadDouble 4247
ReadString 4248
ReadCharArray 4249
ReadShortArray 4250
ReadIntegerArray 4251
ReadLongArray 4252
ReadFloatArray 4253
ReadDoubleArray 4254
ReadObject 4255
CFileTxt 4256
Open 4257
WriteString 4258
ReadString 4259
文字列 **4260**
CString 4261
Str 4263
Len 4264
Copy 4265
Fill 4266
Assign 4267
Append 4268
Insert 4269
Compare 4270
CompareNoCase 4271

© 2000–2025, MetaQuotes Ltd.

Left 4272
Right 4273
Mid 4274
Trim 4275
TrimLeft 4276
TrimRight 4277
Clear 4278
ToUpper 4279
ToLower 4280
Reverse 4281
Find 4282
FindRev 4283
Remove 4284
Replace 4285
グラフィックオブジェクト 4286
CChartObject 4287
ChartId 4290
Window 4291
Name 4292
NumPoints 4293
Attach 4294
SetPoint 4295
Delete 4296
Detach 4297
ShiftObject 4298
ShiftPoint 4299
Time 4300
Price 4302
Color 4304
Style 4305
Width 4306
Background 4307
Selected 4308
Selectable 4309
Description 4310
Tooltip 4311
Timeframes 4312
Z_Order 4313
CreateTime 4314
LevelsCount 4315
LevelColor 4316
LevelStyle 4318
LevelWidth 4320
LevelValue 4322
LevelDescription 4324
GetInteger 4326
SetInteger 4328
GetDouble 4330
SetDouble 4332
GetString 4334
SetString 4336
Save 4338
Load 4339
Type 4340
ラインオブジェクト 4341
CChartObjectVLine 4342
Create 4343
Type 4344

© 2000–2025, MetaQuotes Ltd.

CChartObjectHLine 4345
Create 4346
Type 4347
CChartObjectTrend 4348
Create 4350
RayLeft 4351
RayRight 4352
Save 4353
Load 4354
Type 4355
CChartObjectTrendByAngle 4356
Create 4358
Angle 4359
Type 4360
CChartObjectCycles 4361
Create 4362
Type 4363
チャンネルオブジェクト 4364
CChartObjectChannel 4365
Create 4367
Type 4368
CChartObjectRegression 4369
Create 4371
Type 4372
CChartObjectStdDevChannel 4373
Create 4375
Deviations 4376
Save 4377
Load 4378
Type 4379
CChartObjectPitchfork 4380
Create 4382
Type 4383
ギャンツール 4384
CChartObjectGannLine 4385
Create 4387
PipsPerBar 4388
Save 4389
Load 4390
Type 4391
CChartObjectGannFan 4392
Create 4394
PipsPerBar 4395
Downtrend 4396
Save 4397
Load 4398
Type 4399
CChartObjectGannGrid 4400
Create 4402
PipsPerBar 4403
Downtrend 4404
Save 4405
Load 4406
Type 4407
フィボナッチツール 4408
CChartObjectFibo 4409
Create 4411
Type 4412

© 2000–2025, MetaQuotes Ltd.

CChartObjectFiboTimes 4413
Create 4414
Type 4415
CChartObjectFiboFan 4416
Create 4417
Type 4418
CChartObjectFiboArc 4419
Create 4421
Scale 4422
Ellipse 4423
Save 4424
Load 4425
Type 4426
CChartObjectFiboChannel 4427
Create 4429
Type 4430
CChartObjectFiboExpansion 4431
Create 4433
Type 4434
エリオットツール 4435
CChartObjectElliottWave3 4436
Create 4438
Degree 4439
Lines 4440
Save 4441
Load 4442
Type 4443
CChartObjectElliottWave5 4444
Create 4446
Type 4448
図形オブジェクト 4449
CChartObjectRectangle 4450
Create 4451
Type 4452
CChartObjectTriangle 4453
Create 4454
Type 4455
CChartObjectEllipse 4456
Create 4457
Type 4458
矢印オブジェクト 4459
CChartObjectArrow 4460
Create 4462
ArrowCode 4464
Anchor 4466
Save 4468
Load 4469
Type 4470
固定コードを持つ矢印 4471
Create 4473
ArrowCode 4475
Type 4476
オブジェクト制御 4477
CChartObjectText 4478
Create 4480
Angle 4481
Font 4482
FontSize 4483

© 2000–2025, MetaQuotes Ltd.

Anchor 4484
Save 4485
Load 4486
Type 4487
CChartObjectLabel 4488
Create 4490
X_Distance 4491
Y_Distance 4492
X_Size 4493
Y_Size 4494
Corner 4495
Time 4496
Price 4497
Save 4498
Load 4499
Type 4500
CChartObjectEdit 4501
Create 4503
TextAlign 4504
X_Size 4505
Y_Size 4506
BackColor 4507
BorderColor 4508
ReadOnly 4509
Angle 4510
Save 4511
Load 4512
Type 4513
CChartObjectButton 4514
State 4516
Save 4517
Load 4518
Type 4519
CChartObjectSubChart 4520
Create 4522
X_Distance 4523
Y_Distance 4524
Corner 4525
X_Size 4526
Y_Size 4527
Symbol 4528
Period 4529
Scale 4530
DateScale 4531
PriceScale 4532
Time 4533
Price 4534
Save 4535
Load 4536
Type 4537
CChartObjectBitmap 4538
Create 4540
BmpFile 4541
X_Offset 4542
Y_Offset 4543
Save 4544
Load 4545
Type 4546

© 2000–2025, MetaQuotes Ltd.

CChartObjectBmpLabel 4547
Create 4549
X_Distance 4550
Y_Distance 4551
X_Offset 4552
Y_Offset 4553
Corner 4554
X_Size 4555
Y_Size 4556
BmpFileOn 4557
BmpFileOff 4558
State 4559
Time 4560
Price 4561
Save 4562
Load 4563
Type 4564
CChartObjectRectLabel 4565
Create 4567
X_Size 4568
Y_Size 4569
BackColor 4570
Angle 4571
BorderType 4572
Save 4573
Load 4574
Type 4575
カスタムグラフィックス 4576
CCanvas 4577
Attach 4581
Arc 4582
Pie 4585
FillPolygon 4588
FillEllipse 4589
GetDefaultColor 4590
ChartObjectName 4591
Circle 4592
CircleAA 4593
CircleWu 4594
Create 4595
CreateBitmap 4596
CreateBitmapLabel 4598
Destroy 4600
Ellipse 4601
EllipseAA 4602
EllipseWu 4603
Erase 4604
Fill 4605
FillCircle 4606
FillRectangle 4607
FillTriangle 4608
FontAngleGet 4609
FontAngleSet 4610
FontFlagsGet 4611
FontFlagsSet 4612
FontGet 4613
FontNameGet 4614
FontNameSet 4615

© 2000–2025, MetaQuotes Ltd.

FontSet 4616
FontSizeGet 4617
FontSizeSet 4618
Height 4619
Line 4620
LineAA 4621
LineWu 4622
LineHorizontal 4623
LineVertical 4624
LineStyleSet 4625
LineThick 4626
LineThickVertical 4627
LineThickHorizontal 4628
LoadFromFile 4629
PixelGet 4630
PixelSet 4631
PixelSetAA 4632
Polygon 4633
PolygonAA 4634
PolygonWu 4635
PolygonThick 4636
PolygonSmooth 4637
Polyline 4638
PolylineSmooth 4639
PolylineThick 4640
PolylineWu 4641
PolylineAA 4642
Rectangle 4643
Resize 4644
ResourceName 4645
TextHeight 4646
TextOut 4647
TextSize 4648
TextWidth 4649
TransparentLevelSet 4650
Triangle 4651
TriangleAA 4652
TriangleWu 4653
Update 4654
Width 4655
CChartCanvas 4656
ColorBackground 4659
ColorBorder 4660
ColorText 4661
ColorGrid 4662
MaxData 4663
MaxDescrLen 4664
ShowFlags 4665
IsShowLegend 4666
IsShowScaleLeft 4667
IsShowScaleRight 4668
IsShowScaleTop 4669
IsShowScaleBottom 4670
IsShowGrid 4671
IsShowDescriptors 4672
IsShowPercent 4673
VScaleMin 4674
VScaleMax 4675

© 2000–2025, MetaQuotes Ltd.

NumGrid 4676
DataOffset 4677
DataTotal 4678
DrawDescriptors 4679
DrawData 4680
Create 4681
AllowedShowFlags 4682
ShowLegend 4683
ShowScaleLeft 4684
ShowScaleRight 4685
ShowScaleTop 4686
ShowScaleBottom 4687
ShowGrid 4688
ShowDescriptors 4689
ShowValue 4690
ShowPercent 4691
LegendAlignment 4692
Accumulative 4693
VScaleParams 4694
DescriptorUpdate 4695
ColorUpdate 4696
ValuesCheck 4697
Redraw 4698
DrawBackground 4699
DrawLegend 4700
DrawLegendVertical 4701
DrawLegendHorizontal 4702
CalcScales 4703
DrawScales 4704
DrawScaleLeft 4705
DrawScaleRight 4706
DrawScaleTop 4707
DrawScaleBottom 4708
DrawGrid 4709
DrawChart 4710
CHistogramChart 4711
Gradient 4716
BarGap 4717
BarMinSize 4718
BarBorder 4719
Create 4720
SeriesAdd 4721
SeriesInsert 4722
SeriesUpdate 4723
SeriesDelete 4724
ValueUpdate 4725
DrawData 4726
DrawBar 4727
GradientBrush 4728
CLineChart 4729
Filled 4733
Create 4734
SeriesAdd 4735
SeriesInsert 4736
SeriesUpdate 4737
SeriesDelete 4738
ValueUpdate 4739
DrawChart 4740

© 2000–2025, MetaQuotes Ltd.

DrawData 4741
CalcArea 4742
CPieChart 4743
Create 4747
SeriesSet 4748
ValueAdd 4749
ValueInsert 4750
ValueUpdate 4751
ValueDelete 4752
DrawChart 4753
DrawPie 4754
LabelMake 4755
3Dグラフィックス 4756
CCanvas3D 4757
AmbientColorGet 4759
AmbientColorSet 4760
Attach 4761
Create 4762
Destroy 4763
DXContext 4764
DXDispatcher 4765
InputScene 4766
LightColorGet 4767
LightColorSet 4768
LightDirectionGet 4769
LightDirectionSet 4770
ObjectAdd 4771
ProjectionMatrixGet 4772
ProjectionMatrixSet 4773
Render 4774
RenderBegin 4775
RenderEnd 4776
ViewMatrixGet 4777
ViewMatrixSet 4778
ViewPositionSet 4779
ViewRotationSet 4780
ViewTargetSet 4781
ViewUpDirectionSet 4782
価格チャート 4783
ChartID 4788
Mode 4789
Foreground 4790
Shift 4791
ShiftSize 4792
AutoScroll 4793
Scale 4794
ScaleFix 4795
ScaleFix_11 4796
FixedMax 4797
FixedMin 4798
PointsPerBar 4799
ScalePPB 4800
ShowOHLC 4801
ShowLineBid 4802
ShowLineAsk 4803
ShowLastLine 4804
ShowPeriodSep 4805
ShowGrid 4806

© 2000–2025, MetaQuotes Ltd.

ShowVolumes 4807
ShowObjectDescr 4808
ShowDateScale 4809
ShowPriceScale 4810
ColorBackground 4811
ColorForeground 4812
ColorGrid 4813
ColorBarUp 4814
ColorBarDown 4815
ColorCandleBull 4816
ColorCandleBear 4817
ColorChartLine 4818
ColorVolumes 4819
ColorLineBid 4820
ColorLineAsk 4821
ColorLineLast 4822
ColorStopLevels 4823
VisibleBars 4824
WindowsTotal 4825
WindowIsVisible 4826
WindowHandle 4827
FirstVisibleBar 4828
WidthInBars 4829
WidthInPixels 4830
HeightInPixels 4831
PriceMin 4832
PriceMax 4833
Attach 4834
FirstChart 4835
NextChart 4836
Open 4837
Detach 4838
Close 4839
BringToTop 4840
EventObjectCreate 4841
EventObjectDelete 4842
IndicatorAdd 4843
IndicatorDelete 4844
IndicatorsTotal 4845
IndicatorName 4846
Navigate 4847
Symbol 4848
Period 4849
Redraw 4850
GetInteger 4851
SetInteger 4852
GetDouble 4853
SetDouble 4854
GetString 4855
SetString 4856
SetSymbolPeriod 4857
ApplyTemplate 4858
ScreenShot 4859
WindowOnDropped 4860
PriceOnDropped 4861
TimeOnDropped 4862
XOnDropped 4863
YOnDropped 4864

© 2000–2025, MetaQuotes Ltd.

Save 4865
Load 4866
Type 4867
科学的チャート **4868**
GraphPlot 4869
CAxis 4873
AutoScale 4875
Min 4876
Max 4877
Step 4878
Name 4879
Color 4880
ValuesSize 4881
ValuesWidth 4882
ValuesFormat 4883
ValuesDateTimeMode 4884
ValuesFunctionFormat 4885
ValuesFunctionFormatCBData 4887
NameSize 4888
ZeroLever 4889
DefaultStep 4890
MaxLabels 4891
MinGrace 4892
MaxGrace 4893
SelectAxisScale 4894
CColorGenerator 4895
Next 4896
Reset 4897
CCurve 4898
Type 4900
Name 4901
Color 4902
XMax 4903
XMin 4904
YMax 4905
YMin 4906
Size 4907
PointsSize 4908
PointsFill 4909
PointsColor 4910
GetX 4911
GetY 4912
LinesStyle 4913
LinesIsSmooth 4914
LinesSmoothTension 4915
LinesSmoothStep 4916
LinesEndStyle 4917
LinesWidth 4918
HistogramWidth 4920
CustomPlotCBData 4921
CustomPlotFunction 4922
PointsType 4926
StepsDimension 4927
TrendLineCoefficients 4928
TrendLineColor 4929
TrendLineVisible 4930
Update 4932
Visible 4934

© 2000–2025, MetaQuotes Ltd.

CGraphic 4935
Create 4938
Destroy 4939
Update 4940
ChartObjectName 4941
ResourceName 4942
XAxis 4943
YAxis 4944
GapSize 4945
BackgroundColor 4946
BackgroundMain 4947
BackgroundMainSize 4948
BackgroundMainColor 4949
BackgroundSub 4950
BackgroundSubSize 4951
BackgroundSubColor 4952
GridLineColor 4953
GridBackgroundColor 4954
GridCircleRadius 4955
GridCircleColor 4956
GridHasCircle 4957
GridAxisLineColor 4958
HistoryNameWidth 4959
HistoryNameSize 4960
HistorySymbolSize 4961
TextAdd 4962
LineAdd 4963
CurveAdd 4964
CurvePlot 4967
CurvePlotAll 4968
CurveGetByIndex 4969
CurveGetByName 4970
CurvesTotal 4971
CurveRemoveByName 4972
CurveRemoveByIndex 4973
MarksToAxisAdd 4974
MajorMarkSize 4975
FontSet 4976
FontGet 4977
Attach 4978
CalculateMaxMinValues 4979
Height 4980
IndentDown 4981
IndentLeft 4982
IndentRight 4983
IndentUp 4984
Redraw 4985
ResetParameters 4986
ScaleX 4987
ScaleY 4988
SetDefaultParameters 4989
Width 4990
指標 4991
基本クラス 4992
CSpreadBuffer 4993
Size 4995
SetSymbolPeriod 4996
At 4997

© 2000–2025, MetaQuotes Ltd.

Refresh 4998
RefreshCurrent 4999
CTimeBuffer 5000
Size 5002
SetSymbolPeriod 5003
At 5004
Refresh 5005
RefreshCurrent 5006
CTickVolumeBuffer 5007
Size 5009
SetSymbolPeriod 5010
At 5011
Refresh 5012
RefreshCurrent 5013
CRealVolumeBuffer 5014
Size 5016
SetSymbolPeriod 5017
At 5018
Refresh 5019
RefreshCurrent 5020
CDoubleBuffer 5021
Size 5023
SetSymbolPeriod 5024
At 5025
Refresh 5026
RefreshCurrent 5027
COpenBuffer 5028
Refresh 5029
RefreshCurrent 5030
CHighBuffer 5031
Refresh 5032
RefreshCurrent 5033
CLowBuffer 5034
Refresh 5035
RefreshCurrent 5036
CCloseBuffer 5037
Refresh 5038
RefreshCurrent 5039
CIndicatorBuffer 5040
Offset 5042
Name 5043
At 5044
Refresh 5045
RefreshCurrent 5046
CSeries 5047
Name 5049
BuffersTotal 5050
Timeframe 5051
Symbol 5052
Period 5053
RefreshCurrent 5054
BufferSize 5055
BufferResize 5056
Refresh 5057
PeriodDescription 5058
CPriceSeries 5059
BufferResize 5061
GetData 5062

© 2000–2025, MetaQuotes Ltd.

Refresh 5063
MinIndex 5064
MinValue 5065
MaxIndex 5066
MaxValue 5067
CIndicator 5068
Handle 5071
Status 5072
FullRelease 5073
Create 5074
BufferResize 5075
BarsCalculated 5076
GetData 5077
Refresh 5080
Minimum 5081
MinValue 5082
Maximum 5083
MaxValue 5084
MethodDescription 5085
PriceDescription 5086
VolumeDescription 5087
AddToChart 5088
DeleteFromChart 5089
CIndicators 5090
Create 5091
Refresh 5092
時系列クラス 5093
CiSpread 5094
Create 5096
BufferResize 5097
GetData 5098
Refresh 5100
CiTime 5101
Create 5103
BufferResize 5104
GetData 5105
Refresh 5107
CiTickVolume 5108
Create 5110
BufferResize 5111
GetData 5112
Refresh 5114
CiRealVolume 5115
Create 5117
BufferResize 5118
GetData 5119
Refresh 5121
CiOpen 5122
Create 5124
GetData 5125
CiHigh 5127
Create 5129
GetData 5130
CiLow 5132
Create 5134
GetData 5135
CiClose 5137
Create 5139

© 2000–2025, MetaQuotes Ltd.

GetData 5140
トレンド指標 5142
CiADX 5143
MaPeriod 5145
Create 5146
Main 5147
Plus 5148
Minus 5149
Type 5150
CiADXWilder 5151
MaPeriod 5153
Create 5154
Main 5155
Plus 5156
Minus 5157
Type 5158
CiBands 5159
MaPeriod 5161
MaShift 5162
Deviation 5163
Applied 5164
Create 5165
Base 5166
Upper 5167
Lower 5168
Type 5169
CiEnvelopes 5170
MaPeriod 5172
MaShift 5173
MaMethod 5174
Deviation 5175
Applied 5176
Create 5177
Upper 5178
Lower 5179
Type 5180
CiIchimoku 5181
TenkanSenPeriod 5183
KijunSenPeriod 5184
SenkouSpanBPeriod 5185
Create 5186
TenkanSen 5187
KijunSen 5188
SenkouSpanA 5189
SenkouSpanB 5190
ChinkouSpan 5191
Type 5192
CiMA 5193
MaPeriod 5195
MaShift 5196
MaMethod 5197
Applied 5198
Create 5199
Main 5200
Type 5201
CiSAR 5202
SarStep 5204
Maximum 5205

© 2000–2025, MetaQuotes Ltd.

Create 5206
Main 5207
Type 5208
CiStdDev 5209
MaPeriod 5211
MaShift 5212
MaMethod 5213
Applied 5214
Create 5215
Main 5216
Type 5217
CiDEMA 5218
MaPeriod 5220
IndShift 5221
Applied 5222
Create 5223
Main 5224
Type 5225
CiTEMA 5226
MaPeriod 5228
IndShift 5229
Applied 5230
Create 5231
Main 5232
Type 5233
CiFrAMA 5234
MaPeriod 5236
IndShift 5237
Applied 5238
Create 5239
Main 5240
Type 5241
CiAMA 5242
MaPeriod 5244
FastEmaPeriod 5245
SlowEmaPeriod 5246
IndShift 5247
Applied 5248
Create 5249
Main 5250
Type 5251
CiVIDyA 5252
CmoPeriod 5254
EmaPeriod 5255
IndShift 5256
Applied 5257
Create 5258
Main 5259
Type 5260
オシレーター 5261
CiATR 5262
MaPeriod 5264
Create 5265
Main 5266
Type 5267
CiBearsPower 5268
MaPeriod 5270
Create 5271

© 2000–2025, MetaQuotes Ltd.

Main 5272
Type 5273
CiBullsPower 5274
MaPeriod 5276
Create 5277
Main 5278
Type 5279
CiCCI 5280
MaPeriod 5282
Applied 5283
Create 5284
Main 5285
Type 5286
CiChaikin 5287
FastMaPeriod 5289
SlowMaPeriod 5290
MaMethod 5291
Applied 5292
Create 5293
Main 5294
Type 5295
CiDeMarker 5296
MaPeriod 5298
Create 5299
Main 5300
Type 5301
CiForce 5302
MaPeriod 5304
MaMethod 5305
Applied 5306
Create 5307
Main 5308
Type 5309
CiMACD 5310
FastEmaPeriod 5312
SlowEmaPeriod 5313
SignalPeriod 5314
Applied 5315
Create 5316
Main 5317
Signal 5318
Type 5319
CiMomentum 5320
MaPeriod 5322
Applied 5323
Create 5324
Main 5325
Type 5326
CiOsMA 5327
FastEmaPeriod 5329
SlowEmaPeriod 5330
SignalPeriod 5331
Applied 5332
Create 5333
Main 5334
Type 5335
CiRSI 5336
MaPeriod 5338

© 2000–2025, MetaQuotes Ltd.

Applied 5339
Create 5340
Main 5341
Type 5342
CiRVI 5343
MaPeriod 5345
Create 5346
Main 5347
Signal 5348
Type 5349
CiStochastic 5350
Kperiod 5352
Dperiod 5353
Slowing 5354
MaMethod 5355
PriceField 5356
Create 5357
Main 5358
Signal 5359
Type 5360
CiTriX 5361
MaPeriod 5363
Applied 5364
Create 5365
Main 5366
Type 5367
CiWPR 5368
CalcPeriod 5370
Create 5371
Main 5372
Type 5373
ボリューム指標 5374
CiAD 5375
Applied 5377
Create 5378
Main 5379
Type 5380
CiMFI 5381
MaPeriod 5383
Applied 5384
Create 5385
Main 5386
Type 5387
CiOBV 5388
Applied 5390
Create 5391
Main 5392
Type 5393
CiVolumes 5394
Applied 5396
Create 5397
Main 5398
Type 5399
ビルウィリアムズ指標 5400
CiAC 5401
Create 5403
Main 5404
Type 5405

© 2000–2025, MetaQuotes Ltd.

CiAlligator 5406
JawPeriod 5408
JawShift 5409
TeethPeriod 5410
TeethShift 5411
LipsPeriod 5412
LipsShift 5413
MaMethod 5414
Applied 5415
Create 5416
Jaw 5417
Teeth 5418
Lips 5419
Type 5420
CiAO 5421
Create 5423
Main 5424
Type 5425
CiFractals 5426
Create 5428
Upper 5429
Lower 5430
Type 5431
CiGator 5432
JawPeriod 5434
JawShift 5435
TeethPeriod 5436
TeethShift 5437
LipsPeriod 5438
LipsShift 5439
MaMethod 5440
Applied 5441
Create 5442
Upper 5443
Lower 5444
Type 5445
CiBWMFI 5446
Applied 5448
Create 5449
Main 5450
Type 5451
カスタム指標 5452
NumBuffers 5453
NumParams 5454
ParamType 5455
ParamLong 5456
ParamDouble 5457
ParamString 5458
Type 5459
トレードクラス 5460
CAccountInfo 5461
Login 5463
TradeMode 5464
TradeModeDescription 5465
Leverage 5466
StopoutMode 5467
StopoutModeDescription 5468
MarginMode 5469

© 2000–2025, MetaQuotes Ltd.

MarginModeDescription 5470
TradeAllowed 5471
TradeExpert 5472
LimitOrders 5473
Balance 5474
Credit 5475
Profit 5476
Equity 5477
Margin 5478
FreeMargin 5479
MarginLevel 5480
MarginCall 5481
MarginStopOut 5482
Name 5483
Server 5484
Currency 5485
Company 5486
InfoInteger 5487
InfoDouble 5488
InfoString 5489
OrderProfitCheck 5490
MarginCheck 5491
FreeMarginCheck 5492
MaxLotCheck 5493
CSymbolInfo 5494
Refresh 5498
RefreshRates 5499
Name 5500
Select 5501
IsSynchronized 5502
Volume 5503
VolumeHigh 5504
VolumeLow 5505
Time 5506
Spread 5507
SpreadFloat 5508
TicksBookDepth 5509
StopsLevel 5510
FreezeLevel 5511
Bid 5512
BidHigh 5513
BidLow 5514
Ask 5515
AskHigh 5516
AskLow 5517
Last 5518
LastHigh 5519
LastLow 5520
TradeCalcMode 5521
TradeCalcModeDescription 5522
TradeMode 5523
TradeModeDescription 5524
TradeExecution 5525
TradeExecutionDescription 5526
SwapMode 5527
SwapModeDescription 5528
SwapRollover3days 5529
SwapRollover3daysDescription 5530

© 2000–2025, MetaQuotes Ltd.

MarginInitial 5531
MarginMaintenance 5532
MarginLong 5533
MarginShort 5534
MarginLimit 5535
MarginStop 5536
MarginStopLimit 5537
TradeTimeFlags 5538
TradeFillFlags 5539
Digits 5540
Point 5541
TickValue 5542
TickValueProfit 5543
TickValueLoss 5544
TickSize 5545
ContractSize 5546
LotsMin 5547
LotsMax 5548
LotsStep 5549
LotsLimit 5550
SwapLong 5551
SwapShort 5552
CurrencyBase 5553
CurrencyProfit 5554
CurrencyMargin 5555
Bank 5556
Description 5557
Path 5558
SessionDeals 5559
SessionBuyOrders 5560
SessionSellOrders 5561
SessionTurnover 5562
SessionInterest 5563
SessionBuyOrdersVolume 5564
SessionSellOrdersVolume 5565
SessionOpen 5566
SessionClose 5567
SessionAW 5568
SessionPriceSettlement 5569
SessionPriceLimitMin 5570
SessionPriceLimitMax 5571
InfoInteger 5572
InfoDouble 5573
InfoString 5574
NormalizePrice 5575
COrderInfo 5576
Ticket 5578
TimeSetup 5579
TimeSetupMsc 5580
OrderType 5581
TypeDescription 5582
State 5583
StateDescription 5584
TimeExpiration 5585
TimeDone 5586
TimeDoneMsc 5587
TypeFilling 5588
TypeFillingDescription 5589

© 2000–2025, MetaQuotes Ltd.

TypeTime 5590
TypeTimeDescription 5591
Magic 5592
PositionId 5593
VolumeInitial 5594
VolumeCurrent 5595
PriceOpen 5596
StopLoss 5597
TakeProfit 5598
PriceCurrent 5599
PriceStopLimit 5600
Symbol 5601
Comment 5602
InfoInteger 5603
InfoDouble 5604
InfoString 5605
StoreState 5606
CheckState 5607
Select 5608
SelectByIndex 5609
CHistoryOrderInfo 5610
TimeSetup 5612
TimeSetupMsc 5613
OrderType 5614
TypeDescription 5615
State 5616
StateDescription 5617
TimeExpiration 5618
TimeDone 5619
TimeDoneMsc 5620
TypeFilling 5621
TypeFillingDescription 5622
TypeTime 5623
TypeTimeDescription 5624
Magic 5625
PositionId 5626
VolumeInitial 5627
VolumeCurrent 5628
PriceOpen 5629
StopLoss 5630
TakeProfit 5631
PriceCurrent 5632
PriceStopLimit 5633
Symbol 5634
Comment 5635
InfoInteger 5636
InfoDouble 5637
InfoString 5638
Ticket 5639
SelectByIndex 5640
CPositionInfo 5641
Time 5643
TimeMsc 5644
TimeUpdate 5645
TimeUpdateMsc 5646
PositionType 5647
TypeDescription 5648
Magic 5649

© 2000–2025, MetaQuotes Ltd.

Identifier 5650
Volume 5651
PriceOpen 5652
StopLoss 5653
TakeProfit 5654
PriceCurrent 5655
Commission 5656
Swap 5657
Profit 5658
Symbol 5659
Comment 5660
InfoInteger 5661
InfoDouble 5662
InfoString 5663
Select 5664
SelectByIndex 5665
SelectByMagic 5666
SelectByTicket 5667
StoreState 5668
CheckState 5669
CDealInfo 5670
Order 5672
Time 5673
TimeMsc 5674
DealType 5675
TypeDescription 5676
Entry 5677
EntryDescription 5678
Magic 5679
PositionId 5680
Volume 5681
Price 5682
Commision 5683
Swap 5684
Profit 5685
Symbol 5686
Comment 5687
InfoInteger 5688
InfoDouble 5689
InfoString 5690
Ticket 5691
SelectByIndex 5692
CTrade 5693
LogLevel 5697
SetExpertMagicNumber 5698
SetDeviationInPoints 5699
SetTypeFilling 5700
SetTypeFillingBySymbol 5701
SetAsyncMode 5702
SetMarginMode 5703
OrderOpen 5704
OrderModify 5706
OrderDelete 5707
PositionOpen 5708
PositionModify 5709
PositionClose 5710
PositionClosePartial 5711
PositionCloseBy 5713

© 2000–2025, MetaQuotes Ltd.

Buy 5714
Sell 5715
BuyLimit 5716
BuyStop 5717
SellLimit 5718
SellStop 5719
Request 5720
RequestAction 5721
RequestActionDescription 5722
RequestMagic 5723
RequestOrder 5724
RequestSymbol 5725
RequestVolume 5726
RequestPrice 5727
RequestStopLimit 5728
RequestSL 5729
RequestTP 5730
RequestDeviation 5731
RequestType 5732
RequestTypeDescription 5733
RequestTypeFilling 5734
RequestTypeFillingDescription 5735
RequestTypeTime 5736
RequestTypeTimeDescription 5737
RequestExpiration 5738
RequestComment 5739
RequestPosition 5740
RequestPositionBy 5741
Result 5742
ResultRetcode 5743
ResultRetcodeDescription 5744
ResultDeal 5745
ResultOrder 5746
ResultVolume 5747
ResultPrice 5748
ResultBid 5749
ResultAsk 5750
ResultComment 5751
CheckResult 5752
CheckResultRetcode 5753
CheckResultRetcodeDescription 5754
CheckResultBalance 5755
CheckResultEquity 5756
CheckResultProfit 5757
CheckResultMargin 5758
CheckResultMarginFree 5759
CheckResultMarginLevel 5760
CheckResultComment 5761
PrintRequest 5762
PrintResult 5763
FormatRequest 5764
FormatRequestResult 5765
CTerminalInfo 5766
Build 5768
IsConnected 5769
IsDLLsAllowed 5770
IsTradeAllowed 5771
IsEmailEnabled 5772

© 2000–2025, MetaQuotes Ltd.

IsFtpEnabled 5773
MaxBars 5774
CodePage 5775
CPUCores 5776
MemoryPhysical 5777
MemoryTotal 5778
MemoryAvailable 5779
MemoryUsed 5780
IsX64 5781
OpenCLSupport 5782
DiskSpace 5783
Language 5784
Name 5785
Company 5786
Path 5787
DataPath 5788
CommonDataPath 5789
InfoInteger 5790
InfoString 5791
ストラテジーモジュール 5792
エキスパートアドバイザーの基本クラス 5794
CExpertBase 5795
InitPhase 5797
TrendType 5798
UsedSeries 5799
EveryTick 5800
Open 5801
High 5802
Low 5803
Close 5804
Spread 5805
Time 5806
TickVolume 5807
RealVolume 5808
Init 5809
Symbol 5810
Period 5811
Magic 5812
ValidationSettings 5813
SetPriceSeries 5814
SetOtherSeries 5815
InitIndicators 5816
InitOpen 5817
InitHigh 5818
InitLow 5819
InitClose 5820
InitSpread 5821
InitTime 5822
InitTickVolume 5823
InitRealVolume 5824
PriceLevelUnit 5825
StartIndex 5826
CompareMagic 5827
CExpert 5828
Init 5832
Magic 5833
InitSignal 5834
InitTrailing 5835

© 2000–2025, MetaQuotes Ltd.

InitMoney 5836
InitTrade 5837
Deinit 5838
OnTickProcess 5839
OnTradeProcess 5840
OnTimerProcess 5841
OnChartEventProcess 5842
OnBookEventProcess 5843
MaxOrders 5844
Signal 5845
ValidationSettings 5846
InitIndicators 5847
OnTick 5848
OnTrade 5849
OnTimer 5850
OnChartEvent 5851
OnBookEvent 5852
InitParameters 5853
DeinitTrade 5854
DeinitSignal 5855
DeinitTrailing 5856
DeinitMoney 5857
DeinitIndicators 5858
Refresh 5859
Processing 5860
SelectPosition 5862
CheckOpen 5863
CheckOpenLong 5864
CheckOpenShort 5865
OpenLong 5866
OpenShort 5867
CheckReverse 5868
CheckReverseLong 5869
CheckReverseShort 5870
ReverseLong 5871
ReverseShort 5872
CheckClose 5873
CheckCloseLong 5874
CheckCloseShort 5875
CloseAll 5876
Close 5877
CloseLong 5878
CloseShort 5879
CheckTrailingStop 5880
CheckTrailingStopLong 5881
CheckTrailingStopShort 5882
TrailingStopLong 5883
TrailingStopShort 5884
CheckTrailingOrderLong 5885
CheckTrailingOrderShort 5886
TrailingOrderLong 5887
TrailingOrderShort 5888
CheckDeleteOrderLong 5889
CheckDeleteOrderShort 5890
DeleteOrders 5891
DeleteOrder 5892
DeleteOrderLong 5893
DeleteOrderShort 5894

© 2000–2025, MetaQuotes Ltd.

LotOpenLong 5895
LotOpenShort 5896
LotReverse 5897
PrepareHistoryDate 5898
HistoryPoint 5899
CheckTradeState 5900
WaitEvent 5901
NoWaitEvent 5902
TradeEventPositionStopTake 5903
TradeEventOrderTriggered 5904
TradeEventPositionOpened 5905
TradeEventPositionVolumeChanged 5906
TradeEventPositionModified 5907
TradeEventPositionClosed 5908
TradeEventOrderPlaced 5909
TradeEventOrderModified 5910
TradeEventOrderDeleted 5911
TradeEventNotIdentified 5912
TimeframeAdd 5913
TimeframesFlags 5914
CExpertSignal 5915
BasePrice 5918
UsedSeries 5919
Weight 5920
PatternsUsage 5921
General 5922
Ignore 5923
Invert 5924
ThresholdOpen 5925
ThresholdClose 5926
PriceLevel 5927
StopLevel 5928
TakeLevel 5929
Expiration 5930
Magic 5931
ValidationSettings 5932
InitIndicators 5933
AddFilter 5934
CheckOpenLong 5935
CheckOpenShort 5936
OpenLongParams 5937
OpenShortParams 5938
CheckCloseLong 5939
CheckCloseShort 5940
CloseLongParams 5941
CloseShortParams 5942
CheckReverseLong 5943
CheckReverseShort 5944
CheckTrailingOrderLong 5945
CheckTrailingOrderShort 5946
LongCondition 5947
ShortCondition 5948
Direction 5949
CExpertTrailing 5950
CheckTrailingStopLong 5951
CheckTrailingStopShort 5952
CExpertMoney 5953
Percent 5954

© 2000–2025, MetaQuotes Ltd.

ValidationSettings 5955
CheckOpenLong 5956
CheckOpenShort 5957
CheckReverse 5958
CheckClose 5959
売買シグナルのモジュール 5960
AC オシレーター(Accelerator Oscillator)指標のシグナル 5963
適応型移動平均(Adaptive Moving Average)指標のシグナル 5965
オーサムオシレーター(Awesome Oscillator)指標のシグナル 5969
ベアパワー(Bears Power)オシレーター指標のシグナル 5973
ブルパワー(Bulls Power)オシレーター指標のシグナル 5975
CCI(Commodity Channel Index、コモディティチャンネルインデックス)指標のシグナル 5977
デマーカー(DeMarker)オシレーター指標のシグナル 5981
2 重指数移動平均(Double Exponential Moving Average)指標のシグナル 5985
エンベロープ(Envelopes)指標のシグナル 5989
フラクタル適応型移動平均(Fractal Adaptive Moving Average)指標のシグナル 5992
日中時間フィルタのシグナル 5996
MACD オシレーターのシグナル 5997
移動平均(Moving Average)指標のシグナル 6003
パラボリック SAR(Parabolic SAR) 指標のシグナル 6007
相対力指数(Relative Strength Index)オシレーターのシグナル 6009
相対力指数(Relative Vigor Index)オシレーターのシグナル 6015
ストキャスティックス(Stochastic)のシグナル 6017
3 重指数平均(Triple Exponential Average)オシレーターのシグナル 6022
3 重指数移動平均(Triple Exponential Moving Average)指標のシグナル 6026
ウィリアムパーセントレンジ(Williams Percent Range)オシレーターのシグナル 6030
トレール注文のクラス 6033
CTrailingFixedPips 6034
StopLevel 6036
ProfitLevel 6037
ValidationSettings 6038
CheckTrailingStopLong 6039
CheckTrailingStopShort 6040
CTrailingMA 6041
Period 6042
Shift 6043
Method 6044
Applied 6045
InitIndicators 6046
ValidationSettings 6047
CheckTrailingStopLong 6048
CheckTrailingStopShort 6049
CTrailingNone 6050
CheckTrailingStopLong 6051
CheckTrailingStopShort 6052
CTrailingPSAR 6053
Step 6055
Maximum 6056
InitIndicators 6057
CheckTrailingStopLong 6058
CheckTrailingStopShort 6059
資金管理のクラス 6060
CMoneyFixedLot 6061
Lots 6063
ValidationSettings 6064
CheckOpenLong 6065
CheckOpenShort 6066

© 2000-2025, MetaQuotes Ltd.

CMoneyFixedMargin 6067
CheckOpenLong 6068
CheckOpenShort 6069
CMoneyFixedRisk 6070
CheckOpenLong 6071
CheckOpenShort 6072
CMoneyNone 6073
ValidationSettings 6074
CheckOpenLong 6075
CheckOpenShort 6076
CMoneySizeOptimized 6077
DecreaseFactor 6079
ValidationSettings 6080
CheckOpenLong 6081
CheckOpenShort 6082
パネルとダイアログ 6083
CRect 6085
Left 6086
Top 6087
Right 6088
Bottom 6089
Width 6090
Height 6091
SetBound 6092
Move 6093
Shift 6094
Contains 6095
Format 6096
CDateTime 6097
MonthName 6099
ShortMonthName 6100
DayName 6101
ShortDayName 6102
DaysInMonth 6103
DateTime 6104
Date 6105
Time 6106
Sec 6107
Min 6108
Hour 6109
Day 6110
Mon 6111
Year 6112
SecDec 6113
SecInc 6114
MinDec 6115
MinInc 6116
HourDec 6117
HourInc 6118
DayDec 6119
DayInc 6120
MonDec 6121
MonInc 6122
YearDec 6123
YearInc 6124
CWnd 6125
Create 6129
Destroy 6130

© 2000–2025, MetaQuotes Ltd.

OnEvent 6131
OnMouseEvent 6132
Name 6133
ControlsTotal 6134
Control 6135
ControlFind 6136
Rect 6137
Left 6138
Top 6139
Right 6140
Bottom 6141
Width 6142
Height 6143
Move 6144
Shift 6145
Resize 6146
Contains 6147
Alignment 6148
Align 6149
Id 6150
IsEnabled 6151
Enable 6152
Disable 6153
IsVisible 6154
Visible 6155
Show 6156
Hide 6157
IsActive 6158
Activate 6159
Deactivate 6160
StateFlags 6161
StateFlagsSet 6162
StateFlagsReset 6163
PropFlags 6164
PropFlagsSet 6165
PropFlagsReset 6166
MouseX 6167
MouseY 6168
MouseFlags 6169
MouseFocusKill 6170
OnCreate 6171
OnDestroy 6172
OnMove 6173
OnResize 6174
OnEnable 6175
OnDisable 6176
OnShow 6177
OnHide 6178
OnActivate 6179
OnDeactivate 6180
OnClick 6181
OnChange 6182
OnMouseDown 6183
OnMouseUp 6184
OnDragStart 6185
OnDragProcess 6186
OnDragEnd 6187
DragObjectCreate 6188

© 2000–2025, MetaQuotes Ltd.

DragObjectDestroy 6189
CWndObj 6190
OnEvent 6192
Text 6193
Color 6194
ColorBackground 6195
ColorBorder 6196
Font 6197
FontSize 6198
ZOrder 6199
OnObjectCreate 6200
OnObjectChange 6201
OnObjectDelete 6202
OnObjectDrag 6203
OnSetText 6204
OnSetColor 6205
OnSetColorBackground 6206
OnSetFont 6207
OnSetFontSize 6208
OnSetZOrder 6209
OnDestroy 6210
OnChange 6211
CWndContainer 6212
Destroy 6214
OnEvent 6215
OnMouseEvent 6216
ControlsTotal 6217
Control 6218
ControlFind 6219
Add 6220
Delete 6221
Move 6222
Shift 6223
Id 6224
Enable 6225
Disable 6226
Show 6227
Hide 6228
MouseFocusKill 6229
Save 6230
Load 6231
OnResize 6232
OnActivate 6233
OnDeactivate 6234
CLabel 6235
Create 6240
OnSetText 6241
OnSetColor 6242
OnSetFont 6243
OnSetFontSize 6244
OnCreate 6245
OnShow 6246
OnHide 6247
OnMove 6248
CBmpButton 6249
Create 6256
Border 6257
BmpNames 6258

© 2000–2025, MetaQuotes Ltd.

BmpOffName 6259
BmpOnName 6260
BmpPassiveName 6261
BmpActiveName 6262
Pressed 6263
Locking 6264
OnSetZOrder 6265
OnCreate 6266
OnShow 6267
OnHide 6268
OnMove 6269
OnChange 6270
OnActivate 6271
OnDeactivate 6272
OnMouseDown 6273
OnMouseUp 6274
CButton 6275
Create 6282
Pressed 6283
Locking 6284
OnSetText 6285
OnSetColor 6286
OnSetColorBackground 6287
OnSetColorBorder 6288
OnSetFont 6289
OnSetFontSize 6290
OnCreate 6291
OnShow 6292
OnHide 6293
OnMove 6294
OnResize 6295
OnMouseDown 6296
OnMouseUp 6297
CEdit 6298
Create 6303
ReadOnly 6304
TextAlign 6305
OnObjectEndEdit 6306
OnSetText 6307
OnSetColor 6308
OnSetColorBackground 6309
OnSetColorBorder 6310
OnSetFont 6311
OnSetFontSize 6312
OnSetZOrder 6313
OnCreate 6314
OnShow 6315
OnHide 6316
OnMove 6317
OnResize 6318
OnChange 6319
OnClick 6320
CPanel 6321
Create 6326
BorderType 6327
OnSetText 6328
OnSetColorBackground 6329
OnSetColorBorder 6330

© 2000–2025, MetaQuotes Ltd.

OnCreate 6331
OnShow 6332
OnHide 6333
OnMove 6334
OnResize 6335
OnChange 6336
CPicture 6337
Create 6342
Border 6343
BmpName 6344
OnCreate 6345
OnShow 6346
OnHide 6347
OnMove 6348
OnChange 6349
CScroll 6350
Create 6352
OnEvent 6353
MinPos 6354
MaxPos 6355
CurrPos 6356
CreateBack 6357
CreateInc 6358
CreateDec 6359
CreateThumb 6360
OnClickInc 6361
OnClickDec 6362
OnShow 6363
OnHide 6364
OnChangePos 6365
OnThumbDragStart 6366
OnThumbDragProcess 6367
OnThumbDragEnd 6368
CalcPos 6369
CScrollV 6370
CreateInc 6376
CreateDec 6377
CreateThumb 6378
OnResize 6379
OnChangePos 6380
OnThumbDragStart 6381
OnThumbDragProcess 6382
OnThumbDragEnd 6383
CalcPos 6384
CScrollH 6385
CreateInc 6391
CreateDec 6392
CreateThumb 6393
OnResize 6394
OnChangePos 6395
OnThumbDragStart 6396
OnThumbDragProcess 6397
OnThumbDragEnd 6398
CalcPos 6399
CWndClient 6400
Create 6402
OnEvent 6403
ColorBackground 6404

© 2000–2025, MetaQuotes Ltd.

ColorBorder 6405
BorderType 6406
VScrolled 6407
HScrolled 6408
CreateBack 6409
CreateScrollV 6410
CreateScrollH 6411
OnResize 6412
OnVScrollShow 6413
OnVScrollHide 6414
OnHScrollShow 6415
OnHScrollHide 6416
OnScrollLineDown 6417
OnScrollLineUp 6418
OnScrollLineLeft 6419
OnScrollLineRight 6420
Rebound 6421
CListView 6422
Create 6428
OnEvent 6429
TotalView 6430
AddItem 6431
Select 6432
SelectByText 6433
SelectByValue 6434
Value 6435
CreateRow 6436
OnResize 6437
OnVScrollShow 6438
OnVScrollHide 6439
OnScrollLineDown 6440
OnScrollLineUp 6441
OnItemClick 6442
Redraw 6443
RowState 6444
CheckView 6445
CComboBox 6446
Create 6452
OnEvent 6453
AddItem 6454
ListViewItems 6455
Select 6456
SelectByText 6457
SelectByValue 6458
Value 6459
CreateEdit 6460
CreateButton 6461
CreateList 6462
OnClickEdit 6463
OnClickButton 6464
OnChangeList 6465
ListShow 6466
ListHide 6467
CCheckBox 6468
Create 6475
OnEvent 6476
Text 6477
Color 6478

© 2000–2025, MetaQuotes Ltd.

Checked 6479
Value 6480
CreateButton 6481
CreateLabel 6482
OnClickButton 6483
OnClickLabel 6484
CCheckGroup 6485
Create 6491
OnEvent 6492
AddItem 6493
Value 6494
CreateButton 6495
OnVScrollShow 6496
OnVScrollHide 6497
OnScrollLineDown 6498
OnScrollLineUp 6499
OnChangeItem 6500
Redraw 6501
RowState 6502
CRadioButton 6503
Create 6505
OnEvent 6506
Text 6507
Color 6508
State 6509
CreateButton 6510
CreateLabel 6511
OnClickButton 6512
OnClickLabel 6513
CRadioGroup 6514
Create 6520
OnEvent 6521
AddItem 6522
Value 6523
CreateButton 6524
OnVScrollShow 6525
OnVScrollHide 6526
OnScrollLineDown 6527
OnScrollLineUp 6528
OnChangeItem 6529
Redraw 6530
RowState 6531
Select 6532
CSpinEdit 6533
Create 6539
OnEvent 6540
MinValue 6541
MaxValue 6542
Value 6543
CreateEdit 6544
CreateInc 6545
CreateDec 6546
OnClickInc 6547
OnClickDec 6548
OnChangeValue 6549
CDialog 6550
Create 6552
OnEvent 6553

© 2000–2025, MetaQuotes Ltd.

Caption 6554
Add 6555
CreateWhiteBorder 6556
CreateBackground 6557
CreateCaption 6558
CreateButtonClose 6559
CreateClientArea 6560
OnClickCaption 6561
OnClickButtonClose 6562
ClientAreaVisible 6563
ClientAreaLeft 6564
ClientAreaTop 6565
ClientAreaRight 6566
ClientAreaBottom 6567
ClientAreaWidth 6568
ClientAreaHeight 6569
OnDialogDragStart 6570
OnDialogDragProcess 6571
OnDialogDragEnd 6572
CAppDialog 6573
Create 6576
Destroy 6577
OnEvent 6578
Run 6579
ChartEvent 6580
Minimized 6581
IniFileSave 6582
IniFileLoad 6583
IniFileName 6584
IniFileExt 6585
CreateCommon 6586
CreateExpert 6587
CreateIndicator 6588
CreateButtonMinMax 6589
OnClickButtonClose 6590
OnClickButtonMinMax 6591
OnAnotherApplicationClose 6592
Rebound 6593
Minimize 6594
Maximize 6595
CreateInstanceId 6596
ProgramName 6597
SubwinOff 6598

36 MQL4 からの移行 6599

37 MQL5関数の表 6602

38 MQL5定数の表 6633

© 2000–2025, MetaQuotes Ltd.

MQL5 リファレンス

MetaQuotes Language 5 (MQL5) は、金融取引を自動化するテクニカル指標、売買ロボットやユーティリティアプリケーションを開発するために設計された高レベル言語です。MQL5 は MetaQuotes によって独自の取引プラットフォームのために開発されました。その言語構文は、オブジェクト指向プログラミング(OOP)スタイルでアプリケーションを開発するプログラマを可能にし、C ++ に非常に近いものです。

MQL5 言語に加えて、取引プラットフォームパッケージには、テンプレート、スニペット、デバッグ、プロファイリングや自動補完ツールといった高度なコード作成ツールを持つ MetaEditor IDE や、ファイルのバージョン管理を可能にする内蔵された MQL5 Storage を含みます。

言語サポートは、大規模な無償のコードベースと大量の記事を持つMQL5 Algotrading community ウェブサイトで提供されます。これらの記事は、ニューラルネットワーク、統計や分析、高頻度取引、裁定、テスト、及び取引ストラテジーの最適化、取引の自動化ロボットの使用、などの現代的な取引の全ての側面をカバーしています。

トレーダーとMQL5　プログラム開発者は、フォーラム上で通信し、フリーランスサービスを使用してアプリケーションを注文及び開発し、自動取引アプリケーションの市場で保護されたプログラムを売買出来ます。

MQL5 言語は専門的な取引関数と事前定義されたイベントハンドラを提供し、プログラマが、特定の取引ルールに従って取引プロセスを制御するエキスパートアドバイザー(EAs)を開発するのを支援します。EAs に加えて、MQL5 は、カスタムテクニカル指標、スクリプト及びライブラリの開発を可能にします。

この MQL5 言語リファレンスは、関数、演算、予約語及び他の言語の構造のカテゴリに分かれています。このリファレンスには、取引ストラテジー、制御パネル、カスタムグラフィックスを展開し、ファイルアクセスを可能にするために使用される標準ライブラリクラスの記述も含まれます。

更に、コードベースには様々な数学的な問題を解決するために使用出来るALGLIB数値解析ライブラリを含まれます。

アルゴリズム取引の本

新しいことを学び始めるのは常に挑戦です。初心者をサポートするために、アルゴリズム取引のロボットやアプリケーションの作成を学びたい人向けに設計された、MQL5プログラミングに関する2冊の包括的な書籍をリリースしました。

学習プロセスをよりスムーズに進められるよう、これらの書籍は資料を体系的かつ構造的にまとめています。自動売買ロボットやアプリケーションの段階的な作成が詳細なコード例で解説されており、アルゴリズム取引のポイントを深く理解できます。また、実践的な演習が多数含まれており、習得した知識を強化し、実際の取引環境でプログラミングスキルを向上させることができます。

「トレーダー向けMQL5プログラミング」は、MQL5に関する最も完全で詳細なチュートリアルであり、あらゆるレベルのプログラマーに対応しています。初心者はこの本を通じて基礎を学び、開発ツールや基本的なプログラミングの概念を学びます。MetaTrader 5取引プラットフォーム上で、最初のアプリケーションを作成、コンパイル、実行する方法も紹介されています。他のプログラミング言語の経験がある方は、すぐに自動売買ロボットや分析アプリケーションの作成に進むことができます。

「MQL5を使用したアルゴリズム取引のためのニューラルネットワーク」は、MetaTrader 5プラットフォームで自動売買ロボットに機械学習を応用するためのガイドです。ニューラルネットワークの基礎からアルゴリズム取引での応用までがステップごとに解説されています。学習を進めながら、独自のAIソリューションを構築・訓練し、新しい機能を追加していくことができます。また、MQL5に加えて、PythonやOpenCLのプログラミングス

© 2000–2025, MetaQuotes Ltd.

キルを習得し、行列やベクトルの手法を使用して、複雑な数学的問題を効率的に解決する方法を学びます。

取引アプリケーションの開発に関する記事

MQL5記事は、この言語の可能性を最大限に引き出すための素晴らしいリソースで、幅広いアルゴリズム取引の実践的なタスクがカバーされています。記事は例、エキスパートアドバイザー、機械学習などのセクションに整理されているため、簡単に目的の記事を見つけることができます。毎月、MQL5アルゴ取引コミュニティのWebサイトには、トレーダーが他のトレーダー向けに書いた数十の記事が公開されます。これらの記事を通じて、最新のアルゴリズム取引を学び、議論に参加してください。初心者向けに、MQL5をすぐに理解するための推奨記事16選も用意しています。

MQL5 アプリケーションの種類

MQL5 プログラムは、実装する取引オートメーションタスクに基づいて、4つの専門的な種類に分類されます。

- **エキスパートアドバイザー** は、特定のチャートにリンクされる自動取引システム（MTS）です。エキスパートアドバイザーは、適切な取引ストラテジー要素の実行を有効にする事前定義されたイベントを処理するためのイベントハンドラを持っています。プログラム初期化と初期化解除イベント、新規ティックイベント、タイマーイベント、板情報の変更イベント、チャートとカスタムイベントがその例です。
  実装ルールに基づいた売買シグナルの計算に加えて、エキスパートアドバイザーは自動売買を実行して、取引サーバに直接送信することが出来ます。エキスパートアドバイザーは、*<Terminal_Directory>\MQL5\Experts* に格納されています。
- **カスタム指標** は、標準的な指標に加えてユーザが開発した技術的な指標で、取引プラットフォームに統合されています。カスタム指標も標準的な指標も、自動売買は出来ず、出来るのは分析関数を実装することだけです。カスタム指標は計算のために他の指標の値を利用することができ、エキスパートアドバイザーから呼び出すことが出来ます。
  カスタム指標は*<Terminal_Directory>\MQL5\Indicators* に格納されます。
- **スクリプト** は、アクションを1回実行するためのプログラムです。エキスパートアドバイザーとは異なり、スクリプトはトリガ、初期化と初期化解除以外のイベントを処理しません。スクリプトコードには OnStart 処理関数が必要です。
  スクリプトは*<Terminal_DIrectory>\MQL5\Scripts に格納されています。*
- **サービス**は、指標、エキスパートアドバイザー、スクリプトと違って、チャートから独立して動作できます。スクリプトと同様に、サービスはトリガー以外のイベントを処理しません。サービスを起動するには、そのコードにOnStartハンドラ関数を含める必要があります。サービスはStart以外のイベントを受け付けませんが、EventChartCustomを使用してチャートにカスタムイベントを送信できます。サービスは*〈Terminal_ Directory〉¥MQL5¥Servicesに格納されます。*
- **ライブラリ**はカスタム関数のセットです。ライブラリは、一般的に使用されるカスタムプログラムアルゴリズムの保存及び配布を意図しています。
  ライブラリは *<Terminal_Directory>\MQL5\Libraries*に格納されています。
- **インクルードファイル**はカスタムプログラムの最も頻繁に使用されるブロックのソーステキストです。このようなファイルは、コンパイル段階で、エキスパートアドバイザー、スクリプト、カスタム指標及びライブラリのソーステキストに含むことが出来ます。インクルードファイルの使用は、負担を発生するライブラリ関数の呼び出しより好ましいです。
  インクルードファイルは、元のファイルと同じディレクトリに格納することが出来ます。この場合 #include ディ

© 2000–2025, MetaQuotes Ltd.

レクティブが二重引用符と使用されます。別のオプションは、インクルードファイルを *<Terminal_Directory>\MQL5\Include* に格納することです。この場合 #include は角括弧と使用されるべきです。

© 2000-2025、MetaQuotes Ltd

言語基礎

MetaQuotes Language 5（MQL5）は、様々な金融市場の分析のために自動化された取引ストラテジーやカスタムテクニカル指標を書くための高レベルのオブジェクト指向プログラミング言語です。リアルタイムで動作するエキスパートシステムの開発に加え、独自のグラフィカルツールを作成して取引に必要な決断を支援することも出来ます。

MQL5 は人気高い C++ プログラミング言語の概念に基づいています。MQL4 と比較すると、MQL5 には新たに列挙、構造体、クラスとイベントハンドルが含まれています。基本データ型の数の増加によって、DLL を介しての MQL5 プログラムと他のアプリケーションとの相互作用が簡単かつ可能にされました。MQL5 の構文は C++ に似ており、近代的なプログラミング言語から MQL5 プログラムへの変換は簡単に出来ます。

MQL5 言語を学びやすくするために、トピックは以下のセクションに分類されています。

- 構文
- データ型
- 演算子と式
- 演算子
- 関数
- 変数
- プリプロセッサ
- オブジェクト指向プログラミング
- 名前空間

© 2000–2025, MetaQuotes Ltd.

構文

トレーディングストラテジーのプログラミングに使用される MQL5 言語の構文は、一部の機能を除いて、C ++ プログラミング言語に非常に似ています。

- アドレス計算は出来ません。
- goto 構文はありません。
- 匿名の列挙を宣言することは出来ません。
- 多重継承はありません。

参照

列挙、構造体とクラス、継承

© 2000–2025, MetaQuotes Ltd.

コメント

複数行のコメントは、始めに「/*」記号、終わりに「*/」記号を記載します。この種類のコメントは入れ子に出来ません。 一行コメントは、「//」記号で始まって改行記号で終わり、複数行コメントに入れ子にすることが出来ます。コメントはスペースが書き込められるところならどこにも書き込むことが可能で、コメント自体にも任意の数のスペースを使用出来ます。

例：

```
//--- 一行コメント
/*   複数
   行の            // 入れ子にされた一行コメント
   コメント
*/
```

© 2000-2025, MetaQuotes Ltd.

識別子

識別子は、変数名及び関数名として使用されます。識別子の長さは63文字を超えることは出来ません。

識別子に利用出来る記号は、0 ～ 9 の数字、ラテン文字の大文字と小文字の a ～ z、A ～ Z(異なる記号として認識)、及び下線(_)の記号です。最初の記号は数字であることは出来ません。

識別子は予約語と同一ではいけません。

例：

```
NAME1 name1 Total_5 Paper
```

参照

変数、関数

© 2000–2025, MetaQuotes Ltd.

予約語

下記の識別子は、予約語として使用されています。 それぞれに特定のアクションが割り当てられており、他の目的での使用は出来ません。

データ型

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

アクセス指定子

const	private	virtual
delete	protected	
override	public	

メモリクラス

extern	input	static

演算子

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

その他

this	#define	#import
true	#ifdef	#include
false	#ifndef	#property

© 2000–2025, MetaQuotes Ltd.

this	#define	#import
template	#else	group
typename	#endif	namespace

© 2000–2025, MetaQuotes Ltd.

データ型

プログラムはデータで動作します。データは目的によって異なった型で存在します。例えば、整数データは配列要素のアクセスに使用されます。価格データは倍精度浮動小数点に属します。これは MQL5 には価格データ型のための特別なデータ型がないという事実と関係があります。

データ処理速度は型によって違います。最も速く処理されるのは整数データです。倍精度データの処理には、特別なコプロセッサが使用されます。しかし、浮動小数点データは内部表現が複雑なために、整数より処理速度が遅くなります。

文字列データは動的なコンピュータメモリの割り当て / 再配分を必要とするために処理速度が最も遅くなります。

基本データ型は次の通りです。

- 整数型（char、short、int、long、uchar、ushort、uint、ulong）
- 論理型（bool）
- リテラル型（ushort）
- 文字列型（string）
- 浮動小数点型（double、float）
- カラー型（color）
- 日付時刻型（datetime）
- 列挙型（enum）

複合データ型は次の通りです。

- 構造体
- クラス

OOPの視点からすると、複合データ型は抽象データ型と呼ばれています。

*カラー*及び*日付時刻*型は、外部のエキスパートアドバイザーまたはカスタム指標のプロパティで定義されたパラメータの可視化と入力を容易にするためにのみ意味をなします（インプットタブ）。*カラー*及び*日付時刻*型のデータは整数として表されます。整数型と浮動小数点型は、算術演算（数値）型と呼ばれています。

式には、明示的なキャストが指定されていない限り、暗黙の型キャストのみが利用されます。

参照

型キャスト

© 2000–2025, MetaQuotes Ltd.

整数型

MQL5 では整数は 11 の型で表されます。そのうちのいくつかは、プログラムロジックで必要とされる場合、他の型と一緒に使用されますが、この場合は型キャストの規則を覚えていなければなりません。

下の表は各型の特徴を示します。また、最後の列は、各型に対応する C++ の型を示します。

Type	バイトサイズ	最小値	最大値	C++
char	1	-128	127	char
uchar	1	0	255	unsigned char、BYTE
bool	1	0（false、偽）	1（true、真）	bool
short	2	-32 768	32 767	short、wchar_t
ushort	2	0	65 535	unsigned short、WORD
int	4	- 2 147 483 648	2 147 483 647	int
uint	4	0	4 294 967 295	unsigned int、DWORD
color	4	-1	16 777 215	int、COLORREF
long	8	-9 223 372 036 854 775 808	9 223 372 036 854 775 807	__int64
ulong	8	0	18 446 744 073 709 551 615	unsigned __int64
datetime	8	0（1970.01.01 0:00:00）	32 535 244 799（3000.12.31 23:59:59）	__time64_t

整数型の値は、数値定数、色リテラル、日付時刻リテラル、文字定数と列挙としても表示出来ます。

参照

変換関数、数値定数

© 2000–2025, MetaQuotes Ltd.

Char、Short、Int 及び Long 型

char

char 型は、メモリの 1 バイト（8 ビット）を占め、2 進数で 2^8=256 個の値を表現することが出来ます。*char* 型は、正と負の両方の値を含むことが出来ます。値の範囲は -128 から 127 です。

uchar

uchar 整数型も *char* 型のようにメモリの 1 バイトを占めますが、*uchar* は char と違って正の値のみを含むことが出来ます。最小値は 0 で最大値は 255 です。*uchar* 型の名称の最初の文字の u は *unsigned* の略です。

short

short 型は 2 バイト（16 ビット）で、2 から 2 の 16 乗の 2^16 = 65,536 までの数を表すことが出来ます。*short* 型は符号付きなので、-32,768 から 32,767 の正と負の値を含むことが出来ます。

ushort

符号なしの *short* 型は *ushort* で 2 バイトです。最小値は 0 で最大値は 65,535 です。

int

int 型は 4 バイト（32 ビット）です。最小値は -2,147,483,648 で最大値は 2,147,483,647 です。

uint

符号なしの整数型は *uint* です。この型は 4 バイトのメモリを取り、0 から4,294,967,295 の整数を表すことが出来ます。

long

long 型は 8 バイト（64 ビット）です。最小値が -9,223,372,036,854,775,808 で最大値は 9,223,372,036,854,775,807 です。

ulong

ulong 型も 8 バイトを占し、0 から18,446,744,073,709,551,615 の値を格納することが出来ます。

例：

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

符号なしの整数型は負の値の格納には設計されていないので、格納の試みは予期しない結果につながることがあります。下記の簡単なスクリプトが無限ループにつながります。

```
//--- 無限ループ
```

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
   uchar  u_ch;

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

正しいバージョンは下記です。

```
//--- 正しいバージョン
void OnStart()
  {
   uchar  u_ch;

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

結果:

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

例:

```
//--- 負の値は unsigned 型に格納出来ない
```

© 2000-2025, MetaQuotes Ltd.

```
uchar  u_ch=-120;
ushort u_sh=-5000;
uint   u_in=-401280;
```

16 進数は、数字の 0～9 及び 10～15 の値を記す a～f または A～F の文字で表され、0x または 0X で始まります。

例：

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

整数型の変数の値は、B プレフィックスを使用して2 進数で設定出来ます。例えば、取引セッションの作業時間はint 型の変数にエンコードし、必要なアルゴリズムに従って、情報を使用することが出来ます。

```
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 操作時間には 1、それ以外には 0 を設定する
   int AsianSession    =B'111111111'; // 亜州セッション( 0:00～9:00)
   int EuropeanSession=B'1111111110000000000'; // 欧州セッション(9:00～18:00)
   int AmericanSession =B'111111110000000000000011'; // 北米セッション(16:00～02:00)
//--- セッションの数値を導き出す
   PrintFormat("Asian session hours as value =%d",AsianSession);
   PrintFormat("European session hours as value is %d",EuropeanSession);
   PrintFormat("American session hours as value is %d",AmericanSession);
//--- セッションの操作時間を文字列として表示する
   Print("Asian session ",GetHoursForSession(AsianSession));
   Print("European session ",GetHoursForSession(EuropeanSession));
   Print("American session ",GetHoursForSession(AmericanSession));
//---
  }
//+------------------------------------------------------------------+
//|  セッションの操作時間を文字列として返す                           |
//+------------------------------------------------------------------+
string GetHoursForSession(int session)
  {
//--- チェックには AND ビット演算子を使用して 1 ビットで左シフトする( <<=1)
//--- 最下位ビットからチェックを開始
   int bit=1;
   string out="working hours: ";
//--- 0～23 の 24 ビットをすべてチェックする
   for(int i=0;i<24;i++)
     {
      //--- ビットの状態を数で受け取る
      bool workinghour=(session&bit)==bit;
      //--- 時間の数をメッセージに追加する
      if(workinghour )out=out+StringFormat("%d ",i);
      //--- 左に 1 ビットシフトし、次の値をチェックする
```

© 2000-2025, MetaQuotes Ltd.

```
      bit<<=1;
     }
//--- 結果の文字列
   return out;
  }
```

参照

型キャスト

© 2000–2025, MetaQuotes Ltd.

文字定数

MQL5 では文字列の要素としての文字は、Unicode 文字セットのインデックスです。文字は整数にキャスト出来る16 進数値で、加算と減算のような整数演算によって操作することが出来ます。

引用符内の文字及び「¥x10」などの文字の 16 進数の ASCII コードは文字定数で、ushort 型です。例えば、「0」タイプのレコードは、文字のテーブルのインデックス 0 に対応し、数値 30 を持ちます。

例:

```
void OnStart()
  {
//--- 文字定数を定義する
   int symbol_0='0';
   int symbol_9=symbol_0+9; // 「9」シンボルを取得
//--- 定数の出力値
   printf("In a decimal form: symbol_0 = %d,  symbol_9 = %d",symbol_0,symbol_9);
   printf("In a hexadecimal form: symbol_0 = 0x%x,  symbol_9 = 0x%
x",symbol_0,symbol_9);
//--- 文字列に定数を入れる
   string test="";
   StringSetCharacter(test,0,symbol_0);
   StringSetCharacter(test,1,symbol_9);
//--- 文字列で表示する
   Print(test);
  }
```

バックスラッシュは、コンパイラがプログラムのソーステキスト内の定数文字列と文字定数を扱う際の制御文字です。単一引用符(')、二重引用符(″)、バックスラッシュ(¥)及び制御文字などの記号は、以下の表に従って、バックスラッシュ(¥) で始まるシンボルの組み合わせとして表すことが出来ます。

文字名	ニーモニックコードや画像	MQL5レコード	数値
改行(ラインフィード)	LF	'\n'	10
水平タブ	HT	'\t'	9
キャリッジリターン	CR	'\r'	13
バックスラッシュ	\	'\\'	92
単一引用符	'	'\''	39
二重引用符	"	'\"'	34
16 進コード	hhhh	'\xhhhh'	1～4 個の16 進文字
10 進コード	d	'\d'	0～65,535 の 10 進数

バックスラッシュに上記以外の文字が続く場合、結果は未定義です。

例

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//--- 文字定数を宣言する
   int a='A';
   int b='$';
   int c='©';       // コード 0xA9
   int d='\xAE';    // シンボル ® のコード
//--- 定数を出力する
   Print(a,b,c,d);
//--- 文字列に文字を追加する
   string test="";
   StringSetCharacter(test,0,a);
   Print(test);
//--- 文字列内の文字を置換する
   StringSetCharacter(test,0,b);
   Print(test);
//--- 文字列内の文字を置換する
   StringSetCharacter(test,0,c);
   Print(test);
//--- 文字列内の文字を置換する
   StringSetCharacter(test,0,d);
   Print(test);
//--- 文字を数として表す
   int a1=65;
   int b1=36;
   int c1=169;
   int d1=174;
//--- 文字列に文字を追加する
   StringSetCharacter(test,1,a1);
   Print(test);
//--- 文字列に文字を追加する
   StringSetCharacter(test,1,b1);
   Print(test);
//--- 文字列に文字を追加する
   StringSetCharacter(test,1,c1);
   Print(test);
//--- 文字列に文字を追加する
   StringSetCharacter(test,1,d1);
   Print(test);
  }
```

上述されたように、文字定数（または変数）の値は、文字のテーブルのインデックスです。インデックスは整数なので、様々な方法で記述することが出来ます。

```
void OnStart()
  {
//---
   int a=0xAE;      // ® のコードは '¥xAE' リテラルに対応する
   int b=0x24;      // $ のコードは '¥x24' リテラルに対応する
```

© 2000-2025, MetaQuotes Ltd.

```
   int c=0xA9;      // © のコードは '¥xA9' リテラルに対応する
   int d=0x263A;    // ☺ のコードは '¥x263A' リテラルに対応する
//--- 値を表示する
   Print(a,b,c,d);
//--- 文字列に文字を追加する
   string test="";
   StringSetCharacter(test,0,a);
   Print(test);
//--- 文字列内の文字を置換する
   StringSetCharacter(test,0,b);
   Print(test);
//--- 文字列内の文字を置換する
   StringSetCharacter(test,0,c);
   Print(test);
//--- 文字列内の文字を置換する
   StringSetCharacter(test,0,d);
   Print(test);
//--- スートのコード
   int a1=0x2660;
   int b1=0x2661;
   int c1=0x2662;
   int d1=0x2663;
//--- スペードの文字を追加する
   StringSetCharacter(test,1,a1);
   Print(test);
//--- ハートの文字を追加する
   StringSetCharacter(test,2,b1);
   Print(test);
//--- ダイヤの文字を追加する
   StringSetCharacter(test,3,c1);
   Print(test);
//--- クラブの文字を追加する
   StringSetCharacter(test,4,d1);
   Print(test);
//--- 文字列内の文字リテラルの例
   test="Queen\x2660Ace\x2662";
   printf("%s",test);
  }
```

リテラル文字の内部表現は ushort 型です。文字定数は 0 から 6,5535 までの値を受け入れることが出来ます。

参照

StringSetCharacter()、StringGetCharacter()、ShortToString()、ShortArrayToString()、StringToShortArray()

© 2000-2025, MetaQuotes Ltd.

日付時刻型

日付時刻型は日付と時刻を 1970 年 1 月 1 日からの経過秒数として格納するために意図されています。この型はメモリの 8 バイトを有します。

日付と時刻の定数は、年、月、日（または日、月、年）、時間、分、秒の 6 つの数値を示す部分で構成され、文字列リテラルとして表すことが出来ます。定数は、一重引用符で囲まれ、D の文字で始まります。

値の範囲は 1970 年 1 月 1 日から 3000 年 12 月 31 日までです。日付（年、月、日）と時間（時間、分、秒）は片方づつ、または両方とも、省略することが出来ます。

リテラルの日付指定では、年、月及び日を指定することが望ましいです。そうでないとコンパイラが不完全なエントリについて警告を返します。

例：

```
datetime NY=D'2015.01.01 00:00';      // 2015 年の初めの時刻
datetime d1=D'1980.07.19 12:30:27';   // 年月日時分秒
datetime d2=D'19.07.1980 12:30:27';   // D'1980.07.19 12:30:27' と同じ
datetime d3=D'19.07.1980 12';         // D'1980.07.19 12:00:00' と同じ
datetime d4=D'01.01.2004';            // D'01.01.2004 00:00:00' と同じ
datetime compilation_date=__DATE__;              // コンパイル日付
datetime compilation_date_time=__DATETIME__;     // コンパイル日付と時刻
datetime compilation_time=__DATETIME__-__DATE__;// コンパイル時刻
//--- コンパイラが警告を返す宣言の例
datetime warning1=D'12:30:27';        // D'[date of compilation] 12:30:27' と同じ
datetime warning2=D'';                // __DATETIME__ と同じ
```

参照

データ型構造体、日付時刻、TimeToString、StringToTime

© 2000-2025, MetaQuotes Ltd.

カラー型

カラー型は、色に関する情報を格納するもので、メモリ内で 4 バイトを有します。最初のバイトは無視され、残りの 3 バイトが RGB 成分を含有します。

カラー定数は、リテラル、整数、または（名称のついた ウェブカラー のみ）名称によっての 3 つの方法で表現することが出来ます。

リテラル表現は、三原色（赤、緑、青）の割合を表す 3 つの部分から構成されます。定数は C で始まり、単一引用符で囲まれています。色成分の割合の範囲は 0～255 です。

整数値は 16 進数または 10 進数として表現されます。16 進数は 0x00BBGGRR のような形をとります。RR は赤色、GG は緑色、BB は青色成分の割合です。10 進数の定数は直接 RGB を反映しません。それらは 16 進数表現の 10 進数での値を表します。

特定の色がいわゆる ウェブカラー セットを表します。

例：

```
//--- リテラル
C'128,128,128'     // 灰色
C'0x00,0x00,0xFF' // 青色
// 色の名称
clrRed             // 赤色
clrYellow          // 黄色
clrBlack           // 黒色
//--- 整数表現
0xFFFFFF           // 白色
16777215           // 白色
0x008000           // 緑色
32768              // 緑色
```

参照

ウェブカラー、ColorToString、StringToColor、型 キャスト

© 2000–2025, MetaQuotes Ltd.

ブール型

ブール型は true または false の論理値の格納に意図されています。それぞれの数値表現は 1 と 0 です。

例：

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

内部表現は 1 バイトサイズの整数です。論理式では、他の整数または実数型や式を使用することが出来ることに留意すべきです。コンパイラはエラーを生成しません。この場合、ゼロ値は false として解釈され、他の全ての値は true とされます。

例：

```
   int i=5;
   double d=-2.5;
   if(i) Print("i = ",i," and is set to true");
   else Print("i = ",i," and is set to false");

   if(d) Print("d = ",d," and has the true value");
   else Print("d = ",d," and has the false value");

   i=0;
   if(i) Print("i = ",i," and has the true value");
   else Print("i = ",i," and has the false value");

   d=0.0;
   if(d) Print("d = ",d," and has the true value");
   else Print("d = ",d," and has the false value");

//--- 実行結果
//   i= 5 and has the true value
//   d= -2.5 and has the true value
//   i= 0 and has the false value
//   d= 0 and has the false value
```

参照

ブール演算子、優先順位のルール

© 2000-2025, MetaQuotes Ltd.

列挙型

列挙 型のデータは特定の限定されたセットに属します。下記が列挙型の定義です。

```
enum 列挙型名
  {
  値のリスト
  };
```

値のリストは、コンマで区切られた名前付き定数の識別子のリストです。

例:

```
enum months  // 名前付き定数の列挙
   {
    January,
    February,
    March,
    April,
    May,
    June,
    July,
    August,
    September,
    October,
    November,
    December
   };
```

列挙の宣言後、新しい 4 バイトの整数値を持つデータ型が出現します。列挙は新しい名前付き定数を導入するため、新しいデータ型の宣言によって、コンパイラは渡されたパラメータの型を厳密に制御するようになります。上記の例では、January と名付けられた定数が 0、February が 1、December が 11 です。

規則:列挙型のメンバである名前付き定数に 特定の値が割り当てられていない場合は、新しい値が自動的に形成されることになります。それが列挙体の最初のメンバである場合、0 の値が割り当てられます。後続するメンバの全ての値は、前のメンバの値に 1 を足すことによって計算されます。

例:

```
enum intervals  // 名前付き定数の列挙
   {
    month=1,     // 1 ヶ月の間隔
    two_months,  // 2 ヶ月
    quarter,     // 3ヶ月(四半期)
    halfyear=6,  // 半年
    year=12,     // 1 年(12ヶ月)
   };
```

注意事項

- C++とは異なり、MQL5 での列挙型の内部表現のサイズは常に 4 バイトです。つまり、sizeof(月の名称)は 4 を返します。

© 2000-2025, MetaQuotes Ltd.

- C++とは異なり、MQL5 では匿名の列挙は宣言することは出来ません。つまり、enum キーワードの後に常に一意の名称を指定する必要があります。

参照

型キャスト

© 2000–2025, MetaQuotes Ltd.

浮動小数点数型（double、float）

実数型（または浮動小数点型）は、小数部分を持つ値を表します。MQL5 言語には 2 種類の浮動小数点型があります。コンピュータメモリ内での実数値の表現方法は、IEEE 754 標準によって定義され、プラットフォーム、オペレーティングシステムまたはプログラミング言語とは無関係です。

Type	バイトサイズ	最小の正の値	最大値	C++
float	4	1.175494351e-38	3.402823466e+38	float
double	8	2.225073858507 2014e-308	1.797693134862 3158e+308	double

double という名称は、その数値の精度が float の 2 倍であるということを示しています。ほとんどの場合 double 型が一番便利です。float の限られた精度は多くの場合に充分ではありません。float 型がまだ使用されているのはメモリ節約のためです（これは実数の大きな配列のために重要です）。

浮動小数点定数は、整数部、点（ . ）と小数部で構成されています。整数部と小数部は十進数の配列です。

例：

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

実数には科学表記があり、多くの場合、科学表記での記録は、従来のものよりもコンパクトです。

例：

```
double c1=1.12123515e-25;
double c2=0.00000000000000000000000000112123515; // 小数点の後に 24 のゼロ

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

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

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

実数は、一般的に 10 進表記が使用されていてもメモリでは限られた精度を持つ 2 進数システムで格納されていることを忘れてはなりません。これが、10進数では正確に表現されている数字の多くが 2 進数システムでは無限小数としてのみ書くことが可能な理由です。

例えば、2 の冪乗である 0.25 が正確に保存されているのに比べて、0.3 及び 0.7 はコンピュータによって無限小数として表されます。

この点から、2 つの実数の平等さを比較することは不正なので推奨されません。

© 2000–2025, MetaQuotes Ltd.

例:

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

どうしても 2 つの実数の平等さを比較する必要がある場合は、2 つの方法があります。最初の方法は、2 つの数の差を比較の精度を指定する小さな数と比較することです。

例:

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

上記の例では epsilon の値は事前定義された定数 DBL_EPSILON より小さく出来ないことにご注意ください。この定数の値は 2.2204460492503131e-016 です。float 型に対しての定数は FLT_EPSILON = 1.192092896e-07 です。これらの値は 1.0 + DBL_EPSILON = 1.0(float 型の値に対しては 1.0 + FLT_EPSILON!= 1.0)を満たす最小値です。

© 2000-2025, MetaQuotes Ltd.

2 つ目の方法では 2 つの実数の正規化後の差分とゼロを比較します。正規化された数との数学的な操作は、非正規化された結果をもたらすので、正規化された数の差をゼロと比較することは無意味です。

例:

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

数学コプロセッサでの演算は無効な実数をもたらすことがあります。無効な実数との演算の結果は定義されていないため、そのような実数は数学演算や比較の演算に使用することは出来ません。例えば、2 のアークサインを計算しようとすれば、結果は負の無限大です。

例:

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

負の無限大の他には正の無限大とNaN(非数)があります。数の無効性を決定するためには MathIsValidNumber() が使用出来ます。IEEE標準によれば、これらには特殊な機械表現があります。例えば、double 型の正の無限大は 0x7FF0 0000 0000 0000 です。

例:

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

//--- 開始
   str1 s1;
   str2 s2;
```

© 2000-2025, MetaQuotes Ltd.

```
//---
   s1.d=MathArcsin(2.0);         // 無効な数を取得 -1.#IND
   s2=s1;
   printf("1.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0xFFFF000000000000;      // 無効な数 -1.#QNAN
   s1=s2;
   printf("2.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FF7000000000000;      // 最大非数 SNaN
   s1=s2;
   printf("3.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FF8000000000000;      // 最小非数QNaN
   s1=s2;
   printf("4.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FFF000000000000;      // 最大非数 QNaN
   s1=s2;
   printf("5.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FF0000000000000;      // 正の無限大 1.#INF と最小非数 SNaN
   s1=s2;
   printf("6.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0xFFF0000000000000;      // 負の無限大 -1.#INF
   s1=s2;
   printf("7.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x8000000000000000;      // 負のゼロ -0.0
   s1=s2;
   printf("8.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x3FE0000000000000;      // 0.5
   s1=s2;
   printf("9.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x3FF0000000000000;      // 1.0
   s1=s2;
   printf("10.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FEFFFFFFFFFFFFF;      // 最大の正規化数（MAX_DBL）
   s1=s2;
   printf("11.  %.16e %I64X",s1.d,s2.l);
//---
   s2.l=0x0010000000000000;      // 最小の正の正規化数(MIN_DBL)
   s1=s2;
   printf("12.  %.16e %.16I64X",s1.d,s2.l);
//---
```

© 2000-2025, MetaQuotes Ltd.

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

参照

DoubleToString、NormalizeDouble、数値定数

© 2000–2025, MetaQuotes Ltd.

複素数（complex）

組み込みのcomplexタイプは、2つのdoubleフィールドを持つ構造体です。

```
struct complex
  {
   double            real;    // 実数部分
   double            imag;    // 虚数部分
  };
```

「complex」型は、MQL5関数のパラメータとして値で渡すことができます（通常の構造体が参照によってのみ渡されるのとは対照的です）。DLLからインポートされた関数の場合、「complex」型は参照によってのみ渡されます。

「i」接尾辞はcomplex定数を説明するために使用されます。

```
complex square(complex c)
  {
   return(c*c);
  }

void OnStart()
  {
   Print(square(1+2i));   // 定数がパラメータとして渡される
  }
// 複素数の文字列表現である「( -3, 4 ) 」が出力される
```

現在、複素数には単純な演算（=, +, -, *, /, +=, -=, *=, /=, ==, !=）のみが使用できます。

追加の数学関数のサポートは後で追加され、絶対値、正弦、余弦などの計算が可能になります。

© 2000-2025, MetaQuotes Ltd.

文字列型

文字列型は、テキスト文字列を格納するために使用されます。テキスト文字列は、ゼロで終わる Unicode 形式の文字列です。文字列定数は、文字列変数に割り当てることが出来ます。文字列定数は、"これは文字列定数です" のように二重引用符で囲まれた Unicode 文字の配列です。

文字列に二重引用符 (") を含む必要がある場合は、その前にバックスラッシュ (¥) を置かれなければなりません。特別文字定数はバックスラッシュ文字 (¥) が前に入力された場合文字列で記述することが出来ます。

例 :

```
string svar="This is a character string";
string svar2=StringSubstr(svar,0,4);
Print("Copyright symbol\t\x00A9");
FileWrite(handle,"This string contains a new line symbols \n");
string MT5path="C:\\Program Files\\MetaTrader 5";
```

ソースコードを読みやすくするために、長い定数文字列は、加算演算なしで分割することが出来ます。コンパイル時に、各部分は1つの長い文字列に結合されます。

```
//--- 長い文字列定数を宣言する
   string HTML_head="<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.0
Transitional//EN\""
                    " \"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd\">\n"
                    "<html xmlns=\"http://www.w3.org/1999/xhtml\">\n"
                    "<head>\n"
                    "<meta http-equiv=\"Content-Type\" content=\"text/html;
charset=utf-8\" />\n"
                    "<title>Trade Operations Report</title>\n"
                    "</head>";
//--- 文字列定数をログに出力する
   Print(HTML_head);
  }
```

参照

変換関数、文字列関数、FileOpen、FileReadString、FileWriteString

© 2000-2025, MetaQuotes Ltd.

構造体、クラス、インターフェイス

構造体

構造体は void 型を除いた任意のタイプの要素の集合です。従って、構造体は、異なるタイプの論理的に関連するデータを組み合わせます。

構造体の宣言

構造体のデータ型は、以下の記載によって決定されます。

```
struct 構造体名
  {
  要素の記述
  };
```

構造体名は識別子（変数または関数の名称）として使用することは出来ません。MQL5 では構造要素が位置合わせ（アライン）されず、直接に前後に並んでいることに留意すべきです。C++ では、このような順序は、次の命令を使用して、コンパイラに行われます。

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

構造体内で位置合わせをしたい場合は、適切なサイズに補助メンバである「フィラー」を使用します。

例：

```
struct trade_settings
  {
   uchar  slippage;     // 使用可能の slippage サイズが 1 バイト
   char   reserved1;    // 1 バイトを抜かす
   short  reserved2;    // 2 バイトを抜かす
   int    reserved4;    // 4 バイトを抜かし、8 バイト境界の整列を保証する。
   double take;         // 利益固定の値
   double stop;         // プロテクティブストップの値
  };
```

このような整合構造体の記述は、インポートされた DLL 関数に転送するためのみに必要です。

注意：この例は誤って設計されたデータを示します。初めに *take* 及び *stop* の double型の大きいデータを宣言してから uchar 型の *slippage* を宣言したほうがよりよい宣言となります。*この場合には、データの内部表現は#pragma pack()*で指定された値に関係なく、常に同じになります。

構造体が string 型の変数及び/または動的配列オブジェクトを含む場合、コンパイラによって暗黙のコンストラクタが割り当てられます。このコンストラクタは、string 型の全ての構造体のメンバをリセットし、動的配列オブジェクトを正しく初期化します。

基本構造体

文字列、クラスオブジェクト、ポインタ、動的配列オブジェクトが含まれていない構造体は基本構造体と呼ばれます。基本構造体型の変数や基本構造体の配列型の変数は DLL からインポートされた関数にパラメータとして渡されることが出来ます。

© 2000-2025, MetaQuotes Ltd.

基本構造体をコピーすることは、2つの場合にのみ許可されます。

- オブジェクトが同じ構造体型の場合
- オブジェクトが、一つの構造体があと一つの構造体の子孫であることを意味する系統によって関連付けられている場合

例として、内容が内蔵されたMqlTickと同一なCustomMqlTickカスタム構造体を開発してみましょう。 コンパイラは、MqlTickオブジェクト値をCustomMqlTick型オブジェクトにコピーすることを許可しません。必要な型に直接型キャストすると、コンパイルエラーも発生します。

```
      //--- 異なる型の基本構造体をコピーすることは禁じられている
      my_tick1=last_tick;                 // コンパイラはここでエラーを返す

      //--- 異なる型の構造体を互いに型キャストすることも禁止されている
      my_tick1=(CustomMqlTick)last_tick;// コンパイラはここでエラーを返す
```

したがって、残るのは、構造体要素の値を1つずつコピーするという選択肢だけです。同じ型のCustomMqlTickの値をコピーすることはできます。

```
      CustomMqlTick my_tick1,my_tick2;
      //--- 同じ型のCustomMqlTickオブジェクトは次のようにコピーすることができる
      my_tick2=my_tick1;

      //--- CustomMqlTick基本構造体オブジェクトから配列を作成し、それに値を書き込み
      CustomMqlTick arr[2];
      arr[0]=my_tick1;
      arr[1]=my_tick2;
```

操作ログにarr[]配列値を表示するのにはArrayPrint()関数が呼び出されます。

```
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 内蔵されたMqlTickと似ている構造体を開発する
   struct CustomMqlTick
     {
      datetime          time;          // 直近約定値更新時間
      double            bid;           // 現在の売値
      double            ask;           // 現在の買値
      double            last;          // 最終取引価格（Last）
      ulong             volume;        // 現在の直近約定ボリューム
      long              time_msc;      // ミリ秒単位の直近約定値更新時間
      uint              flags;         // ティックフラグ
     };
   //--- 直近のティック値を取得する
   MqlTick last_tick;
   CustomMqlTick my_tick1,my_tick2;
//--- MqlTick からCustomMqlTickへのデータのコピーを試みる
   if(SymbolInfoTick(Symbol(),last_tick))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- 無関係な基本構造体をコピーすることは禁止されている
       //1.  my_tick1=last_tick;                  // コンパイラはここでエラーを返す

       //--- 無関係の構造体を互いに型キャストすることも禁止されている
       //2.  my_tick1=(CustomMqlTick)last_tick; // コンパイラはここでエラーを返す

       //--- よって、構造体のメンバを一つずつコピーする
       my_tick1.time=last_tick.time;
       my_tick1.bid=last_tick.bid;
       my_tick1.ask=last_tick.ask;
       my_tick1.volume=last_tick.volume;
       my_tick1.time_msc=last_tick.time_msc;
       my_tick1.flags=last_tick.flags;

       //--- 同じ型のCustomMqlTickオブジェクトは次のようにコピーすることができる
       my_tick2=my_tick1;

       //--- CustomMqlTick基本構造体オブジェクトから配列を作成し、それに値を書き込み
       CustomMqlTick arr[2];
       arr[0]=my_tick1;
       arr[1]=my_tick2;
       ArrayPrint(arr);
//--- CustomMqlTick型のオブジェクトを含む配列の値を表示する例
       /*
                          [time]    [bid]    [ask]    [last] [volume]     [time_msc]
[flags]
       [0] 2017.05.29 15:04:37 1.11854 1.11863 +0.00000  1450000 1496070277157
2
       [1] 2017.05.29 15:04:37 1.11854 1.11863 +0.00000  1450000 1496070277157
2
       */
      }
   else
      Print("SymbolInfoTick() failed, error = ",GetLastError());
  }
```

2番目の例は、基本構造体を系統にコピーする関数を示します。Animal基本構造体があって、そこからCat構造体とDog構造体が継承されているとします。Animalオブジェクトと Catオブジェクト、またAnimalオブジェクトとDogオブジェクトをお互いにコピーすることはできますが、CatとDogは両方ともAnimal構造体の子孫ですがお互いにコピーすることはできません。

```
//--- 犬を記述するための構造体
struct Dog: Animal
  {
   bool              hunting;        // 狩猟犬
  };
//--- 猫を記述するための構造体
struct Cat: Animal
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   bool              home;          // ペット用の猫
  };
//--- 子構造体のオブジェクトを作成する
   Dog dog;
   Cat cat;
//--- 祖先から子孫にコピーすることができる（Animal ==> Dog）
   dog=some_animal;
   dog.swim=true;    // 犬は泳げる
//--- 子構造のオブジェクトをコピーすることはできない（Dog != Cat）
   cat=dog;          // コンパイラがエラーを返す
```

この例の完全なコード：

```
//--- 動物を記述するための基本構造体
struct Animal
  {
   int               head;          // 頭の数
   int               legs;          // 脚の数
   int               wings;         // 羽の数
   bool              tail;          // 尾
   bool              fly;           // 飛ぶ
   bool              swim;          // 泳ぐ
   bool              run;           // 走る
  };
//--- 犬を記述するための構造体
struct Dog: Animal
  {
   bool              hunting;       // 狩猟犬
  };
//--- 猫を記述するための構造体
struct Cat: Animal
  {
   bool              home;          // ペット用の猫
  };
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 基本的なAnimal型のオブジェクトを作成して記述する
   Animal some_animal;
   some_animal.head=1;
   some_animal.legs=4;
   some_animal.wings=0;
   some_animal.tail=true;
   some_animal.fly=false;
   some_animal.swim=false;
   some_animal.run=true;
//--- 子の型のオブジェクトを作成する
```

© 2000-2025, MetaQuotes Ltd.

```
   Dog dog;
   Cat cat;
//--- 祖先から子孫にコピーすることができる（Animal ==> Dog）
   dog=some_animal;
   dog.swim=true;     // 犬は泳げる
//--- 子構造のオブジェクトをコピーすることはできない（Dog != Cat）
   //cat=dog;          // コンパイラはここでエラーを返す
//--- したがって、要素を1つずつコピーすることのみが可能
   cat.head=dog.head;
   cat.legs=dog.legs;
   cat.wings=dog.wings;
   cat.tail=dog.tail;
   cat.fly=dog.fly;
   cat.swim=false;    // 猫は泳げない
//--- 子孫から祖先に値をコピーすることは可能
   Animal elephant;
   elephant=cat;
   elephant.run=false;// ゾウは走れない
   elephant.swim=true;// ゾウは泳げる
//--- 配列を作成する
   Animal animals[4];
   animals[0]=some_animal;
   animals[1]=dog;
   animals[2]=cat;
   animals[3]=elephant;
//--- プリントする
   ArrayPrint(animals);
//--- 実行結果
/*
        [head] [legs] [wings] [tail] [fly] [swim] [run]
   [0]      1      4        0   true false  false  true
   [1]      1      4        0   true false   true  true
   [2]      1      4        0   true false  false false
   [3]      1      4        0   true false   true false
*/
  }
```

基本データ型をコピーするもう一つの方法は、共用体を使用することです。構造体のオブジェクトは、同じ共用体のメンバーでなければなりません。unionの例を参照してください。

構造体メンバへのアクセス

構造体の名称が新しいデータ型になり、この型の変数を宣言することが出来るようになります。構造体は、プロジェクト内で一度だけ宣言することが出来ます。構造体のメンバは点演算（ . ）を使用してアクセスされます。

例：

```
struct trade_settings
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   double take;          // 利益固定価格
   double stop;          // プロテクティブストップの値
   uchar  slippage;      // 使用可能のスリッページの値
  };
//--- trade_settings 型の変数を作成して初期化する
trade_settings my_set={0.0,0.0,5};
if (input_TP>0) my_set.take=input_TP;
```

'pack' は構造体およびクラスのフィールドを位置合わせします。

特別の pack 属性によって、構造体およびクラスのフィールドを位置合わせすることができます。

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

ここで n は 1、2、4、8、16 のいずれかです。指定は任意です。

例:

```
   struct pack(sizeof(long)) MyStruct
     {
      // 構造体のメンバは 8 バイト境界で位置合わせされている
     };
```

または

```
   struct MyStruct pack(sizeof(long))
     {
      // 構造体のメンバは 8 バイト境界で位置合わせされている
     };
```

'pack(1)' はデフォルトで構造体に適用されています。これは、構造体のメンバがメモリ内で次々に配置され、構造体サイズがメンバのサイズの合計に等しいことを意味します。

例:

```
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 位置合わせなしの基本構造体
   struct Simple_Structure
     {
      char              c; // sizeof(char)=1
      short             s; // sizeof(short)=2
      int               i; // sizeof(int)=4
      double            d; // sizeof(double)=8
     };
   //--- 基本構造体型のインスタンスを宣言する
   Simple_Structure s;
//--- それぞれの構造体メンバのサイズを表示する
   Print("sizeof(s.c)=",sizeof(s.c));
   Print("sizeof(s.s)=",sizeof(s.s));
```

© 2000-2025, MetaQuotes Ltd.

```
   Print("sizeof(s.i)=",sizeof(s.i));
   Print("sizeof(s.d)=",sizeof(s.d));
//--- POD構造体のサイズがそのメンバのサイズの合計と等しいことを確認する
   Print("sizeof(simple_structure)=",sizeof(simple_structure));
/*
  結果:
   sizeof(s.c)=1
   sizeof(s.s)=2
   sizeof(s.i)=4
   sizeof(s.d)=8
   sizeof(simple_structure)=15
*/
  }
```

構造体フィールドの位置合わせは、そのような位置合わせが適用されているサードパーティライブラリ(*.DLL)とデータを交換するときに必要になることがあります。

位置合わせがどのように機能するかを示すためにいくつかの例を使用しましょう。位置合わせなしの 4 つのメンバからなる構造体を適用します。

```
//--- 位置合わせなしの基本構造体
   struct Simple_Structure pack() // サイズが指定されていない、1 バイト境界で位置合わせ
     {
      char              c; // sizeof(char)=1
      short             s; // sizeof(short)=2
      int               i; // sizeof(int)=4
      double            d; // sizeof(double)=8
     };
//--- 基本構造体型のインスタンスを宣言する
   Simple_Structure s;
```

構造体フィールドは、宣言の順序と型サイズに従って、メモリー内に次々に配置されます。構造体のサイズは 15 ですが、配列内の構造体フィールドへのオフセットは未定義です。



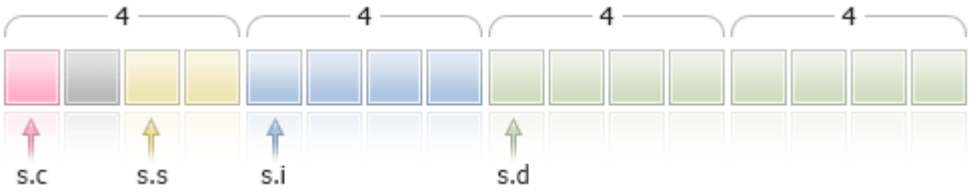
4 バイトの位置合わせで同じ構造体を宣言し、コードを実行します。

```
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 4 バイトで位置合わせされた基本構造体
   struct Simple_Structure pack(4)
     {
      char              c; // sizeof(char)=1
      short             s; // sizeof(short)=2
```

© 2000-2025, MetaQuotes Ltd.

```
      int               i; // sizeof(int)=4
      double            d; // sizeof(double)=8
     };
   //--- 基本構造体型のインスタンスを宣言する
   Simple_Structure s;
//--- それぞれの構造体メンバのサイズを表示する
   Print("sizeof(s.c)=",sizeof(s.c));
   Print("sizeof(s.s)=",sizeof(s.s));
   Print("sizeof(s.i)=",sizeof(s.i));
   Print("sizeof(s.d)=",sizeof(s.d));
//--- POD構造体のサイズがそのメンバのサイズの合計と等しいことを確認する
   Print("sizeof(simple_structure)=",sizeof(simple_structure));
/*
  結果:
   sizeof(s.c)=1
   sizeof(s.s)=2
   sizeof(s.i)=4
   sizeof(s.d)=8
   sizeof(simple_structure)=16 // 構造体サイズが変化した
*/
  }
```

構造体のサイズが変更され、4 バイト以上のすべてのメンバの構造体の先頭からのオフセットが 4 バイトの倍数になりました。より小さいメンバは、自身のサイズ境界に合わせて配置されます（たとえば、 'short'の場合は 2）。これがその外観です（追加されたバイトは灰色で表示されています）。



この場合、s.cメンバの後に 1 バイトが追加されるため、s.s（sizeof(short)==2）フィールドの境界が 2 バイトになります（'short' 型の位置合わせ）。

配列内の構造体の先頭へのオフセットも 4 バイト境界に揃えられます。つまり、a[0]、a[1]...a[n]要素のアドレスは、Simple_Structure arr[]では 4 バイトの倍数になります。

同様の型からなって 4 バイトで位置合わせされているがメンバの順序が異なる構造体をもう 2 つ考えてみましょう。最初の構造体では、メンバは型サイズの昇順で配置されています。

```
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 4 バイト境界で位置合わせされた基本構造体
   struct CharShortInt pack(4)
     {
      char              c; // sizeof(char)=1
```

© 2000-2025, MetaQuotes Ltd.

```
      short             s; // sizeof(short)=2
      int               i; // sizeof(double)=4
     };
//--- 基本構造体型のインスタンスを宣言する
   CharShortInt ch_sh_in;
//--- それぞれの構造体メンバのサイズを表示する
   Print("sizeof(ch_sh_in.c)=",sizeof(ch_sh_in.c));
   Print("sizeof(ch_sh_in.s)=",sizeof(ch_sh_in.s));
   Print("sizeof(ch_sh_in.i)=",sizeof(ch_sh_in.i));

//--- POD構造体のサイズがそのメンバのサイズの合計と等しいことを確認する
   Print("sizeof(CharShortInt)=",sizeof(CharShortInt));
/*
  結果:
   sizeof(ch_sh_in.c)=1
   sizeof(ch_sh_in.s)=2
   sizeof(ch_sh_in.i)=4
   sizeof(CharShortInt)=8
*/
  }
```

ご覧のとおり、構造体サイズは 8 で、2 つの 4 バイトブロックで構成されています。最初のブロックには 'char' 型と 'short' 型のフィールドが含まれ、2 番目のブロックには 'int' 型のフィールドが含まれています。



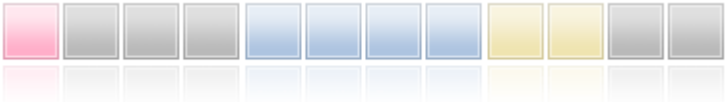
ここで1番目の構造体を2番目の構造体に変換します。これらで違うのはフィールドの順番だけです。'short' 型メンバを最後に移動します。

```
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 4 バイト境界で位置合わせされた基本構造体
   struct CharIntShort pack(4)
     {
      char              c; // sizeof(char)=1
      int               i; // sizeof(double)=4
      short             s; // sizeof(short)=2
     };
//--- 基本構造体型のインスタンスを宣言する
   CharIntShort ch_in_sh;
//--- それぞれの構造体メンバのサイズを表示する
   Print("sizeof(ch_in_sh.c)=",sizeof(ch_in_sh.c));
   Print("sizeof(ch_in_sh.i)=",sizeof(ch_in_sh.i));
   Print("sizeof(ch_in_sh.s)=",sizeof(ch_in_sh.s));
//--- POD構造体のサイズがそのメンバのサイズの合計と等しいことを確認する
```

© 2000–2025, MetaQuotes Ltd.

```
   Print("sizeof(CharIntShort)=",sizeof(CharIntShort));
/*
  結果:
   sizeof(ch_in_sh.c)=1
   sizeof(ch_in_sh.i)=4
   sizeof(ch_in_sh.s)=2
   sizeof(CharIntShort)=12
*/
  }
```

構造体の内容は変更されていませんが、メンバの順序を変更するとサイズが大きくなります。



位置合わせは継承によっても考慮される必要があります。単一の 'char' 型メンバを持つ単純な Parent 構造体を使ってこれを実証しましょう。位置合わせなしの構造体のサイズは 1 です。

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

'short' (sizeof(short)=2) 型メンバを持った Children 子クラスを作成しましょう。

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

その結果、位置合わせを 2 バイトに設定した場合、そのメンバのサイズは 3 ですが、構造体のサイズは 4 になります。この例では、2 バイトが Parent クラスに割り当てられるため、子クラスの 'short' フィールドへのアクセスは 2 バイトに揃えられます。

MQL5アプリケーションがファイルまたはストリームレベルで書き込み/読み取りを行うことによってサードパーティのデータとやり取りする場合は、構造体メンバにメモリがどのように割り当てられるかについての知識が必要です。

標準ライブラリの MQL5¥Include¥WinAPI ディレクトリには、WinAPI 関数を使用するための関数が含まれています。これらの関数は、WinAPI での作業に必要な場合に、指定された位置合わせを構造体に適用します。

offsetof は pack 属性に直接関連した特別のコマンドで、構造体の先頭からのオフセットを取得するのに使用されます。

```
//--- Children 型変数を宣言する
   Children child;
//--- 構造体の先頭からのオフセットを検知する
   Print("offsetof(Children,c)=",offsetof(Children,c));
   Print("offsetof(Children,s)=",offsetof(Children,s));
/*
```

© 2000-2025, MetaQuotes Ltd.

```
   結果:
    offsetof(Children,c)=0
    offsetof(Children,s)=2
*/
```

final修飾子

final修飾子は、構造体の宣言時にそこからのさらなる継承を禁止します。構造体がそこへ更なる変更を加える必要がないものである、またはセキュリティ上の理由で変更が許可されていない場合は、構造体をfinal修飾子で宣言してください。また、全ての構造体のメンバーも暗黙的にfinalとみなされます。

```
struct settings final
  {
  //--- тело структуры
  };

struct trade_settings : public settings
  {
  //--- 構造体
  };
```

final修飾子を使用した構造体からの継承を試行すると、上記の例で示したように、コンパイルはエラーを出します。

```
cannot inherit from 'settings' as it has been declared as 'final'
see declaration of 'settings'
```

クラス

クラスと構造体の異なる点は次の通りです。

- クラスの宣言には class キーワードが使用されます。
- クラスメンバは、他の値が示されていない限り、デフォルトで private アクセス指定子を持っています。構造体のデータメンバは、他の値が示されていない限り、デフォルトで public アクセスを持っています。
- クラスオブジェクトは、クラス内で仮想関数が宣言されない場合でも常に仮想関数の表を持っています。構造体は仮想関数を持つことが出来ません。
- new 演算子はクラスオブジェクトに適用することが出来ますが、構造体に適用することは出来ません。
- クラスは他のクラスからのみ継承でき、構造体は他の構造体のみから継承出来ます。

クラスと構造体は、明示的なコンストラクタとデストラクタを持つことが出来ます。コンストラクタが明示的に定義されている場合は、初期化シーケンスを使用しての構造体またはクラス変数の初期化は不可能です。

例:

```
struct trade_settings
  {
   double take;          // 利益固定価格
   double stop;          // プロテクティブストップの値
   uchar  slippage;      // 使用可能のスリッページの値
   //--- コンストラクタ
          trade_settings() { take=0.0; stop=0.0; slippage=5; }
   //--- デストラクタ
         ~trade_settings() { Print("This is the end"); }
```

© 2000-2025, MetaQuotes Ltd.

```
  };
//--- コンパイラが初期化不可能のエラーメッセージを生成する
trade_settings my_set={0.0,0.0,5};
```

コンストラクタとデストラクタ

コンストラクタは、構造体またはクラスのオブジェクトを作成する時に自動的に呼び出され、通常クラスメンバの初期化に使用されます。ここからはクラスのみに参照しますが、特に指定がない限り、同じことが構造体にも適用されます。コンストラクタ名は、クラス名と一致する必要があります。コンストラクタには戻り値の型がありません（void 型を指定出来ます）。

定義されたクラスメンバのうち文字列、動的配列及び初期化を必要とするオブジェクトはコンストラクタのあるなしに関係なく初期化されます。

各クラスには、パラメータの数と初期化リストにより異なる複数のコンストラクタが存在出来ます。パラメータの指定が必要なコンストラクタは、パラメトリックコンストラクタと呼ばれています。

パラメータのないコンストラクタは、デフォルトコンストラクタと呼ばれます。クラスでコンストラクタが宣言されていない場合、コンパイラはコンパイル時にデフォルトコンストラクタを作成します。

```
//+------------------------------------------------------------------+
//|  日付を操作するクラス                                            |
//+------------------------------------------------------------------+
class MyDateClass
  {
private:
   int               m_year;           // 年
   int               m_month;          // 月
   int               m_day;            // 日
   int               m_hour;           // 時間
   int               m_minute;         // 分
   int               m_second;         // 秒
public:
   //--- デフォルトコンストラクタ
                     MyDateClass(void);
   //--- パラメトリックコンストラクタ
                     MyDateClass(int h,int m,int s);
  };
```

コンストラクタは、クラス記述の中で宣言することができ、その本体も定義することが出来ます。例えば、MyDateClass の 2 つのコンストラクタは、次のように定義することが出来ます。

```
//+------------------------------------------------------------------+
//|  デフォルトコンストラクタ                                        |
//+------------------------------------------------------------------+
MyDateClass::MyDateClass(void)
  {
//---
   MqlDateTime mdt;
   datetime t=TimeCurrent(mdt);
   m_year=mdt.year;
```

© 2000–2025, MetaQuotes Ltd.

```
   m_month=mdt.mon;
   m_day=mdt.day;
   m_hour=mdt.hour;
   m_minute=mdt.min;
   m_second=mdt.sec;
   Print(__FUNCTION__);
  }
//+------------------------------------------------------------------+
//|  パラメトリックコンストラクタ                                    |
//+------------------------------------------------------------------+
MyDateClass::MyDateClass(int h,int m,int s)
  {
   MqlDateTime mdt;
   datetime t=TimeCurrent(mdt);
   m_year=mdt.year;
   m_month=mdt.mon;
   m_day=mdt.day;
   m_hour=h;
   m_minute=m;
   m_second=s;
   Print(__FUNCTION__);
  }
```

デフォルトコンストラクタ内では、クラスの全メンバがTimeCurrent()関数で書き入れられます。パラメトリックコンストラクタ内ではhour値がのみが書き入れられます。クラスの他メンバ(m_year、m_monthとm_day)は、自動的に現在の日付で初期化されます。

デフォルトコンストラクタは、クラスのオブジェクトの配列の初期化に特別な目的を持っています。全てのパラメータに初期値があるコンストラクタはデフォルトコンストラクタではありません。下記が例です。

```
//+------------------------------------------------------------------+
//|  デフォルトコンストラクタを持つクラス                            |
//+------------------------------------------------------------------+
class CFoo
  {
   datetime          m_call_time;     // 最後のオブジェクト呼び出し時
public:
   //--- パラメータに初期値があるコンストラクタはデフォルトコンストラクタではない
                     CFoo(const datetime t=0){m_call_time=t;};
   //--- コピーコンストラクタ
                     CFoo(const CFoo &foo){m_call_time=foo.m_call_time;};

   string ToString(){return(TimeToString(m_call_time,TIME_DATE|TIME_SECONDS));};
  };
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                              |
//+------------------------------------------------------------------+
void OnStart()
  {
// CFoo foo;  // デフォルトコンストラクタが設定されていないので使用出来ない
```

© 2000-2025, MetaQuotes Ltd.

```
//--- CFoo オブジェクト作成に可能な選択肢
   CFoo foo1(TimeCurrent());      // パラメトリックコンストラクタの明示的呼び出し
   CFoo foo2();                   // デフォルトパラメータを使用したパラメトリックコンストラクタの明示的呼び出し
   CFoo foo3=D'2009.09.09';       // パラメトリックコンストラクタの暗黙的呼び出し
   CFoo foo40(foo1);              // コピーコンストラクタの明示的呼び出し
   CFoo foo41=foo1;               // コピーコンストラクタの暗黙的呼び出し
   CFoo foo5;                     // デフォルトコンストラクタの明示的呼び出し(デフォルトコンストラクタのない場合は
                                  // パラメトリックコンストラクタが初期値で呼ばれる)
//--- CFoo ポインタの受け取りに可能な選択肢
   CFoo *pfoo6=new CFoo();        // 動的オブジェクトの作成とポインタの受け取り
   CFoo *pfoo7=new CFoo(TimeCurrent());// 動的オブジェクト作成のもう 1 つの選択肢
   CFoo *pfoo8=GetPointer(foo1); // pfoo8 が foo1 オブジェクトに参照する
   CFoo *pfoo9=pfoo7;             // pfoo9 と pfoo7 が同じオブジェクトに参照
   // CFoo foo_array[3];          // デフォルトコンストラクタが指定されていないので使用出来ない
//--- m_call_time の値を表示
   Print("foo1.m_call_time=",foo1.ToString());
   Print("foo2.m_call_time=",foo2.ToString());
   Print("foo3.m_call_time=",foo3.ToString());
   Print("foo4.m_call_time=",foo4.ToString());
   Print("foo5.m_call_time=",foo5.ToString());
   Print("pfoo6.m_call_time=",pfoo6.ToString());
   Print("pfoo7.m_call_time=",pfoo7.ToString());
   Print("pfoo8.m_call_time=",pfoo8.ToString());
   Print("pfoo9.m_call_time=",pfoo9.ToString());
//--- 動的に作成した配列を削除する
   delete pfoo6;
   delete pfoo7;
   //delete pfoo8;  // pfoo8 は自動的に作成されたオブジェクトのfoo1を指すので、明示的に削除する必要がない
   //delete pfoo9;  // pfoo9 は pfoo7 と同じオブジェクトを参照するため明示的に pfoo9 を削除する必要はない
  }
```

下記のコメントが解除された場合

```
   //CFoo foo_array[3];      // デフォルトコンストラクタが設定されていないので使用出来ない
```

または

```
   //CFoo foo_dyn_array[];   // デフォルトコンストラクタが設定されていないので使用出来ない
```

コンパイラは「default constructor is not defined(デフォルトコンストラクタが定義されていない)」とのエラーメッセージを生成します。

クラスに、ユーザ定義のコンストラクタがある場合、コンパイラはデフォルトコンストラクタを生成しません。つまり、クラス内でパラメトリックコンストラクタが宣言されていてもデフォルトコンストラクタが宣言されていない場合は、このクラスのオブジェクトの配列を宣言することは出来ません。コンパイラは、このスクリプトに対してエラーを返します。

```
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|  デフォルトコンストラクタを持たないクラス                                  |
//+------------------------------------------------------------------+
class CFoo
  {
   string            m_name;
public:
                     CFoo(string name) { m_name=name;}
  };
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- コンパイル時に「default constructor is not defined(デフォルトコンストラクタが定義されていない)」エラーが出る
   CFoo badFoo[5];
  }
```

この例では、CFoo クラスにはパラメトリックコンストラクタが宣言されています。このような場合には、コンパイラは、コンパイル時に自動的にデフォルトコンストラクタを作成しません。同時に、オブジェクトの配列を宣言する時はオブジェクトは全て自動的に作成されて初期化されていることが想定されます。オブジェクトの自動初期化中にはデフォルトコンストラクタを呼び出す必要がありますが、デフォルトのコンストラクタが明示的に宣言されず自動的にコンパイラによって生成されていないため、このようなオブジェクトを作成することは不可能です。このため、コンパイル段階でコンパイラエラーが発生します。

コンストラクタを使用してオブジェクトを初期化するためには特別な構文があります。構造体やクラスのメンバのコンストラクタの初期化子(初期化のための特別な構造)は初期化リストで指定することが出来ます。

初期化リストは、コンストラクタのパラメータリストの後のコロンの後及び本体に先行するコンマで区切られた初期化子のリストです(左中括弧の前に配置されます)。いくつかの要件があります。

- 初期化リストは、コンストラクタ内でのみ使用することが出来ます。
- 親メンバは初期化リストで初期化することは出来ません。
- 初期化リストは関数の定義(実装)で続けられる必要があります。

下記はクラスのメンバを初期化するためのコンストラクタの例です。

```
//+------------------------------------------------------------------+
//|  文字名を格納するためのクラス                                          |
//+------------------------------------------------------------------+
class CPerson
  {
   string            m_first_name;     // 名
   string            m_second_name;    // 姓
public:
   //--- 空のデフォルトコンストラクタ
                     CPerson() {Print(__FUNCTION__);};
   //--- パラメトリックコンストラクタ
                     CPerson(string full_name);
   //--- 初期化リストを持ったコンストラクタ
```

© 2000-2025, MetaQuotes Ltd.

```
                     CPerson(string surname,string name): m_second_name(surname),
m_first_name(name) {};
   void PrintName(){PrintFormat("Name=%s Surname=%s",m_first_name,m_second_name);};
  };
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
CPerson::CPerson(string full_name)
  {
   int pos=StringFind(full_name," ");
   if(pos>=0)
     {
      m_first_name=StringSubstr(full_name,0,pos);
      m_second_name=StringSubstr(full_name,pos+1);
     }
  }
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 「default constructor is not defined(デフォルトコンストラクタが定義されていない)」エラーを取得
   CPerson people[5];
   CPerson Tom="Tom Sawyer";                          // トムソーヤ
   CPerson Huck("Huckleberry","Finn");                // ハックルベリー フィンの冒険
   CPerson *Pooh = new CPerson("Winnie","Pooh");  // くまのプーさん
   //--- 出力値
   Tom.PrintName();
   Huck.PrintName();
   Pooh.PrintName();

   //--- 動的に作成されたオブジェクトを削除する
   delete Pooh;
  }
```

この場合、CPersonクラスには、次の 3 つのコンストラクタがあります。

1. クラスオブジェクトの配列を作成出来る明示的なデフォルトコンストラクタ
2. 1 つのパラメータとして姓名を取得し、見つかったスペースに応じて名に分割するコンストラクタ
3. 初期化リストを含む 2 つのパラメータを持ったコンストラクタイニシャライザ - m_second_name(surname) と m_first_name(name)]

リストを使用した初期化が代入を置き換えていることにご注意ください。個々のメンバは次のように初期化する必要があります。

クラスメンバ(式のリスト)

初期化リストでは、メンバは任意の順序を持てますが、その順を従って初期化されます。つまり、3 番目のコンストラクタでは、最初に指定された m_first_name メンバが最初に初期化され、m_second_name がその

© 2000-2025, MetaQuotes Ltd.

後に初期化されます。クラスメンバの初期化が他のクラスメンバの値に依存する場合にはこのことが考慮されるべきです。

デフォルトコンストラクタが基本クラスで宣言されていないと同時にパラメータを持つ 1 つ以上のコンストラクタが宣言されている場合は、初期化リスト内の基本クラスのコンストラクタのいずれかが呼び出されるべきです。それは、初期化リスト内のどこにあるかに関係なく、リストの通常のメンバとしてコンマを通過し、オブジェクトの初期化時に最初に呼び出されます。

```
//+------------------------------------------------------------------+
//|  基本クラス                                                      |
//+------------------------------------------------------------------+
class CFoo
  {
   string            m_name;
public:
   //--- 初期化リストを持ったコンストラクタ
                     CFoo(string name) : m_name(name) { Print(m_name);}
  };
//+------------------------------------------------------------------+
//|  CFoo の派生クラス                                               |
//+------------------------------------------------------------------+
class CBar : CFoo
  {
   CFoo              m_member;       // クラスメンバが親のオブジェクト
public:
   //--- 初期化リスト内のデフォルトコンストラクタが、親のコンストラクタを呼び出す
                     CBar(): m_member(_Symbol), CFoo("CBAR") {Print(__FUNCTION__);}
  };
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                              |
//+------------------------------------------------------------------+
void OnStart()
  {
   CBar bar;
  }
```

この例では、bar オブジェクトを作成する時に、デフォルトコンストラクタ CBar() が呼び出されます。その中で最初に親 CFoo のコンストラクタが呼び出され、その後 m_member クラスメンバのコンストラクタが呼び出されます。

デストラクタは、クラスのオブジェクトが破壊された時に自動的に呼び出される特殊な関数です。デストラクタ名はクラス名とチルダ(~)として記述されています。初期化解除とする文字列、動的配列やオブジェクトは、デストラクタの有無に関係なく初期化解除されます。デストラクタがある場合、これらのアクションは、デストラクタを呼び出した後に実行されます。

デストラクタは virtual キーワードと宣言されたかどうかにかかわらず常に virtual(仮想)です。

クラスメソッドの定義

クラス関数メソッドは、クラスの内部及びクラス宣言の外側の両方で定義することが出来ます。メソッドがクラス内で定義されている場合は、その本体はメソッドの宣言の後に来ます。

© 2000-2025, MetaQuotes Ltd.

例：

```
class CTetrisShape
  {
protected:
   int               m_type;
   int               m_xpos;
   int               m_ypos;
   int               m_xsize;
   int               m_ysize;
   int               m_prev_turn;
   int               m_turn;
   int               m_right_border;
public:
   void              CTetrisShape();
   void              SetRightBorder(int border) { m_right_border=border; }
   void              SetYPos(int ypos)          { m_ypos=ypos;           }
   void              SetXPos(int xpos)          { m_xpos=xpos;           }
   int               GetYPos()                  { return(m_ypos);        }
   int               GetXPos()                  { return(m_xpos);        }
   int               GetYSize()                 { return(m_ysize);       }
   int               GetXSize()                 { return(m_xsize);       }
   int               GetType()                  { return(m_type);        }
   void              Left()                     { m_xpos-=SHAPE_SIZE;    }
   void              Right()                    { m_xpos+=SHAPE_SIZE;    }
   void              Rotate()                   { m_prev_turn=m_turn; if(++m_turn>3)
m_turn=0; }
   virtual void      Draw()                     { return;                }
   virtual bool      CheckDown(int& pad_array[]);
   virtual bool      CheckLeft(int& side_row[]);
   virtual bool      CheckRight(int& side_row[]);
  };
```

SetRightBorder(int border) から Draw() までの関数は CTetrisShape クラス内部で直接定義及び宣言されています。

CTetrisShape() コンストラクタと CheckDown(int& pad_array[])、CheckLeft(int& side_row[]) 及び CheckRight(int& side_row[]) メソッドはクラス内で宣言されただけでまだ定義されていません。これらの関数の定義は、コードの後に出てきます。クラス外にメソッドを定義するためには[スコープ解決演算子](#)が使用され、クラス名がスコープとして利用されます。

例：

```
//+------------------------------------------------------------------+
//|  基本クラスのコンストラクタ                                      |
//+------------------------------------------------------------------+
void CTetrisShape::CTetrisShape()
  {
   m_type=0;
   m_ypos=0;
```

© 2000-2025, MetaQuotes Ltd.

```
   m_xpos=0;
   m_xsize=SHAPE_SIZE;
   m_ysize=SHAPE_SIZE;
   m_prev_turn=0;
   m_turn=0;
   m_right_border=0;
  }
//+------------------------------------------------------------------+
//| （スティックとキューブの）下に移動する能力を確認する                                  |
//+------------------------------------------------------------------+
bool CTetrisShape::CheckDown(int& pad_array[])
  {
   int i,xsize=m_xsize/SHAPE_SIZE;
//---
   for(i=0; i<xsize; i++)
     {
      if(m_ypos+m_ysize>=pad_array[i]) return(false);
     }
//---
   return(true);
  }
```

Public、Protected 及び Private アクセス修飾子

新しいクラスを開発する場合には、外部からのメンバへのアクセスを制限することをお勧めします。このために、キーワード private もしくは protected が使用されます。この場合には、隠されたデータは、同じクラスの関数メソッドからのみアクセスすることが出来ます。もし *protected* キーワードが使用されれば、隠されたデータは承継クラスのメソッドからもアクセス出来ます。同様の方法はクラスの関数メソッドへのアクセスを制限するために使用出来ます。

メンバ及び/またはクラスメソッドへのアクセスを完全にオープンにする必要がある場合は public キーワードが使用されるべきです。

例:

```
class CTetrisField
  {
private:
   int               m_score;                              // 得点
   int               m_ypos;                               // 図形の現在位置
   int               m_field[FIELD_HEIGHT][FIELD_WIDTH]; // ウェルの行列
   int               m_rows[FIELD_HEIGHT];                 // ウェルの行の番号付け
   int               m_last_row;                           // 最後の空いている行
   CTetrisShape     *m_shape;                              // テトリスの図形
   bool              m_bover;                              // ゲームオーバー
public:
   void              CTetrisField() { m_shape=NULL; m_bover=false; }
   void              Init();
   void              Deinit();
   void              Down();
```

© 2000–2025, MetaQuotes Ltd.

```
   void              Left();
   void              Right();
   void              Rotate();
   void              Drop();
private:
   void              NewShape();
   void              CheckAndDeleteRows();
   void              LabelOver();
  };
```

public: 指定子の後（と次のアクセス指定子の前）に宣言されたクラスメンバとメソッドはプログラムによってクラスオブジェクトへの参照として利用出来ます。この例では、CTetrisField()、Init()、Deinit()、Down()、Left()、Right()、Rotate() 及び Drop() 関数がそれらのメンバです。

private: 指定子の後（と次のアクセス指定子の前）に宣言されたクラスメンバはこのクラスのメンバ関数によってのみ使用出来ます。要素へのアクセスの指定子は、必ずコロン（ : ）で終了し、クラス定義に何回でも使用することが出来ます。

protected: 指定子（と次のアクセス指定子の前）に宣言されたクラスメンバはこのクラスのメンバ関数とクラスの子孫のメンバ関数によってのみ使用出来ます。外部からprivateやprotected指定子を備えたメンバを参照しようとすると、コンパイル段階エラーが発生します。例：

```
class A
  {
protected:
   //--- the copy operator is available only inside class A and its descendants
   void operator=(const A &)
     {
     }
  };
class B
  {
   //--- class A object declared
   A                 a;
  };
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   //--- B型変数を2つ宣言する
   B b1, b2;
   //--- オブジェクトを別のオブジェクトにコピーしようとする
   b2=b1;
  }
```

コードをコンパイルすると、リモートコピー演算子の呼び出し試行についてのエラーメッセージが発生します。

```
attempting to reference deleted function 'void B::operator=(const B&)'   trash3.mq5
 32   6
```

© 2000-2025, MetaQuotes Ltd.

次の2番目の文字列は、より詳細な説明を提供します。クラスAの使用できないコピー演算子が呼び出されるため、クラスBのコピー演算子は明示的に削除されました。

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

基本クラスのメンバへのアクセスは、派生クラスの承継中に再定義することが出来ます。

「delete」指定子

delete指定子は、使用できないクラスメンバ関数をマークします。つまり、プログラムがこのような関数を明示的または暗黙的に参照する場合、コンパイル段階ですでにエラーが発生します。たとえば、この指定子を使用すると、子クラスで親メソッドを使用できないようにすることができます。親クラスのprivate領域で関数を宣言すると、同じ結果が得られます（privateセクションでの宣言）。ここでは、deleteを使用すると、子孫のレベルでコードがより読みやすく、管理しやすくなります。

```
class A
  {
public:
                     A(void) {value=5;};
   double            GetValue(void) {return(value);}
private:
   double            value;
  };
class B: public A
  {
   double            GetValue(void)=delete;
  };
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- A型の変数を宣言する
   A a;
   Print("a.GetValue()=", a.GetValue());
//--- B型変数から値を取得しようとする
   B b;
   Print("b.GetValue()=", b.GetValue()); // 個の文字列でコンパイラがエラーを表示する
  }
```

以下がコンパイラメッセージです。

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

The 'delete' specifier allows disabling auto casting or the copy constructor, which otherwise would have to be hidden in the private section as well. 例:

```
class A
  {
public:
```

© 2000-2025, MetaQuotes Ltd.

```
   void              SetValue(double v) {value=v;}
   //--- int 型の呼び出しを無効にする
   void              SetValue(int) = delete;
   //--- コピー演算子を無効にする
   void              operator=(const A&) = delete;
private:
   double            value;
  };
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- A型の変数を2つ宣言する
   A a1, a2;
   a1.SetValue(3);       // error!
   a1.SetValue(3.14);    // OK
   a2=a1;                // エラー
  }
```

コンパイル中に、エラーメッセージが発生します：

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

final修飾子

final修飾子は、クラスの宣言時にそこからのさらなる継承を禁止します。クラスのインターフェイスがそこへ更なる変更を加える必要がないものである、またはセキュリティ上の理由で変更が許可されていない場合は、クラスをfinal修飾子で宣言してください。また、全てのクラスのメソッドも暗黙的にfinalとみなされます。

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

class CBar : public CFoo
  {
  //--- クラス本体
  };
```

final修飾子を使用したクラスからの継承を試行すると、上記の例で示したように、コンパイルはエラーを出します。

```
cannot inherit from 'CFoo' as it has been declared as 'final'
see declaration of 'CFoo'
```

共同体（union）

© 2000-2025, MetaQuotes Ltd.

共同体は、同じメモリ領域を共有する複数の変数からなる特別なデータ型です。したがって、共用体は、同じビットシーケンスを 2 つ(またはそれ以上)の異なる方法で解釈する機能を提供します。共同体の宣言は structure の宣言と似ていて、union キーワードから始まります。

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

構造体とは異なり、さまざまな共用体メンバは同じメモリ領域に属します。この例では、LongDouble の共用体は、同じメモリ領域を共有する long および double 型値で宣言されます。構造体とは異なり、この共同体で *long* 整数値と *double* 実数値を同時に設定することは不可能です。これは、long_value 変数と double_value 変数は(メモリ内で)重複しているためです。一方、MQL5プログラムは、共用体に含まれるデータをいつでも整数値(long)または実数値(double)として処理することができます。そのため、共用体を使用すれば、同じデータシーケンスを表すために2つ(またはそれ以上)のオプションを受け取ることができます。

共用体が宣言されると、コンパイラは自動的にサイズが最大の型を格納するのに十分なメモリ領域を変数共用体に割り当てます。共用体要素にアクセスするためには、構造体の場合と同じ構文が使用されます。これはドット演算子です。

```
union LongDouble
{
  long   long_value;
  double double_value;
};
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   LongDouble lb;
//--- 無効な -nan(ind) 番号を取得して表示する
   lb.double_value=MathArcsin(2.0);
   printf("1.  double=%f                integer=%
I64X",lb.double_value,lb.long_value);
//--- 最大の正規値 (DBL_MAX)
   lb.long_value=0x7FEFFFFFFFFFFFFF;
   printf("2.  double=%.16e  integer=%I64X",lb.double_value,lb.long_value);
//--- 最小の正の正規値 (DBL_MIN)
   lb.long_value=0x0010000000000000;
   printf("3.  double=%.16e  integer=%.16I64X",lb.double_value,lb.long_value);
  }
/*  実行結果
    1.  double=-nan(ind)                integer=FFF8000000000000
    2.  double=1.7976931348623157e+308  integer=7FEFFFFFFFFFFFFF
    3.  double=2.2250738585072014e-308  integer=0010000000000000
*/
```

© 2000-2025, MetaQuotes Ltd.

共用体はプログラムが同じメモリデータを異なった方法で解釈することを可能にするので、普通でない型変換が必要なときに使われます。

その特性によって、共用体は継承に使用することも、static メンバを持つこともできません。他のすべての側面では、*共用体*は、すべてのメンバがゼロオフセットを持つ構造体のように動作します。次の型は、共用体のメンバになることはできません。

- 動的配列
- 文字列
- オブジェクトポインタおよび関数ポインタ
- クラスオブジェクト
- コンストラクタまたはデストラクタを持つ構造体オブジェクト
- ポイント1～5のメンバを持つ構造体オブジェクト

クラス同様、共用体はコンストラクタ、デストラクタ、メンバを持つことができます。デフォルトで、共用体メンバのアクセスタイプは public です。private 要素を作成するには、private キーワードを使用します。これらの可能性はすべて、ColorToARGB() 関数のように color 型を ARGB に変換する例で示すことができます。

```
//+------------------------------------------------------------------+
//|  color(BGR) を ARGB に変換する共用体                             |
//+------------------------------------------------------------------+
union ARGB
  {
   uchar             argb[4];
   color             clr;
   //--- コンストラクタ
                     ARGB(color col,uchar a=0){Color(col,a);};
                    ~ARGB(){};
   //--- public メソッド
public:
   uchar   Alpha(){return(argb[3]);};
   void    Alpha(const uchar alpha){argb[3]=alpha;};
   color   Color(){ return(color(clr));};
   //--- private メソッド
private:
   //+------------------------------------------------------------------+
   //|  アルファチャンネルの値と色を設定する                            |
   //+------------------------------------------------------------------+
   void    Color(color col,uchar alpha)
     {
      //--- clr メンバの色を設定する
      clr=col;
      //--- アルファコンポーネント(不透明度)の値を設定する
      argb[3]=alpha;
      //--- R(赤) および B(青) コンポーネントのバイトを交換する
      uchar t=argb[0];argb[0]=argb[2];argb[2]=t;
     };
  };
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|  スクリプトプログラムを開始する関数                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 0x55 は 55/255=21.6 % を意味する(0% - 完全に透明)
   uchar alpha=0x55;
//--- カラー型は 0x00BBGGRR で示される
   color test_color=clrDarkOrange;
//--- ARGB 共用体からのバイトの値はここで受け入れられる
   uchar argb[];
   PrintFormat("0x%.8X - here is how the 'color' type look like for %s, BGR=(%s)",
               test_color,ColorToString(test_color,true),ColorToString(test_color));
//--- ARGB 型は 0x00RRGGBB で示され、RR および BB コンポーネントが交換されている
   ARGB argb_color(test_color);
//--- バイト配列を複製する
   ArrayCopy(argb,argb_color.argb);
//--- ARGB 表現では次のようになる
   PrintFormat("0x%.8X - ARGB representation with the alpha channel=0x%.2x, ARGB=(%d,%d,%d,%d)",
               argb_color.clr,argb_color.Alpha(),argb[3],argb[2],argb[1],argb[0]);
//--- 不透明度を追加する
   argb_color.Alpha(alpha);
//--- ARGB を 'color' 型として定義してみる
   Print("ARGB как color=(",argb_color.clr,")  alpha channel=",argb_color.Alpha());
//--- バイト配列を複製する
   ArrayCopy(argb,argb_color.argb);
//--- ARGB 表現では次のようになる
   PrintFormat("0x%.8X - ARGB representation with the alpha channel=0x%.2x, ARGB=(%d,%d,%d,%d)",
               argb_color.clr,argb_color.Alpha(),argb[3],argb[2],argb[1],argb[0]);
//--- ColorToARGB() 関数の結果で確認する
   PrintFormat("0x%.8X - result of ColorToARGB(%s,0x%.2x)",ColorToARGB(test_color,alpha),
               ColorToString(test_color,true),alpha);
  }
/* 実行結果
   0x00008CFF - here is how the color type looks for clrDarkOrange, BGR=(255,140,0)
   0x00FF8C00 - ARGB representation with the alpha channel=0x00, ARGB=(0,255,140,0)
   ARGB as color=(0,140,255)  alpha channel=85
   0x55FF8C00 - ARGB representation with the alpha channel=0x55, ARGB=(85,255,140,0)
   0x55FF8C00 - result of ColorToARGB(clrDarkOrange,0x55)
*/
```

インターフェイス

インターフェイスは、クラスが今後実装することができる特定の機能を決定する為にあります。実際には、このクラスはメンバを含めたり、コンストラクタやデストラクタを持つことはできません。インターフェイス宣言された全てのメソッドは、明示的定義なしでも純粋仮想です。

© 2000-2025, MetaQuotes Ltd.

例の通り、interfaceというキーワードを使ってインターフェイスを定義します。

```
//--- アニマルを記述する為の基本的なインターフェイス
interface IAnimal
  {
//--- インターフェイスメソッドはデフォルトでpublicアクセスを持っています
   void Sound();  // アニマルが生成するサウンド
  };
//+------------------------------------------------------------------+
//|  CCatクラスはIAnimalインターフェイスから継承される                |
//+------------------------------------------------------------------+
class CCat : public IAnimal
  {
public:
                     CCat() { Print("Cat was born"); }
                    ~CCat() { Print("Cat is dead");  }
   //--- IAnimalインターフェイスのSoundメソッドを実装します
   void Sound(){ Print("meou"); }
  };
//+------------------------------------------------------------------+
//|  CDogクラスはIAnimalインターフェイスから継承されます             |
//+------------------------------------------------------------------+
class CDog : public IAnimal
  {
public:
                     CDog() { Print("Dog was born"); }
                    ~CDog() { Print("Dog is dead");  }
   //--- IAnimalインターフェイスのSoundメソッドを実装します
   void Sound(){ Print("guaf"); }
  };
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- IAnimal型のオブジェクトへのポインタ配列
   IAnimal *animals[2];
//--- IAnimalの子孫を生成し、配列にそれらへのポインタを保存します
   animals[0]=new CCat;
   animals[1]=new CDog;
//--- 各子孫にIAnimalベースインターフェイスのSound()メソッドを呼び出します
   for(int i=0;i<ArraySize(animals);++i)
      animals[i].Sound();
//--- オブジェクトを削除します
   for(int i=0;i<ArraySize(animals);++i)
      delete animals[i];
//--- 実行結果
/*
   Cat was born
```

© 2000-2025, MetaQuotes Ltd.

```
   Dog was born
   meou
   guaf
   Cat is dead
   Dog is dead
*/
  }
```

抽象クラスの場合のように、継承なしにインターフェイスのオブジェクトを作成することはできません。インターフェイスは他のインターフェイスからのみ継承することができ、クラスの祖先とすることもできます。また、インターフェイスは常に公共の可視性を持っています。

インターフェイスはクラス宣言または構造内で宣言することができませんが、インターフェイスへのポインタはvoid *型変数に格納することができます。一般的に、void *型変数にはあらゆるクラスのオブジェクトへのポインタを保存することができます。void *ポインタを特定のクラスのオブジェクトへのポインタに変換するには、dynamic_cast演算子を使用する必要があります。変換ができない場合には、dynamic_castの操作結果はNULLとなります。

参照

オブジェクト指向プログラミング

© 2000-2025, MetaQuotes Ltd.

動的配列オブジェクト

動的配列

最大 4 次元の配列の宣言が可能です。動的配列（最初の角括弧ペアで値が指定されていない配列）が宣言されると、コンパイラは自動的にこのような構造の変数（動的配列オブジェクト）を作成し、適切な初期化のためのコードを提供します。

動的配列は、宣言されたブロックの視認性の領域を越えた時に、自動的に解放されます。

例：

```
double matrix[][10][20]; // 3 次元動的配列
ArrayResize(matrix,5);   // 最初の次元のサイズを設定する
```

静的配列

大サイズの配列の次元が明示的に指定されている場合、コンパイラは必要なメモリサイズを事前に割り当てます。このような配列は、静的配列と呼ばれています。それにもかかわらず、コンパイラは事前に割り当てられた静的バッファ（配列を格納するためのメモリの一部）に関連付けられている動的配列（オブジェクト）のために追加のメモリを割り当てます。

動的配列オブジェクトを作成するのは、この静的配列を関数にパラメータとして渡すことが必要になる可能性があるからです。

例：

```
double stat_array[5]; // 1 次元静的配列
some_function(stat_array);
...
bool some_function(double& array[])
  {
   if(ArrayResize(array,100)<0) return(false);
   ...
   return(true);
  }
```

構造体内の配列

静的配列が構造体のメンバとして宣言されている場合には、動的配列オブジェクトは作成されません。これは、Windows API で使用されるデータ構造の互換性を確保するためです。

しかし、構造体のメンバとして宣言されている静的配列も MQL5 関数に渡されることが出来ます。この場合、パラメータを渡す時に、一時オブジェクトの動的配列が作成されます。この配列は、構造体のメンバである静的配列にリンクされています。

参照

配列関数、変数の初期化、変数のアクセス権スコープとライフタイム、オブジェクトの作成と削除

© 2000–2025, MetaQuotes Ltd.

行列とベクトル

タイプベクトルは、MQL5の特別なデータ型であり、ベクトル演算を可能にします。ベクトルは、 double型の1次元配列です。これは線形代数の基本的な概念の1つであり、物理学、幾何学など、多くの科学分野で使用されています。ベクトルは、3Dグラフィックスやその他のアプリケーション分野で、連立一次方程式を解くために使用されます。ベクトルは加算および乗算できます。ベクトル間の長さまたは距離は、ノルムを介して取得できます。プログラミングでは、ベクトルは通常、同種の要素の配列で表されます。これには、通常のベクトル演算がない(配列を追加または乗算できず、ノルムがない)場合があります。

行列を操作する場合、ベクトルは行ベクトルおよび文字列ベクトルとして表すことができます。また、線形代数ではベクトルの共変性と反変性の概念を使用します。ベクトル型の各オブジェクトが何であるかを決定するのはプログラマーだけなので、これらの概念はMQL5コードを書くときに何の違いももたらしません。たとえば、3Dグラフィックスの回転、変位、または圧縮ベクトルにすることができます。

一般的に言えば、線形代数の観点からは、数もベクトルですが、1次元のベクトル空間にあります。ベクトル自体は、行列の特殊なケースと見なすことができます。

タイプ行列は、あと1つのMQL5の特別なデータ型であり、行列を表します。行列は、実際にはdouble型の2次元配列です。ベクトルと行列は、特定のタイプのデータセットでの操作を容易にするためにMQL5に導入されました。それらを使用すると、開発者は、単純で数学のような形式で線形代数の可能性から利益を得ることができます。行列を使用すれば、線形方程式系または微分方程式系をコンパクトに記述できます。行列の行の数は方程式の数に対応し、列の数は未知数の数に等しくなります。その結果、線形方程式系は行列演算によって解くことができます。

行列には、次の代数演算が定義されています。

- 同じサイズの行列の追加
- 適切なサイズの行列の乗算：左側の行列の列の数は、右側の行列の行の数と等しくなければなりません
- 列ベクトルによる行列の乗算。行列乗算規則に従った行列による行ベクトルの乗算。この意味で、ベクトルは行列の特殊なケースです
- 数値(スカラー)による行列の乗算

数学では多くの異なる行列タイプを考慮します。単位行列、対称行列、歪対称行列、上三角行列と下三角行列、およびその他のタイプが例です。 行列理論では、さまざまな正規形が重要な役割を果たします。それらは、特定の変換によって取得できる、特定の標準形の行列を表します。実際には、安定性などの追加のプロパティを持つ正規形が使用されます。

ベクトルと行列、またはむしろ関連するタイプの特別なメソッドを使用すると、数学表記に近い、より単純かつ簡潔でより明確なコードの作成が可能になります。これらのメソッドを使用すると、ネストされたループを作成したり、計算で配列の正しいインデックスを作成したりする必要がなくなります。したがって、これらのメソッドを使用すると、複雑なプログラムの開発における信頼性と速度が向上します。

行列とベクトルのメソッドのリスト

タイプ行列およびベクトルには、関連するNumPyライブラリメソッドに対応するメソッドが含まれます。これらのメソッドを使用すると、最小限の労力でアルゴリズムとコードをPythonからMQL5に変換できます。多くのデータ処理タスク、数式、ニューラルネットワーク、機械学習タスクは、既製のPythonメソッドとライブラリを使用して解決できます。

© 2000-2025, MetaQuotes Ltd.

メソッド 行 列 /ベクト ル	NumPyの類 似 の 方 法	説 明
void matrix.Eye(const int rows, const int cols, const int ndiag=0)	eye	対 角 線 上 に1があって他 の場 所 に0が ある行 列 を作 成 します
void matrix.Identity(const int rows)	identity	主 対 角 線 上 に正 方 行 列 を作 成 しま す
void matrix.Ones(const int rows, const int cols)	ones	与 えられた行 と列 の新 しい行 列 を作 成 し、1で埋 めます
void matrix.Zeros(const int rows, const int cols)	zeros	ゼロで埋 められた、指 定 された行 と列 の新 しい行 列 を作 成 します
void matrix.Full(const int rows, const int cols, const scalar value)	full	スカラー値 で満 たされた、指 定 された 行 と列 の新 しい行 列 を作 成 します
void matrix.Copy(const matrix a)	copy	与 えられた行 列 のコピーを作 成 します
void matrix.FromBuffer(const int rows, const int cols, const scalar array[], const int count=-1, const int offset=0)	frombuffer	1次 元 配 列 から作 成 された行 列 を作 成 します
void matrix.FromFile(const int rows, condt int cols, const int file_handle, const int count=-1, const int offset=0)	fromfile	テキスト またはバイナリファイルのデータ から行 列 を作 成 します
void vector.FromString(const string source, const string sep=" ")	fromstring	文 字 列 内 のテキスト データから初 期 化 されたベクト ルを作 成 します
void vector.Arange(const scalar start, const scalar stop, const scalar step=1)	arange	指 定 された間 隔 内 で等 間 隔 の値 を 作 成 します
void matrix.Diag(const vector v, const int ndiag=0)	diag	対 角 線 を抽 出 するか、対 角 行 列 を 作 成 します
void matrix.Tri(const int rows, const int cols, const int ndiag=0)	tri	指 定 された対 角 線 以 下 に1があり、他 の場 所 に0がある行 列 を作 成 します
void matrix.Tril(const int rows, const int cols, const scalar array[], const int ndiag=0)	tril	k番 目 の対 角 線 がゼロより上 にある要 素 を含 む行 列 のコピーを返 します
void matrix.Triu(const int rows, const int cols, const scalar array[], const int ndiag=0)	triu	k番 目 の対 角 線 がゼロになっている下 の要 素 を含 む行 列 のコピーを返 します
void matrix.Vander(const vector v, const int cols=-1, const bool increasing=false)	vander	ヴァンデルモンド の行 列 を生 成 します

© 2000–2025, MetaQuotes Ltd.

メソッド 行列/ベクトル	NumPyの類似の方法	説明
vector matrix.Row(const unsigned nrow)		行ベクトルを返します
vector matrix.Col(const unsigned ncol)		列ベクトルを返します
unsigned matrix.Rows()		行列の行数を返します
unsigned matrix.Cols()		行列の列数を返します
void matrix.Init()		行列を初期化します
matrix matrix.Transpose()	transpose	行列の軸を反転または並べ替えます。変更された行列を返します
matrix matrix.Dot(const matrix b)	dot	2つの行列の内積
matrix matrix.Inner(const matrix b)	inner	2つの行列の内積
matrix matrix.Outer(const matrix b)	outer	2つの行列の外積を計算します
matrix matrix.MatMul(const matrix b)	matmul	2つの行列の行列積
matrix matrix.MatrixPower(const int power)	matrix_power	正方行列のn(整数)乗
matrix matrix.Kron(const matrix b)	kron	2つの行列のクロネッカー積を返します
bool matrix.Cholesky(matrix& L)	cholesky	コレスキー分解を返します
bool matrix.QR(matrix& Q, matrix& R)	qr	行列のQR分解を計算します
bool matrix.SVD(matrix& U, matrix& V, vector& singular_values)	svd	特異値分解
bool matrix.Eig(matrix& eigen_vectors, vector& eigen_values)	eig	正方行列の固有値と右固有ベクトルを計算します
bool matrix.EigH(matrix& eigen_vectors, vector& eigen_values)	eigh	エルミート行列の固有値と固有ベクトルを返します
bool matrix.EigVals(vector& eigen_values)	eigvals	一般的な行列の固有値を計算します
bool matrix.EigValsH(vector& eigen_values)	eigvalsh	エルミート行列の固有値を計算します
bool matrix.LU(matrix& L, matrix& U)		下三角行列と上三角行列の積としての行列のLU分解

© 2000–2025, MetaQuotes Ltd.

メソッド 行 列 /ベクト ル	NumPyの類 似 の 方 法	説 明
bool matrix.LUP(matrix& L, matrix& U, matrix& P)		部 分 的なピボットを使 用したLUP分 解 。これは、行 の順 列 のみを使 用した LU分 解 を指します。PA= LU
double matrix.Norm(const norm)	norm	行 列 またはベクト ルのノルムを返します
double matrix.Cond(const norm)	cond	行 列 の条 件 数 を計 算します
vector matrix.Spectrum()		行 列 のスペクト ルを、積 AT*Aからの固 有 値 のセット として計 算します
double matrix.Det()	det	配 列 の行 列 式 を計 算します
int matrix.Rank()	matrix_rank	ガウスの方 法 を使 用して配 列 の行 列 階 数 を返します
int matrix.SLogDet(int& sign)	slogdet	配 列 の行 列 式 の符 号 と対 数 を計 算します
double matrix.Trace()	trace	行 列 の対 角 線 に沿って合 計 を返します
vector matrix.Solve(const vector b)	solve	線 形 行 列 方 程 式 系 または線 形 代 数 方 程 式 系 を解きます
vector matrix.LstSq(const vector b)	lstsq	線 形 代 数 方 程 式 の最 小 二 乗 解 を返します(非 二 乗 行 列 または縮 退 行 列 の場 合)
matrix matrix.Inv()	inv	行 列 の(乗 法)逆 数 を計 算します
matrix matrix.PInv()	pinv	ムーア・ペンローズ法 により、行 列 の疑 似 逆 行 列 を計 算します
int matrix.Compare(const matrix matrix_c, const double epsilon) int matrix.Compare(const matrix matrix_c, const int digits) int vector.Compare(const vector vector_c, const double epsilon) int vector.Compare(const vector vector_c, const int digits)		指 定 された精 度 で2つの行 列 /ベクト ルの要 素 を比 較します
double matrix.Flat(const ulong index) bool matrix.Flat(const ulong index,const double value)	flat	2つではなく1つのインデックスを介して 行 列 要 素 をアドレス指 定 できます
double vector.ArgMax() double matrix.ArgMax() vector matrix.ArgMax(const int axis)	argmax	最 大 値 のインデックスを返します

© 2000–2025, MetaQuotes Ltd.

メソッド 行 列 /ベクト ル	NumPyの類 似 の 方 法	説 明
double vector.ArgMin() double matrix.ArgMin() vector matrix.ArgMin(const int axis)	argmin	最 小 値 のインデックスを返します
double vector.Max() double matrix.Max() vector matrix.Max(const int axis)	max	行 列 /ベクト ルの最 大 値 を返します
double vector.Mean() double matrix.Mean() vector matrix.Mean(const int axis)	mean	要 素 値 の算 術 平 均を計 算します
double vector.Min() double matrix.Min() vector matrix.Min(const int axis)	min	行 列 /ベクト ルの最 小 値 を返します
double vector.Sum() double matrix.Sum() vector matrix.Sum(const int axis)	sum	指 定 された軸 に対 しても実 行 できる 行 列 /ベクト ル要 素 の合 計 を返します
void vector.Clip(const double min_value,const double max_value) void matrix.Clip(const double min_value,const double max_value)	clip	行 列 /ベクト ルの要 素 を有 効 な値 の 指 定 された範 囲 に制 限します
vector vector.CumProd() vector matrix.CumProd() matrix matrix.CumProd(const int axis)	cumprod	指 定 された軸 に沿 ったものを含 む、行 列 /ベクト ル要 素 の累 積 積 を返します
vector vector.CumSum() vector matrix.CumSum() matrix matrix.CumSum(const int axis)	cumsum	指 定 された軸 に沿 ったものを含 む、行 列 /ベクト ル要 素 の累 積 和 を返します
double vector.Prod(const double initial=1) double matrix.Prod(const double initial=1) vector matrix.Prod(const int axis,const double initial=1)	prod	指 定 された軸 に対 しても実 行 できる 行 列 /ベクト ル要 素 の積 を返します
void matrix.Reshape(const ulong rows, const ulong cols)	reshape	データを変 更 せずに行 列 の形 状 を変 更 します
void matrix.Resize(const ulong rows,const ulong cols)	resize	形 状 とサイズが変 更 された新 しい行 列 を返します
bool matrix.SwapRows(const ulong row1, const ulong row2)		行 列 の行 を入 れ替 えます

© 2000–2025, MetaQuotes Ltd.

メソッド 行列/ベクトル	NumPyの類似の方法	説明
bool matrix.SwapCols(const ulong col1, const ulong col2)		行列の列を入れ替えます
double vector.Ptp() double matrix.Ptp() vector matrix.Ptp(const int axis)	ptp	Max() - Min()と同等の、行列/ベクトルまたは指定された行列軸の値の範囲を返します
double vector.Percentile(const int percent) double matrix.Percentile(const int percent) vector matrix.Percentile(const int percent,const int axis)	percentile	行列/ベクトル要素または指定された軸に沿った要素の値の指定されたパーセンタイルを返します。percentパラメータの有効な値は[0, 100]の範囲です
double vector.Quantile(const int percent) double matrix.Quantile(const int percent) vector matrix.Quantile(const int percent,const int axis)	quantile	行列/ベクトル要素または指定された軸に沿った要素の値の指定された分位数を返します。percentパラメータの値は[0, 100]の範囲です
double vector.Median() double matrix.Median() vector matrix.Median(const int axis)	median	行列/ベクトル要素の中央値を計算します。中央値とは、配列/ベクトル要素の上位半分を要素の下位半分から分離する真ん中の値です
double vector.Average() double matrix.Average() vector matrix.Average(const int axis)	average	行列/ベクトル値の算術平均を計算します。分母の重みの合計を0に等しくすることはできませんが、一部の重みは0にすることができます
double vector.Std() double matrix.Std() vector matrix.Std(const int axis)	std	行列/ベクトル要素または指定された軸に沿った要素の値の標準偏差を返します
double vector.Var() double matrix.Var() vector matrix.Var(const int axis)	var	行列/ベクトル要素の値の分散を計算します
double vector.CorrCoef(const vector& v) matrix matrix.CorrCoef()	corrcoef	ピアソン相関係数(線形相関係数)を計算します。相関係数は[-1, 1]の範囲にあります
vector vector.Correlate(const vector& v,enum mode)	correlate	2つのベクトルの相互相関を計算します。modeパラメータは、線形畳み込み計算モードを決定します
vector vector.Convolve(const vector& v, enum mode)	convolve	2つのベクトルの離散線形畳み込みを返します。modeパラメータは、線形畳み込み計算モードを決定します

© 2000-2025, MetaQuotes Ltd.

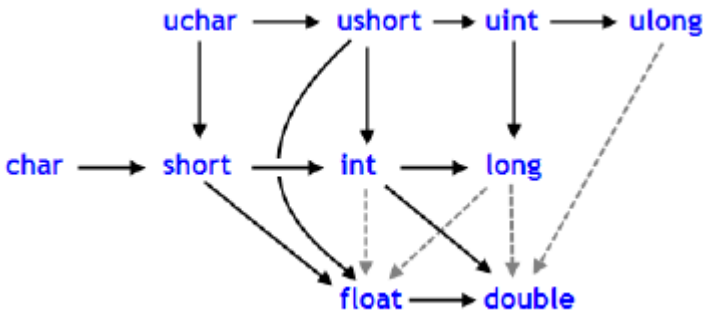
メソッド 行 列 /ベクト ル	NumPyの類 似 の 方 法	説 明
matrix matrix.Cov() matrix vector.Cov(const vector& v); (resulting matrix 2 x 2)	cov	共 分 散 行 列を計 算します。2つの標 本（2つの確 率 変 数）の共 分 散は、それらの線 形 依 存 性 の尺 度 です

© 2000–2025, MetaQuotes Ltd.

型キャスト

数値型のキャスト

数値型を別の数値型に変換する必要性はよく発生します。全ての数値型の変換が可能なわけではありません。許可されたキャストのスキームは次の通りです。



黒い矢印は、キャストがほとんど情報を失うことなく行われることを示します。char 型の代わりにbool 型（共に 1 バイト）、int 型の代わりに color 型（4 バイト）、long 型の代わりに datetime 型（8 バイト）も使用出来ます。4 本の灰色の破線の矢印は、キャストで精度の損失が発生する可能性があることを示します。例えば、123,456,789（int）に等しい整数の桁数は float で表すことが出来る桁数よりも大きいです。

```
   int n=123456789;
   float f=n;     // f の中身は 1.234567892E8 に等しい
   Print("n = ",n,"   f = ",f);
   // 結果 n= 123456789    f= 123456792.00000
```

float に変換された数は同じ位数を持っていますが、正確差が劣ります。黒矢印と反対方向の変換は、データの損失を伴う可能性がありますが行うことが出来ます。char と uchar、short と ushort、int と uint、long と ulong の間の変換（両方向）は、データの損失につながる可能性があります。

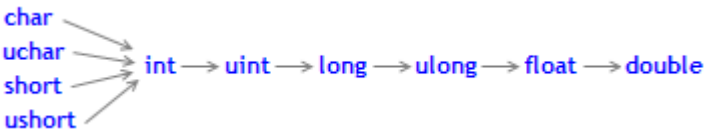
浮動小数点値を整数型に変換した結果、小数部分は常に削除されます。float を最も近い整数に四捨五入する場合には（多くの場合、実用性がより高いため）、MathRound() が使用されるべきです。

例：

```
//--- 重力加速度
   double g=9.8;
   double round_g=(int)g;
   double math_round_g=MathRound(g);
   Print("round_g = ",round_g);
   Print("math_round_g = ",math_round_g);
/*
  結果:
   round_g = 9
   math_round_g = 10
*/
```

2 つの値が 2 進数演算子で結合される場合は、演算が実行される前に、以下のスキームに与えられた優先度に応じて、低型のオペランドがより高型に変換されます。

© 2000-2025, MetaQuotes Ltd.



データ型 char、uchar、short 及び ushort は無条件で int 型に変換されます。

例：

```
   char    c1=3;
//--- 例1
   double d2=c1/2+0.3;
   Print("c1/2 + 0.3 = ",d2);
// 結果:    c1/2+0.3 = 1.3

//--- 例 2
   d2=c1/2.0+0.3;
   Print("c1/2.0 + 0.3 = ",d2);
// 結果:    c1/2.0+0.3 = 1.8
```

計算された式は、2 つの演算から構成されています。例 1 では、除算演算で第 2 オペランドである定数 2 がより高い int 型であるため、char 型の変数 c1 は int 型の一時変数に変換されます。3/2 の整数除算の結果、int 型の値 1 が取得されます。

例 1 の 2 番目の演算では、第 2 オペランドが double 型定数 0.3 であるため、1 番目の演算の結果は、1.0 の値を持つ double 型の一時変数に変換されます。

例 2 では、除算演算で第 2 オペランドである定数 2.0 が double 型であるため、char 型変数の c1 は、double 型の一時変数に変換されます。更なる変換は行われません。

数値型の型キャスト

MQL5 言語の式では、明示的及び暗黙的な型キャストの両方を使用することが出来ます。明示的な型キャストは次のように書かれます:

```
var_1 = (型)var_2;
```

式または関数の実行結果が var_2 変数として使用出来ます。明示的な型キャストの関数スタイルの記法も可能です。

```
var_1 = 型(var_2);
```

例 1 に基づいて明示的な型キャストを考えてみましょう。

```
//--- 例 3
   double d2=(double)c1/2+0.3;
   Print("(double)c1/2 + 0.3 = ",d2);
// 結果:    (double)c1/2+0.3 = 1.80000000
```

除算演算が実行される前に、変数 c1 は、明示的に double 型にキャストされます。最初のオペランドが double 型変換にされた結果として、整数定数 2 が double 型の値 2.0 にキャストされます。実際には、明示的な型キャストは単項演算です。

© 2000-2025, MetaQuotes Ltd.

また、型キャストしようとした時に、結果が許容範囲を越える可能性があります。この場合には、切り捨てが発生します。例えば、

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

（代入以外の）演算が実行される前に、データが最優先タイプに変換されます。代入演算が実行される前に、データは、ターゲットの型にキャストされます。

例：

```
   int     i=1/2;          // 型キャストなし。結果は 0
   Print("i = 1/2  ",i);

   int k=1/2.0;            // 式が double にキャストされ
   Print("k = 1/2  ",k);   // そしてターゲット型の int にキャストされる。結果は 0

   double d=1.0/2.0;       // 型キャストなし。結果は 0.5
   Print("d = 1/2.0; ",d);

   double e=1/2.0;         // 式が double にキャストされ
   Print("e = 1/2.0; ",e);// それがターゲット型と同じ。結果は 0.5

   double x=1/2;           // int 型の式がターゲットの double 型にキャストされる
   Print("x = 1/2; ",x);   // 結果は 0.0
```

long/ulong 型を double に変換する場合、整数が 9,223,372,036,854,774,784 より大きく -9,223,372,036,854,774,784 未満の場合、精度が失われることがあります。

```
void OnStart()
  {
   long l_max=LONG_MAX;
   long l_min=LONG_MIN+1;
//--- double にキャストされた時に正確さを失わない最大の整数値を定義する
   while(l_max!=long((double)l_max))
      l_max--;
//--- double にキャストされた時に正確さを失わない最小の整数値を定義する
   while(l_min!=long((double)l_min))
      l_min++;
//--- 見つかった整数値の間隔を導出する
   PrintFormat("When casting an integer value to double, it must be "
               "within [%I64d, %I64d] interval",l_min,l_max);
//--- 値がこの間隔外にある場合どうなるか見てみる
   PrintFormat("l_max+1=%I64d, double(l_max+1)=%.f, ulong(double(l_max+1))=%I64d",
               l_max+1,double(l_max+1),long(double(l_max+1)));
   PrintFormat("l_min-1=%I64d, double(l_min-1)=%.f, ulong(double(l_min-1))=%I64d",
               l_min-1,double(l_min-1),long(double(l_min-1)));
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 次の結果を受け取る
// 数値を double にキャストする場合 [-9223372036854774784, 9223372036854774784] 間隔でなければいけない。
// l_max+1=9223372036854774785, double(l_max+1)=9223372036854774800, ulong(double(l_max+1))=9223372036854774784
// l_min-1=-9223372036854774785, double(l_min-1)=-9223372036854774800, ulong(double(l_min-1))=-9223372036854774784
  }
```

文字列型の型キャスト

文字列型は、基本データ型の中で最も高い優先順位を有します。そのために、演算のオペランドの 1 つが文字列型である場合には2　つ目のオペランドは自動的に文字列にキャストされます。文字列のために、また、単一の二項2位の動作が可能であることにご注意ください。文字列の任意の数値型への明示的なキャストは許可されています。

例：

```
   string s1=1.0/8;                // 式が double 型にキャストされてから、
   Print("s1 = 1.0/8; ",s1);       //  ターゲットの string 型にキャストされる
// 結果は "0.12500000"（10 文字を含む文字列）

   string s2=NULL;                 // 文字列の初期化
   Print("s2 = NULL; ",s2);        // 結果は空の文字列
   string s3="Ticket N"+12345;     // 式が string 型にキャストされる
   Print("s3 = \"Ticket N\"+12345",s3);

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

基本クラスのポインタから派生クラスのポインタへの型キャスト

オープン生成されたクラスオブジェクトは、対応する基本クラスのオブジェクトと見なすことが出来ます。これは、いくつかの興味深い結果をもたらします。例えば、単一の基本クラスによって生成された異なるクラスのオブジェクトは、互いに大きく異なる可能性があるという事実にもかかわらず、両クラスが基本型のオブジェクトとしてみなされるために、それらのリンクリスト（リスト）を作成することが出来ます。しかし、逆は真ではありません。基本クラスのオブジェクトは自動的に派生クラスのオブジェクトにはなりません。

基本クラスのポインタを派生クラスのポインタに変換するためには、明示的なキャストが使用されます。しかし、このようなキャストは間違えなしに行う必要があります。さもないと深刻な実行時エラーが発生し　MQL5 プログラムが停止されます。

© 2000-2025, MetaQuotes Ltd.

dynamic_cast演算子を使用した型の動的動作

クラスのポインタを適用することができるdynamic_cast演算子を使った型の動的キャストがあります。このタイプの精度の検証は、実行時に行われます。これはつまり、dynamic_cast演算子の使用時に、コンパイラは使用するデータの型の検証を行わないということを意味します。もし、実際のタイプではないオブジェクトのデータ型へのポインタの変換がある場合、結果はNULLとなります。

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

角括弧の中のパラメータ *type-id* は、以前に定義されたクラスのタイプのポインタでなければなりません。オペランドタイプ*expression*は(C++とは異なり)void以外の任意のものであっても良いです。

例：

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

void OnStart()
  {
   CBar bar;
//--- *barの*fooへのポインタタイプの動的キャストは許可されました
   CFoo *foo = dynamic_cast<CFoo *>(&bar); // 重大な実行エラーは発生しません
   Print(foo);                             // foo=NULL
//--- Bar型オブジェクトのFoo型オブジェクトへの明示的参照のキャストの試みは禁止されています
   foo=(CFoo *)&bar;                       // 重大な実行エラーが発生します
   Print(foo);                             // この行は実行されません
  }
```

参照

データ型

© 2000-2025, MetaQuotes Ltd.

Void 型とNULL 定数

文法的に void 型は、char 型、uchar 型、bool 型、short 型、ushort 型、int 型、uint 型、color 型、long 型、ulong 型、datetime 型、float 型、double 型及び string 型と同じように基本型です。この型は、関数が値を返さないことを示すために使用されるか、または関数のパラメータとしては、パラメータが存在しないことを意味します。

事前定義された定数変数 **NULL** は *void* 型です。NULL は、変換なしで他の基本的な型の変数に割り当てることが出来ます。また、基本型の変数と **NULL** の比較も可能です。

例:

```
//--- 文字列が初期化されていない場合は、事前定義された値を割り当てる
if(some_string==NULL) some_string="empty";
```

また **NULL** は new 演算子で作成されたポインタと比較することもできます。

参照

変数、関数

© 2000–2025, MetaQuotes Ltd.

ユーザー定義型

C ++のtypedefキーワードを使用すると、ユーザー定義のデータ型を作成できます。これを行うには、既存のデータ型に新しいデータ型名を指定するだけです。新しいデータ型は作成されません。代わりに既存の型の新しい名前が定義されます。アプリケーションの柔軟性はユーザー定義型によって向上します。typedef命令を置換マクロ(#define)に変えるだけで十分です。typedefを使用すると標準データ型にカスタム名を適用することができるため、コードの可読性もユーザー定義型によって向上します。ユーザー定義型を作成するためのエントリの一般的な形式は下記の通りです。

```
   typedef type new_name;
```

ここで*type*は任意のデータ型で*new_name*は型の新しい名前です。新しい名前は、既存の型の名前への追加(置換ではない)として設定されます。MQL5では typedefを使用して関数へのポインタを作成することも可能です。

関数へのポインタ

関数へのポインタは、通常、次の形式で定義されます。

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

ここではtypedefに続いて、関数シグニチャー(入力パラメーターの数と型、関数が返す結果の型)が設定されています。以下は、関数へのポインタの作成と適用の簡単な例です。

```
//--- 2つのint パラメーターを受け入れる関数へのポインタを宣言する
   typedef int (*TFunc)(int,int);
//--- TFuncは型で、関数に変数ポインタを宣言することは可能
   TFunc func_ptr; // pointer to the function
//--- TFuncの記述に対応する関数を宣言する
   int sub(int x,int y) { return(x-y); }  // 1つの数を別の数から減算する
   int add(int x,int y) { return(x+y); }  // 2つの数の和
   int neg(int x)       { return(~x);  }  // 変数のビットを反転する
//--- func_ptr 変数は後でそれを宣言するために関数のアドレスを格納することができる
   func_ptr=sub;
   Print(func_ptr(10,5));
   func_ptr=add;
   Print(func_ptr(10,5));
   func_ptr=neg;           // エラー:negの型はint(int,int) でない
   Print(func_ptr(10));    // エラー:2つのパラメータが必要
```

この例では、関数へのTFuncポインタで定義されているように、*func_ptr*変数はそれぞれint型の2つの入力を持っている*sub*関数と*add*関数を受け取ることができます。これに比べて*neg*関数は異なるシグネチャーを持っているためfunc_ptrポインタには割り当てできません。

ユーザーインターフェイスでのイベントモデルの配置

関数へのポインタを使用すると、ユーザーインターフェイスの作成時にイベント処理を簡単に作成できます。CButtonセクションの例を使って、ボタンを作成してそれを押すための関数を追加する方法を示してみましょう。まず、ボタンを押して呼び出す *TAction* 関数へのポインタを定義し *TActionの記述に従って3つの関数を作成します。*

© 2000-2025, MetaQuotes Ltd.

```
//--- カスタム関数型を作成する
typedef int(*TAction)(string,int);
//+------------------------------------------------------------------+
//|  ファイルを開く                                                  |
//+------------------------------------------------------------------+
int Open(string name,int id)
  {
   PrintFormat("%s function called (name=%s id=%d)",__FUNCTION__,name,id);
   return(1);
  }
//+------------------------------------------------------------------+
//|   ファイルを保存する                                             |
//+------------------------------------------------------------------+
int Save(string name,int id)
  {
   PrintFormat("%s function called (name=%s id=%d)",__FUNCTION__,name,id);
   return(2);
  }
//+------------------------------------------------------------------+
//|   ファイルを閉じる                                               |
//+------------------------------------------------------------------+
int Close(string name,int id)
  {
   PrintFormat("%s function called (name=%s id=%d)",__FUNCTION__,name,id);
   return(3);
  }
```

その後CButtonからMyButtonクラスを作成し、関数へのTActionポインタを追加します。

```
//+------------------------------------------------------------------+
//|  イベント処理関数を使用してボタンクラスを作成する                |
//+------------------------------------------------------------------+
class MyButton: public CButton
  {
private:
   TAction           m_action;                    // チャートイベントハンドラ
public:
                     MyButton(void){}
                    ~MyButton(void){}
   //--- ボタンテキストとイベント処理関数へのポインタを指定するコンストラクタ
                     MyButton(string text, TAction act)
     {
      Text(text);
      m_action=act;
     }
   //--- OnEvent() イベントハンドラから呼び出されたカスタム関数を設定する
   void              SetAction(TAction act){m_action=act;}
   //--- 標準的なチャートイベントハンドラ
```

© 2000-2025, MetaQuotes Ltd.

```
   virtual bool      OnEvent(const int id,const long &lparam,const double
&dparam,const string &sparam) override
     {
      if(m_action!=NULL && lparam==Id())
        {
         //--- カスタムm_action()ハンドラを呼び出す
         m_action(sparam,(int)lparam);
         return(true);
        }
      else
      //--- CButton親クラスからハンドラを呼び出した結果を返す
         return(CButton::OnEvent(id,lparam,dparam,sparam));
     }
  };
```

CAppDialogからCControlsDialog派生クラスを作成して、*MyButton*型のボタンを格納するための*m_buttons*配列と*AddButton(MyButton &button)*及び*CreateButtons()*メソッドを追加します。

```
//+------------------------------------------------------------------+
//|  CControlsDialogクラス                                           |
//|  目的:アプリケーションを管理するためのグラフィカルパネル         |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CArrayObj         m_buttons;                    // ボタン配列
public:
                     CControlsDialog(void){};
                    ~CControlsDialog(void){};
   //--- 作成する
   virtual bool      Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2) override;
   //--- ボタンを追加する
   bool              AddButton(MyButton &button)
{return(m_buttons.Add(GetPointer(button)));m_buttons.Sort();};
protected:
   //--- ボタンを作成する
   bool              CreateButtons(void);
  };
//+------------------------------------------------------------------+
//|  チャートにCControlsDialogオブジェクトを作成する                 |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2)
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
   return(CreateButtons());
//---
  }
```

```
//+------------------------------------------------------------------+
//| define                                                           |
//+------------------------------------------------------------------+
//--- インデントとギャップ
#define INDENT_LEFT                         (11)      // 左からのインデント(境界幅の余裕を含む)
#define INDENT_TOP                          (11)      // 上からのインデント(境界幅の余裕を含む)
#define CONTROLS_GAP_X                      (5)       // ギャップのX座標
#define CONTROLS_GAP_Y                      (5)       // ギャップのY座標
//--- ボタン
#define BUTTON_WIDTH                        (100)     // サイズのX座標
#define BUTTON_HEIGHT                       (20)      // サイズのY座標
//--- 表示領域
#define EDIT_HEIGHT                         (20)      // サイズのY座標
//+------------------------------------------------------------------+
//| CControlsDialogパネルを作成してボタンを追加する                  |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButtons(void)
  {
//--- ボタン座標を計算する
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2;
   int y2=y1+BUTTON_HEIGHT;
//--- 関数ポインタとともにボタンオブジェクトを追加する
   AddButton(new MyButton("Open",Open));
   AddButton(new MyButton("Save",Save));
   AddButton(new MyButton("Close",Close));
//--- グラフィカルにボタンを作成する
   for(int i=0;i<m_buttons.Total();i++)
     {
      MyButton *b=(MyButton*)m_buttons.At(i);
      x1=INDENT_LEFT+i*(BUTTON_WIDTH+CONTROLS_GAP_X);
      x2=x1+BUTTON_WIDTH;
      if(!b.Create(m_chart_id,m_name+"bt"+b.Text(),m_subwin,x1,y1,x2,y2))
        {
         PrintFormat("Failed to create button %s %d",b.Text(),i);
         return(false);
        }
      //--- CControlsDialogコンテナにそれぞれのボタンを追加する
      if(!Add(b))
         return(false);
     }
//--- 成功
   return(true);
  }
```

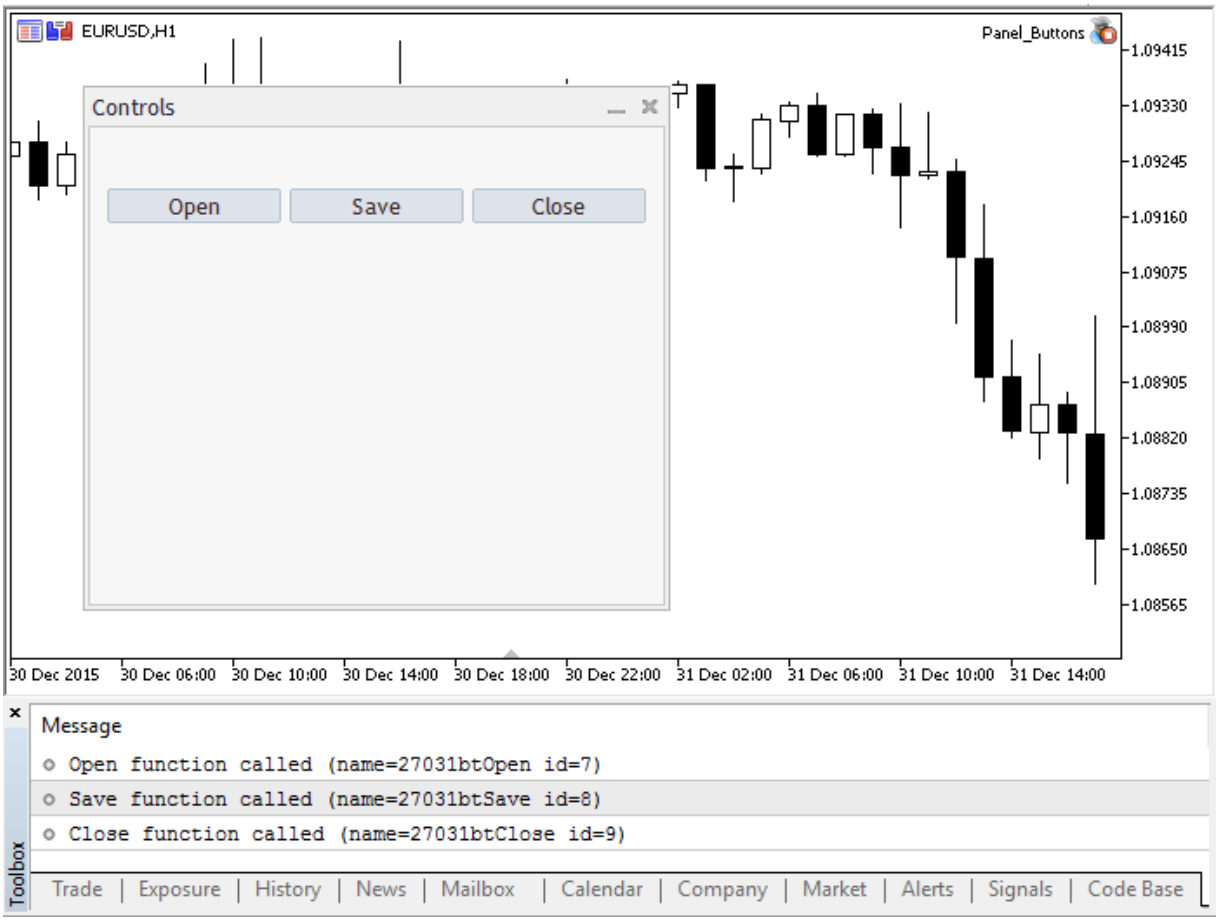
© 2000-2025, MetaQuotes Ltd.

これで、CControlsDialogコントロールパネルを使って、Open、Save、Closeの3つのボタンを持つプログラムを開発することができます。ボタンをクリックするとTActionポインタの形をもつ適切な関数が呼び出されます。

```
//--- オブジェクトをグローバルレベルで宣言してプログラムを起動するときに自動的に作成する
CControlsDialog MyDialog;
//+------------------------------------------------------------------+
//| エキスパート初期化に使用される関数                                |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- チャートにオブジェクトを作成する
   if(!MyDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- アプリケーションを起動する
   MyDialog.Run();
//--- アプリケーションが正常に初期化された
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| エキスパート初期化解除に使用される関数                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- ダイアログを破壊する
   MyDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| エキスパートチャートイベント関数                                  |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // イベントID
                  const long& lparam,   // long型のイベントパラメータ
                  const double& dparam, // double型のイベントパラメータ
                  const string& sparam) // string型のイベントパラメータ
  {
//--- チャートイベントのハンドラを親クラス(ここではCAppDialog)から呼び出す
   MyDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

起動したアプリケーションの外観とボタンクリックの結果は、スクリーンショットで見られます。

© 2000-2025, MetaQuotes Ltd.



プログラムの完全なソースコードは下記の通りです。

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

#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The panel with several CButton buttons"
#include <Controls\Dialog.mqh>
#include <Controls\Button.mqh>
//+------------------------------------------------------------------+
//| define                                                           |
//+------------------------------------------------------------------+
//--- インデントとギャップ
#define INDENT_LEFT                         (11)      // 左からのインデント（境界幅の余裕を含む）
#define INDENT_TOP                          (11)      // 上からのインデント（境界幅の余裕を含む）
#define CONTROLS_GAP_X                      (5)       // ギャップのX座標
```

© 2000–2025, MetaQuotes Ltd.

```
#define CONTROLS_GAP_Y                  (5)       // ギャップのY座標
//--- ボタン
#define BUTTON_WIDTH                    (100)     // サイズのX座標
#define BUTTON_HEIGHT                   (20)      // サイズのY座標
//--- 表示領域
#define EDIT_HEIGHT                     (20)      // サイズのY座標

//--- カスタム関数型を作成する
typedef int(*TAction)(string,int);
//+------------------------------------------------------------------+
//| ファイルを開く                                                   |
//+------------------------------------------------------------------+
int Open(string name,int id)
  {
   PrintFormat("%s function called (name=%s id=%d)",__FUNCTION__,name,id);
   return(1);
  }
//+------------------------------------------------------------------+
//|  ファイルを保存する                                              |
//+------------------------------------------------------------------+
int Save(string name,int id)
  {
   PrintFormat("%s function called (name=%s id=%d)",__FUNCTION__,name,id);
   return(2);
  }
//+------------------------------------------------------------------+
//|  ファイルを閉じる                                                |
//+------------------------------------------------------------------+
int Close(string name,int id)
  {
   PrintFormat("%s function called (name=%s id=%d)",__FUNCTION__,name,id);
   return(3);
  }
//+------------------------------------------------------------------+
//| イベント処理関数を使用してボタンクラスを作成する                |
//+------------------------------------------------------------------+
class MyButton: public CButton
  {
private:
   TAction           m_action;                    // チャートイベントハンドラ
public:
                     MyButton(void){}
                    ~MyButton(void){}
   //--- ボタンテキストとイベント処理関数へのポインタを指定するコンストラクタ
                     MyButton(string text,TAction act)
     {
      Text(text);
      m_action=act;
     }
```

© 2000–2025, MetaQuotes Ltd.

```
   //--- OnEvent() イベントハンドラから呼び出されたカスタム関数を設定する
   void              SetAction(TAction act){m_action=act;}
   //--- 標準的なチャートイベントハンドラ
   virtual bool      OnEvent(const int id,const long &lparam,const double
&dparam,const string &sparam) override
     {
      if(m_action!=NULL && lparam==Id())
        {
         //--- カスタムハンドラを呼び出す
         m_action(sparam,(int)lparam);
         return(true);
        }
      else
      //--- CButton親クラスからハンドラを呼び出した結果を返す
         return(CButton::OnEvent(id,lparam,dparam,sparam));
     }
  };
//+------------------------------------------------------------------+
//| CControlsDialogクラス                                            |
//| 目的:アプリケーションを管理するためのグラフィカルパネル          |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CArrayObj         m_buttons;                    // ボタン配列
public:
                     CControlsDialog(void){};
                    ~CControlsDialog(void){};
   //--- 作成する
   virtual bool      Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2) override;
   //--- ボタンを追加する
   bool              AddButton(MyButton &button)
{return(m_buttons.Add(GetPointer(button)));m_buttons.Sort();};
protected:
   //--- ボタンを作成する
   bool              CreateButtons(void);
  };
//+------------------------------------------------------------------+
//| チャートにCControlsDialogオブジェクトを作成する                  |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2)
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
   return(CreateButtons());
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  CControlsDialogパネルを作成してボタンを追加する                 |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButtons(void)
  {
//--- ボタン座標を計算する
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2;
   int y2=y1+BUTTON_HEIGHT;
//--- 関数ポインタとともにボタンオブジェクトを追加する
   AddButton(new MyButton("Open",Open));
   AddButton(new MyButton("Save",Save));
   AddButton(new MyButton("Close",Close));
//--- グラフィカルにボタンを作成する
   for(int i=0;i<m_buttons.Total();i++)
     {
      MyButton *b=(MyButton*)m_buttons.At(i);
      x1=INDENT_LEFT+i*(BUTTON_WIDTH+CONTROLS_GAP_X);
      x2=x1+BUTTON_WIDTH;
      if(!b.Create(m_chart_id,m_name+"bt"+b.Text(),m_subwin,x1,y1,x2,y2))
        {
         PrintFormat("Failed to create button %s %d",b.Text(),i);
         return(false);
        }
      //--- CControlsDialogコンテナにそれぞれのボタンを追加する
      if(!Add(b))
         return(false);
     }
//--- 成功
   return(true);
  }
//--- オブジェクトをグローバルレベルで宣言してプログラムを起動するときに自動的に作成する
CControlsDialog MyDialog;
//+------------------------------------------------------------------+
//|  エキスパート初期化に使用される関数                              |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- チャートにオブジェクトを作成する
   if(!MyDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- アプリケーションを起動する
   MyDialog.Run();
//--- アプリケーションがっ正常に初期化された
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//|  エキスパート初期化解除に使用される関数                          |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- ダイアログを破壊する
   MyDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| エキスパートチャートイベント関数                                 |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // イベントID
                  const long& lparam,   // long型のイベントパラメータ
                  const double& dparam, // double型のイベントパラメータ
                  const string& sparam) // string型のイベントパラメータ
  {
//--- チャートイベントのハンドラを親クラス(ここではCAppDialog)から呼び出す
   MyDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

参照

変数、関数

© 2000-2025, MetaQuotes Ltd.

MQL5 は複合型オブジェクトの動的作成を可能にします。これは、作成されたオブジェクトの記述子を返す「new」演算子を使用して行われます。記述子のサイズは8バイトです。構文的に、MQL5のオブジェクト記述子はC++ポインタに似ています。

例：

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

C++ とは異なり、上記の例の「hobject」変数はメモリへのポインターではなく、オブジェクト記述子です。さらに、MQL5では、関数パラメータ内のすべてのオブジェクトは参照によって渡される必要があります。以下の例は、オブジェクトを関数パラメータとして渡す方法を示しています。

```
class Foo
  {
public:
   string            m_name;
   int               m_id;
   static int        s_counter;
   //--- コンストラクタとデストラクタ
                     Foo(void){Setup("noname");};
                     Foo(string name){Setup(name);};
                    ~Foo(void){};
   //--- Fooオブジェクトを初期化する
   void              Setup(string name)
     {
      m_name=name;
      s_counter++;
      m_id=s_counter;
     }
  };
int Foo::s_counter=0;
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 自動作成を使用してオブジェクトを変数として宣言する
   Foo foo1;
//--- 参照によってオブジェクトを渡す
   PrintObject(foo1);

//--- オブジェクトへのポインタを宣言し「new」演算子を使用して作成する
   Foo *foo2=new Foo("foo2");
//--- 参照によってオブジェクトへのポインタを渡す
   PrintObject(foo2); // the pointer to the object is converted by the compiler automatically

//--- Fooオブジェクトの配列を宣言する
   Foo foo_objects[5];
//--- オブジェクトの配列を渡す
```

© 2000-2025, MetaQuotes Ltd.

```
   PrintObjectsArray(foo_objects); // a separate function for passing an array of
objects

//--- Foo 型オブジェクトへのポインタの配列を宣言する
   Foo *foo_pointers[5];
   for(int i=0;i<5;i++)
      foo_pointers[i]=new Foo("foo_pointer");
//--- ポインタの配列を渡す
   PrintPointersArray(foo_pointers); // a separate function for passing an array of
pointers

//--- 終了する前に、ポインタとして作成されたオブジェクトを必ず削除する
   delete(foo2);
//--- ポインタ配列を削除する
   int size=ArraySize(foo_pointers);
   for(int i=0;i<5;i++)
      delete(foo_pointers[i]);
//---
  }
//+------------------------------------------------------------------+
//| オブジェクトは常に参照によって渡される                                  |
//+------------------------------------------------------------------+
void PrintObject(Foo &object)
  {
   Print(__FUNCTION__,": ",object.m_id," Object name=",object.m_name);
  }
//+------------------------------------------------------------------+
//| オブジェクトの配列を渡す                                              |
//+------------------------------------------------------------------+
void PrintObjectsArray(Foo &objects[])
  {
   int size=ArraySize(objects);
   for(int i=0;i<size;i++)
      PrintObject(objects[i]);
  }
//+------------------------------------------------------------------+
//| オブジェクトポインタの配列を渡す                                        |
//+------------------------------------------------------------------+
void PrintPointersArray(Foo* &objects[])
  {
   int size=ArraySize(objects);
   for(int i=0;i<size;i++)
      PrintObject(objects[i]);
  }
//+------------------------------------------------------------------+
```

使用前にポインタを確認

© 2000-2025, MetaQuotes Ltd.

不正なポインタを呼び出す試みはプログラムの緊急終了を生じます。CheckPointer関数を使用して、使用前にポインタをチェックします。ポインタは、次のような場合に不正であることがあります。

- ポインタがNULLに等しい
- delete演算子を使用してオブジェクトが破棄されました。

この関数は、ポインタの検証に使用できます。ゼロ以外の値は、このポインタでデータにアクセスできることを示します。

```
class CMyObject
 {
protected:
  double             m_value;
public:
                     CMyObject(void);
                     CMyObject(double value) {m_value=value;};
                    ~CMyObject(void){};
  //---
  double             Value(void) {return(m_value);}
 };
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- 初期化されていないオブジェクトを作成する
  CMyObject *pointer;
  if(CheckPointer(pointer)==POINTER_INVALID)
    Print("1. pointer is ", EnumToString(CheckPointer(pointer)));
  else
    Print("1. pointer.Value()=", pointer.Value());

//--- ポインタを初期化する
  pointer=new CMyObject(M_PI);
  if(CheckPointer(pointer)==POINTER_INVALID)
    Print("2. pointer is ", EnumToString(CheckPointer(pointer)));
  else
    Print("2. pointer.Value()=", pointer.Value());

//--- オブジェクトを削除する
  delete(pointer);
  if(CheckPointer(pointer)==POINTER_INVALID)
    Print("3. pointer is ", EnumToString(CheckPointer(pointer)));
  else
    Print("3. pointer.Value()=", pointer.Value());
 }
/*
   1. pointer is POINTER_INVALID
   2. pointer.Value()=3.141592653589793
   3. pointer is POINTER_INVALID
```

© 2000-2025, MetaQuotes Ltd.

```
*/
```

ポインタをすばやく検証するにはCheckPointer関数の暗黙的な呼び出しを介してチェックする演算子「!」（LNOT）も使用できます。これにより、より簡潔で明確なコードを記述できます。以下は前の例のチェックです。

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- 初期化されていないオブジェクトを作成する
  CMyObject *pointer;
  if(!pointer)
    Print("1. pointer is ", EnumToString(CheckPointer(pointer)));
  else
    Print("1. pointer.Value()=", pointer.Value());

//--- ポインタを初期化する
  pointer=new CMyObject(M_PI);
  if(!pointer)
    Print("2. pointer is ", EnumToString(CheckPointer(pointer)));
  else
    Print("2. pointer.Value()=", pointer.Value());

//--- オブジェクトを削除する
  delete(pointer);
  if(!pointer)
    Print("3. pointer is ", EnumToString(CheckPointer(pointer)));
  else
    Print("3. pointer.Value()=", pointer.Value());
 }
/*
   1. pointer is POINTER_INVALID
   2. pointer.Value()=3.141592653589793
   3. pointer is POINTER_INVALID
*/
```

演算子「==」は、NULL のクイック チェックに使用されます。例：ptr==NULLまたはptr!=NULL。

参照

変数、変数の初期化、変数のアクセス権スコープとライフタイム、オブジェクトの作成と解徐

© 2000–2025, MetaQuotes Ltd.

参照：修飾子「 & 」とキーワード「 this 」

参照でのパラメータの引き渡し

MQL5では基本データ型のパラメータは値と参照の両方によって渡されることが出来ますが、複合型のパラメータは常に参照によって渡されます。パラメータが参照によって渡される必要があるとコンパイラに知らせるには、アンパサンド「 & 」がパラメータ名の前に追加されます。

参照によってパラメータを渡すことは変数のアドレスを渡すことを意味します。よって、参照によって渡されたパラメータの変更はすぐにソース変数に反映されます。参照によってパラメータを渡すことによって、関数のいくつかの結果を同時に返すことが出来ます。参照によって渡されたパラメータの変更を防ぐには const 修飾子を使用します。

関数の入力パラメータが配列や構造体またはクラスのオブジェクトの場合は、「 & 」シンボルが関数ヘッダー内で変数の型と名称の間に配置されます。

例

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

public:
   void              setArray(double &array[]);
  };
//+------------------------------------------------------------------+
//| 配列に書き込む                                                   |
//+------------------------------------------------------------------+
void  CDemoClass::setArray(double &array[])
  {
   if(ArraySize(array)>0)
     {
     ArrayResize(m_array,ArraySize(array));
     ArrayCopy(m_array, array);
     }
  }
```

上記の例ではクラス CDemoClass が宣言され、private メンバである double 型の配列 m_array[] を含みます。関数 setArray() が宣言され、array[] が参照で渡されます。関数ヘッダーに参照での受け渡しについての指示が含まれていない場合、つまり、アンパサンドが含まれていない場合、コードをコンパイルが試まれるとエラーメッセージが生成されます。

配列が参照によって渡されるという事実にもかかわらず、配列を別の配列に割り当てることは出来ません。元の配列の内容を要素単位で受け取り側の配列に複製する必要があります。配列や構造体が関数のパラメータとして渡された時に関数の記述に「 & 」を含むことは絶対条件です。

「 this 」キーワード

© 2000-2025, MetaQuotes Ltd.

クラス型（オブジェクト）の変数は、参照によってもポインタによっても受け渡し出来ます。参照と同様に、ポインタはオブジェクトへのアクセスを可能にします。オブジェクトポインタは、宣言後に new オペレータの適用によって作成及び初期化されるべきです。

予約語「 this 」はクラスまたは構造体のメソッド内で自身へのオブジェクトの参照を得るために使用されます。「 this 」は常に、それが使用されているメソッドでオブジェクトを参照します。GetPointer(this) は GetPointer() の呼び出しが行われるメンバ関数内でのオブジェクトのポインタを提供します。MQL5 では関数はオブジェクトを返すことは出来ませんが、オブジェクトポインタを返すことは出来ます。

オブジェクトを返す関数が必要な場合は、GetPointer(this) を使用してオブジェクトのポインタを返すことが出来ます。CDemoClass の記述に、このクラスのオブジェクトのポインタを返す関数の getDemoClass() を追加してみましょう。

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

public:
   void              setArray(double &array[]);
   CDemoClass       *getDemoClass();
  };
//+------------------------------------------------------------------+
//| 配列に書き込む                                                   |
//+------------------------------------------------------------------+
void  CDemoClass::setArray(double &array[])
  {
   if(ArraySize(array)>0)
     {
      ArrayResize(m_array,ArraySize(array));
      ArrayCopy(m_array,array);
     }
  }
//+------------------------------------------------------------------+
//| 自分へのポインタを返す                                           |
//+------------------------------------------------------------------+
CDemoClass *CDemoClass::getDemoClass(void)
  {
   return(GetPointer(this));
  }
```

構造体にはポインタがないので、*new* 及び *delete* 演算子や GetPointer(this) は適用出来ません。

参照

オブジェクトポインタ、オブジェクトの作成と解除、変数のアクセス権 スコープとライフタイム

© 2000–2025, MetaQuotes Ltd.

演算子と式

文字や文字列には特に重要なものがいくつかあります。これらは俗に演算記号と言われるもので、例として下記があります。

+ - * / %	算術演算子
&& ||	論理演算子
= += *=	文字代入演算子

演算記号は式の中で使用され、適切なオペランドが与えられた時に意味を持ちます。句読点も同様に強調されています。これらは、括弧、中括弧、カンマ、コロン、及びセミコロンです。

操作記号、句読点、及びスペースは、独立した言語要素を分割するのに使用されます。

このセクションには以下のトピックの説明が含まれます。

- 式
- 算術演算子
- 代入演算子
- 関係の演算子
- ブール演算子
- ビット演算子
- その他の演算子
- 優先順位と操作順序

© 2000-2025, MetaQuotes Ltd.

式

式は 1 つ以上のオペランドと演算記号で構成されています。式は複数行で記述することが出来ます。

例：

```
a++; b = 10;           // 複数の式が 1 行に記述されている
//---  1 つの式がいくつかの行に分割されている
x = (y * z) /
     (w + 2) + 127;
```

セミコロン(;)で終了する式が演算子です。

参照

優先順位のルール

© 2000-2025, MetaQuotes Ltd.

算術演算子

算術演算には加法と乗法の演算が含まれます。

```
変数の加算                  i = j + 2;
変数の減算                  i = j - 3;
符号の反転                  x = - x;
変数の乗算                  z = 3 * x;
除算                        i = j / 5;
剰余                        minutes = time % 60;
変数に 1 を加算する         i++;
変数に 1 を加算する         ++i;
変数から 1 を減算する       k--;
変数から 1 を減算する       --k;
```

インクリメントとデクリメント演算は変数だけに適用され、定数に適用することは出来ません。プレインクリメント（ ++i ）とプレデクリメント（ —k ）は、この変数が式の中で使用される直前に適用されます。

ポストインクリメント（ i++ ）とポストデクリメント（ k— ）は、この変数が式の中で使用された直後に適用されます。

重要事項

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

上記の式を別のプログラミング環境に移動（例えば Borland C++ から MQL5）すると計算問題が発生することがあります。一般的に、計算の順序はコンパイラの実装に依存します。実際には、ポストデクリメント（ポストインクリメント）を実装する方法は 2 つあります。

1. 式全体を計算した後、ポストデクリメント（ポストインクリメント）が変数に適用される
2. 演算時すぐに、ポストデクリメント（ポストインクリメント）が変数に適用される

現在 MQL5 では、1 つ目の計算方法が実装されています。しかし、この知識があっても、その使用法と実験することは推奨されていません。

例：

```
int a=3;
a++;              // 有効な式
int b=(a++)*3;    // 無効な式
```

参照

優先順位のルール

© 2000-2025, MetaQuotes Ltd.

代入演算子

与えられた演算を含む式の値が代入後の左のオペランドの値です。

y 変数に x 値を代入	y = x;

以下の演算は、算術演算または論理演算と代入演算を団結します。

y 変数に x を加算	y += x
y 変数から x を減算	y -= x;
y 変数 に x を乗算	y *= x;
y 変数を x で除算	y /= x;
y 変数を x で除算した残り	y %= x;
y の 2 進数表現を右に x ビットだけシフトする	y >>= x;
yの 2 進数表現を左に x ビットだけシフトする	y <<= x;
y とx の 2 進数表現の AND ビット演算	y &= x;
y とx の 2 進数表現の OR ビット演算	y |= x;
y とx の 2 進数表現の XOR ビット演算	y ^= x;

ビット演算は整数のみに適用することが出来ます。y 表現の左右への x ビット論理シフトが実行される際には、x 値の 2 進数表現の下 5 桁が使用され、残りは破棄されます。つまりシフトは 0～31 ビットに行われます。

%= 演算（x のモジュールによってのＹ値）では結果の符号は分割数の符号と同じです。

代入演算子は式の中で複数回使用出来ます。この場合、式の処理は左から右に実行されます。

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

まず、変数 x の値が 3 に割り当てられ、その後 y に変数 x の値（すなわち 3）が割り当てられます。

参照

優先順位のルール

© 2000-2025, MetaQuotes Ltd.

関係の演算子

論理値の偽（ FALSE ）は整数のゼロで表されます。一方、論理値の真（ TRUE ）は任意のゼロ以外の値で表されます。

関係の演算子や論理演算を含む式の値は FALSE（ 0）か TRUE（1）です。

```
a と b が等しければ True       a == b;
a と b が等しくなければ True    a != b;
a が b より小さければ True      a < b;
a が b より大きければ True      a > b;
a が b 以下であれば True        a <= b;
a が b 以上であれば True        a >= b;
```

2 つの実数 は比較出来ません。一般的に、一見同一である 2 つの数が小数第 15 位の値が異なるために不均等であることがあります。2 つの実数を正しく比較するためには、これらの数の正規化後の差をゼロと比較します。

例：

```
bool CompareDoubles(double number1,double number2)
  {
   if(NormalizeDouble(number1-number2,8)==0) return(true);
   else return(false);
  }
void OnStart()
  {
   double first=0.3;
   double second=3.0;
   double third=second-2.7;
   if(first!=third)
     {
      if(CompareDoubles(first,third))
         printf("%.16f and %.16f are equal",first,third);
     }
  }
// 結果: 0.3000000000000000  0.2999999999999998  are equal
```

参照

優先順位のルール

© 2000-2025, MetaQuotes Ltd.

ブール演算子

論理否定 NOT(!)

論理否定(!)の演算対象となる値は算術型でなければなりません。演算対象の値が FALSE(0)の場合、結果は TRUE(1)になり、演算対象の値が FALSE(0)でない場合、結果はFALSE(0)になります。

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

OR(||)の論理演算

x と y の値の OR(||)論理演算。x か y の値が TRUE(null でない)の場合、式の値は TRUE(1)になります。その他の場合は FALSE(0)になります。

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

AND(&&)の論理演算

x と y の値の AND(&&)論理演算。x と y の値が TRUE(null でない)の場合、式の値は TRUE(1)になります。その他の場合は FALSE(0)になります。

ブール演算の短判断

いわゆる「短判断」のスキームはブール演算に適用されます。式の結果を正確に推定することが出来る時に式の計算が終了されます。

```
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 短判断の 1 つ目の例
   if(func_false() && func_true())
     {
      Print("Operation &&: You will never see this expression");
     }
   else
     {
      Print("Operation &&: Result of the first expression is false, so the second
wasn't calculated");
     }
//--- 短判断の 2 つ目の例
   if(!func_false() || !func_true())
     {
      Print("Operation ||: Result of the first expression is true, so the second
wasn't calculated");
     }
   else
     {
      Print("Operation ||: You will never see this expression");
     }
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| この関数は常に false を返す                                      |
//+------------------------------------------------------------------+
bool func_false()
  {
   Print("Function func_false()");
   return(false);
  }
//+------------------------------------------------------------------+
//| この関数は常に true を返す                                       |
//+------------------------------------------------------------------+
bool func_true()
  {
   Print("Function func_true()");
   return(true);
  }
```

参照

優先順位のルール

© 2000-2025, MetaQuotes Ltd.

ビット演算子

1 の補数

変数値の1の補数。式の値は、2 進数での各桁で、変数の桁の値が 1 の場合には 0、0 の場合には 1 となります。

```
b = ~n;
```

例:

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

右シフト

x の 2 進数表現が y 桁だけ右にシフトされます。シフトする値が符号なしの型の場合には論理右シフトが行われ、解放された左側のビットはゼロで満たされます。

シフトする値が符号付きの型の場合には算術右シフトが行われ、解放された左側のビットは符号ビットの値で満たされます(数が正であれば符号ビットの値は 0 で数が負であれば符号ビットの値は 1)。

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

例:

```
   char a='a',b='b';
   Print("Before:  a = ",a, "  b = ",b);
//--- 右にシフトする
   b=a>>1;
   Print("After:   a = ",a, "  b = ",b);
// 結果:
// Before:  a = 97   b = 98
// After:   a = 97   b = 48
```

左シフト

xの 2 進数表現が y 桁だけ左にシフトされ、解放された右側のビットはゼロで満たされます。

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

例:

```
   char a='a',b='b';
   Print("Before:  a = ",a, "  b = ",b);
//--- 左へシフトする
   b=a<<1;
   Print("After:   a = ",a, "  b = ",b);
// 結果:
// Before:  a = 97   b = 98
```

© 2000-2025, MetaQuotes Ltd.

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

シフトされる変数の長さ以上の数でのシフトは結果が定義されていないために推奨されません。

AND ビット演算

x と y の 2 進数表現の AND ビット演算式の値は、2 進数での各桁の内、x と y 両方がゼロ以外を含む桁は 1（TRUE）、それ以外は 0（FALSE）となります。

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

例：

```
   char a='a',b='b';
//--- AND 演算
   char c=a&b;
   Print("a = ",a,"  b = ",b);
   Print("a & b = ",c);
// 結果:
// a = 97   b = 98
// a & b = 96
```

OR ビット演算

y と x の 2 進数表現の OR ビット演算式の値は、2 進数での各桁のうち x または y がゼロ以外を含む桁は 1（TRUE）、それ以外は 0（FALSE）となります。式の値は、2 進数での各桁のうち x と y 両方がゼロ以外を含む桁は 1（TRUE）、それ以外は0（FALSE）となります。

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

例：

```
   char a='a',b='b';
//--- OR 演算
   char c=a|b;
   Print("a = ",a,"  b = ",b);
   Print("a | b = ",c);
// 結果:
// a = 97   b = 98
// a | b = 99
```

XOR ビット演算

y と x の 2 進数表現の XOR ビット演算式の値は、2 進数での各桁のうち x と y が違う値を含む桁は1（TRUE）、それ以外は 0（FALSE）となります。

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

例：

```
   char a='a', b='b';
//--- XOR 演算
   char c=a^b;
```

© 2000–2025, MetaQuotes Ltd.

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

ビット演算は整数のみに適用出来ます。

参照

優先順位のルール

© 2000-2025, MetaQuotes Ltd.

その他の演算子

インデックス作成 ([])

配列のi番目の要素にアドレス指定された場合、式の値は、iの通し番号を持つ変数の値になります。

例:

```
array[i] = 3; // 配列の i 番目の要素に 3 を割り当てる
```

整数のみが配列のインデックスになれます。4 次元以下の配列が使用可能です。それぞれの次元のインデックスは 0 から 次元のサイズ-1 です。例えば、50 の要素からなる1次元配列がある場合には、最初の要素は array[0] と参照され、最後の要素は array[49] となります。

配列の範囲以外のアドレスが指定された場合、実行サブシステムは重大エラーを発生し、プログラムが停止されます。

x1, x2 ,..., xn 引数での関数の呼び出し

各引数は、定数、変数、または対応する型式を表すことが出来ます。渡された引数をコンマで区切って指定し、呼び出された関数名の後ろの丸括弧の中に書く必要があります。

式の値とは、関数によって返される値です。もし、返される値の型が void の場合、関数呼び出しは、代入演算子の右側に配置出来ません。x1,..., xnの表現の実行順は保証されることにご注意ください。

例:

```
   int length=1000000;
   string a="a",b="b",c;
//---その他の演算子
   int start=GetTickCount(),stop;
   long i;
   for(i=0;i<length;i++)
     {
      c=a+b;
     }
   stop=GetTickCount();
   Print("time for 'c = a + b' = ",(stop-start)," milliseconds, i = ",i);
```

コンマ演算子 (,)

コンマで区切られた式は、左から右に実行されます。左式計算の副作用は右側の式を計算する前に現れることがあります。結果の型と値は右の式のものと一致します。受け渡されるパラメータのリスト(上記参照)は例として考えることが出来ます。

例:

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

ドット演算子 (.)

© 2000-2025, MetaQuotes Ltd.

ドット演算子は構造体とクラスの public メンバへのアクセス を直接行うのに使用されます。構文：

```
構造体型の変数名 . メンバ名
```

例：

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

スコープ定義演算子（::）

MQL5 プログラムの各関数には独自の実行スコープがあります。例えば、Print() システム関数はグローバルスコープで実行されます。Imported 関数は対応するインポートの範囲で呼び出されます。クラスのメソッド関数は対応するクラスのスコープを持ちます。スコープ解決演算の構文は次の通りです。

```
[スコープ名]::関数名(パラメータ)
```

スコープ名の不在は、グローバルスコープを使用する明示的な指示です。スコープ解決演算子がない場合、関数は最も近い範囲で求められます。ローカルスコープ内に関数がない場合、検索はグローバルスコープで行われます。

スコープ解決演算子はクラスメンバの 関数定義 にも使用されます。

```
型 クラス名::関数名(パラメータ)
   {
// 関数の本体
   }
```

同名の関数をプログラム内の複数の実行コンテキストから使用すると、曖昧さを引き起こす可能性があります。明示的なスコープを指定せずに呼び出される時の関数の優先順位は次の通りです。

1. クラスメソッド。指定された名称の関数がクラスに存在しない場合は、次のレベルに移動します。
2. MQL5関数。MQL5 言語にそのような関数がない場合、次のレベルに移動します。
3. ユーザ定義のグローバル関数。指定された名称の関数が見つからない場合は、次のレベルに移動します。
4. インポートされた関数。指定された名称の関数が見つからない場合、コンパイラがエラーを返します。

関数呼び出しのあいまいさを避けるために、常に明示的なスコープ解決演算を使用して関数スコープを指定するべきです。

© 2000-2025, MetaQuotes Ltd.

例：

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

class CCheckContext
  {
   int          m_id;
public:
                CCheckContext() { m_id=1234; }
protected:
   int          GetLastError() { return(m_id); }
  };
class CCheckContext2 : public CCheckContext
  {
   int          m_id2;
public:
                CCheckContext2() { m_id2=5678; }
   void         Print();
protected:
   int          GetLastError() { return(m_id2); }
  };
void CCheckContext2::Print()
  {
   ::Print("Terminal GetLastError",::GetLastError());
   ::Print("kernel32 GetLastError",kernel32::GetLastError());
   ::Print("parent GetLastError",CCheckContext::GetLastError());
   ::Print("our GetLastError",GetLastError());
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   CCheckContext2 test;
   test.Print();
  }
//+------------------------------------------------------------------+
```

データ型のサイズまたは任意のデータ型オブジェクトのサイズ取得の操作（sizeof）

sizeof 演算子を使用すると、識別子または型に対応するメモリのサイズを定義することが出来ます。sizeof 演算の形式は次の通りです。

例：

© 2000-2025, MetaQuotes Ltd.

```
sizeof(式)
```

括弧で囲まれた識別子または型名は式として使用することが出来ます。void 型の名称は使用出来ないことにご注意ください。また、識別子は、ビットフィールドに属したり関数名であることは出来ません。

式が静的配列（つまり、最初の次元が指定されている）の名称である場合、結果は配列全体のサイズ（要素数と型の長さの積）です。式が動的配列（つまり、最初の次元が指定されていない）の名称である場合、結果は動的配列のオブジェクトサイズです。

sizeof が構造体またはクラス型や構造またはクラス型の識別子に適用された場合、結果は構造体またはクラスの実際のサイズです。

例：

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

サイズは、コンパイル段階で計算されます。

参照

優先順位のルール

© 2000-2025, MetaQuotes Ltd.

優先順位のルール

以下の表は演算子を同じ優先順位ごとにまとめます。グループは優先順位が高い順に示されています。優先順位のルールは演算子と演算する値の集まりによって決定します。

注意事項：MQL5 言語での演算の優先順位は C ++ で採用されている優先度に対応するため、MQL4 言語での優先順位とは異なります。

演算子	記述	実行順序
() [] にて。	関数の呼び出し 配列要素への参照 構造体要素への参照	左から右
! ~ - ++ -- (type) sizeof	論理否定 ビット否定（補数） 符号変換 1 でインクリメント 1 でデクリメント 型キャスト バイトサイズの決定	右から左
* / %	乗算 除算 剰余	左から右
+ -	加算 減算	左から右
<< >>	左シフト 右シフト	左から右
< <= > >=	より小さい 以下 より大きい 以上	左から右
== !=	等しい 等しくない	左から右
&	AND ビット演算	左から右
^	XOR ビット演算	左から右
|	OR ビット演算	左から右
&&	論理積演算	左から右
||	論理和演算	左から右
?:	条件演算	右から左
= *= /= %=	代入 乗算代入 除算代入 剰余代入	右から左

© 2000–2025, MetaQuotes Ltd.

演算子	記述	実行順序
+= -= <<= >>= &= ^= |=	加算代入 減算代入 左シフト代入 右シフト代入 ビット論理積代入 ビット排他的論理和代入 ビット論理和代入	
,	コンマ	左から右

演算の実行順序を変更するには、より高い優先順位を持つものに括弧を使用します。

© 2000–2025, MetaQuotes Ltd.

演算子

言語演算子はタスク達成のために実行しなければならないいくつかのアルゴリズムの動作を示します。プログラム本体は、このような処理の連続です。1つ1つの演算はセミコロンで区切られます。

演算子	説明
重文演算子 ({})	中括弧 {} で囲まれた1つまたは複数の任意の型の演算子。
式の演算子 (;)	セミコロン (;) で終わる式。
return 演算子	関数を終了し、制御を呼び出し元のプログラムに返します。
if-else 条件演算子	選択が必要な時に使用されます。
?: 条件演算子	if-else 条件演算子より簡単な構文で同じ機能を持ちます。
switch 選択演算子	式の値と一致する定数に処理の制御を与えます。
while 反復演算子	式が false になるまで、演算が反復的に実行されます。式は各反復の前にチェックされます。
for 反復演算子	式が false になるまで、演算が反復的に実行されます。式は各反復の前にチェックされます。
do-while 反復演算子	式が false になるまで、演算が反復的に実行されます。終了条件は各ループの後にチェックされます。ループ本体は常に少なくとも1回実行されます。
break 演算子	最も近くにある外側の switch、while、do-while、または for の処理から抜けます。
continue 演算子	最も近くにある外側の while、do-while、または for の反復処理の始まりに制御を与えます。
new 演算子	適切なサイズのオブジェクトを作成し作成したオブジェクトの記述子を返します。
delete 演算子	new 演算子によって作成されたオブジェクトを削除します。

1 つの演算子が複数の行を占めることがあり、反対に複数の演算子が 1 行で記述されることもあります。実行命令を制御する演算子 (if、if-else、switch、while 及び for) は互いに入れ子にすることが出来ます。

例:

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

参照

変数の初期化、変数のアクセス権 スコープとライフタイム、オブジェクトの作成と解徐

© 2000-2025, MetaQuotes Ltd.

重文演算子

重文演算子（ブロック）は、中括弧 {}で囲まれた1つまたは複数のさまざまな型で構成されます。括弧の後のセミコロン（；）は不要です。

例：

```
if(x==0)
  {
   Print("invalid position x = ",x);
   return;
  }
```

参照

変数の初期化、変数のアクセス権 スコープとライフタイム、オブジェクトの作成と解徐

© 2000–2025, MetaQuotes Ltd.

式の演算子

セミコロン(;)で終了する式は演算子です。式の演算子の例としては次のものがあります。

代入演算子

識別子 = 式;

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

代入演算子は式の中で複数回使用出来ます。このケースでは処理は右から左へ行われます。

関数呼び出し演算子

関数名（引数1,..., 引数N);

```
FileClose(file);
```

空 演算子

セミコロン(;)のみで構成され、制御演算子の本体が空であることを示すために使用されます。

参照

変数の初期化、変数のアクセス権 スコープとライフタイム、オブジェクトの作成と解徐

© 2000-2025, MetaQuotes Ltd.

Return 演算子

return 演算子は実行中の関数を終了して、制御を呼び出し元のプログラムに返します。式の演算結果は呼び出し元の関数に返されます。式は代入演算子を含むことが出来ます。

例:

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

void 型の戻り値を持つ関数では、式の部分を含まない return 演算子が使用されるべきです。

```
void SomeFunction()
  {
   Print("Hello!");
   return;     // この演算子は省ける
  }
```

関数の右中括弧は、式の部分を含まない return 演算子の実行を暗黙的に示します。

返せるものとしては、基本データ型、基本構造体、及びオブジェクトポインタがあります。*return*演算子は、配列、クラスオブジェクトや複合構造体の変数を返すことは出来ません。

参照

変数の初期化、変数のアクセス権 スコープとライフタイム、オブジェクトの作成と解徐

© 2000-2025, MetaQuotes Ltd.

If-Else 条件演算子

IF - ELSE 条件演算子は選択が必要な時に使用されます。形式的には、構文は次の通りです。

```
if (式)
     演算子 1
else
     演算子 2
```

式が true の場合、演算子 1 が実行され、制御は演算子 2 の次の演算子に与えられます(演算子 2 は実行されません)。式が false の場合、演算子 2 が実行されます。

if 演算子の else 部分は省くことが出来ます。else 部分を省略すると入れ子にされた if で分岐が現れるかもしれません。この場合、else は同ブロックにあって最も近い else 部分を持たない if演算子と関係します。

例:

```
//--- else 部分は2 つ目の if 演算子と関係する
if(x>1)
   if(y==2) z=5;
else      z=6;
//--- else 部分は1 つ目の if 演算子と関係する
if(x>1)
  {
   if(y==2) z=5;
  }
else         z=6;
//--- 入り子にされた演算子
if(x=='a')
  {
   y=1;
  }
else if(x=='b')
  {
   y=2;
   z=3;
  }
else if(x=='c')
  {
   y=4;
  }
else Print("ERROR");
```

参照

変数の初期化、変数のアクセス権 スコープとライフタイム、オブジェクトの作成と解徐

© 2000-2025, MetaQuotes Ltd.

三項演算子 ?:

三項演算子の一般的な形式は次の通りです。

```
式 1 ?式 2 : 式 3
```

1 番目のオペランド（式 1 ）には bool 型の値の結果を持つ任意の式を使用出来ます。結果が true の場合、2 番目のオペランドによって設定された演算子、つまり「式 2 」が実行されます。

1 番目のオペランドが false の場合、3 番目のオペランド（式 3 ）が実行されます。2 番目と 3 番目のオペランド、つまり「式 2 」と「式 3 」は同じ型の値を返し、これは void 型であってはいけません。条件演算子の実行結果は、式 1 の結果に応じて、式 2 または式 3 の結果となります。

```
//--- 一日の始値と終値の差を正規化
double true_range = (High==Low)?0:(Close-Open)/(High-Low);
```

これは下記と同じです。

```
   double true_range;
   if(High==Low)true_range=0;                 // 高/安値が等しい場合
   else true_range=(Close-Open)/(High-Low); // 範囲が null でない場合
```

演算子の使用制限

この演算子は、「式 1 」の値に基づいて「式 2 」または「式 3 」の値を返さなければなりません。これらの式には、いくつかの制限があります。

1. ユーザ定義型と基本データ型や列挙を混用してはいけません。ポインタにはNULLを使用出来ます。
2. 値が基本型であれば、一番大きい型のオペレータが使用されます（型キャストをご参照ください）。
3. 値のいずれかが列挙型で 2 つ目が数値型の場合、列挙型は int 型に置き換えられ、2 番目のルールが適用されます。
4. 両方の値が列挙型の場合は、その型が同一である必要があり、演算子は列挙型となります。

ユーザ定義型（クラスまたは構造体）の制約事項は下記です。

a) 型は同一であるか 2 つ目が 1 つ目からの派生型である必要があります。

b) 型が同一（または継承）でない場合、子の型は暗黙的に親にキャストされ、演算子は親の型になります。

c) オブジェクトとポインタを混用してはいけません。式は両方ともオブジェクトまたはポインタでなければいけません。ポインタにはNULL が利用出来ます。

注意事項

条件演算子の結果の型はプログラムのコンパイル時に定義されているので、条件演算子をオーバーロードされた関数の引数として使用する場合はご注意ください。この型は「式 2 」と「式 3 」の型のうち大きい方の型として定義されています。

例：

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

© 2000–2025, MetaQuotes Ltd.

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

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

   func(Expression1?Expression2:Expression3);    // 文字列への暗黙的キャストに警告
   func(!Expression1?Expression2:Expression3);   // 文字列への暗黙的キャストに警告
  }

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

参照

変数の初期化、変数のアクセス権 スコープとライフタイム、オブジェクトの作成と解徐

© 2000–2025, MetaQuotes Ltd.

Switch 演算子

式の値を全ての *case* と比較し、式の値と一致する case に処理の制御を与えます。各*case*の変数は整数定数、リテラル定数または定数式でマークすることが出来ます。定数式は、変数や関数呼び出しを含むことは出来ません。*switch*演算子の式は整数型でなければいけません – intまたはuint。

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

default ラベルでマークされた演算子は式の値がいずれの *case* 演算子の値とも異なる場合に実行されます。*default* は省くことができ、また最後に位置しないことも可能です。*default* が不在で、全ての case 演算子の値が式の値と異なる場合には、何も実行されません。

case キーワードと定数はただのラベルです。*case* の処理の実行後、プログラムは後に続く*break* 演算子まで更に進んで実行されます。これにて後に続くいくつかの演算を結びつけることが可能になります。

定数式はコンパイル時に計算されます。1 つの*switch* 内で複数の定数が同じ値を持つことは出来ません。

例：

```
//--- 例1
switch(x)
  {
   case 'A':
      Print("CASE A");
      break;
   case 'B':
   case 'C':
      Print("CASE B or C");
      break;
   default:
      Print("NOT A, B or C");
      break;
  }

//---  例 2
   string res="";
   int i=0;
   switch(i)
     {
      case 1:
         res=i;break;
      default:
         res="default";break;
      case 2:
```

© 2000–2025, MetaQuotes Ltd.

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

参照

変数の初期化、変数のアクセス権スコープとライフタイム、オブジェクトの作成と解徐

© 2000-2025, MetaQuotes Ltd.

while反復演算子

while 演算子は満たされなければならない条件式と演算子で構成されています。

```
while(式)
  演算子;
```

式が true の場合、式が false になるまで演算子が実行されます。式が false の場合、制御は次の演算子に与えられます。式の値は演算子が実行される前に定義されています。そのために、もし、式が先頭から false である場合は、演算子がまったく実行されません。

注意事項

ループの反復数が多いと予期される場合は IsStopped() 関数を使用して強制的なプログラム終了を確認することをお勧めします。

例:

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

参照

変数の初期化、変数のアクセス権 スコープとライフタイム、オブジェクトの作成と解徐

© 2000-2025, MetaQuotes Ltd.

For 反復演算子

for 演算子は、3 つの式と実行可能な演算子で構成されています。

```
for(式 1 ; 式 2 ; 式 3 )
  演算子;
```

式 1 はループ初期設定を記述します。式 2 はループの終了条件をチェックします。式 2 が true の場合、ループ本体の for が実行されます。ループは式が false になるまで繰り返します。 式 2 がfalse の場合、ループは終了し、制御は次の演算子に与えられます。式 3 は、各反復の後で計算されます。

for 演算子は次の演算子の連続に相当します。

```
式1;
while(式 2 )
  {
  演算子;
  式 3 ;
  };
```

for 演算子では 3 つのうちのどの式も同時に省くことは出来ますが、間にあるセミコロン(;)を省くことは出来ません。式 2 が省略された場合、式 2 が常に true であると見なされます。for(;;) 演算子は while(1) 演算子と同等の連続ループです。式 1 または 3 はそれぞれコンマ(,)演算子によって結合された複数の式から成り立つことが出来ます。

注意事項

ループの反復数が多いと予期される場合は IsStopped() 関数を使用して強制的なプログラム終了を確認することをお勧めします。

例：

```
for(x=1;x<=7000; x++)
  {
   if(IsStopped())
      break;
   Print(MathPower(x,2));
  }
//--- 例 2
for(;!IsStopped();)
  {
   Print(MathPower(x,2));
   x++;
   if(x>10) break;
  }
//--- 例 3
for(i=0,j=n-1;i<n && !IsStopped();i++,j--) a[i]=a[j];
```

参照

変数の初期化、変数のアクセス権 スコープとライフタイム、オブジェクトの作成と解徐

© 2000-2025, MetaQuotes Ltd.

do while 反復演算子

for 及び while ループは各ループの後でなく前に終了条件をチェックします。3 番目の反復演算子 do - while は各ループ反復の後に終了条件をチェックします。ループ本体は常に少なくとも1 回実行されます。

```
do
   演算子;
while(式);
```

まず演算子が実行され、その後で式が計算されます。式がtrueである場合、演算子が再び実行され、これが繰り返されます。式が false になると、反復が終了します。

注意事項

ループの反復数が多いと予期される場合は IsStopped() 関数を使用して強制的なプログラム終了を確認することをお勧めします。

例:

```
//--- フィボナッチ数列を計算する
   int counterFibonacci=15;
   int i=0,first=0,second=1;
   int currentFibonacciNumber;
   do
     {
      currentFibonacciNumber=first+second;
      Print("i = ",i,"  currentFibonacciNumber = ",currentFibonacciNumber);
      first=second;
      second=currentFibonacciNumber;
      i++; // この演算子なしでは無限ループが発生する
     }
   while(i<counterFibonacci && !IsStopped());
```

参照

変数の初期化、変数のアクセス権 スコープとライフタイム、オブジェクトの作成と解徐

© 2000–2025, MetaQuotes Ltd.

Break 演算子

break 演算子は最も近くにある外側の switch、while、do-while 及び for 演算子から抜けます。制御は、終了した処理に続く演算子に与えられます。この演算子の目的のひとつは、変数が特定の値になった場合にループの実行を終了することです。

例:

```
//--- 最初の 0 要素を探す
for(i=0;i<array_size;i++)
  if(array[i]==0)
    break;
```

参照

変数の初期化、変数のアクセス権 スコープとライフタイム、オブジェクトの作成と解徐

© 2000-2025, MetaQuotes Ltd.

Continue 演算子

continue 演算子は、最も近くにある外側の while、do-while または for の反復処理の始まりに制御を与えて、次の繰り返しを呼び出します。この演算子の目的は break 処理の反対です。

例:

```
//--- 0 以外の要素の合計
int func(int array[])
  {
   int array_size=ArraySize(array);
   int sum=0;
   for(int i=0;i<array_size; i++)
     {
      if(a[i]==0) continue;
      sum+=a[i];
     }
   return(sum);
  }
```

参照

変数の初期化、変数のアクセス権 スコープとライフタイム、オブジェクトの作成と解徐

© 2000–2025, MetaQuotes Ltd.

new オブジェクト作成演算子

new 演算子は、対応するサイズのオブジェクトを自動的に作成し、オブジェクトのコンストラクタを呼び出して作成されたオブジェクトの記述子を返します。失敗した場合、この演算子は NULL 定数と比較出来る null 記述子を返します。

new 演算子はクラスオブジェクトのみに適応出来ます。構造体には適応出来ません。

この演算子はオブジェクトの配列の作成には使用できません。オブジェクトの配列の作成には ArrayResize() 関数が使用されます。

例:

```
//+------------------------------------------------------------------+
//|  図の作成                                                        |
//+------------------------------------------------------------------+
void CTetrisField::NewShape()
  {
   m_ypos=HORZ_BORDER;
//--- 7 つの可能な図形から 1 つをランダムに選んで作成する
   int nshape=rand()%7;
   switch(nshape)
     {
      case 0: m_shape=new CTetrisShape1; break;
      case 1: m_shape=new CTetrisShape2; break;
      case 2: m_shape=new CTetrisShape3; break;
      case 3: m_shape=new CTetrisShape4; break;
      case 4: m_shape=new CTetrisShape5; break;
      case 5: m_shape=new CTetrisShape6; break;
      case 6: m_shape=new CTetrisShape7; break;
     }
//--- 描画する
   if(m_shape!=NULL)
     {
      //--- 事前設定
      m_shape.SetRightBorder(WIDTH_IN_PIXELS+VERT_BORDER);
      m_shape.SetYPos(m_ypos);
      m_shape.SetXPos(VERT_BORDER+SHAPE_SIZE*8);
      //--- 描画する
      m_shape.Draw();
     }
//---
  }
```

オブジェクト記述子はメモリアドレスへのポインタではないことには留意すべきです。

new 演算子を使用して作成したオブジェクトは、明示的に delete 演算子を使用して解除されるべきです。

参照

© 2000-2025, MetaQuotes Ltd.

変数の初期化、変数のアクセス権 スコープとライフタイム、オブジェクトの作成と解徐

© 2000–2025, MetaQuotes Ltd.

delete オブジェクト解除演算子

delete 演算子は new 演算子で作成されたオブジェクトを解除し、対応するクラスのデストラクタを呼び出し、オブジェクトによって占有されていたメモリを解放します。既存のオブジェクトの実際の記述子がオペランドとして使用されます。削除操作が実行された後、オブジェクト記述子は無効になります。

例:

```
      //--- 図を解除する
      delete m_shape;
      m_shape=NULL;
      //--- 新しい図を作成する
      NewShape();
```

参照

変数の初期化、変数のアクセス権 スコープとライフタイム、オブジェクトの作成と解徐

© 2000-2025, MetaQuotes Ltd.

関数

全てのタスクはサブタスクに分割することができ、それぞれが直接コードの形で表されるか更に小さいサブタスクに分割することが出来ます。このメソッドは*段階的詳細化*と呼ばれています。関数はサブタスク解決のコードを記述するために使用されます。関数が実行する内容を記述したコードは *関数定義と呼ばれます。*

```
function_header
  {
   instructions
  }
```

最初のブレースの前にあるのは関数定義のヘッダのみで、ブレースに挟まれているのが関数定義の *本体* です。関数ヘッダーは、戻り値の型、名称（識別子）と仮パラメータの記述を含みます。関数に渡されるパラメータの数は限られており、64 を超えることは出来ません。

関数は、プログラムの他の部分から必要に応じて何回でも呼び出すことが出来ます。実際には、戻り値の型、関数識別子とパラメータ型が*関数のプロトタイプ*を構成します。

関数プロトタイプは、関数宣言ではなく関数定義です。戻り型の明示的な宣言と引数の型のリストによって関数呼び出し時に厳密な型チェックと暗黙の型キャストが可能です。非常に多くの場合、コードの可読性を向上させるために関数の宣言はクラス内で行われています。

関数定義は、確実に宣言と一致する必要があります。宣言された関数は全て定義される必要があります。

例:

```
double                          // 戻り値の型
linfunc (double a, double b) // 関数名とパラメータリスト
  {
                                // 複合演算子
   return (a + b);              // 戻り値
  }
```

return 演算子は自己に含まれる式の値を返すことが出来ます。必要であれば、式の値は関数結果の型に変換されます。返せるものとしては、基本データ型、基本構造体、及びオブジェクトポインタがあります。*return* 演算子は、配列、クラスオブジェクトや複合構造体の変数を返すことは出来ません。

値を返さない関数は void 型として記述されるべきです。

例:

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

関数に渡されたパラメータは、その型の定数によって定義された初期値を持つことが出来ます。

例:

© 2000-2025, MetaQuotes Ltd.

```
int somefunc(double a,
             double d=0.0001,
             int n=5,
             bool b=true,
             string s="passed string")
  {
   Print("Required parameter a = ",a);
   Print("Pass the following parameters: d = ",d," n = ",n," b = ",b," s = ",s);
   return(0);
  }
```

パラメータのいずれかが初期値を持つ場合、それ以降の全てのパラメータは、初期値を持たなければいけません。

下記は不正な宣言の例です。

```
int somefunc(double a,
             double d=0.0001,     // 初期値 0.0001 を宣言
             int n,               // 初期値が指定されていない
             bool b,              // 初期値が指定されていない
             string s="passed string")
  {
  }
```

参照

オーバーロード、仮想関数、ポリモーフィズム

© 2000-2025, MetaQuotes Ltd.

関数の呼び出し

以前に記載されていない名称が式の中で表示され左括弧が続く場合、それは文脈上で関数の名称とみなされます。

```
関数名（x1, x2,..., xn）
```

引数（仮パラメータ）は値によって関数に渡されます。つまりx1、⋯、XNの各式が算出されその値が関数に渡されます。式の計算順序と値の読み込みの順序は保証されていません。実行中に、システムは関数に渡される引数の数と型をチェックします。このように関数に対処する方法は値渡しと呼ばれています。

関数呼び出しは、関数によって返される値を値とする式です。上記で説明された関数の型は戻り値の型と一致する必要があります。関数はプログラムのグローバルスコープ　の任意の部分、つまり関数の外側で宣言または記述されることが出来ます。関数は別の関数の内部で宣言されたり記述されたりすることは出来ません。

例：

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

デフォルトパラメータを使用した関数の呼び出しでは、渡されるパラメータのリストは制限されることがありますが、最初のデフォルトパラメータの前では無制限です。

例：

```
void somefunc(double init,
              double sec=0.0001, //初期値を設定する
              int level=10);
//...
somefunc();                       // 不正な呼び方（ 1 番目のパラメータの存在が必要）
somefunc(3.14);                   // 正しい呼び方
somefunc(3.14,0.0002);            // 正しい呼び方
somefunc(3.14,0.0002,10);         // 正しい呼び方
```

関数を呼び出す時は、初期値を持つパラメータも抜かせません。

```
somefunc(3.14, , 10);             // 不正な呼び方（ 2 番目のパラメータが抜かされた）
```

参照

オーバーロード、仮想関数、ポリモーフィズム

© 2000-2025, MetaQuotes Ltd.

パラメータの引き渡し

機械語がサブプログラム(関数)に引数を渡す方法は 2 つあります。1 番目の方法はパラメータの値渡しです。この方法では、引数値が仮関数引数に複製されます。従って、関数内でのこのパラメータの変更は、対応する呼び出しの引数に影響を及ぼしません。

```
//+------------------------------------------------------------------+
//|  パラメータの値渡し                                              |
//+------------------------------------------------------------------+
double FirstMethod(int i,int j)
  {
   double res;
//---
   i*=2;
   j/=2;
   res=i+j;
//---
   return(res);
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int a=14,b=8;
   Print("a and b before call:",a," ",b);
   double d=FirstMethod(a,b);
   Print("a and b after call:",a," ",b);
  }
//--- スクリプト実行の結果
//  a and b before call: 14 8
//  a and b after call: 14 8
```

2 番目の方法は、参照渡しです。この場合、(値でなく)パラメータへの参照が関数パラメータに渡されます。それが、関数の内部で、呼び出しで指定された実際のパラメータを参照するために使用されます。従って、パラメータの変更は関数の呼び出しに使用される引数に影響を与えます。

```
//+------------------------------------------------------------------+
//|  パラメータの参照渡し                                            |
//+------------------------------------------------------------------+
double SecondMethod(int &i,int &j)
  {
   double res;
//---
   i*=2;
   j/=2;
   res=i+j;
//---
   return(res);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int a=14,b=8;
   Print("a and b before call:",a," ",b);
   double d=SecondMethod(a,b);
   Print("a and b after call:",a," ",b);
  }
//+------------------------------------------------------------------+
//--- スクリプト実行の結果
//  a and b before call: 14 8
//  a and b after call: 28 4
```

MQL5 言語では、配列、構造型の変数やクラスオブジェクトは常に参照によって渡されますが、その例外を除いて両方法が使用されます。実際のパラメータ(関数呼び出しで渡される引数)の変化を避けるには const アクセス指定子を使用します。*const* 指定子で宣言された変数の内容を変更しようとするとコンパイラがエラーを生成します。

注意事項

パラメータは逆の順序で関数に渡されること、すなわち、最後のパラメータが最初に計算されて、最後から 2 つ目のパラメータが次に計算されて引き渡されることには留意すべきです。最後に計算されて渡されたパラメータは、括弧を開いた後の最初のパラメータです。

例:

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

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

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

© 2000-2025, MetaQuotes Ltd.

最初の呼び出し（上記の例を参照）では、*i* 変数は最初に文字列の連結に使用されます。

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

ここではその値は変更されません。その後 *i* 変数は *a[i++]* 配列要素の計算に使用されます。インデックス付き配列要素iがアクセスされた時、*i* 変数は 増加されます。そしてこの後初めて最初のパラメータが変更された *i* の値をもって計算されます。

2 番目の呼び出しでは、同じ i の（関数呼び出しの最初の段階で計算された）値が 3 つのパラメータの全てを算出する際に使用されます。最初のパラメータが算出された後で初めて *i* 変数がもう一度変更されます。

参照

変数のアクセス権 スコープとライフタイム、オーバーロード、仮想関数、ポリモーフィズム

© 2000–2025, MetaQuotes Ltd.

関数の多重定義（オーバーロード）

通常、関数名にはその主な目的を反映する傾向があります。原則として、人間によって読まれるのが可能なプログラムには、うまく選択された 識別子 が含まれます。しかし、たまに 1 つの目的のために複数の関数が使用されることがあります。例として、double の配列の平均値を算出する関数と整数の配列の平均値を算出する関数について考えましょう。両方とも AverageFromArray と呼ぶのが便利です。

```
//+------------------------------------------------------------------+
//| double 型配列の平均値の計算                                      |
//+------------------------------------------------------------------+
double AverageFromArray(const double & array[],int size)
  {
   if(size<=0) return 0.0;
   double sum=0.0;
   double aver;
//---
   for(int i=0;i<size;i++)
     {
      sum+=array[i];     // double の合計
     }
   aver=sum/size;        // 合計を要素数で割る
//---
   Print("Calculation of the average for an array of double type");
   return aver;
  }
//+------------------------------------------------------------------+
//| int 型配列の平均値の計算                                         |
//+------------------------------------------------------------------+
double AverageFromArray(const int & array[],int size)
  {
   if(size<=0) return 0.0;
   double aver=0.0;
   int sum=0;
//---
   for(int i=0;i<size;i++)
     {
      sum+=array[i];     // int の合計
     }
   aver=(double)sum/size;// double 型にキャストしてから割る
//---
   Print("Calculation of the average for an array of int type");
   return aver;
  }
```

どちらの関数も Print() 関数を使用してメッセージを出力します。

```
   Print("Calculation of the average for an array of int type");
```

コンパイラは、引数の型と数に応じて必要な関数を選択します。このような選択ルールは*署名マッチングアルゴリズム*と呼ばれます。署名とは、関数宣言で使用される型のリストです。

© 2000-2025, MetaQuotes Ltd.

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int    a[5]={1,2,3,4,5};
   double b[5]={1.1,2.2,3.3,4.4,5.5};
   double int_aver=AverageFromArray(a,5);
   double double_aver=AverageFromArray(b,5);
   Print("int_aver = ",int_aver,"   double_aver = ",double_aver);
  }
//--- スクリプトの結果
// Calculate the average for an array of int type
// Calculate the average for an array of double type
// int_aver= 3.00000000    double_aver= 3.30000000
```

関数オーバーロードとは、パラメータが異なるいくつかの同名の関数を作成する過程です。つまり、関数のオーバーロードバリアントでは引数の数及び/またはその型は異なっていなければなりません。特定の関数バリアントは、関数宣言のパラメータのリストと関数を呼び出す引数のリストの対応関係に基づいて選択されます。

オーバーロードされた関数が呼び出されると、コンパイラに適切な関数を選択するためのアルゴリズムが存在する必要があります。この選択アルゴリズムは存在する型のキャストに基づきます。最良のマッチは一意である必要があります。オーバーロードされた関数は、少なくとも 1 つの引数にとって全てのバリアントの内で最良のマッチである必要があります。同時に、他の全ての引数に対しての一致も他のバリアントに劣らない必要があります。

以下は各引数のマッチングアルゴリズムです。

オーバーロード関数選択アルゴリズム

1. （可能な場合）厳密なマッチングを使用します。
2. 標準型の増加を試します。
3. 標準型キャストを試します。

標準型の増加は、他の標準的な変換よりも優れています。増加とは float を double、bool、char、short 及び enum を intに変換することです。類似の 整数型 配列の型キャストも型キャストに属します。類似の型は次の通りです。シングルバイト整数である bool、char、及び uchar の 3 型。ダブルバイト整数である short と ushort。4 バイト整数の int、uint 及び color。long、ulong、及び datetime。

厳密なマッチングが一番です。このような一貫性を達成するために型キャスト を使用することが出来ます。コンパイラは、あいまいな状況に対処することは出来ません。従って、オーバーロード関数をあいまいにする型の微妙な違いや暗黙の変換に依存してはいけません。

はっきりしない場合には厳格な遵守を確保するために明示的な変換を使用してください。

MQL5 におけるオーバーロード関数の例は ArrayInitialize() 関数の例でご覧になれます。

© 2000-2025, MetaQuotes Ltd.

関数オーバーロードの規則はクラスメソッドのオーバーロードにも適応されます。

システム関数のオーバーロードは可能ですが、コンパイラが正確に必要な関数を選択することが出来なければいけません。例えば、システム関数 MathMax() のオーバーロードの仕方は 4 つありますが、そのうち2 つのみが正しいものです。

例:

```
// 1. オーバーロードは可能 - 関数は内蔵された MathMax() と違う数のパラメータを持つ
double MathMax(double a,double b,double c);

// 2. オーバーロードは不可能
// パラメータ数が異なるが、最終パラメータに初期値がある
// これは呼び出す際にシステム関数の隠蔽につながり、容認されない
double MathMax(double a,double b,double c=DBL_MIN);

// 3. オーバーロードは可能 - パラメータ a と b の型による通常のオーバーロード
double MathMax(int a,int b);

// 4. オーバーロードは不可能
// パラメータの数と種類が元の double MathMax(double a,double b) と同じ
int MathMax(double a,double b);
```

参照

オーバーロード、仮想関数、ポリモーフィズム

© 2000-2025, MetaQuotes Ltd.

演算子の多重定義（オーバーロード）

コードの読み込みと書き込みを容易にするためには、いくつかの演算のオーバーロードが可能です。演算子のオーバーロードは、キーワード operator を使用して書かれます。可能な演算子のオーバーロードは次の通りです。

- 二項 +、-、/、*、%、<<、>>、==、!=、<、>、<=、>=、=、+=、-=、/=、*=、%=、&=、|=、^=、<<=、>>=、&&、||、&、|、^
- 単項 +、-、++、--、!、~
- 代入演算子 =
- インデックス作成演算子 []

操作のオーバーロードは構造体とクラスのように複雑なオブジェクトの（単純な式の形で書かれた）演算の表記を可能にします。より複雑な実装が隠されているので、オーバーロードを使用して演算の式を書くことでソースコードの表示が簡素化します。

例えば、実部と虚部で構成された複素数を見てみましょう。これらは数学で幅広く使用されています。MQL5 言語には複素数を表現するデータ型がありませんが構造体やクラスを使用して新しいデータ型を作成することが可能です。複雑な構造体を宣言し、四則演算を実装する 4 つのメソッドを定義してみます。

```
//+------------------------------------------------------------------+
//|  複素数演算のための構造体                                        |
//+------------------------------------------------------------------+
struct complex
  {
   double            re; // 実部
   double            im; // 虚部
   //--- コンストラクタ
                     complex():re(0.0),im(0.0) {  }
                     complex(const double r):re(r),im(0.0) {  }
                     complex(const double r,const double i):re(r),im(i) {  }
                     complex(const complex &o):re(o.re),im(o.im) { }
   //--- 算術演算
   complex           Add(const complex &l,const complex &r) const;  // 加算
   complex           Sub(const complex &l,const complex &r) const;  // 減算
   complex           Mul(const complex &l,const complex &r) const;  // 乗算
   complex           Div(const complex &l,const complex &r) const;  // 除算
  };
```

さて、コードで複素数を表す変数を宣言して使用することが出来ます。

例えば、

```
void OnStart()
  {
//--- complex 型の変数を宣言して初期化する
   complex a(2,4),b(-4,-2);
   PrintFormat("a=%.2f+i*%.2f,   b=%.2f+i*%.2f",a.re,a.im,b.re,b.im);
//--- 2 つの数を足す
```

© 2000-2025, MetaQuotes Ltd.

```
   complex z;
   z=a.Add(a,b);
   PrintFormat("a+b=%.2f+i*%.2f",z.re,z.im);
//--- 2 つの数を掛ける
   z=a.Mul(a,b);
   PrintFormat("a*b=%.2f+i*%.2f",z.re,z.im);
//--- 2 つの数を割る
   z=a.Div(a,b);
   PrintFormat("a/b=%.2f+i*%.2f",z.re,z.im);
//---
  }
```

しかし複素数の算術演算に「/」、「+」、「-」、「*」及び「/」の通常の演算子を使用するのが便利です。

Keyword 演算子は型変換を実行するメンバ関数の定義に使用されます。クラスオブジェクトの変数の単項及び二項操作は非静的メンバ関数としてオーバーロードすることが出来ます。これらは暗黙的にクラスオブジェクトに使用されます。

ほとんどの二項演算は、クラス変数、またはこのクラスオブジェクトへのポインタなどの 1 つまたは両方の引数を受ける通常の関数のようにオーバーロードすることが出来ます。例の複素数型は、宣言でオーバーロードすると、次のようになります。

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

スクリプトの完全な例は下記です。

```
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- complex 型の変数を宣言して初期化する
   complex a(2,4),b(-4,-2);
   PrintFormat("a=%.2f+i*%.2f,   b=%.2f+i*%.2f",a.re,a.im,b.re,b.im);
   //a.re=5;
   //a.im=1;
   //b.re=-1;
   //b.im=-5;
//--- 2 つの数を足す
   complex z=a+b;
   PrintFormat("a+b=%.2f+i*%.2f",z.re,z.im);
//--- 2 つの数を乗算する

   z=a*b;
   PrintFormat("a*b=%.2f+i*%.2f",z.re,z.im);
//--- 2 つの数を割る
   z=a/b;
```

© 2000-2025, MetaQuotes Ltd.

```
   PrintFormat("a/b=%.2f+i*%.2f",z.re,z.im);
//---
  }
//+------------------------------------------------------------------+
//| 複素数演算のための構造体                                         |
//+------------------------------------------------------------------+
struct complex
  {
   double            re; // 実部
   double            im; // 虚部
   //--- コンストラクタ
                     complex():re(0.0),im(0.0) {  }
                     complex(const double r):re(r),im(0.0) {  }
                     complex(const double r,const double i):re(r),im(i) {  }
                     complex(const complex &o):re(o.re),im(o.im) { }
   //--- 算術演算
   complex           Add(const complex &l,const complex &r) const;  // 加算
   complex           Sub(const complex &l,const complex &r) const;  // 減算
   complex           Mul(const complex &l,const complex &r) const;  // 乗算
   complex           Div(const complex &l,const complex &r) const;  // 除算
   //--- 二項演算子
   complex operator+(const complex &r) const { return(Add(this,r)); }
   complex operator-(const complex &r) const { return(Sub(this,r)); }
   complex operator*(const complex &r) const { return(Mul(this,r)); }
   complex operator/(const complex &r) const { return(Div(this,r)); }
  };
//+------------------------------------------------------------------+
//| 加算                                                             |
//+------------------------------------------------------------------+
complex complex::Add(const complex &l,const complex &r) const
  {
   complex res;
//---
   res.re=l.re+r.re;
   res.im=l.im+r.im;
//--- 結果
   return res;
  }
//+------------------------------------------------------------------+
//| 減算                                                             |
//+------------------------------------------------------------------+
complex complex::Sub(const complex &l,const complex &r) const
  {
   complex res;
//---
   res.re=l.re-r.re;
   res.im=l.im-r.im;
//--- 結果
   return res;
```

© 2000–2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| 乗算                                                             |
//+------------------------------------------------------------------+
complex complex::Mul(const complex &l,const complex &r) const
  {
   complex res;
//---
   res.re=l.re*r.re-l.im*r.im;
   res.im=l.re*r.im+l.im*r.re;
//--- 結果
   return res;
  }
//+------------------------------------------------------------------+
//| 除算                                                             |
//+------------------------------------------------------------------+
complex complex::Div(const complex &l,const complex &r) const
  {
//--- 空の複素数
   complex res(EMPTY_VALUE,EMPTY_VALUE);
//--- 値がゼロかチェックする
   if(r.re==0 && r.im==0)
     {
      Print(__FUNCTION__+": number is zero");
      return(res);
     }
//--- 補助変数
   double e;
   double f;
//--- 計算バージョンを選択する
   if(MathAbs(r.im)<MathAbs(r.re))
     {
      e = r.im/r.re;
      f = r.re+r.im*e;
      res.re=(l.re+l.im*e)/f;
      res.im=(l.im-l.re*e)/f;
     }
   else
     {
      e = r.re/r.im;
      f = r.im+r.re*e;
      res.re=(l.im+l.re*e)/f;
      res.im=(-l.re+l.im*e)/f;
     }
//--- 結果
   return res;
  }
```

© 2000–2025, MetaQuotes Ltd.

ほとんどの単項演算は、クラス変数、またはこのクラスオブジェクトへのポインタなどの引数を受ける通常の関数のようにオーバーロードすることが出来ます。ここでは単項演算子「 -」と「 ! 」のオーバーロードを追加します。

```
//+------------------------------------------------------------------+
//|  複素数演算のための構造体                                        |
//+------------------------------------------------------------------+
struct complex
  {
   double            re;       // 実部
   double            im;       // 虚部
...
   //--- 単項演算子
   complex operator-()  const; // 単項マイナス
   bool    operator!()  const; // 否定
  };
...
//+------------------------------------------------------------------+
//|  単項マイナス演算子のオーバーロード                              |
//+------------------------------------------------------------------+
complex complex::operator-() const
  {
   complex res;
//---
   res.re=-re;
   res.im=-im;
//--- 結果
   return res;
  }
//+------------------------------------------------------------------+
//|  「論理否定」演算子のオーバーロード                              |
//+------------------------------------------------------------------+
bool complex::operator!() const
  {
//--- 複素数の実部と虚部がゼロ?
   return (re!=0 && im!=0);
  }
```

複素数の値がゼロかをチェックし、負の値を取得します。

```
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- complex 型の変数を宣言して初期化する
   complex a(2,4),b(-4,-2);
   PrintFormat("a=%.2f+i*%.2f,   b=%.2f+i*%.2f",a.re,a.im,b.re,b.im);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 2 つの数を割る
   complex z=a/b;
   PrintFormat("a/b=%.2f+i*%.2f",z.re,z.im);
//--- 複素数はデフォルトではゼロ(デフォルトコンストラクタで re==0 及び im==0)
   complex zero;
   Print("!zero=",!zero);
//--- 負の値を割り当てる
   zero=-z;
   PrintFormat("z=%.2f+i*%.2f,  zero=%.2f+i*%.2f",z.re,z.im, zero.re,zero.im);
   PrintFormat("-zero=%.2f+i*%.2f",-zero.re,-zero.im);
//--- もう1 回値がゼロかチェックする
   Print("!zero=",!zero);
//---
  }
```

基本データ型の構造体 は互いに直接にコピー出来るため、代入演算子「 = 」のオーバーロードは必要がなかったことにご注目ください。このように通常の方法で複素数を含む計算のコードを書くことが出来ます。

索引付け演算子のオーバーロードは、単純かつなじみ深い方法で、オブジェクトで囲まれた配列の値を取得することができ、ソースコードの読みやすさに貢献します。例えば、文字列内の指定された位置にあるシンボルへのアクセスを提供する必要があるとします。MQL5 言語の文字列はシンボルの配列ではない別の string 型ですが、インデックス演算のオーバーロードの助けを借りて、生成された CString クラスで単純かつ分かりやすい利用法を提供することが出来ます。

```
//+------------------------------------------------------------------+
//|  シンボルの配列のように文字列内のシンボルにアクセスするクラス                  |
//+------------------------------------------------------------------+
class CString
  {
   string            m_string;

public:
                     CString(string str=NULL):m_string(str) { }
   ushort operator[] (int x) { return(StringGetCharacter(m_string,x)); }
  };
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 文字列からシンボルを受け取るための配列
   int      x[]={ 19,4,18,19,27,14,15,4,17,0,19,14,17,27,26,28,27,5,14,
                  17,27,2,11,0,18,18,27,29,30,19,17,8,13,6 };
   CString str("abcdefghijklmnopqrstuvwxyz[ ]CS");
   string  res;
//--- str 変数のシンボルを使用してフレーズを作る
   for(int i=0,n=ArraySize(x);i<n;i++)
     {
      res+=ShortToString(str[x[i]]);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 結果を表示する
   Print(res);
  }
```

索引操作のオーバーロードの別例は、行列演算です。行列は配列のサイズが予め定義されていない 2 次元の動的配列を表します。従って、2 次元目のサイズを指定せずに array[][] の形式で配列を宣言することも、パラメータとしてこの配列を渡すことも出来ません。可能な解決策は、CRow クラスオブジェクトの配列が含まれている特別なクラス CMatrix です。

```
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 加算と乗算の行列演算
   CMatrix A(3),B(3),C();
//--- 配列の行を準備する
   double a1[3]={1,2,3}, a2[3]={2,3,1}, a3[3]={3,1,2};
   double b1[3]={3,2,1}, b2[3]={1,3,2}, b3[3]={2,1,3};
//--- 行列に書き込む
   A[0]=a1; A[1]=a2; A[2]=a3;
   B[0]=b1; B[1]=b2; B[2]=b3;
//--- エキスパート操作ログに行列を出力する
   Print("---- Elements of matrix A");
   Print(A.String());
   Print("---- Elements of matrix B");
   Print(B.String());

//--- 行列の加算
   Print("---- Addition of matrices A and B");
   C=A+B;
//--- フォーマットされた文字列表現を出力する
   Print(C.String());

//--- 行列の乗算
   Print("---- Multiplication of matrices A and B");
   C=A*B;
   Print(C.String());

//--- 次に動的配列 matrix[i][j] のスタイルで値を取得する方法を示す
   Print("Output the values of matrix C elementwise");
//--- ループを使用して行列の行（CRow オブジェクト）を順番にみる
   for(int i=0;i<3;i++)
     {
      string com="| ";
      //--- 行列の行の値を形成する
      for(int j=0;j<3;j++)
        {
         //--- 行と列の数を使用して行列要素を取得する
```

© 2000-2025, MetaQuotes Ltd.

```
         double element=C[i][j];// [i] - m_rows[] 配列で CRow にアクセスする
                                // [j] - CRowでのインデックス作成演算オーバーロード
         com=com+StringFormat("a(%d,%d)=%G ; ",i,j,element);
        }
      com+="|";
      //--- 行の値を出力する
      Print(com);
     }
  }
//+------------------------------------------------------------------+
//| Row クラス                                                       |
//+------------------------------------------------------------------+
class CRow
  {
private:
   double            m_array[];
public:
   //--- コンストラクタとデストラクタ
                     CRow(void)           { ArrayResize(m_array,0);     }
                     CRow(const CRow &r)  { this=r;                     }
                     CRow(const double &array[]);
                    ~CRow(void){};
   //--- 行の要素数
   int               Size(void) const     { return(ArraySize(m_array));}
   //--- 値を持つ文字列を返す
   string            String(void) const;
   //--- インデックス作成演算子
   double            operator[](int i) const  { return(m_array[i]);    }
   //--- 代入演算子
   void              operator=(const double  &array[]); // 配列
   void              operator=(const CRow & r);         // あと 1 つの CRow オブジェクト
   double            operator*(const CRow &o);          //乗算に使用される CRow オブジェクト
  };
//+------------------------------------------------------------------+
//| 行を配列と初期化するためのコンストラクタ                         |
//+------------------------------------------------------------------+
void  CRow::CRow(const double &array[])
  {
   int size=ArraySize(array);
//--- 配列が空でない場合
   if(size>0)
     {
      ArrayResize(m_array,size);
      //--- 値を書き込む
      for(int i=0;i<size;i++)
         m_array[i]=array[i];
     }
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 配列の代入演算子                                                 |
//+------------------------------------------------------------------+
void CRow::operator=(const double &array[])
  {
   int size=ArraySize(array);
   if(size==0) return;
//--- 配列に値を書き込む
   ArrayResize(m_array,size);
   for(int i=0;i<size;i++) m_array[i]=array[i];
//---
  }
//+------------------------------------------------------------------+
//| CRow の 代入演算子                                               |
//+------------------------------------------------------------------+
void CRow::operator=(const CRow  &r)
  {
   int size=r.Size();
   if(size==0) return;
//--- 配列に値を書き込む
   ArrayResize(m_array,size);
   for(int i=0;i<size;i++) m_array[i]=r[i];
//---
  }
//+------------------------------------------------------------------+
//| 別の行の乗算演算子                                               |
//+------------------------------------------------------------------+
double CRow::operator*(const CRow &o)
  {
   double res=0;
//--- 検証
   int size=Size();
   if(size!=o.Size() || size==0)
     {
      Print(__FUNCSIG__,": Failed to multiply two matrices, their sizes are
different");
      return(res);
     }
//--- 配列の要素ごとに乗算し、結果を加算する
   for(int i=0;i<size;i++)
      res+=m_array[i]*o[i];
//--- 結果
   return(res);
  }
//+------------------------------------------------------------------+
//| フォーマットされた文字列表現を返す                               |
//+------------------------------------------------------------------+
string CRow::String(void) const
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   string out="";
//--- 配列のサイズがゼロでない場合
   int size=ArraySize(m_array);
//--- 要素の数がゼロ以外の配列を取り扱う
   if(size>0)
     {
      out="{";
      for(int i=0;i<size;i++)
        {
         //--- 値を文字列に収集する
         out+=StringFormat(" %G;",m_array[i]);
        }
      out+=" }";
     }
//--- 結果
   return(out);
  }
//+------------------------------------------------------------------+
//| Matrix クラス                                                    |
//+------------------------------------------------------------------+
class CMatrix
  {
private:
   CRow              m_rows[];

public:
   //--- コンストラクタとデストラクタ
                     CMatrix(void);
                     CMatrix(int rows)  { ArrayResize(m_rows,rows);             }
                    ~CMatrix(void){};
   //--- 行列のサイズを取得する
   int               Rows()       const { return(ArraySize(m_rows));            }
   int               Cols()       const { return(Rows()>0? m_rows[0].Size():0); }
   //--- CRow の行の形で列の値を返す
   CRow              GetColumnAsRow(const int col_index) const;
   //--- マトリックス値を持つ文字列を返す
   string            String(void) const;
   //--- 索引付け演算子は文字列をその数で返す
   CRow *operator[](int i) const        { return(GetPointer(m_rows[i]));        }
   //--- 加算演算子
   CMatrix           operator+(const CMatrix &m);
   //--- 乗算演算子
   CMatrix           operator*(const CMatrix &m);
   //--- 代入演算子
   CMatrix          *operator=(const CMatrix &m);
  };
//+------------------------------------------------------------------+
//| サイズがゼロの行の配列を作成するフォルトコンストラクタ           |
//+------------------------------------------------------------------+
```

© 2000–2025, MetaQuotes Ltd.

```
CMatrix::CMatrix(void)
  {
//--- 行列の行がゼロ
   ArrayResize(m_rows,0);
//---
  }
//+------------------------------------------------------------------+
//| CRow の行の形で列の値を返す                                      |
//+------------------------------------------------------------------+
CRow  CMatrix::GetColumnAsRow(const int col_index) const
  {
//--- 列から値を取得する変数
   CRow row();
//--- 行列の行数
   int rows=Rows();
//--- 行の数がゼロより大きい場合、演算を実行する
   if(rows>0)
     {
      //--- インデックス col_index で列の値を受け取る配列
      double array[];
      ArrayResize(array,rows);
      //---配列に書き込む
      for(int i=0;i<rows;i++)
        {
         //--- 配列の境界を超えるかもしれないので 行 i の列の数を確認する
         if(col_index>=this[i].Size())
           {
            Print(__FUNCSIG__,": Error! Column number ",col_index,"> row size ",i);
            break; // 行は初期化されていないオブジェクトになる
           }
         array[i]=this[i][col_index];
        }
      //--- 配列の値に基づいて CRow の行を作成する
      row=array;
     }
//--- 結果
   return(row);
  }
//+------------------------------------------------------------------+
//| 2 つの行列の加算                                                 |
//+------------------------------------------------------------------+
CMatrix CMatrix::operator+(const CMatrix &m)
  {
//--- 渡された行列の行と列の数
   int cols=m.Cols();
   int rows=m.Rows();
//--- 加算結果を受け取る行列
   CMatrix res(rows);
//--- 行列のサイズが一致していなければいけない
```

© 2000-2025, MetaQuotes Ltd.

```
   if(cols!=Cols() || rows!=Rows())
     {
      //--- 加算が不可能
      Print(__FUNCSIG__,": Failed to add two matrices, their sizes are different");
      return(res);
     }
//--- 補助配列
   double arr[];
   ArrayResize(arr,cols);
//--- 行を 1 つづつ加算する
   for(int i=0;i<rows;i++)
     {
      //--- 行列の文字列の加算結果を配列に書く
      for(int k=0;k<cols;k++)
        {
         arr[k]=this[i][k]+m[i][k];
        }
      //--- 配列を行列の行に配置する
      res[i]=arr;
     }
//--- 行列の加算の結果を返す
   return(res);
  }
//+------------------------------------------------------------------+
//| 2 つの行列の乗算                                                 |
//+------------------------------------------------------------------+
CMatrix CMatrix::operator*(const CMatrix &m)
  {
//--- 1 番目の行列の列数、行列に渡された行数
   int cols1=Cols();
   int rows2=m.Rows();
   int rows1=Rows();
   int cols2=m.Cols();
//--- 加算結果を受け取る行列
   CMatrix res(rows1);
//--- 行列は調整を必要とする
   if(cols1!=rows2)
     {
      //--- 乗算が不可能
      Print(__FUNCSIG__,": Failed to multiply two matrices, format is not compatible
"
            "- number of columns in the first factor should be equal to the number
of rows in the second");
      return(res);
     }
//--- 補助配列
   double arr[];
   ArrayResize(arr,cols1);
//--- 乗算行列の行に書き込む
```

© 2000–2025, MetaQuotes Ltd.

```
   for(int i=0;i<rows1;i++)// 行ごとに処理する
     {
      //--- 受け取り側の配列をリセットする
      ArrayInitialize(arr,0);
      //--- 行の要素を 1 つずつみる
      for(int k=0;k<cols1;k++)
        {
         //--- CRow の形で行列 m の列 k の値をとる
         CRow column=m.GetColumnAsRow(k);
         //--- 2 つの行を乗算し、i 番目の要素内のベクトルのスカラー乗算の結果を書き込む
         arr[k]=this[i]*column;
        }
      //--- 配列 arr[] を行列の i 番目の行に配置する
      res[i]=arr;
     }
//--- 行列の乗算の結果を返す
   return(res);
  }
//+------------------------------------------------------------------+
//| 代入演算                                                         |
//+------------------------------------------------------------------+
CMatrix *CMatrix::operator=(const CMatrix &m)
  {
//--- 行数を検索して設定する
   int rows=m.Rows();
   ArrayResize(m_rows,rows);
//--- 渡された行列の行の値で行を埋める
   for(int i=0;i<rows;i++) this[i]=m[i];
//---
   return(GetPointer(this));
  }
//+------------------------------------------------------------------+
//| 行列の文字列表現                                                 |
//+------------------------------------------------------------------+
string CMatrix::String(void) const
  {
   string out="";
   int rows=Rows();
//---文字列で文字列を作る
   for(int i=0;i<rows;i++)
     {
      out=out+this[i].String()+"\r\n";
     }
//--- 結果
   return(out);
  }
```

参照

© 2000-2025, MetaQuotes Ltd.

多重定義（オーバーロード）、算術演算、関数多重定義、優先順位のルール

© 2000–2025, MetaQuotes Ltd.

外部関数の記述

別のモジュールで定義された外部関数は明示的に記述されるべきです。記述は、戻り型、関数名及び入力パラメータとその型を含みます。そのような記述が存在しないとコンパイル、ビルド、またはプログラムの実行中にエラーが起こることがあります。外部オブジェクトを記述する時は*#import* キーワードを使用してモジュールを示します。

例：

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

import の助けを借りれば、外部 DLL またはコンパイル済みの EX5 ライブラリから呼び出される関数の記述が容易に出来ます。EX5ライブラリは library プロパティを持つ、コンパイル済みの ex5 ファイルです。export 修飾子 のついた関数のみが EX5 ライブラリからインポート出来ます。

DLL と EX5 ライブラリが一緒にインポートされる場合は（配置されているディレクトリに関係なく）別の名称を持つ必要があるのでご注意ください。インポートされた関数は全てライブラリの「ファイル名」に対応するスコープの解像度を持っています。

例：

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

class CFoo
  {
public:
   int            GetLastError() { return(12345); }
   void           func()
     {
      Print(GetLastError());           // クラスメソッドの呼び出し
      Print(::GetLastError());         // MQL5 関数の呼び出し
      Print(kernel32::GetLastError()); // kernel32.dll からの DLL ライブラリの呼び出し
      Print(lib::GetLastError());      // lib.ex5 からの EX5 ライブラリ関数の呼び出し
     }
  };

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

© 2000–2025, MetaQuotes Ltd.

参照

オーバーロード、仮想関数、ポリモーフィズム

© 2000-2025, MetaQuotes Ltd.

関数のエクスポート

MQL5 プログラムで *export* 後置修飾語とともに宣言された関数は別の MQL5 プログラムで使用することが出来ます。このような関数は、エクスポータブル（エクスポート可能）と呼ばれ、コンパイル後に、他のプログラムから呼び出すことが出来ます。

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

この修飾子は、この EX5 ファイルによってエクスポートされる EX5 関数の表にこの関数を追加するようにコンパイラに指示します。この修飾子を持った関数のみが他の MQL5 プログラムから使用可能また可視です。

library プロパティは、この EX5 ファイルがライブラリになることをコンパイラに伝え、コンパイラがそれを EX5 のヘッダに表示します。

エクスポート可能として計画されている全ての関数は、*export* 修飾子でマークされる必要があります。

参照

オーバーロード、仮想関数、ポリモーフィズム

© 2000–2025, MetaQuotes Ltd.

イベント処理の関数

MQL5 言語はいくつかの定義済みイベントの処理を提供します。これらのイベントを処理するための関数は MQL5 プログラムで定義されている必要があります。関数名、戻り値の型、パラメータの組成物（もしあれば）と型は、イベントハンドラ関数の記述に厳格に適合しなければなりません。

クライアント端末のイベントハンドラは、戻り値とパラメータの型によって、各イベントハンドラ関数を識別します。以下の説明に対応していないパラメータが指定された場合、また別の戻り値の型が使用された場合、その関数はイベントハンドラとして使用されません。

OnStart

OnStart() 関数は Start イベントハンドラで**スクリプトの実行のみ**に自動的に生成されます。この関数は void 型でパラメータを持ってはいけません。

```
void OnStart();
```

OnStart() 関数では int 型の戻り値を指定することが出来ます。

OnInit

OnInit() 関数は Init イベントハンドラです。この関数は void または int 型でパラメータを持ちません。

```
void OnInit();
```

Initイベントは、エキスパートアドバイザーや指標がダウンロードされた直後に生成されます。このイベントは、スクリプトには生成されません。OnInit() 関数は初期化に使用されます。OnInit() がint 型の戻り値を持つ場合、ゼロ以外のリターンコードは失敗した初期化を意味し初期化解除の理由である REASON_INITFAILED のコードを含む Deinit イベントが生成されます。

INIT_FAILEDを返すと、EAはチャートから強制的にアンロードされます。

INIT_FAILEDを返すと、指標はチャートからアンロードされません。チャート上に残っている指標は動作しません。イベントハンドラは指標内で呼び出されません。

エキスパートアドバイザーの入力パラメータを最適化するためには、リターンコードとして ENUM_INIT_RETCODE 列挙の値を使用することをお勧めします。これらの値は、最適なテストエージェントの選択を含んだ、最適化の過程を整理するために使用されています。テストの開始前に起こるエキスパートアドバイザーの初期化中には TerminalInfoInteger() 関数を使用してエージェントの構成とリソース（コア数、空きメモリの量など）に関する情報を要求することが出来ます。取得された情報に基づいて、エキスパートアドバイザーの最適化の際にこのテストエージェントを使用するかどうかを決めることが出来ます。

ENUM_INIT_RETCODE

識別子	説明
INIT_SUCCEEDED	初期化が成功し、エキスパートアドバイザーのテストを続けることが出来ます。 このコードは NULL 値と同じ意味を持ちます。エキスパートアドバイザーはテスタ内で正常に初期化されています。
INIT_FAILED	初期化に失敗しました。致命的なエラーが起こったのでテストを続ける理由がありません。例えば、エキスパートアドバイザーが機能す

© 2000-2025, MetaQuotes Ltd.

識別子	説明
	るのに必要とされる指標の作成に失敗しました。 この戻り値はゼロ以外の値と同じ意味です。 テスタ内でのエキスパートアドバイザーの初期化に失敗しました。
INIT_PARAMETERS_INCORRECT	この値は、入力パラメータの不正確さを意味します。このリターンコードを含む結果文字列は、「一般的な最適化」の表に赤色で強調表示されます。 所定のパラメータセットのエキスパートアドバイザーテストは実行されません。エージェントが新しいタスクを受け取ることが出来ます。 この値を受け取ると、ストラテジーテスターはこのタスクを他のエージェントに再試行のために渡さないことがあります。
INIT_AGENT_NOT_SUITABLE	初期化中にエラーはありませんでしたが、エージェントが何らかの理由でテストに適していません。例えば、メモリ不足、OpenCL サポートの不足などです。 このコードが返された後は、エージェントは この最適化が終了するまでタスクを受信しません。

void型の OnInit() 関数は常時初期化の成功を示します。

OnDeinit

OnDeinit() 関数は初期化解除中に呼び出され Deinit イベントハンドラでもあります。この関数は void 型して宣言する必要があり const int 型のパラメータを 1 つ持つ必要があります。このパラメータは 初期化解除の理由のコードです。 違う型が宣言された場合、コンパイラは警告を生成しますが、関数は呼び出されません。スクリプトでは Deinit イベントが生成されないので、OnDeinit() 関数はスクリプトでは使用出来ません。

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

以下の場合 Deinit イベントがエキスパートアドバイザーと指標に生成されます。

- シンボルまたはチャートの期間の変更に伴う再初期化の前にMQL5 プログラムが接続されている場合
- 入力パラメータの変更に伴う再初期化の前
- プログラムをアンロードする前

OnTick

NewTick イベントは、エキスパートアドバイザーが接続されたチャートにシンボルの新しいティックが受け取られた時に**エキスパートアドバイザのみ** に作成されます。カスタム指標またはスクリプトでは NewTick イベントが生成されないので OnTick() 関数を定義するのは無意味です。

Tick イベントは、エキスパートアドバイザーのために生成されますが、エキスパートアドバイザーには NewTick イベントの他に Timer、BookEvent とChartEvent も生成されるので、エキスパートアドバイザーが 必ずしも OnTick() 関数を必要とするわけではありません。この関数は void 型でパラメータを持ちません。

```
void OnTick();
```

OnTimer

© 2000–2025, MetaQuotes Ltd.

OnTimer() 関数は Timer イベントがエキスパートアドバイザーと指標のシステムタイマーによって生成される時に呼ばれます。スクリプトでは利用出来ません。イベントの発生頻度は EventSetTimer() 関数がこのイベントを受け取る通知にサブスクライブする際に設定されます。

特定のエキスパートアドバイザーの timer イベント通知解除にはEventKillTimer() 関数を使用します。この関数は void 型でパラメータを持ちません。

```
void OnTimer();
```

OnInit() 関数で EventSetTimer() 関数を一度呼び出すことがお勧めです。また OnDeinit()で EventKillTimer() 関数を一度呼び出すことも必要です。

エキスパートアドバイザーならびに指標は、全て独自のタイマーで動作しそこからのみイベントを受けとります。MQL5 プログラムが停止した場合、作成されて EventKillTimer() 関数で無効にされていないタイマーは強制的に破壊されます。

OnTrade

この関数は Trade イベントの発生時に呼ばれます。 このイベントは 出された注文、ポジション、注文履歴と約定履歴の表を変更する時に発生します。取引活動（未決注文を出す、注文を出す、ポジションの決済、ストップ設定、未決注文トリガ）が行われると、注文履歴や約定履歴及びポジションと現在の注文のリストがそれに応じて変更されます。

```
void OnTrade();
```

このようなイベントが受け取られた場合（取引ストラテジー条件によって必要な場合）、コード内の取引口座状態を検証するコードを実装する必要があります。OrderSend() 関数の呼び出しが正常に完了し true が返された場合、取引サーバが注文を実行のためにキューに入れてチケット番号を割り当てたことが意味されます。サーバがこの注文を処理するとすぐに Trade イベントが生成されます。チケット値があれば OnTrade() イベント処理の際にこの値を使用して注文の状態を調べることが出来ます。

OnTradeTransaction

取引口座にいくつかの明確なアクションを実行すると、状態が変更されます。アクションの例は下記です。

- クライアント端末内の任意の MQL5 アプリケーションから OrderSend と OrderSendAsync 関数を使用して取引リクエストを送信し、取引が実行される。
- 端末のグラフィカルインターフェースを使用して取引リクエストを送信し、取引が実行される。
- サーバ上で未決注文と逆指値注文が執行される。
- 取引サーバ側で操作を行う。

これらのアクションの結果として以下の取引トランザクションが実行されます。

- 取引リクエストの処理
- 未執行注文の変更
- 注文履歴の変更
- 約定履歴の変更
- ポジションの変更

例えば、買いの成行注文を送信すると、注文が処理され、買い注文が口座に記録されます。その後注文が実行されオープン注文の表から削除されて注文履歴に追加されます。 約定が履歴に追加され新しいポ

© 2000–2025, MetaQuotes Ltd.

ジションが作成されます。これらのアクションは全て取引トランザクションです。このようなトランザクションの端末への到着が TradeTransaction イベントです。それは OnTradeTransaction ハンドラを呼びます。

```
void  OnTradeTransaction(
   const MqlTradeTransaction&    trans,        // 取引トランザクション構造体
   const MqlTradeRequest&        request,      //リクエスト構造体
   const MqlTradeResult&         result        // 結果構造体
   );
```

ハンドラには 3 つのパラメータがあります。

- **trans** – このパラメータは取引口座に適用される取引トランザクションを記述する MqlTradeTransaction 構造体を取得します。
- **request** – このパラメータは取引リクエストを記述する MqlTradeRequest 構造体を取得します。
- **result** – このパラメータは取引リクエスト実行の結果を記述する MqlTradeResult 構造体を取得します。

最後の 2 つの **request** と **result** パラメータは TRADE_TRANSACTION_REQUEST 型のトランザクションでのみ値が書き入れられます。トランザクションのデータは **trans** 変数の *type* パラメータから受け取られます。この場合 **result** 変数の *request_id* フィールドは **request** 取引リクエストの ID を含んでいます。実行後は **trans** 変数で記述された 取引トランザクションが終了しています。リクエスト ID は実行されるアクション（OrderSend または OrderSendAsync 関数の呼び出し）と OnTradeTransaction() に送られたアクションの結果を関連付けることが出来ます。

手動で端末から送信された、または OrderSend()/OrderSendAsync() 関数によってリクエストされた 1 つの取引リクエストが取引サーバ上で複数の連続したトランザクションを生成することがあります。端末におけるこれらのトランザクションの到着の優先順位は保証されません。よって取引アルゴリズムを開発する際にトランザクション到着の順番を仮定すべきではありません。

- 全ての取引トランザクションの種類は ENUM_TRADE_TRANSACTION_TYPE 列挙に記述されています。
- 取引トランザクションを記述する MqlTradeTransaction 構造体はトランザクションの種類によって様々な方法で書き込まれます。例えば TRADE_TRANSACTION_REQUEST 型のトランザクションでは type フィールド（取引トランザクションの種類）の分析のみが必要です。OnTradeTransaction 関数の 1、2 番目のパラメータ（ request と result ）は追加のデータを得るために分析されなければいけません。詳しくは 取引トランザクション構造体をご覧ください。
- 取引トランザクションの記述は注文、約定やポジションに関する全ての入手可能な情報の配信を行いません（例えば、コメント）。OrderGet*、HistoryOrderGet*、HistoryDealGet* また PositionGet* 関数は繊細を取得するために使用されます。

取引トランザクションは、クライアント口座への適用後、端末の取引トランザクションのキューに置かれます。ここから端末への到着順に OnTradeTransaction エントリーポイントに送られます。

OnTradeTransaction ハンドラを使用してエキスパートアドバイザーによって取引トランザクションを取り扱う場合、端末は新たに到着した取引トランザクションの取り扱いを継続します。そのため、取引口座の状態はすでに OnTradeTransaction 動作中に変更することがあります。例えば、MQL5 プログラムが新しい注文を追加するイベントを処理している間に、注文が、実行可能の未執行注文のリストから削除され、履歴に移動されることもあります。更にアプリケーションはこれらのイベントの通知を受けます。

トランザクションキューは 1,024 要素を備えます。OnTradeTransaction が新しいトランザクションを処理するのに長くかかりすぎると、代わりにキュー内の新しいトランザクションが処理されることがあります。

© 2000–2025, MetaQuotes Ltd.

- 一般的に言って、OnTrade と OnTradeTransaction の呼び出しの数の正確な比率はありません。1つの OnTrade 呼び出しは 1 つまたは複数の OnTradeTransaction に対応します。
- OnTrade は当該な OnTradeTransaction の後で呼び出されます。

OnTester

OnTester() 関数は自動的に選択された区間のエキスパートアドバイザーの履歴テストの後に生成される Tester イベントのハンドラです。この関数は double 型でパラメータを 2 つ持ちます。

```
double OnTester();
```

この関数は OnDeinit() 呼び出しの直前に呼び出され、同様に double の戻り値を返します。OnTester() はエキスパートアドバイザーのテストに使用出来ます。この関数の主な目的は入力パラメータの遺伝的最適化におけるカスタム最大基準として使用される特定の値を計算することです。

遺伝的最適化では一世代内の結果に降順のソートが適用されます。すなわち最適化基準の観点からすると、最良の結果は（OnTester 関数によって返されるカスタム最大の最適化基準値が考慮されるので）最大値を有するものです。このようなソートでは最悪値が終わりに配置されるので次の世代の形成に関与しません。

OnTesterInit

この関数は、 TesterInitイベントが発生してストラテジーテスターでの最適化に先立って必要なアクションが実行されるときに、EAで呼び出されます。この関数の型は 2 つあります。

結果を返すバージョン

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

戻り値

int型の値。ゼロは、最適化が開始される前にチャート上で起動されたEAの初期化に成功したことを意味します。

実行結果を返すOnTesterInit() を呼び出すと、プログラムを初期化できるだけでなく、最適化が異常に終了した場合にはエラーコードが返されるため、これを使用することをお勧めします。INIT_SUCCEEDED (0)以外の戻り値はエラーを意味し、最適化は行われません。

結果を返さないバージョンは古いコードとの相互性の為のみに残されています。使用は推奨されません。

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

最適化が始まると、エキスパートアドバイザーは OnTesterDeinit() または OnTesterPass() ハンドラと共に自動的にテスタの別の端末チャートに指定された銘柄と期間と読み込み、TesterInit イベントを受けとります。この関数は最適化の開始前にエキスパートアドバイザーの初期化に使用され、更に 最適化の結果の処理をします。

OnTesterPass

OnTesterPass() 関数は、ストラテジーテスターにおけるエキスパートアドバイザーの最適化の間にフレームが受信された際に自動的に生成されるTesterPass イベントのハンドラです。この関数は void 型で定義される必要があります。パラメータはありません。

© 2000–2025, MetaQuotes Ltd.

```
void OnTesterPass();
```

OnTesterPass() ハンドラを持つエキスパートアドバイザーは自動的にテスタの別の端末チャートにテストのために指定された銘柄と期間と読み込まれ、最適化中にフレームが受信された時に TesterPass イベントを受け取ります。この関数は最適化の結果 をその完了を待たずに動的に「その場で」処理するために使用されます。フレームは FrameAdd() 関数を用いて追加されます。この関数は OnTester() ハンドラ—のシングルパスが完了した後で呼ぶことが出来ます。

OnTesterDeinit

OnTesterDeinit() は ストラテジーテスターでのエキスパートアドバイザーの最適化の終了時に自動的に生成される TesterDeinit イベントのハンドラです。 この関数は void 型で定義される必要があります。パラメータはありません。

```
void OnTesterDeinit();
```

TesterDeinit() ハンドラを持つエキスパートアドバイザーは最適化の開始時に自動的にチャートに読み込まれ、終了時にTesterDeinit を受け取ります。この関数は全ての最適化の結果の最終処理のために使用されます。

OnBookEvent

OnBookEvent() 関数は BookEvent ハンドラです。BookEvent は板情報の変更時にエキスパートアドバイザーと指標のために生成されます。この関数は void 型で、1 つの文字列型のパラメータを持ちます。

```
void OnBookEvent (const string& symbol);
```

任意のシンボルの BookEvent イベントを受け取るには MarketBookAdd() 関数を使用して、このシンボルの BookEvent イベント受信を事前に設定する必要があります。特定のシンボルの BookEvent イベントの受け取りを停止するには MarketBookRelease() を呼び出します。

他のイベントとは異なり BookEvent イベントはブロードキャストされます。これは、1 つのエキスパートアドバイザーが MarketBookAdd を使用して BookEvent イベントの受信に加入した場合、OnBookEvent() ハンドラを持っている他のエキスパートアドバイザーの全てがこのイベントを受け取ることを意味します。よってハンドラに *const string& symbol* パラメータとして渡された銘柄名を分析する必要があります。

OnChartEvent

OnChartEvent() はChartEvent イベントグループのハンドラです。

- CHARTEVENT_KEYDOWN — チャートウィンドウがフォーカスされた時のキーストロークのイベント
- CHARTEVENT_MOUSE_MOVE — マウス移動イベントとマウスクリックイベント（CHART_EVENT_MOUSE_MOVE=true がチャートに設定された場合）
- CHARTEVENT_OBJECT_CREATE — グラフィックオブジェクトの作成イベント（CHART_EVENT_OBJECT_CREATE=true がチャートに設定された場合）
- CHARTEVENT_OBJECT_CHANGE — プロパティダイアログを介してのオブジェクトプロパティ変更イベント
- CHARTEVENT_OBJECT_DELETE — グラフィックオブジェクト削除イベント（CHART_EVENT_OBJECT_DELETE=trueがチャートに設定された場合）
- CHARTEVENT_CLICK — チャート上でのマウスクリックイベント
- CHARTEVENT_OBJECT_CLICK — チャートに属するグラフィックオブジェクトでのマウスクリックイベント
- CHARTEVENT_OBJECT_DRAG — マウスを使用してのグラフィカルオブジェクトの移動イベント

© 2000–2025, MetaQuotes Ltd.

- CHARTEVENT_OBJECT_ENDEDIT — LabelEditグラフィックオブジェクトの入力ボックスに完成したテキスト編集のイベント
- CHARTEVENT_CHART_CHANGE — チャート変更イベント
- CHARTEVENT_CUSTOM+n — ユーザイベントID（nは0～65535）
- CHARTEVENT_CUSTOM_LAST — カスタムイベントの最大の使用可能なID（CHARTEVENT_CUSTOM+65535）

この関数はエキスパートアドバイザーと指標のみに呼ばれます。この関数はvoid型で4つのパラメータを持ちます。

```
void OnChartEvent(const int id,          // イベント識別子
                  const long& lparam,    // long型イベントパラメータ
                  const double& dparam,  // double型イベントパラメータ
                  const string& sparam   // string型イベントパラメータ
  );
```

OnChartEvent()関数の入力パラメータはイベントの種類ごとにイベント処理に必要とされる一定の値を有しています。これらのパラメータを使用して渡されたイベントと値は以下の表に記載されています。

イベント	id パラメータ値	lparam パラメータ値	dparam パラメータ値	sparam パラメータ値
キーストロークイベント	CHARTEVENT_KEYDOWN	押されたキーのコード	繰り返し回数（ユーザのキーの長押しがキーストロークの繰り返しとみなされた場合の繰り返し回数）	キーボードボタンの状態を記述するビットマスクの文字列値
マウスイベント（CHART_EVENT_MOUSE_MOVE=true がチャートに設定された場合）	CHARTEVENT_MOUSE_MOVE	X座標	Y座標	マウスボタンの状態を記述するビットマスクの文字列の値
グラフィックオブジェクトの作成イベント（CHART_EVENT_OBJECT_CREATE=true がチャートに設定された場合）	CHARTEVENT_OBJECT_CREATE	—	—	作成したグラフィックオブジェクトの名称
プロパティダイアログでのオブジェクトプロパティ変更イベント	CHARTEVENT_OBJECT_CHANGE	—	—	変更したグラフィックオブジェクトの名称
グラフィックオブジェクト削除イベ	CHARTEVENT_OBJECT_DELETE	—	—	削除したグラフィックオブジェクトの名

© 2000–2025, MetaQuotes Ltd.

イベント	id パラメータ値	lparam パラメータ値	dparam パラメータ値	sparam パラメータ値
ント（チャートで CHART_EVENT_OBJECT_DELETE=true の場合）				称
チャート上でのマウスクリックイベント	CHARTEVENT_CLICK	X 座標	Y 座標	—
チャートに属するグラフィックオブジェクトでのマウスクリックイベント	CHARTEVENT_OBJECT_CLICK	X 座標	Y 座標	イベントが発生したグラフィック·オブジェクトの名称
マウスを使用してグラフィックオブジェクトのドラッグイベント	CHARTEVENT_OBJECT_DRAG	—	—	移動したグラフィックオブジェクトの名称
LabelEdit グラフィックオブジェクトの入力ボックスでのテキスト編集完成のイベント	CHARTEVENT_OBJECT_ENDEDIT	—	—	テキスト編集が完成した LabelEdit グラフィックオブジェクトの名称
チャート変更イベント	CHARTEVENT_CHART_CHANGE	—	—	—
N 番号を使用したユーザイベント ID	CHARTEVENT_CUSTOM+N	EventChartCustom() 関数で設定された値	EventChartCustom() 関数で設定された値	EventChartCustom() 関数で設定された値

OnCalculate

OnCalculate() 関数は Calculate イベントによって指標値の計算が必要な時にカスタム指標でのみ呼ばれます。これは通常指標が計算されたシンボルの新しいティックが受け取られた時に起こります。この指標はこのシンボルの価格表に接続される必要はありません。

OnCalculate() 関数は int 型の戻り値を持たなければなりません。2 つの定義が可能です。1 つの指標に対してこの関数の両バージョンを使用することは出来ません。

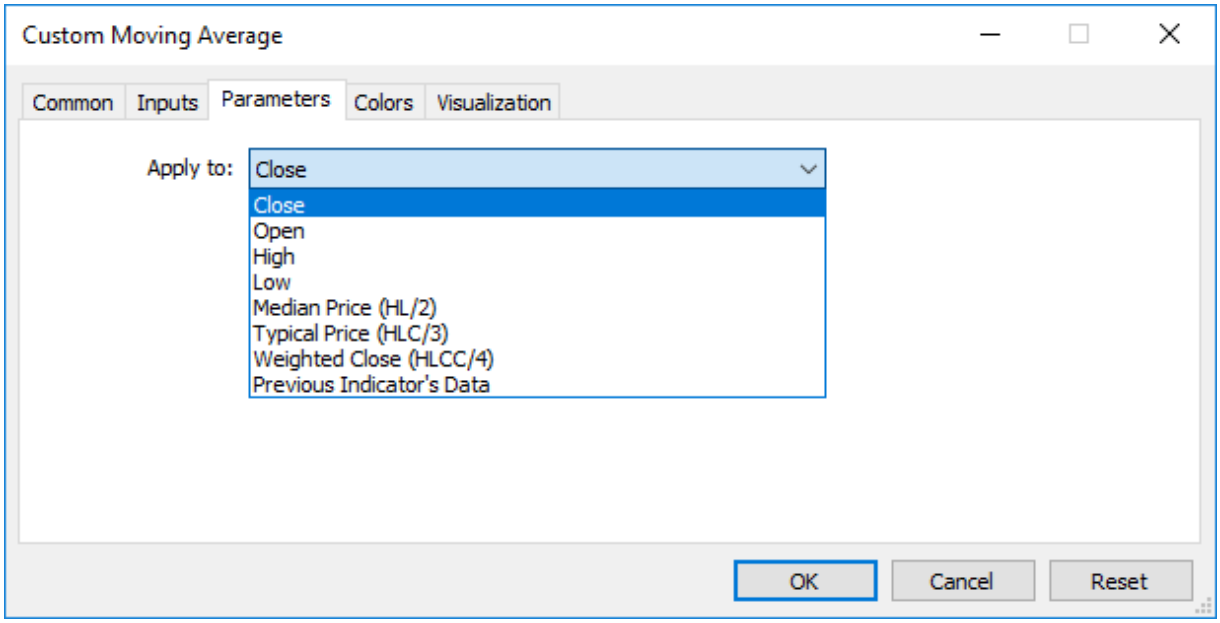
1 番目の定義は単一のデータバッファに基づいて計算することが出来る指標ために意図されています。カスタム平均移動線はこの種類の指標の一例です。

```
int OnCalculate (const int rates_total,      // price[] 配列のサイズ
                 const int prev_calculated,  // 以前の呼び出しで処理されたバー
                 const int begin,            // 重要データの開始点
                 const double& price[]       // 計算対象の配列
   );
```

© 2000–2025, MetaQuotes Ltd.

price[] 配列のように、時系列またはいくつかの指標の計算されたバッファが渡されます。price[] 配列内の索引付けの方向を決定するためには ArrayGetAsSeries()を呼び出します。初期値への依存を避けるためにはこれらの配列に対して ArraySetAsSeries() 関数を無条件に呼び出す必要があります。

price[] 配列として使用される必要な時系列または指標は指標開始時に「パラメータ(Parameters)」タブでユーザ選択が出来ます。そのためには「適用(Apply to)」フィールドのドロップダウンリストに必要な項目を指定する必要があります。



他の MQL5 プログラムからカスタム指標の値を受信するには、その後の操作のための指標ハンドルを返す iCustom() 関数が使用されます。また、当該なprice[] 配列または他の指標ハンドルを指定することが出来ます。このパラメータは、カスタム指標の入力変数のリストに最後に送信されるべきです。
例：

```
void OnStart()
  {
//---
   string terminal_path=TerminalInfoString(STATUS_TERMINAL_PATH);
   int handle_customMA=iCustom(Symbol(),PERIOD_CURRENT, "Custom Moving
Average",13,0, MODE_EMA,PRICE_TYPICAL);
   if(handle_customMA>0)
      Print("handle_customMA = ",handle_customMA);
   else
      Print("Cannot open or not EX5 file '"+terminal_path+"\\MQL5\\Indicators\
\"+"Custom Moving Average.ex5'");
  }
```

この例では、最後のパラメータは PRICE_TYPICAL 値です(ENUM_APPLIED_PRICE 列挙から)。これは、カスタム指標が(高値+安値+終値)/3 で取得された典型的な価格に基づいて構築されることを示します。このパラメータが指定されていない場合は、指標は PRICE_CLOSE 値、すなわち各足の終値に基づいて構築されます。

price[] 配列を指定するために最後のパラメータに指標ハンドラを渡すもう 1 つの例は iCustom() 関数の説明にてみられます。

© 2000-2025, MetaQuotes Ltd.

2 番目の定義は、計算のために複数の時系列を使用する指標のために意図されています。

```
int OnCalculate (const int rates_total,      // 入力時系列のサイズ
                 const int prev_calculated,  // 以前の呼び出しで処理されたバー
                 const datetime& time[],     // 時間
                 const double& open[],       // 始値
                 const double& high[],       // 高値
                 const double& low[],        // 安値
                 const double& close[],      // 終値
                 const long& tick_volume[],  // ティックボリューム
                 const long& volume[],       // ボリューム
                 const int& spread[]         // スプレッド
   );
```

open[]、high[]、low[] 及び close[] パラメータは現在の時間軸での始値、高/安値と終値の配列を含みます。time[] パラメータは オープンタイム値の配列、spread[] パラメータはスプレッドの履歴の配列（スプレッドがセキュリティの取引のために提供されている場合）を含みます。volume[] とtick_volume[] パラメータはそれぞれ、取引量とティックボリュームの履歴を含みます。

time[]、open[]、high[]、low[]、close[]、tick_volume[]、volume[] 及び spread[] の索引付けの方法を決定するには ArrayGetAsSeries() を呼び出します。初期値への依存を避けるためにはこれらの配列に対して ArraySetAsSeries() 関数を無条件に呼び出す必要があります。

初めの rates_total パラメータは指標計算に使用出来る、チャートで使用可能なバーの数に相当するバーの数を含みます。

ここで OnCalculate() の戻り値と prev_calculated の 2 番目の入力パラメータとの関係が指摘されるべきです。関数呼び出しの際、prev_calculated パラメータは以前の OnCalculate() 呼び出しに返された 値を持ちます。これは、この関数の前回の実行以降に変更されていない足の反復計算を避けてカスタム指標を計算するための経済的なアルゴリズムを可能にします。

このために、通常は現在の関数呼び出しでの足数を含む rates_total パラメータの値を返すので充分です。価格データが OnCalculate() の最後の呼び出し以降に変更された場合は（履歴ダウンロードや履歴ブランクの書き込み）、端末がprev_calculated 入力パラメータの値をゼロに設定します。

注意事項：OnCalculate がゼロを返した場合、指標値はクライアント端末のデータウィンドウには表示されません。

理解を深めるために、以下にコードが添付されている指標を起動してみてもいいでしょう。

指標の例：

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- Line をプロットする
#property indicator_label1  "Line"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDarkBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 指標バッファ
```

© 2000-2025, MetaQuotes Ltd.

```
double         LineBuffer[];
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 指標バッファマッピング
   SetIndexBuffer(0,LineBuffer,INDICATOR_DATA);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime& time[],
                const double& open[],
                const double& high[],
                const double& low[],
                const double& close[],
                const long& tick_volume[],
                const long& volume[],
                const int& spread[])
  {
//--- 現在のシンボルとチャート期間に利用可能なバーの数を取得する
   int bars=Bars(Symbol(),0);
   Print("Bars = ",bars,", rates_total = ",rates_total,",  prev_calculated =
",prev_calculated);
   Print("time[0] = ",time[0]," time[rates_total-1] = ",time[rates_total-1]);
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

参照

プログラムの実行、クライアント端末イベント、イベントの操作

© 2000–2025, MetaQuotes Ltd.

変数

変数の宣言

変数は、使用する前に宣言する必要があります。変数を識別するためには、一意の名称が使用されます。変数を宣言するには、型と一意の名称の指定が必要です。変数の宣言は演算ではありません。

基本データ型は下記の通りです。

- char、short、int、long、uchar、ushort、uint、ulong の整数
- color – RGB 色を表す整数
- datetime – 日時（1970年1月1日0時からの秒数を含む符号なしの整数）
- bool – ブール値（*true* と *false*）
- double – 倍精度浮動小数点数
- float – 単精度浮動小数点数
- string – 文字列

例：

```
string szInfoBox;
int    nOrders;
double dSymbolPrice;
bool   bLog;
datetime tBegin_Data   = D'2004.01.01 00:00';
color    cModify_Color = C'0x44,0xB9,0xE6';
```

複雑または複合型は次の通りです。

構造体は、他の型を使用して構築された複合データ型です。

```
struct MyTime
  {
   int hour;    // 0-23
   int minute;  // 0-59
   int second;  // 0-59
  };
...
MyTime strTime; // 以前に宣言された MyTime 構造体の変数
```

構造体を宣言するまで、その構造体型の変数を宣言することは出来ません。

配列

配列は同じ型のデータをインデックス付きで収めます。

```
int    a[50];        // 50 の整数の 1 次元配列
double m[7][50];     // 7 列の 2 次元配列
                     // 各列が 50 の数を含む
MyTime t[100];       // MyTime のような要素を含む配列
```

© 2000-2025, MetaQuotes Ltd.

配列のインデックスとなれるのは整数のみです。4 次元以下の配列のみが使用可能です。配列要素の番号付けは 0 から始まります。1 次元配列の最後の要素は、配列のサイズより 1 小さいインデックスを有します。例えば 50 の整数からなる配列の最後の要素の呼び出しは a[49] となります。多次元配列の次元も同じように 0 から次元サイズ－1までのインデックスを持ちます。例の 2 次元配列の最後の要素は m[6][49] となります。

静的配列は時系列としては表すことが出来ません。末尾から先頭に向かっての配列要素アクセスを設定する ArraySetAsSeries() 関数は静的配列に適応出来ません。時系列と同じアクセスを提供する場合には動的配列オブジェクトが使用されるべきです。

配列の範囲以外のアドレス指定がされた場合、実行するサブシステムは重大エラーを発生し、プログラムが停止されます。

配列操作の組み込みメソッド

配列関数セクションの関数と組み込みメソッドを使用して配列を処理できます。

メソッド	同等	説明
void array.Fill(const scalar value, const int start_pos=0, const int count=-1);	ArrayFill、ArrayInitialize	配列を指定された値に設定します。
void array.Free();	ArrayFree	動的配列バッファを解放し、ゼロ次元サイズをゼロに設定します
int array.Resize(const int range0_size, const int reserve); int array.Resize(const int range_sizes[], const int reserve);	ArrayResize	最初の次元のサイズを新しく設定します
int array.Print();	ArrayPrint	操作ログに単純型配列の値を表示します
int array.Size(const int range=-1);	ArraySize, ArrayRange	配列全体（範囲=-1）または指定された配列次元の要素数を返します
bool array.IsDynamic();	ArrayIsDynamic	配列が動的であるかどうかをチェックします
bool array.IsIndicatorBuffer();		配列が指標バッファであるかどうかをチェックします
bool array.IsSeries();	ArrayIsSeries	配列が時系列であるかどうかをチェックします
bool array.AsSeries();	ArrayGetAsSeries	配列の索引付けの方向をチェックします
bool array.AsSeries(const bool as_series);	ArraySetAsSeries	配列の索引付けの方向を設定します
int array.Copy(const src_array[], const int dst_start, const int src_start, const int cnt);	ArrayCopy	配列要素を別の配列に複製します

© 2000-2025, MetaQuotes Ltd.

メソッド	同等	説明
int array.Compare(const src_array[], const int dst_start, const int src_start, const int cnt);	ArrayCompare	2つの単純型配列またはカスタム構造体を比較した結果を返します
int array.Insert(const src_array[], const int dst_start, const int src_start, const int cnt);	ArrayInsert	指定されたインデックスから開始して、指定された数の要素をソース配列からターゲット配列に挿入します
int array.Remove(const int start_pos, const int count);	ArrayRemove	指定されたインデックスから始めて指定された数の要素を配列から削除します
int array.Reverse(const int start_pos, const int count);	ArrayReverse	指定されたインデックスから始めて指定された数の要素の順番を反転します
bool array.Swap(array& arr[]);	ArraySwap	同じ型の別の動的配列とコンテンツを交換します
void array.Sort(sort_function);	ArraySort	数値配列を初次元によってソートします。
int array.Search(scalar value, search_function);	ArrayBsearch	初次元で最初に見つかった要素のインデックスを返します。
int array.Find((scalar value, search_function);		渡された関数を使用して配列内で検索を実行し、最初に検出された要素のインデックスを返します
array array.Select(scalar value, search_function);		渡された関数を使用して配列内で検索を実行し、検出されたすべての要素を含む配列を返します

アクセス指定子

アクセス指定は、コンパイラが変数や構造体やクラスのメンバにアクセスする方法を定義します。

const 指定子は、変数を定数として宣言し、実行時の変更を不可能にします。宣言時には、変数の単一の初期化が許可されます。

例:

```
int OnCalculate (const int rates_total,      // price[] 配列のサイズ
                 const int prev_calculated,  // 以前の呼び出しで処理されたバー
                 const int begin,            // 重要データの開始点
                 const double& price[]       // 計算対象の配列
   );
```

構造体やクラスのメンバにアクセスするには次の修飾子を使用します。

- public – 変数やクラスメソッドへの無制限のアクセスを許可します。

© 2000–2025, MetaQuotes Ltd.

- protected – このクラスのメソッド、またpublic で継承されたクラスからのアクセスを許可します。その他のアクセスは不可能です。
- private – 同じクラスのメソッドからのみの変数やクラスメソッドへのアクセスを許可します。
- virtual – クラスメソッドのみ(構造体のメソッドは含まない)に適用され、クラスを仮想関数のテーブルに配置する必要があるとコンパイラに指示します。

ストレージクラス

ストレージクラスは 3 つあります。static、input 及び externです。これらのストレージクラス修飾子は、対応する変数がグローバルプールと呼ばれるメモリの事前割り当て領域に分布していることをコンパイラに明示的に指示します。また、これらの修飾子は、可変データの特殊な処理を示します。ローカルレベルで宣言した変数が static でない場合、そのような変数のためのメモリはプログラムスタックに自動的に割り当てられます。配列が宣言されたブロックの視認性の領域を越えると、非静的配列に割り当てられたメモリの解放が自動的に行われます。

参照

データ型、型のカプセル化と拡張性、変数の初期化、変数のアクセス権 スコープとライフタイム、オブジェクトの作成と解徐、クラスの静的メンバ

© 2000–2025, MetaQuotes Ltd.

ローカル変数

関数内で宣言された変数はローカル変数です。ローカル変数の有効範囲は、宣言された関数の内部に制限されています。ローカル変数は任意の式の結果によって初期化出来ます。関数の全ての呼び出しは、ローカル変数を初期化します。ローカル変数は、対応する関数のメモリ領域に格納されます。

例:

```
int somefunc()
  {
   int ret_code=0;
   ...
   return(ret_code);
  }
```

変数の有効範囲とは変数を参照することが出来るプログラムの部分です。(内部レベルでの)ブロック内で宣言された変数の有効範囲はそのブロックです。ブロックスコープは、変数の宣言で始まり、最後の右中括弧で終了します。

関数の先頭で宣言されたローカル変数も関数パラメータ と同じくブロックの有効範囲を持っています。ブロックは変数の宣言を含むことが出来ます。ブロックが入り子にされ、外部ブロックの識別子が内部ブロック内の識別子と同名の場合、外部ブロックの識別子は、内部ブロックの動作が終わるまで隠されています。

例:

```
void OnStart()
  {
//---
   int i=5;       // 関数のローカル変数
     {
      int i=10;  // 関数変数
      Print("Inside block i = ",i); // 結果は  i=10
     }
   Print("Outside block i = ",i);  // 結果は  i=5
  }
```

これは、内部ブロックの実行中には、外部ブロックにおける同名の識別子の値ではなく、それ自身のローカルの識別子の値が見られることを意味します。

例:

```
void OnStart()
  {
//---
   int i=5;       // 関数のローカル変数
   for(int i=0;i<3;i++)
      Print("Inside for i = ",i);
   Print("Outside the block i = ",i);
  }
/* 実行結果
Inside for i = 0
Inside for i = 1
```

© 2000-2025, MetaQuotes Ltd.

```
Inside for i = 2
Outside block i = 5
*/
```

static として宣言されたローカル変数は、プログラム開始時に存在しているにもかかわらず、有効範囲はブロックです。

スタック

全ての MQL5 プログラムでは、自動的に作成されたローカル関数の変数を格納するために、スタックと呼ばれる特殊なメモリ領域が割り当てます。 One stack is allocated for all functions, its default size for indicators is equal to 1Mb. エキスパートアドバイザーとスクリプトでは、スタックサイズはスタックサイズをバイト単位で設定する#property stacksize コンパイラ指令を使用して管理することができ、デフォルトでは、 スタックのために8Mbのメモリが割り当てられます。

静的ローカル変数は、他の静的及びグローバル変数が格納されている、スタックとは別に存在する特殊なメモリ領域に保存されます。動的に作成された変数もスタックとは別のメモリ領域を使用します。

各関数呼び出し時に、内部の非静的変数にスタック内の場所が割り当てられます。メモリは関数の終了後に再使用出来るようになります。

1 番目の関数から 2 番目の関数が呼び出された場合、2 番目の関数は残りのスタックメモリから変数に必要なサイズを使用します。このように、付属の関数を使用する場合には、スタックメモリは順次各関数によって使用されます。これは、スタックオーバーフローと呼ばれる関数呼び出し中のメモリ不足につながる可能性があります。

従って、ローカルデータの量が大きい場合は動的メモリが使用されるべきです。関数に入る際に、ローカルに必要とされるメモリをシステムに割り当て（new、ArrayResize()）、関数から出る際にメモリ解除します（delete、ArrayFree()）。

参照

データ型、型のカプセル化と拡張性、変数の初期化、変数のアクセス権 スコープとライフタイム、オブジェクトの作成と解徐

© 2000–2025, MetaQuotes Ltd.

仮引数

関数に渡されたパラメータはローカルです。有効範囲は関数ブロック内です。仮引数は外部変数や関数の中で定義されたローカル変数とは違う名称を持つ必要があります。関数ブロック内では仮引数に値を代入することが出来ます。仮引数が const 修飾子と宣言された場合、その値は関数内で変更することは出来ません。

例:

```
void func(const int & x[], double y, bool z)
  {
   if(y>0.0 && !z)
      Print(x[0]);
   ...
  }
```

仮引数は定数で初期化出来ます。この場合には、初期化値が初期値と見なされます。横にあるパラメータも初期化する必要があります。

例:

```
void func(int x, double y = 0.0, bool z = true)
  {
   ...
  }
```

このような関数を呼び出す時には初期化されたパラメータを省略することが出来ます。その場合、デフォルトが使用されます。

例:

```
func(123, 0.5);
```

基本データ型 のパラメータは値で渡されます。この型に対応するローカル変数の関数内での変更は呼び出し元の関数には反映されません。任意の型と構造型のデータの配列は、常に参照によって渡されます。配列や構造体の内容を変更することを禁止する必要がある場合には、これらの型のパラメータは *const* キーワードと宣言されるべきです。

基本データ型のパラメータを参照によって渡すことも出来ます。この場合、呼び出し元の関数内でのパラメータの変更は、参照によって渡された変数にも影響を与えます。パラメータが参照によって渡されることを示すには、データ型の後に &修飾子が置かれます。

例:

```
void func(int& x, double& y, double & z[])
  {
   double calculated_tp;
   ...
   for(int i=0; i<OrdersTotal(); i++)
     {
      if(i==ArraySize(z))        break;
      if(OrderSelect(i)==false) break;
      z[i]=OrderOpenPrice();
```

© 2000–2025, MetaQuotes Ltd.

```
      }
    x=i;
    y=calculated_tp;
  }
```

参照によって渡されたパラメータは、初期値で初期化することは出来ません。

関数には最大 64 のパラメータが渡されることが出来ます。

参照

入力変数、データ型、型のカプセル化と拡張性、変数の初期化、変数のアクセス権スコープとライフタイム、オブジェクトの作成と解徐

© 2000–2025, MetaQuotes Ltd.

静的変数

static ストレージクラスは静的変数を定義します。static 修飾子は、データ型の前に示されます。

例:

```
int somefunc()
  {
   static int flag=10;
   ...
   return(flag);
  }
```

任意の式で初期化することが出来るシンプルなローカル変数とは異なり、静的変数は型に対応する定数または定数式によっての初期化が可能です。

静的変数はプログラム実行の瞬間から存在しており、特殊な OnInit() 関数が呼び出される前に一度だけ初期化されます。初期値が指定されていない場合は、静的ストレージクラスの変数の初期値はゼロとなります。

static キーワードで宣言されたローカル変数は、関数のライフタイムを通して値を保持します。このようなローカル変数は、関数の呼び出し時には、前回の呼び出し中に持っていた値が含みます。

関数の仮引数以外の全ての変数は静的に定義出来ます。ローカルレベルで宣言した変数が静的でない場合、そのような変数のためのメモリはプログラムスタックに自動的に割り当てられます。

例:

```
int Counter()
  {
   static int count;
   count++;
   if(count%100==0) Print("Function Counter has been called ",count," times");
   return count;
  }
void OnStart()
  {
//---
   int c=345;
   for(int i=0;i<1000;i++)
     {
      int c=Counter();
     }
   Print("c =",c);
  }
```

参照

データ型、型のカプセル化と拡張性、変数の初期化、変数のアクセス権 スコープとライフタイム、オブジェクトの作成と解徐、静的クラスメンバ

© 2000–2025, MetaQuotes Ltd.

グローバル変数

グローバル変数は、関数の記述外で宣言されることによって作成されます。グローバル変数は関数と同じレベルで定義されます。つまり、ブロックにローカルではありません。

例:

```
int GlobalFlag=10;   // グローバル変数
int OnStart()
  {
   ...
  }
```

グローバル変数の有効範囲は、プログラム全体です。グローバル変数は、プログラムで定義された全ての関数からアクセス出来ます。別の初期値が明示的に定義されていない限り、グローバル変数はゼロに初期化されます。グローバル変数は、その型に対応する定数または定数式でのみ初期化することが出来ます。

グローバル変数は、プログラムがクライアント端末のメモリに読み込まれた後かつ初めの Init イベントハンドリングの前に一度だけ初期化されます。クラスオブジェクトを表すグローバル変数の場合、その初期化中に対応するコンストラクタが呼び出されます。スクリプト内ではグローバル変数は Start イベント処理の前に初期化されます。

注意事項: グローバルレベルで宣言された変数は、GlobalVariable...() 関数を使用してアクセス出来るクライアント端末のグローバル変数と混同されてはなりません。

参照

データ型、型のカプセル化と拡張性、変数の初期化、変数のアクセス権 スコープとライフタイム、オブジェクトの作成と解徐

© 2000-2025, MetaQuotes Ltd.

Input変数

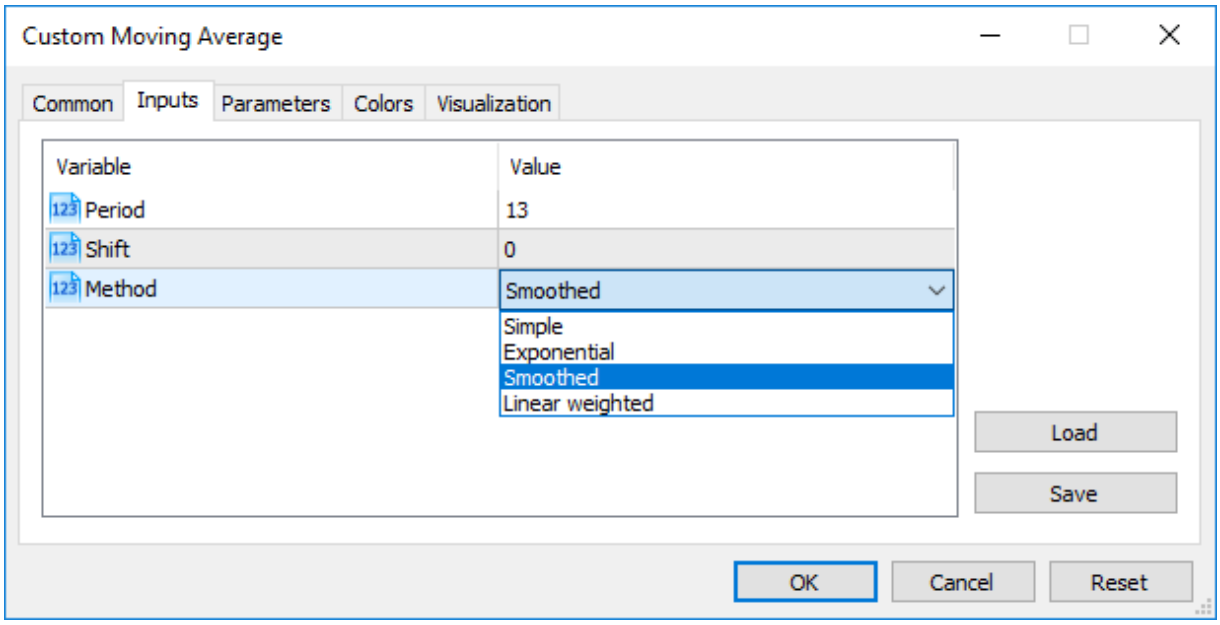
input ストレージクラスは、外部変数を定義しています。*input* 修飾子は、データ型の前に示されます。input 修飾子を持つ変数はMQL5 プログラム内で変更することはできず、読み込み専用でアクセスすることが出来ます。input 変数の値はプログラムのプロパティウィンドウから、ユーザによってのみ変更することが出来ます。

入力変数名の最大長は63文字です。文字列型の入力パラメータの場合、値の最大の長さ(文字列の長さ)は191～253文字です (注を参照)。最小の長さは0文字です(値は設定されていません)。

例:

```
//--- 入力パラメータ
input int             MA_Period=13;
input int             MA_Shift=0;
input ENUM_MA_METHOD MA_Method=MODE_SMMA;
```

Input 変数は、プログラムの入力パラメータを決定します。プログラムのプロパティウィンドウから使用出来ます。

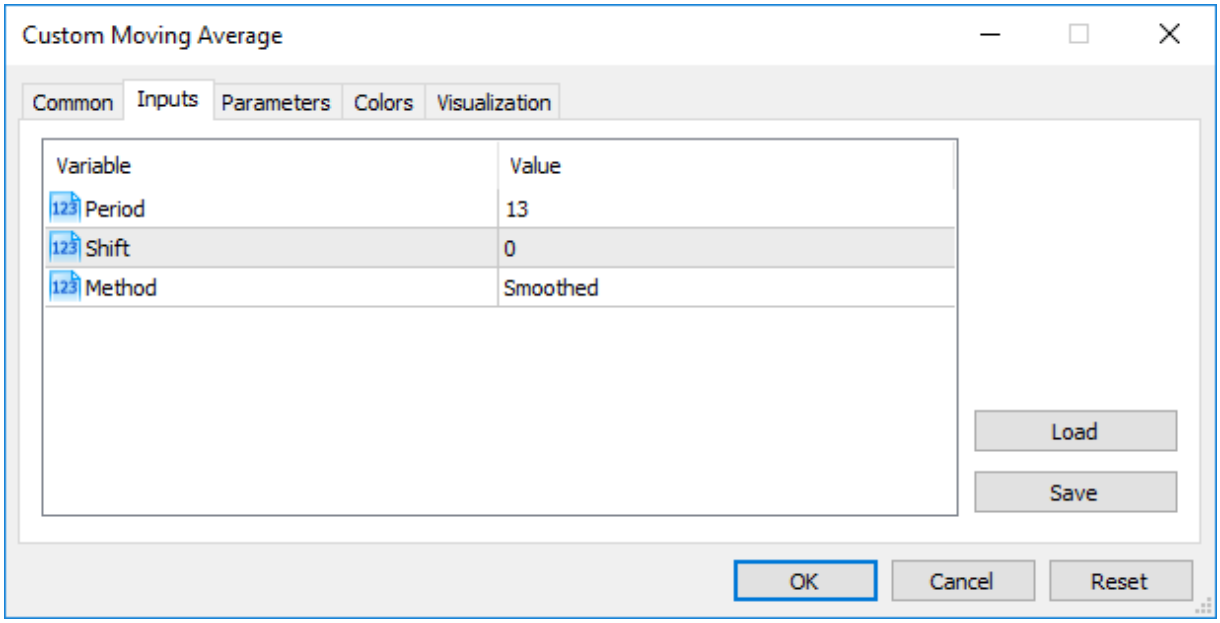


「パラメータ(Inputs)」タブでの入力パラメータの外観を設定する方法はもう 1 つあります。入力パラメータの記述と同じ行へのコメントの追加です。これによって、入力パラメータの名称をユーザによって理解しやすくします。

例:

```
//--- 入力パラメータ
input int             InpMAPeriod=13;          // 平滑化期間
input int             InpMAShift=0;            // ライン水平シフト
input ENUM_MA_METHOD InpMAMethod=MODE_SMMA;  // 平滑化の方法
```

© 2000–2025, MetaQuotes Ltd.



注意事項：配列及び複合型の変数は入力パラメータとしては使用できません。

注意事項：入力変数の文字列のコメントの長さは63文字を超えることは出来ません。

注意事項：文字列型の入力変数の場合、値の長さ(文字列の長さ)の制限は次の条件によって設定されます。

- パラメータ値は「parameter_name = parameter_value」文字列で表されます(「=」が考慮されます)。
- 最大表現長255文字(*total_length_max* = 255または254文字(「=」を除く)。
- </ t0> *parameter_name_length* </ *t1*> 文字列パラメータの最大長 = 63文字。</ t2>

したがって、文字列パラメータの最大文字列サイズは、次の式を使用して計算されます。

```
parameter_value_length=total_length_max-parameter_name_length=254-
parameter_name_length
```

最大文字列サイズは191(parameter_name_length = 63)から253文字(parameter_name_length = 1)になります。

MQL5 プログラムからカスタム指標を呼び出す際のパラメータの受け渡し

カスタム指標は iCustom() 関数を使用して呼び出されます。カスタム指標名の後には、このカスタム指標の入力変数の宣言に厳密に従ってパラメータが指定される必要があります。示されたパラメータがカスタム指標で宣言された入力変数よりも少ない場合、不足しているパラメータは、変数の宣言時に指定された値で書き入れられます。

カスタム指標が最初の種類の OnCalculate 関数を使用する場合（すなわち、指標が同じデータ配列を使用して計算される場合）、カスタム指標を呼ぶのに ENUM_APPLIED_PRICE 値の 1 つまたは別の指標ハンドルが最後のパラメータとして使用されるべきです。入力変数に対応する全てのパラメータは明確に示されなければなりません。

入力パラメータとしての列挙

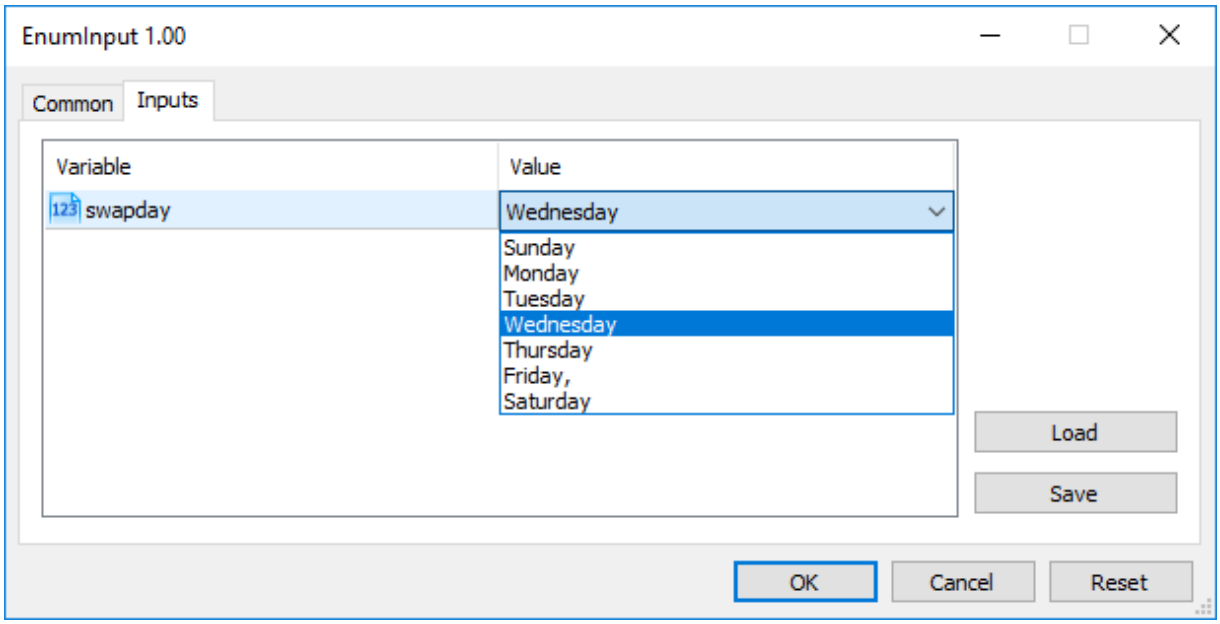
© 2000–2025, MetaQuotes Ltd.

MQL5 の提供する内蔵の列挙だけでなくユーザ定義の変数も入力変数（MQL5 プログラムの入力パラメータ）として使用することが出来ます。例えば、曜日を記述する dayOfWeek 列挙体を作成し、入力変数を使用して特定の曜日を数の代わりにもっと一般的な方法で指定することが出来ます。

例：

```
#property script_show_inputs
//--- 曜日
enum dayOfWeek
  {
   S=0,      // 日曜
   M=1,      // 月曜
   T=2,      // 火曜
   W=3,      // 水曜
   Th=4,     // 木曜
   Fr=5,     // 金曜
   St=6,     // 土曜
  };
//--- 入力パラメータ
input dayOfWeek swapday=W;
```

スクリプトの起動時にプロパティウィンドウからの必要な値の選択を可能にするには、プリプロセッサコマンド #property script_show_inputs を使用します。スクリプトを起動して、リストから dayOfWeek 列挙体のいずれかの値を選択することが出来ます。EnumInInput スクリプトを起動して「パラメータ（ Inputs ）」タブに移動します。swapday の値（3 重スワップチャージの日）はデフォルトでは水曜日（W=3）ですが、他の値を指定して、プログラムの動作を変更することが出来ます。



列挙では可能な値の数は限られています。入力値を選択するためにはドロップダウンリストが使用されます。一覧に表示された値に使用されるのは列挙型メンバのニーモニック名です。この例のようにコメントがニーモニック名に関連付けられている場合、ニーモニック名でなくコメントの内容が使用されます。

© 2000–2025, MetaQuotes Ltd.

dayOfWeek 列挙の値は 0 から 6 までですが、パラメータのリストではそれぞれの値に指定されたコメントが表示されます。これは、入力パラメータの明確な記述を含むプログラムを作成するための柔軟性を提供します。

sinput 修飾子を持つ変数

input 修飾子を持つ変数はプログラムを起動時に外部パラメータ値を設定するだけでなくストラテジーテスターで取引ストラテジーを最適化する際にも必要とされます。文字列型のものを除く各入力変数は、最適化に使用することが出来ます。

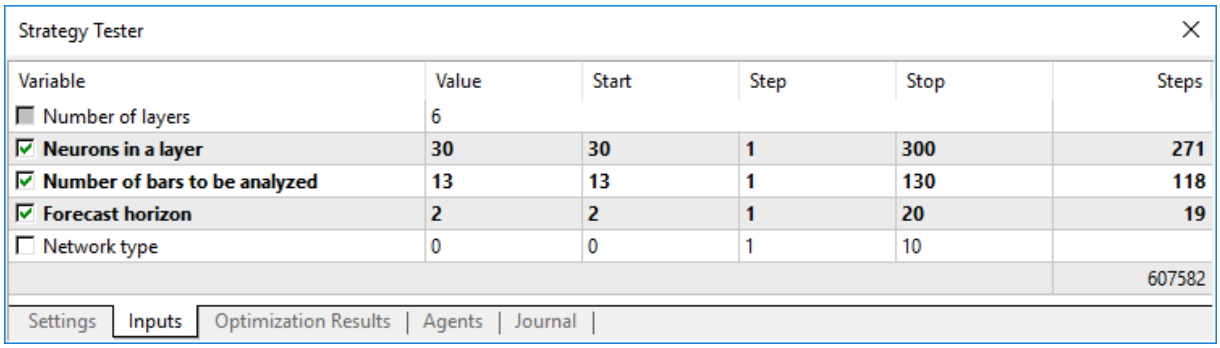
時々、テスタ内の全てのパスの領域から外部プログラムのパラメータを除外することが必要です。sinput メモリ修飾子は、そのような場合のために導入されています。sinput は静的外部変数の宣言（sinput=静的入力）の略です。つまり、エキスパートアドバイザーコードで次の宣言

```
sinput       int layers=6;   // 層数
```

が完全な宣言に相当します。

```
static input int layers=6;   // 層数
```

sinput 修飾子で宣言された変数は MQL5 プログラムの入力パラメータです。このパラメータの値はプログラム起動時に変更することが出来ます。この変数は、入力パラメータの最適化には使用されません。言い方を変えると、指定された条件に合う最良のパラメータセットを検索する時には、その値は列挙されません。



上に示したエキスパートアドバイザーには 5 つの外部パラメータがあります。「Number of layers」は sinput として宣言されており 6 に等しいです。このパラメータは、取引ストラテジーの最適化中に変更することは出来ません。後で使用するために必要な値を指定することは出来ます。Start、Step と Stop のフィールドは、このような変数には使用出来ません。

よって、変数の sinput 修飾子が指定された後、ユーザがこのパラメータを最適化することは出来ません。つまり、端末ユーザは、最適化時に、ストラテジーテスターで指定された範囲内の自動列挙の初期値と最終値を設定することが出来ません。

ただし、この規則には例外が1つあります。sinput 変数は ParameterSetRange() 関数を使用して最適化タスクで変更することが出来ます。この関数は、static input(sinput)として宣言されたものを含む input 変数の使用可能な値のプログラム制御のために特異的に導入されました。ParameterGetRange() 関数は、最適化の起動時に(OnTesterInit() ハンドラで)入力変数の値を受け取り、最適化されたパラメータ値が列挙される範囲内の変更ステップ値と範囲を再設定します。

このように sinput 修飾子と入力パラメータで動作する 2 つの関数を組み合わせて、他の入力パラメータの値に依存する入力パラメータの最適化間隔を設定するための柔軟なルールを作成することが出来ます。

© 2000–2025, MetaQuotes Ltd.

入力パラメータのグループ化

MQL5プログラムでの作業を簡単にするために、入力パラメータはgroupキーワードを使用して名前付きブロックに分割できます。これにより、埋め込まれたロジックに基づいて、パラメータを視覚的に分離できます。

```
input group           "Group name"
input int             variable1 = ...
input double          variable2 = ...
input double          variable3= ...
```

このように宣言されると、すべての入力パラメータは視覚的に指定されたグループに結合され、チャートまたはストラテジーテスターで起動する際のMQL5ユーザのパラメータ設定を簡素化します。各グループの指定は、新しいグループ宣言が表示されるまで有効です。

```
input group           "Group name #1"
input int             group1_var1 = ...
input double          group1_var2 = ...
input double          group1_var3 = ...

input group           "Group name #2
input int             group2_var1 = ...
input double          group2_var2 = ...
input double          group2_var3 = ...
```

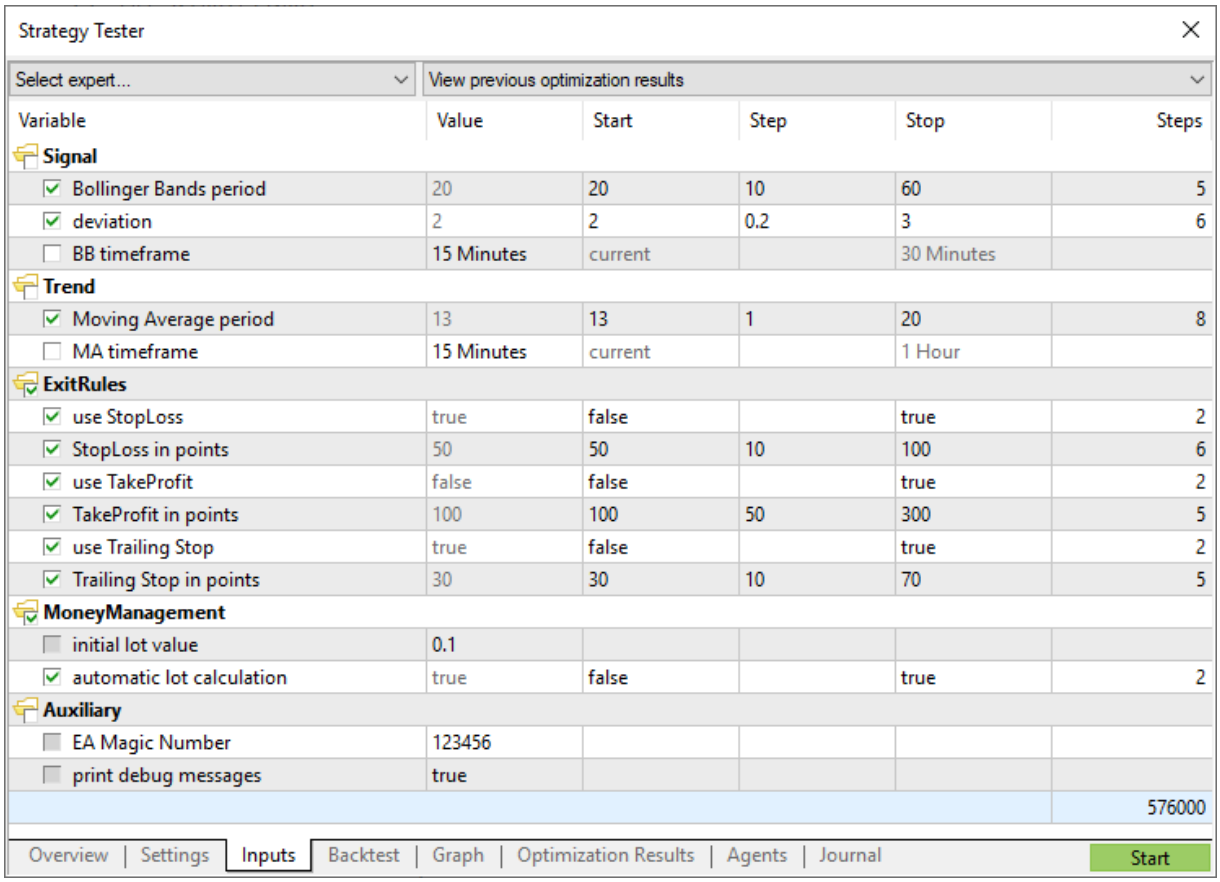
以下目的別に分けられた入力ブロックを特徴とするEAの例です。

```
input group           "Signal"
input int             ExtBBPeriod    = 20;       // ボリンジャーバンド期間
input double          ExtBBDeviation= 2.0;       // 偏差
input ENUM_TIMEFRAMES ExtSignalTF=PERIOD_M15;    // BB時間枠

input group           "Trend"
input int             ExtMAPeriod    = 13;       // 移動平均期間
input ENUM_TIMEFRAMES ExtTrendTF=PERIOD_M15;     // MA時間枠

input group           "ExitRules"
input bool            ExtUseSL       = true;     // StopLossを使用
input int             Ext_SL_Points  = 50;       // ポイント単位のStopLoss
input bool            ExtUseTP       = false;    // TakeProfitを使用
input int             Ext_TP_Points  = 100;      // ポイント単位のTakeProfit
input bool            ExtUseTS       = true;     // Trailing Stopを使用
input int             Ext_TS_Points  = 30;       // ポイント単位のTrailing Stop

input group           "MoneyManagement"
sinput double         ExtInitialLot  = 0.1;      // 初期ロット値
input bool            ExtUseAutoLot  = true;     // 自動ロット計算

input group           "Auxiliary"
```

© 2000-2025, MetaQuotes Ltd.

```
sinput int           ExtMagicNumber = 123456;   // EAマジックナンバー
sinput bool          ExtDebugMessage= true;     // デバッグメッセージを出力する
```

ストラテジーテスターでそのようなEAを起動する場合、グループ名をダブルクリックして入力ブロックを折りたたみ/展開し、グループチェックボックスをクリックして、最適化のためにすべてのパラメータを選択できます。



参照

iCustom、列挙、プログラムプロパティ

© 2000–2025, MetaQuotes Ltd.

Extern 変数

extern キーワードは変数名の宣言に静的ストレージクラスとグローバルライフタイムの識別子として使用されます。これらの変数はプログラムの先頭から存在し、メモリはプログラムの開始直後に割り当てられ初期化されます。

複数のソースファイルで構成されたプログラムを作成することは可能です。この場合、プリプロセッサに指示する #include ディレクティブが使用されます。同じ型と識別子を持つ extern と宣言された変数は、プロジェクト内の複数のソースファイルに存在することが出来ます。

プロジェクト全体をコンパイルする時、同じ型と識別子を持つ全ての extern 変数は、グローバル変数プールのメモリの一部に関連しています。Extern 変数は、ソースファイルの分割コンパイルするのに便利です。Extern 変数が初期化出来るのは一度だけです。同じ型と識別子を持つ extern 変数の複数の初期化は禁止されています。

参照

データ型、型のカプセル化と拡張性、変数の初期化、変数のアクセス権スコープとライフタイム、オブジェクトの作成と解徐

© 2000–2025, MetaQuotes Ltd.

変数の初期化

変数は、定義時に初期化することが出来ます。変数が明示的に初期化されていない場合、格納された値は予知出来ません。暗黙的な初期化は使用されません。

グローバルまたは静的変数は対応する型の定数または定数式のみで初期化することが出来ます。ローカル変数は定数だけではなく任意の式で初期化することが出来ます。

グローバル変数と静的変数の初期化は一度だけ実行されます。ローカル変数の初期化は、対応する関数を呼び出すたびに行われます。

例：

```
int    n       = 1;
string s       = "hello";
double f[]     = { 0.0, 0.236, 0.382, 0.5, 0.618, 1.0 };
int    a[4][4] = { {1, 1, 1, 1}, {2, 2, 2, 2}, {3, 3, 3, 3}, {4, 4, 4, 4 } };
//--- テトリスから
int    right[4]={WIDTH_IN_PIXELS+VERT_BORDER,WIDTH_IN_PIXELS+VERT_BORDER,
                 WIDTH_IN_PIXELS+VERT_BORDER,WIDTH_IN_PIXELS+VERT_BORDER};
//--- 構造体の全てのフィールドをゼロで初期化する
MqlTradeRequest request={};
```

配列要素の値のリストは、中括弧で囲む必要があります。初期化シーケンスがない場合、値は 0 だと見なされます。初期化シーケンスは、少なくとも 1 つの値を持っている必要があります。

初期化された配列のサイズが指定されていない場合は、コンパイラが初期化シーケンスのサイズに基づいて決定します。

例：

```
struct str3
  {
   int               low_part;
   int               high_part;
  };
struct str10
  {
   str3              s3;
   double            d1[10];
   int               i3;
  };
void OnStart()
  {
   str10 s10_1={{1,0},{1.0,2.1,3.2,4.4,5.3,6.1,7.8,8.7,9.2,10.0},100};
   str10 s10_2={{1,0},{},100};
   str10 s10_3={{1,0},{1.0}};
//---
   Print("1.  s10_1.d1[5] = ",s10_1.d1[5]);
   Print("2.  s10_2.d1[5] = ",s10_2.d1[5]);
   Print("3.  s10_3.d1[5] = ",s10_3.d1[5]);
```

© 2000–2025, MetaQuotes Ltd.

```
   Print("4.  s10_3.d1[0] = ",s10_3.d1[0]);
  }
```

構造体タイプ及び静的配列の変数の場合、(暗黙的に設定されたサイズで)部分のみの初期化が許可されています。構造体または配列の1以上の第一の要素を初期化することができ、他の要素は、ゼロで初期化されます。

参照

データ型、型のカプセル化と拡張性、変数のアクセス権スコープとライフタイム、オブジェクトの作成と解徐

© 2000-2025, MetaQuotes Ltd.

変数のアクセス権の有効範囲とライフタイム

有効範囲は基本的に 2 つあります。それらはローカルスコープとグローバルスコープです。

全ての関数の外で宣言された変数は、グローバルスコープに属します。このような変数はプログラムのどこからでもアクセス可能です。それらはメモリのグローバルプールに配置されているので、ライフタイムはプログラムのライフタイムと一致します。

ブロック(中括弧で囲まれたコードの一部)の内側に宣言された変数はローカルスコープに属します。このような変数は、それが宣言されているブロック外からは見えません(従って使用出来ません)。ローカル宣言の中で最も一般的なケースは、関数内で宣言された変数です。ローカルに宣言された変数は、スタック内に位置しライフタイムは関数のライフタイムに等しいです。

ローカル変数の有効範囲はそれが宣言されているブロックなので、他のブロックで宣言された変数と、またグローバルレベルまでの上位レベルで宣言された変数と同名の変数を宣言することが可能です。

例:

```
void CalculateLWMA(int rates_total,int prev_calculated,int begin,const double
&price[])
  {
   int        i,limit;
   static int weightsum=0;
   double     sum=0;
//---
   if(prev_calculated==0)
     {
      limit=MA_Period+begin;
      //--- 初めのリミットバーに空の値を設定する
      for(i=0; i<limit; i++) LineBuffer[i]=0.0;
      //--- 最初に表示される値を計算する
      double firstValue=0;
      for(int i=begin; i<limit; i++)
        {
         int k=i-begin+1;
         weightsum+=k;
         firstValue+=k*price[i];
        }
      firstValue/=(double)weightsum;
      LineBuffer[limit-1]=firstValue;
     }
   else
     {
      limit=prev_calculated-1;
     }

   for(i=limit;i<rates_total;i++)
     {
      sum=0;
      for(int j=0; j<MA_Period; j++) sum+=(MA_Period-j)*price[i-j];
```

© 2000-2025, MetaQuotes Ltd.

```
      LineBuffer[i]=sum/weightsum;
     }
//---
  }
```

同じ行で宣言された変数 i にご注意ください。

```
      for(int i=begin; i<limit; i++)
        {
         int k=i-begin+1;
         weightsum+=k;
         firstValue+=k*price[i];
        }
```

i の有効範囲はループのみです。ループの外には、関数の初めで宣言された同名の変数が存在します。また、変数 k はループ本体で宣言されているので、その有効範囲はループ本体です。

ローカル変数は、アクセス指定子 static を使用して宣言することが出来ます。この場合、コンパイラはメモリのグローバルプール内に変数を有します。従って、静的変数のライフタイムは、プログラムのライフタイムに等しいです。ここではそのような変数の有効範囲は、それが宣言されているブロックに制限されています。

参照

データ型、型のカプセル化と拡張性、変数の初期化、オブジェクトの作成と解徐

© 2000–2025, MetaQuotes Ltd.

オブジェクトの作成と解除

MQL5 プログラムが実行のために読み込まれた後、メモリは、その型に応じて各変数に割り当てられます。すべての変数は、アクセスレベルに応じてグローバル変数とローカル変数の 2 種類に分かれています。メモリクラスによると、それらは MQL5 プログラムの入力パラメータ、静的と自動に分けられます。変数は必要に応じて対応する値に初期化されます。使用後には初期化解除され、変数によって使用されたメモリは MQL5 実行システムに戻されます。

グローバル変数の初期化と初期化解除

グローバル変数は、MQL5 プログラムが読み入れられた直後、関数が 1 つでも呼び出される前に自動的に初期化されます。初期化中に、基本データ型の変数には初期値が割り当てられ、オブジェクトにはコンストラクタ（もしあれば）が呼び出されます。入力変数はは常にグローバルレベルで宣言され、プログラム開始時にダイアログでユーザが設定した値で初期化されます。

静的変数は通常ローカルレベルで宣言されていますが、グローバル変数と同様に、メモリが前もって割り当てられ、初期化はプログラム読み込みの直後に行われます。

初期化の順序は、プログラムの変数宣言順に対応しています。初期化解除は逆の順序で実行されます。この規則は、new 演算子によって作成されていない変数に対してのみ適応します。このような変数は、プログラム読み込み後に自動的に作成/初期化され、プログラムのアンロードの前に初期化解除されます。

ローカル変数の初期化と初期化解除

ローカルレベルで宣言した変数が静的でない場合は、メモリが自動的に割り当てられます。グローバル変数と同様に、ローカル変数は、プログラムの実行が宣言を満たした瞬間に自動的に初期化されます。従って、初期化の順序は、宣言の順序に対応しています。

ローカル変数は、宣言されたプログラムブロックの最後に宣言の逆の順序で初期化解除されます。プログラムブロックは複合演算子で switch 選択演算子、反復演算子（ for、while、do-while ）、関数本体及び if-else 演算子の一部になることがあります。

ローカル変数は、プログラムの実行が変数宣言を満たす時に初めて初期化されます。変数が宣言されているブロックが実行されなかった場合は、そのような変数は初期化されていません。

配置されたオブジェクトの初期化と初期化解除

ポインタの宣言は、対応するオブジェクトの初期化を伴わないのでオブジェクトポインタは特殊となります。動的に配置されたオブジェクトは、new 演算子がクラスのサンプル作成した時点で初期化されます。オブジェクトの初期化は、対応するクラスのコンストラクタの呼び出しを前提としています。クラス内に対応するコンストラクタがない場合、基本データ型のメンバは自動的には初期化されません。文字列、動的配列及び複合型オブジェクト型のメンバは自動的に初期化されます。

ポインタは、ローカルまたはグローバルレベルで宣言することができ、NULLの空の値、または 同じ型か継承された 型を持つポインタの値で初期化出来ます。*new* 演算子がローカルレベルで宣言されたポインタに対して呼び出さた場合、このポインタの *delete* 演算はレベルを終了する前に実行される必要があります。さもないと、ポインタが失われ、オブジェクトの明示的な削除は失敗します。

object_pointer=new *Class_name* の式を使用して作成されたオブジェクトは delete(*object_pointer*) 演算子で削除されなければいけません。このような変数が delete 演算子 によってプログラム終了時に削除され

© 2000–2025, MetaQuotes Ltd.

ない場合は、対応するエントリが「エキスパート」操作ログに表示されます。複数の変数を宣言し、それらの全てに 1 つのオブジェクトポインタを割り当てることは可能です。

動的に作成されたオブジェクトにコンストラクタがある場合、このコンストラクタは、*new* 演算子の実行の瞬間に呼び出されます。オブジェクトにデストラクタがある場合、それは *delete* 演算子の実行中に呼び出されます。

このように、動的に配置されたオブジェクトは *new* 演算子で初めて作成され、*delete* 演算子またはプログラムアンロード時に MQL5 の実行システムによって自動的に削除されます。 動的に作成されたオブジェクトのポインタの宣言の順序は、初期化の順序には影響しません。初期化と初期化解除の順序はプログラマによって完全に制御されます。

MQL5 における動的なメモリ割り当て

動的配列を使用すると、解放されたメモリは、すぐにオペレーティングシステムに戻されます。

new 演算子を使用して動的なクラスのオブジェクトを作成する時、メモリは初め、メモリマネージャが使用しているクラスメモリプールから要求されます。充分なメモリがプール内にない場合、メモリは、オペレーティングシステムから要求されます。delete 演算子を使用して動的なクラスのオブジェクトを削除する時、解放されたメモリはすぐにクラスメモリプールに戻されます。

メモリマネージャは OnInit()、OnDeinit()、OnStart()、OnTick()、OnCalculate()、OnTimer()、OnTrade()、OnTester()、OnTesterInit()、OnTesterPass()、OnTesterDeinit()、OnChartEvent()、OnBookEvent() イベント処理関数の終了後、解放したメモリを直ちにオペレーティングシステムに返します。

変数の特徴

作成と削除の順序、またコンストラクタとデストラクタの呼び出しに関する主な情報は以下の表に示されています。

	グローバル自動変数	ローカル自動変数	動的に作成されたオブジェクト
初期化	MQL5 プログラム読み込みの直後	宣言されたコード行の実行時	new 演算子の実行時
初期化順	宣言の順序	宣言の順序	宣言の順序と関係なし
初期化解除	MQL5 プログラムのアンロード前	宣言ブロックの実行終了時	delete 演算子の実行時またはMQL5 プログラムのアンロード前
初期化解除順	初期化順の反対	初期化順の反対	初期化順と無関係
コンストラクタの呼び出し	MQL5 プログラム読み込み時	初期化時	*new* 演算子の実行時
デストラクタの呼び出し	MQL5 プログラムのアンロード時	変数が初期化されたブロックの終了時	*delete* 演算子の実行時
エラーログ	自動的に作成されたオブジェクトを削除しよ	自動的に作成されたオブジェクトを削除しよ	MQL5 プログラムのアンロード時に削除を取り

© 2000–2025, MetaQuotes Ltd.

	グローバル自動変数	ローカル自動変数	動的に作成されたオブジェクト
	うとする試みについて「エキスパート」操作ログにメッセージを記録	うとする試みについて「エキスパート」操作ログにメッセージを記録	消すに動的に作成されたオブジェクトについて「エキスパート」操作ログにメッセージを記録

参照

データ型、型のカプセル化と拡張性、変数の初期化、変数のアクセス権 スコープとライフタイム

© 2000-2025, MetaQuotes Ltd.

プリプロセッサ

プリプロセッサは、プログラムがコンパイルされる直前のプログラムソースコードの準備を目的とした MQL5 コンパイラの特別なサブシステムです。

プリプロセッサはソースコードの読みやすさを向上します。コードは、MQL5 プログラムのソースコードを含む特殊ファイルを含めることにより、構造化することが出来ます。定数への呼び名の割り当ては、コードの読みやすさの向上に貢献します。

またプリプロセッサは MQL5 プログラムの特定パラメータを決定することも出来ます。

- 定数の宣言
- プログラムプロパティの設定
- プログラムテキスト内の Include ファイル
- 関数のインポート
- 条件付きコンパイル

プリプロセッサディレクティブはコンパイルする前にソースコードを前処理するためにコンパイラによって使用されます。ディレクティブは常に#で始まります。したがって、コンパイラは変数名、関数名などでのこのシンボルの使用を禁止します。

各ディレクティブはエントリごとに指定されます。1つのエントリで複数のディレクティブを使用することはできません。ディレクティブエントリが大きすぎる場合は、¥記号を使用して複数の行に分割します。この場合、次の行はディレクティブエントリの続きと見なされます。

```
//+------------------------------------------------------------------+
//|  foreach擬似演算子                                               |
//+------------------------------------------------------------------+
#define ForEach(index, array) for (int index = 0,                    \
   max_##index=ArraySize((array));                                   \
   index<max_##index; index++)
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string array[]={"12","23","34","45"};
//--- ForEachを使用して配列をバイパスする
   ForEach(i,array)
     {
      PrintFormat("%d: array[%d]=%s",i,i,array[i]);
     }
  }
//+------------------------------------------------------------------+
/* 結果の出力
   0: array[0]=12
   1: array[1]=23
   2: array[2]=34
   3: array[3]=45
*/
```

© 2000-2025, MetaQuotes Ltd.

これら3つの #defineディレクティブは、コンパイラには1つの長い行のように見えます。上記の例は ##も適用します。これは、2つのマクロトークンを1つにマージする#defineマクロで使用されるマージ演算子です。トークンマージ演算子は、マクロ定義の最初または最後として使用することはできません。

© 2000-2025, MetaQuotes Ltd.

マクロ代入（#define）

プリプロセッサディレクティブはコンパイルする前にソースコードを前処理するためにコンパイラによって使用されます。ディレクティブは常に#で始まります。したがって、コンパイラは変数名、関数名などでのこのシンボルの使用を禁止します。

各ディレクティブはエントリごとに指定されます。1つのエントリで複数のディレクティブを使用することはできません。ディレクティブエントリが大きすぎる場合は、¥記号を使用して複数の行に分割します。この場合、次の行はディレクティブエントリの続きと見なされます。

#define ディレクティブは、定数にニーモニック名を割り当てるために使用することが出来ます。#define には 2 つの形式があります。

```
#define 識別子式                                    // パラメータなしの形式
#define 識別子（パラメータ1, . . . パラメータ8） 式     // パラメトリック形式
```

#define ディレクティブは後に続くソーステキストで*識別子*があった場合その全てに**式**を代入します。代入は*識別子*が独立したトークンである場合にのみ行われます。代入は*識別子*がコメントの一部、文字列の一部、または他の長い識別子の一部である場合には行われません。

定数識別子は、変数名と同じ規則によって扱われます。値は任意の型のものとすることが出来ます。

```
#define ABC                100
#define PI                 3.14
#define COMPANY_NAME       "MetaQuotes Software Corp."
...
void ShowCopyright()
  {
   Print("Copyright  2001-2009, ",COMPANY_NAME);
   Print("https://www.metaquotes.net");
  }
```

式はキーワード、定数、定数及び非定数式などのいくつかのトークンで構成されます。**式**は行の終わりで終了し、次の行に続くことは出来ません。

例:

```
#define TWO         2
#define THREE       3
#define INCOMPLETE  TWO+THREE
#define COMPLETE   (TWO+THREE)
void OnStart()
  {
   Print("2 + 3*2 = ",INCOMPLETE*2);
   Print("(2 + 3)*2 = ",COMPLETE*2);
  }
// 結果
// 2 + 3*2 = 8
// (2 + 3)*2 = 10
```

© 2000–2025, MetaQuotes Ltd.

パラメトリック形式の #define

パラメトリック形式の場合、後続する識別子の全てが実際のパラメータを考慮して代入されます。例えば、

```
// 2 つのパラメータ a と b を使用する例
#define A 2+3
#define B 5-1
#define MUL(a, b) ((a)*(b))

double c=MUL(A,B);
Print("c=",c);
/*
double c=MUL(A, B); の式は
double c=((2+3)*(5-1)); と同じです。
*/
// 結果
// c=20
```

見つけにくい非明白なエラーを防ぐために、式の中でパラメータを使用する際にはパラメータを括弧で囲むようにしてください。ブラケットを使用せずにコードを書き換えると、結果が異なります。

```
// 2 つのパラメータ a と b を使用する例
#define A 2+3
#define B 5-1
#define MUL(a, b) a*b

double c=MUL(A,B);
Print("c=",c);
/*
double c=MUL(A, B); の式は
double c=2+3*5-1; と同じです。
*/
// 結果
// c=16
```

パラメトリック形式では、最大で 8 つのパラメータが使用可能です。

```
// パラメトリック形式の正しい使い方
#define LOG(text)  Print(__FILE__,"(",__LINE__,") :",text)          // パラメータは「text」の 1 つ


// パラメトリック形式の不正な使い方
#define WRONG_DEF(p1, p2, p3, p4, p5, p6, p7, p8, p9)   p1+p2+p3+p4 // パラメータが p1 から p9 の 8 つ以上
```

#undef ディレクティブ

#undef は、前に定義されたマクロ代入の宣言を解除します。

例：

© 2000-2025, MetaQuotes Ltd.

```
#define MACRO

void func1()
  {
#ifdef MACRO
   Print("MACRO is defined in ",__FUNCTION__);
#else
   Print("MACRO is not defined in ",__FUNCTION__);
#endif
  }

#undef MACRO

void func2()
  {
#ifdef MACRO
   Print("MACRO is defined in ",__FUNCTION__);
#else
   Print("MACRO is not defined in ",__FUNCTION__);
#endif
  }

void OnStart()
  {
   func1();
   func2();
  }

/* 結果:
 MACRO is defined in func1
 MACRO is not defined in func2
*/
```

参照

識別子、文字定数

© 2000–2025, MetaQuotes Ltd.

プログラムのプロパティ（#property）

全ての MQL5 プログラムは、明示的に起動せずにプログラムの中で適切なサービスを提供してクライアント端末を補助する #property という付加的な特定のパラメータを指定することが出来ます。これは、まず第一に、指標の外部設定に関連します。インクルードファイルに記載されているプロパティは完全に無視されます。プロパティは、メイン mq5 ファイルで指定する必要があります。

```
#property 識別子の値
```

コンパイラは、実行モジュールの設定に宣言された値を書き込みます。

定数	Type	説明
icon	string	EX5 プログラムのアイコンとして使用される画像ファイルへのパス。パス指定のルールはリソースの場合と同じです。このプロパティは、MQL5 のソースコードのメインモジュールで指定される必要があります。アイコンファイルは ICO 形式である必要があります。
link	string	会社のウェブサイトへのリンク。
copyright	string	会社名。
version	string	プログラムのバージョン（最大 31 文字）
説明	string	MQL5 プログラムの簡単な説明文。複数の*説明*が存在でき、各々がテキストの1行を記述します。合計した*説明*の長さは、改行を含んで 511 文字を超えることは出来ません。
stacksize	int	MQL5 プログラムスタックサイズ。関数の再帰呼び出しを実行する際には充分なサイズのスタックが必要です。 スクリプトまたはチャート上のエキスパートアドバイザーを起動する際には、8 MB 以上のスタックが割り当てられます。指標の場合には、スタックサイズは常に 1 MB に固定されています。 プログラムはがストラテジーテスターで起動される場合には、常に 16 MB のスタックが割り当てられます。
library		ライブラリ。start 関数は割り当てられません。export 修飾子のついた関数は他の MQL5 プログラムにインポート出来ます。
indicator_applied_price	int	「適用価格」フィールドの初期値を指定します。値は ENUM_APPLIED_PRICE から選べます。プロパティが指定されていない場合、初期値は PRICE_CLOSE です。
indicator_chart_window		チャートウィンドウで指標を表示します。
indicator_separate_window		別ウィンドウで指標を表示します。
indicator_height	int	ピクセル単位での指標サブウィンドウの固定された高さ（INDICATOR_HEIGHT プロパティ）。

© 2000–2025, MetaQuotes Ltd.

定数	Type	説明
indicator_buffers	int	指標計算に使用されるバッファ数。
indicator_plots	int	指標のグラフィックシリーズの数。
indicator_minimum	double	個別の指標ウィンドウのスケーリングの下限。
indicator_maximum	double	個別の指標ウィンドウのスケーリングの上限。
indicator_labelN	string	データウィンドウに表示された N 番目のグラフィックシリーズ のラベルの設定。複数の指標バッファ（DRAW_CANDLES、DRAW_FILLINGなど）を必要とするグラフィックシリーズでは、ラベル名はセパレータ「 ; 」を使用して定義されています。
indicator_colorN	color	線 N の表示色。N はグラフィックシリーズの番号で 1 から始まります。
indicator_widthN	int	グラフィックシリーズの線の太さ。N はグラフィックシリーズの番号で 1 から始まります。
indicator_styleN	int	ENUM_LINE_STYLE を使用して指定されたグラフィックシリーズの線のスタイル。N はグラフィックシリーズの番号で 1 から始まります。
indicator_typeN	int	ENUM_DRAW_TYPE を使用して指定されたグラフィカルプロットの種類。. N はグラフィックシリーズの番号で 1 から始まります。</t4>
indicator_levelN	double	個別の指標ウィンドウにおける N の水平レベル。
indicator_levelcolor	color	指標の水平レベルの色。
indicator_levelwidth	int	指標の水平レベルの太さ。
indicator_levelstyle	int	指標の水平レベルのスタイル。
script_show_confirm		スクリプト実行の前に、確認メッセージウィンドウを表示します。
script_show_inputs		スクリプト実行の前にプロパティウィンドウを表示して確認メッセージウィンドウを無効化します。
tester_indicator	string	「*indicator_name.ex5*」の形式でのカスタム指標の名称。 対応するパラメータが定数文字列を介して設定されている場合、テストが必要な指標はiCustom() の呼び出しから自動的に定義されています。他の全ての場合（IndicatorCreate() 関数の使用や指標名を設定するパラメータでの非定数文字列の使用）、このプロパティが必要とされています。
tester_file	string	テスタのファイル名。二重引用符内で拡張子の表示も含みます(文字列定数)。指定されたファイルはテスタに渡されます。テストされるべき入力ファイルがある場合、指定が必ず必要です。

© 2000–2025, MetaQuotes Ltd.

定数	Type	説明
tester_library	string	二重引用符内で拡張子を含むライブラリ名。ライブラリは拡張子として「dll」または「ex5」を持つことが出来ます。テストを必要とするライブラリは自動的に定義されます。しかし、ライブラリのいずれかがカスタム指標で使用されている場合、このプロパティが必要です。
tester_set	string	入力パラメータの値とステップを含むセットファイルの名前。このファイルは、テスト及び最適化の前にテスタに渡されます。ファイル名は、拡張子を含み二重引用符で定数文字列として指定されます。 EA名とバージョン番号をセットファイル名に"<expert_name>_<number>.set"として指定すると、パラメータバージョンのダウンロードメニューに<number>バージョン番号の下に自動的に追加されます。たとえば、"MACD Sample_4.set"という名前は、これがバージョン番号が4の"MACD Sample.mq5" EAのセットファイルであることを意味します。 フォーマットを調べるには、ストラテジーテスターでテスト/最適化設定を手動で保存し、この方法で作成したセットファイルを開くことをお勧めします。
tester_no_cache	string	ストラテジーテスターは最適化を実行する際にすべての実行されたテストパスを最適化キャッシュに保存します。ここではテスト結果は入力パラメータセットごとに保存されます。これにより、無駄な時間を再計算にかけることなく、再最適化中に同じパラメータに対する既製の結果を使用することができます。 しかし、一部のタスク（例：数学計算）では、既成の結果が最適化キャッシュにあるかどうかにかかわらず、計算を実行する必要があります。この場合、ファイルには*tester_no_cache*プロパティが含まれるべきです。テスト結果はキャッシュに保存されるため、実行されたテストパスのすべてのデータはストラテジーテスターで確認できます。
tester_everytick_calculate	string	ストラテジーテスターでは、指標はそのデータがアクセスされるとき、すなわち指標バッファの値が要求されたときにのみ計算されます。各ティックで指標値を取得する必要がない場合、これによってテストと最適化の速度が大幅に上がります。 *tester_everytick_calculate*プロパティを指定することによって、全てのディックでの指標の計算を強要できます。 ストラテジーテスターの指標は、以下の場合にはすべてのティックで強制的に計算されます。 • 可視化モードでのテスト

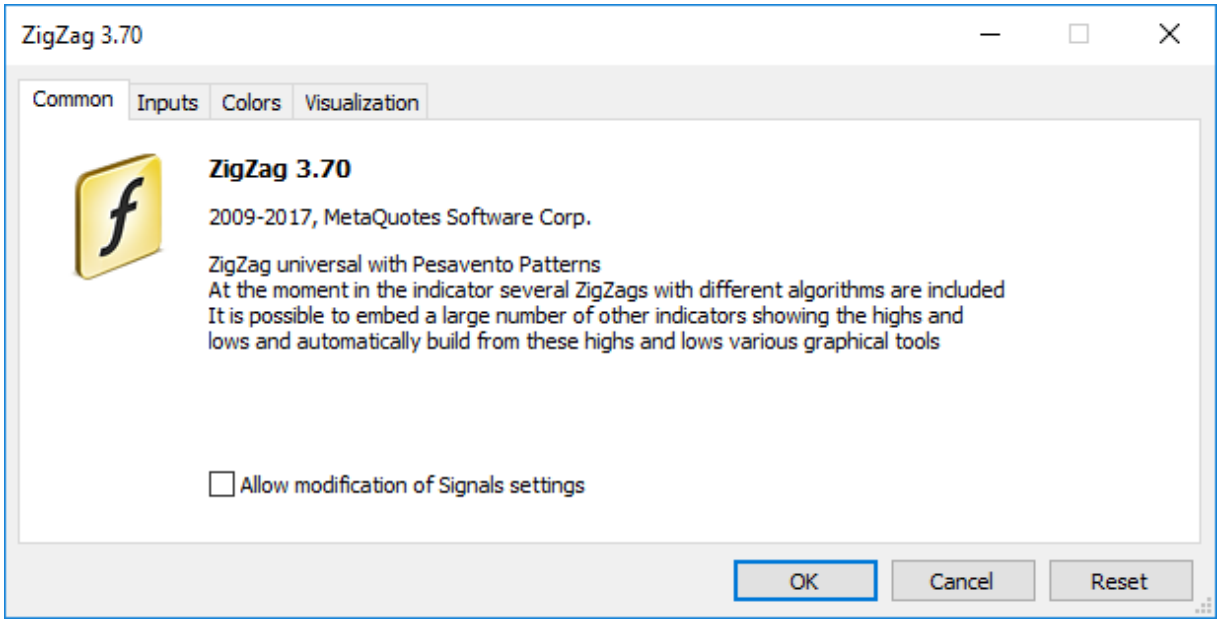
© 2000–2025, MetaQuotes Ltd.

定数	Type	説明
		• 指標にEventChartCustom、OnChartEvent、OnTimer関数がある • 指標はビルド番号1916以前のコンパイラで作成された この関数はストラテジーテスターにのみ適用され、端末での指標は常に受信したティックごとに計算されます。
optimization_chart_mode	string	最適化結果の視覚化に使用されるチャートタイプと2つの入力パラメータの名前を指定します。たとえば、「3d, InpX, InpY」は、テストされたInpXおよびInpYパラメータ値に基づいた座標軸を使用して、結果が3Dチャートに表示されることを意味します。したがって、このプロパティを使用すると、最適化チャートとチャートタイプをプログラムコードで直接表示するために使用されるパラメータを指定できます。 可能なオプション: • 「3d, input_parameter_name1, input_parameter_name2」は3D視覚化チャートを意味し、回転、拡大、縮小が可能です。チャートは2つのパラメータを使用して作成されます。 • 「2d, input_parameter_name1, input_parameter_name2」は、各セルが結果に応じて特定の色で描画される2Dグリッドチャートを意味します。チャートは2つのパラメータを使用して作成されます。 • 「1d, input_parameter_name1, input_parameter_name2」は線形チャートを意味し、結果は指定されたパラメータでソートされます。各パスはポイントとして表示されます。チャートは1つのパラメータに基づいて作成されます。 • 「0d, input_parameter_name1, input_parameter_name2」は、パス結果の到着時間でソートされた結果を持つ通常のチャートを意味します。各パスはチャートのポイントとして表示されます。パラメータの表示は必須ではありませんが、指定されたパラメータは他のチャートタイプへの手動切り替えに使用できます。 オプションで、1つまたは2つの入力パラメータを指定せずに、チャートタイプのみを指定できます。この場合、ターミナルは必要なパラメータを選択して最適化チャートを表示します。

説明とバージョン番号に関するサンプルタスク

© 2000–2025, MetaQuotes Ltd.

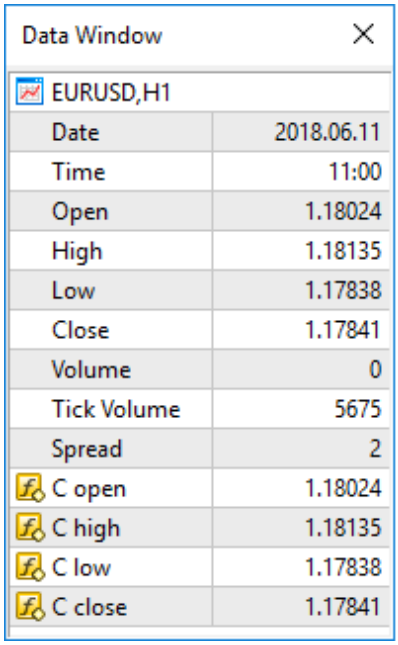
```
#property version     "3.70"      // エキスパートアドバイザーの現在のバージョン
#property description "ZigZag universal with Pesavento Patterns"
#property description "At the moment in the indicator several ZigZags with different algorithms are included"
#property description "It is possible to embed a large number of other indicators showing the highs and"
#property description "lows and automatically build from these highs and lows various graphical tools"
```



各指標バッファ用に別々のラベルを指定する例（"C open; C high; C low; C close"）

```
#property indicator_chart_window
#property indicator_buffers 4
#property indicator_plots   1
#property indicator_type1   DRAW_CANDLES
#property indicator_width1  3
#property indicator_label1  "C open;C high;C low;C close"
```

© 2000–2025, MetaQuotes Ltd.





© 2000–2025, MetaQuotes Ltd.

ファイルの読み込み（#include）

#include コマンドラインはプログラムのどこにでも配置することが出来ますが、通常は全てがソースコードの先頭に配置されます。呼び出し形式は下記の通りです。

```
#include <ファイル名>
#include "ファイル名"
```

例：

```
#include <WinUser32.mqh>
#include "mylib.mqh"
```

プリプロセッサは、*#include* *<ファイル名>* をWinUser32.mqh の内容に入れ替えます。山カッコはWinUser32.mqh ファイルが標準ディレクトリ（通常 *terminal_installation_directory\MQL5\Include*）から取得されることを示します。現在のディレクトリは検索に含まれません。

ファイル名は引用符で囲まれている場合、検索は（メインソースファイルが含まれている）現在のディレクトリで行われます。標準ディレクトリは検索に含まれません。

参照

標準ライブラリ、関数のインポート

© 2000–2025, MetaQuotes Ltd.

関数のインポート（#import）

関数はコンパイル済みの MQL5 モジュール（ *.ex5 ファイル ）及びオペレーティングシステムモジュール（ *.dll ファイル ）からインポートされます。モジュール名は *#import* ディレクティブで指定されます。コンパイラがインポートされた関数の呼び出しを正しく形成し、適切な伝達パラメータを整理するには、関数 の完全な記述が必要とされています。関数の記述は *#import "モジュール名"* ディレクティブの直後に続きます。新しいコマンド *#import*（パラメータなし）がインポートされた関数の記述ブロックを完了します。

```
#import "file_name"
    func1 define;
    func2 define;
    ...
    funcN define;
#import
```

インポートされた関数は任意の名称を持つことが出来ます。異なるモジュールからの同名の関数も同時にインポートされることが出来ます。インポートされた関数は、組み込み関数の名称と同じ名称を持つことが出来ます。スコープ解決演算がどちらの関数が呼び出されるかを定義します。

#import キーワードの後に指定されたファイルの検索順序は、インポートした関数の呼び出しで説明されています。

インポートされた関数はコンパイルされたモジュールの外側にあるので、コンパイラは渡されたパラメータの妥当性を検証することは出来ません。そのため、ランタイムエラーを回避するためには、インポートされた関数に渡されたパラメータの組成と順序を正確に記述する必要があります。（EX5とDLLモジュールから）インポートされた関数に渡されたパラメータは、初期値を持つことが出来ます。

以下は、インポートされた関数のパラメータとして使用することは出来ません。

- ポインタ（ * ）
- 動的配列を含むオブジェクトへのリンク及び/またはポインタ

クラス、文字列、または文字列及び/または任意の型の動的配列を含んでいる複合オブジェクトは、DLLからインポートされた関数にパラメータとして渡すことはできません。

例：

```
#import "stdlib.ex5"
string ErrorDescription(int error_code);
int    RGB(int red_value,int green_value,int blue_value);
bool   CompareDoubles(double number1,double number2);
string DoubleToStrMorePrecision(double number,int precision);
string IntegerToHexString(int integer_number);
#import "ExpertSample.dll"
int    GetIntValue(int);
double GetDoubleValue(double);
string GetStringValue(string);
double GetArrayItemValue(double &arr[],int,int);
bool   SetArrayItemValue(double &arr[],int,int,double);
double GetRatesItemValue(double &rates[][6],int,int,int);
#import
```

© 2000–2025, MetaQuotes Ltd.

MQL5 プログラムの実行時に関数をインポートするには、事前結合が使用されます。これは、ライブラリはそれを利用する EX5 プログラムの読み込み時に読み込みされることを意味します。

Drive:\Directory\FileName.Ext のように読み込まれるモジュールの完全修飾名を使用することは推薦されていません。MQL5 ライブラリは*terminal_dir\MQL5\Libraries* フォルダから読み込まれます。

インポートされた関数の呼び出しバージョンが32ビット版と64ビット版Windowsで異なる場合は、両方ともインポートし、_IsX64変数を使って正しい関数バージョンを明確的に呼び出す必要があります。

例:

```
#import "user32.dll"
//--- 32ビットシステム
int    MessageBoxW(uint hWnd,string lpText,string lpCaption,uint uType);
//--- 64ビットシステム
int    MessageBoxW(ulong hWnd,string lpText,string lpCaption,uint uType);
#import
//+------------------------------------------------------------------+
//|  MessageBox_32_64_bit は MessageBoxW() の適切なバージョンを使用する  |
//+------------------------------------------------------------------+
int MessageBox_32_64_bit()
  {
   int res=-1;
   //--- 64 ビット Windows 使用の場合
   if(_IsX64)
     {
      ulong hwnd=0;
      res=MessageBoxW(hwnd,"64-bit MessageBoxW call example","MessageBoxW 64 bit",MB_OK|MB_ICONINFORMATION);
     }
   else  // 32 ビット Windows 使用の場合
     {
      uint hwnd=0;
      res=MessageBoxW(hwnd,"32-bit MessageBoxW call example","MessageBoxW 32 bit",MB_OK|MB_ICONINFORMATION);
     }
   return (res);
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int ans=MessageBox_32_64_bit();
   PrintFormat("MessageBox_32_64_bit returned %d",ans);
  }
```

.NETライブラリからの関数のインポート

© 2000-2025, MetaQuotes Ltd.

.NETライブラリ関数を使用するには、特定の関数を定義せずにDLL自体をインポートします。MetaEditorは自動的に使用される関数をインポートします。

- 単純な構造体(POD、plain old data) — 単純なデータ型のみを含む構造体
- 単純な型とPOD構造体またはその配列のみが使用されるパラメータを持つpublic static関数

ライブラリからの関数を呼び出すには、単にインポートします。

```
#import "TestLib.dll"
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int x=41;
   TestClass::Inc(x);
   Print(x);
  }
```

TestClassのInc関数のC#コードは下記のとおりです。

```
public class TestClass
  {
   public static void Inc(ref int x)
     {
      x++;
     }
  }
```

実行されると、スクリプトは42を返します。

参照

ファイルの読み込み

© 2000–2025, MetaQuotes Ltd.

条件付きコンパイル(#ifdef、#ifndef、#else、#endif)

プリプロセッサディレクティブはコンパイルする前にソースコードを前処理するためにコンパイラによって使用されます。ディレクティブは常に#で始まります。したがって、コンパイラは変数名、関数名などでのこのシンボルの使用を禁止します。

各ディレクティブはエントリごとに指定されます。1つのエントリで複数のディレクティブを使用することはできません。ディレクティブエントリが大きすぎる場合は、¥記号を使用して複数の行に分割します。この場合、次の行はディレクティブエントリの続きと見なされます。

プリプロセッサの条件付きコンパイルディレクティブは、特定の条件の成立に応じて、プログラムの一部をコンパイルまたはスキップすることを可能にします。

条件は、次のいずれかの形式を取ることが出来ます。

```
#ifdef 識別子
   //ここにあるコードは、識別子がすでに #define ディレクティブのプリプロセッサに定義されている場合にコンパイルされます。
#endif
```

```
#ifndef 識別子
   // ここにあるコードは、識別子がすでに #define ディレクティブのプリプロセッサに定義されていない場合にコンパイルされます。
#endif
```

条件付きコンパイルディレクティブは複数の行で続けられ、任意の #else を含んで #endif で終わります。検証条件が true の場合、#else と #endif の間の行は無視されます。検証条件が満たされていない場合は、チェックと #else ディレクティブ(#else が存在しない場合は #endif ディレクティブ)の間の行は全て無視されます。

例:

```
#ifndef TestMode
   #define TestMode
#endif
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
   #ifdef TestMode
      Print("Test mode");
   #else
      Print("Normal mode");
   #endif
  }
```

標準マクロは、プログラムの種類とコンパイルモードに応じて以下のように定義されています。

__MQL5__ マクロは *.mq5 ファイル、__MQL4__ は *.mq4 ファイルをコンパイルする時に定義されます。
_DEBUG マクロはデバグモードでコンパイルする時に定義されます。
_RELEASE マクロはリリースモードでコンパイルする時に定義されます。

© 2000-2025, MetaQuotes Ltd.

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
   #ifdef __MQL5__
      #ifdef _DEBUG
         Print("Hello from MQL5 compiler [DEBUG]");
      #else
        #ifdef _RELEASE
           Print("Hello from MQL5 compiler [RELEASE]");
        #endif
     #endif
   #else
      #ifdef __MQL4__
         #ifdef _DEBUG
           Print("Hello from MQL4 compiler [DEBUG]");
        #else
           #ifdef _RELEASE
              Print("Hello from MQL4 compiler [RELEASE]");
           #endif
        #endif
     #endif
   #endif
  }
```

© 2000–2025, MetaQuotes Ltd.

オブジェクト指向プログラミング

オブジェクト指向プログラミング（OOP）は、データと動作が密接に繋がっていながら、主に、データに焦点を当てたプログラミングです。オブジェクトがクラスインスタンスでありながら、クラスはデータと操作演算の両方によって構成されます。

オブジェクト指向のアプローチのコンポーネントは次の通りです。

- 型の封装と拡張
- 継承
- ポリモーフィズム
- 多重定義（オーバーロード）
- 仮想関数

OOP は演算を操作演算のモデル化とみなします。モデル化されているのは、計算抽象化によって表されるオブジェクトです。よく知られているゲーム「テトリス」を書きたいとしましょう。これを行うためには、4 つの正方形で構成されて辺で接合されたランダムな図形をモデル化する方法を学ぶ必要があります。また図形の落下速度を調整し、図形の回転と移動の動作を定義する必要があります。画面上の図形の移動は井戸の境界によって限られており、この要件もモデル化されなければなりません。それに加えて、正方形のそろった行を破棄して得点を数える必要があります。

このように、この理解しやすいゲームは、図形モデル、井戸モデル、図形運動モデルといったいくつかのモデル作成を要します。これらの全てのモデルは抽象で、コンピュータでの計算で示されます。これらのモデルを説明するためには抽象データ型、ADT（またはコンプレックスデータ型）の概念が使用されます。厳密に言うと、DOM 内の「図形」運動のモデルはデータ型ではなく、「井戸」データ型の制限を使用した「図形」のデータ型に対する操作のセットです。

オブジェクトは クラス 変数です。オブジェクト指向プログラミングは、ADTの作成と使用を簡単にします。オブジェクト指向プログラミングは、継承メカニズムを使用します。継承の利点は、すでにユーザによって定義されたデータ型から派生型を得ることが可能になることです。

例えば、テトリス図形を作成するには、まず Shape 基本クラスを作成すると便利です。全ての 7 つの可能な図形の型を表現する他のクラスは、その基礎の上に作られます。図形の動作は基本クラスで定義され、各別個の図形の動作は派生クラスで実装されます。

OOPでは、オブジェクト自身がそれぞれの動作に責任を持ちます。ADTの開発者は、対応するオブジェクトから通常予測される動作を説明するためのコードを含むべきです。オブジェクト自体がその動作に責任を持つという事実は、このオブジェクトのプログラミング作業を非常に簡素化します。

画面上に図形を描画したい場合、どこが中心になってそれをどのように描画するかを知る必要があります。別々の図形がそれ自身を描画する方法を知っていれば、プログラマはこのような図形を使用した場合、「draw」メッセージを送信出来ます。

MQL5 言語は C++ に似ており ADT 実施のためにカプセル化メカニズムを備えています。カプセル化は特定の型の実装の内部の詳細を結合する一方、この型のオブジェクトに影響を与えることが出来る外部からアクセス可能の関数を組み合わせます。実装の詳細はこの型を使用するプログラムのからはアクセス出来ません。

OOP の概念は次のような関連する概念のセットで成り立ちます。

- 現実世界アクションのシミュレーション

© 2000–2025, MetaQuotes Ltd.

- ユーザ定義のデータ型
- 実装の詳細の隠蔽
- 継承によるコード再利用の可能性
- 実行時の関数呼び出しの解釈

これらの概念のいくつかはかなり曖昧、いくつかは抽象的でいくつかは一般的です。

© 2000–2025, MetaQuotes Ltd.

型の封装と拡張

OOP はソフトウェアを書くのにバランスのとれたアプローチです。データと操作演算は一体化されています。このカプセル化は、ユーザ定義のデータ型を作成し、言語データ型を拡張しそれらと相呼応します。型の拡張は言語に使いやすいユーザ定義のデータ型、ならびに基本型を追加する機会です。

例えば抽象データ型の文字列は、理想的なよく知られた操作演算の種類を描画します。

文字列のユーザは、連結やプリントなどの文字列操作が特定の動作を持っていることを知っています。このような操作はメソッドと呼ばれます。

ADT の特定の実装は、例えば、文字列の長さの制限などを有することがあります。これらの制限は全ての操作演算に影響を与えます。同時に、内部またはプライベート実装の詳細はユーザがオブジェクトを見る方法には直接影響しません。例えば、文字列は多くの場合配列として実装されていて、配列の内部ベースアドレスや名称はユーザには必要でありません。

カプセル化はユーザ定義型へのオープンインタフェースが提供された時に実装の詳細を隠す機能です。MQL5 言語ではクラスと構造体の定義（classと struct ）は private、protected 及び public のアクセスキーワードとともにカプセル化の規定に使用されています。

public キーワードは、キーワードの使用されたメンバへのアクセスが制限なくオープンなことを示します。このキーワードが指定されなかった場合、クラスメンバは、デフォルトでロックされています。private メンバは、そのクラスのメンバ関数のみからアクセス出来ます。

Protected クラス関数はこのクラスでだけでなくその継承クラスからも利用可能です。Public クラス関数は、クラス宣言の有効範囲内の全ての関数から利用可能です。この様な保護は、クラスの実装の一部を隠してデータの構造の予期しない変更を防止します。アクセス制限やデータ隠蔽はオブジェクト指向プログラミングの特徴です。

通常、クラス関数は保護されており protected 修飾子を使用して宣言されます。値の読み書きは public 修飾子を用いて定義された特殊ないわゆるセットメソッドとゲットメソッドを用いて行われます。

例：

```
class CPerson
  {
protected:
   string            m_name;                    // 名称
public:
   void              SetName(string n){m_name=n;}// 名称を付ける
   string            GetName(){return (m_name);} // 名称を返す
  };
```

このアプローチにはいくつかの利点があります。まず、関数名で関数の機能がわかります。つまりクラスメンバの値をセットまたはゲットするものです。次に、将来的に CPerson クラスまたはその派生クラスのいずれかで m_name 変数の型を変更する必要があるかもしれません。

この場合必要なのは SetName() と GetName() 関数の実装の変更だけです。m_name のデータ型の変更はユーザに知られる余地もなく、CPerson クラスのオブジェクトはコード変更なしでプログラムで使用できます。

© 2000–2025, MetaQuotes Ltd.

例:

```
struct Name
  {
   string            first_name;                  // 名
   string            last_name;                   // 姓
  };

class CPerson
  {
protected:
   Name              m_name;                      // 名前
public:
   void              SetName(string n);
   string            GetName(){return(m_name.first_name+" "+m_name.last_name);}
private:
   string            GetFirstName(string full_name);
   string            GetLastName(string full_name);
  };

void CPerson::SetName(string n)
  {
   m_name.first_name=GetFirstName(n);
   m_name.last_name=GetLastName(n);
  }

string CPerson::GetFirstName(string full_name)
  {
   int pos=StringFind(full_name," ");
   if(pos>0) StringSetCharacter(full_name,pos,0);
   return(full_name);
  }

string CPerson::GetLastName(string full_name)
  {
   string ret_string;
   int pos=StringFind(full_name," ");
   if(pos>0) ret_string=StringSubstr(full_name,pos+1);
   else      ret_string=full_name;
   return(ret_string);
  }
```

参照

データ型

© 2000–2025, MetaQuotes Ltd.

継承

継承によるコードの再利用の促進は OOP の特徴です。新しいクラスは、基本クラスと呼ばれる既存するクラスから作られます。派生クラスは基本クラスのメンバを使用するだけでなくそれらを変更したり補完したりすることが出来ます。

多くの型は既存の型のバリエーションです。多くの場合、各バリエーションのために新しいコードを開発することは面倒です。また、新しいコードは新しいエラーを意味します。派生クラスは基本クラスの記述を継承するのでコードの再開発と再テストは不要です。継承関係は階層構造を持ちます。

階層構造は、要素の全ての多様性と複雑さを複製することを可能にします。それはオブジェクトの分類を紹介します。例えば、元素表はガスを含みます。ガスは全ての元素に固有の特性を保有します。

不活性ガスは次に重要なサブクラスを構成します。ここでみられる階層は、アルゴン等の不活性ガスがガスで、ガスはひいてはシステムの一部であるということです。このような階層を使用すれば、不活性ガスの挙動が簡単に解釈出来ます。不活性ガスは他の全ての要素と同様に原子に陽子と電子があることがわかります。

不活性ガスは全てのガスのように常温で気体状態であることもわかります。不活性ガスのサブクラスに属するガスは全ての不活性ガスの特性であるように他の元素との通常の化学反応に入らないこともわかります。

幾何学的図形の継承の例を考えてみましょう。数々の単純な図形を記述するのに(円形、三角形、四角形、正方形等)一番良い方法は、全ての派生クラスの祖先である基本クラスを作成することです(ADT)。

図形を記述する最も一般的なメンバを含む基本クラス CShape を作成してみましょう。これらのメンバは図形の特徴であるプロパティ(図形の種類とメインアンカーポイントの座標)を記述します。

例:

```
//--- 基本クラス Shape
class CShape
  {
protected:
   int       m_type;                    // 図形の種類
   int       m_xpos;                    // 基点の X 座標
   int       m_ypos;                    // 基点の Y 座標
public:
             CShape(){m_type=0; m_xpos=0; m_ypos=0;} // コンストラクタ
   void      SetXPos(int x){m_xpos=x;} // X を設定
   void      SetYPos(int y){m_ypos=y;} // Y を設定
  };
```

次に、基本クラスから派生した新しいクラスを作成し、それぞれのクラスを指定するのに必要なフィールドを追加します。円形の場合には、半径値が含まれているメンバを追加する必要があります。正方形は側面値によって特徴付けられます。従って、基本クラスの CShape から継承された派生クラスは次のように宣言されます。

```
//--- 派生クラス circle
class CCircle : public CShape         // コロンの後に基本クラスを定義する
  {                                   // 継承の元
private:
```

© 2000-2025, MetaQuotes Ltd.

```
   int               m_radius;            // 円の半径

public:
                     CCircle(){m_type=1;}// コンストラクタ タイプ 1
  };
```

正方形クラスの宣言も似ています。

```
//--- 派生クラス Square
class CSquare : public CShape         // コロンの後で基本クラスを定義する
  {                                   // 継承の元
private:
   int             m_square_side;     // 正方形の側面

public:
                     CSquare(){m_type=2;} // コンストラクタ(タイプ 2)
  };
```

オブジェクト作成時に基本クラスのコンストラクタが最初に呼び出されその後で派生クラスのコンストラクタが呼び出されることには留意するべきです。オブジェクト破壊時には派生クラスのデストラクタが呼び出されてから基本クラスのデストラクタが呼び出されます。

従って、最も一般的なメンバを基本クラスの中で宣言することによって、派生クラスに特定のクラスを指定するメンバを追加することが出来ます。継承は何度も再利用することが出来る力強いコードライブラリの作成を可能にします。

既存クラスから派生クラスを作成するための構文は次の通りです。

```
class クラス名 :
          (public | protected | private) opt 基本クラス名
  {
   クラスメンバの宣言
  };
```

派生クラスの側面の 1 つは、そのメンバの後継者の視認性(オープンネス)です。public、protected 及び private キーワードは 基本クラスのメンバが派生クラスにどの程度使用可能かを示すものです。派生クラスヘッダーのコロンの後の public キーワードは、基本クラス CShape の protected 及び public メンバが派生クラス CCircle の protected 及び public メンバとして継承される必要があることを示します。

基本クラスの private クラスメンバは、派生クラスでは使用出来ません。public 継承は、派生クラス(CCircle と CSquare)が CShapes であることを意味します。つまり、Square(CSquare)は図形(CShape)ですが、その図形は必ずしも正方形である必要はありません。

派生クラスは基本クラスのバージョン例で基本クラスの protected と public メンバを継承します。基本クラスのコンストラクタとデストラクタは継承出来ません。基本クラスのメンバに加えて、派生クラスには新しいメンバが追加されます。

派生クラスは、基本クラスとは違うメンバ関数の実装を含むことが出来ます。これは、同名の関数の意味がシグネチャの違いによって異なるものになるオーバーロードとは全く違うものです。

© 2000-2025, MetaQuotes Ltd.

protected 継承では、基本クラスの public 及び protected メンバは派生クラスの protected メンバになります。private 継承では、基本クラスの public 及び protected メンバは派生クラスの private メンバになります。

protected 及び private 継承では「派生クラスのオブジェクトは基本クラスのオブジェクトである」という関係は当てはまりません。protected 及び private 継承タイプはまれであり慎重に使用される必要があります。

継承の種類（public、protected 及び private）は派生クラスが基本クラスメンバにアクセスする方法に影響を及ぼさないことは留意されるべきです。全ての継承のタイプで、public 及び protected のアクセス指定とともに宣言された基本クラスのメンバのみが派生クラスによって利用出来ます。次の例を見てみましょう。

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//+------------------------------------------------------------------+
//| いくつかのアクセスタイプを持つクラスの例                                |
//+------------------------------------------------------------------+
class CBaseClass
  {
private:              //--- private メンバは派生クラスから利用出来ない
   int                m_member;
protected:            //--- 基本クラスの protected メソッドは派生クラスでも利用出来る
   int                Member(){return(m_member);}
public:               //--- クラスコンストラクタは全てのクラスメンバから利用出来る
                      CBaseClass(){m_member=5;return;};
private:              //--- m_member に値を代入する private メソッド
   void               Member(int value) { m_member=value;};

  };
//+------------------------------------------------------------------+
//| 間違えを含む派生クラス                                              |
//+------------------------------------------------------------------+
class CDerived: public CBaseClass // 継承は public がデフォルトであるため、public の指定は省略出来る
  {
public:
   void Func() // 派生クラスで、基本クラスメンバを呼び出す関数を定義する
     {
      //--- 基本クラスの private メンバを変更する試み
      m_member=0;         // エラー。基本クラスの private メンバは利用出来ない
      Member(0);          // エラー。基本クラスの private メソッドは派生クラスで利用出来ない
      //--- 基本クラスメンバを読む
      Print(m_member);    // エラー。基本クラスの private メンバは利用出来ない
      Print(Member());    // 成功。基本クラスの protected メソッドは派生クラスで使用出来る
     }
  };
```

上記の例では CBaseClass の唯一の public メソッドはコンストラクタです。コンストラクタはクラスオブジェクトを作成する時に自動的に呼び出されます。従って、private メンバ m_member と protected メソッド

© 2000-2025, MetaQuotes Ltd.

Member() は外から呼び出すことは出来ません。しかし public 継承では、基本クラスの Member() メソッドは派生クラスから利用出来るようになります。

protected 継承では、基本クラスの public 及び protected メンバは派生クラスの protected メンバになります。これは、もし基本クラスの public データメンバとメソッドが外からアクセス出来たとすると、protected 継承ではそれらは派生クラスまた派生クラスの派生クラスなどによってのみ利用可能なことを意味します。

```
//+------------------------------------------------------------------+
//|  いくつかのアクセスタイプを持つクラスの例                                |
//+------------------------------------------------------------------+
class CBaseMathClass
  {
private:              //--- private メンバは派生クラスから利用出来ない
   double             m_Pi;
public:               //--- m_Pi  値の取得と設定
   void               SetPI(double v){m_Pi=v;return;};
   double             GetPI(){return m_Pi;};
public:               // クラスコンストラクタは全てのメンバから利用出来る
                      CBaseMathClass() {SetPI(3.14);  PrintFormat("%
s",__FUNCTION__);};
  };
//+------------------------------------------------------------------+
//|  m_Pi  が変更出来ない 派生クラス                                      |
//+------------------------------------------------------------------+
class CProtectedChildClass: protected CBaseMathClass // Protected 継承
  {
private:
   double             m_radius;
public:               //--- 派生クラスの Public メソッド
   void               SetRadius(double r){m_radius=r; return;};
   double             GetCircleLength(){return GetPI()*m_radius;};
  };
//+------------------------------------------------------------------+
//|  スクリプトを開始する関数                                             |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 派生クラスの作成時に、基本クラスのコンストラクタは自動的に呼び出される
   CProtectedChildClass pt;
//--- 半径を指定する
   pt.SetRadius(10);
   PrintFormat("Length=%G",pt.GetCircleLength());
//--- SetPI() は protected であるため、以下の文字列をコメントした場合コンパイルの段階でエラーが発生する
// pt.SetPI(3);

//--- 基本クラスの変数を宣言し Pi  定数を 10 に設定してみる
   CBaseMathClass bc;
   bc.SetPI(10);
//---結果
```

© 2000-2025, MetaQuotes Ltd.

```
   PrintFormat("bc.GetPI()=%G",bc.GetPI());
  }
```

この例は基本クラス CBaseMathClass の SetPI() 及び GetPI() メソッドがプログラムのどこからでも呼び出せることを示します。 しかし同時に、それらから派生する CProtectedChildClass では、これらのメソッドは CProtectedChildClass クラスまたはその派生クラスのメソッドのみからしか呼び出せません。

private 継承では基本クラスの public 及びprotected メンバは派生クラスの private メンバになり、それ以上の継承を通してそれらを呼び出すことは不可能です。

MQL5 言語は多重継承を持っていません。

参照

構造体とクラス

© 2000-2025, MetaQuotes Ltd.

ポリモーフィズム

ポリモーフィズムは継承によって繋がったオブジェクトクラスが同じ関数要素を呼び出す時に使用されます。これは基本クラスだけでなくその子孫クラスの振る舞いも記述する普遍的なメカニズムを作成するのに役立ちます。

図形の面積を計算するために設計された基本クラス CShape を開発しメンバ関数 GetArea() を定義していきましょう。基本クラスからの継承によって生成された子孫クラスの全てで、特定の図形の面積を計算する規則に従って、この関数を定義し直します。

正方形（CSquare クラス）では面積がその辺を介して計算され、円（CCircle クラス）では面積が半径で表現されます。基本クラスと全ての子孫クラスの CShape 型のオブジェクトを格納出来る配列を作成することが出来ます。更に配列の各要素に同じ関数を呼び出すことが出来ます。

例：

```
//--- 基本クラス
class CShape
  {
protected:
   int               m_type;                 // 図形の種類
   int               m_xpos;                 // 基点の X 座標
   int               m_ypos;                 // 基点の Y 座標
public:
   void              CShape(){m_type=0;};    // コンストラクタ(タイプ 0 )
   int               GetType(){return(m_type);};// 図形の種類を返す
virtual
   double            GetArea(){return (0); }// 図形の面積を返す
  };
```

これで、派生クラスの全てがゼロの値を返すメンバ関数 getArea() を持つことになります。この関数の実装は各子孫関数内で異なります。

```
//--- 派生クラス circle
class CCircle : public CShape              // コロンの後に基本クラスを定義する
  {                                        // 継承の元
private:
   double            m_radius;             // 円の半径

public:
   void              CCircle(){m_type=1;};  // コンストラクタ(タイプ 1 )
   void              SetRadius(double r){m_radius=r;};
   virtual double GetArea(){return (3.14*m_radius*m_radius);}// 円の面積
  };
```

Square クラスの宣言も同じです。

```
//--- 派生クラス Square
class CSquare : public CShape              // コロンの後に基本クラスを定義する
  {                                        // 継承の元
private:
```

© 2000-2025, MetaQuotes Ltd.

```
   double           m_square_side;         // 正方形の側面

public:
   void             CSquare(){m_type=2;}; // コンストラクタ(タイプ 1 )
   void             SetSide(double s){m_square_side=s;};
   virtual double   GetArea(){return (m_square_side*m_square_side);}// 正方形の面積
  };
```

正方形と円の面積を計算するためにはそれぞれ m_radius と m_square_side の値が必要となるので、それぞれのクラス宣言に関数 SetRadius() と SetSide() を追加します。

1 つの基本型 CShape から派生した異なる種類のオブジェクト (CCircle と CSquare) が両方 1 つのプログラムで使用されていると仮定します。ポリモーフィズムによって、基本クラス CShape オブジェクトの配列が作成出来ますが、配列の宣言時にはこれらのオブジェクトはまだ未知で型も定義されていません。

配列の各要素に含まれるオブジェクト型はプログラムの実行中に直接決定されます。これは関係するクラスオブジェクトの動的作成を必要とするのでオブジェクトのかわりにオブジェクトポインタを使用することが必要となります。

new 演算子はオブジェクトの動的作成に使用されます。このようなオブジェクトは、個別に明示的に delete 演算子を使用して削除される必要があります。そこで、CShape 型のポインタの配列を宣言し、次のスクリプトの例に示すように各要素に適切な型のオブジェクトを作成します (new クラス名)。

```
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 基本型のオブジェクトポインタの配列を宣言する
   CShape *shapes[5];    // CShape オブジェクトポインタの配列

//--- 配列に派生オブジェクトを書き込む
//--- CCircle 型のオブジェクトポインタを宣言する
   CCircle *circle=new CCircle();
//--- circle ポインタでオブジェクトのプロパティを設定する
   circle.SetRadius(2.5);
//--- shapes[0] にポインタ値を入れる
   shapes[0]=circle;

//--- CCircle オブジェクトをあと 1 つ作成し shapes[1] にそのポインタを書き込む
   circle=new CCircle();
   shapes[1]=circle;
   circle.SetRadius(5);

//--- 意図的に shapes[2] の値を設定するのを忘れる
//circle=new CCircle();
//circle.SetRadius(10);
//shapes[2]=circle;

//--- 使用されない要素は NULL に設定する
```

© 2000-2025, MetaQuotes Ltd.

```
   shapes[2]=NULL;

//--- CSquare オブジェクトを作成し shapes[3] にそのポインタを書き込む
   CSquare *square=new CSquare();
   square.SetSide(5);
   shapes[3]=square;

//--- CSquare オブジェクトを作成し shapes[4] にそのポインタを書き込む
   square=new CSquare();
   square.SetSide(10);
   shapes[4]=square;

//--- ポインタ配列ができたのでサイズを取得する
   int total=ArraySize(shapes);
//--- 配列内のポインタを全てループに渡す
   for(int i=0; i<5;i++)
     {
      //--- 指定されたインデックスのポインタが有効の場合
      if(CheckPointer(shapes[i])!=POINTER_INVALID)
        {
         //--- 図形の種類とスクウェアをログに記録する
         PrintFormat("The object of type %d has the square %G",
               shapes[i].GetType(),
               shapes[i].GetArea());
        }
      //--- ポインタの種類が POINTER_INVALID の場合
      else
        {
         //--- エラーの通知
         PrintFormat("Object shapes[%d] has not been initialized! Its pointer is %
s",
                     i,EnumToString(CheckPointer(shapes[i])));
        }
     }

//--- 作成した全ての動的オブジェクトは削除が必要
   for(int i=0;i<total;i++)
     {
      //--- POINTER_DYNAMIC 型のポインタを持つオブジェクトのみを削除可能
      if(CheckPointer(shapes[i])==POINTER_DYNAMIC)
        {
         //--- 削除の通知
         PrintFormat("Deleting shapes[%d]",i);
         //--- オブジェクトをポインタを通して削除
         delete shapes[i];
        }
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

delete 演算子を使用してオブジェクトを削除する時にはそのポインタの種類のチェックが必要なのでご注意ください。POINTER_DYNAMIC ポインタを持つオブジェクトのみが delete を使用して削除することが出来ます。他のポインタ型の場合はエラーが返されます。

しかし、ポリモーフィズムの定義には、継承時の関数の再定義に加えて、クラス内での 1 つの関数のパラメータで異なる複数の実装が含まれます。これはクラス内に同名で型及び/またパラメータの異なる関数がいくつか存在出来ることを意味します。この場合、ポリモーフィズムは関数オーバーロードを通して実装されます。

参照

標準ライブラリ

© 2000–2025, MetaQuotes Ltd.

多重定義（オーバーロード）

1　つのクラス内で同名であるがパラメータの数が異なる複数のメソッドを定義することは可能です。この場合、メソッドはオーバーロードされたといわれ、その過程は、メソッドオーバーロードと呼ばれています。

メソッドオーバーロードはポリモルフィズム実現の方法の 1 つです。メソッドのオーバーロードは関数のオーバーロードと同じルールに従って行われます。

呼び出された関数が正確に一致していない場合、コンパイラは順次 3 つのレベルで適切な関数を検索します。

1. 自己のクラスメソッド内での検索
2. 自己に最も近い先祖から基本クラスに順にさかのぼって基本クラスのメソッドを検索
3. 他の関数内での検索

もし全てのレベルで正確な一致が 1 つもなく、異なるレベルでいくつかの適切な関数がある場合は、最低レベルでの関数が使用されます。各レベル内で、複数の適切な関数が存在することはありません。

参照

関数の多重定義（オーバーロード）

© 2000–2025, MetaQuotes Ltd.

仮想関数

virtual キーワードは、実行時に動的に基本クラスと派生クラスの関数から適切な関数メンバを選択するメカニズムを提供する関数指定子です。構造体は仮想関数を含むことが出来ません。virtual キーワードは、関数メンバのみの宣言を変更することが出来ます。

仮想関数は通常の関数同様に実行可能な本体を持たなければなりません。呼び出される際にはそのセマンティックは他の関数の物と同様です。

仮想関数は派生クラスでオーバーライドされる場合があります。仮想関数に対してどの関数定義が呼び出されるかは(実行時に)動的に選択されます。基本クラスに仮想関数が含まれ、派生クラスがこの関数の独自のバージョンを持っているのが典型的なケースです。

基本クラスへのポインタは、基本クラスオブジェクトまたは派生クラスのオブジェクトのいずれかを示すことが出来ます。呼び出されるメンバ関数の選択は実行時に行われ、ポインタの型ではなくオブジェクトの型に依存します。派生型のメンバが存在しない場合は、基本クラスの仮想関数がデフォルトで使用されます。

デストラクタは virtual キーワードで宣言されたかされなかったにかかわらずは常に仮想です。

MT5_Tetris.mq5 の例で仮想関数の使用を考えてみましょう。仮想関数 Draw を持つ基本クラス CTetrisShapeis がインクルードファイル MT5_TetisShape.mqh で定義されています。

```
//+------------------------------------------------------------------+
class CTetrisShape
  {
protected:
   int               m_type;
   int               m_xpos;
   int               m_ypos;
   int               m_xsize;
   int               m_ysize;
   int               m_prev_turn;
   int               m_turn;
   int               m_right_border;
public:
   void              CTetrisShape();
   void              SetRightBorder(int border) { m_right_border=border; }
   void              SetYPos(int ypos)          { m_ypos=ypos;           }
   void              SetXPos(int xpos)          { m_xpos=xpos;           }
   int               GetYPos()                  { return(m_ypos);        }
   int               GetXPos()                  { return(m_xpos);        }
   int               GetYSize()                 { return(m_ysize);       }
   int               GetXSize()                 { return(m_xsize);       }
   int               GetType()                  { return(m_type);        }
   void              Left()                     { m_xpos-=SHAPE_SIZE;    }
   void              Right()                    { m_xpos+=SHAPE_SIZE;    }
   void              Rotate()                   { m_prev_turn=m_turn; if(++m_turn>3)
m_turn=0; }
   virtual void      Draw()                     { return;                }
   virtual bool      CheckDown(int& pad_array[]);
   virtual bool      CheckLeft(int& side_row[]);
```

© 2000-2025, MetaQuotes Ltd.

```
   virtual bool      CheckRight(int& side_row[]);
  };
```

また、それぞれの派生クラスでは、この関数は子孫クラスの特性に応じて実施されます。例えば、最初の図形 CTetrisShape1 には独自の Draw() 関数が実装されています。

```
class CTetrisShape1 : public CTetrisShape
  {
public:
   //--- 図形を描画する
   virtual void      Draw()
     {
      int    i;
      string name;
      //---
      if(m_turn==0 || m_turn==2)
        {
         //--- 水平
         for(i=0; i<4; i++)
           {
            name=SHAPE_NAME+(string)i;
            ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos+i*SHAPE_SIZE);
            ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos);
           }
        }
      else
        {
         //--- 垂直
         for(i=0; i<4; i++)
           {
            name=SHAPE_NAME+(string)i;
            ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos);
            ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos+i*SHAPE_SIZE);
           }
        }
     }
  }
```

Square 図形は CTetrisShape6 クラスによって記述され独自の Draw() メソッドを実装します。

```
class CTetrisShape6 : public CTetrisShape
  {
public:
   //--- 図形を描画する
   virtual void      Draw()
     {
      int    i;
      string name;
      //---
      for(i=0; i<2; i++)
```

© 2000–2025, MetaQuotes Ltd.

```
         {
          name=SHAPE_NAME+(string)i;
          ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos+i*SHAPE_SIZE);
          ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos);
         }
       for(i=2; i<4; i++)
         {
          name=SHAPE_NAME+(string)i;
          ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos+(i-2)*SHAPE_SIZE);
          ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos+SHAPE_SIZE);
         }
      }
  };
```

作成したオブジェクトが属するクラスに応じて、このクラスの仮想関数かその派生クラスの仮想関数が呼び出されます。

```
void CTetrisField::NewShape()
  {
//--- 7 つの可能な図形から 1 つをランダムに選んで作成する
   int nshape=rand()%7;
   switch(nshape)
     {
      case 0: m_shape=new CTetrisShape1; break;
      case 1: m_shape=new CTetrisShape2; break;
      case 2: m_shape=new CTetrisShape3; break;
      case 3: m_shape=new CTetrisShape4; break;
      case 4: m_shape=new CTetrisShape5; break;
      case 5: m_shape=new CTetrisShape6; break;
      case 6: m_shape=new CTetrisShape7; break;
     }
//--- 描画する
   m_shape.Draw();
//---
  }
```

override修飾子

override修飾子は、宣言される関数は必ず親クラスのメソッドをオーバーライドする必要があるということを意味しています。この修飾子を使用することで、オーバーライド時のメソッドシグネチャのランダムな変更といったエラーを回避することができます。例として、基底クラスで引数としてint型変数を取るfuncメソッドを定義します。

```
class CFoo
  {
   void virtual func(int x) const { }
  };
```

次にメソッドを派生クラスでオーバーライドします:

© 2000-2025, MetaQuotes Ltd.

```
class CBar : public CFoo
  {
   void func(short x) { }
  };
```

しかしエラーによって引数の型がintからshortに変わります。実際には、このような場合、オーバーライドではなくメソッドのオーバーロードが起こります。オーバーロードされた関数定義のアルゴリズムに従い、特定のケースにおいてコンパイラがオーバーライドされたメソッドの代わりに規定クラスで定義されたメソッドを選択することがあります。

このようなエラーを回避する為には、明示的にオーバーライドメソッドにoverride修飾子を追加する必要があります。

```
class CBar : public CFoo
  {
   void func(short x) override { }
  };
```

オーバーライド時にメソッドシグネチャが変更された場合、コンパイラは親クラスで全く同じシグネチャを見つけることができず、コンパイルエラーを出します。

```
'CBar::func' method is declared with 'override' specifier but does not override any
base class method
```

final修飾子

final修飾子は逆に、派生クラスでのメソッドのオーバーライドを禁止します。メソッドの実装が十分であり、完全に禁止されている場合、派生クラスで変更されないようにするためにfinal修飾子を使って宣言してください。

```
class CFoo
  {
   void virtual func(int x) final { }
  };

class CBar : public CFoo
  {
   void func(int) { }
  };
```

final修飾子を使用したメソッドのオーバーライドを試行すると、上記の例で示したように、コンパイルはエラーを出します。

```
'CFoo::func' method declared as 'final' cannot be overridden by 'CBar::func'
see declaration of 'CFoo::func'
```

参照

標準ライブラリ

© 2000–2025, MetaQuotes Ltd.

クラス及び構造体の静的メンバ

静的メンバ

クラスのメンバは、ストレージクラス修飾子 static を使用して宣言することが出来ます。これらのデータメンバはクラスの全てのインスタンスで共有されて 1 つの場所に格納されます。非静的データメンバは、各クラスのオブジェクト変数のために作成されます。

クラスの静的メンバを宣言することが出来なければこれらのデータをプログラムのグローバルレベルで宣言することが必要になります。これはデータとクラス間の関係を破り、クラスでデータとそれらを処理するためのメソッドを結合するという OOP の基本的なパラダイムと一致しないでしょう。静的メンバは特定のインスタンスに固有でないクラスのデータがクラススコープ内に存在することを可能にします。

静的クラスメンバは特定のインスタンスに依存しないため、それへの参照は次のようになされます。

```
class_name::variable
```

ここで *class_name* がクラス名で *variable* がクラスメンバー名です。

ご覧のように、クラスの静的メンバにアクセスするにはコンテキスト解決演算子 :: が使用されます。クラスのメソッド内で静的メンバにアクセスする際にはコンテキスト演算子は任意です。

クラスの静的メンバは、明示的に所望の値で初期化する必要があります。このためには、クラスの静的メンバはグローバルスコープで初期化また宣言されなければなりません。静的メンバの初期化の順序は、そのグローバルスコープでの宣言の順序に対応します。

例えば、テキスト解析クラス *CParser* があって、処理された単語や文字の合計数を数える必要があるとします。必要なクラスのメンバのみを static として宣言し、それらをグローバルレベルで初期化する必要があります。クラスの全てのインスタンスは、単語や文字の共通のカウンタを使用します。

```
//+------------------------------------------------------------------+
//|  テキスト解析クラス                                              |
//+------------------------------------------------------------------+
class CParser
  {
public:
   static int        s_words;
   static int        s_symbols;
   //--- コンストラクタとデストラクタ
                     CParser(void);
                    ~CParser(void){};
  };
...
//--- グローバルレベルでの Parser クラスの静的メンバの初期化
int CParser::s_words=0;
int CParser::s_symbols=0;
```

静的クラスメンバは *const* キーワードを使用して宣言されます。静的定数はグローバルレベルで *const* キーワードを使用して初期化されます。

```
//+------------------------------------------------------------------+
//|  処理されたデータを格納するためのスタッククラス                  |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
class CStack
  {
public:
                     CStack(void);
                    ~CStack(void){};
...
private:
   static const int  s_max_length; // スタックの最大許容
  };

//--- CStack クラスの静的定数の初期化
const int CStack::s_max_length=1000;
```

this ポインタ

キーワード this はそれ自体（つまり、メソッドが実行されたコンテキスト内の特定のクラスインスタンス）に暗黙的に宣言されたポインタを示します。これは、クラスの非静的メソッド内のみで使用することが出来ます。this ポインタはクラスの暗黙的な非静的メンバです。

静的関数では、クラスの静的メンバと静的メソッドのみにアクセスすることが出来ます。

静的メソッド

MQL5 では static 型のメンバ関数が使用出来ます。*static* 修飾子は、クラス内の宣言では関数の戻り値の型を先行しなければなりません。

```
class CStack
  {
public:
   //--- コンストラクタとデストラクタ
                     CStack(void){};
                    ~CStack(void){};
   //--- 最大スタック容量
   static int        Capacity();
private:
   int               m_length;      // スタック内の要素数
   static const int  s_max_length; // スタックの最大許容
  };
//+------------------------------------------------------------------+
//| スタックに格納出来る要素の最大数を返す                           |
//+------------------------------------------------------------------+
int CStack::Capacity(void)
  {
   return(s_max_length);
  }
//--- CStack クラスの静的定数の初期化
const int CStack::s_max_length=1000;
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                               |
```

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 型の変数を宣言
   CStack stack;
//--- オブジェクトの静的メソッドを呼び出す
   Print("CStack.s_max_length=",stack.Capacity());
//--- メソッドが静的でありオブジェクトの存在を必要としないため次のような呼び出しも可能
   Print("CStack.s_max_length=",CStack::Capacity());
  }
```

const 修飾子を持つメソッドは定数と呼ばれ、そのクラスの暗黙のメンバを変更することは出来ません。クラスの定数関数と定数パラメータの宣言は*定数コレクトネス*制御と呼ばれます。この制御により、コンパイラがオブジェクトの値の一貫性を確保し、問題がある場合にはコンパイル時にエラーを返すことを確認出来ます。

const 修飾子はクラス宣言内で引数リストの後に配置されます。クラス外での定義も *const* 修飾子を含む必要があります。

```
//+------------------------------------------------------------------+
//|  四角形クラス                                                    |
//+------------------------------------------------------------------+
class CRectangle
  {
private:
   double            m_width;       // 幅
   double            m_height;      // 高さ
public:
   //--- コンストラクタとデストラクタ
                     CRectangle(void):m_width(0),m_height(0){};
                     CRectangle(const double w,const double
h):m_width(w),m_height(h){};
                    ~CRectangle(void){};
   //--- 面積の計算
   double            Square(void) const;
   static double     Square(const double w,const double h);// { return(w*h); }
  };
//+------------------------------------------------------------------+
//|  四角形オブジェクトの面積を返す                                  |
//+------------------------------------------------------------------+
double CRectangle::Square(void) const
  {
   return(Square(m_width,m_height));
  }
//+------------------------------------------------------------------+
//|  2 つの変数の積を返す                                            |
//+------------------------------------------------------------------+
static double CRectangle::Square(const double w,const double h)
  {
   return(w*h);
```

© 2000–2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 側面が 5 と 6 の四角形 rect を作成する
   CRectangle rect(5,6);
//--- 定数メソッドを使用して四角形の面積を求める
   PrintFormat("rect.Square()=%.2f",rect.Square());
//--- クラス CRectangle の静的メソッドを使用して数値の積を求める
   PrintFormat("CRectangle::Square(2.0,1.5)=%f",CRectangle::Square(2.0,1.5));
  }
```

定数制御を使用するもう 1 つの利点は、コンパイラが、定数オブジェクトを読み出し専用メモリに配置するなどの特別な最適化を適応することです。

const 修飾子は関数を呼び出す時にインスタンスメンバの不変を保証するため、静的な関数には const は使用できません。しかし、前述したように、静的関数は非静的クラスメンバにアクセスすることは出来ません。

参照

静的変数、変数、参照修飾子「 & 」とキーワード「 this 」

© 2000-2025, MetaQuotes Ltd.

関数テンプレート

オーバーロードされた関数は一般的に様々なデータ型に対して同様の演算を行うために使用されます。ArraySize() は MQL5 でのそのような関数の簡単な例で、配列のサイズを返します。実際には、このシステム関数はオーバーロードされており、このようなオーバーロードの実装は全体的に MQL5 アプリケーションの開発者から隠されています。

```
int  ArraySize(
   void&  array[]      // チェックされた配列
   );
```

つまり MQL5 言語コンパイラはこの関数の呼び出しごとに必要な実装を挿入するわけです。例えば、整数型の配列には下記が行われます。

```
int  ArraySize(
   int&  array[]      // int 型要素の配列
   );
```

ArraySize() は履歴データ形式の相場を操作するための MqlRates 型の配列にしては次のように表示されます。

```
int  ArraySize(
   MqlRates&  array[] // MqlRates 型の配列
   );
```

よって、同様の関数を異なる型の演算をするため使用するのが非常に便利です。しかし、準備作業が必要です。つまり、必要な関数は演算される全てのデータ型に対してオーバーロードされるべきです。

これには便利な解決法があります。それぞれのデータ型に同様の演算が実行される場合には、関数テンプレートの使用が可能です。この場合、プログラマは 1 つの関数テンプレートの記述をする必要があります。テンプレートを記述する際は、関数が作業する明確なデータ型の代わりに一部のみの仮パラメータが指定されます。コンパイラは関数が呼び出される時に使用される引数の型に基づいて自動的に各データ型の適切な取り扱いのための関数を生成します。

関数テンプレートの定義は template キーワードで始まり角括弧内の仮パラメータのリストが続きます。仮パラメータには typename キーワードが先行します。仮パラメータ型は組み込み型またはユーザ定義型です。用例は次の通りです。

- 関数の引数の型を指定する
- 関数の戻り値の型を指定する
- 関数定義内で変数を宣言する

テンプレートパラメータの数は8を超えることは出来ません。テンプレート定義の各仮パラメータは、少なくとも 1 度は関数パラメータのリストに表示されるべきです。仮パラメータの名称は一意である必要があります。

数値型（整数や実数）の配列で最高値を検索する関数テンプレートの例は次の通りです。

```
template<typename T>
T ArrayMax(T &arr[])
  {
   uint size=ArraySize(arr);
```

© 2000-2025, MetaQuotes Ltd.

```
   if(size==0) return(0);

   T max=arr[0];
   for(uint n=1;n<size;n++)
      if(max<arr[n]) max=arr[n];
//---
   return(max);
  }
```

このテンプレートは、渡された配列の最高値を求める関数を定義し、この値を結果として返します。MQL5に内蔵された ArrayMaximum() 関数は最高値のインデックスを返すので、それから値そのものを見つけることが可能になります。例：

```
//--- 配列を作成する
   double array[];
   int size=50;
   ArrayResize(array,size);
//---  ランダムな値を書き込む
   for(int i=0;i<size;i++)
     {
      array[i]=MathRand();
     }

//---配列内の最大値のインデックスを見つける
   int max_position=ArrayMaximum(array);
//--- 配列内の最大値を取得する
   double max=array[max_position];
//--- 見つけた値を表示する
   Print("Max value = ",max);
```

従って、配列の最大値を取得するのに必要な 2 つのステップが実行されました。ArrayMax() 関数テンプレートに適切な型の配列を渡すだけで必要な型の結果を得ることが出来ます。最後の 2 行

```
//---配列内の最大値のインデックスを見つける
   int max_position=ArrayMaximum(array);
//--- 配列内の最大値を取得する
   double max=array[max_position];
```

は、返される結果が関数に渡された配列と同じ型を持っている1行に出来ます。

```
//--- 最大値を見つける
   double max=ArrayMax(array);
```

この場合、ArrayMax() 関数によって返される結果の型は自動的に配列の型と一致されます。

さまざまなデータ型を演算出来る汎用メソッドを作成するために引数の型を文字列として取得するには typename キーワードを使用します。データ型を文字列として返す関数の具体例を考えてみましょう。

```
#include <Trade\Trade.mqh>
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|                                                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   CTrade trade;
   double d_value=M_PI;
   int i_value=INT_MAX;
   Print("d_value: type=",GetTypeName(d_value), ",   value=", d_value);
   Print("i_value: type=",GetTypeName(i_value), ",   value=", i_value);
   Print("trade: type=",GetTypeName(trade));
//---
  }
//+------------------------------------------------------------------+
//|  型が文字列として返される                                        |
//+------------------------------------------------------------------+
template<typename T>
string GetTypeName(const T &t)
  {
//--- 型を文字列として返す
   return(typename(T));
//---
  }
```

関数テンプレートはクラスメソッドにも使用することが出来ます。例えば、

```
class CFile
  {
   ...
public:
   ...
   template<typename T>
   uint WriteStruct(T &data);
  };

template<typename T>
uint CFile::WriteStruct(T &data)
  {
   ...
   return(FileWriteStruct(m_handle,data));
  }
```

関数テンプレートは export、virtual 及び #import キーワードと宣言されるべきではありません。

テンプレート関数の多重定義（オーバーロード）

テンプレート関数の多重定義はしばしば必要です。例えば型キャスティングを使用して1番目のパラメータに2番目のパラメータの値を書き込むテンプレート関数があるとします。MQL5ではstringからboolへの型キャス

© 2000–2025, MetaQuotes Ltd.

ティングは不可能です。しかし、テンプレート関数の多重定義を作成すれば、それは可能になります。例は下記です。

```
//+------------------------------------------------------------------+
//|  テンプレート関数                                                |
//+------------------------------------------------------------------+
template<typename T1,typename T2>
string Assign(T1 &var1,T2 var2)
  {
   var1=(T1)var2;
   return(__FUNCSIG__);
  }
//+------------------------------------------------------------------+
//|  bool+stringのための特別な多重定義                               |
//+------------------------------------------------------------------+
string Assign(bool &var1,string var2)
  {
   var1=(StringCompare(var2,"true",false) || StringToInteger(var2)!=0);
   return(__FUNCSIG__);
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                              |
//+------------------------------------------------------------------+
void OnStart()
  {
   int i;
   bool b;
   Print(Assign(i,"test"));
   Print(Assign(b,"test"));
  }
```

コード実行の結果、Assign()テンプレート関数はint+stringのペアで使用されていますが、多重定義されたバージョンは2回目の呼び出しでbool+stringのペアですでに使用されています。

```
string Assign<int,string>(int&,string)
string Assign(bool&,string)
```

参照

多重定義（オーバーロード）

© 2000-2025, MetaQuotes Ltd.

テンプレートの利点

関数テンプレートは、さまざまなデータ型で配列内の最大要素を検索するなどの同様の操作を実行する必要がある場合に使用されます。テンプレートを適用する主な利点は、型ごとに別々の多重定義（オーバーロード）をコーディングする必要がないことです。複数の多重定義を型ごとに宣言する代わりに、</ t3>

```
double ArrayMax(double array[])
  {
   ...
  }
int ArrayMax(int array[])
  {
   ...
  }
uint ArrayMax(uint array[])
  {
   ...
  }
long ArrayMax(long array[])
  {
   ...
  }
datetime ArrayMax(datetime array[])
  {
   ...
  }
```

テンプレート関数を1つ書くだけです

```
template<typename T>
T ArrayMax(T array[])
  {
   if(ArraySize()==0)
      return(0);
   uint max_index=ArrayMaximum(array);
   return(array[max_index]);
  }
```

コード内で使うには:

```
double high[];
datetime time[];
....
double max_high=ArrayMax(high);
datetime lasttime=ArrayMax(time);
```

ここで、使用されるデータの型を指定するT仮パラメータは、コンパイル時に実際に適用される型に置き換えられます。つまり、コンパイラはdouble、datetimeなどの型ごとに別々の関数を生成します。MQL5では、このアプローチのすべての利点を使ってクラステンプレートを開発することもできます。

© 2000–2025, MetaQuotes Ltd.

クラステンプレート

クラステンプレートはtemplate キーワードの後に typenameキーワードで仮パラメータのリストを列挙する山括弧 <> を使用して宣言されます 。このエントリは、クラスを実装するときに実際の変数型を定義するT仮パラメータを持つジェネリッククラスを扱うようにコンパイラに通知します。例としてT 型の要素を持つ配列を格納するためのベクトルクラスを作成しましょう。</ t10>

```
#define TOSTR(x) #x+" "   // オブジェクト名を表示するマクロ
//+------------------------------------------------------------------+
//| T型要素を格納するためのベクトルクラス                                |
//+------------------------------------------------------------------+
template <typename T>
class TArray
  {
protected:
   T                 m_array[];
public:
   //--- コンストラクタはデフォルトで10要素の配列を作成する
   void TArray(void){ArrayResize(m_array,10);}
   //--- 指定された配列サイズのベクトルを作成するコンストラクタ
   void TArray(int size){ArrayResize(m_array,size);}
   //--- TArray型オブジェクトに格納されているデータの型と量を返す
   string Type(void){return(typename(m_array[0])+":"+(string)ArraySize(m_array));};
  };
```

次に、さまざまな種類の作業を行うプログラムの3つのTArray オブジェクトを作成するさまざまな方法を適用しましょう。

```
void OnStart()
  {
   TArray<double> double_array;    // ベクトルのデフォルトサイズは10
   TArray<int> int_array(15);      // ベクトルのサイズは15
   TArray<string> *string_array;   // TArray<string>ベクトルへのポインタ
//--- 動的オブジェクトを作成する
   string_array=new TArray<string>(20);
//--- 操作ログにオブジェクト名、データ型、ベクトルサイズを表示する
   PrintFormat("%s (%s)",TOSTR(double_array),double_array.Type());
   PrintFormat("%s (%s)",TOSTR(int_array),int_array.Type());
   PrintFormat("%s (%s)",TOSTR(string_array),string_array.Type());
//--- プログラムを完了する前に動的オブジェクトを削除する
   delete(string_array);
  }
```

スクリプト実行の結果：

```
  double_array  (double:10)
  int_array  (int:15)
  string_array  (string:20)
```

ここでは、double、int、およびstringの異なるデータ型を持つ3つのベクトルがあります。

© 2000-2025, MetaQuotes Ltd.

クラステンプレートは、任意の型の他のオブジェクトをカプセル化するために設計されるオブジェクトであるコンテナの開発にも適しています。コンテナオブジェクトは既に特定の種類のオブジェクトを含むコレクションです。通常、保存されたデータの処理は、コンテナに即座に組み込まれます。

例えば、配列外の要素にアクセスすることを許可しないクラステンプレートを作成することによって"out of range"(範囲外)重大エラーを避けることができます。

```
//+------------------------------------------------------------------+
//|  配列要素への自由なアクセスのためのクラス                        |
//+------------------------------------------------------------------+
template<typename T>
class TSafeArray
  {
protected:
   T                 m_array[];
public:
   //--- デフォルトコンストラクタ
   void              TSafeArray(void){}
   //--- 指定されたサイズの配列を作成するコンストラクタ
   void              TSafeArray(int size){ArrayResize(m_array,size);}
   //--- 配列サイズ
   int               Size(void){return(ArraySize(m_array));}
   //--- 配列サイズを変更する
   int               Resize(int size,int reserve)
{return(ArrayResize(m_array,size,reserve));}
   //--- 配列を解放する
   void              Erase(void){ZeroMemory(m_array);}
   //--- インデックスによって配列要素にアクセスする演算子
   T                 operator[](int index);
   //--- 配列からすべての要素を一度に受け取る代入演算子
   void              operator=(const T  &array[]); // T type array
  };
//+------------------------------------------------------------------+
//|  インデックスによる要素の受け取り                                |
//+------------------------------------------------------------------+
template<typename T>
T TSafeArray::operator[](int index)
  {
   static T invalid_value;
//---
   int max=ArraySize(m_array)-1;
   if(index<0 || index>=ArraySize(m_array))
     {
      PrintFormat("%s index %d is not in range (0-%d)!",__FUNCTION__,index,max);
      return(invalid_value);
     }
//---
   return(m_array[index]);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  配列の代入                                                      |
//+------------------------------------------------------------------+
template<typename T>
void TSafeArray::operator=(const T  &array[])
  {
   int size=ArraySize(array);
   ArrayResize(m_array,size);
//--- T型はコピーオペレータをサポートすべき
   for(int i=0;i<size;i++)
      m_array[i]=array[i];
//---
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                              |
//+------------------------------------------------------------------+
void OnStart()
  {
   int copied,size=15;
   MqlRates rates[];
//--- 相場の配列をコピーする
   if((copied=CopyRates(_Symbol,_Period,0,size,rates))!=size)
     {
      PrintFormat("CopyRates(%s,%s,0,%d) returned %d error code",
      _Symbol,EnumToString(_Period),size,GetLastError());
      return;
     }
//--- コンテナを作成し、それにMql Ratesの値配列を挿入する
   TSafeArray<MqlRates> safe_rates;
   safe_rates=rates;
   //--- 配列内のインデックス
   int index=3;
   PrintFormat("Close[%d]=%G",index,safe_rates[index].close);
   //--- 配列外のインデックス
   index=size;
   PrintFormat("Close[%d]=%G",index,safe_rates[index].close);
  }
```

テンプレート宣言は、クラス宣言の外にメソッドを記述する際にも使用する必要がありますのでご注意ください。

```
template<typename T>
T TSafeArray::operator[](int index)
  {
   ...
  }
template<typename T>
void TSafeArray::operator=(const T  &array[])
  {
```

© 2000–2025, MetaQuotes Ltd.

```
   ...
  }
```

クラステンプレートと関数テンプレートを使用すると、カンマで区切られた複数の仮パラメータの定義が可能です。下記の例は"key – value"のペアを格納するMapコレクションです。

```
template<typename Key, template Value>
class TMap
  {
   ...
  }
```

参照

関数テンプレート、多重定義（オーバーロード）

© 2000–2025, MetaQuotes Ltd.

抽象クラスと純粋仮想関数

抽象クラスは、将来におけるより具体的な派生クラスを作成する一般的なエンティティを作成する為に作られています。抽象クラスは、いくつかの他のクラスの為に基底クラスとして使用することができるだけなので、抽象クラスのオブジェクトタイプを作成することはできません。

純粋仮想関数を一つでも含むクラスは、抽象クラスとなります。したがって、抽象クラスから派生したクラスは、その純粋仮想関数を実装する必要があり、そうしないと、これらは同様に抽象クラスになります。

仮想関数は、純粋指定子構文を使用し、『純粋』として宣言されます。例として、共通機能を提供する為だけ型に作成されるCAnimalクラスを見てみましょう。CAnimalのオブジェクト自体は、実際の使用には一般的すぎます。したがって、CAnimalクラスは、抽象クラスの為の良い例となります。

```
class CAnimal
  {
public:
                        CAnimal();      //コンストラクタ
   virtual void         Sound() = 0;    // 純粋仮想関数
private:
   double               m_legs_count;   // 動物の足の数
  };
```

ここでのSound()関数は、純粋仮想関数PURE(=0)の指定子で宣言されている為、純粋仮想となります。

(=NULL)または(=0)と純粋指定子PUREが指定されている仮想関数のみが純粋仮想関数となります。抽象クラスの使用と宣言例：

```
class CAnimal
  {
public:
   virtual void         Sound()=NULL;    // PUREメソッドは派生クラスでオーバーライドする必要があり、CAnimalクラス自体は抽象クラスになり、作成されません
  };
//--- 抽象クラスからの派生
class CCat : public CAnimal
 {
public:
  virtual void          Sound() { Print("Myau"); } // PUREはオーバーライドする必要があり、CCatクラスは抽象クラスではなく、作成することができます
 };

//--- 間違った使用例
new CAnimal;            //『CAnimal』エラーは、コンパイラは「抽象クラスをインスタンス化できません」というエラーを出します
CAnimal some_animal; // 『CAnimal』エラーは、コンパイラは「抽象クラスをインスタンス化できません」というエラーを出します

//--- 正しい使用例
new CCat;   // エラーはなく、CCatクラスは抽象ではありません
CCat cat;   // エラーはなく、CCatクラスは抽象ではありません
```

© 2000-2025, MetaQuotes Ltd.

抽象クラス使用の制限

純粋仮想関数（直接または間接）の抽象クラスのコンストラクタによる呼び出しの際、結果は未定義になります。

```
//+------------------------------------------------------------------+
//|  抽象基底クラス                                                  |
//+------------------------------------------------------------------+
class CAnimal
  {
public:
   //--- 純粋仮想関数
   virtual void       Sound(void)=NULL;
   //--- 関数
   void               CallSound(void) { Sound(); }
   //--- コンストラクタ
   CAnimal()
    {
     //--- 仮想メソッドの明示的な呼び出し
     Sound();
     //--- 明示的ではない呼び出し（第三の関数を介した）
     CallSound();
     //--- コンストラクタまたはデストラクタは常に自分の関数を呼び出す
     //--- 派生クラスでの呼び出された関数によるオーバーライドや仮想性に関わらず
     //--- もし、呼び出される関数が純粋仮想の場合、
     //--- 呼び出しは重大なランタイムエラーをもたらします:『純粋仮想関数呼び出し』
    }
  };
```

しかしながら、抽象クラスのコンストラクタとデストラクタは、他のメンバー関数を呼び出すことができます。

© 2000-2025, MetaQuotes Ltd.

名前空間

名前空間は、変数、関数、クラスなど、さまざまなIDが定義されている特別に宣言された領域で、namespaceキーワードを使用して設定されます。

```
namespace name_of_space {
  // 関数、クラス、変数定義のリスト
}
```

「namespace」を適用すると、グローバル名前空間をサブスペースに分割できます。名前空間内のすべてのIDは、仕様なしで相互に利用可能です。::演算子（コンテキスト解決操作）は、外部から名前空間メンバーにアクセスするために使用されます。

```
namespace ProjectData
{
class DataManager
  {
public:
   void              LoadData() {}
  };
void Func(DataManager& manager) {}
}
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- ProjectData名前空間の操作
   ProjectData::DataManager mgr;
   mgr.LoadData();
   ProjectData::Func(mgr);
  }
```

名前空間は、プログラムで複数のライブラリが使用されるときに発生する可能性のある名前の競合を回避するために、論理グループの形式でコードを配置するために使用されます。そのような場合、各ライブラリをその名前空間で宣言して、各ライブラリの必要な関数とクラスに明示的にアクセスできるようになります。

名前空間は、1つまたは複数のファイルの複数のブロックで宣言できます。コンパイラは、前処理中にすべての部分を結合し、結果の名前空間には、すべての部分で宣言されたすべてのメンバが含まれます。Sample.mqhインクルードファイルに実装されたAクラスがあると仮定します。

```
//+------------------------------------------------------------------+
//|                                                   Sample.mqh |
//+------------------------------------------------------------------+
class A
  {
public:
                     A() {Print(__FUNCTION__);}
  };
```

© 2000-2025, MetaQuotes Ltd.

このクラスをプロジェクトで使用したいのですが、Aクラスはすでに存在します。両方のクラスを使用してIDの競合を回避するには、含まれているファイルを名前空間でラップするだけです。

```
//--- 1番目のAクラスを宣言
class A
  {
public:
                     A() {Print(__FUNCTION__);}
  };
//--- 競合を避けるために、Library名前空間のSample.mqhファイルにAクラスをラップする
namespace Library
{
#include "Sample.mqh"
}
//--- Library名前空間にクラスを追加する
namespace Library
{
class B
  {
public:
                     B() {Print(__FUNCTION__);}
  };
}
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- グローバル名前空間のAクラスを使用
   A a1;
//--- Library名前空間のAクラスを使用<分節
   Library::A a2;
   Library::B b;
  }
//+------------------------------------------------------------------+

/*
結果:
   A::A
   Library::A::A
   Library::B::B
*/
```

名前空間はネストできます。ネストされた名前空間は、その親名前空間のメンバに無制限にアクセスできますが、親名前空間のメンバは、ネストされた名前空間には無制限にアクセスできません。

```
namespace General
{
int Func();
```

© 2000-2025, MetaQuotes Ltd.

```
namespace Details
{
 int Counter;
 int Refresh()  {return Func(); }
}

int GetBars()   {return(iBars(Symbol(), Period()));};
int Size(int i) {return Details::Counter;}
}
```

グローバル名前空間

名前空間で明示的に宣言されていないIDは、グローバル名前空間の暗黙的な部分と見なされます。グローバルIDを明示的に設定するには、名前なしでスコープ解決演算子を使用します。これにより、このIDを、異なる名前空間にある同じ名前の他の要素と区別できます。例として、下記の関数をインポートするとします。

```
#import "lib.dll"
int Func();
#import
//+------------------------------------------------------------------+
//|   関数                                                           |
//+------------------------------------------------------------------+
int Func()
  {
   return(0);
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//+---  インポートされた関数を呼び出す
   Print(lib::Func());
//+---  関数を呼び出す
   Print(::Func());
  }
```

この場合、DLL関数からインポートされたすべての関数は、同じ名前の名前空間に含まれて、コンパイラは呼び出される関数を明確に決定できました。

参照

グローバル変数、ローカル変数、変数のアクセス権 スコープとライフタイム、オブジェクトの作成と解徐

© 2000–2025, MetaQuotes Ltd.

標準的な定数、列挙と構造体

プログラムの書き込みを簡素化し、プログラムテキストをより理解しやすくするために、MQL5 言語は事前に定義された標準的な定数と列挙を提供します。加えて、サービス構造体は情報の格納に使用されます。

標準的な定数はマクロに似ていて int 型です。

定数は、目的によってグループ分けされています。

- チャート定数は、価格チャートの作業（チャートを開く、ナビゲーション、パラメータ設定）に使用されています。
- オブジェクト定数は、チャート内に作成して表示出来るグラフィカルオブジェクトの処理に使用されます。
- 指標定数は、標準及びカスタム指標の操作に使用されます。
- 環境状態定数は、MQL5 プログラムの特性を記述し、クライアント端末、金融商品及び現在の口座に関する情報を表示します。
- 取引定数は、取引の過程で様々な情報を指定することが出来ます。
- 名前付き定数は MQL5 言語の定数です。
- データ構造体は使用されるデータ保存形式を記述します。
- エラーコードと警告コードは、コンパイラメッセージと取引サーバの取引リクエストへの回答を説明します。
- IN / OUTの定数はファイル関数操作、またMessageBox() 関数で画面にメッセージを表示するのに使用されます。

© 2000–2025, MetaQuotes Ltd.

チャート定数

チャートのさまざまなプロパティを記述する定数は、次のグループに分類されます。

- イベントの種類 – チャート操作時に起こるイベント
- チャート時間軸 – 内蔵された標準期間
- チャートプロパティ - チャート関数のパラメータとしてつかわれる識別子
- 位置定数 - ChartNavigate() 関数のパラメータ値
- チャート表示 – チャート外観の設定

© 2000–2025, MetaQuotes Ltd.

チャートイベントの種類

事前定義された OnChartEvent() 関数を使用して処理出来るイベントには 11 種類あります。カスタムイベントには CHARTEVENT_CUSTOM 以上 CHARTEVENT_CUSTOM_LAST 以下の 65,535 の識別子が提供されています。カスタムイベントの作成には EventChartCustom() 関数が使用されます。

ENUM_CHART_EVENT

ID	説明
CHARTEVENT_KEYDOWN	キーストローク。
CHARTEVENT_MOUSE_MOVE	マウスの移動、マウスクリック（チャートで CHART_EVENT_MOUSE_MOVE=true の場合）。
CHARTEVENT_MOUSE_WHEEL	マウスホイールの押下やスクロール（チャートで CHART_EVENT_MOUSE_WHEEL=True）。
CHARTEVENT_OBJECT_CREATE	作成されたグラフィックオブジェクト（チャートで CHART_EVENT_OBJECT_CREATE=true の場合）。
CHARTEVENT_OBJECT_CHANGE	プロパティダイアログで変更されたグラフィックオブジェクトのプロパティ。
CHARTEVENT_OBJECT_DELETE	削除されたグラフィックオブジェクト（チャートで CHART_EVENT_OBJECT_DELETE=true の場合）。
CHARTEVENT_CLICK	チャートのクリック。
CHARTEVENT_OBJECT_CLICK	グラフィックオブジェクトのクリック。
CHARTEVENT_OBJECT_DRAG	グラフィックオブジェクトのドラッグアンドドロップ。
CHARTEVENT_OBJECT_ENDEDIT	Edit グラフィックオブジェクトのテキスト編集の終了。
CHARTEVENT_CHART_CHANGE	プロパティダイアログでのチャートサイズやチャートプロパティの変更。
CHARTEVENT_CUSTOM	カスタムイベントの範囲からのイベントの初期番号。
CHARTEVENT_CUSTOM_LAST	カスタムイベントの範囲からのイベントの最終番号。

OnChartEvent() 関数の入力パラメータはイベントの種類ごとにイベント処理に必要とされる一定の値を有しています。パラメータで受け渡しされるイベントと値は以下の表に記載されています。

イベント	id パラメータ値	lparam パラメータ値	dparam パラメータ値	sparam パラメータ値
キーストロークイベント	CHARTEVENT_KEYDOWN	押されたキーのコード	繰り返し回数（ユーザのキーの長押しがキーストロークの繰り返しとみな	キーボードボタンの状態を記述するビットマスクの文字列値

© 2000–2025, MetaQuotes Ltd.

イベント	id パラメータ値	lparam パラメータ値	dparam パラメータ値	sparam パラメータ値
			された場合の繰り返し回数）	
マウスイベント（チャートで CHART_EVENT_MOUSE_MOVE=true の場合）	CHARTEVENT_MOUSE_MOVE	X 座標	Y 座標	マウスボタンの状態を記述するビットマスクの文字列の値
マウスホイールイベント（チャートで CHART_EVENT_MOUSE_WHEEL=trueの場合）	CHARTEVENT_MOUSE_WHEEL	キーとマウスボタン、マウスポインタのX座標とY座標の状態のフラグ。下の例の説明を参照してください。	マウスホイールスクロールのデルタ値。	—
グラフィックオブジェクトの作成イベント（チャートで CHART_EVENT_OBJECT_CREATE=true の場合）	CHARTEVENT_OBJECT_CREATE	—	—	作成したグラフィックオブジェクトの名称
プロパティダイアログでのオブジェクトプロパティ変更イベント	CHARTEVENT_OBJECT_CHANGE	—	—	変更したグラフィックオブジェクトの名称
グラフィックオブジェクト削除イベント（チャートで CHART_EVENT_OBJECT_DELETE=true の場合）	CHARTEVENT_OBJECT_DELETE	—	—	削除したグラフィックオブジェクトの名称
チャート上でのマウスクリックイベント	CHARTEVENT_CLICK	X 座標	Y 座標	—
チャートに属するグラフィックオブジェクトでのマウスクリックイベント	CHARTEVENT_OBJECT_CLICK	X 座標	Y 座標	イベントが発生したグラフィック·オブジェクトの名称
マウスを使用してグラフィックオブジェクトのドラッグイベント	CHARTEVENT_OBJECT_DRAG	—	—	移動したグラフィックオブジェクトの名称

© 2000–2025, MetaQuotes Ltd.

イベント	id パラメータ値	lparam パラメータ値	dparam パラメータ値	sparam パラメータ値
LabelEdit グラフィックオブジェクトの入力ボックスでのテキスト編集完成のイベント	CHARTEVENT_OBJECT_ENDEDIT	—	—	テキスト編集が完成した LabelEdit グラフィックオブジェクトの名称
チャートサイズの変更イベントやプロパティダイアログを通したチャートのプロパティの変更	CHARTEVENT_CHART_CHANGE	—	—	—
N 番号を使用したユーザイベント ID	CHARTEVENT_CUSTOM+N	EventChartCustom() 関数で設定された値	EventChartCustom() 関数で設定された値	EventChartCustom() 関数で設定された値

例:

```
#define KEY_NUMPAD_5        12
#define KEY_LEFT            37
#define KEY_UP              38
#define KEY_RIGHT           39
#define KEY_DOWN            40
#define KEY_NUMLOCK_DOWN    98
#define KEY_NUMLOCK_LEFT   100
#define KEY_NUMLOCK_5      101
#define KEY_NUMLOCK_RIGHT  102
#define KEY_NUMLOCK_UP     104
//+------------------------------------------------------------------+
//|  エキスパート初期化に使用される関数                                |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   Print("The expert with name ",MQL5InfoString(MQL5_PROGRAM_NAME)," is running");
//--- オブジェクト作成イベントを有効にする
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_CREATE,true);
//--- オブジェクト削除イベントを有効にする
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_DELETE,true);
//--- チャートプロパティを強制的に更新して、イベント処理の準備を整える
   ChartRedraw();
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//|  ChartEvent 関数                                                 |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // イベント識別子
                  const long& lparam,   // long 型のイベントパラメータ
                  const double& dparam, // double 型のイベントパラメータ
                  const string& sparam  // string 型のイベントパラメータ
                 )
  {
//--- チャート上でマウスの左ボタンが押された
   if(id==CHARTEVENT_CLICK)
     {
      Print("The coordinates of the mouse click on the chart are: x = ",lparam,"  y
= ",dparam);
     }
//--- マウスがグラフィックオブジェクトでクリックされた
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      Print("The mouse has been clicked on the object with name '"+sparam+"'");
     }
//---キーが押された
   if(id==CHARTEVENT_KEYDOWN)
     {
      switch(lparam)
        {
         case KEY_NUMLOCK_LEFT:  Print("The KEY_NUMLOCK_LEFT has been pressed");
break;
         case KEY_LEFT:          Print("The KEY_LEFT has been pressed");
break;
         case KEY_NUMLOCK_UP:    Print("The KEY_NUMLOCK_UP has been pressed");
break;
         case KEY_UP:            Print("The KEY_UP has been pressed");
break;
         case KEY_NUMLOCK_RIGHT: Print("The KEY_NUMLOCK_RIGHT has been pressed");
break;
         case KEY_RIGHT:         Print("The KEY_RIGHT has been pressed");
break;
         case KEY_NUMLOCK_DOWN:  Print("The KEY_NUMLOCK_DOWN has been pressed");
break;
         case KEY_DOWN:          Print("The KEY_DOWN has been pressed");
break;
         case KEY_NUMPAD_5:      Print("The KEY_NUMPAD_5 has been pressed");
break;
         case KEY_NUMLOCK_5:     Print("The KEY_NUMLOCK_5 has been pressed");
break;
         default:                Print("Some not listed key has been pressed");
        }
      ChartRedraw();
     }
//--- オブジェクトが削除された
   if(id==CHARTEVENT_OBJECT_DELETE)
     {
      Print("The object with name ",sparam," has been deleted");
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- オブジェクトが作成された
   if(id==CHARTEVENT_OBJECT_CREATE)
      {
       Print("The object with name ",sparam," has been created");
      }
//--- オブジェクトが移動されたか、アンカーポイントの座標が変更された
   if(id==CHARTEVENT_OBJECT_DRAG)
      {
       Print("The anchor point coordinates of the object with name ",sparam," has
been changed");
      }
//--- 編集オブジェクトのテキストが変更した
   if(id==CHARTEVENT_OBJECT_ENDEDIT)
      {
       Print("The text in the Edit field of the object with name ",sparam," has been
changed");
      }
  }
```

CHARTEVENT_MOUSE_MOVE イベントでは、キーボードとマウスボタンの状態に関する情報が sparam 文字列パラメータに含まれています。

ビット	説明
1	マウスの左ボタンの状態
2	マウスの右ボタンの状態
3	SHIFT ボタンの状態
4	CTRL ボタンの状態
5	マウスの中央ボタンの状態
6	マウスの追加ボタン 1 の状態
7	マウスの追加ボタン 2 の状態

例：

```
//+------------------------------------------------------------------+
//|  エキスパート初期化に使用される関数                                |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- CHART_EVENT_MOUSE_MOVE  メッセージを有効にする
   ChartSetInteger(0,CHART_EVENT_MOUSE_MOVE,1);
//--- チャートのコンテキストメニューを無効にします（右側）
   ChartSetInteger(0,CHART_CONTEXT_MENU,0);
//--- 十字線を無効にします（中央のボタンで）
   ChartSetInteger(0,CHART_CROSSHAIR_TOOL,0);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- チャートプロパティを強制的に更新して、イベント処理の準備を整える
   ChartRedraw();
  }
//+------------------------------------------------------------------+
//| マウスの状態                                                     |
//+------------------------------------------------------------------+
string MouseState(uint state)
  {
   string res;
   res+="\nML: "   +(((state& 1)== 1)?"DN":"UP");   // 左ボタン
   res+="\nMR: "   +(((state& 2)== 2)?"DN":"UP");   // 右ボタン
   res+="\nMM: "   +(((state&16)==16)?"DN":"UP");   // 中央ボタン
   res+="\nMX: "   +(((state&32)==32)?"DN":"UP");   // 追加ボタン1
   res+="\nMY: "   +(((state&64)==64)?"DN":"UP");   // 追加ボタン 2
   res+="\nSHIFT: "+(((state& 4)== 4)?"DN":"UP");   // shift キー
   res+="\nCTRL: " +(((state& 8)== 8)?"DN":"UP");   // control キー
   return(res);
  }
//+------------------------------------------------------------------+
//| ChartEvent 関数                                                  |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,const long &lparam,const double &dparam,const string
&sparam)
  {
   if(id==CHARTEVENT_MOUSE_MOVE)
      Comment("POINT: ",(int)lparam,",",(int)dparam,"\n",MouseState((uint)sparam));
  }
```

CHARTEVENT_MOUSE_WHEEL イベントの場合、lparamパラメータとdparam パラメータは Ctrlキー及びShiftキー、マウスボタン、カーソル座標、およびマウスホイールのスクロール値の状態に関する情報を含みます。より良い理解のためには、このエキスパートアドバイザーをチャート上で実行し、マウスホイールをスクロールしながら、異なるボタンを押しながらコードに記載されているキーを押し続けます。

下記はCHARTEVENT_MOUSE_WHEELイベント処理の例です。

```
//+------------------------------------------------------------------+
//| エキスパート初期化関数                                           |
//+------------------------------------------------------------------+
init OnInit()
  {
//--- マウスホイールスクロールメッセージの有効化
   ChartSetInteger(0,CHART_EVENT_MOUSE_WHEEL,1);
//--- チャートプロパティを強制的に更新して、イベント処理の準備を整える
   ChartRedraw();
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| ChartEvent関数                                                   |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,const long &lparam,const double &dparam,const string &sparam)
  {
   if(id==CHARTEVENT_MOUSE_WHEEL)
     {
      //--- このイベントのマウスボタンとホイールの状態を考慮する
      int flg_keys = (int)(lparam>>32);          // Ctrl キーとShift キー、およびマウスボタンの状態のフラグ
      int x_cursor = (int)(short)lparam;         // マウスホイールイベントが発生したX座標
      int y_cursor = (int)(short)(lparam>>16);   // マウスホイールイベントが発生したY座標
      int delta    = (int)dparam;                // +120または-120に達したときにトリガーされるマウススクロールの合計値
      //--- フラグの処理
      string str_keys="";
      if((flg_keys&0x0001)!=0) str_keys+="LMOUSE ";
      if((flg_keys&0x0002)!=0) str_keys+="RMOUSE ";
      if((flg_keys&0x0004)!=0) str_keys+="SHIFT ";
      if((flg_keys&0x0008)!=0) str_keys+="CTRL ";
      if((flg_keys&0x0010)!=0) str_keys+="MMOUSE ";
      if((flg_keys&0x0020)!=0) str_keys+="X1MOUSE ";
      if((flg_keys&0x0040)!=0) str_keys+="X2MOUSE ";

      if(str_keys!="")
         str_keys=", keys='"+StringSubstr(str_keys,0,StringLen(str_keys)-1) + "'";
      PrintFormat("%s: X=%d, Y=%d, delta=%d%s",EnumToString(CHARTEVENT_MOUSE_WHEEL),x_cursor,y_cursor,delta,str_keys);
     }
  }
//+------------------------------------------------------------------+ /*
   出力例
   CHARTEVENT_MOUSE_WHEEL: Ctrl pressed: X=193, Y=445, delta=-120
   CHARTEVENT_MOUSE_WHEEL: Shift pressed: X=186, Y=446, delta=120
   CHARTEVENT_MOUSE_WHEEL:  X=178, Y=447, delta=-120
   CHARTEVENT_MOUSE_WHEEL:  X=231, Y=449, delta=120
   CHARTEVENT_MOUSE_WHEEL: MiddleButton pressed: X=231, Y=449, delta=120
   CHARTEVENT_MOUSE_WHEEL: LeftButton pressed: X=279, Y=320, delta=-120
   CHARTEVENT_MOUSE_WHEEL: RightButton pressed: X=253, Y=330, delta=120  */
```

S参照

イベント処理の関数、イベント操作

© 2000-2025, MetaQuotes Ltd.

チャート時間軸

事前定義されたチャート時間軸の全ては、一意の識別子を持っています。PERIOD_CURRENT 識別子は、MQL5 プログラムが実行されているチャートの現在の期間を意味します。

ENUM_TIMEFRAMES

ID	説明
PERIOD_CURRENT	現在の時間軸
PERIOD_M1	1分
PERIOD_M2	2分
PERIOD_M3	3分
PERIOD_M4	4分
PERIOD_M5	5分
PERIOD_M6	6分
PERIOD_M10	10分
PERIOD_M12	12分
PERIOD_M15	15分
PERIOD_M20	20分
PERIOD_M30	30分
PERIOD_H1	1時間
PERIOD_H2	2時間
PERIOD_H3	3時間
PERIOD_H4	4時間
PERIOD_H6	6時間
PERIOD_H8	8時間
PERIOD_H12	12時間
PERIOD_D1	1日
PERIOD_W1	1週間
PERIOD_MN1	1ヶ月

例：

```
   string chart_name="test_Object_Chart";
   Print("Let's try to create a Chart object with the name ",chart_name);
//--- オブジェクトが存在しない場合は作成する
   if(ObjectFind(0,chart_name)<0)ObjectCreate(0,chart_name,OBJ_CHART,0,0,0,0,0);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- シンボルの定義
   ObjectSetString(0,chart_name,OBJPROP_SYMBOL,"EURUSD");
//--- アンカーポイントの X 座標を設定
   ObjectSetInteger(0,chart_name,OBJPROP_XDISTANCE,100);
//--- アンカーポイントの Y 座標を設定
   ObjectSetInteger(0,chart_name,OBJPROP_YDISTANCE,100);
//--- チャート幅を設定
   ObjectSetInteger(0,chart_name,OBJPROP_XSIZE,400);
//--- チャートの高さを設定
   ObjectSetInteger(0,chart_name,OBJPROP_YSIZE,300);
//--- 時間軸を設定
   ObjectSetInteger(0,chart_name,OBJPROP_PERIOD,PERIOD_D1);
//--- スケールを設定( 0 ～ 5 )
   ObjectSetDouble(0,chart_name,OBJPROP_SCALE,4);
//--- マウスでの選択を無効化する
   ObjectSetInteger(0,chart_name,OBJPROP_SELECTABLE,false);
```

時系列識別子

時系列識別子はiHighest()及びiLowest()関数で使用されます。これらは

ENUM_SERIESMODE列挙体の値と等しくなれます。

識別子	説明
MODE_OPEN	始値
MODE_LOW	安値
MODE_HIGH	高値
MODE_CLOSE	終値
MODE_VOLUME	ティックボリューム
MODE_REAL_VOLUME	実のボリューム
MODE_SPREAD	スプレッド

参照

PeriodSeconds、Period、日付と時刻、オブジェクトの可視性

© 2000-2025, MetaQuotes Ltd.

チャートプロパティ

ENUM_CHART_PROPERTY　列挙の識別子はチャート操作の関数のパラメータとして使用されます。「プロパティ型」欄にある r/o は read-only の略で、プロパティが読み込み専用で変更不可能であることを意味します。「プロパティ型」欄にある w/o は write-only の略で、プロパティが書き込み専用で受信不可能であることを意味します。特定のプロパティにアクセスする際には、チャートサブウィンドウの番号を示す追加パラメータ修飾子（修飾子）を指定する必要があります。0 はメインチャートウィンドウを意味します。

チャートのプロパティを定義する関数は、実際にチャートに変更コマンドを送信するのに使用されます。これらの関数が正常に実行された場合、コマンドがチャートイベントの共用キューに追加されます。チャートの変更は、チャートイベントのキューの処理時に実装されます。

従って、これらの関数の呼び出し後の、チャートの即時の視覚更新は期待出来ません。一般的には、チャートは、新たな相場の到着やチャートウィンドウのサイズ変更などの変更イベントの後に、端末によって自動更新されます。強制的にチャートを更新するには ChartRedraw() 関数を使用します。

ChartSetInteger() 及び ChartGetInteger() 関数

ENUM_CHART_PROPERTY_INTEGER

ID	説明	プロパティ型
CHART_SHOW	価格チャートを描画します。falseの場合、時間および価格スケール、クイックナビゲーションバー、カレンダーイベントラベル、取引ラベル、インディケータおよびバーツールヒント、インディケータサブウィンドウ、ボリュームヒストグラムなどの価格チャートの属性は無効にされ、すべてのチャート境界のインデントが削除されます。 図面を無効にすることは、グラフィカルリソースを使用してカスタムプログラムインターフェイスを作成するための完璧なソリューションです。 グラフィックオブジェクトはCHART_SHOWプロパティ値にかかわらず、常に描画されます。	bool
CHART_IS_OBJECT	チャートオブジェクト（OBJ_CHART）を識別し、グラフィックオブジェクトにはtrue を返します。チャートの場合falseを返します。	bool r/o
CHART_BRING_TO_TOP	他のチャートの上に重ねてチャートを表示します。	bool
CHART_CONTEXT_MENU	右クリックでコンテキストメニューへのアクセスを有効/無効にします。 CHART_CONTEXT_MENU=falseの場合、チャートのコンテキストメニューのみが無効になります。チャート上のオブジェクトのコンテキストメニューは利用可能なままです。	bool（デフォルトはtrue）
CHART_CROSSHAIR_TOOL	中クリックで Crosshairツールへのアクセスを有効/無効にします。	bool（デフォルトはtrue）

© 2000-2025, MetaQuotes Ltd.

ID	説明	プロパティ型
CHART_MOUSE_SCROLL	マウスの左ボタンを使用してチャートを水平方向にスクロールします。CHART_SCALEFIX、CHART_SCALEFIX_11 または CHART_SCALE_PT_PER_BAR プロパティーの値がtrue の場合には、垂直方向のスクロールも可能です。CHART_MOUSE_SCROLL=falseの場合、マウスホイールを使用したチャートのスクロールは利用できません。	bool
CHART_EVENT_MOUSE_WHEEL	チャート上のすべてのmql5プログラムにマウスホイールイベント (CHARTEVENT_MOUSE_WHEEL) に関するメッセージを送信します。	bool（デフォルトはtrue）
CHART_EVENT_MOUSE_MOVE	チャート上の全ての MQL5 プログラムに、マウス移動イベントとマウスクリックイベントの通知を送信します（CHARTEVENT_MOUSE_MOVE）。	bool
CHART_EVENT_OBJECT_CREATE	チャート上の全てのMQL5 プログラムに、新規オブジェクト作成イベントの通知を送信します（CHARTEVENT_OBJECT_CREATE）。	bool
CHART_EVENT_OBJECT_DELETE	チャート上の全てのMQL5 プログラムに、削除イベントの通知を送信します（CHARTEVENT_OBJECT_DELETE）。	bool
CHART_MODE	チャートの種類（ローソク足、バー、またはライン）。	enum ENUM_CHART_MODE
CHART_FOREGROUND	前景の価格チャート。	bool
CHART_SHIFT	右側の境界線からの価格チャートのインデントのモード。	bool
CHART_AUTOSCROLL	チャートの右側の境界線への自動移動のモード。	bool
CHART_KEYBOARD_CONTROL	キーボード（"Home"、"End"、"PgUp"、"+"、"-"、"上矢印" など）を使用してチャートを管理できます。CHART_KEYBOARD_CONTROLをfalseに設定すると、チャートのスクロールとスケーリングが無効になりますが、OnChartEvent()でキー押下イベントをを受け取ることはできます。	bool
CHART_QUICK_NAVIGATION	チャートがスペースとEnterキーの入力を傍受するのを許可してクイックナビゲーションバーをアクティブにします。クイックナビゲーションバーはマウスのダブルクリックまたはスペース/Enterが押された後、チャートの下部に自動	bool

© 2000-2025, MetaQuotes Ltd.

ID	説明	プロパティ型
	的に表示されます。これによって銘柄、時間枠と最初に表示されるバーの日付を迅速に変更することができます。	
CHART_SCALE	Scale	int 0 ～ 5
CHART_SCALEFIX	固定スケールモード。	bool
CHART_SCALEFIX_11	1:1 スケールモード。	bool
CHART_SCALE_PT_PER_BAR	指定されたスケール（バーあたりのポイント単位）。	bool
CHART_SHOW_TICKER	左上隅に銘柄のティッカーを表示します。CHART_SHOW_TICKERを「false」に設定するとCHART_SHOW_OHLCも「false」になり、OHLCが無効になります。	bool
CHART_SHOW_OHLC	左上隅にOHLC値を表示します。CHART_SHOW_OHLCを「true」に設定するとCHART_SHOW_TICKERも「true」になり、ティッカーが有効になります。	bool
CHART_SHOW_BID_LINE	買気配値をチャートに水平線として表示。	bool
CHART_SHOW_ASK_LINE	売気配値をチャートに水平線として表示。	bool
CHART_SHOW_LAST_LINE	直近価格をチャートに水平線として表示。	bool
CHART_SHOW_PERIOD_SEP	隣接する期間の間に垂直方向のセパレータを表示。	bool
CHART_SHOW_GRID	チャートにグリッドを表示。	bool
CHART_SHOW_VOLUMES	チャートにボリュームを表示。	enum ENUM_CHART_VOLUME_MODE
CHART_SHOW_OBJECT_DESCR	オブジェクトの説明テキストの表示（全てのオブジェクトの説明が表示されるわけではありません）。	bool
CHART_VISIBLE_BARS	チャートに表示出来るバーの数。	int r/o
CHART_WINDOWS_TOTAL	指標サブウィンドウを含んだチャートウィンドウの総数。	int r/o
CHART_WINDOW_IS_VISIBLE	サブウィンドウの可視性。	bool r/o 修飾子 – サブウィンドウ番号。
CHART_WINDOW_HANDLE	チャートウィンドウハンドル（HWND）。	int r/o

© 2000–2025, MetaQuotes Ltd.

ID	説明	プロパティ型
CHART_WINDOW_YDISTANCE	指標サブウィンドウの上部フレームとメインチャートウィンドウの上部フレームとの縦 Y 軸に沿った距離（ピクセル単位）。指標サブウィンドウでのグラフィカルオブジェクトの座標がサブウィンドウの左上隅を基準に設定されているのに比べ、マウスイベントでのカーソル座標は、メインチャートウィンドウの座標で渡されます。 この値は、図形オブジェクトを正しく取り扱うために、メインチャートの絶対座標をサブウィンドウフレームの左上隅を基準に設定されているサブウィンドウのローカル座標に変換するために必要とされます。	int r/o 修飾子 - サブウィンドウ番号。
CHART_FIRST_VISIBLE_BAR	チャート上で最初に見えるバーの番号。バーのインデックスは時系列式に付けられます。	int r/o
CHART_WIDTH_IN_BARS	チャートの幅（バー単位）。	int r/o
CHART_WIDTH_IN_PIXELS	チャートの幅（ピクセル単位）。	int r/o
CHART_HEIGHT_IN_PIXELS	チャートの高さ(ピクセル単位)。	int 修飾子 = サブウィンドウ番号
CHART_COLOR_BACKGROUND	チャートの背景色。	color
CHART_COLOR_FOREGROUND	軸、スケール及びOHLCラインの色。	color
CHART_COLOR_GRID	グリッドの色。	color
CHART_COLOR_VOLUME	ボリュームやポジションオープンレベルの色。	color
CHART_COLOR_CHART_UP	強気ローソク足のアップバー、影とボーダーの色。	color
CHART_COLOR_CHART_DOWN	弱気ローソク足のダウンバー、影とボーダーの色。	color
CHART_COLOR_CHART_LINE	折れ線グラフの色と「同時線」日本製ローソク足の色。	color
CHART_COLOR_CANDLE_BULL	強気ローソク足の実体の色。	color
CHART_COLOR_CANDLE_BEAR	弱気ローソク足の実体の色。	color
CHART_COLOR_BID	売値レベルの色。	color

© 2000-2025, MetaQuotes Ltd.

ID	説明	プロパティ型
CHART_COLOR_ASK	買値レベルの色。	color
CHART_COLOR_LAST	直近の約定価格（Last）の線の色。	color
CHART_COLOR_STOP_LEVEL	逆指注文値（決済逆指値及び決済指値）レベルの色。	color
CHART_SHOW_TRADE_LEVELS	チャートでの取引レベル（ポジション、決済逆指値、決済指値及び未決注文のレベル）を表示。	bool
CHART_DRAG_TRADE_LEVELS	チャートで取引レベルをマウスでドラッグする許可。ドラッグモードの初期値は有効（true）です。	bool
CHART_SHOW_DATE_SCALE	チャートでタイムスケールを表示。	bool
CHART_SHOW_PRICE_SCALE	チャートに価格スケールを表示。	bool
CHART_SHOW_ONE_CLICK	チャートに「ワンクリック取引」パネルを表示。	bool
CHART_IS_MAXIMIZED	チャートウィンドウが最大化されている。	bool r/o
CHART_IS_MINIMIZED	チャートウィンドウが最小化されている。	bool r/o
CHART_IS_DOCKED	チャートウィンドウがドックされている。falseに設定されると、チャートは端末領域の外側にドラッグされて移動されることができます。	bool
CHART_FLOAT_LEFT	仮想画面に相対したドッキング解除されたチャートウィンドウの左座標	int
CHART_FLOAT_TOP	仮想画面に相対したドッキング解除されたチャートウィンドウの上部座標	int
CHART_FLOAT_RIGHT	仮想画面に相対したドッキング解除されたチャートウィンドウの右座標	int
CHART_FLOAT_BOTTOM	仮想画面に相対したドッキング解除されたチャートウィンドウの下部座標	int

ChartSetDouble() 及び ChartGetDouble()関数

ENUM_CHART_PROPERTY_DOUBLE

ID	説明	プロパティ型
CHART_SHIFT_SIZE	百分率で表された右境界線からのゼロバーインデントのサイズ。	double（10～50 パーセント）

© 2000–2025, MetaQuotes Ltd.

ID	説明	プロパティ型
CHART_FIXED_POSITION	パーセント値で表された、左の境界からのチャートの固定位置。チャートの固定位置は、水平時間軸上の小さな灰色の三角形でマークされています。これはティック着信時のチャート右への自動スクロールが無効になっている場合にのみ表示されます（CHART_AUTOSCROLL プロパティを参照）。固定位置にあるバーはズームインやズームアウトする時にも移動しません。	double
CHART_FIXED_MAX	固定されたチャートの最大値。	double
CHART_FIXED_MIN	固定されたチャートの最小値。	double
CHART_POINTS_PER_BAR	スケール（バーあたりのポイント単位）。	double
CHART_PRICE_MIN	チャートの最小値。	double r/o 修飾子 – サブウィンドウ番号。
CHART_PRICE_MAX	チャートの最大値。	double r/o 修飾子 – サブウィンドウ番号。

ChartSetString() 及び ChartGetString()関数

ENUM_CHART_PROPERTY_STRING

ID	説明	プロパティ型
CHART_COMMENT	チャート内のコメントテキスト。	string
CHART_EXPERT_NAME	指定されたchart_idを持つチャート上に起動	string r/o

© 2000–2025, MetaQuotes Ltd.

ID	説明	プロパティ型
	されたエキスパートアドバイザの名前	
CHART_SCRIPT_NAM	指定されたchart_idを持つチャート上に起動されたスクリプトの名前	string r/o

例:

```
int chartMode=ChartGetInteger(0,CHART_MODE);
switch(chartMode)
  {
   case(CHART_BARS):    Print("CHART_BARS");   break;
   case(CHART_CANDLES): Print("CHART_CANDLES");break;
   default:Print("CHART_LINE");
  }
bool shifted=ChartGetInteger(0,CHART_SHIFT);
if(shifted) Print("CHART_SHIFT =true");
else Print("CHART_SHIFT =false");
bool autoscroll=ChartGetInteger(0,CHART_AUTOSCROLL);
if(autoscroll) Print("CHART_AUTOSCROLL =true");
else Print("CHART_AUTOSCROLL =false");
int chartHandle=ChartGetInteger(0,CHART_WINDOW_HANDLE);
Print("CHART_WINDOW_HANDLE = ",chartHandle);
int windows=ChartGetInteger(0,CHART_WINDOWS_TOTAL);
Print("CHART_WINDOWS_TOTAL = ",windows);
if(windows>1)
  {
   for(int i=0;i<windows;i++)
     {
      int height=ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,i);
      double priceMin=ChartGetDouble(0,CHART_PRICE_MIN,i);
      double priceMax=ChartGetDouble(0,CHART_PRICE_MAX,i);
      Print(i+": CHART_HEIGHT_IN_PIXELS = ",height," pixels");
      Print(i+": CHART_PRICE_MIN = ",priceMin);
      Print(i+": CHART_PRICE_MAX = ",priceMax);
     }
  }
```

参照

チャート操作の例

© 2000-2025, MetaQuotes Ltd.

位置定数

ENUM_CHART_POSITION リストの 3 つの識別子は ChartNavigate() 関数の *position* パラメータとして使用可能です。

ENUM_CHART_POSITION

ID	説明
CHART_BEGIN	チャート先頭（最古の価格）。
CHART_CURRENT_POS	現在の位置。
CHART_END	チャートの最後（最新の価格）。

例：

```
long handle=ChartOpen("EURUSD",PERIOD_H12);
if(handle!=0)
  {
   ChartSetInteger(handle,CHART_AUTOSCROLL,false);
   ChartSetInteger(handle,CHART_SHIFT,true);
   ChartSetInteger(handle,CHART_MODE,CHART_LINE);
   ResetLastError();
   bool res=ChartNavigate(handle,CHART_END,150);
   if(!res) Print("Navigate failed. Error = ",GetLastError());
   ChartRedraw();
  }
```

© 2000–2025, MetaQuotes Ltd.

チャート表示

価格チャートには表示方法が 3 つあります。

- バー
- ローソク足
- ライン

価格チャートを表示する特定の方法は ChartSetInteger(chart_handle,CHART_MODE, chart_mode) 関数で設定されます。ここで chart_mode は ENUM_CHART_MODE 列挙の値のいずれかです。

ENUM_CHART_MODE

ID	説明
CHART_BARS	バーの列として表示。
CHART_CANDLES	日本製ローソク足として表示。
CHART_LINE	終値に引かれた線として表示。

価格チャートにボリュームを表示する方法の指定には ChartSetInteger(chart_handle, CHART_SHOW_VOLUMES, volume_mode) 関数が使用されます。ここで volume_mode は ENUM_CHART_VOLUME_MODE 列挙の値のいずれかです。

ENUM_CHART_VOLUME_MODE

ID	説明
CHART_VOLUME_HIDE	ボリュームの非表示。
CHART_VOLUME_TICK	ティックボリューム。
CHART_VOLUME_REAL	取引高。

例:

```
//--- 現在のチャートのハンドルを取得する
   long handle=ChartID();
   if(handle>0) // 成功したら、更にカスタマイズする
     {
      //--- 自動スクロールを無効にする
      ChartSetInteger(handle,CHART_AUTOSCROLL,false);
      //--- チャートの右側の境界線のインデントを設定する
      ChartSetInteger(handle,CHART_SHIFT,true);
      //--- ローソク足として表示する
      ChartSetInteger(handle,CHART_MODE,CHART_CANDLES);
      //--- 履歴開始から 100 足分スクロールする
      ChartNavigate(handle,CHART_CURRENT_POS,100);
      //--- ティックボリュームの表示モードを設定する
      ChartSetInteger(handle,CHART_SHOW_VOLUMES,CHART_VOLUME_TICK);
     }
```

© 2000-2025, MetaQuotes Ltd.

参照

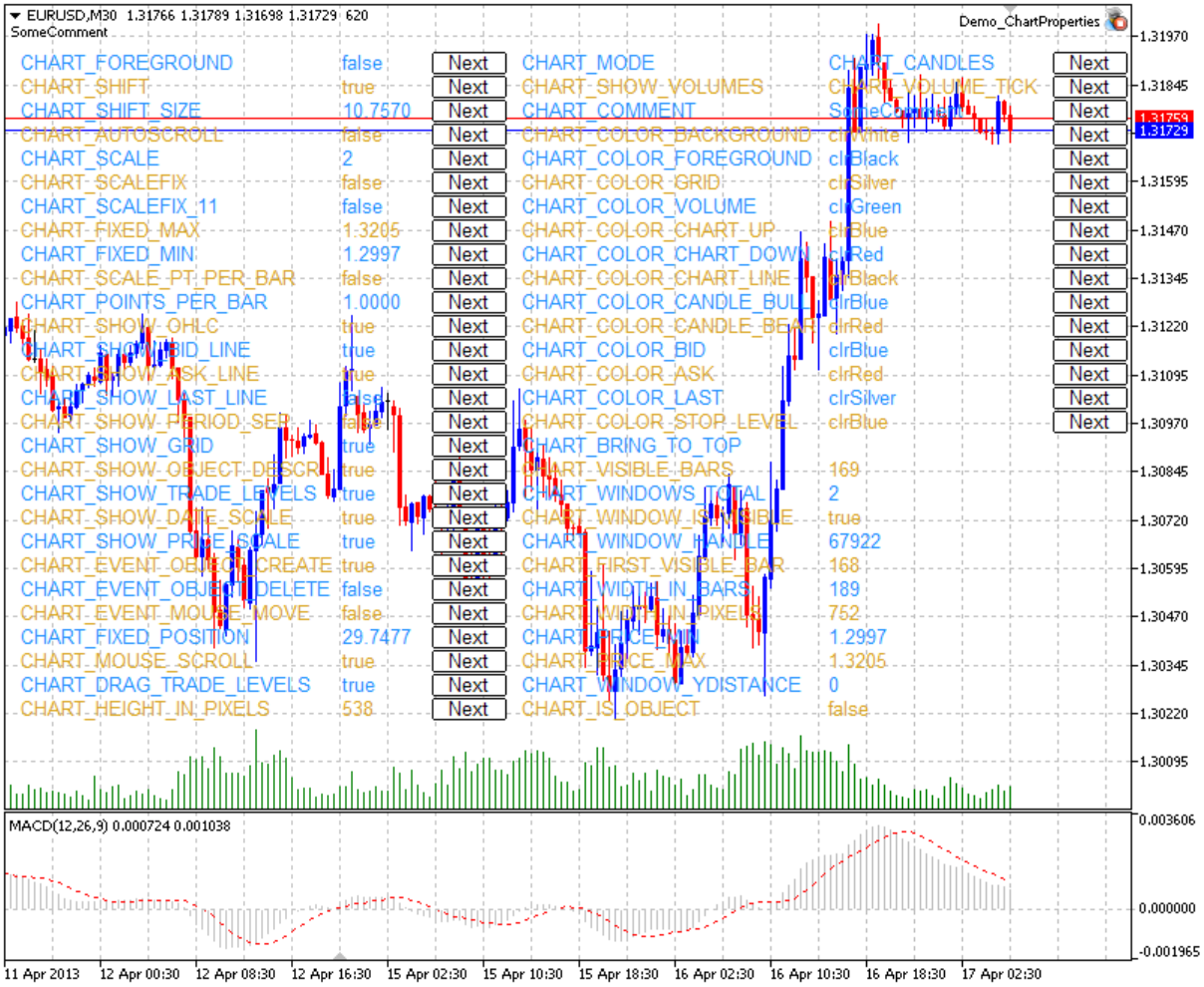
ChartOpen、ChartID

© 2000–2025, MetaQuotes Ltd.

チャート操作の例

このセクションはチャートプロパティを操作する例を含みます。各プロパティに対して、1 つまたは 2 つの完全な関数が表示されます。これらの関数は、プロパティ値の設定/取得を可能にします。これらの関数はカスタム MQL5 アプリケーションでそのまま使用することが出来ます。

下のスクリーンショットは、チャートプロパティの変更とともにグラフィックパネルの外観がどのように変化するかを示しています。「 Next 」ボタンをクリックすると、該当するプロパティの新しい値が設定され、チャートウィンドウで変更内容を表示出来ます。



パネルのソースコードは下にあります。

チャートプロパティ及びその操作に使用されるサンプル関数

- **CHART_IS_OBJECT** はオブジェクトが実際のチャートかグラフィックオブジェクトかを定義します。

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| オブジェクトがチャートかどうかをチェックする。グラフィックオブジェクトの場合,  |
//| 結果は true で、チャートの場合                                    |
//| 結果は false                                                     |
//+------------------------------------------------------------------+
bool ChartIsObject(bool &result,const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
   long value;
//--- エラー値をリセットする
   ResetLastError();
//--- チャートプロパティを取得する
   if(!ChartGetInteger(chart_ID,CHART_IS_OBJECT,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      //--- false を返す
      return(false);
     }
//--- チャートプロパティの値をメモリ内に格納する
   result=value;
//--- 実行成功
   return(true);
  }
```

- **CHART_BRING_TO_TOP** はチャートを他の全てのチャートの上に表示します。

```
//+------------------------------------------------------------------+
//| チャートを他の全てのチャートの上に表示するコマンドを端末に送信する  |
//+------------------------------------------------------------------+
bool ChartBringToTop(const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- チャートを他の全てのチャートの上に表示する
   if(!ChartSetInteger(chart_ID,CHART_BRING_TO_TOP,0,true))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

- CHART_MOUSE_SCROLL はマウスの左ボタンを使用してチャートをスクロールするためのプロパティです。

```
//+------------------------------------------------------------------+
//|  マウスの左ボタンを使用してのチャートスクロールが有効かをチェック                  |
//+------------------------------------------------------------------+
bool ChartMouseScrollGet(bool &result,const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
   long value;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_MOUSE_SCROLL,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- チャートプロパティの値をメモリ内に格納する
   result=value;
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  マウスの左ボタンを使用してのチャートスクロールを有/無効化する                   |
//+------------------------------------------------------------------+
bool ChartMouseScrollSet(const bool value,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetInteger(chart_ID,CHART_MOUSE_SCROLL,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- CHART_EVENT_MOUSE_MOVE は移動イベントとマウスクリックイベントに関するメッセージを MQL アプリケーションに送信するプロパティです（CHARTEVENT_MOUSE_MOVE）。

```
//+------------------------------------------------------------------+
//|  移動イベントとマウスクリックイベントに関するメッセージが                        |
//|  MQL アプリケーションに送信されるかをチェックする                          |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool ChartEventMouseMoveGet(bool &result,const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
   long value;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_EVENT_MOUSE_MOVE,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- チャートプロパティの値をメモリ内に格納する
   result=value;
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------------------+
//|  移動イベントとマウスクリックイベントに関するメッセージの                    |
//|  MQL アプリケーションへの送信を有効または無効にする                          |
//+------------------------------------------------------------------------------+
bool ChartEventMouseMoveSet(const bool value,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetInteger(chart_ID,CHART_EVENT_MOUSE_MOVE,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- **CHART_EVENT_OBJECT_CREATE** はグラフィックオブジェクト作成イベントに関するメッセージを MQL アプリケーションに送信するプロパティです（CHARTEVENT_OBJECT_CREATE）。

```
//+------------------------------------------------------------------+
//|  グラフィックオブジェクト作成イベントに関するメッセージが        |
//|  チャート上の全ての MQL アプリケーションに送信されるかをチェック |
//+------------------------------------------------------------------+
bool ChartEventObjectCreateGet(bool &result,const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
   long value;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_EVENT_OBJECT_CREATE,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- チャートプロパティの値をメモリ内に格納する
   result=value;
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------------+
//| チャート上の全ての MQL アプリケーションへのグラフィックオブジェクト                |
//| 作成イベントに関するメッセージの送信を有効または無効にする                      |
//+------------------------------------------------------------------------+
bool ChartEventObjectCreateSet(const bool value,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetInteger(chart_ID,CHART_EVENT_OBJECT_CREATE,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- **CHART_EVENT_OBJECT_DELETE** はグラフィックオブジェクト削除イベントに関するメッセージを MQL アプリケーションに送信するプロパティです（CHARTEVENT_OBJECT_DELETE）。

```
//+------------------------------------------------------------------------+
//| グラフィックオブジェクト削除イベントに関するメッセージが                          |
//| チャート上の全ての MQL アプリケーションに送信されるかをチェック                  |
//+------------------------------------------------------------------------+
bool ChartEventObjectDeleteGet(bool &result,const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
   long value;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_EVENT_OBJECT_DELETE,0,value))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- エキスパート操作ログでエラーメッセージを表示する
       Print(__FUNCTION__+", Error Code = ",GetLastError());
       return(false);
      }
//--- チャートプロパティの値をメモリ内に格納する
   result=value;
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------------+
//| チャート上の全ての MQL アプリケーションへのグラフィックオブジェクト                |
//| 削除イベントに関するメッセージの送信を有効または無効にする                |
//+------------------------------------------------------------------------+
bool ChartEventObjectDeleteSet(const bool value,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetInteger(chart_ID,CHART_EVENT_OBJECT_DELETE,0,value))
      {
       //--- エキスパート操作ログでエラーメッセージを表示する
       Print(__FUNCTION__+", Error Code = ",GetLastError());
       return(false);
      }
//--- 実行成功
   return(true);
  }
```

- **CHART_MODE** はチャートの種類です（ローソク足、バー、及びライン）。

```
//+------------------------------------------------------------------+
//| チャートの表示タイプ(ローソク足、バー、及びライン)を取得する                |
//+------------------------------------------------------------------+
ENUM_CHART_MODE ChartModeGet(const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
   long result=WRONG_VALUE;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_MODE,0,result))
      {
       //--- エキスパート操作ログでエラーメッセージを表示する
       Print(__FUNCTION__+", Error Code = ",GetLastError());
      }
//--- チャートプロパティの値を返す
   return((ENUM_CHART_MODE)result);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//|  チャートの表示タイプ(ローソク足、バー、及びライン)を設定する                  |
//+------------------------------------------------------------------+
bool ChartModeSet(const long value,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetInteger(chart_ID,CHART_MODE,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- CHART_FOREGROUND は価格チャートを前景に表示するプロパティです。

```
//+------------------------------------------------------------------+
//|  価格チャートが前景に表示されるかをチェック                          |
//+------------------------------------------------------------------+
bool ChartForegroundGet(bool &result,const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
   long value;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_FOREGROUND,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- チャートプロパティの値をメモリ内に格納する
   result=value;
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  価格チャートの前景での表示を有効または無効にする                    |
//+------------------------------------------------------------------+
bool ChartForegroundSet(const bool value,const long chart_ID=0)
  {
//--- エラー値をリセットする
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetInteger(chart_ID,CHART_FOREGROUND,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- CHART_SHIFT は右側の境界線からの価格チャートのシフトのモードです。

```
//+------------------------------------------------------------------+
//|  右側の境界線からの価格チャートのシフトが有効かをチェック                                  |
//+------------------------------------------------------------------+
bool ChartShiftGet(bool &result,const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
   long value;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_SHIFT,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- チャートプロパティの値をメモリ内に格納する
   result=value;
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------------------------
+
//|  右側の境界線からの価格チャートのシフトを有効または無効にする
|
//+------------------------------------------------------------------------------------
+
bool ChartShiftSet(const bool value,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetInteger(chart_ID,CHART_SHIFT,0,value))
     {
```

© 2000–2025, MetaQuotes Ltd.

```
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- CHART_AUTOSCROLL はチャートの右側の境界線への自動シフトのモードです。

```
//+------------------------------------------------------------------+
//|  チャートの右側の境界線への自動シフトが                          |
//|  新しいティックの到着時に有効かをチェック                        |
//+------------------------------------------------------------------+
bool ChartAutoscrollGet(bool &result,const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
   long value;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_AUTOSCROLL,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- チャートプロパティの値をメモリ内に格納する
   result=value;
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  新しいティックの到着時のチャートの右側の境界線への自動シフトを  |
//|  有効または無効にする                                            |
//+------------------------------------------------------------------+
bool ChartAutoscrollSet(const bool value,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetInteger(chart_ID,CHART_AUTOSCROLL,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
```

- CHART_SCALE はチャート スケールプロパティです。

```
//+------------------------------------------------------------------+
//|  0から5のチャートスケールを取得する                                |
//+------------------------------------------------------------------+
int ChartScaleGet(const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
   long result=-1;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_SCALE,0,result))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- チャートプロパティの値を返す
   return((int)result);
  }
//+------------------------------------------------------------------+
//|  0から5のチャートスケールを設定する                                |
//+------------------------------------------------------------------+
bool ChartScaleSet(const long value,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetInteger(chart_ID,CHART_SCALE,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- CHART_SCALEFIX はチャート の固 定 スケールのモード です。

```
//+------------------------------------------------------------------+
```

© 2000–2025, MetaQuotes Ltd.

```
//|  チャートの固定スケールのモードが有効かをチェック                                |
//+------------------------------------------------------------------+
bool ChartScaleFixGet(bool &result,const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
   long value;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_SCALEFIX,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- チャートプロパティの値をメモリ内に格納する
   result=value;
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  固定スケールモードを有効または無効にする                                |
//+------------------------------------------------------------------+
bool ChartScaleFixSet(const bool value,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetInteger(chart_ID,CHART_SCALEFIX,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- **CHART_SCALEFIX_11** は 1:1 チャート スケールモード です。

```
//+------------------------------------------------------------------+
//|  1:1 スケールが有効かをチェック                                      |
//+------------------------------------------------------------------+
bool ChartScaleFix11Get(bool &result,const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
   long value;
//--- エラー値をリセットする
```

© 2000–2025, MetaQuotes Ltd.

```
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_SCALEFIX_11,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- チャートプロパティの値をメモリ内に格納する
   result=value;
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  1:1 スケールモードを有効または無効にする                        |
//+------------------------------------------------------------------+
bool ChartScaleFix11Set(const bool value,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetInteger(chart_ID,CHART_SCALEFIX_11,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- CHART_SCALE_PT_PER_BAR はチャート スケールをバーあたりのポイント で指 定 するモード です。

```
//+------------------------------------------------------------------+
//| 「バーあたりのポイント」スケールモードが有効かをチェック         |
//+------------------------------------------------------------------+
bool ChartScalePerBarGet(bool &result,const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
   long value;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_SCALE_PT_PER_BAR,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- チャートプロパティの値をメモリ内に格納する
   result=value;
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 「バーあたりのポイント」チャートスケールモードを有効または無効にする                  |
//+------------------------------------------------------------------+
bool ChartScalePerBarSet(const bool value,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetInteger(chart_ID,CHART_SCALE_PT_PER_BAR,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- **CHART_SHOW_OHLC** は OHLC 値を左上隅に表示するプロパティです。

```
//
+------------------------------------------------------------------------------+
//| OHLC 値の左上隅の表示が有効かをチェック
 |
//
+------------------------------------------------------------------------------+
bool ChartShowOHLCGet(bool &result,const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
   long value;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_SHOW_OHLC,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- チャートプロパティの値をメモリ内に格納する
   result=value;
//--- 実行成功
```

© 2000–2025, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------------------+
//|  チャートの左上隅での OHLC 値の表示を有効または無効にする                    |
//+------------------------------------------------------------------------------+
bool ChartShowOHLCSet(const bool value,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetInteger(chart_ID,CHART_SHOW_OHLC,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- CHART_SHOW_BID_LINE は売値をチャートの水平線として表示するプロパティです。

```
//+------------------------------------------------------------------+
//|  チャート上での売り線の表示が有効かをチェック                    |
//+------------------------------------------------------------------+
bool ChartShowBidLineGet(bool &result,const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
   long value;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_SHOW_BID_LINE,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- チャートプロパティの値をメモリ内に格納する
   result=value;
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  チャート上での売り線の表示を有効または無効にする                |
//+------------------------------------------------------------------+
bool ChartShowBidLineSet(const bool value,const long chart_ID=0)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetInteger(chart_ID,CHART_SHOW_BID_LINE,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- CHART_SHOW_ASK_LINE は買値をチャートの水平線として表示するプロパティです。

```
//+------------------------------------------------------------------+
//| チャート上での買い線の表示が有効かをチェック                     |
//+------------------------------------------------------------------+
bool ChartShowAskLineGet(bool &result,const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
   long value;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_SHOW_ASK_LINE,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- チャートプロパティの値をメモリ内に格納する
   result=value;
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| チャート上での買い線の表示を有効または無効にする                 |
//+------------------------------------------------------------------+
bool ChartShowAskLineSet(const bool value,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetInteger(chart_ID,CHART_SHOW_ASK_LINE,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- CHART_SHOW_LAST_LINE は最終値をチャートの水平線として表示するプロパティです。

```
//+------------------------------------------------------------------------------+
//| 最後の約定価格のラインの表示が有効かをチェック                                |
//+------------------------------------------------------------------------------+
bool ChartShowLastLineGet(bool &result,const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
   long value;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_SHOW_LAST_LINE,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- チャートプロパティの値をメモリ内に格納する
   result=value;
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------------------+
//| 最後の約定価格のライン表示を有効または無効にする                             |
//+------------------------------------------------------------------------------+
bool ChartShowLastLineSet(const bool value,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetInteger(chart_ID,CHART_SHOW_LAST_LINE,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

- CHART_SHOW_PERIOD_SEP は隣接する期間の間に垂直方向の区切り文字を表示するプロパティです。

```
//+------------------------------------------------------------------------------
+
//| 隣接する期間の間の垂直方向の区切り文字の表示が有効かをチェック
|
//+------------------------------------------------------------------------------
+
bool ChartShowPeriodSeparatorGet(bool &result,const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
   long value;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_SHOW_PERIOD_SEP,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- チャートプロパティの値をメモリ内に格納する
   result=value;
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------------------+
//| 隣接する期間の間の垂直方向の区切り文字の表示を有効または無効にする                |
//+------------------------------------------------------------------------------+
bool ChartShowPeriodSepapatorSet(const bool value,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetInteger(chart_ID,CHART_SHOW_PERIOD_SEP,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- CHART_SHOW_GRID はチャートグリッドを表示するプロパティです。

```
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| チャートグリッドが表示されているかをチェック                                    |
//+------------------------------------------------------------------+
bool ChartShowGridGet(bool &result,const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
   long value;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_SHOW_GRID,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- チャートプロパティの値をメモリ内に格納する
   result=value;
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| チャート上でのグリッド表示を有効または無効にする                                |
//+------------------------------------------------------------------+
bool ChartShowGridSet(const bool value,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetInteger(chart_ID,CHART_SHOW_GRID,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- CHART_SHOW_VOLUMES はチャートにボリュームを表示するプロパティです。

```
//+------------------------------------------------------------------+
//| チャートにボリュームが表示されているかをチェック。                              |
//| フラグはボリューム表示モードを示します。                                      |
//+------------------------------------------------------------------+
ENUM_CHART_VOLUME_MODE ChartShowVolumesGet(const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
   long result=WRONG_VALUE;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_SHOW_VOLUMES,0,result))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- チャートプロパティの値を返す
   return((ENUM_CHART_VOLUME_MODE)result);
  }
//+------------------------------------------------------------------+
//| チャート上でのボリューム表示モードを設定する                                |
//+------------------------------------------------------------------+
bool ChartShowVolumesSet(const long value,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetInteger(chart_ID,CHART_SHOW_VOLUMES,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- **CHART_SHOW_OBJECT_DESCR** はグラフィカルオブジェクトのポップアップ説明のプロパティです。

```
//+------------------------------------------------------------------+
//| グラフィカルオブジェクトのポップアップ説明が                                |
//| マウスをホーバーした時に表示されるかをチェック                              |
//+------------------------------------------------------------------+
bool ChartShowObjectDescriptionGet(bool &result,const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
   long value;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_SHOW_OBJECT_DESCR,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
     }
//--- チャートプロパティの値をメモリ内に格納する
   result=value;
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------------+
//| マウスホーバー時のグラフィカルオブジェクトのポップアップ説明を          |
//| 有効または無効にする                                                    |
//+------------------------------------------------------------------------+
bool ChartShowObjectDescriptionSet(const bool value,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetInteger(chart_ID,CHART_SHOW_OBJECT_DESCR,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- CHART_VISIBLE_BARS はチャート上表示のために利用可能なバーの数を定義します。

```
//+------------------------------------------------------------------------+
//| チャートウィンドウに表示されている(可視)バーの数を取得する              |
//+------------------------------------------------------------------------+
int ChartVisibleBars(const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
   long result=-1;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_VISIBLE_BARS,0,result))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- チャートプロパティの値を返す
   return((int)result);
  }
```

© 2000-2025, MetaQuotes Ltd.

- CHART_WINDOWS_TOTAL は指標サブウィンドウを含んだチャートウィンドウの総数を定義します。

```
//+------------------------------------------------------------------+
//|  指標サブウィンドウを含んだチャートウィンドウの総数を取得する                  |
//+------------------------------------------------------------------+
int ChartWindowsTotal(const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
   long result=-1;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_WINDOWS_TOTAL,0,result))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- チャートプロパティの値を返す
   return((int)result);
  }
```

- CHART_WINDOW_IS_VISIBLE はサブウィンドウの可視性を定義します。

```
//+------------------------------------------------------------------+
//|  現在のチャートウィンドウまたはサブウィンドウの可視性をチェック                |
//+------------------------------------------------------------------+
bool ChartWindowsIsVisible(bool &result,const long chart_ID=0,const int
sub_window=0)
  {
//--- プロパティ値を取得する変数を準備する
   long value;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_WINDOW_IS_VISIBLE,sub_window,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- チャートプロパティの値をメモリ内に格納する
   result=value;
//--- 実行成功
   return(true);
  }
```

- CHART_WINDOW_HANDLE はチャートハンドルを返します。

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  チャートハンドルを取得する                                      |
//+------------------------------------------------------------------+
int ChartWindowsHandle(const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
   long result=-1;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_WINDOW_HANDLE,0,result))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- チャートプロパティの値を返す
   return((int)result);
  }
```

- **CHART_WINDOW_YDISTANCE** は指標サブウィンドウの上部フレームとメインチャートウィンドウの上部フレームとの縦 Y 軸に沿ったピクセル単位での距離を定義します。

```
//+------------------------------------------------------------------+
//|  ピクセル単位でのサブウィンドウの上部フレームと                  |
//|  メインチャートウィンドウの上部フレームとの距離を取得する       |
//+------------------------------------------------------------------+
int ChartWindowsYDistance(const long chart_ID=0,const int sub_window=0)
  {
//--- プロパティ値を取得する変数を準備する
   long result=-1;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_WINDOW_YDISTANCE,sub_window,result))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- チャートプロパティの値を返す
   return((int)result);
  }
```

- **CHART_FIRST_VISIBLE_BAR** はチャート上に最初に表示されるバーの番号を返します（バーの索引付けは時系列に従います）。

```
//+------------------------------------------------------------------+
//|  チャート上に最初に表示されるバーの番号を取得する               |
```

© 2000-2025, MetaQuotes Ltd.

```
//|  バーの索引付けは時系列に従い、最新のバーは、最小のインデックスを持っている                  |
//+------------------------------------------------------------------------------+
int ChartFirstVisibleBar(const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
   long result=-1;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_FIRST_VISIBLE_BAR,0,result))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- チャートプロパティの値を返す
   return((int)result);
  }
```

- **CHART_WIDTH_IN_BARS** はチャート幅をバーの数で返します。

```
//+------------------------------------------------------------------+
//|  チャート幅をバーの数で取得する                                  |
//+------------------------------------------------------------------+
int ChartWidthInBars(const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
   long result=-1;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_WIDTH_IN_BARS,0,result))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- チャートプロパティの値を返す
   return((int)result);
  }
```

- **CHART_WIDTH_IN_PIXELS** はチャート幅をピクセル数で返します。

```
//+------------------------------------------------------------------+
//|  チャート幅をピクセル数で取得する                                |
//+------------------------------------------------------------------+
int ChartWidthInPixels(const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
```

© 2000–2025, MetaQuotes Ltd.

```
   long result=-1;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_WIDTH_IN_PIXELS,0,result))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- チャートプロパティの値を返す
   return((int)result);
  }
```

- CHART_HEIGHT_IN_PIXELS はピクセル単位でのチャートの高さを示すプロパティです。

```
//+------------------------------------------------------------------+
//|  チャートの高さをピクセル数で取得する                                |
//+------------------------------------------------------------------+
int ChartHeightInPixelsGet(const long chart_ID=0,const int sub_window=0)
  {
//--- プロパティ値を取得する変数を準備する
   long result=-1;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_HEIGHT_IN_PIXELS,sub_window,result))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- チャートプロパティの値を返す
   return((int)result);
  }
//+------------------------------------------------------------------+
//|  チャートの高さをピクセル数で設定する                                |
//+------------------------------------------------------------------+
bool ChartHeightInPixelsSet(const int value,const long chart_ID=0,const int
sub_window=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetInteger(chart_ID,CHART_HEIGHT_IN_PIXELS,sub_window,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 実行成功
   return(true);
  }
```

- CHART_COLOR_BACKGROUND はチャートの背景色です。

```
//+------------------------------------------------------------------+
//| チャートの背景色を取得する                                       |
//+------------------------------------------------------------------+
color ChartBackColorGet(const long chart_ID=0)
  {
//--- 色を受け取る変数を準備する
   long result=clrNONE;
//--- エラー値をリセットする
   ResetLastError();
//--- チャートの背景色を受け取る
   if(!ChartGetInteger(chart_ID,CHART_COLOR_BACKGROUND,0,result))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- チャートプロパティの値を返す
   return((color)result);
  }
//+------------------------------------------------------------------+
//| チャートの背景色を設定する                                       |
//+------------------------------------------------------------------+
bool ChartBackColorSet(const color clr,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- チャートの背景色を設定する
   if(!ChartSetInteger(chart_ID,CHART_COLOR_BACKGROUND,clr))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- CHART_COLOR_FOREGROUND は軸、スケールとOHLCラインの色です。

```
//+------------------------------------------------------------------+
//| 軸、スケールとOHLCラインの色を取得する                           |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
color ChartForeColorGet(const long chart_ID=0)
  {
//--- 色を受け取る変数を準備する
   long result=clrNONE;
//--- エラー値をリセットする
   ResetLastError();
//--- 軸、スケールとOHLCラインの色を受け取る
   if(!ChartGetInteger(chart_ID,CHART_COLOR_FOREGROUND,0,result))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- チャートプロパティの値を返す
   return((color)result);
  }
//+------------------------------------------------------------------+
//|  軸、スケールとOHLCラインの色を設定する                          |
//+------------------------------------------------------------------+
bool ChartForeColorSet(const color clr,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 軸、スケールとOHLCラインの色を設定する
   if(!ChartSetInteger(chart_ID,CHART_COLOR_FOREGROUND,clr))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- **CHART_COLOR_GRID** はチャート グリッド の色 です。

```
//+------------------------------------------------------------------+
//|  チャートグリッド の色を取得する                                 |
//+------------------------------------------------------------------+
color ChartGridColorGet(const long chart_ID=0)
  {
//--- 色を受け取る変数を準備する
   long result=clrNONE;
//--- エラー値をリセットする
   ResetLastError();
//--- チャートグリッド の色を受け取る
   if(!ChartGetInteger(chart_ID,CHART_COLOR_GRID,0,result))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
```

© 2000-2025, MetaQuotes Ltd.

```
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- チャートプロパティの値を返す
   return((color)result);
  }
//+------------------------------------------------------------------+
//| チャートグリッドの色を設定する                                   |
//+------------------------------------------------------------------+
bool ChartGridColorSet(const color clr,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- チャートグリッドの色を設定する
   if(!ChartSetInteger(chart_ID,CHART_COLOR_GRID,clr))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- CHART_COLOR_VOLUME はボリュームとポジションの初 期 レベルの色 です。

```
//+------------------------------------------------------------------+
//| ボリュームとマーケットエントリーレベルの色を取得する             |
//+------------------------------------------------------------------+
color ChartVolumeColorGet(const long chart_ID=0)
  {
//--- 色を受け取る変数を準備する
   long result=clrNONE;
//--- エラー値をリセットする
   ResetLastError();
//--- ボリュームとマーケットエントリーレベルの色を受け取る
   if(!ChartGetInteger(chart_ID,CHART_COLOR_VOLUME,0,result))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- チャートプロパティの値を返す
   return((color)result);
  }
//+------------------------------------------------------------------+
//| ボリュームとマーケットエントリーレベルの色を設定する             |
//+------------------------------------------------------------------+
bool ChartVolumeColorSet(const color clr,const long chart_ID=0)
  {
```

© 2000–2025, MetaQuotes Ltd.

```
//--- エラー値をリセットする
   ResetLastError();
//--- ボリュームとマーケットエントリーレベルの色を設定する
   if(!ChartSetInteger(chart_ID,CHART_COLOR_VOLUME,clr))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- CHART_COLOR_CHART_UP はアップバーとその影、及び強気ローソク足の実体の縁の色です。

```
//+------------------------------------------------------------------------------+
//|  バーとその影、及び強気ローソク足の縁の色を取得する                          |
//+------------------------------------------------------------------------------+
color ChartUpColorGet(const long chart_ID=0)
  {
//--- 色を受け取る変数を準備する
   long result=clrNONE;
//--- エラー値をリセットする
   ResetLastError();
//--- バーとその影、及び強気ローソク足の縁の色を受け取る
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CHART_UP,0,result))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- チャートプロパティの値を返す
   return((color)result);
  }
//+------------------------------------------------------------------+
//|  バーとその影、及び強気ローソク足の縁の色を設定する              |
//+------------------------------------------------------------------+
bool ChartUpColorSet(const color clr,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- バーとその影、及び強気ローソク足の縁の色を設定する
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CHART_UP,clr))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
```

- CHART_COLOR_CHART_DOWN はダウンバーとその影、及び弱気ローソク足の縁の色です。

```
//+------------------------------------------------------------------------------+
//|  ダウンバーとその影、及び弱気ローソク足の縁の色を取得する                    |
//+------------------------------------------------------------------------------+
color ChartDownColorGet(const long chart_ID=0)
  {
//--- 色を受け取る変数を準備する
   long result=clrNONE;
//--- エラー値をリセットする
   ResetLastError();
//--- ダウンバーとその影、及び弱気ローソク足の縁の色を受け取る
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CHART_DOWN,0,result))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- チャートプロパティの値を返す
   return((color)result);
  }
//+------------------------------------------------------------------------------+
//|  ダウンバーとその影、及び弱気ローソク足の縁の色を設定する                    |
//+------------------------------------------------------------------------------+
bool ChartDownColorSet(const color clr,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- ダウンバーとその影、及び弱気ローソク足の縁の色を設定する
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CHART_DOWN,clr))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- CHART_COLOR_CHART_LINE はチャートの線と同時線の色です。

```
//+------------------------------------------------------------------+
//|  チャートの線と同時線の色を取得する                              |
//+------------------------------------------------------------------+
color ChartLineColorGet(const long chart_ID=0)
```

© 2000–2025, MetaQuotes Ltd.

```
  {
//--- 色を受け取る変数を準備する
   long result=clrNONE;
//--- エラー値をリセットする
   ResetLastError();
//--- チャートの線と同時線の色を受け取る
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CHART_LINE,0,result))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- チャートプロパティの値を返す
   return((color)result);
  }
//+------------------------------------------------------------------+
//|  チャートの線と同時線の色を設定する                                |
//+------------------------------------------------------------------+
bool ChartLineColorSet(const color clr,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- チャートの線と同時線の色を設定する
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CHART_LINE,clr))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- **CHART_COLOR_CANDLE_BULL** は強気ローソク足の実体の色です。

```
//+------------------------------------------------------------------+
//|  強気ローソク足の実体の色を取得する                                |
//+------------------------------------------------------------------+
color ChartBullColorGet(const long chart_ID=0)
  {
//--- 色を受け取る変数を準備する
   long result=clrNONE;
//--- エラー値をリセットする
   ResetLastError();
//--- 強気ローソク足の実体の色を受け取る
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CANDLE_BULL,0,result))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
```

© 2000–2025, MetaQuotes Ltd.

```
      }
//--- チャートプロパティの値を返す
   return((color)result);
  }
//+------------------------------------------------------------------+
//| 強気ローソク足の実体の色を設定する                                |
//+------------------------------------------------------------------+
bool ChartBullColorSet(const color clr,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 強気ローソク足の実体の色を設定する
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CANDLE_BULL,clr))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- CHART_COLOR_CANDLE_BEAR は弱気ローソク足の実体の色です。

```
//+------------------------------------------------------------------+
//| 弱気ローソク足の実体の色を取得する                                |
//+------------------------------------------------------------------+
color ChartBearColorGet(const long chart_ID=0)
  {
//--- 色を受け取る変数を準備する
   long result=clrNONE;
//--- エラー値をリセットする
   ResetLastError();
//--- 弱気ローソク足の実体の色を受け取る
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CANDLE_BEAR,0,result))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- チャートプロパティの値を返す
   return((color)result);
  }
//+------------------------------------------------------------------+
//| 弱気ローソク足の実体の色を設定する                                |
//+------------------------------------------------------------------+
bool ChartBearColorSet(const color clr,const long chart_ID=0)
  {
//--- エラー値をリセットする
```

© 2000–2025, MetaQuotes Ltd.

```
   ResetLastError();
//--- 弱気ローソク足の実体の色を設定する
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CANDLE_BEAR,clr))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- CHART_COLOR_BID は売値線の色です。

```
//+------------------------------------------------------------------+
//| 売値線の色を取得する。</t2>                                      |
//+------------------------------------------------------------------+
color ChartBidColorGet(const long chart_ID=0)
  {
//--- 色を受け取る変数を準備する
   long result=clrNONE;
//--- エラー値をリセットする
   ResetLastError();
//--- 売値線の色を受け取る。</t2>
   if(!ChartGetInteger(chart_ID,CHART_COLOR_BID,0,result))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- チャートプロパティの値を返す
   return((color)result);
  }
//+------------------------------------------------------------------+
//| 売り線の色を設定する                                             |
//+------------------------------------------------------------------+
bool ChartBidColorSet(const color clr,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 売り線の色を設定する
   if(!ChartSetInteger(chart_ID,CHART_COLOR_BID,clr))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
```

- CHART_COLOR_ASK は買値線の色です。

```
//+------------------------------------------------------------------+
//| 買値線の色を取得する                                             |
//+------------------------------------------------------------------+
color ChartAskColorGet(const long chart_ID=0)
  {
//--- 色を受け取る変数を準備する
   long result=clrNONE;
//--- エラー値をリセットする
   ResetLastError();
//--- 買値線の色を受け取る
   if(!ChartGetInteger(chart_ID,CHART_COLOR_ASK,0,result))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- チャートプロパティの値を返す
   return((color)result);
  }
//+------------------------------------------------------------------+
//| 買値線の色を設定する                                             |
//+------------------------------------------------------------------+
bool ChartAskColorSet(const color clr,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 買値線の色を設定する
   if(!ChartSetInteger(chart_ID,CHART_COLOR_ASK,clr))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- CHART_COLOR_LAST は最後の約定価格の線の色です。

```
//+------------------------------------------------------------------+
//| 最後の約定価格の線の色を取得する                                 |
//+------------------------------------------------------------------+
color ChartLastColorGet(const long chart_ID=0)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 色を受け取る変数を準備する
   long result=clrNONE;
//--- エラー値をリセットする
   ResetLastError();
//--- 最後の約定価格の線の色を受け取る
   if(!ChartGetInteger(chart_ID,CHART_COLOR_LAST,0,result))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- チャートプロパティの値を返す
   return((color)result);
  }
//+------------------------------------------------------------------+
//|  最後の約定価格の線の色を設定する                                |
//+------------------------------------------------------------------+
bool ChartLastColorSet(const color clr,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 最後の約定価格の線の色を設定する
   if(!ChartSetInteger(chart_ID,CHART_COLOR_LAST,clr))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- CHART_COLOR_STOP_LEVEL は逆指値レベル(決済逆指値及び決済指値)の色です。

```
//+------------------------------------------------------------------+
//|  逆指値レベル(決済逆指値及び決済指値)の色を取得する              |
//+------------------------------------------------------------------+
color ChartStopLevelColorGet(const long chart_ID=0)
  {
//--- 色を受け取る変数を準備する
   long result=clrNONE;
//--- エラー値をリセットする
   ResetLastError();
//--- 逆指値レベル(決済逆指値及び決済指値)の色を受け取る
   if(!ChartGetInteger(chart_ID,CHART_COLOR_STOP_LEVEL,0,result))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- チャートプロパティの値を返す
   return((color)result);
  }
//+------------------------------------------------------------------+
//|  逆指値レベル(決済逆指値及び決済指値)の色を設定する                |
//+------------------------------------------------------------------+
bool ChartStopLevelColorSet(const color clr,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 逆指値レベル(決済逆指値及び決済指値)の色を設定する
   if(!ChartSetInteger(chart_ID,CHART_COLOR_STOP_LEVEL,clr))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- **CHART_SHOW_TRADE_LEVELS** はチャート上に取引レベルを表示するプロパティです（ポジション、決済逆指値、決済指値及び未決注文のレベル）。

```
//+------------------------------------------------------------------+
//|  取引レベルがチャートに表示されているかをチェック                  |
//+------------------------------------------------------------------+
bool ChartShowTradeLevelsGet(bool &result,const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
   long value;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_SHOW_TRADE_LEVELS,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- チャートプロパティの値をメモリ内に格納する
   result=value;
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  取引レベルの表示を有効または無効にする                          |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool ChartShowTradeLevelsSet(const bool value,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetInteger(chart_ID,CHART_SHOW_TRADE_LEVELS,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- **CHART_DRAG_TRADE_LEVELS** はチャートの取引レベルをマウスでドラッグする能力のプロパティです。

```
//+------------------------------------------------------------------+
//|  チャートの取引レベルをマウスでドラッグする事が出来るかをチェック                  |
//+------------------------------------------------------------------+
bool ChartDragTradeLevelsGet(bool &result,const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
   long value;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_DRAG_TRADE_LEVELS,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- チャートプロパティの値をメモリ内に格納する
   result=value;
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  チャートの取引レベルのマウスドラッグを有効または無効にする                    |
//+------------------------------------------------------------------+
bool ChartDragTradeLevelsSet(const bool value,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetInteger(chart_ID,CHART_DRAG_TRADE_LEVELS,0,value))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- CHART_SHOW_DATE_SCALE はチャートにタイムスケールを表示するプロパティです。

```
//+------------------------------------------------------------------+
//|  チャートにタイムスケールが表示されるかをチェック                |
//+------------------------------------------------------------------+
bool ChartShowDateScaleGet(bool &result,const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
   long value;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_SHOW_DATE_SCALE,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- チャートプロパティの値をメモリ内に格納する
   result=value;
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  チャートでのタイムスケール表示を有効または無効にする            |
//+------------------------------------------------------------------+
bool ChartShowDateScaleSet(const bool value,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetInteger(chart_ID,CHART_SHOW_DATE_SCALE,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

© 2000–2025, MetaQuotes Ltd.

- CHART_SHOW_PRICE_SCALE はチャートに価格スケールを表示するプロパティです。

```
//+------------------------------------------------------------------+
//|  チャートに価格スケール表示する                                  |
//+------------------------------------------------------------------+
bool ChartShowPriceScaleGet(bool &result,const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
   long value;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_SHOW_PRICE_SCALE,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- チャートプロパティの値をメモリ内に格納する
   result=value;
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  チャートでの価格スケール表示を有効または無効にする              |
//+------------------------------------------------------------------+
bool ChartShowPriceScaleSet(const bool value,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetInteger(chart_ID,CHART_SHOW_PRICE_SCALE,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- CHART_SHOW_ONE_CLICK はチャートに「1クリック取引」パネルを表示するプロパティです。

```
//+------------------------------------------------------------------+
//|  チャートに「1クリック取引」パネルが表示されているかをチェック  |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool ChartShowOneClickPanelGet(bool &result,const long chart_ID=0)
  {
//--- プロパティ値を取得する変数を準備する
   long value;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetInteger(chart_ID,CHART_SHOW_ONE_CLICK,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- チャートプロパティの値をメモリ内に格納する
   result=value;
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  チャートでの「1クリック取引」パネルの表示を有効または無効にする                  |
//|                                                                  |
//+------------------------------------------------------------------+
bool ChartShowOneClickPanelSet(const bool value,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetInteger(chart_ID,CHART_SHOW_ONE_CLICK,0,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- CHART_SHIFT_SIZE は百分率で表された右境界線からのゼロバーのシフトサイズです。

```
//+------------------------------------------------------------------+
//|  右境界線からのゼロバーのシフトサイズを取得する                          |
//|  (10% から 50%)                                                  |
//+------------------------------------------------------------------+
double ChartShiftSizeGet(const long chart_ID=0)
  {
//--- 結果取得のために変数を準備する
   double result=EMPTY_VALUE;
//--- エラー値をリセットする
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetDouble(chart_ID,CHART_SHIFT_SIZE,0,result))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- チャートプロパティの値を返す
   return(result);
  }
//+------------------------------------------------------------------------------+
//| 右境界線からのゼロバーのシフトサイズを取得する                                 |
//| （10% から50%）                                                              |
//| シフトモードを有効にするには、CHART_SHIFT プロパティ値を true に設定する必要がある |
//+------------------------------------------------------------------------------+
bool ChartShiftSizeSet(const double value,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetDouble(chart_ID,CHART_SHIFT_SIZE,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- CHART_FIXED_POSITION は百分率で表された、左の境界からのチャートの固定位置です。

```
//
+------------------------------------------------------------------------------
-----+
//| 百分率で表された、左の境界からのチャートの固定位置を取得する
        |
//
+------------------------------------------------------------------------------
-----+
double ChartFixedPositionGet(const long chart_ID=0)
  {
//--- 結果取得のために変数を準備する
   double result=EMPTY_VALUE;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetDouble(chart_ID,CHART_FIXED_POSITION,0,result))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- チャートプロパティの値を返す
   return(result);
  }
//
+------------------------------------------------------------------------------
------+
//| 百分率で表された、左の境界からのチャートの固定位置を取得する
        |
//| チャートの固定位置をみるには CHART_AUTOSCROLL プロパティ値
        |
//| を false にするのが必要
         |
//
+------------------------------------------------------------------------------
------+
bool ChartFixedPositionSet(const double value,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetDouble(chart_ID,CHART_FIXED_POSITION,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- CHART_FIXED_MAX はチャートの固定された最大限です。

```
//+------------------------------------------------------------------+
//| チャートの固定された最大限を取得する                             |
//+------------------------------------------------------------------+
double ChartFixedMaxGet(const long chart_ID=0)
  {
//--- 結果取得のために変数を準備する
   double result=EMPTY_VALUE;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetDouble(chart_ID,CHART_FIXED_MAX,0,result))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- チャートプロパティの値を返す
   return(result);
  }
//+------------------------------------------------------------------+
//| チャートの固定された最大限を設定する                             |
//| プロパティ値をかえるには、CHART_SCALEFIX プロパティ              |
//| 値を予め true にすることが必要                                   |
//+------------------------------------------------------------------+
bool ChartFixedMaxSet(const double value,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetDouble(chart_ID,CHART_FIXED_MAX,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- CHART_FIXED_MIN は チャート の固 定 された最 小 限 です。

```
//+------------------------------------------------------------------+
//| チャートの固定された最大限を取得                                 |
//+------------------------------------------------------------------+
double ChartFixedMinGet(const long chart_ID=0)
  {
//--- 結果取得のために変数を準備する
   double result=EMPTY_VALUE;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetDouble(chart_ID,CHART_FIXED_MIN,0,result))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- チャートプロパティの値を返す
   return(result);
  }
//+------------------------------------------------------------------+
//| チャートの固定された最大限を設定する                             |
//| プロパティ値をかえるには、CHART_SCALEFIX プロパティ              |
//| 値を予め true にすることが必要                                   |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool ChartFixedMinSet(const double value,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetDouble(chart_ID,CHART_FIXED_MIN,value))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- **CHART_POINTS_PER_BAR** はポイント単位のバーあたりのスケールです。

```
//+------------------------------------------------------------------+
//|  ポイント単位のバーあたりのスケールを取得する                                  |
//+------------------------------------------------------------------+
double ChartPointsPerBarGet(const long chart_ID=0)
  {
//--- 結果取得のために変数を準備する
   double result=EMPTY_VALUE;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetDouble(chart_ID,CHART_POINTS_PER_BAR,0,result))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- チャートプロパティの値を返す
   return(result);
  }
//+------------------------------------------------------------------+
//|  チャートスケールを設定する(バーあたりのポイント単位)                          |
//|  このプロパティの値の変化を見るには、                                      |
//|  CHART_SCALE_PT_PER_BAR プロパティを予め true にすることが必要              |
//+------------------------------------------------------------------+
bool ChartPointsPerBarSet(const double value,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetDouble(chart_ID,CHART_POINTS_PER_BAR,value))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- **CHART_PRICE_MIN** はチャートの最小値を返します。

```
//+------------------------------------------------------------------+
//| メインウィンドウかサブウィンドウでチャートの最小値を取得する                |
//+------------------------------------------------------------------+
double ChartPriceMin(const long chart_ID=0,const int sub_window=0)
  {
//--- 結果取得のために変数を準備する
   double result=EMPTY_VALUE;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetDouble(chart_ID,CHART_PRICE_MIN,sub_window,result))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- チャートプロパティの値を返す
   return(result);
  }
```

- **CHART_PRICE_MAX** はチャートの最大値を返します。

```
//+------------------------------------------------------------------+
//| メインウィンドウかサブウィンドウでチャートの最大値を取得する                |
//+------------------------------------------------------------------+
double ChartPriceMax(const long chart_ID=0,const int sub_window=0)
  {
//--- 結果取得のために変数を準備する
   double result=EMPTY_VALUE;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetDouble(chart_ID,CHART_PRICE_MAX,sub_window,result))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- チャートプロパティの値を返す
```

© 2000-2025, MetaQuotes Ltd.

```
   return(result);
  }
```

- CHART_COMMENT はチャート上のコメントです。

```
//+------------------------------------------------------------------+
//| チャートの左上隅のコメントを取得する                             |
//+------------------------------------------------------------------+
bool ChartCommentGet(string &result,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受け取る
   if(!ChartGetString(chart_ID,CHART_COMMENT,result))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| チャートの左上隅のコメントを取得する                             |
//+------------------------------------------------------------------+
bool ChartCommentSet(const string str,const long chart_ID=0)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を設定する
   if(!ChartSetString(chart_ID,CHART_COMMENT,str))
     {
      //--- エキスパート操作ログでエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

- CHART_IS_MAXIMIZED - チャートウィンドウが最大化されている

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  現在のチャートウィンドウが最大化されているかを定義する                |
//+------------------------------------------------------------------+
bool ChartWindowsIsMaximized(bool &result,const long chart_ID=0)
  {
//--- プロパティ値の受信に向けて変数を準備する
   long value;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受信する
   if(!ChartGetInteger(chart_ID,CHART_IS_MAXIMIZED))
     {
      //--- エキスパートログにエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- チャートプロパティ値を変数に格納する
   result=value;
//--- 正常な実行
   return(true);
  }
```

- **CHART_IS_MINIMIZED** - チャート ウィンドウが最小化されている.

```
//+------------------------------------------------------------------+
//|  現在のチャートウィンドウが最小化されているかを定義する                |
//+------------------------------------------------------------------+
bool ChartWindowsIsMinimized(bool &result,const long chart_ID=0)
  {
//--- プロパティ値の受信に向けて変数を準備する
   long value;
//--- エラー値をリセットする
   ResetLastError();
//--- プロパティ値を受信する
   if(!ChartGetInteger(chart_ID,CHART_IS_MINIMIZED))
     {
      //--- エキスパートにエラーメッセージを表示する
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- チャートプロパティ値を変数に格納する
   result=value;
//--- 正常な実行
   return(true);
  }
```

チャート プロパティパネル

```
//--- 制御要素のライブラリを接続する
#include <ChartObjects\ChartObjectsTxtControls.mqh>
//--- 定義済み定数
#define X_PROPERTY_NAME_1     10  // 最初の列のプロパティ名の x 座標
#define X_PROPERTY_VALUE_1    225 // 最初の列のプロパティ値の x 座標
```

© 2000-2025, MetaQuotes Ltd.

```
#define X_PROPERTY_NAME_2     345 // 2 番目の列のプロパティ名の x 座標
#define X_PROPERTY_VALUE_2    550 // 2 番目の列のプロパティ値の x 座標
#define X_BUTTON_1            285 // 最初の列のボタンの x 座標
#define X_BUTTON_2            700 // 2 番目の列のボタンの x 座標
#define Y_PROPERTY_1          30  // 最初及び 2 番目の列の先頭の y 座標
#define Y_PROPERTY_2          286 // 3 番目の列の先頭の y 座標
#define Y_DISTANCE            16  // ライン間のY 軸方向の距離
#define LAST_PROPERTY_NUMBER 111 // 最後のグラフィックプロパティの数
//--- 入力パラメータ
input color InpFirstColor=clrDodgerBlue; // 奇数の線の色
input color InpSecondColor=clrGoldenrod; // 偶数の線の色
//--- 変数と配列
CChartObjectLabel  ExtLabelsName[];  // プロパティ名を表示するためのラベル
CChartObjectLabel  ExtLabelsValue[]; // プロパティ値を表示するためのラベル
CChartObjectButton ExtButtons[];     // ボタン
int                ExtNumbers[];     // プロパティインデックス
string             ExtNames[];       // プロパティ名
uchar              ExtDataTypes[];   // プロパティのデータ型( integer, double, string)
uint               ExtGroupTypes[];  // グループのいずれかに属する特性のデータを格納する配列
uchar              ExtDrawTypes[];   // プロパティ表示の種類のデータを格納する配列
double             ExtMaxValue[];    // パネルの操作時に可能な最大のプロパティ値
double             ExtMinValue[];    // パネルの操作時に可能な最小のプロパティ値
double             ExtStep[];        // プロパティ変更ステップ
int                ExtCount;         // プロパティ総数
color              ExtColors[2];     // 線表示の色の配列
string             ExtComments[2];   // コメント配列(CHART_COMMENT プロパティ)
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- チャートにコメントを表示する
   Comment("SomeComment");
//--- 後で切り替えが出来るように色を配列に収納する
   ExtColors[0]=InpFirstColor;
   ExtColors[1]=InpSecondColor;
//--- 後で切り替えが出来るようにコメントを配列に収納する
   ExtComments[0]="FirstComment";
   ExtComments[1]="SecondComment";
//--- チャートのプロパティを管理するためのコントロールパネルを準備して表示する
   if(!PrepareControls())
      return(INIT_FAILED);
//--- 実行成功
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| エキスパートの初期化解除関数                                     |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- チャートのコメントを削除する
   Comment("");
  }
//+------------------------------------------------------------------+
//| チャートイベントハンドラ                                         |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- チャートクリックイベントをチェック
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      //--- オブジェクト名をセパレータで分ける
      string obj_name[];
      StringSplit(sparam,'_',obj_name);
      //--- オブジェクトがボタンかどうかチェック
      if(obj_name[0]=="Button")
        {
         //--- ボタンインデックスを受け取る
         int index=(int)StringToInteger(obj_name[1]);
         //--- ボタンを放す
         ExtButtons[index].State(false);
         //--- プロパティの新しい値を型に応じて設定する
         if(ExtDataTypes[index]=='I')
            ChangeIntegerProperty(index);
         if(ExtDataTypes[index]=='D')
            ChangeDoubleProperty(index);
         if(ExtDataTypes[index]=='S')
            ChangeStringProperty(index);
        }
     }
//--- プロパティ値を描き直す
   RedrawProperties();
   ChartRedraw();
  }
//+------------------------------------------------------------------+
//| チャートの整数プロパティを変更する                               |
//+------------------------------------------------------------------+
void ChangeIntegerProperty(const int index)
  {
//--- 現プロパティ値を受け取る
   long value=ChartGetInteger(0,(ENUM_CHART_PROPERTY_INTEGER)ExtNumbers[index]);
//--- 次のプロパティ値を定義する
   switch(ExtDrawTypes[index])
     {
      case 'C':
```

© 2000-2025, MetaQuotes Ltd.

```
         value=GetNextColor((color)value);
         break;
      default:
         value=(long)GetNextValue((double)value,index);
         break;
     }
//--- 新しいプロパティ値を設定する
   ChartSetInteger(0,(ENUM_CHART_PROPERTY_INTEGER)ExtNumbers[index],0,value);
  }
//+------------------------------------------------------------------+
//| チャートの double 型プロパティを変更する                          |
//+------------------------------------------------------------------+
void ChangeDoubleProperty(const int index)
  {
//--- 現プロパティ値を受け取る
   double value=ChartGetDouble(0,(ENUM_CHART_PROPERTY_DOUBLE)ExtNumbers[index]);
//--- 次のプロパティ値を定義する
   value=GetNextValue(value,index);
//--- 新しいプロパティ値を設定する
   ChartSetDouble(0,(ENUM_CHART_PROPERTY_DOUBLE)ExtNumbers[index],value);
  }
//+------------------------------------------------------------------+
//| チャートの string 型プロパティを変更する                          |
//+------------------------------------------------------------------+
void ChangeStringProperty(const int index)
  {
//--- ExtComments配列内部を切り替えるための静的変数
   static uint comment_index=1;
//--- 別のコメントを受信するためのインデックスを変更する
   comment_index=1-comment_index;
//--- 新しいプロパティ値を設定する
   ChartSetString(0,(ENUM_CHART_PROPERTY_STRING)
ExtNumbers[index],ExtComments[comment_index]);
  }
//+------------------------------------------------------------------+
//| 次のプロパティ値を取得する                                       |
//+------------------------------------------------------------------+
double GetNextValue(const double value,const int index)
  {
   if(value+ExtStep[index]<=ExtMaxValue[index])
      return(value+ExtStep[index]);
   else
      return(ExtMinValue[index]);
  }
//+------------------------------------------------------------------+
//| color 型プロパティの次の色を取得する                             |
//+------------------------------------------------------------------+
color GetNextColor(const color clr)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 次の color 値を返す
   switch(clr)
     {
      case clrWhite: return(clrRed);
      case clrRed:   return(clrGreen);
      case clrGreen: return(clrBlue);
      case clrBlue:  return(clrBlack);
      default:       return(clrWhite);
     }
  }
//+------------------------------------------------------------------+
//| プロパティ値を描き直す                                           |
//+------------------------------------------------------------------+
void RedrawProperties(void)
  {
//--- プロパティ値のテキスト
   string text;
   long   value;
//--- プロパティ数のループ
   for(int i=0;i<ExtCount;i++)
     {
      text="";
      switch(ExtDataTypes[i])
        {
         case 'I':
            //--- 現プロパティ値を受け取る
            if(!ChartGetInteger(0,(ENUM_CHART_PROPERTY_INTEGER)
ExtNumbers[i],0,value))
            break;
            //--- 整数プロパティのテキスト
            switch(ExtDrawTypes[i])
              {
               //--- color プロパティ
               case 'C':
                  text=(string)((color)value);
                  break;
                  //--- boolean プロパティ
               case 'B':
                  text=(string)((bool)value);
                  break;
                  //--- ENUM_CHART_MODE 列挙プロパティ
               case 'M':
                  text=EnumToString((ENUM_CHART_MODE)value);
                  break;
                  //--- ENUM_CHART_VOLUME_MODE 列挙プロパティ
               case 'V':
                  text=EnumToString((ENUM_CHART_VOLUME_MODE)value);
                  break;
                  //--- int 型の数
```

© 2000-2025, MetaQuotes Ltd.

```
                default:
                   text=IntegerToString(value);
                   break;
                }
             break;
          case 'D':
             //--- double 型プロパティのテキスト
             text=DoubleToString(ChartGetDouble(0,(ENUM_CHART_PROPERTY_DOUBLE)
ExtNumbers[i]),4);
             break;
          case 'S':
             //--- string 型プロパティのテキスト
             text=ChartGetString(0,(ENUM_CHART_PROPERTY_STRING)ExtNumbers[i]);
             break;
          }
       //--- プロパティ値を表示する
       ExtLabelsValue[i].Description(text);
      }
   }
//+------------------------------------------------------------------+
//| チャートプロパティを管理するためのパネルを作成する                |
//+------------------------------------------------------------------+
bool PrepareControls(void)
   {
//--- 予備を含めて配列にメモリを割り当てる
   MemoryAllocation(LAST_PROPERTY_NUMBER+1);
//--- 変数
   int i=0;      // ループ変数
   int col_1=0; // 最初の列のプロパティの数
   int col_2=0; // 2 番目の列のプロパティの数
   int col_3=0; // 3 番目の列のプロパティの数
//--- 現在のプロパティ数は0
   ExtCount=0;
//--- ループでプロパティを探す
   while(i<=LAST_PROPERTY_NUMBER)
     {
      //--- 現プロパティの数を保存する
      ExtNumbers[ExtCount]=i;
      //--- ループ変数の値を増やす
      i++;
      //--- そのような番号を持つプロパティがあるかどうかをチェック
      if(CheckNumber(ExtNumbers[ExtCount],ExtNames[ExtCount],ExtDataTypes[ExtCount],
ExtGroupTypes[ExtCount],ExtDrawTypes[ExtCount]))
        {
         //--- プロパティのための制御要素を作成する
         switch(ExtGroupTypes[ExtCount])
           {
            case 1:
               //--- プロパティのラベルとボタンを作成する
```

© 2000-2025, MetaQuotes Ltd.

```
                if(!ShowProperty(ExtCount,0,X_PROPERTY_NAME_1,X_PROPERTY_VALUE_1,X_BU
TTON_1,Y_PROPERTY_1+col_1*Y_DISTANCE,true))
                return(false);
                //--- 最初の列内の要素の数が増加した
                col_1++;
                break;
             case 2:
                //--- プロパティのラベルとボタンを作成する
                if(!ShowProperty(ExtCount,1,X_PROPERTY_NAME_2,X_PROPERTY_VALUE_2,X_BU
TTON_2,Y_PROPERTY_1+col_2*Y_DISTANCE,true))
                return(false);
                //--- 2 番目の列内の要素の数が増加した
                col_2++;
                break;
             case 3:
                //--- プロパティのみにラベルを作成する
                if(!ShowProperty(ExtCount,2,X_PROPERTY_NAME_2,X_PROPERTY_VALUE_2,0,Y_
PROPERTY_2+col_3*Y_DISTANCE,false))
                return(false);
                //--- 3 番目の列内の要素の数が増加した
                col_3++;
                break;
            }
          //--- プロパティの最大値と最小値とステップを定義する
          GetMaxMinStep(ExtNumbers[ExtCount],ExtMaxValue[ExtCount],ExtMinValue[ExtCou
nt],ExtStep[ExtCount]);
          //--- プロパティ数を増加する
          ExtCount++;
         }
      }
//--- 配列に使用されていないメモリを解放する
   MemoryAllocation(ExtCount);
//--- プロパティ値を描き直す
   RedrawProperties();
   ChartRedraw();
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 配列にメモリを割り当てる                                         |
//+------------------------------------------------------------------+
void MemoryAllocation(const int size)
  {
   ArrayResize(ExtLabelsName,size);
   ArrayResize(ExtLabelsValue,size);
   ArrayResize(ExtButtons,size);
   ArrayResize(ExtNumbers,size);
   ArrayResize(ExtNames,size);
   ArrayResize(ExtDataTypes,size);
```

© 2000-2025, MetaQuotes Ltd.

```
   ArrayResize(ExtGroupTypes,size);
   ArrayResize(ExtDrawTypes,size);
   ArrayResize(ExtMaxValue,size);
   ArrayResize(ExtMinValue,size);
   ArrayResize(ExtStep,size);
  }
//+------------------------------------------------------------------+
//| プロパティインデックスが                                         |
//| ENUM_CHART_PROPERTIES 列挙値の 1 つかどうかチェックする          |
//+------------------------------------------------------------------+
bool CheckNumber(const int ind,string &name,uchar &data_type,uint &group_type,uchar
&draw_type)
  {
//--- プロパティが整数型かどうかチェックする
   ResetLastError();
   name=EnumToString((ENUM_CHART_PROPERTY_INTEGER)ind);
   if(_LastError==0)
     {
      data_type='I';                          // ENUM_CHART_PROPERTY_INTEGER 列挙のプロパティ
      GetTypes(ind,group_type,draw_type); // プロパティの表示パラメータを定義する
      return(true);
     }
//--- プロパティが double 型かどうかチェックする
   ResetLastError();
   name=EnumToString((ENUM_CHART_PROPERTY_DOUBLE)ind);
   if(_LastError==0)
     {
      data_type='D';                          // ENUM_CHART_PROPERTY_DOUBLE 列挙のプロパティ
      GetTypes(ind,group_type,draw_type); // プロパティの表示パラメータを定義する
      return(true);
     }
//--- プロパティが string 型かどうかチェックする
   ResetLastError();
   name=EnumToString((ENUM_CHART_PROPERTY_STRING)ind);
   if(_LastError==0)
     {
      data_type='S';                          // ENUM_CHART_PROPERTY_STRING 列挙のプロパティ
      GetTypes(ind,group_type,draw_type); // プロパティの表示パラメータを定義する
      return(true);
     }
//--- プロパティがいずれの列挙にも属しない
   return(false);
  }
//+------------------------------------------------------------------+
//| プロパティを格納すべきグループ、ならびにその表示タイプを         |
//| 定義する                                                         |
//+------------------------------------------------------------------+
void GetTypes(const int property_number,uint &group_type,uchar &draw_type)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- プロパティが3 番目のグループに属するかどうかチェックする
//--- 3 番目のグループのプロパティは、CHART_BRING_TO_TOP から始まる2 番目の列に表示されている
   if(CheckThirdGroup(property_number,group_type,draw_type))
      return;
//--- プロパティが 2 番目のグループに属するかどうかチェックする
//--- 2 番目のグループのプロパティは、2 番目の列に表示されている
   if(CheckSecondGroup(property_number,group_type,draw_type))
      return;
//--- ここに来た場合、プロパティは最初のグループ(最初の列)に属している
   CheckFirstGroup(property_number,group_type,draw_type);
  }
//+------------------------------------------------------------------+
//| プロパティが3 番目のグループに属するかどうかチェックして、                |
//| 属する場合はその表示タイプを定義する                                |
//+------------------------------------------------------------------+
bool CheckThirdGroup(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- プロパティが3 番目のグループに属するかどうかチェックする
   switch(property_number)
     {
      //--- boolean プロパティ
      case CHART_IS_OBJECT:
      case CHART_WINDOW_IS_VISIBLE:
         draw_type='B';
         break;
         //--- integer プロパティ
      case CHART_VISIBLE_BARS:
      case CHART_WINDOWS_TOTAL:
      case CHART_WINDOW_HANDLE:
      case CHART_WINDOW_YDISTANCE:
      case CHART_FIRST_VISIBLE_BAR:
      case CHART_WIDTH_IN_BARS:
      case CHART_WIDTH_IN_PIXELS:
         draw_type='I';
         break;
         //--- double 型プロパティ
      case CHART_PRICE_MIN:
      case CHART_PRICE_MAX:
         draw_type='D';
         break;
         //--- 実際には、このプロパティはチャートを他の全てのチャートの上に表示するコマンドである
         //--- このウィンドウは使用される前には常に他のウィンドウの上にあるので
         //--- パネルに適用する要はない
      case CHART_BRING_TO_TOP:
         draw_type=' ';
         break;
         //--- プロパティが3 番目のグループに属しない
      default:
         return(false);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- プロパティが3 番目のグループに属す
   group_type=3;
   return(true);
  }
//+------------------------------------------------------------------------+
//| プロパティが第 2 グループに属するかどうかチェックして、                                |
//| 属する場合はその表示タイプを定義する                                          |
//+------------------------------------------------------------------------+
bool CheckSecondGroup(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- プロパティが 2 番目のグループに属するかどうかチェックする
   switch(property_number)
     {
      //--- ENUM_CHART_MODE 型のプロパティ
      case CHART_MODE:
         draw_type='M';
         break;
         //--- ENUM_CHART_VOLUME_MODE 型のプロパティ
      case CHART_SHOW_VOLUMES:
         draw_type='V';
         break;
         //--- string 型プロパティ
      case CHART_COMMENT:
         draw_type='S';
         break;
         //--- color プロパティ
      case CHART_COLOR_BACKGROUND:
      case CHART_COLOR_FOREGROUND:
      case CHART_COLOR_GRID:
      case CHART_COLOR_VOLUME:
      case CHART_COLOR_CHART_UP:
      case CHART_COLOR_CHART_DOWN:
      case CHART_COLOR_CHART_LINE:
      case CHART_COLOR_CANDLE_BULL:
      case CHART_COLOR_CANDLE_BEAR:
      case CHART_COLOR_BID:
      case CHART_COLOR_ASK:
      case CHART_COLOR_LAST:
      case CHART_COLOR_STOP_LEVEL:
         draw_type='C';
         break;
         //--- プロパティが第 2 グループに属しない
      default:
         return(false);
     }
//--- プロパティが3 番目のグループに属す
   group_type=2;
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| プロパティが第 2 及び第三のプロパティグループに属していないことだけが既に知られている時のみに |
//| 呼び出される                                                     |
//+------------------------------------------------------------------+
void CheckFirstGroup(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- プロパティが1 番目のグループに属す
   group_type=1;
//--- プロパティの表示タイプを定義する
   switch(property_number)
     {
      //--- integer プロパティ
      case CHART_SCALE:
      case CHART_HEIGHT_IN_PIXELS:
         draw_type='I';
         return;
         //--- double 型プロパティ
      case CHART_SHIFT_SIZE:
      case CHART_FIXED_POSITION:
      case CHART_FIXED_MAX:
      case CHART_FIXED_MIN:
      case CHART_POINTS_PER_BAR:
         draw_type='D';
         return;
         //--- boolean プロパティのみが残される
      default:
         draw_type='B';
         return;
     }
  }
//+------------------------------------------------------------------+
//| プロパティのラベルとボタンを作成する                             |
//+------------------------------------------------------------------+
bool ShowProperty(const int ind,const int type,const int x1,const int x2,
                  const int xb,const int y,const bool btn)
  {
//--- ExtColors 色配列の内部を切り替えるための静的変数
   static uint color_index[3]={1,1,1};
//--- 別の色を受け取るためにインデックスを変更する
   color_index[type]=1-color_index[type];
//--- プロパティのラベルとボタン(btn=true の場合)を表示する
   if(!LabelCreate(ExtLabelsName[ind],"name_"+(string)
ind,ExtNames[ind],ExtColors[color_index[type]],x1,y))
      return(false);
   if(!LabelCreate(ExtLabelsValue[ind],"value_"+(string)
ind,"",ExtColors[color_index[type]],x2,y))
      return(false);
   if(btn && !ButtonCreate(ExtButtons[ind],(string)ind,xb,y+1))
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| ラベルを作成する                                                 |
//+------------------------------------------------------------------+
bool LabelCreate(CChartObjectLabel &lbl,const string name,const string text,
                 const color clr,const int x,const int y)
  {
   if(!lbl.Create(0,"Label_"+name,0,x,y)) return(false);
   if(!lbl.Description(text))             return(false);
   if(!lbl.FontSize(10))                  return(false);
   if(!lbl.Color(clr))                    return(false);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| ボタンを作成する                                                 |
//+------------------------------------------------------------------+
bool ButtonCreate(CChartObjectButton &btn,const string name,
                  const int x,const int y)
  {
   if(!btn.Create(0,"Button_"+name,0,x,y,50,15)) return(false);
   if(!btn.Description("Next"))                  return(false);
   if(!btn.FontSize(10))                         return(false);
   if(!btn.Color(clrBlack))                      return(false);
   if(!btn.BackColor(clrWhite))                  return(false);
   if(!btn.BorderColor(clrBlack))                return(false);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| プロパティの最大値と最小値とステップを定義する                   |
//+------------------------------------------------------------------+
void GetMaxMinStep(const int property_number,double &max,double &min,double &step)
  {
   double value;
//--- プロパティ値を型に応じて設定する
   switch(property_number)
     {
      case CHART_SCALE:
         max=5;
         min=0;
         step=1;
         break;
      case CHART_MODE:
      case CHART_SHOW_VOLUMES:
         max=2;
```

© 2000–2025, MetaQuotes Ltd.

```
         min=0;
         step=1;
         break;
      case CHART_SHIFT_SIZE:
         max=50;
         min=10;
         step=2.5;
         break;
      case CHART_FIXED_POSITION:
         max=90;
         min=0;
         step=15;
         break;
      case CHART_POINTS_PER_BAR:
         max=19;
         min=1;
         step=3;
         break;
      case CHART_FIXED_MAX:
         value=ChartGetDouble(0,CHART_FIXED_MAX);
         max=value*1.25;
         min=value;
         step=value/32;
         break;
      case CHART_FIXED_MIN:
         value=ChartGetDouble(0,CHART_FIXED_MIN);
         max=value;
         min=value*0.75;
         step=value/32;
         break;
      case CHART_HEIGHT_IN_PIXELS:
         max=700;
         min=520;
         step=30;
         break;
         //--- 初期値
      default:
         max=1;
         min=0;
         step=1;
     }
  }
```

© 2000–2025, MetaQuotes Ltd.

オブジェクト定数

作成され、価格チャートに表示出来るグラフィックオブジェクトは 44 あります。オブジェクト操作に使用される定数は 9 つのグループに分かれています。

- オブジェクトの型 – グラフィックオブジェクトの識別子
- オブジェクトプロパティ – グラフィックオブジェクトのプロパティの設定と取得
- オブジェクト結合の方法 – チャート内のオブジェクトの位置の定数
- 結合コーナー – オブジェクトが相対的に位置されるチャートのコーナー
- オブジェクトの可視性 – オブジェクトが可視である時間軸の設定
- エリオット波動レベル – 波階調
- ギャンオブジェクト – ギャンファンとギャングリッドのトレンド定数
- ウェブカラー – 事前に定義されたウェブカラーの定数
- Wingdingsフォント – Wingdingsフォントの文字のコードの一覧

© 2000–2025, MetaQuotes Ltd.

オブジェクト型

グラフィックオブジェクトが ObjectCreate() 関数で作成される際、ENUM_OBJECT 列挙体の値のいずれかである作成中のオブジェクトの型を指定する必要があります。オブジェクトプロパティの更なる仕様はグラフィックオブジェクト操作関数を使用して可能となります。

ENUM_OBJECT

ID		説明
OBJ_VLINE		垂直線。
OBJ_HLINE		水平線。
OBJ_TREND		傾向線。
OBJ_TRENDBYANGLE		角度による傾向線。
OBJ_CYCLES		サイクルライン。
OBJ_ARROWED_LINE		矢印。
OBJ_CHANNEL		等距離チャンネル。
OBJ_STDDEVCHANNEL		標準偏差チャンネル。
OBJ_REGRESSION		線形回帰チャンネル。
OBJ_PITCHFORK		アンドリューズピッチフォーク。
OBJ_GANNLINE		ギャンライン。
OBJ_GANNFAN		ギャンファン（扇）。
OBJ_GANNGRID		ギャングリッド。
OBJ_FIBO		フィボナッチリトレースメント。
OBJ_FIBOTIMES		フィボナッチ タイムゾーン。
OBJ_FIBOFAN		フィボナッチファン（扇）。
OBJ_FIBOARC		フィボナッチアーク（円弧）。
OBJ_FIBOCHANNEL		フィボナッチ チャンネル。
OBJ_EXPANSION		フィボナッチ拡張。
OBJ_ELLIOTWAVE5		エリオット動機波。
OBJ_ELLIOTWAVE3		エリオット修正波。
OBJ_RECTANGLE		長方形。
OBJ_TRIANGLE		三角形。
OBJ_ELLIPSE		楕円形。
OBJ_ARROW_THUMB_UP		サムズアップ。
OBJ_ARROW_THUMB_DOWN		サムズダウン。

© 2000–2025, MetaQuotes Ltd.

ID		説明
OBJ_ARROW_UP		上矢印。
OBJ_ARROW_DOWN		下矢印。
OBJ_ARROW_STOP		ストップサイン。
OBJ_ARROW_CHECK		チェックサイン。
OBJ_ARROW_LEFT_PRICE		左プライスラベル。
OBJ_ARROW_RIGHT_PRICE		右プライスラベル。
OBJ_ARROW_BUY		買いサイン。
OBJ_ARROW_SELL		売りサイン。
OBJ_ARROW		矢印。
OBJ_TEXT		テキスト。
OBJ_LABEL		ラベル。
OBJ_BUTTON		ボタン。
OBJ_CHART		チャート。
OBJ_BITMAP		ビットマップ。
OBJ_BITMAP_LABEL		ビットマップラベル。
OBJ_EDIT		編集。
OBJ_EVENT		経済カレンダーのイベントに対応する「イベント」オブジェクト。
OBJ_RECTANGLE_LABEL		カスタムグラフィカルインタフェースを作成及び設計するための「四角形ラベル」オブジェクト。

© 2000–2025, MetaQuotes Ltd.

OBJ_VLINE

垂直線



注意事項

垂直線を描画する際は、全てのチャートウィンドウのライン表示モードを設定することが出来ます（OBJPROP_RAYプロパティ）。

例

次のスクリプトは、チャート上で垂直線を作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script draws \"Vertical Line\" graphical object."
#property description "Anchor point date is set in percentage of"
#property description "the chart window width in bars."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string           InpName="VLine";      // 線の名称
input int              InpDate=25;           // % で表されたイベントの日
input color            InpColor=clrRed;      // 線の色
input ENUM_LINE_STYLE  InpStyle=STYLE_DASH;  // 線のスタイル
input int              InpWidth=3;           // 線の幅
input bool             InpBack=false;        // 背景の線
input bool             InpSelection=true;    // 強調表示して移動
input bool             InpRay=true;          // 線が下に続く
```

© 2000–2025, MetaQuotes Ltd.

```
input bool             InpHidden=true;       // オブジェクトリストに隠す
input long             InpZOrder=0;          // マウスクリックの優先順位
//+------------------------------------------------------------------+
//| 垂直線を作成する                                                 |
//+------------------------------------------------------------------+
bool VLineCreate(const long            chart_ID=0,        // チャート識別子
                 const string          name="VLine",      // 線の名称
                 const int             sub_window=0,      // サブウィンドウ番号
                 datetime              time=0,            // 線の時間
                 const color           clr=clrRed,        // 線の色
                 const ENUM_LINE_STYLE style=STYLE_SOLID, // 線のスタイル
                 const int             width=1,           // 線の幅
                 const bool            back=false,        // 背景で表示する
                 const bool            selection=true,    // 強調表示して移動
                 const bool            ray=true,          // 線が下に続く
                 const bool            hidden=true,       // オブジェクトリストに隠す
                 const long            z_order=0)         // マウスクリックの優先順位
  {
//--- 線の時間が設定されていない場合、最後のバーで描画する
   if(!time)
      time=TimeCurrent();
//--- エラー値をリセットする
   ResetLastError();
//--- 垂直線を作成する
   if(!ObjectCreate(chart_ID,name,OBJ_VLINE,sub_window,time,0))
     {
      Print(__FUNCTION__,
            ": failed to create a vertical line! Error code = ",GetLastError());
      return(false);
     }
//--- 線の色を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 線の表示スタイルを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 線の幅を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- マウスで線を移動させるモードを有効(true)か無効(false)にする
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- チャートサブウィンドウで線を表示するモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY,ray);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 垂直線を移動する                                                 |
//+------------------------------------------------------------------+
bool VLineMove(const long   chart_ID=0,   // チャート識別子
               const string name="VLine", // 線の名称
               datetime     time=0)       // 線の時間
  {
//--- 時間が設定されていない場合、最後のバーに移動する
   if(!time)
      time=TimeCurrent();
//--- エラー値をリセットする
   ResetLastError();
//--- 垂直線を移動する
   if(!ObjectMove(chart_ID,name,0,time,0))
     {
      Print(__FUNCTION__,
            ": failed to move the vertical line! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 垂直線を削除する                                                 |
//+------------------------------------------------------------------+
bool VLineDelete(const long   chart_ID=0,   // チャート識別子
                 const string name="VLine") // 線の名称
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 垂直線を削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete the vertical line! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                               |
//+------------------------------------------------------------------+
void OnStart()
  {
```

© 2000–2025, MetaQuotes Ltd.

```
//--- 入力パラメータの正しさをチェックする
   if(InpDate<0 || InpDate>100)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- イベントアンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
//--- メモリ割り当て
   ArrayResize(date,bars);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 線を描画するポイントを定義する
   int d=InpDate*(bars-1)/100;
//--- 垂直線を作成する
   if(!VLineCreate(0,InpName,0,date[d],InpColor,InpStyle,InpWidth,InpBack,
      InpSelection,InpRay,InpHidden,InpZOrder))
      return;
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- 線を移動する
//--- ループカウンタ
   int h_steps=bars/2;
//--- 線を移動する
   for(int i=0;i<h_steps;i++)
     {
      //--- 次の値を使用する
      if(d<bars-1)
         d+=1;
      //--- ポイントを移動する
      if(!VLineMove(0,InpName,date[d]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
      // 0.03 秒の遅れ
      Sleep(30);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 1 秒の遅れ
   Sleep(1000);
//--- チャートからチャンネルを削除する
   VLineDelete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_HLINE

水平線



例

次のスクリプトは、チャート上で水平線を作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script draws \"Horizontal Line\" graphical object."
#property description "Anchor point price is set in percentage of the height of"
#property description "the chart window."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string           InpName="HLine";      // 線の名称
input int              InpPrice=25;          // % で表された線の価格
input color            InpColor=clrRed;      // 線の色
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // 線のスタイル
input int              InpWidth=3;           // 線の幅
input bool             InpBack=false;        // 背景の線
input bool             InpSelection=true;    // 強調表示して移動
input bool             InpHidden=true;       // オブジェクトリストに隠す
input long             InpZOrder=0;          // マウスクリックの優先順位
//+------------------------------------------------------------------+
//|  水平線を作成する                                                |
//+------------------------------------------------------------------+
```

© 2000–2025, MetaQuotes Ltd.

```
bool HLineCreate(const long            chart_ID=0,        // チャート識別子
                 const string          name="HLine",      // 線の名称
                 const int             sub_window=0,      // サブウィンドウ番号
                 double                price=0,           // 線の価格
                 const color           clr=clrRed,        // 線の色
                 const ENUM_LINE_STYLE style=STYLE_SOLID, // 線のスタイル
                 const int             width=1,           // 線の幅
                 const bool            back=false,        // 背景で表示する
                 const bool            selection=true,    // 強調表示して移動
                 const bool            hidden=true,       // オブジェクトリストに隠す
                 const long            z_order=0)         // マウスクリックの優先順位
  {
//--- 価格が設定されていない場合、現在の売値レベルで設定する
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- 水平線を作成する
   if(!ObjectCreate(chart_ID,name,OBJ_HLINE,sub_window,0,price))
     {
      Print(__FUNCTION__,
            ": failed to create a horizontal line! Error code = ",GetLastError());
      return(false);
     }
//--- 線の色を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 線の表示スタイルを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 線の幅を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- マウスで線を移動させるモードを有効(true)か無効(false)にする
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  水平線を移動する                                                |
//+------------------------------------------------------------------+
bool HLineMove(const long   chart_ID=0,   // チャート識別子
```

© 2000-2025, MetaQuotes Ltd.

```
                const string name="HLine", // 線の名称
                double       price=0)      // 線の価格
  {
//--- 価格が設定されていない場合、現在の売値レベルへ移す
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- 水平線を移動する
   if(!ObjectMove(chart_ID,name,0,0,price))
     {
      Print(__FUNCTION__,
            ": failed to move the horizontal line! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  水平線を削除する                                                |
//+------------------------------------------------------------------+
bool HLineDelete(const long   chart_ID=0,   // チャート識別子
                 const string name="HLine") // 線の名称
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 水平線を削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete a horizontal line! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
   if(InpPrice<0 || InpPrice>100)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- 価格配列サイズ
   int accuracy=1000;
```

© 2000–2025, MetaQuotes Ltd.

```
//--- アンカーポイントの座標の設定と変更に使用される
//--- 価格の値を格納するための配列
   double price[];
//--- メモリ割り当て
   ArrayResize(price,accuracy);
//--- 価格配列に書き込む
//--- チャートの最高値と最安値を見つける
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 線を描画するポイントを定義する
   int p=InpPrice*(accuracy-1)/100;
//--- 水平線を作成する
   if(!HLineCreate(0,InpName,0,price[p],InpColor,InpStyle,InpWidth,InpBack,
      InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- 線を移動する
//--- ループカウンタ
   int v_steps=accuracy/2;
//--- 線を移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p<accuracy-1)
         p+=1;
      //--- ポイントを移動する
      if(!HLineMove(0,InpName,price[p]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- チャートから削除する
   HLineDelete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
```

© 2000-2025, MetaQuotes Ltd.

```
//---
  }
```

© 2000–2025, MetaQuotes Ltd.

OBJ_TREND

傾向線



注意事項

傾向線では表示継続のモードを右及び/または左に指定することが出来ます（OBJPROP_RAY_RIGHT 及び OBJPROP_RAY_LEFT プロパティ）。

例

次のスクリプトは、チャート上で傾向線を作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script draws \"Trend Line\" graphical object."
#property description "Anchor point coordinates are set in percentage of"
#property description "the chart window size."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string           InpName="Trend";      // 線の名称
input int              InpDate1=35;          // 1 番目のポイントの日付( % )
input int              InpPrice1=60;         // 1 番目のポイントの価格( % )
input int              InpDate2=65;          // 2 番目のポイントの日付( % )
input int              InpPrice2=40;         // 2 番目のポイントの価格( % )
input color            InpColor=clrRed;      // 線の色
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // 線のスタイル
```

© 2000–2025, MetaQuotes Ltd.

```
input int              InpWidth=2;          // 線の幅
input bool             InpBack=false;       // 背景の線
input bool             InpSelection=true;   // 強調表示して移動
input bool             InpRayLeft=false;    // 線の左への継続
input bool             InpRayRight=false;   // 線の右への継続
input bool             InpHidden=true;      // オブジェクトリストに隠す
input long             InpZOrder=0;         // マウスクリックの優先順位
//+------------------------------------------------------------------+
//| 与えられた座標で傾向線を作成する                                 |
//+------------------------------------------------------------------+
bool TrendCreate(const long            chart_ID=0,        // チャート識別子
                 const string          name="TrendLine",  // 線の名称
                 const int             sub_window=0,      // サブウィンドウ番号
                 datetime              time1=0,           // 1 番目のポイントの時間
                 double                price1=0,          // 1 番目のポイントの価格
                 datetime              time2=0,           // 2 番目のポイントの時間
                 double                price2=0,          // 2 番目のポイントの価格
                 const color           clr=clrRed,        // 線の色
                 const ENUM_LINE_STYLE style=STYLE_SOLID, // 線のスタイル
                 const int             width=1,           // 線の幅
                 const bool            back=false,        // 背景で表示する
                 const bool            selection=true,    // 強調表示して移動
                 const bool            ray_left=false,    // 線の左への継続
                 const bool            ray_right=false,   // 線の右への継続
                 const bool            hidden=true,       // オブジェクトリストに隠す
                 const long            z_order=0)         // マウスクリックの優先順位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeTrendEmptyPoints(time1,price1,time2,price2);
//--- エラー値をリセットする
   ResetLastError();
//--- 与えられた座標で傾向線を作成する
   if(!ObjectCreate(chart_ID,name,OBJ_TREND,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": failed to create a trend line! Error code = ",GetLastError());
      return(false);
     }
//--- 線の色を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 線の表示スタイルを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 線の幅を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- マウスで線を移動させるモードを有効(true)か無効(false)にする
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
```

© 2000-2025, MetaQuotes Ltd.

```
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 線の表示を左に延長するモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- 線の表示を右に延長するモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 傾向線のアンカーポイントを移動する                                |
//+------------------------------------------------------------------+
bool TrendPointChange(const long   chart_ID=0,       // チャート識別子
                      const string name="TrendLine", // 線の名称
                      const int    point_index=0,    // アンカーポイントのインデックス
                      datetime     time=0,           // アンカーポイントの時間座標
                      double       price=0)          // アンカーポイントの価格座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- 傾向線のアンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| この関数はチャートから傾向線を削除する                           |
//+------------------------------------------------------------------+
bool TrendDelete(const long   chart_ID=0,       // チャート識別子
                 const string name="TrendLine") // 線の名称
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 傾向線を削除する
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete a trend line! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 傾向線のアンカーポイントの値をチェックして                       |
//| 空の物には初期値を設定する                                       |
//+------------------------------------------------------------------+
void ChangeTrendEmptyPoints(datetime &time1,double &price1,
                            datetime &time2,double &price2)
  {
//--- 1 番目のポイントの時間が設定されていない場合、現在足になる
   if(!time1)
      time1=TimeCurrent();
//--- 1 番目のポイントの価格が設定されていない場合、売値になる
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 2 番目のポイントの時間が設定されていない場合、2 番目から 9 バー左に置かれる
   if(!time2)
     {
      //--- 最後の 10 バーのオープン時間を受信するための配列
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- 2 番目のポイントを最初のものから 9 バー左に設定する
      time2=temp[0];
     }
//--- 2 番目のポイントの価格は設定されていない場合は 1 番目のポイントと等しい
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 価格配列サイズ
   int accuracy=1000;
//--- アンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
   double   price[];
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 価格配列に書き込む
//--- チャートの最高値と最安値を見つける
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 線を描画するポイントを定義する
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- 傾向線を作成する
   if(!TrendCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor
,InpStyle,
      InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- 線のアンカーポイントを移動する
//--- ループカウンタ
   int v_steps=accuracy/5;
//--- 1 番目のアンカーポイントを垂直方向に移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p1>1)
         p1-=1;
      //--- ポイントを移動する
```

© 2000-2025, MetaQuotes Ltd.

```
      if(!TrendPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 2 番目のアンカーポイントを垂直方向に移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p2<accuracy-1)
         p2+=1;
      //--- ポイントを移動する
      if(!TrendPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 半秒の遅れ
   Sleep(500);
//--- ループカウンタ
   int h_steps=bars/2;
//--- 両方のアンカーポイントを同時に水平方向に移動させる
   for(int i=0;i<h_steps;i++)
     {
      //--- 次の値を使用する
      if(d1<bars-1)
         d1+=1;
      if(d2>1)
         d2-=1;
      //--- ポイントをシフトする
      if(!TrendPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!TrendPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
      // 0.03 秒の遅れ
      Sleep(30);
     }
//--- 1 秒の遅れ
```

© 2000-2025, MetaQuotes Ltd.

```
   Sleep(1000);
//--- 傾向線を削除する
   TrendDelete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//---
  }
```

© 2000–2025, MetaQuotes Ltd.

OBJ_TRENDBYANGLE

角度による傾向線



注意事項

角度による傾向線では表示継続のモードを右及び/または左に指定することが出来ます（OBJPROP_RAY_RIGHT 及び OBJPROP_RAY_LEFT プロパティ）。

角度及び 2 番目のアンカー点の座標の両方が、線の傾きを設定するために使用出来ます。

例

次のスクリプトは、チャート上で傾向線を作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script draws \"Trend Line By Angle\" graphical object."
#property description "Anchor point coordinates are set in percentage of the size
of"
#property description "the chart window."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string           InpName="Trend";      // 線の名称
input int              InpDate1=50;          // 1 番目のポイントの日付( % )
input int              InpPrice1=75;         // 1 番目のポイントの価格( % )
input int              InpAngle=0;           // 線の傾斜
input color            InpColor=clrRed;      // 線の色
```

© 2000–2025, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // 線のスタイル
input int             InpWidth=2;          // 線の幅
input bool            InpBack=false;       // 背景の線
input bool            InpSelection=true;   // 強調表示して移動
input bool            InpRayLeft=false;    // 線の左への継続
input bool            InpRayRight=true;    // 線の右への継続
input bool            InpHidden=true;      // オブジェクトリストに隠す
input long            InpZOrder=0;         // マウスクリックの優先順位
//+------------------------------------------------------------------+
//| 角度での傾向線を作成する                                         |
//+------------------------------------------------------------------+
bool TrendByAngleCreate(const long             chart_ID=0,        // チャート識別子
                        const string           name="TrendLine",  // 線の名称
                        const int              sub_window=0,      // サブウィンドウ番号
                        datetime               time=0,            // ポイントの時間
                        double                 price=0,           // ポイントの価格
                        const double           angle=45.0,        // 傾斜
                        const color            clr=clrRed,        // 線の色
                        const ENUM_LINE_STYLE  style=STYLE_SOLID, // 線のスタイル
                        const int              width=1,           // 線の幅
                        const bool             back=false,        // 背景で表示する
                        const bool             selection=true,    // 強調表示して移動
                        const bool             ray_left=false,    // 線の左への継続
                        const bool             ray_right=true,    // 線の右への継続
                        const bool             hidden=true,       // オブジェクトリストに隠す
                        const long             z_order=0)         // マウスクリックの優先順
位
  {
//--- マウスの傾向線のドラッグを容易にするために、2 番目の点を作成する
   datetime time2=0;
   double   price2=0;
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeTrendEmptyPoints(time,price,time2,price2);
//--- エラー値をリセットする
   ResetLastError();
//--- 2 点を使用して傾向線を作成する
   if(!ObjectCreate(chart_ID,name,OBJ_TRENDBYANGLE,sub_window,time,price,time2,price
2))
     {
      Print(__FUNCTION__,
            ": failed to create a trend line! Error code = ",GetLastError());
      return(false);
     }
//--- 傾向線の傾斜角度を変更する。
//--- 角度の変更時に、2 番目のポイントの座標は、角度の新しい値に応じて自動的に再定義される
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- 線の色を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 線のスタイルを設定
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 線の幅を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- マウスで線を移動させるモードを有効(true)か無効(false)にする
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 線の表示を左に延長するモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- 線の表示を右に延長するモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 傾向線のアンカーポイントを変更する                                |
//+------------------------------------------------------------------+
bool TrendPointChange(const long   chart_ID=0,        // チャート識別子
                      const string name="TrendLine", // 線の名称
                      datetime     time=0,           // アンカーポイントの時間座標
                      double       price=0)          // アンカーポイントの価格座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- 傾向線のアンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 傾向線の傾斜角度を変更する                                       |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool TrendAngleChange(const long   chart_ID=0,        // チャート識別子
                      const string name="TrendLine", // 傾向線の名称
                      const double angle=45)         // 傾向線の傾斜角度
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 傾向線の傾斜角度を変更する
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle))
     {
      Print(__FUNCTION__,
            ": failed to change the line's slope angle! Error code =
",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 傾向線を削除する                                                 |
//+------------------------------------------------------------------+
bool TrendDelete(const long   chart_ID=0,        // チャート識別子
                 const string name="TrendLine") // 線の名称
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 傾向線を削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete a trend line! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 傾向線のアンカーポイントの値をチェックして                       |
//| 空の物には初期値を設定する                                       |
//+------------------------------------------------------------------+
void ChangeTrendEmptyPoints(datetime &time1,double &price1,
                            datetime &time2,double &price2)
  {
//--- 1 番目のポイントの時間が設定されていない場合、現在足になる
   if(!time1)
      time1=TimeCurrent();
//--- 1 番目のポイントの価格が設定されていない場合、売値になる
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 2 番目のポイント(補助点)の座標を設定する
//--- 2 番目のポイントは最初のものから 9 バー左で同じ価格を持つ
   datetime second_point_time[10];
   CopyTime(Symbol(),Period(),time1,10,second_point_time);
   time2=second_point_time[0];
   price2=price1;
  }
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 価格配列サイズ
   int accuracy=1000;
//--- アンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
   double   price[];
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 価格配列に書き込む
//--- チャートの最高値と最安値を見つける
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 線を描画するポイントを定義する
   int d1=InpDate1*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- 傾向線を作成する
   if(!TrendByAngleCreate(0,InpName,0,date[d1],price[p1],InpAngle,InpColor,InpStyle,
```

© 2000-2025, MetaQuotes Ltd.

```
       InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
      {
       return;
      }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- 線を移動して回転させる
//--- ループカウンタ
   int v_steps=accuracy/2;
//--- アンカーポイントを移動してラインの傾斜角度を変更する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p1>1)
         p1-=1;
      //--- ポイントを移動する
      if(!TrendPointChange(0,InpName,date[d1],price[p1]))
         return;
      if(!TrendAngleChange(0,InpName,18*(i+1)))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- チャートから削除する
   TrendDelete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//---
  }
```

© 2000–2025, MetaQuotes Ltd.

OBJ_CYCLES

サイクルライン



注意事項

線の間の距離は、オブジェクトの 2 つのアンカー点の時間座標で設定されています。

例

次のスクリプトは、チャート上でサイクルラインを作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script creates cycle lines on the chart."
#property description "Anchor point coordinates are set in percentage"
#property description "percentage of the chart window size."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string           InpName="Cycles";   // オブジェクト名
input int              InpDate1=10;        // 1 番目のポイントの日付( % )
input int              InpPrice1=45;       // 1 番目のポイントの価格( % )
input int              InpDate2=20;        // 2 番目のポイントの日付( % )
input int              InpPrice2=55;       // 2 番目のポイントの価格( % )
input color            InpColor=clrRed;    // サイクルラインの色
input ENUM_LINE_STYLE  InpStyle=STYLE_DOT; // サイクルラインのスタイル
input int              InpWidth=1;         // サイクルラインの幅
```

© 2000–2025, MetaQuotes Ltd.

```
input bool              InpBack=false;       // 背景オブジェクト
input bool              InpSelection=true;   // 強調表示して移動
input bool              InpHidden=true;      // オブジェクトリストに隠す
input long              InpZOrder=0;         // マウスクリックの優先順位
//+------------------------------------------------------------------+
//| サイクルラインを作成する                                         |
//+------------------------------------------------------------------+
bool CyclesCreate(const long            chart_ID=0,         // チャート識別子
                  const string          name="Cycles",      // オブジェクト名
                  const int             sub_window=0,       // サブウィンドウ番号
                  datetime              time1=0,            // 1 番目のポイントの時間
                  double                price1=0,           // 1 番目のポイントの価格
                  datetime              time2=0,            // 2 番目のポイントの時間
                  double                price2=0,           // 2 番目のポイントの価格
                  const color           clr=clrRed,         // サイクルラインの色
                  const ENUM_LINE_STYLE style=STYLE_SOLID, // サイクルラインのスタイル
                  const int             width=1,            // サイクルラインの幅
                  const bool            back=false,         // 背景で表示する
                  const bool            selection=true,     // 強調表示して移動
                  const bool            hidden=true,        // オブジェクトリストに隠す
                  const long            z_order=0)          // マウスクリックの優先順位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeCyclesEmptyPoints(time1,price1,time2,price2);
//--- エラー値をリセットする
   ResetLastError();
//--- 与えられた座標でサイクルラインを作成する
   if(!ObjectCreate(chart_ID,name,OBJ_CYCLES,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": failed to create cycle lines! Error code = ",GetLastError());
      return(false);
     }
//--- 線の色を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 線の表示スタイルを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 線の幅を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- マウスで線を移動させるモードを有効(true)か無効(false)にする
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| アンカーポイントを移動する                                       |
//+------------------------------------------------------------------+
bool CyclesPointChange(const long   chart_ID=0,     // チャート識別子
                       const string name="Cycles", // オブジェクト名
                       const int    point_index=0, // アンカーポイントのインデックス
                       datetime     time=0,        // アンカーポイントの時間座標
                       double       price=0)       // アンカーポイントの価格座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| サイクルラインを削除する                                         |
//+------------------------------------------------------------------+
bool CyclesDelete(const long   chart_ID=0,     // チャート識別子
                  const string name="Cycles") // オブジェクト名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- サイクルラインを削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete cycle lines! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  サイクルラインのアンカーポイントの値をチェックして                |
//|  空の物には初期値を設定する                                       |
//+------------------------------------------------------------------+
void ChangeCyclesEmptyPoints(datetime &time1,double &price1,
                             datetime &time2,double &price2)
  {
//--- 1 番目のポイントの時間が設定されていない場合、現在足になる
   if(!time1)
      time1=TimeCurrent();
//--- 1 番目のポイントの価格が設定されていない場合、売値になる
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 2 番目のポイントの時間が設定されていない場合、2 番目から 9 バー左に置かれる
   if(!time2)
     {
      //--- 最後の 10 バーのオープン時間を受信するための配列
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- 2 番目のポイントを最初のものから 9 バー左に設定する
      time2=temp[0];
     }
//--- 2 番目のポイントの価格は設定されていない場合は 1 番目のポイントと等しい
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 価格配列サイズ
   int accuracy=1000;
//--- アンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
   double   price[];
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 価格配列に書き込む
//--- チャートの最高値と最安値を見つける
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- サイクルライン描画のポイントを定義する
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- 傾向線を作成する
   if(!CyclesCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- アンカーポイントを移動する
//--- ループカウンタ
   int h_steps=bars/5;
//--- 2 番目のアンカーポイントを移動する
   for(int i=0;i<h_steps;i++)
     {
      //--- 次の値を使用する
      if(d2<bars-1)
         d2+=1;
      //--- ポイントを移動する
      if(!CyclesPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
      // 0.05 秒の遅れ
      Sleep(50);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 1 秒の遅れ
   Sleep(1000);
//--- ループカウンタ
   h_steps=bars/4;
//--- 1 番目のアンカーポイントを移動する
   for(int i=0;i<h_steps;i++)
     {
      //--- 次の値を使用する
      if(d1<bars-1)
         d1+=1;
      //--- ポイントを移動する
      if(!CyclesPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
      // 0.05 秒の遅れ
      Sleep(50);
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- チャートからオブジェクトを削除する
   CyclesDelete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROWED_LINE

矢印



例

次のスクリプトは、チャート上で矢印を作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script draws \"Arrowed line\" graphical object."
#property description "Anchor point coordinates are set in percentage of"
#property description "the chart window size."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string           InpName="ArrowedLine"; // 線の名称
input int              InpDate1=35;           // 1 番目のポイントの日付( % )
input int              InpPrice1=60;          // 1 番目のポイントの価格( % )
input int              InpDate2=65;           // 2 番目のポイントの日付( % )
input int              InpPrice2=40;          // 2 番目のポイントの価格( % )
input color            InpColor=clrRed;       // 線の色
input ENUM_LINE_STYLE InpStyle=STYLE_DASH;    // 線のスタイル
input int              InpWidth=2;            // 線の幅
input bool             InpBack=false;         // 背景の線
input bool             InpSelection=true;     // 強調表示して移動
input bool             InpHidden=true;        // オブジェクトリストに隠す
input long             InpZOrder=0;           // マウスクリックの優先順位
```

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  与えられた座標で矢印を作成する                                  |
//+------------------------------------------------------------------+
bool ArrowedLineCreate(const long            chart_ID=0,          // チャート識別子
                       const string          name="ArrowedLine", // 線の名称
                       const int             sub_window=0,       // サブウィンドウ番号
                       datetime              time1=0,            // 1 番目のポイントの時
間
                       double                price1=0,           // 1 番目のポイントの価
格
                       datetime              time2=0,            // 2 番目のポイントの時
間
                       double                price2=0,           // 2 番目のポイントの値
                       const color           clr=clrRed,         // 線の色
                       const ENUM_LINE_STYLE style=STYLE_SOLID,  // 線のスタイル
                       const int             width=1,            // 線の幅
                       const bool            back=false,         // 背景で表示する
                       const bool            selection=true,     // 強調表示して移動
                       const bool            hidden=true,        // オブジェクトリストに隠す
                       const long            z_order=0)          // マウスクリックの優先順
位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeArrowedLineEmptyPoints(time1,price1,time2,price2);
//--- エラー値をリセットする
   ResetLastError();
//--- 与えられた座標で矢印を作成する
   if(!ObjectCreate(chart_ID,name,OBJ_ARROWED_LINE,sub_window,time1,price1,time2,pri
ce2))
     {
      Print(__FUNCTION__,
            ": failed to create an arrowed line! Error code = ",GetLastError());
      return(false);
     }
//--- 線の色を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 線の表示スタイルを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 線の幅を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- マウスで線を移動させるモードを有効(true)か無効(false)にする
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 矢印のアンカーポイントを移動する                                 |
//+------------------------------------------------------------------+
bool ArrowedLinePointChange(const long   chart_ID=0,          // チャート識別子
                            const string name="ArrowedLine", // 線の名称
                            const int    point_index=0,      // アンカーポイントのインデックス
                            datetime     time=0,             // アンカーポイントの時間座標
                            double       price=0)            // アンカーポイントの価格座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- 線のアンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| この関数はチャートから矢印を削除する                             |
//+------------------------------------------------------------------+
bool ArrowedLineDelete(const long   chart_ID=0,          // チャート識別子
                       const string name="ArrowedLine") // 線の名称
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 矢印を削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to create an arrowed line! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  アンカーポイントの値をチェックして                                |
//|  空の物には初期値を設定する                                        |
//+------------------------------------------------------------------+
void ChangeArrowedLineEmptyPoints(datetime &time1,double &price1,
                                  datetime &time2,double &price2)
  {
//--- 1 番目のポイントの時間が設定されていない場合、現在足になる
   if(!time1)
      time1=TimeCurrent();
//--- 1 番目のポイントの価格が設定されていない場合、売値になる
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 2 番目のポイントの時間が設定されていない場合、2 番目から 9 バー左に置かれる
   if(!time2)
     {
      //--- 最後の 10 バーのオープン時間を受信するための配列
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- 2 番目のポイントを最初のものから 9 バー左に設定する
      time2=temp[0];
     }
//--- 2 番目のポイントの価格は設定されていない場合は 1 番目のポイントと等しい
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 価格配列サイズ
   int accuracy=1000;
//--- アンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
   double   price[];
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 価格配列に書き込む
//--- チャートの最高値と最安値を見つける
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 線を描画するポイントを定義する
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- 矢印を作成する
   if(!ArrowedLineCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- 線のアンカーポイントを移動する
//--- ループカウンタ
   int v_steps=accuracy/5;
//--- 2 番目のアンカーポイントを垂直方向に移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p2<accuracy-1)
         p2+=1;
      //--- ポイントを移動する
      if(!ArrowedLinePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 1 番目のアンカーポイントを垂直方向に移動する
   for(int i=0;i<v_steps;i++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- 次の値を使用する
       if(p1>1)
          p1-=1;
       //--- ポイントを移動する
       if(!ArrowedLinePointChange(0,InpName,0,date[d1],price[p1]))
          return;
       //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
       if(IsStopped())
          return;
       //--- チャートを再描画する
       ChartRedraw();
      }
//--- 半秒の遅れ
   Sleep(500);
//--- ループカウンタ
   int h_steps=bars/2;
//--- 両方のアンカーポイントを同時に水平方向に移動させる
   for(int i=0;i<h_steps;i++)
      {
       //--- 次の値を使用する
       if(d1<bars-1)
          d1+=1;
       if(d2>1)
          d2-=1;
       //--- ポイントをシフトする
       if(!ArrowedLinePointChange(0,InpName,0,date[d1],price[p1]))
          return;
       if(!ArrowedLinePointChange(0,InpName,1,date[d2],price[p2]))
          return;
       //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
       if(IsStopped())
          return;
       //--- チャートを再描画する
       ChartRedraw();
       // 0.03 秒の遅れ
       Sleep(30);
      }
//--- 1 秒の遅れ
   Sleep(1000);
//--- 矢印を削除する
   ArrowedLineDelete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//---
  }
```

© 2000–2025, MetaQuotes Ltd.

OBJ_CHANNEL

等距離チャンネル



注意事項

等距離チャンネルでは、継続のモードを右及び/または左に指定することが出来ます（OBJPROP_RAY_RIGHT 及び OBJPROP_RAY_LEFT プロパティ）。チャンネルを色で塗りつぶすモードも設定することが出来ます。

例

次のスクリプトは、チャート上で等距離チャンネルを作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script draws \"Equidistant Channel\" graphical object."
#property description "Anchor point coordinates are set in percentage of the size of"
#property description "the chart window."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string           InpName="Channel";   // チャンネル名
input int              InpDate1=25;         // 1 番目のポイントの日付( % )
input int              InpPrice1=60;        // 1 番目のポイントの価格( % )
input int              InpDate2=65;         // 2 番目のポイントの日付( % )
input int              InpPrice2=80;        // 2 番目のポイントの価格( % )
input int              InpDate3=30;         // 3 番目のポイントの日付( % )
```

© 2000–2025, MetaQuotes Ltd.

```
input int              InpPrice3=40;         // 3 番目のポイントの価格( % )
input color            InpColor=clrRed;      // チャンネルの色
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // チャンネル線のスタイル
input int              InpWidth=2;           // チャンネル線の幅
input bool             InpBack=false;        // 背景チャンネル
input bool             InpFill=false;        // チャンネルを色で塗りつぶす
input bool             InpSelection=true;    // 強調表示して移動
input bool             InpRayLeft=false;     // チャンネルの左への継続
input bool             InpRayRight=false;    // チャンネルの右への継続
input bool             InpHidden=true;       // オブジェクトリストに隠す
input long             InpZOrder=0;          // マウスクリックの優先順位
//+------------------------------------------------------------------+
//| 与えられた座標で等距離チャンネルを作成する                                |
//+------------------------------------------------------------------+
bool ChannelCreate(const long            chart_ID=0,        // チャート識別子
                   const string          name="Channel",    // チャンネル名
                   const int             sub_window=0,      // サブウィンドウ番号
                   datetime              time1=0,           // 1 番目のポイントの時間
                   double                price1=0,          // 1 番目のポイントの価格
                   datetime              time2=0,           // 2 番目のポイントの時間
                   double                price2=0,          // 2 番目のポイントの価格
                   datetime              time3=0,           // 3 番目のポイントの時間
                   double                price3=0,          // 3 番目のポイントの価格
                   const color           clr=clrRed,        // チャンネルの色
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // チャンネルの線のスタイル
                   const int             width=1,           // チャンネルの線の幅
                   const bool            fill=false,        // チャンネルを色で塗りつぶす
                   const bool            back=false,        // 背景で表示する
                   const bool            selection=true,    // 強調表示して移動
                   const bool            ray_left=false,    // チャンネルの左への継続
                   const bool            ray_right=false,   // チャンネルの右への継続
                   const bool            hidden=true,       // オブジェクトリストに隠す
                   const long            z_order=0)         // マウスクリックの優先順位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeChannelEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- エラー値をリセットする
   ResetLastError();
//--- 与えられた座標でチャンネルを作成する
   if(!ObjectCreate(chart_ID,name,OBJ_CHANNEL,sub_window,time1,price1,time2,price2,t
ime3,price3))
     {
      Print(__FUNCTION__,
            ": failed to create an equidistant channel! Error code =
",GetLastError());
      return(false);
     }
//--- チャンネルの色を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- チャンネルの線のスタイルを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- チャンネルの線の幅を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- チャンネルを色で塗りつぶすモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 強調表示してチャンネルを移動するモードを有効(true)か無効(false)にする。
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- チャンネルの表示を左に延長するモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- チャンネルの表示を右に延長するモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| チャンネルのアンカーポイントを移動する                                |
//+------------------------------------------------------------------+
bool ChannelPointChange(const long   chart_ID=0,      // チャート識別子
                        const string name="Channel", // チャンネル名
                        const int    point_index=0,  // アンカーポイントのインデックス
                        datetime     time=0,         // アンカーポイントの時間座標
                        double       price=0)        // アンカーポイントの価格座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| チャンネルを削除する                                             |
//+------------------------------------------------------------------+
bool ChannelDelete(const long   chart_ID=0,      // チャート識別子
                   const string name="Channel") // チャンネル名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- チャンネルを削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete the channel! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------------+
//| チャンネルのアンカーポイントの値をチェックして                         |
//| 空の物には初期値を設定する                                             |
//+------------------------------------------------------------------------+
void ChangeChannelEmptyPoints(datetime &time1,double &price1,datetime &time2,
                              double &price2,datetime &time3,double &price3)
  {
//--- 2 番目(右)のポイントの時間が設定されていない場合、現在足になる
   if(!time2)
      time2=TimeCurrent();
//--- 2 番目のポイントの価格が設定されていない場合、売値になる
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 1 番目(左)のポイントの時間が設定されていない場合、2 番目から 9 バー左に置かれる
   if(!time1)
     {
      //--- 最後の 10 バーのオープン時間を受信するための配列
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- 1 番目のポイントを 2 番目から 9 バー左に設定する
      time1=temp[0];
     }
//--- 1 番目のポイントの価格が設定されていない場合、2 番目より 300 ポイント高くする
   if(!price1)
      price1=price2+300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- 3 番目のポイントの時間が設定されていない場合、最初のポイントに等しい
   if(!time3)
      time3=time1;
//--- 3 番目のポイントの価格が設定されていない場合、2 番目のポイントに等しい
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!price3)
      price3=price2;
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 価格配列サイズ
   int accuracy=1000;
//--- アンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
   double   price[];
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 価格配列に書き込む
//--- チャートの最高値と最安値を見つける
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- チャンネル描画のポイントを定義する
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 等距離チャンネルを作成する
   if(!ChannelCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],price[p3],InpColor,
      InpStyle
,InpWidth,InpFill,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- チャンネルのアンカーポイントを移動する
//--- ループカウンタ
   int h_steps=bars/6;
//--- 2 番目のアンカーポイントを移動する
   for(int i=0;i<h_steps;i++)
     {
      //--- 次の値を使用する
      if(d2<bars-1)
         d2+=1;
      //--- ポイントを移動する
      if(!ChannelPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
      // 0.05 秒の遅れ
      Sleep(50);
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- 1 番目のアンカーポイントを移動する
   for(int i=0;i<h_steps;i++)
     {
      //--- 次の値を使用する
      if(d1>1)
         d1-=1;
      //--- ポイントを移動する
      if(!ChannelPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
      // 0.05 秒の遅れ
      Sleep(50);
```

© 2000–2025, MetaQuotes Ltd.

```
      }
//--- 1 秒の遅れ
   Sleep(1000);
//--- ループカウンタ
   int v_steps=accuracy/10;
//--- 3 番目のアンカーポイントを移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p3>1)
         p3-=1;
      //--- ポイントを移動する
      if(!ChannelPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- チャートからチャンネルを削除する
   ChannelDelete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//---
  }
```

© 2000–2025, MetaQuotes Ltd.

OBJ_STDDEVCHANNEL

標準偏差チャンネル



注意事項

標準偏差チャンネルでは、継続のモードを右及び/または左に指定することが出来ます（OBJPROP_RAY_RIGHT 及び OBJPROP_RAY_LEFT プロパティ）。チャンネルを色で塗りつぶすモードも設定することが出来ます。

OBJPROP_DEVIATION プロパティがチャンネル偏差の値を変更するために使用されます。

例

次のスクリプトは、チャート上で標準偏差チャンネルを作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script draws \"Standard Deviation Channel\" graphical object."
#property description "Anchor point coordinates are set in percentage of the size of"
#property description "the chart window."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string           InpName="StdDevChannel";    // チャンネル名
input int              InpDate1=10;                // 1 番目のポイントの日付( % )
input int              InpDate2=40;                // 2 番目のポイントの日付( % )
```

© 2000–2025, MetaQuotes Ltd.

```
input double           InpDeviation=1.0;           // 偏差値
input color            InpColor=clrRed;            // チャンネルの色
input ENUM_LINE_STYLE  InpStyle=STYLE_DASHDOTDOT; // チャンネルの線のスタイル
input int              InpWidth=2;                 // チャンネルの線の幅
input bool             InpFill=false;              // チャンネルを色で塗りつぶす
input bool             InpBack=false;              // 背景チャンネル
input bool             InpSelection=true;          // 強調表示して移動
input bool             InpRayLeft=false;           // チャンネルの左への継続
input bool             InpRayRight=false;          // チャンネルの右への継続
input bool             InpHidden=true;             // オブジェクトリストに隠す
input long             InpZOrder=0;                // マウスクリックの優先順位
//+------------------------------------------------------------------+
//|  与えられた座標で標準偏差チャンネルを作成する                                  |
//+------------------------------------------------------------------+
bool StdDevChannelCreate(const long            chart_ID=0,        // チャート識別子
                         const string          name="Channel",    // チャンネル名
                         const int             sub_window=0,      // サブウィンドウ番号
                         datetime              time1=0,           // 1 番目のポイントの時間
                         datetime              time2=0,           // 2 番目のポイントの時間
                         const double          deviation=1.0,     // 偏差値
                         const color           clr=clrRed,        // チャンネルの色
                         const ENUM_LINE_STYLE style=STYLE_SOLID, // チャンネルの線のスタイル
                         const int             width=1,           // チャンネルの線の幅
                         const bool            fill=false,        // チャンネルを色で塗りつぶす
                         const bool            back=false,        // 背景で表示する
                         const bool            selection=true,    // 強調表示して移動
                         const bool            ray_left=false,    // チャンネルの左への継続
                         const bool            ray_right=false,   // チャンネルの右への継続
                         const bool            hidden=true,       // オブジェクトリストに隠す
                         const long            z_order=0)         // マウスクリックの優先順位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeChannelEmptyPoints(time1,time2);
//--- エラー値をリセットする
   ResetLastError();
//--- 与えられた座標でチャンネルを作成する
   if(!ObjectCreate(chart_ID,name,OBJ_STDDEVCHANNEL,sub_window,time1,0,time2,0))
     {
      Print(__FUNCTION__,
            ": failed to create standard deviation channel! Error code = ",GetLastError());
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- チャンネル幅に影響を与える偏差値を設定する
   ObjectSetDouble(chart_ID,name,OBJPROP_DEVIATION,deviation);
//--- チャンネルの色を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- チャンネルの線のスタイルを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- チャンネルの線の幅を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- チャンネルを色で塗りつぶすモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 強調表示してチャンネルを移動するモードを有効(true)か無効(false)にする。
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- チャンネルの表示を左に延長するモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- チャンネルの表示を右に延長するモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| チャンネルのアンカーポイントを移動する                                |
//+------------------------------------------------------------------+
bool StdDevChannelPointChange(const long    chart_ID=0,      // チャート識別子
                              const string name="Channel", // チャンネル名
                              const int     point_index=0,  // アンカーポイントのインデックス
                              datetime      time=0)         // アンカーポイントの時間座標
  {
//--- ポイントの時間が設定されていない場合、ポイントを現在足に移動する
   if(!time)
      time=TimeCurrent();
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,point_index,time,0))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| チャンネルの偏差を変更する                                       |
//+------------------------------------------------------------------+
bool StdDevChannelDeviationChange(const long   chart_ID=0,      // チャート識別子
                                  const string name="Channel", // チャンネル名
                                  const double deviation=1.0)  // 偏差値
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 傾向線の傾斜角度を変更する
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_DEVIATION,deviation))
     {
      Print(__FUNCTION__,
            ": failed to change channel deviation! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| チャンネルを削除する                                             |
//+------------------------------------------------------------------+
bool StdDevChannelDelete(const long   chart_ID=0,      // チャート識別子
                         const string name="Channel") // チャンネル名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- チャンネルを削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete the channel! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------------+
//| チャンネルのアンカーポイントの値をチェックして                         |
//| 空の物には初期値を設定する                                             |
//+------------------------------------------------------------------------+
void ChangeChannelEmptyPoints(datetime &time1,datetime &time2)
  {
//--- 2 番目のポイントの時間が設定されていない場合、現在足になる
   if(!time2)
      time2=TimeCurrent();
```

© 2000–2025, MetaQuotes Ltd.

```
//--- 1 番目のポイントの時間が設定されていない場合、2 番目から 9 バー左に置かれる
   if(!time1)
     {
      //--- 最後の 10 バーのオープン時間を受信するための配列
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- 1 番目のポイントを 2 番目から 9 バー左に設定する
      time1=temp[0];
     }
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
   if(InpDate1<0 || InpDate1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 価格配列サイズ
   int accuracy=1000;
//--- アンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
   double   price[];
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 価格配列に書き込む
//--- チャートの最高値と最安値を見つける
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- チャンネル描画のポイントを定義する
```

© 2000-2025, MetaQuotes Ltd.

```
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
//--- 標準偏差チャンネルを作成する
   if(!StdDevChannelCreate(0,InpName,0,date[d1],date[d2],InpDeviation
,InpColor,InpStyle,
      InpWidth
,InpFill,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- チャンネルを右に水平に移動して伸ばす
//--- ループカウンタ
   int h_steps=bars/2;
//--- チャンネルを移動する
   for(int i=0;i<h_steps;i++)
     {
      //--- 次の値を使用する
      if(d1<bars-1)
         d1+=1;
      if(d2<bars-1)
         d2+=1;
      //--- アンカーポイントを移動する
      if(!StdDevChannelPointChange(0,InpName,0,date[d1]))
         return;
      if(!StdDevChannelPointChange(0,InpName,1,date[d2]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
      // 0.05 秒の遅れ
      Sleep(50);
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- ループカウンタ
   double v_steps=InpDeviation*2;
//--- チャンネルを拡張する
   for(double i=InpDeviation;i<v_steps;i+=10.0/accuracy)
     {
      if(!StdDevChannelDeviationChange(0,InpName,i))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- チャートからチャンネルを削除する
   StdDevChannelDelete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//---
  }
```

© 2000–2025, MetaQuotes Ltd.

OBJ_REGRESSION

線形回帰チャンネル



注意事項

線形回帰チャンネルでは、継続のモードを右及び/または左に指定することが出来ます（OBJPROP_RAY_RIGHT 及び OBJPROP_RAY_LEFT プロパティ）。チャンネルを色で塗りつぶすモードも設定することが出来ます。

例

次のスクリプトは、チャート上で線形回帰チャンネルを作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script draws \"Linear Regression Channel\" graphical object."
#property description "Anchor point coordinates are set in percentage of the size of"
#property description "the chart window."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string           InpName="Regression"; // チャンネル名
input int              InpDate1=10;          // 1 番目のポイントの日付( % )
input int              InpDate2=40;          // 2 番目のポイントの日付( % )
input color            InpColor=clrRed;      // チャンネルの色
input ENUM_LINE_STYLE InpStyle=STYLE_DASH;  // チャンネルの線のスタイル
input int              InpWidth=2;           // チャンネルの線の幅
```

© 2000–2025, MetaQuotes Ltd.

```
input bool            InpFill=false;        // チャンネルを色で塗りつぶす
input bool            InpBack=false;        // 背景チャンネル
input bool            InpSelection=true;    // 強調表示して移動
input bool            InpRayLeft=false;     // チャンネルの左への継続
input bool            InpRayRight=false;    // チャンネルの右への継続
input bool            InpHidden=true;       // オブジェクトリストに隠す
input long            InpZOrder=0;          // マウスクリックの優先順位
//+------------------------------------------------------------------+
//| 与えられた座標で直線回帰チャンネルを作成する                                |
//+------------------------------------------------------------------+
bool RegressionCreate(const long            chart_ID=0,         // チャート識別子
                      const string          name="Regression",  // チャンネル名
                      const int             sub_window=0,       // サブウィンドウ番号
                      datetime              time1=0,            // 1 番目のポイントの時間
                      datetime              time2=0,            // 2 番目のポイントの時間
                      const color           clr=clrRed,         // チャンネルの色
                      const ENUM_LINE_STYLE style=STYLE_SOLID,  // チャンネルの線のスタイル
                      const int             width=1,            // チャンネルの線の幅
                      const bool            fill=false,         // チャンネルを色で塗りつぶす
                      const bool            back=false,         // 背景で表示する
                      const bool            selection=true,     // 強調表示して移動
                      const bool            ray_left=false,     // チャンネルの左への継続
                      const bool            ray_right=false,    // チャンネルの右への継続
                      const bool            hidden=true,        // オブジェクトリストに隠す
                      const long            z_order=0)          // マウスクリックの優先順位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeRegressionEmptyPoints(time1,time2);
//--- エラー値をリセットする
   ResetLastError();
//--- 与えられた座標でチャンネルを作成する
   if(!ObjectCreate(chart_ID,name,OBJ_REGRESSION,sub_window,time1,0,time2,0))
     {
      Print(__FUNCTION__,
            ": failed to create linear regression channel! Error code =
",GetLastError());
      return(false);
     }
//--- チャンネルの色を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- チャンネルの線のスタイルを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- チャンネルの線の幅を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- チャンネルを色で塗りつぶすモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 強調表示してチャンネルを移動するモードを有効(true)か無効(false)にする。
```

© 2000-2025, MetaQuotes Ltd.

```
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- チャンネルの表示を左に延長するモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- チャンネルの表示を右に延長するモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| チャンネルのアンカーポイントを移動する                                |
//+------------------------------------------------------------------+
bool RegressionPointChange(const long   chart_ID=0,      // チャート識別子
                           const string name="Channel", // チャンネル
                           const int    point_index=0,  // アンカーポイントのインデックス
                           datetime     time=0)         // アンカーポイントの時間座標
  {
//--- ポイントの時間が設定されていない場合、ポイントを現在足に移動する
   if(!time)
      time=TimeCurrent();
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,point_index,time,0))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| チャンネルを削除する                                              |
//+------------------------------------------------------------------+
bool RegressionDelete(const long   chart_ID=0,      // チャート識別子
                      const string name="Channel") // チャンネル名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- チャンネルを削除する
   if(!ObjectDelete(chart_ID,name))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print(__FUNCTION__,
             ": failed to delete the channel! Error code = ",GetLastError());
       return(false);
      }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------------------+
//|  チャンネルのアンカーポイントの値をチェックして                                |
//|  空の物には初期値を設定する                                                    |
//+------------------------------------------------------------------------------+
void ChangeRegressionEmptyPoints(datetime &time1,datetime &time2)
  {
//--- 2 番目のポイントの時間が設定されていない場合、現在足になる
   if(!time2)
      time2=TimeCurrent();
//--- 1 番目のポイントの時間が設定されていない場合、2 番目から 9 バー左に置かれる
   if(!time1)
     {
      //--- 最後の 10 バーのオープン時間を受信するための配列
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- 1 番目のポイントを 2 番目から 9 バー左に設定する
      time1=temp[0];
     }
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
   if(InpDate1<0 || InpDate1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 価格配列サイズ
   int accuracy=1000;
//--- アンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
   double   price[];
//--- メモリ割り当て
   ArrayResize(date,bars);
```

© 2000-2025, MetaQuotes Ltd.

```
   ArrayResize(price,accuracy);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 価格配列に書き込む
//--- チャートの最高値と最安値を見つける
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- チャンネル描画のポイントを定義する
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
//--- 直線回帰チャンネルを作成する
   if(!RegressionCreate(0,InpName,0,date[d1],date[d2],InpColor,InpStyle,InpWidth,
      InpFill,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- チャンネルを右に水平に移動する
//--- ループカウンタ
   int h_steps=bars/2;
//--- チャンネルを移動する
   for(int i=0;i<h_steps;i++)
     {
      //--- 次の値を使用する
      if(d1<bars-1)
         d1+=1;
      if(d2<bars-1)
         d2+=1;
      //--- アンカーポイントを移動する
      if(!RegressionPointChange(0,InpName,0,date[d1]))
         return;
      if(!RegressionPointChange(0,InpName,1,date[d2]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
```

© 2000-2025, MetaQuotes Ltd.

```
      // 0.05 秒の遅れ
      Sleep(50);
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- チャートからチャンネルを削除する
   RegressionDelete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_PITCHFORK

アンドリューズピッチフォーク



注意事項

アンドリューズピッチフォークでは、継続のモードを右及び/または左に指定することが出来ます（OBJPROP_RAY_RIGHT 及び OBJPROP_RAY_LEFT プロパティ）。

ラインレベル、その値と色も指定することが出来ます。

例

次のスクリプトは、チャート上でアンドリューズピッチフォークを作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script draws \"Andrews' Pitchfork\" graphical object."
#property description "Anchor point coordinates are set in percentage of"
#property description "the chart window size."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string           InpName="Pitchfork";  // ピッチフォーク名
input int              InpDate1=14;          // 1 番目のポイントの日付( % )
input int              InpPrice1=40;         // 1 番目のポイントの価格( % )
input int              InpDate2=18;          // 2 番目のポイントの日付( % )
input int              InpPrice2=50;         // 2 番目のポイントの価格( % )
input int              InpDate3=18;          // 3 番目のポイントの日付( % )
input int              InpPrice3=30;         // 3 番目のポイントの価格( % )
```

© 2000–2025, MetaQuotes Ltd.

```
input color           InpColor=clrRed;       // ピッチフォークの色
input ENUM_LINE_STYLE InpStyle=STYLE_SOLID; // ピッチフォークの線のスタイル
input int             InpWidth=1;            // ピッチフォークの線の幅
input bool            InpBack=false;         // 背景のピッチフォーク
input bool            InpSelection=true;     // 強調表示して移動
input bool            InpRayLeft=false;      // ピッチフォークの左への継続
input bool            InpRayRight=false;     // ピッチフォークの右への継続
input bool            InpHidden=true;        // オブジェクトリストに隠す
input long            InpZOrder=0;           // マウスクリックの優先順位
//+------------------------------------------------------------------+
//| 与えられた座標でアンドリューズピッチフォークを作成する                |
//+------------------------------------------------------------------+
bool PitchforkCreate(const long            chart_ID=0,        // チャート識別子
                     const string          name="Pitchfork",  // ピッチフォーク名
                     const int             sub_window=0,      // サブウィンドウ番号
                     datetime              time1=0,           // 1 番目のポイントの時間
                     double                price1=0,          // 1 番目のポイントの価格
                     datetime              time2=0,           // 2 番目のポイントの時間
                     double                price2=0,          // 2 番目のポイントの価格
                     datetime              time3=0,           // 3 番目のポイントの時間
                     double                price3=0,          // 3 番目のポイントの価格
                     const color           clr=clrRed,        // ピッチフォークの線の色
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // ピッチフォークの線のスタイル
                     const int             width=1,           // ピッチフォークの線の幅
                     const bool            back=false,        // 背景で表示する
                     const bool            selection=true,    // 強調表示して移動
                     const bool            ray_left=false,    // ピッチフォークの左への継続
                     const bool            ray_right=false,   // ピッチフォークの右への継続
                     const bool            hidden=true,       // オブジェクトリストに隠す
                     const long            z_order=0)         // マウスクリックの優先順位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeChannelEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- エラー値をリセットする
   ResetLastError();
//--- 与えられた座標でアンドリューズピッチフォークを作成する
   if(!ObjectCreate(chart_ID,name,OBJ_PITCHFORK,sub_window,time1,price1,time2,price2
,time3,price3))
     {
      Print(__FUNCTION__,
            ": failed to create \"Andrews' Pitchfork\"! Error code =
",GetLastError());
      return(false);
     }
//--- 色を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 線のスタイルを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 線の幅を設定する
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 強調表示してピッチフォークを移動するモードを有効(true)か無効(false)にする
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ピッチフォークの表示を左に延長するモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- ピッチフォークの表示を右に延長するモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| アンドリューズピッチフォークのレベルとパラメータを設定する                          |
//+------------------------------------------------------------------+
bool PitchforkLevelsSet(int             levels,           // レベル線の数
                        double          &values[],        // レベル線の値
                        color           &colors[],        // レベル線の色
                        ENUM_LINE_STYLE &styles[],        // レベル線のスタイル
                        int             &widths[],        // レベル線の幅
                        const long      chart_ID=0,       // チャート識別子
                        const string    name="Pitchfork") // ピッチフォークの名称
  {
//--- 配列のサイズをチェックする
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": array length does not correspond to the number of
levels, error!");
      return(false);
     }
//--- レベル数を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- ループ内でレベルプロパティを設定する
   for(int i=0;i<levels;i++)
     {
      //--- レベル値
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- レベルの色
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- レベルのスタイル
```

© 2000-2025, MetaQuotes Ltd.

```
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- レベルの幅
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- レベルの説明
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i]
,1));
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| ピッチフォークのアンカーポイントを移動する                                |
//+------------------------------------------------------------------+
bool PitchforkPointChange(const long   chart_ID=0,       // チャート識別子
                          const string name="Pitchfork", // チャンネル名
                          const int    point_index=0,    // アンカーポイントのインデックス
                          datetime     time=0,           // アンカーポイントの時間座標
                          double       price=0)          // アンカーポイントの価格座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| アンドリューズピッチフォークを削除する                                  |
//+------------------------------------------------------------------+
bool PitchforkDelete(const long   chart_ID=0,       // チャート識別子
                     const string name="Pitchfork") // チャンネル名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- チャンネルを削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
```

© 2000-2025, MetaQuotes Ltd.

```
            ": failed to delete \"Andrews' Pitchfork\"! Error code =
",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------------+
//| アンドリューズピッチフォークのアンカーポイントの値をチェックして                          |
//| 空の物には初期値を設定する                                                |
//+------------------------------------------------------------------------+
void ChangeChannelEmptyPoints(datetime &time1,double &price1,datetime &time2,
                              double &price2,datetime &time3,double &price3)
  {
//--- 2 番目(右上)のポイントの時間が設定されていない場合、現在足になる
   if(!time2)
      time2=TimeCurrent();
//--- 2 番目のポイントの価格が設定されていない場合、売値になる
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 1 番目(左)のポイントの時間が設定されていない場合、2 番目から 9 バー左に置かれる
   if(!time1)
     {
      //--- 最後の 10 バーのオープン時間を受信するための配列
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- 1 番目のポイントを 2 番目から 9 バー左に設定する
      time1=temp[0];
     }
//--- 1 番目のポイントの価格が設定されていない場合、2 番目より 200 ポイント低くする
   if(!price1)
      price1=price2-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- 3 番目のポイントの時間が設定されていない場合、2 番目のポイントに等しい
   if(!time3)
      time3=time2;
//--- 3 番目のポイントの価格が設定されていない場合、1 番目より 200 ポイント低くする
   if(!price3)
      price3=price1-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 価格配列サイズ
   int accuracy=1000;
//--- アンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
   double   price[];
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 価格配列に書き込む
//--- チャートの最高値と最安値を見つける
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- アンドリューズピッチフォーク描画のポイントを定義する
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- ピッチフォークを作成する
   if(!PitchforkCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],pr
ice[p3],
      InpColor
,InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- ピッチフォークのアンカーポイントを移動する
//--- ループカウンタ
```

© 2000-2025, MetaQuotes Ltd.

```
   int v_steps=accuracy/10;
//--- 1 番目のアンカーポイントを移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p1>1)
         p1-=1;
      //--- ポイントを移動する
      if(!PitchforkPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- ループカウンタ
   int h_steps=bars/8;
//--- 3 番目のアンカーポイントを移動する
   for(int i=0;i<h_steps;i++)
     {
      //--- 次の値を使用する
      if(d3<bars-1)
         d3+=1;
      //--- ポイントを移動する
      if(!PitchforkPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
      //--- チャートを再描画する
      ChartRedraw();
      // 0.05 秒の遅れ
      Sleep(50);
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- ループカウンタ
   v_steps=accuracy/10;
//--- 2 番目のアンカーポイントを移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p2>1)
         p2-=1;
```

© 2000–2025, MetaQuotes Ltd.

```
      //--- ポイントを移動する
      if(!PitchforkPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- チャートからピッチフォークを削除する
   PitchforkDelete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_GANNLINE

ギャンライン



注意事項

ギャンラインでは継続のモードを右及び/または左に指定することが出来ます（OBJPROP_RAY_RIGHT 及び OBJPROP_RAY_LEFT プロパティ）。

スケールを持ったギャン角度及び第2アンカー点の座標の両方が、線の傾きを設定するために使用出来ます。

例

次のスクリプトは、チャート上でギャンラインを作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script draws \"Gann Line\" graphical object."
#property description "Anchor point coordinates are set in percentage of"
#property description "the chart window size."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string           InpName="GannLine";         // 線の名称
input int              InpDate1=20;                // 1 番目のポイントの日付( % )
input int              InpPrice1=75;               // 1 番目のポイントの価格( % )
input int              InpDate2=80;                // 2 番目のポイントの日付( % )
input double           InpAngle=0.0;               // ギャン角度
```

© 2000–2025, MetaQuotes Ltd.

```
input double           InpScale=1.0;               // 規模
input color            InpColor=clrRed;            // 線の色
input ENUM_LINE_STYLE  InpStyle=STYLE_DASHDOTDOT;  // 線のスタイル
input int              InpWidth=2;                 // 線の幅
input bool             InpBack=false;              // 背景の線
input bool             InpSelection=true;          // 強調表示して移動
input bool             InpRayLeft=false;           // 線の左への継続
input bool             InpRayRight=true;           // 線の右への継続
input bool             InpHidden=true;             // オブジェクトリストに隠す
input long             InpZOrder=0;                // マウスクリックの優先順位
//+------------------------------------------------------------------+
//|  与えられた座標、角度とスケールでギャンラインを作成する                |
//+------------------------------------------------------------------+
bool GannLineCreate(const long             chart_ID=0,        // チャート識別子
                    const string           name="GannLine",   // 線の名称
                    const int              sub_window=0,      // サブウィンドウ番号
                    datetime               time1=0,           // 1 番目のポイントの時間
                    double                 price1=0,          // 1 番目のポイントの価格
                    datetime               time2=0,           // 2 番目のポイントの時間
                    const double           angle=1.0,         // ギャン角度
                    const double           scale=1.0,         // 規模
                    const color            clr=clrRed,        // 線の色
                    const ENUM_LINE_STYLE  style=STYLE_SOLID, // 線のスタイル
                    const int              width=1,           // 線の幅
                    const bool             back=false,        // 背景で表示する
                    const bool             selection=true,    // 強調表示して移動
                    const bool             ray_left=false,    // 線の左への継続
                    const bool             ray_right=true,    // 線の右への継続
                    const bool             hidden=true,       // オブジェクトリストに隠す
                    const long             z_order=0)         // マウスクリックの優先順位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeGannLineEmptyPoints(time1,price1,time2);
//--- エラー値をリセットする
   ResetLastError();
//--- 与えられた座標でギャンラインを作成する
//--- 第2アンカー点の正しい座標は
//--- ギャン角度及び/または規模の変更後に自動的に再定義される
   if(!ObjectCreate(chart_ID,name,OBJ_GANNLINE,sub_window,time1,price1,time2,0))
     {
      Print(__FUNCTION__,
            ": failed to create \"Gann Line\"! Error code = ",GetLastError());
      return(false);
     }
//--- ギャン角度を変更する
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- スケール(バーあたりのピップ数)を変更する
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- 線の色を設定する
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 線の表示スタイルを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 線の幅を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 強調表示して線を移動するモードを有効(true)か無効(false)にする
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 線の表示を左に延長するモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- 線の表示を右に延長するモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| ギャンラインのアンカーポイントを移動する                                 |
//+------------------------------------------------------------------+
bool GannLinePointChange(const long   chart_ID=0,       // チャート識別子
                         const string name="GannLine", // 線の名称
                         const int    point_index=0,   // アンカーポイントのインデックス
                         datetime     time=0,          // アンカーポイントの時間座標
                         double       price=0)         // アンカーポイントの価格座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- 線のアンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//|  ギャン角度を変更する                                              |
//+------------------------------------------------------------------+
bool GannLineAngleChange(const long   chart_ID=0,       // チャート識別子
                         const string name="GannLine", // 線の名称
                         const double angle=1.0)       // ギャン角度
  {
//--- エラー値をリセットする
   ResetLastError();
//--- ギャン角度を変更する
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle))
     {
      Print(__FUNCTION__,
            ": failed to change Gann angle! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  ギャンラインの規模を変更する                                      |
//+------------------------------------------------------------------+
bool GannLineScaleChange(const long   chart_ID=0,       // チャート識別子
                         const string name="GannLine", // 線の名称
                         const double scale=1.0)       // 規模
  {
//--- エラー値をリセットする
   ResetLastError();
//--- スケール(バーあたりのピップ数)を変更する
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale))
     {
      Print(__FUNCTION__,
            ": failed to change the scale! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  この関数はチャートからギャンラインを削除する                      |
//+------------------------------------------------------------------+
bool GannLineDelete(const long   chart_ID=0,       // チャート識別子
                    const string name="GannLine") // 線の名称
  {
//--- エラー値をリセットする
   ResetLastError();
//--- ギャンラインを削除する
   if(!ObjectDelete(chart_ID,name))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print(__FUNCTION__,
             ": failed to delete \"Gann Line\"! Error code = ",GetLastError());
       return(false);
      }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  ギャンラインのアンカーポイントの値をチェックして                |
//|  空の物には初期値を設定する                                      |
//+------------------------------------------------------------------+
void ChangeGannLineEmptyPoints(datetime &time1,double &price1,datetime &time2)
  {
//--- 2 番目のポイントの時間が設定されていない場合、現在足になる
   if(!time2)
      time2=TimeCurrent();
//--- 1 番目のポイントの時間が設定されていない場合、2 番目から 9 バー左に置かれる
   if(!time1)
     {
      //--- 最後の 10 バーのオープン時間を受信するための配列
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- 1 番目のポイントを 2 番目から 9 バー左に設定する
      time1=temp[0];
     }
//--- 1 番目のポイントの価格が設定されていない場合、売値になる
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 価格配列サイズ
   int accuracy=1000;
//--- アンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
```

© 2000-2025, MetaQuotes Ltd.

```
   double   price[];
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 価格配列に書き込む
//--- チャートの最高値と最安値を見つける
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- ギャンライン描画のポイントを定義する
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- ギャンラインを作成する
   if(!GannLineCreate(0,InpName,0,date[d1],price[p1],date[d2],InpAngle
,InpScale,InpColor,
      InpStyle
,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- 線のアンカーポイントを移動して角度を変更する
//--- ループカウンタ
   int v_steps=accuracy/2;
//--- 1 番目のアンカーポイントを垂直方向に移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p1>1)
         p1-=1;
      //--- ポイントを移動する
      if(!GannLinePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 半秒の遅れ
   Sleep(500);
//--- ギャン角度の現在値を定義する
//--- (最初のアンカーポイントを移動した後に変更された)
   double curr_angle;
   if(!ObjectGetDouble(0,InpName,OBJPROP_ANGLE,0,curr_angle))
      return;
//--- ループカウンタ
   v_steps=accuracy/8;
//--- ギャン角度を変更する
   for(int i=0;i<v_steps;i++)
     {
      if(!GannLineAngleChange(0,InpName,curr_angle-0.05*i))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- チャートから線を削除する
   GannLineDelete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_GANNFAN

ギャンファン



注意事項

ギャンファンでは、トレンドの種類を ENUM_GANN_DIRECTION 列挙で指定することが可能です。スケール値（OBJPROP_SCALE）の調節によって、ファンの線の傾斜角度を変更することが出来ます。

例

次のスクリプトは、チャート上でギャンファンを作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script draws \"Gann Fan\" graphical object."
#property description "Anchor point coordinates are set in percentage of"
#property description "the chart window size."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string          InpName="GannFan";          // ファンの名称
input int             InpDate1=15;                // 1 番目のポイントの日付( % )
input int             InpPrice1=25;               // 1 番目のポイントの価格( % )
input int             InpDate2=85;                // 2 番目のポイントの日付( % )
input double          InpScale=2.0;               // 規模
input bool            InpDirection=false;         // トレンドの方向
input color           InpColor=clrRed;            // ファンの色
```

© 2000–2025, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // ファンの線のスタイル
input int             InpWidth=1;                // ファンの線の幅
input bool            InpBack=false;             // 背景のファン
input bool            InpSelection=true;         // 強調表示して移動
input bool            InpHidden=true;            // オブジェクトリストに隠す
input long            InpZOrder=0;               // マウスクリックの優先順位
//+------------------------------------------------------------------+
//| ギャンファンを作成する                                           |
//+------------------------------------------------------------------+
bool GannFanCreate(const long            chart_ID=0,        // チャート識別子
                   const string          name="GannFan",    // ファンの名称
                   const int             sub_window=0,      // サブウィンドウ番号
                   datetime              time1=0,           // 1 番目のポイントの時間
                   double                price1=0,          // 1 番目のポイントの価格
                   datetime              time2=0,           // 2 番目のポイントの時間
                   const double          scale=1.0,         // 規模
                   const bool            direction=true,    // トレンドの方向
                   const color           clr=clrRed,        // ファンの色
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // ファン線のスタイル
                   const int             width=1,           // ファン線の幅
                   const bool            back=false,        // 背景で表示する
                   const bool            selection=true,    // 強調表示して移動
                   const bool            hidden=true,       // オブジェクトリストに隠す
                   const long            z_order=0)         // マウスクリックの優先順位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeGannFanEmptyPoints(time1,price1,time2);
//--- エラー値をリセットする
   ResetLastError();
//--- 与えられた座標でギャンファンを作成する
   if(!ObjectCreate(chart_ID,name,OBJ_GANNFAN,sub_window,time1,price1,time2,0))
     {
      Print(__FUNCTION__,
            ": failed to create \"Gann Fan\"! Error code = ",GetLastError());
      return(false);
     }
//--- スケール(バーあたりのピップ数)を変更する
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- ギャンファンのトレンド方向を変更する(true - 下降, false - 上行)
   ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction);
//--- ファンの色を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- ファンの線の表示スタイルを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- ファンの線の幅を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 強調表示してファンを移動するモードを有効(true)か無効(false)にする
```

© 2000-2025, MetaQuotes Ltd.

```
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| ギャンファンのアンカーポイントを移動する                                  |
//+------------------------------------------------------------------+
bool GannFanPointChange(const long   chart_ID=0,      // チャート識別子
                        const string name="GannFan", // ファンの名称
                        const int    point_index=0,  // アンカーポイントのインデックス
                        datetime     time=0,         // アンカーポイントの時間座標
                        double       price=0)        // アンカーポイントの価格座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- ファンのアンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| ギャンファンの規模を変更する                                          |
//+------------------------------------------------------------------+
bool GannFanScaleChange(const long   chart_ID=0,      // チャート識別子
                        const string name="GannFan", // ファンの名称
                        const double scale=1.0)      // 規模
  {
//--- エラー値をリセットする
   ResetLastError();
//--- スケール(バーあたりのピップ数)を変更する
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print(__FUNCTION__,
             ": failed to change the scale! Error code = ",GetLastError());
       return(false);
      }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| ギャンファンのトレンド方向を変更する                                |
//+------------------------------------------------------------------+
bool GannFanDirectionChange(const long   chart_ID=0,      // チャート識別子
                            const string name="GannFan", // ファンの名称
                            const bool   direction=true) // トレンドの方向
  {
//--- エラー値をリセットする
   ResetLastError();
//--- ギャンファンのトレンド方向を変更する
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction))
     {
      Print(__FUNCTION__,
            ": failed to change trend direction! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| この関数はチャートからギャンファンを削除する                       |
//+------------------------------------------------------------------+
bool GannFanDelete(const long   chart_ID=0,      // チャート識別子
                   const string name="GannFan") // ファンの名称
  {
//--- エラー値をリセットする
   ResetLastError();
//--- ギャンファンを削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete \"Gann Fan\"! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
// | ギャンファンのアンカーポイントの値をチェックして                  |
//| 空の物には初期値を設定する                                        |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void ChangeGannFanEmptyPoints(datetime &time1,double &price1,datetime &time2)
  {
//--- 2 番目のポイントの時間が設定されていない場合、現在足になる
   if(!time2)
      time2=TimeCurrent();
//--- 1 番目のポイントの時間が設定されていない場合、2 番目から 9 バー左に置かれる
   if(!time1)
     {
      //--- 最後の 10 バーのオープン時間を受信するための配列
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- 1 番目のポイントを 2 番目から 9 バー左に設定する
      time1=temp[0];
     }
//--- 1 番目のポイントの価格が設定されていない場合、売値になる
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 価格配列サイズ
   int accuracy=1000;
//--- アンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
   double   price[];
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 価格配列に書き込む
```

© 2000-2025, MetaQuotes Ltd.

```
//--- チャートの最高値と最安値を見つける
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- ギャンファン描画のポイントを定義する
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- ギャンファンを作成する
   if(!GannFanCreate(0,InpName,0,date[d1],price[p1],date[d2],InpScale,InpDirection,
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- ファンのアンカーポイントを移動する
//--- ループカウンタ
   int v_steps=accuracy/2;
//--- 1 番目のアンカーポイントを垂直方向に移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p1<accuracy-1)
         p1+=1;
      //--- ポイントを移動する
      if(!GannFanPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- ファンのトレンド方向を下降に変更する
   GannFanDirectionChange(0,InpName,true);
//--- チャートを再描画する
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//--- チャートからファンを削除する
   GannFanDelete(0,InpName);
   ChartRedraw();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 1 秒の遅れ
   Sleep(1000);
//---
  }
```

© 2000–2025, MetaQuotes Ltd.

OBJ_GANNGRID

ギャングリッド



注意事項

ギャングリッドでは、トレンドの種類を ENUM_GANN_DIRECTION 列挙で指定することが可能です。スケール値（OBJPROP_SCALE）の調節によってグリッド線の傾斜角度を変更することが出来ます。

例

次のスクリプトは、チャート上でギャングリッドを作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script draws \"Gann Grid\" graphical object."
#property description "Anchor point coordinates of the grid are set in percentage of"
#property description "the chart window size."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string           InpName="GannGrid";        // グリッド名
input int              InpDate1=15;               // 1 番目のポイントの日付( % )
input int              InpPrice1=25;              // 1 番目のポイントの価格( % )
input int              InpDate2=35;               // 2 番目のポイントの日付( % )
input double           InpScale=3.0;              // 規模
input bool             InpDirection=false;        // トレンドの方向
input color            InpColor=clrRed;           // グリッドの色
```

© 2000–2025, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // グリッド線のスタイル
input int             InpWidth=1;                // ファンの線の幅
input bool            InpBack=false;             // 背景のグリッド
input bool            InpSelection=true;         // 強調表示して移動
input bool            InpHidden=true;            // オブジェクトリストに隠す
input long            InpZOrder=0;               // マウスクリックの優先順位
//+------------------------------------------------------------------+
//| ギャングリッドを作成する                                         |
//+------------------------------------------------------------------+
bool GannGridCreate(const long            chart_ID=0,        // チャート識別子
                    const string          name="GannGrid",   // グリッド名
                    const int             sub_window=0,      // サブウィンドウ番号
                    datetime              time1=0,           // 1 番目のポイントの時間
                    double                price1=0,          // 1 番目のポイントの価格
                    datetime              time2=0,           // 2 番目のポイントの時間
                    const double          scale=1.0,         // 規模
                    const bool            direction=true,    // トレンドの方向
                    const color           clr=clrRed,        // グリッドの色
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // グリッド線のスタイル
                    const int             width=1,           // グリッド線の幅
                    const bool            back=false,        // 背景で表示する
                    const bool            selection=true,    // 強調表示して移動
                    const bool            hidden=true,       // オブジェクトリストに隠す
                    const long            z_order=0)         // マウスクリックの優先順位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeGannGridEmptyPoints(time1,price1,time2);
//--- エラー値をリセットする
   ResetLastError();
//--- 与えられた座標でギャングリッドを作成する
   if(!ObjectCreate(chart_ID,name,OBJ_GANNGRID,sub_window,time1,price1,time2,0))
     {
      Print(__FUNCTION__,
            ": failed to create \"Gann Grid\"! Error code = ",GetLastError());
      return(false);
     }
//--- スケール(バーあたりのピップ数)を変更する
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- ギャンファンのトレンド方向を変更する(true - 下降, false - 上行)
   ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction);
//--- グリッドの色を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- グリッド線の表示スタイルを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- グリッド線の幅を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 強調表示してグリッドを移動するモードを有効(true)か無効(false)にする
```

© 2000-2025, MetaQuotes Ltd.

```
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| ギャングリッドのアンカーポイントを移動する                                |
//+------------------------------------------------------------------+
bool GannGridPointChange(const long   chart_ID=0,       // チャート識別子
                         const string name="GannGrid", // グリッド名
                         const int    point_index=0,   // アンカーポイントのインデックス
                         datetime     time=0,          // アンカーポイントの時間座標
                         double       price=0)         // アンカーポイントの価格座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- グリッドのアンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| ギャングリッドの規模を変更する                                        |
//+------------------------------------------------------------------+
bool GannGridScaleChange(const long   chart_ID=0,       // チャート識別子
                         const string name="GannGrid", // グリッド
                         const double scale=1.0)       // 規模
  {
//--- エラー値をリセットする
   ResetLastError();
//--- スケール(バーあたりのピップ数)を変更する
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print(__FUNCTION__,
             ": failed to change the scale! Error code = ",GetLastError());
       return(false);
      }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| ギャングリッドのトレンド方向を変更する                                |
//+------------------------------------------------------------------+
bool GannGridDirectionChange(const long   chart_ID=0,       // チャート識別子
                             const string name="GannGrid", // グリッド名
                             const bool   direction=true)  // トレンドの方向
  {
//--- エラー値をリセットする
   ResetLastError();
//--- ギャングリッドのトレンド方向を変更する
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction))
     {
      Print(__FUNCTION__,
            ": failed to change trend direction! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| この関数はチャートからギャンファンを削除する                          |
//+------------------------------------------------------------------+
bool GannGridDelete(const long   chart_ID=0,       // チャート識別子
                    const string name="GannGrid") // グリッド名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- ギャングリッドを削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete \"Gann Grid\"! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| ギャングリッドのアンカーポイントの値をチェックして                    |
//| 空の物には初期値を設定する                                          |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void ChangeGannGridEmptyPoints(datetime &time1,double &price1,datetime &time2)
  {
//--- 2 番目のポイントの時間が設定されていない場合、現在足になる
   if(!time2)
      time2=TimeCurrent();
//--- 1 番目のポイントの時間が設定されていない場合、2 番目から 9 バー左に置かれる
   if(!time1)
     {
      //--- 最後の 10 バーのオープン時間を受信するための配列
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- 1 番目のポイントを 2 番目から 9 バー左に設定する
      time1=temp[0];
     }
//--- 1 番目のポイントの価格が設定されていない場合、売値になる
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 価格配列サイズ
   int accuracy=1000;
//--- アンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
   double   price[];
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 価格配列に書き込む
```

© 2000-2025, MetaQuotes Ltd.

```
//--- チャートの最高値と最安値を見つける
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- ギャングリッド描画のポイントを定義する
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- ギャングリッドを作成する
   if(!GannGridCreate(0,InpName,0,date[d1],price[p1],date[d2],InpScale,InpDirection,
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- グリッドのアンカーポイントを移動する
//--- ループカウンタ
   int v_steps=accuracy/4;
//--- 1 番目のアンカーポイントを垂直方向に移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p1<accuracy-1)
         p1+=1;
      if(!GannGridPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- ループカウンタ
   int h_steps=bars/4;
//--- 2 番目のアンカーポイントを水平方向に移動する
   for(int i=0;i<h_steps;i++)
     {
      //--- 次の値を使用する
      if(d2<bars-1)
         d2+=1;
      if(!GannGridPointChange(0,InpName,1,date[d2],0))
         return;
```

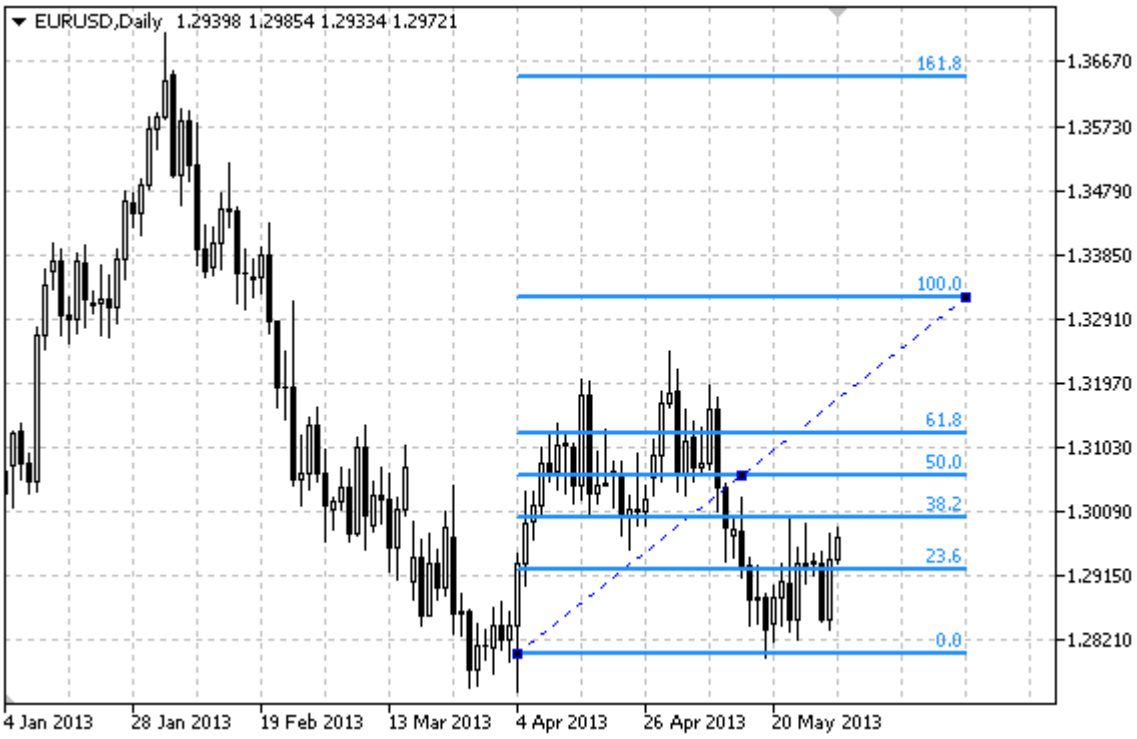
© 2000-2025, MetaQuotes Ltd.

```
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
      // 0.05 秒の遅れ
      Sleep(50);
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- グリッドのトレンド方向を下降に変更する
   GannGridDirectionChange(0,InpName,true);
//--- チャートを再描画する
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//--- チャートからグリッドを削除する
   GannGridDelete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//---
  }
```

© 2000–2025, MetaQuotes Ltd.

OBJ_FIBO

フィボナッチリトレースメント



注意事項

フィボナッチリトレースメントでは、継続のモードを右及び/または左に指定することが出来ます（OBJPROP_RAY_RIGHT 及び OBJPROP_RAY_LEFT プロパティ）

ラインレベル、その値と色も指定することが出来ます。

例

次のスクリプトは、チャート上でフィボナッチリトレースメントを作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script draws \"Fibonacci Retracement\" graphical object."
#property description "Anchor point coordinates are set in percentage of"
#property description "the chart window size."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string           InpName="FiboLevels";      // オブジェクト名
input int              InpDate1=10;               // 1 番目のポイントの日付( % )
input int              InpPrice1=65;              // 1 番目のポイントの価格( % )
input int              InpDate2=90;               // 2 番目のポイントの日付( % )
input int              InpPrice2=85;              // 2 番目のポイントの価格( % )
input color            InpColor=clrRed;           // オブジェクトの色
```

© 2000-2025, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // 線のスタイル
input int             InpWidth=2;                // 線の幅
input bool            InpBack=false;             // 背景オブジェクト
input bool            InpSelection=true;         // 強調表示して移動
input bool            InpRayLeft=false;          // オブジェクトの左への継続
input bool            InpRayRight=false;         // オブジェクトの右への継続
input bool            InpHidden=true;            // オブジェクトリストに隠す
input long            InpZOrder=0;               // マウスクリックの優先順位
//+------------------------------------------------------------------+
//|  与えられた座標でフィボナッチリトレースメントを作成する                |
//+------------------------------------------------------------------+
bool FiboLevelsCreate(const long            chart_ID=0,        // チャート識別子
                      const string          name="FiboLevels", // オブジェクト名
                      const int             sub_window=0,      // サブウィンドウ番号
                      datetime              time1=0,           // 1 番目のポイントの時間
                      double                price1=0,          // 1 番目のポイントの価格
                      datetime              time2=0,           // 2 番目のポイントの時間
                      double                price2=0,          // 2 番目のポイントの価格
                      const color           clr=clrRed,        // オブジェクトの色
                      const ENUM_LINE_STYLE style=STYLE_SOLID, // オブジェクト線のスタイル
                      const int             width=1,           // オブジェクト線の幅
                      const bool            back=false,        // 背景で表示する
                      const bool            selection=true,    // 強調表示して移動
                      const bool            ray_left=false,    // オブジェクトの左への継続
                      const bool            ray_right=false,   // オブジェクトの右への継続
                      const bool            hidden=true,       // オブジェクトリストに隠す
                      const long            z_order=0)         // マウスクリックの優先順位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeFiboLevelsEmptyPoints(time1,price1,time2,price2);
//--- エラー値をリセットする
   ResetLastError();
//--- 与えられた座標でフィボナッチリトレースメントを作成する
   if(!ObjectCreate(chart_ID,name,OBJ_FIBO,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": failed to create \"Fibonacci Retracement\"! Error code =
",GetLastError());
      return(false);
     }
//--- 色を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 線のスタイルを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 線の幅を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 強調表示してチャンネルを移動するモードを有効(true)か無効(false)にする。
```

© 2000-2025, MetaQuotes Ltd.

```
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトの表示を左に延長するモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- オブジェクトの表示を右に延長するモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| レベルの数とパラメータを設定する                                 |
//+------------------------------------------------------------------+
bool FiboLevelsSet(int              levels,           // レベル線の数
                   double           &values[],        // レベル線の値
                   color            &colors[],        // レベル線の色
                   ENUM_LINE_STYLE &styles[],         // レベル線のスタイル
                   int              &widths[],        // レベル線の幅
                   const long       chart_ID=0,       // チャート識別子
                   const string     name="FiboLevels") // オブジェクト名
  {
//--- 配列のサイズをチェックする
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": array length does not correspond to the number of
levels, error!");
      return(false);
     }
//--- レベル数を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- ループ内でレベルプロパティを設定する
   for(int i=0;i<levels;i++)
     {
      //--- レベル値
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- レベルの色
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- レベルのスタイル
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- レベルの幅
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- レベルの説明
```

© 2000-2025, MetaQuotes Ltd.

```
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i]
,1));
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| フィボナッチリトレースメントのアンカーポイントを移動する                          |
//+------------------------------------------------------------------+
bool FiboLevelsPointChange(const long   chart_ID=0,        // チャート識別子
                           const string name="FiboLevels", // オブジェクト名
                           const int    point_index=0,     // アンカーポイントのインデックス
                           datetime     time=0,            // アンカーポイントの時間座標
                           double       price=0)           // アンカーポイントの価格座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| フィボナッチリトレースメントを削除する                                      |
//+------------------------------------------------------------------+
bool FiboLevelsDelete(const long   chart_ID=0,        // チャート識別子
                      const string name="FiboLevels") // オブジェクト名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- オブジェクトを削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete \"Fibonacci Retracement\"! Error code =
",GetLastError());
      return(false);
     }
//--- 実行成功
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| フィボナッチリトレースメントのアンカーポイントの値をチェックして |
//| 空の物には初期値を設定する                                       |
//+------------------------------------------------------------------+
void ChangeFiboLevelsEmptyPoints(datetime &time1,double &price1,
                                 datetime &time2,double &price2)
  {
//--- 2 番目のポイントの時間が設定されていない場合、現在足になる
   if(!time2)
      time2=TimeCurrent();
//--- 2 番目のポイントの価格が設定されていない場合、売値になる
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 1 番目のポイントの時間が設定されていない場合、2 番目から 9 バー左に置かれる
   if(!time1)
     {
      //--- 最後の 10 バーのオープン時間を受信するための配列
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- 1 番目のポイントを 2 番目から 9 バー左に設定する
      time1=temp[0];
     }
//--- 1 番目のポイントの価格が設定されていない場合、2 番目より 200 ポイント低くする
   if(!price1)
      price1=price2-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 価格配列サイズ
   int accuracy=1000;
//--- アンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
   double   price[];
//--- メモリ割り当て
```

© 2000-2025, MetaQuotes Ltd.

```
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 価格配列に書き込む
//--- チャートの最高値と最安値を見つける
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- フィボナッチリトレースメントの描画のポイントを定義する
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- オブジェクトを作成する
   if(!FiboLevelsCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,
      InpStyle
,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- アンカーポイントを移動する
//--- ループカウンタ
   int v_steps=accuracy*2/5;
//--- 1 番目のアンカーポイントを移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p1>1)
         p1-=1;
      //--- ポイントを移動する
      if(!FiboLevelsPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- 1 秒の遅れ
   Sleep(1000);
//--- ループカウンタ
   v_steps=accuracy*4/5;
//--- 2 番目のアンカーポイントを移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p2>1)
         p2-=1;
      //--- ポイントを移動する
      if(!FiboLevelsPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- チャートからオブジェクトを削除する
   FiboLevelsDelete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//---
  }
```

© 2000–2025, MetaQuotes Ltd.

OBJ_FIBOTIMES

フィボナッチタイムゾーン



注意事項

フィボナッチタイムゾーンでは、ラインレベル、その値と色も指定することが出来ます。

例

次のスクリプトは、チャート上でフィボナッチタイムゾーンを作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script draws \"Fibonacci Time Zones\" graphical object."
#property description "Anchor point coordinates are set in percentage of the size of"
#property description "the chart window."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string           InpName="FiboTimes";       // オブジェクト名
input int              InpDate1=10;               // 1 番目のポイントの日付( % )
input int              InpPrice1=45;              // 1 番目のポイントの価格( % )
input int              InpDate2=20;               // 2 番目のポイントの日付( % )
input int              InpPrice2=55;              // 2 番目のポイントの価格( % )
input color            InpColor=clrRed;           // オブジェクトの色
input ENUM_LINE_STYLE  InpStyle=STYLE_DASHDOTDOT; // 線のスタイル
input int              InpWidth=2;                // 線の幅
```

© 2000–2025, MetaQuotes Ltd.

```
input bool              InpBack=false;              // 背景オブジェクト
input bool              InpSelection=true;          // 強調表示して移動
input bool              InpHidden=true;             // オブジェクトリストに隠す
input long              InpZOrder=0;                // マウスクリックの優先順位
//+------------------------------------------------------------------+
//|  与えられた座標でフィボナッチタイムゾーンを作成する                |
//+------------------------------------------------------------------+
bool FiboTimesCreate(const long             chart_ID=0,         // チャート識別子
                     const string           name="FiboTimes",   // オブジェクト名
                     const int              sub_window=0,       // サブウィンドウ番号
                     datetime               time1=0,            // 1 番目のポイントの時間
                     double                 price1=0,           // 1 番目のポイントの価格
                     datetime               time2=0,            // 2 番目のポイントの時間
                     double                 price2=0,           // 2 番目のポイントの価格
                     const color            clr=clrRed,         // オブジェクトの色
                     const ENUM_LINE_STYLE  style=STYLE_SOLID,  // オブジェクト線のスタイル
                     const int              width=1,            // オブジェクト線の幅
                     const bool             back=false,         // 背景で表示する
                     const bool             selection=true,     // 強調表示して移動
                     const bool             hidden=true,        // オブジェクトリストに隠す
                     const long             z_order=0)          // マウスクリックの優先順位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeFiboTimesEmptyPoints(time1,price1,time2,price2);
//--- エラー値をリセットする
   ResetLastError();
//--- 与えられた座標でフィボナッチタイムゾーンを作成する
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOTIMES,sub_window,time1,price1,time2,price2
))
     {
      Print(__FUNCTION__,
            ": failed to create \"Fibonacci Time Zones\"! Error code =
",GetLastError());
      return(false);
     }
//--- 色を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 線のスタイルを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 線の幅を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 強調表示してチャンネルを移動するモードを有効(true)か無効(false)にする。
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| レベルの数とパラメータを設定する                                 |
//+------------------------------------------------------------------+
bool FiboTimesLevelsSet(int              levels,           // レベル線の数
                        double           &values[],        // レベル線の値
                        color            &colors[],        // レベル線の色
                        ENUM_LINE_STYLE  &styles[],        // レベル線のスタイル
                        int              &widths[],        // レベル線の幅
                        const long       chart_ID=0,       // チャート識別子
                        const string     name="FiboTimes") // オブジェクト名
  {
//--- 配列のサイズをチェックする
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": array length does not correspond to the number of
levels, error!");
      return(false);
     }
//--- レベル数を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- ループ内でレベルプロパティを設定する
   for(int i=0;i<levels;i++)
     {
      //--- レベル値
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- レベルの色
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- レベルのスタイル
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- レベルの幅
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- レベルの説明
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(values[i],1))
;
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| フィボナッチタイムゾーンのアンカーポイントを移動する             |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool FiboTimesPointChange(const long   chart_ID=0,        // チャート識別子
                          const string name="FiboTimes", // オブジェクト名
                          const int    point_index=0,    // アンカーポイントのインデックス
                          datetime     time=0,           // アンカーポイントの時間座標
                          double       price=0)          // アンカーポイントの価格座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| フィボナッチタイムゾーンを削除する                                |
//+------------------------------------------------------------------+
bool FiboTimesDelete(const long   chart_ID=0,        // チャート識別子
                     const string name="FiboTimes") // オブジェクト名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- オブジェクトを削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete \"Fibonacci Time Zones\"! Error code =
",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| フィボナッチタイムゾーンの値をチェックして                        |
//| 空の物には初期値を設定する                                        |
//+------------------------------------------------------------------+
void ChangeFiboTimesEmptyPoints(datetime &time1,double &price1,
                                datetime &time2,double &price2)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 1 番目のポイントの時間が設定されていない場合、現在足になる
   if(!time1)
      time1=TimeCurrent();
//--- 1 番目のポイントの価格が設定されていない場合、売値になる
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 2 番目のポイントの時間が設定されていない場合、2 番目から2バー左に置かれる
   if(!time2)
     {
      //--- 最後の 3 バーのオープン時間を受信するための配列
      datetime temp[3];
      CopyTime(Symbol(),Period(),time1,3,temp);
      //--- 1 番目のポイントを2 番目から 2 バー左に設定する
      time2=temp[0];
     }
//--- 2 番目のポイントの価格は設定されていない場合は 1 番目のポイントと等しい
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 価格配列サイズ
   int accuracy=1000;
//--- アンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
   double   price[];
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 価格配列に書き込む
//--- チャートの最高値と最安値を見つける
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- フィボナッチタイムゾーン描画のポイントを定義する
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- オブジェクトを作成する
   if(!FiboTimesCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- アンカーポイントを移動する
//--- ループカウンタ
   int h_steps=bars*2/5;
//--- 2 番目のアンカーポイントを移動する
   for(int i=0;i<h_steps;i++)
     {
      //--- 次の値を使用する
      if(d2<bars-1)
         d2+=1;
      //--- ポイントを移動する
      if(!FiboTimesPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
      // 0.05 秒の遅れ
      Sleep(50);
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- ループカウンタ
   h_steps=bars*3/5;
//--- 1 番目のアンカーポイントを移動する
   for(int i=0;i<h_steps;i++)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 次の値を使用する
      if(d1<bars-1)
         d1+=1;
      //--- ポイントを移動する
      if(!FiboTimesPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
      // 0.05 秒の遅れ
      Sleep(50);
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- チャートからオブジェクトを削除する
   FiboTimesDelete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//---
  }
```

© 2000–2025, MetaQuotes Ltd.

OBJ_FIBOFAN

フィボナッチファン（扇）



注意事項

フィボナッチファンでは、ラインレベルの数、値と色を指定することが出来ます。

例

次のスクリプトは、チャート上でフィボナッチファンを作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script draws \"Fibonacci Fan\" graphical object."
#property description "Anchor point coordinates are set in percentage of"
#property description "the chart window size."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string          InpName="FiboFan";          // ファンの名称
input int             InpDate1=10;                // 1 番目のポイントの日付( % )
input int             InpPrice1=25;               // 1 番目のポイントの価格( % )
input int             InpDate2=30;                // 2 番目のポイントの日付( % )
input int             InpPrice2=50;               // 2 番目のポイントの価格( % )
input color           InpColor=clrRed;            // ファンの線の色
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // 線のスタイル
input int             InpWidth=2;                 // 線の幅
```

© 2000–2025, MetaQuotes Ltd.

```
input bool             InpBack=false;              // 背景オブジェクト
input bool             InpSelection=true;          // 強調表示して移動
input bool             InpHidden=true;             // オブジェクトリストに隠す
input long             InpZOrder=0;                // マウスクリックの優先順位
//+------------------------------------------------------------------+
//|  与えられた座標でフィボナッチファンを作成する                                  |
//+------------------------------------------------------------------+
bool FiboFanCreate(const long             chart_ID=0,         // チャート識別子
                   const string           name="FiboFan",     // ファンの名称
                   const int              sub_window=0,       // サブウィンドウ番号
                   datetime               time1=0,            // 1 番目のポイントの時間
                   double                 price1=0,           // 1 番目のポイントの価格
                   datetime               time2=0,            // 2 番目のポイントの時間
                   double                 price2=0,           // 2 番目のポイントの価格
                   const color            clr=clrRed,         // ファンの線の色
                   const ENUM_LINE_STYLE  style=STYLE_SOLID,  // ファンの線のスタイル
                   const int              width=1,            // ファンの線の幅
                   const bool             back=false,         // 背景で表示する
                   const bool             selection=true,     // 強調表示して移動
                   const bool             hidden=true,        // オブジェクトリストに隠す
                   const long             z_order=0)          // マウスクリックの優先順位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeFiboFanEmptyPoints(time1,price1,time2,price2);
//--- エラー値をリセットする
   ResetLastError();
//--- 与えられた座標でフィボナッチファンを作成する
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOFAN,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": failed to create \"Fibonacci Fan\"! Error code = ",GetLastError());
      return(false);
     }
//--- 色を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 線のスタイルを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 線の幅を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 強調表示してファンを移動するモードを有効(true)か無効(false)にする
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| レベルの数とパラメータを設定する                                 |
//+------------------------------------------------------------------+
bool FiboFanLevelsSet(int             levels,          // レベル線の数
                      double          &values[],       // レベル線の値
                      color           &colors[],       // レベル線の色
                      ENUM_LINE_STYLE &styles[],       // レベル線のスタイル
                      int             &widths[],       // レベル線の幅
                      const long      chart_ID=0,      // チャート識別子
                      const string    name="FiboFan") // ファンの名称
  {
//--- 配列のサイズをチェックする
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": array length does not correspond to the number of
levels, error!");
      return(false);
     }
//--- レベル数を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- ループ内でレベルプロパティを設定する
   for(int i=0;i<levels;i++)
     {
      //--- レベル値
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- レベルの色
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- レベルのスタイル
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- レベルの幅
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- レベルの説明
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i]
,1));
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| フィボナッチファンのアンカーポイントを移動する                   |
//+------------------------------------------------------------------+
bool FiboFanPointChange(const long   chart_ID=0,     // チャート識別子
                        const string name="FiboFan", // ファンの名称
```

© 2000-2025, MetaQuotes Ltd.

```
                          const int     point_index=0,  // アンカーポイントのインデックス
                          datetime      time=0,         // アンカーポイントの時間座標
                          double        price=0)        // アンカーポイントの価格座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| フィボナッチリファンを削除する                                   |
//+------------------------------------------------------------------+
bool FiboFanDelete(const long   chart_ID=0,     // チャート識別子
                   const string name="FiboFan") // ファンの名称
  {
//--- エラー値をリセットする
   ResetLastError();
//--- ファンを削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete \"Fibonacci Fan\"! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| フィボナッチファンのアンカーポイントの値をチェックして           |
//| 空の物には初期値を設定する                                       |
//+------------------------------------------------------------------+
void ChangeFiboFanEmptyPoints(datetime &time1,double &price1,
                              datetime &time2,double &price2)
  {
//--- 2 番目のポイントの時間が設定されていない場合、現在足になる
   if(!time2)
      time2=TimeCurrent();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 2 番目のポイントの価格が設定されていない場合、売値になる
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 1 番目のポイントの時間が設定されていない場合、2 番目から 9 バー左に置かれる
   if(!time1)
     {
      //--- 最後の 10 バーのオープン時間を受信するための配列
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- 1 番目のポイントを 2 番目から 9 バー左に設定する
      time1=temp[0];
     }
//--- 1 番目のポイントの価格が設定されていない場合、2 番目より 200 ポイント低くする
   if(!price1)
      price1=price2-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 価格配列サイズ
   int accuracy=1000;
//--- アンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
   double   price[];
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 価格配列に書き込む
//--- チャートの最高値と最安値を見つける
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
```

© 2000-2025, MetaQuotes Ltd.

```
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- フィボナッチファン描画のポイントを定義する
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- オブジェクトを作成する
   if(!FiboFanCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- ファンのアンカーポイントを移動する
//--- ループカウンタ
   int v_steps=accuracy/2;
//--- 1 番目のアンカーポイントを移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p1<accuracy-1)
         p1+=1;
      //--- ポイントを移動する
      if(!FiboFanPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- ループカウンタ
   int h_steps=bars/4;
//--- 2 番目のアンカーポイントを移動する
   for(int i=0;i<h_steps;i++)
     {
      //--- 次の値を使用する
      if(d2<bars-1)
         d2+=1;
      //--- ポイントを移動する
      if(!FiboFanPointChange(0,InpName,1,date[d2],price[p2]))
```

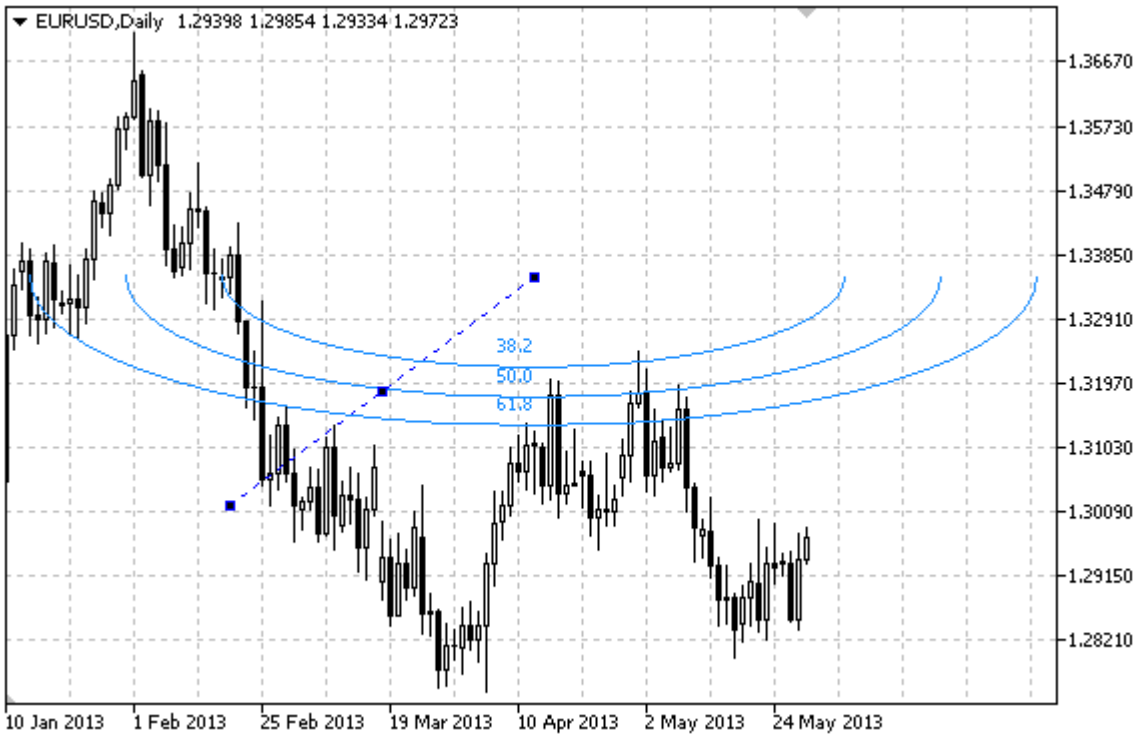
© 2000-2025, MetaQuotes Ltd.

```
            return;
        //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
        if(IsStopped())
            return;
        //--- チャートを再描画する
        ChartRedraw();
        // 0.05 秒の遅れ
        Sleep(50);
       }
//--- 1 秒の遅れ
    Sleep(1000);
//--- チャートからオブジェクトを削除する
    FiboFanDelete(0,InpName);
    ChartRedraw();
//--- 1 秒の遅れ
    Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_FIBOARC

フィボナッチアーク（円弧）



注意事項

フィボナッチアークでは、楕円形全体の表示モードを指定することが可能です。曲率半径は、スケールとアンカーポイントの座標を変更することにより指定することが出来ます。

ラインレベル、その値と色も指定することが出来ます。

例

次のスクリプトは、チャート上でフィボナッチアークを作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script draws \"Fibonacci Arcs\" graphical object."
#property description "Anchor point coordinates are set in percentage of"
#property description "the chart window size."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string          InpName="FiboArc";          // オブジェクト名
input int             InpDate1=25;                // 1 番目のポイントの日付( % )
input int             InpPrice1=25;               // 1 番目のポイントの価格( % )
input int             InpDate2=35;                // 2 番目のポイントの日付( % )
input int             InpPrice2=55;               // 2 番目のポイントの価格( % )
input double          InpScale=3.0;               // 規模
```

© 2000–2025, MetaQuotes Ltd.

```
input bool             InpFullEllipse=true;       // 円弧の図形
input color            InpColor=clrRed;           // 線の色
input ENUM_LINE_STYLE  InpStyle=STYLE_DASHDOTDOT; // 線のスタイル
input int              InpWidth=2;                // 線の幅
input bool             InpBack=false;             // 背景オブジェクト
input bool             InpSelection=true;         // 強調表示して移動
input bool             InpHidden=true;            // オブジェクトリストに隠す
input long             InpZOrder=0;               // マウスクリックの優先順位
//+------------------------------------------------------------------+
//|  与えられた座標でフィボナッチ円弧を作成する                      |
//+------------------------------------------------------------------+
bool FiboArcCreate(const long            chart_ID=0,          // チャート識別子
                   const string          name="FiboArc",      // オブジェクト名
                   const int             sub_window=0,        // サブウィンドウ番号
                   datetime              time1=0,             // 1 番目のポイントの時間
                   double                price1=0,            // 1 番目のポイントの価格
                   datetime              time2=0,             // 2 番目のポイントの時間
                   double                price2=0,            // 2 番目のポイントの価格
                   const double          scale=1.0,           // 規模
                   const bool            full_ellipse=false,  // 円弧の図形
                   const color           clr=clrRed,          // 線の色
                   const ENUM_LINE_STYLE style=STYLE_SOLID,   // 線のスタイル
                   const int             width=1,             // 線の幅
                   const bool            back=false,          // 背景で表示する
                   const bool            selection=true,      // 強調表示して移動
                   const bool            hidden=true,         // オブジェクトリストに隠す
                   const long            z_order=0)           // マウスクリックの優先順位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeFiboArcEmptyPoints(time1,price1,time2,price2);
//--- エラー値をリセットする
   ResetLastError();
//--- 与えられた座標でフィボナッチ円弧を作成する
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOARC,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": failed to create \"Fibonacci Arcs\"! Error code = ",GetLastError());
      return(false);
     }
//--- 規模を設定する
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- 円弧の表示を楕円形(true)またはその半分の弧(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_ELLIPSE,full_ellipse);
//--- 色を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 線のスタイルを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 線の幅を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 強調表示してアークを移動するモードを有効(true)か無効(false)にする
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| レベルの数とパラメータを設定する                                 |
//+------------------------------------------------------------------+
bool FiboArcLevelsSet(int              levels,          // レベル線の数
                      double           &values[],       // レベル線の値
                      color            &colors[],       // レベル線の色
                      ENUM_LINE_STYLE  &styles[],       // レベル線のスタイル
                      int              &widths[],       // レベル線の幅
                      const long       chart_ID=0,      // チャート識別子
                      const string     name="FiboArc") // オブジェクト名
  {
//--- 配列のサイズをチェックする
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": array length does not correspond to the number of
levels, error!");
      return(false);
     }
//--- レベル数を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- ループ内でレベルプロパティを設定する
   for(int i=0;i<levels;i++)
     {
      //--- レベル値
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- レベルの色
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- レベルのスタイル
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- レベルの幅
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- レベルの説明
```

© 2000-2025, MetaQuotes Ltd.

```
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i]
,1));
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| フィボナッチ円弧のアンカーポイントを移動する                                |
//+------------------------------------------------------------------+
bool FiboArcPointChange(const long   chart_ID=0,     // チャート識別子
                        const string name="FiboArc", // オブジェクト名
                        const int    point_index=0,  // アンカーポイントのインデックス
                        datetime     time=0,         // アンカーポイントの時間座標
                        double       price=0)        // アンカーポイントの価格座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| フィボナッチ円弧を削除する                                          |
//+------------------------------------------------------------------+
bool FiboArcDelete(const long   chart_ID=0,     // チャート識別子
                   const string name="FiboArc") // オブジェクト名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- オブジェクトを削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete \"Fibonacci Arcs\"! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| フィボナッチ円弧のアンカーポイントの値をチェックして                |
//| 空の物には初期値を設定する                                        |
//+------------------------------------------------------------------+
void ChangeFiboArcEmptyPoints(datetime &time1,double &price1,
                              datetime &time2,double &price2)
  {
//--- 2 番目のポイントの時間が設定されていない場合、現在足になる
   if(!time2)
      time2=TimeCurrent();
//--- 2 番目のポイントの価格が設定されていない場合、売値になる
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 1 番目のポイントの時間が設定されていない場合、2 番目から 9 バー左に置かれる
   if(!time1)
     {
      //--- 最後の 10 バーのオープン時間を受信するための配列
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- 1 番目のポイントを 2 番目から 9 バー左に設定する
      time1=temp[0];
     }
//--- 1 番目のポイントの価格が設定されていない場合、2 番目より 300 ポイント低くする
   if(!price1)
      price1=price2-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 価格配列サイズ
   int accuracy=1000;
//--- アンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
   double   price[];
//--- メモリ割り当て
   ArrayResize(date,bars);
```

© 2000-2025, MetaQuotes Ltd.

```
   ArrayResize(price,accuracy);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 価格配列に書き込む
//--- チャートの最高値と最安値を見つける
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- フィボナッチ円弧描画のポイントを定義する
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- オブジェクトを作成する
   if(!FiboArcCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpScale,
      InpFullEllipse
,InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- アンカーポイントを移動する
//--- ループカウンタ
   int v_steps=accuracy/5;
//--- 1 番目のアンカーポイントを移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p1<accuracy-1)
         p1+=1;
      //--- ポイントを移動する
      if(!FiboArcPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 1 秒の遅れ
   Sleep(1000);
//--- ループカウンタ
   int h_steps=bars/5;
//--- 2 番目のアンカーポイントを移動する
   for(int i=0;i<h_steps;i++)
     {
      //--- 次の値を使用する
      if(d2<bars-1)
         d2+=1;
      //--- ポイントを移動する
      if(!FiboArcPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
      // 0.05 秒の遅れ
      Sleep(50);
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- チャートからオブジェクトを削除する
   FiboArcDelete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_FIBOCHANNEL

フィボナッチチャンネル



注意事項

フィボナッチチャンネルでは、継続のモードを右及び/または左に指定することが出来ます（OBJPROP_RAY_RIGHT 及び OBJPROP_RAY_LEFT プロパティ）。

ラインレベル、その値と色も指定することが出来ます。

例

次のスクリプトは、チャート上でフィボナッチチャンネルを作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script draws \"Fibonacci Channel\" graphical object."
#property description "Anchor point coordinates are set in percentage of"
#property description "the chart window size."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string           InpName="FiboChannel";      // チャンネル名
input int              InpDate1=20;                // 1 番目のポイントの日付( % )
input int              InpPrice1=10;               // 1 番目のポイントの価格( % )
input int              InpDate2=60;                // 2 番目のポイントの日付( % )
input int              InpPrice2=30;               // 2 番目のポイントの価格( % )
input int              InpDate3=20;                // 3 番目のポイントの日付( % )
```

© 2000–2025, MetaQuotes Ltd.

```
input int              InpPrice3=25;              // 3 番目のポイントの価格( % )
input color            InpColor=clrRed;           // チャンネルの色
input ENUM_LINE_STYLE  InpStyle=STYLE_DASHDOTDOT; // チャンネルの線のスタイル
input int              InpWidth=2;                // チャンネルの線の幅
input bool             InpBack=false;             // 背景チャンネル
input bool             InpSelection=true;         // 強調表示して移動
input bool             InpRayLeft=false;          // チャンネルの左への継続
input bool             InpRayRight=false;         // チャンネルの右への継続
input bool             InpHidden=true;            // オブジェクトリストに隠す
input long             InpZOrder=0;               // マウスクリックの優先順位
//+------------------------------------------------------------------+
//| 与えられた座標でフィボナッチチャンネルを作成する                 |
//+------------------------------------------------------------------+
bool FiboChannelCreate(const long             chart_ID=0,          // チャート識別子
                       const string           name="FiboChannel", // チャンネル名
                       const int              sub_window=0,       // サブウィンドウ番号
                       datetime               time1=0,            // 1 番目のポイントの時
間
                       double                 price1=0,           // 1 番目のポイントの価
格
                       datetime               time2=0,            // 2 番目のポイントの時
間
                       double                 price2=0,           // 2 番目のポイントの値
                       datetime               time3=0,            // 3 番目のポイントの時
間
                       double                 price3=0,           // 3 番目のポイントの価
格
                       const color            clr=clrRed,         // チャンネルの色
                       const ENUM_LINE_STYLE  style=STYLE_SOLID,  // チャンネルの線のスタイル
                       const int              width=1,            // チャンネルの線の幅
                       const bool             back=false,         // 背景で表示する
                       const bool             selection=true,     // 強調表示して移動
                       const bool             ray_left=false,     // チャンネルの左への継続
                       const bool             ray_right=false,    // チャンネルの右への継続
                       const bool             hidden=true,        // オブジェクトリストに隠す
                       const long             z_order=0)          // マウスクリックの優先順
位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeFiboChannelEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- エラー値をリセットする
   ResetLastError();
//--- 与えられた座標でチャンネルを作成する
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOCHANNEL,sub_window,time1,price1,time2,pric
e2,time3,price3))
     {
      Print(__FUNCTION__,
            ": failed to create \"Fibonacci Channel\"! Error code =
",GetLastError());
      return(false);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- チャンネルの色を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- チャンネルの線のスタイルを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- チャンネルの線の幅を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 強調表示してチャンネルを移動するモードを有効(true)か無効(false)にする。
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- チャンネルの表示を左に延長するモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- チャンネルの表示を右に延長するモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| レベルの数とパラメータを設定する                                 |
//+------------------------------------------------------------------+
bool FiboChannelLevelsSet(int              levels,              // レベル線の数
                          double           &values[],           // レベル線の値
                          color            &colors[],           // レベル線の色
                          ENUM_LINE_STYLE  &styles[],           // レベル線のスタイル
                          int              &widths[],           // レベル線の幅
                          const long       chart_ID=0,          // チャート識別子
                          const string     name="FiboChannel") // オブジェクト名
  {
//--- 配列のサイズをチェックする
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": array length does not correspond to the number of
levels, error!");
      return(false);
     }
//--- レベル数を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- ループ内でレベルプロパティを設定する
   for(int i=0;i<levels;i++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- レベル値
       ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
       //--- レベルの色
       ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
       //--- レベルのスタイル
       ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
       //--- レベルの幅
       ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
       //--- レベルの説明
       ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i]
,1));
      }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| フィボナッチチャンネルのアンカーポイントを移動する                |
//+------------------------------------------------------------------+
bool FiboChannelPointChange(const long   chart_ID=0,          // チャート識別子
                            const string name="FiboChannel", // チャンネル名
                            const int    point_index=0,      // アンカーポイントのインデックス
                            datetime     time=0,             // アンカーポイントの時間座標
                            double       price=0)            // アンカーポイントの価格座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| チャンネルを削除する                                             |
//+------------------------------------------------------------------+
bool FiboChannelDelete(const long   chart_ID=0,          // チャート識別子
                       const string name="FiboChannel") // チャンネル名
  {
//--- エラー値をリセットする
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
//--- チャンネルを削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete \"Fibonacci Channel\"! Error code =
",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| フィボナッチリチャンネルのアンカーポイントの値をチェックして                |
//| 空の物には初期値を設定する                                          |
//+------------------------------------------------------------------+
void ChangeFiboChannelEmptyPoints(datetime &time1,double &price1,datetime &time2,
                                  double &price2,datetime &time3,double &price3)
  {
//--- 2 番目(右)のポイントの時間が設定されていない場合、現在足になる
   if(!time2)
      time2=TimeCurrent();
//--- 2 番目のポイントの価格が設定されていない場合、売値になる
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 1 番目(左)のポイントの時間が設定されていない場合、2 番目から 9 バー左に置かれる
   if(!time1)
     {
      //--- 最後の 10 バーのオープン時間を受信するための配列
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- 1 番目のポイントを 2 番目から 9 バー左に設定する
      time1=temp[0];
     }
//--- 1 番目のポイントの価格が設定されていない場合、2 番目より 300 ポイント高くする
   if(!price1)
      price1=price2+300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- 3 番目のポイントの時間が設定されていない場合、最初のポイントに等しい
   if(!time3)
      time3=time1;
//--- 3 番目のポイントの価格が設定されていない場合、2 番目のポイントに等しい
   if(!price3)
      price3=price2;
  }
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                   |
//+------------------------------------------------------------------+
void OnStart()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 入力パラメータの正しさをチェックする
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 価格配列サイズ
   int accuracy=1000;
//--- アンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
   double   price[];
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 価格配列に書き込む
//--- チャートの最高値と最安値を見つける
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- チャンネル描画のポイントを定義する
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- フィボナッチチャンネルを作成する
   if(!FiboChannelCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],
price[p3],
      InpColor
,InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
      return;
     }
```

© 2000–2025, MetaQuotes Ltd.

```
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- チャンネルのアンカーポイントを移動する
//--- ループカウンタ
   int h_steps=bars/10;
//--- 1 番目のアンカーポイントを移動する
   for(int i=0;i<h_steps;i++)
     {
      //--- 次の値を使用する
      if(d1>1)
         d1-=1;
      //--- ポイントを移動する
      if(!FiboChannelPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
      // 0.05 秒の遅れ
      Sleep(50);
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- ループカウンタ
   int v_steps=accuracy/10;
//--- 2 番目のアンカーポイントを移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p2>1)
         p2-=1;
      //--- ポイントを移動する
      if(!FiboChannelPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- ループカウンタ
   v_steps=accuracy/15;
//--- 3 番目のアンカーポイントを移動する
   for(int i=0;i<v_steps;i++)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 次の値を使用する
      if(p3<accuracy-1)
         p3+=1;
      //--- ポイントを移動する
      if(!FiboChannelPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- チャートからチャンネルを削除する
   FiboChannelDelete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//---
  }
```

© 2000–2025, MetaQuotes Ltd.

OBJ_EXPANSION

フィボナッチ拡張



注意事項

フィボナッチ拡張では、継続のモードを右及び/または左に指定することが出来ます（OBJPROP_RAY_RIGHT 及び OBJPROP_RAY_LEFT プロパティ）。

ラインレベル、その値と色も指定することが出来ます。

例

次のスクリプトは、チャート上でフィボナッチ拡張を作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script draws \"Fibonacci Expansion\"graphical object."
#property description "Anchor point coordinates are set in percentage of"
#property description "the chart window size."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string          InpName="FiboExpansion";    // オブジェクト名
input int             InpDate1=10;                // 1 番目のポイントの日付( % )
input int             InpPrice1=55;               // 1 番目のポイントの価格( % )
input int             InpDate2=30;                // 2 番目のポイントの日付( % )
input int             InpPrice2=10;               // 2 番目のポイントの価格( % )
input int             InpDate3=80;                // 3 番目のポイントの日付( % )
```

© 2000–2025, MetaQuotes Ltd.

```
input int              InpPrice3=75;              // 3 番目のポイントの価格( % )
input color            InpColor=clrRed;           // オブジェクトの色
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // 線のスタイル
input int              InpWidth=2;                // 線の幅
input bool             InpBack=false;             // 背景オブジェクト
input bool             InpSelection=true;         // 強調表示して移動
input bool             InpRayLeft=false;          // オブジェクトの左への継続
input bool             InpRayRight=false;         // オブジェクトの右への継続
input bool             InpHidden=true;            // オブジェクトリストに隠す
input long             InpZOrder=0;               // マウスクリックの優先順位
//+-----------------------------------------------------------------+
//| 与えられた座標でフィボナッチ拡張を作成する                                |
//+-----------------------------------------------------------------+
bool FiboExpansionCreate(const long            chart_ID=0,           // チャート識別子
                         const string          name="FiboExpansion", // チャンネル名
                         const int             sub_window=0,         // サブウィンドウ番
号
                         datetime              time1=0,              // 1 番目のポイント
の時間
                         double                price1=0,             // 1 番目のポイント
の価格
                         datetime              time2=0,              // 2 番目のポイント
の時間
                         double                price2=0,             // 2 番目のポイント
の価格
                         datetime              time3=0,              // 3 番目のポイント
の時間
                         double                price3=0,             // 3 番目のポイント
の価格
                         const color           clr=clrRed,           // オブジェクトの色
                         const ENUM_LINE_STYLE style=STYLE_SOLID,    // 線のスタイル
                         const int             width=1,              // 線の幅
                         const bool            back=false,           // 背景で表示する
                         const bool            selection=true,       // 強調表示して移
動
                         const bool            ray_left=false,       // オブジェクトの左へ
の継続
                         const bool            ray_right=false,      // オブジェクトの右へ
の継続
                         const bool            hidden=true,          // オブジェクトリストに
隠す
                         const long            z_order=0)            // マウスクリックの優
先順位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeFiboExpansionEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- エラー値をリセットする
   ResetLastError();
//--- 与えられた座標でフィボナッチ拡張を作成する
   if(!ObjectCreate(chart_ID,name,OBJ_EXPANSION,sub_window,time1,price1,time2,price2
,time3,price3))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print(__FUNCTION__,
             ": failed to create \"Fibonacci Extension\"! Error code =
",GetLastError());
       return(false);
      }
//--- オブジェクトの色を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 線のスタイルを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 線の幅を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 強調表示してチャンネルを移動するモードを有効(true)か無効(false)にする。
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトの表示を左に延長するモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- オブジェクトの表示を右に延長するモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| レベルの数とパラメータを設定する                                 |
//+------------------------------------------------------------------+
bool FiboExpansionLevelsSet(int              levels,                // レベル線の数
                            double           &values[],             // レベル線の値
                            color            &colors[],             // レベル線の色
                            ENUM_LINE_STYLE &styles[],              // レベル線のスタイル
                            int              &widths[],             // レベル線の幅
                            const long       chart_ID=0,            // チャート識別子
                            const string     name="FiboExpansion") // オブジェクト名
  {
//--- 配列のサイズをチェックする
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": array length does not correspond to the number of
levels, error!");
      return(false);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- レベル数を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- ループ内でレベルプロパティを設定する
   for(int i=0;i<levels;i++)
     {
      //--- レベル値
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- レベルの色
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- レベルのスタイル
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- レベルの幅
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- レベルの説明
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,"FE
"+DoubleToString(100*values[i],1));
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  フィボナッチ拡張のアンカーポイントを移動する                                  |
//+------------------------------------------------------------------+
bool FiboExpansionPointChange(const long   chart_ID=0,            // チャート識別子
                              const string name="FiboExpansion", // オブジェクト名
                              const int    point_index=0,        // アンカーポイントのイン
デックス
                              datetime     time=0,               // アンカーポイントの時間
座標
                              double       price=0)              // アンカーポイントの価格
座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| フィボナッチ拡張を削除する                                       |
//+------------------------------------------------------------------+
bool FiboExpansionDelete(const long   chart_ID=0,           // チャート識別子
                         const string name="FiboExpansion") // オブジェクト名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- オブジェクトを削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete \"Fibonacci Expansion\"! Error code =
",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| フィボナッチ拡張のアンカーポイントの値をチェックして             |
//| 空の物には初期値を設定する                                       |
//+------------------------------------------------------------------+
void ChangeFiboExpansionEmptyPoints(datetime &time1,double &price1,datetime &time2,
                                    double &price2,datetime &time3,double &price3)
  {
//--- 3 番目(右)のポイントの時間が設定されていない場合、現在足になる
   if(!time3)
      time3=TimeCurrent();
//--- 3 番目のポイントの価格が設定されていない場合、売値になる
   if(!price3)
      price3=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 1 番目(左)のポイントの時間が設定されていない場合、3 番目から9バー左に置かれる
//--- 最後の 10 バーのオープン時間を受信するための配列
   datetime temp[];
   ArrayResize(temp,10);
   if(!time1)
     {
      CopyTime(Symbol(),Period(),time3,10,temp);
      //--- 1 番目のポイントを 2 番目から 9 バー左に設定する
      time1=temp[0];
     }
//--- 1 番目のポイントの価格が設定されていない場合、3 番目のポイントに等しい
   if(!price1)
      price1=price3;
//--- 2 番目のポイントの時間が設定されていない場合、3 番目から 7 バー左に置かれる
   if(!time2)
      time2=temp[2];
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 2 番目のポイントの価格が設定されていない場合、1 番目より 250 ポイント低くする
   if(!price2)
      price2=price1-250*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 価格配列サイズ
   int accuracy=1000;
//--- アンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
   double   price[];
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 価格配列に書き込む
//--- チャートの最高値と最安値を見つける
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- フィボナッチ拡張描画のポイントを定義する
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
```

© 2000-2025, MetaQuotes Ltd.

```
   int p3=InpPrice3*(accuracy-1)/100;
//--- フィボナッチ拡張を作成する
   if(!FiboExpansionCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3
],price[p3],
      InpColor
,InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- アンカーポイントを移動する
//--- ループカウンタ
   int v_steps=accuracy/10;
//--- 1 番目のアンカーポイントを移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p1>1)
         p1-=1;
      //--- ポイントを移動する
      if(!FiboExpansionPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- ループカウンタ
   v_steps=accuracy/2;
//--- 3 番目のアンカーポイントを移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p3>1)
         p3-=1;
      //--- ポイントを移動する
      if(!FiboExpansionPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 1 秒の遅れ
   Sleep(1000);
//--- ループカウンタ
   v_steps=accuracy*4/5;
//--- 2 番目のアンカーポイントを移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p2<accuracy-1)
         p2+=1;
      //--- ポイントを移動する
      if(!FiboExpansionPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- チャートからオブジェクトを削除する
   FiboExpansionDelete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ELLIOTWAVE5

エリオット動機波



注意事項

エリオット動機波では、ポイントを線でつなぐモードを有効/無効にすることができ(OBJPROP_DRAWLINESプロパティ)波のポジションレベルを(ENUM_ELLIOT_WAVE_DEGREE列挙から)設定することも出来ます。

例

次のスクリプトは、チャート上でエリオット動機波を作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script draws \"Elliott Motive Wave\"."
#property description "Anchor point coordinates are set in percentage of the size of"
#property description "the chart window."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string                InpName="ElliotWave5";   // 動機
input int                   InpDate1=10;             // 1 番目のポイントの日付( % )
input int                   InpPrice1=90;            // 1 番目のポイントの価格( % )
input int                   InpDate2=20;             // 2 番目のポイントの日付( % )
input int                   InpPrice2=40;            // 2 番目のポイントの価格( % )
input int                   InpDate3=30;             // 3 番目のポイントの日付( % )
```

© 2000-2025, MetaQuotes Ltd.

```
input int                      InpPrice3=60;              // 3 番目のポイントの価格( % )
input int                      InpDate4=40;               // 4 番目のポイントの日( % )
input int                      InpPrice4=10;              // 4 番目のポイントの価格( % )
input int                      InpDate5=60;               // 5 番目のポイントの日( % )
input int                      InpPrice5=40;              // 5 番目のポイントの価格( % )
input ENUM_ELLIOT_WAVE_DEGREE  InpDegree=ELLIOTT_MINOR;   // レベル
input bool                     InpDrawLines=true;         // 線を表示
input color                    InpColor=clrRed;           // 線の色
input ENUM_LINE_STYLE          InpStyle=STYLE_DASH;       // 線のスタイル
input int                      InpWidth=2;                // 線の幅
input bool                     InpBack=false;             // 背景オブジェクト
input bool                     InpSelection=true;         // 強調表示して移動
input bool                     InpHidden=true;            // オブジェクトリストに隠す
input long                     InpZOrder=0;               // マウスクリックの優先順位
//+------------------------------------------------------------------+
//|  与えられた座標で「エリオット動機波」を作成する                  |
//+------------------------------------------------------------------+
bool ElliotWave5Create(const long                 chart_ID=0,              // チャート識別子
                       const string               name="ElliotWave5",      // 波の名称
                       const int                  sub_window=0,            // サブウィンドウ番号
                       datetime                   time1=0,                 // 1 番目のポイントの時間
                       double                     price1=0,                // 1 番目のポイントの価格
                       datetime                   time2=0,                 // 2 番目のポイントの時間
                       double                     price2=0,                // 2 番目のポイントの価格
                       datetime                   time3=0,                 // 3 番目のポイントの時間
                       double                     price3=0,                // 3 番目のポイントの価格
                       datetime                   time4=0,                 // 4 番目のポイントの時間
                       double                     price4=0,                // 4 番目のポイントの価格
                       datetime                   time5=0,                 // 5 番目のポイントの時間
                       double                     price5=0,                // 5 番目のポイントの価格
                       const ENUM_ELLIOT_WAVE_DEGREE degree=ELLIOTT_MINUETTE, // 度
                       const bool                 draw_lines=true,         // 線の表示
                       const color                clr=clrRed,              // オブジェクトの色
                       const ENUM_LINE_STYLE      style=STYLE_SOLID,       // 線のスタイル
                       const int                  width=1,                 // 線の幅
```

© 2000–2025, MetaQuotes Ltd.

```
                    const bool                  back=false,              // 背
景で表示する
                    const bool                  selection=true,          // 強
調表示して移動
                    const bool                  hidden=true,             // オブ
ジェクトリストに隠す
                    const long                  z_order=0)               // マウ
スクリックの優先順位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeElliotWave5EmptyPoints(time1,price1,time2,price2,time3,price3,time4,price4,
time5,price5);
//--- エラー値をリセットする
   ResetLastError();
//--- 与えられた座標で「エリオット動機波」を作成する
   if(!ObjectCreate(chart_ID,name,OBJ_ELLIOTWAVE5,sub_window,time1,price1,time2,pric
e2,time3,
      price3,time4,price4,time5,price5))
     {
      Print(__FUNCTION__,
            ": failed to create \"Elliott Motive Wave\"! Error code =
",GetLastError());
      return(false);
     }
//--- 角度(波の大きさ)を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_DEGREE,degree);
//--- 線を表示するモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_DRAWLINES,draw_lines);
//--- オブジェクトの色を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 線のスタイルを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 線の幅を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 強調表示してチャンネルを移動するモードを有効(true)か無効(false)にする。
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| エリオット動機波のアンカーポイントを移動する                                    |
//+------------------------------------------------------------------+
bool ElliotWave5PointChange(const long    chart_ID=0,          // チャート識別子
                            const string name="ElliotWave5", // オブジェクト名
                            const int     point_index=0,       // アンカーポイントのインデックス
                            datetime      time=0,              // アンカーポイントの時間座標
                            double        price=0)             // アンカーポイントの価格座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| エリオット動機波を削除する                                       |
//+------------------------------------------------------------------+
bool ElliotWave5Delete(const long   chart_ID=0,          // チャート識別子
                       const string name="ElliotWave5") // オブジェクト名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- オブジェクトを削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete \"Elliott Motive Wave\"! Error code =
",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| エリオット動機波のアンカーポイントの値をチェックして             |
//| 空の物には初期値を設定する                                       |
//+------------------------------------------------------------------+
void ChangeElliotWave5EmptyPoints(datetime &time1,double &price1,
```

© 2000-2025, MetaQuotes Ltd.

```
                                    datetime &time2,double &price2,
                                    datetime &time3,double &price3,
                                    datetime &time4,double &price4,
                                    datetime &time5,double &price5)
  {
//--- 最後の 10 バーのオープン時間を受信するための配列
   datetime temp[];
   ArrayResize(temp,10);
//--- データを受け取る
   CopyTime(Symbol(),Period(),TimeCurrent(),10,temp);
//--- 現在のチャートでの一ポイントの値を受け取る
   double point=SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- 1 番目のポイントの時間が設定されていない場合、最後のバーから 9 バー左になる
   if(!time1)
      time1=temp[0];
//--- 1 番目のポイントの価格が設定されていない場合、売値になる
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 2 番目のポイントの時間が設定されていない場合、最後のバーから 7 バー左になる
   if(!time2)
      time2=temp[2];
//--- 1 番目のポイントの価格が設定されていない場合、2 番目より 300 ポイント低くする
   if(!price2)
      price2=price1-300*point;
//--- 3 番目のポイントの時間が設定されていない場合、最後のバーの 5 バー左になる
   if(!time3)
      time3=temp[4];
//--- 3 番目のポイントの価格が設定されていない場合、1 番目より250ポイント低くする
   if(!price3)
      price3=price1-250*point;
//--- 4 番目のポイントの時間が設定されていない場合、最後のバーから 3 バー左になる
   if(!time4)
      time4=temp[6];
//--- 4 番目のポイントの価格が設定されていない場合、1 番目より 550 ポイント低くする
   if(!price4)
      price4=price1-550*point;
//--- 4 番目のポイントの時間が設定されていない場合、最後のバーになる
   if(!time5)
      time5=temp[9];
//--- 4 番目のポイントの価格が設定されていない場合、1 番目より 450 ポイント低くする
   if(!price5)
      price5=price1-450*point;
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
```

© 2000-2025, MetaQuotes Ltd.

```
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100 ||
      InpDate4<0 || InpDate4>100 || InpPrice4<0 || InpPrice4>100 ||
      InpDate5<0 || InpDate5>100 || InpPrice5<0 || InpPrice5>100)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 価格配列サイズ
   int accuracy=1000;
//--- アンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
   double   price[];
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 価格配列に書き込む
//--- チャートの最高値と最安値を見つける
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
エリオット動機波の描画のポイントを定義する
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int d4=InpDate4*(bars-1)/100;
   int d5=InpDate5*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
   int p4=InpPrice4*(accuracy-1)/100;
   int p5=InpPrice5*(accuracy-1)/100;
//--- 「エリオット動機波」を作成する
   if(!ElliotWave5Create(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],
price[p3],
```

© 2000–2025, MetaQuotes Ltd.

```
      date[d4],price[p4],date[d5],price[p5],InpDegree
,InpDrawLines,InpColor,InpStyle,InpWidth,
      InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- アンカーポイントを移動する
//--- ループカウンタ
   int v_steps=accuracy/5;
//--- 5 番目のアンカーポイントを移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p5<accuracy-1)
         p5+=1;
      //--- ポイントを移動する
      if(!ElliotWave5PointChange(0,InpName,4,date[d5],price[p5]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- ループカウンタ
   v_steps=accuracy/5;
//--- 2 番目と3 番目のアンカーポイントを移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p2<accuracy-1)
         p2+=1;
      if(p3>1)
         p3-=1;
      //--- ポイントをシフトする
      if(!ElliotWave5PointChange(0,InpName,1,date[d2],price[p2]))
         return;
      if(!ElliotWave5PointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- 1 秒の遅れ
   Sleep(1000);
//--- ループカウンタ
   v_steps=accuracy*4/5;
//--- 1 番目と2 番目のアンカーポイントを移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p1>1)
         p1-=1;
      if(p4<accuracy-1)
         p4+=1;
      //--- ポイントをシフトする
      if(!ElliotWave5PointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!ElliotWave5PointChange(0,InpName,3,date[d4],price[p4]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- チャートからオブジェクトを削除する
   ElliotWave5Delete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//---
  }
```

© 2000–2025, MetaQuotes Ltd.

OBJ_ELLIOTWAVE3

エリオット修正波



注意事項

エリオット修正波では、ポイントを線でつなぐモードを有効/無効にすることができ（OBJPROP_DRAWLINESプロパティ）、波のポジションレベルを（ENUM_ELLIOT_WAVE_DEGREE 列挙から）設定することも出来ます。

例

次のスクリプトは、チャート上でエリオット修正波を作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script draws \"Elliott Correction Wave\" graphical object."
#property description "Anchor point coordinates are set in percentage of the chart's window"
#property description "size."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string                InpName="ElliotWave3";   //オブジェクト名
input int                   InpDate1=10;             // 1 番目のポイントの日付( % )
input int                   InpPrice1=90;            // 1 番目のポイントの価格( % )
input int                   InpDate2=30;             // 2 番目のポイントの日付( % )
input int                   InpPrice2=10;            // 2 番目のポイントの価格( % )
input int                   InpDate3=50;             // 3 番目のポイントの日付( % )
```

© 2000-2025, MetaQuotes Ltd.

```
input int                      InpPrice3=40;              // 3 番目のポイントの価格( % )
input ENUM_ELLIOT_WAVE_DEGREE  InpDegree=ELLIOTT_MINOR; // レベル
input bool                     InpDrawLines=true;         // 線を表示
input color                    InpColor=clrRed;           // 線の色
input ENUM_LINE_STYLE          InpStyle=STYLE_DASH;       // 線のスタイル
input int                      InpWidth=2;                // 線の幅
input bool                     InpBack=false;             // 背景オブジェクト
input bool                     InpSelection=true;         // 強調表示して移動
input bool                     InpHidden=true;            // オブジェクトリストに隠す
input long                     InpZOrder=0;               // マウスクリックの優先順位
//+------------------------------------------------------------------+
//|  与えられた座標で「エリオット修正波」を作成する                                  |
//+------------------------------------------------------------------+
bool ElliotWave3Create(const long                  chart_ID=0,              // チャート識別子
                       const string                name="ElliotWave3",      // 波の名称
                       const int                   sub_window=0,            // サブウィンドウ番号
                       datetime                    time1=0,                 // 1 番目のポイントの時間
                       double                      price1=0,                // 1 番目のポイントの価格
                       datetime                    time2=0,                 // 2 番目のポイントの時間
                       double                      price2=0,                // 2 番目のポイントの価格
                       datetime                    time3=0,                 // 3 番目のポイントの時間
                       double                      price3=0,                // 3 番目のポイントの価格
                       const ENUM_ELLIOT_WAVE_DEGREE degree=ELLIOTT_MINUETTE, // 度
                       const bool                  draw_lines=true,         // 線の表示
                       const color                 clr=clrRed,              // オブジェクトの色
                       const ENUM_LINE_STYLE       style=STYLE_SOLID,       // 線のスタイル
                       const int                   width=1,                 // 線の幅
                       const bool                  back=false,              // 背景で表示する
                       const bool                  selection=true,          // 強調表示して移動
                       const bool                  hidden=true,             // オブジェクトリストに隠す
                       const long                  z_order=0)               // マウスクリックの優先順位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeElliotWave3EmptyPoints(time1,price1,time2,price2,time3,price3);
//--- エラー値をリセットする
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
//--- 与えられた座標で「エリオット修正波」を作成する
   if(!ObjectCreate(chart_ID,name,OBJ_ELLIOTWAVE3,sub_window,time1,price1,time2,pric
e2,time3,price3))
     {
      Print(__FUNCTION__,
            ": failed to create \"Elliott Correction Wave\"! Error code =
",GetLastError());
      return(false);
     }
//--- 角度(波の大きさ)を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_DEGREE,degree);
//--- 線を表示するモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_DRAWLINES,draw_lines);
//--- オブジェクトの色を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 線のスタイルを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 線の幅を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 強調表示してチャンネルを移動するモードを有効(true)か無効(false)にする。
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| エリオット波動のアンカーポイントを移動する                                |
//+------------------------------------------------------------------+
bool ElliotWave3PointChange(const long   chart_ID=0,          // チャート識別子
                            const string name="ElliotWave3", // オブジェクト名
                            const int    point_index=0,       // アンカーポイントのインデックス
                            datetime     time=0,              // アンカーポイントの時間座標
                            double       price=0)             // アンカーポイントの価格座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| エリオット修正波を削除する                                       |
//+------------------------------------------------------------------+
bool ElliotWave3Delete(const long   chart_ID=0,          // チャート識別子
                       const string name="ElliotWave3") // オブジェクト名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- オブジェクトを削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete \"Elliott Correction Wave\"! Error code =
",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| エリオット修正波のアンカーポイントの値をチェックして             |
//| 空の物には初期値を設定する                                       |
//+------------------------------------------------------------------+
void ChangeElliotWave3EmptyPoints(datetime &time1,double &price1,
                                  datetime &time2,double &price2,
                                  datetime &time3,double &price3)
  {
//--- 最後の 10 バーのオープン時間を受信するための配列
   datetime temp[];
   ArrayResize(temp,10);
//--- データを受け取る
   CopyTime(Symbol(),Period(),TimeCurrent(),10,temp);
//--- 現在のチャートでのポイントの値を受け取る
   double point=SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- 1 番目のポイントの時間が設定されていない場合、最後のバーから 9 バー左になる
   if(!time1)
      time1=temp[0];
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 1 番目のポイントの価格が設定されていない場合、売値になる
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 2 番目のポイントの時間が設定されていない場合、最後のバーから 5 バー左になる
   if(!time2)
      time2=temp[4];
//--- 1 番目のポイントの価格が設定されていない場合、2 番目より 300 ポイント低くする
   if(!price2)
      price2=price1-300*point;
//--- 3 番目のポイントの時間が設定されていない場合、最後のバーの 1 バー左になる
   if(!time3)
      time3=temp[8];
//--- 3 番目のポイントの価格が設定されていない場合、1 番目より 200 ポイント低くする
   if(!price3)
      price3=price1-200*point;
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 価格配列サイズ
   int accuracy=1000;
//--- アンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
   double   price[];
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 価格配列に書き込む
//--- チャートの最高値と最安値を見つける
```

© 2000-2025, MetaQuotes Ltd.

```
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- エリオット修正波の描画のポイントを定義する
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- エリオット修正波を作成する
   if(!ElliotWave3Create(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],
price[p3],
      InpDegree
,InpDrawLines,InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- アンカーポイントを移動する
//--- ループカウンタ
   int v_steps=accuracy/5;
//--- 3 番目のアンカーポイントを移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p3<accuracy-1)
         p3+=1;
      //--- ポイントを移動する
      if(!ElliotWave3PointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- ループカウンタ
   v_steps=accuracy*4/5;
//--- 1 番目と2 番目のアンカーポイントを移動する
   for(int i=0;i<v_steps;i++)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 次の値を使用する
      if(p1>1)
         p1-=1;
      if(p2<accuracy-1)
         p2+=1;
      //--- ポイントをシフトする
      if(!ElliotWave3PointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!ElliotWave3PointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- チャートからオブジェクトを削除する
   ElliotWave3Delete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//---
  }
```

© 2000–2025, MetaQuotes Ltd.

OBJ_RECTANGLE

四角形



注意事項

四角形では、色で塗りつぶすモードが OBJPROP_FILL プロパティによって設定されます。

例

次のスクリプトは、チャート上で四角形を作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script creates rectangle on the chart."
#property description "Anchor point coordinates are set in"
#property description "percentage of the chart window size."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string           InpName="Rectangle"; // 四角形名
input int              InpDate1=40;         // 1 番目のポイントの日付( % )
input int              InpPrice1=40;        // 1 番目のポイントの価格( % )
input int              InpDate2=60;         // 2 番目のポイントの日付( % )
input int              InpPrice2=60;        // 2 番目のポイントの価格( % )
input color            InpColor=clrRed;     // 四角形の色
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // 四角形の線のスタイル
input int              InpWidth=2;          // 四角形の線の幅
```

© 2000–2025, MetaQuotes Ltd.

```
input bool             InpFill=true;         // 四角形を色で塗りつぶす
input bool             InpBack=false;        // 背景の重宝系
input bool             InpSelection=true;    // 強調表示して移動
input bool             InpHidden=true;       // オブジェクトリストに隠す
input long             InpZOrder=0;          // マウスクリックの優先順位
//+------------------------------------------------------------------+
//|  与えられた座標で四角形を作成する                                |
//+------------------------------------------------------------------+
bool RectangleCreate(const long             chart_ID=0,        // チャート識別子
                     const string           name="Rectangle",  // 四角形名
                     const int              sub_window=0,      // サブウィンドウ番号
                     datetime               time1=0,           // 1 番目のポイントの時間
                     double                 price1=0,          // 1 番目のポイントの価格
                     datetime               time2=0,           // 2 番目のポイントの時間
                     double                 price2=0,          // 2 番目のポイントの価格
                     const color            clr=clrRed,        // 四角形の色
                     const ENUM_LINE_STYLE  style=STYLE_SOLID, // 四角形の線のスタイルs
                     const int              width=1,           // 四角形の線の幅
                     const bool             fill=false,        // 四角形を色で塗りつぶす
                     const bool             back=false,        // 背景で表示する
                     const bool             selection=true,    // 強調表示して移動
                     const bool             hidden=true,       // オブジェクトリストに隠す
                     const long             z_order=0)         // マウスクリックの優先順位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeRectangleEmptyPoints(time1,price1,time2,price2);
//--- エラー値をリセットする
   ResetLastError();
//--- 与えられた座標で四角形を作成する
   if(!ObjectCreate(chart_ID,name,OBJ_RECTANGLE,sub_window,time1,price1,time2,price2
))
     {
      Print(__FUNCTION__,
            ": failed to create a rectangle! Error code = ",GetLastError());
      return(false);
     }
//--- 四角形の色を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 四角形の線のスタイルを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 四角形の線の幅を設定s
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 四角形を色で塗りつぶすモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 強調表示して 四角形を移動するモードを有効(true)か無効(false)にする
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
```

© 2000-2025, MetaQuotes Ltd.

```
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 四角形のアンカーポイントを移動する                                |
//+------------------------------------------------------------------+
bool RectanglePointChange(const long   chart_ID=0,        // チャート識別子
                          const string name="Rectangle", // 四角形名
                          const int    point_index=0,    // アンカーポイントのインデックス
                          datetime     time=0,           // アンカーポイントの時間座標
                          double       price=0)          // アンカーポイントの価格座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 四角形を削除する                                                 |
//+------------------------------------------------------------------+
bool RectangleDelete(const long   chart_ID=0,        // チャート識別子
                     const string name="Rectangle") // 四角形名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 四角形を削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete rectangle! Error code = ",GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 四角形のアンカーポイントの値をチェックして                       |
//| 空の物には初期値を設定する                                       |
//+------------------------------------------------------------------+
void ChangeRectangleEmptyPoints(datetime &time1,double &price1,
                                datetime &time2,double &price2)
  {
//--- 1 番目のポイントの時間が設定されていない場合、現在足になる
   if(!time1)
      time1=TimeCurrent();
//--- 1 番目のポイントの価格が設定されていない場合、売値になる
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 2 番目のポイントの時間が設定されていない場合、2 番目から 9 バー左に置かれる
   if(!time2)
     {
      //--- 最後の 10 バーのオープン時間を受信するための配列
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- 2 番目のポイントを最初のものから 9 バー左に設定する
      time2=temp[0];
     }
//--- 1 番目のポイントの価格が設定されていない場合、2 番目より 300 ポイント低くする
   if(!price2)
      price2=price1-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 価格配列サイズ
   int accuracy=1000;
//--- アンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
```

© 2000-2025, MetaQuotes Ltd.

```
   datetime date[];
   double   price[];
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 価格配列に書き込む
//--- チャートの最高値と最安値を見つける
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 四角形描画のポイントを定義する
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- 四角形を作成する
   if(!RectangleCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,
      InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- 四角形のアンカーポイントを移動する
//--- ループカウンタ
   int h_steps=bars/2;
//--- アンカーポイントを移動する
   for(int i=0;i<h_steps;i++)
     {
      //--- 次の値を使用する
      if(d1<bars-1)
         d1+=1;
      if(d2>1)
         d2-=1;
      //--- ポイントをシフトする
      if(!RectanglePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!RectanglePointChange(0,InpName,1,date[d2],price[p2]))
```

© 2000–2025, MetaQuotes Ltd.

```
            return;
         //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
         if(IsStopped())
            return;
         //--- チャートを再描画する
         ChartRedraw();
         // 0.05 秒の遅れ
         Sleep(50);
        }
//--- 1 秒の遅れ
   Sleep(1000);
//--- ループカウンタ
   int v_steps=accuracy/2;
//--- アンカーポイントを移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p1<accuracy-1)
         p1+=1;
      if(p2>1)
         p2-=1;
      //--- ポイントをシフトする
      if(!RectanglePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!RectanglePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- チャートから四角形を削除する
   RectangleDelete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_TRIANGLE

三角形



注意事項

三角形では、色で塗りつぶすモードが OBJPROP_FILL プロパティによって設定されます。

例

次のスクリプトは、チャート上で三角形を作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script creates triangle on the chart."
#property description "Anchor point coordinates are set in"
#property description "percentage of the chart window size."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string           InpName="Triangle";         // 三角形の名称
input int              InpDate1=25;                // 1 番目のポイントの日付( % )
input int              InpPrice1=50;               // 1 番目のポイントの価格( % )
input int              InpDate2=70;                // 2 番目のポイントの日付( % )
input int              InpPrice2=70;               // 2 番目のポイントの価格( % )
input int              InpDate3=65;                // 3 番目のポイントの日付( % )
input int              InpPrice3=20;               // 3 番目のポイントの価格( % )
input color            InpColor=clrRed;            // 三角形の色
```

© 2000–2025, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // 三角形の線のスタイル
input int             InpWidth=2;                // 三角形の線の幅
input bool            InpFill=false;             // 三角形を色で塗りつぶす
input bool            InpBack=false;             // 背景の三角形
input bool            InpSelection=true;         // 強調表示して移動
input bool            InpHidden=true;            // オブジェクトリストに隠す
input long            InpZOrder=0;               // マウスクリックの優先順位
//+------------------------------------------------------------------+
//| 与えられた座標で三角形を作成する                                 |
//+------------------------------------------------------------------+
bool TriangleCreate(const long            chart_ID=0,        // チャート識別子
                    const string          name="Triangle",   // 三角形の名称
                    const int             sub_window=0,      // サブウィンドウ番号
                    datetime              time1=0,           // 1 番目のポイントの時間
                    double                price1=0,          // 1 番目のポイントの価格
                    datetime              time2=0,           // 2 番目のポイントの時間
                    double                price2=0,          // 2 番目のポイントの価格
                    datetime              time3=0,           // 3 番目のポイントの時間
                    double                price3=0,          // 3 番目のポイントの価
                    const color           clr=clrRed,        // 三角形の色
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // 三角形の線のスタイル
                    const int             width=1,           // 三角形の線の 幅
                    const bool            fill=false,        // 三角形を色で塗りつぶす
                    const bool            back=false,        // 背景で表示する
                    const bool            selection=true,    // 強調表示して移動
                    const bool            hidden=true,       // オブジェクトリストに隠す
                    const long            z_order=0)         // マウスクリックの優先順位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeTriangleEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- エラー値をリセットする
   ResetLastError();
//--- 与えられた座標で三角形を作成する
   if(!ObjectCreate(chart_ID,name,OBJ_TRIANGLE,sub_window,time1,price1,time2,price2,
time3,price3))
     {
      Print(__FUNCTION__,
            ": failed to create a triangle! Error code = ",GetLastError());
      return(false);
     }
//--- 三角形の色を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 三角形の線のスタイルを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 三角形の線の幅を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 三角形を色で塗りつぶすモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- 前景(false)または背景(true)に表示
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 強調表示して 三角形を移動するモードを有効(true)か無効(false)にする
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 三角形のアンカーポイントを移動する                                |
//+------------------------------------------------------------------+
bool TrianglePointChange(const long   chart_ID=0,       // チャート識別子
                         const string name="Triangle", // 三角形の名称
                         const int    point_index=0,   // アンカーポイントのインデックス
                         datetime     time=0,          // アンカーポイントの時間座標
                         double       price=0)         // アンカーポイントの価格座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 三角形を削除する                                                 |
//+------------------------------------------------------------------+
bool TriangleDelete(const long   chart_ID=0,       // チャート識別子
                    const string name="Triangle") // 三角形の名称
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 三角形を削除する
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete the ellipse! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  三角形のアンカーポイントの値をチェックして                      |
//|  空の物には初期値を設定する                                      |
//+------------------------------------------------------------------+
void ChangeTriangleEmptyPoints(datetime &time1,double &price1,
                               datetime &time2,double &price2,
                               datetime &time3,double &price3)
  {
//--- 1 番目のポイントの時間が設定されていない場合、現在足になる
   if(!time1)
      time1=TimeCurrent();
//--- 1 番目のポイントの価格が設定されていない場合、売値になる
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 2 番目のポイントの時間が設定されていない場合、2 番目から 9 バー左に置かれる
   if(!time2)
     {
      //--- 最後の 10 バーのオープン時間を受信するための配列
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- 2 番目のポイントを最初のものから 9 バー左に設定する
      time2=temp[0];
     }
//--- 1 番目のポイントの価格が設定されていない場合、2 番目より 300 ポイント低くする
   if(!price2)
      price2=price1-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- 3 番目のポイントの時間が設定されていない場合、2 番目のポイントの日に等しい
   if(!time3)
      time3=time2;
//--- 3 番目のポイントの価格が設定されていない場合、最初のポイントに等しい
   if(!price3)
      price3=price1;
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
```

© 2000-2025, MetaQuotes Ltd.

```
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 価格配列サイズ
   int accuracy=1000;
//--- アンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
   double   price[];
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 価格配列に書き込む
//--- チャートの最高値と最安値を見つける
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 三角形描画のポイントを定義する
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- 三角形を作成する
   if(!TriangleCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],pri
ce[p3],
      InpColor,InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- now, move the triangle anchor points
//--- ループカウンタ
   int v_steps=accuracy*3/10;
//--- 1 番目のアンカーポイントを移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p1>1)
         p1-=1;
      //--- ポイントを移動する
      if(!TrianglePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- ループカウンタ
   int h_steps=bars*9/20-1;
//--- 2 番目のアンカーポイントを移動する
   for(int i=0;i<h_steps;i++)
     {
      //--- 次の値を使用する
      if(d2>1)
         d2-=1;
      //--- ポイントを移動する
      if(!TrianglePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
      // 0.05 秒の遅れ
      Sleep(50);
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- ループカウンタ
   v_steps=accuracy/4;
//--- 3 番目のアンカーポイントを移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p3<accuracy-1)
         p3+=1;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- ポイントを移動する
      if(!TrianglePointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- チャートから三角形を削除する
   TriangleDelete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ELLIPSE

楕円形



注意事項

楕円形では、色で塗りつぶすモードが OBJPROP_FILL プロパティによって設定されます。

例

次のスクリプトは、チャート上で楕円形を作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script creates ellipse on the chart."
#property description "Anchor point coordinates are set"
#property description "in percentage of the chart window size."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string           InpName="Ellipse";          // 楕円形名
input int              InpDate1=30;                // 1 番目のポイントの日付( % )
input int              InpPrice1=20;               // 1 番目のポイントの価格( % )
input int              InpDate2=70;                // 2 番目のポイントの日付( % )
input int              InpPrice2=80;               // 2 番目のポイントの価格( % )
input int              InpDate3=50;                // 3 番目のポイントの日付( % )
input int              InpPrice3=60;               // 3 番目のポイントの価格( % )
input color            InpColor=clrRed;            // 楕円形の色
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // 楕円形の線のスタイル
```

© 2000–2025, MetaQuotes Ltd.

```
input int              InpWidth=2;                // 楕円形の線の幅
input bool             InpFill=false;             // 楕円形を色で塗りつぶす
input bool             InpBack=false;             // 背景の楕円形
input bool             InpSelection=true;         // 強調表示して移動
input bool             InpHidden=true;            // オブジェクトリストに隠す
input long             InpZOrder=0;               // マウスクリックの優先順位
//+------------------------------------------------------------------+
//|  与えられた座標で楕円形を作成する                                |
//+------------------------------------------------------------------+
bool EllipseCreate(const long            chart_ID=0,        // チャート識別子
                   const string          name="Ellipse",    // 楕円形の名称
                   const int             sub_window=0,      // サブウィンドウ番号
                   datetime              time1=0,           // 1 番目のポイントの時間
                   double                price1=0,          // 1 番目のポイントの価格
                   datetime              time2=0,           // 2 番目のポイントの時間
                   double                price2=0,          // 2 番目のポイントの価格
                   datetime              time3=0,           // 3 番目のポイントの時間
                   double                price3=0,          // 3 番目のポイントの価格
                   const color           clr=clrRed,        // 楕円形の色
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // 楕円形の線のスタイル
                   const int             width=1,           // 楕円形の線の幅
                   const bool            fill=false,        // 楕円形を色で塗りつぶす
                   const bool            back=false,        // 背景で表示する
                   const bool            selection=true,    // 強調表示して移動
                   const bool            hidden=true,       // オブジェクトリストに隠す
                   const long            z_order=0)         // マウスクリックの優先順位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeEllipseEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- エラー値をリセットする
   ResetLastError();
//--- 与えられた座標で楕円形を作成する
   if(!ObjectCreate(chart_ID,name,OBJ_ELLIPSE,sub_window,time1,price1,time2,price2,t
ime3,price3))
     {
      Print(__FUNCTION__,
            ": failed to create an ellipse! Error code = ",GetLastError());
      return(false);
     }
//--- 楕円形の色を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 楕円形の線のスタイルを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 楕円形の線の幅を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 楕円形を色で塗りつぶすモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
```

© 2000–2025, MetaQuotes Ltd.

```
//--- 強調表示して楕円形を移動するモードを有効(true)か無効(false)にする
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| アンカーポイントを移動する                                       |
//+------------------------------------------------------------------+
bool EllipsePointChange(const long   chart_ID=0,     // チャート識別子
                        const string name="Ellipse", // 楕円形の名称
                        const int    point_index=0,  // アンカーポイントのインデックス
                        datetime     time=0,         // アンカーポイントの時間座標
                        double       price=0)        // アンカーポイントの価格座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 楕円形を削除する                                                 |
//+------------------------------------------------------------------+
bool EllipseDelete(const long   chart_ID=0,     // チャート識別子
                   const string name="Ellipse") // 楕円形の名称
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 楕円形を削除する
   if(!ObjectDelete(chart_ID,name))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print(__FUNCTION__,
             ": failed to delete an ellipse! Error code = ",GetLastError());
       return(false);
      }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 楕円形のアンカーポイントの値をチェックして                                |
//| 空の物には初期値を設定する                                          |
//+------------------------------------------------------------------+
void ChangeEllipseEmptyPoints(datetime &time1,double &price1,
                              datetime &time2,double &price2,
                              datetime &time3,double &price3)
  {
//--- 1 番目のポイントの時間が設定されていない場合、現在足になる
   if(!time1)
      time1=TimeCurrent();
//--- 1 番目のポイントの価格が設定されていない場合、売値になる
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 2 番目のポイントの時間が設定されていない場合、2 番目から 9 バー左に置かれる
   if(!time2)
     {
      //--- 最後の 10 バーのオープン時間を受信するための配列
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- 2 番目のポイントを最初のものから 9 バー左に設定する
      time2=temp[0];
     }
//--- 1 番目のポイントの価格が設定されていない場合、2 番目より 300 ポイント低くする
   if(!price2)
      price2=price1-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- 3 番目のポイントの時間が設定されていない場合、2 番目のポイントの日に等しい
   if(!time3)
      time3=time2;
//--- 3 番目のポイントの価格が設定されていない場合、最初のポイントに等しい
   if(!price3)
      price3=price1;
  }
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
```

© 2000-2025, MetaQuotes Ltd.

```
       InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
      {
       Print("Error! Incorrect values of input parameters!");
       return;
      }
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 価格配列サイズ
   int accuracy=1000;
//--- アンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
   double   price[];
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 価格配列に書き込む
//--- チャートの最高値と最安値を見つける
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 楕円形描画のポイントを定義する
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- 楕円形を作成する
   if(!EllipseCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],pric
e[p3],
      InpColor,InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- 楕円形アンカーポイントを移動する
```

© 2000-2025, MetaQuotes Ltd.

```
//--- ループカウンタ
   int v_steps=accuracy/5;
//--- 1 番目と2 番目のアンカーポイントを移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p1<accuracy-1)
         p1+=1;
      if(p2>1)
         p2-=1;
      //--- ポイントをシフトする
      if(!EllipsePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!EllipsePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- ループカウンタ
   int h_steps=bars/5;
//--- 3 番目のアンカーポイントを移動する
   for(int i=0;i<h_steps;i++)
     {
      //--- 次の値を使用する
      if(d3>1)
         d3-=1;
      //--- ポイントを移動する
      if(!EllipsePointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
      // 0.05 秒の遅れ
      Sleep(50);
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- チャートから楕円形を削除する
   EllipseDelete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
```

© 2000-2025, MetaQuotes Ltd.

```
//---
  }
```

© 2000–2025, MetaQuotes Ltd.

OBJ_ARROW_THUMB_UP

サムズアップ記号



注意事項

記号に相対したアンカーポイントの位置関係は ENUM_ARROW_ANCHOR 列挙から選択出来ます。

5 以上の大きな矢印は MetaEditor でコードを書く際に OBJPROP_WIDTH プロパティ値を設定してのみ作成出来ます。

例

次のスクリプトは、チャート上でサムズアップ記号を作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script draws \"Thumbs Up\" sign."
#property description "Anchor point coordinate is set in percentage of"
#property description "the chart window size."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string             InpName="ThumbUp";     // 記号名
input int                InpDate=75;            // アンカーポイントの日付( % )
input int                InpPrice=25;           // アンカーポイントの価格( % )
input ENUM_ARROW_ANCHOR  InpAnchor=ANCHOR_TOP;  // アンカーの種類
input color              InpColor=clrRed;       // 記号の色
input ENUM_LINE_STYLE    InpStyle=STYLE_DOT;    // 境界線のスタイル
```

© 2000–2025, MetaQuotes Ltd.

```
input int                InpWidth=5;           // 記号サイズ
input bool               InpBack=false;        // 背景の記号
input bool               InpSelection=true;    // 強調表示して移動
input bool               InpHidden=true;       // オブジェクトリストに隠す
input long               InpZOrder=0;          // マウスクリックの優先順位
//+------------------------------------------------------------------+
//|  サムズアップ記号を作成する                                      |
//+------------------------------------------------------------------+
bool ArrowThumbUpCreate(const long              chart_ID=0,           // チャート識別子
                        const string            name="ThumbUp",       // 記号名
                        const int               sub_window=0,         // サブウィンドウ番号
                        datetime                time=0,               // アンカーポイントの時刻
                        double                  price=0,              // アンカーポイントの価格
                        const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // アンカーの種類
                        const color             clr=clrRed,           // 記号の色
                        const ENUM_LINE_STYLE   style=STYLE_SOLID,    // 境界線のスタイル
                        const int               width=3,              // 記号サイズ
                        const bool              back=false,           // 背景で表示する
                        const bool              selection=true,       // 強調表示して移動
                        const bool              hidden=true,          // オブジェクトリストに隠す
                        const long              z_order=0)            // マウスクリックの優先順位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeArrowEmptyPoint(time,price);
//--- エラー値をリセットする
   ResetLastError();
//--- 記号を作成する
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_THUMB_UP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": failed to create \"Thumbs Up\" sign! Error code = ",GetLastError());
      return(false);
     }
//--- アンカーの種類を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- 記号の色を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 境界線のスタイルを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 記号のサイズを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- マウスで記号を移動させるモードを有効(true)か無効(false)にする
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| アンカーポイントを移動する                                       |
//+------------------------------------------------------------------+
bool ArrowThumbUpMove(const long   chart_ID=0,      // チャート識別子
                      const string name="ThumbUp", // オブジェクト名
                      datetime     time=0,         // アンカーポイントの時間座標
                      double       price=0)        // アンカーポイントの価格座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| サムズアップ記号のアンカーの種類を変更する                       |
//+------------------------------------------------------------------+
bool ArrowThumbUpAnchorChange(const long              chart_ID=0,        // チャート識
別子
                              const string            name="ThumbUp",    // オブジェクト
名
                              const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // アンカーの
種類
  {
//--- エラー値をリセットする
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
//--- アンカーの種類を変更する
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": failed to change anchor type! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| サムズアップ記号を削除する                                       |
//+------------------------------------------------------------------+
bool ArrowThumbUpDelete(const long   chart_ID=0,     // チャート識別子
                        const string name="ThumbUp") // 記号名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 記号を削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete \"Thumbs Up\" sign! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| アンカーポイントの値をチェックして                               |
//| 空の物には初期値を設定する                                       |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- ポイントの時間が設定されていない場合、現在足になる
   if(!time)
      time=TimeCurrent();
//--- ポイントの価格が設定されていない場合、売値になる
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print("Error! Incorrect values of input parameters!");
       return;
      }
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 価格配列サイズ
   int accuracy=1000;
//--- アンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
   double   price[];
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 価格配列に書き込む
//--- チャートの最高値と最安値を見つける
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 記号描画のポイントを定義する
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- チャート上にサムズアップ記号を作成する
   if(!ArrowThumbUpCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- アンカーポイントを移動し、記号との相互位置を変更する
//--- ループカウンタ
   int h_steps=bars/4;
//--- アンカーポイントを移動する
   for(int i=0;i<h_steps;i++)
     {
      //--- 次の値を使用する
```

© 2000-2025, MetaQuotes Ltd.

```
      if(d>1)
         d-=1;
      //--- ポイントを移動する
      if(!ArrowThumbUpMove(0,InpName,date[d],price[p]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
      // 0.05 秒の遅れ
      Sleep(50);
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- ループカウンタ
   int v_steps=accuracy/4;
//--- アンカーポイントを移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p<accuracy-1)
         p+=1;
      //--- ポイントを移動する
      if(!ArrowThumbUpMove(0,InpName,date[d],price[p]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 記号に相対したアンカーポイントの位置関係を変更する
   ArrowThumbUpAnchorChange(0,InpName,ANCHOR_BOTTOM);
//--- チャートを再描画する
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//--- チャートから記号を削除する
   ArrowThumbUpDelete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_THUMB_DOWN

サムズダウン記号



注意事項

記号に相対したアンカーポイントの位置関係は ENUM_ARROW_ANCHOR 列挙から選択出来ます。

5 以上の大きな矢印は MetaEditor でコードを書く際に OBJPROP_WIDTH プロパティ値を設定してのみ作成出来ます。

例

次のスクリプトは、チャート上でサムズダウン記号を作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script draws \"Thumbs Down\" sign."
#property description "Anchor point coordinate is set in percentage of"
#property description "the chart window size."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string             InpName="ThumbDown";      // 記号名
input int                InpDate=25;               // アンカーポイントの日付( % )
input int                InpPrice=75;              // アンカーポイントの価格( % )
input ENUM_ARROW_ANCHOR  InpAnchor=ANCHOR_BOTTOM;  // アンカーの種類
input color              InpColor=clrRed;          // 記号の色
```

© 2000–2025, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;       // 境界線のスタイル
input int               InpWidth=5;               // 記号サイズ
input bool              InpBack=false;            // 背景の記号
input bool              InpSelection=true;        // 強調表示して移動
input bool              InpHidden=true;           // オブジェクトリストに隠す
input long              InpZOrder=0;              // マウスクリックの優先順位
//+------------------------------------------------------------------+
//|  サムズダウン記号を作成する                                      |
//+------------------------------------------------------------------+
bool ArrowThumbDownCreate(const long              chart_ID=0,           // チャート識別子
                          const string            name="ThumbDown",     // サイン名
                          const int               sub_window=0,         // サブウィンドウ番号
                          datetime                time=0,               // アンカーポイントの時刻
                          double                  price=0,              // アンカーポイントの価格
                          const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // アンカーの種類
                          const color             clr=clrRed,           // 記号の色
                          const ENUM_LINE_STYLE   style=STYLE_SOLID,    // 境界線のスタイル
                          const int               width=3,              // 記号サイズ
                          const bool              back=false,           // 背景で表示する
                          const bool              selection=true,       // 強調表示して移動
                          const bool              hidden=true,          // オブジェクトリストに隠す
                          const long              z_order=0)            // マウスクリックの優先順位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeArrowEmptyPoint(time,price);
//--- エラー値をリセットする
   ResetLastError();
//--- 記号を作成する
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_THUMB_DOWN,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": failed to create \"Thumbs Down\" sign! Error code = ",GetLastError());
      return(false);
     }
//--- アンカーの種類を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- 記号の色を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 境界線のスタイルを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
```

© 2000–2025, MetaQuotes Ltd.

```
//--- 記号のサイズを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- マウスで記号を移動させるモードを有効(true)か無効(false)にする
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| アンカーポイントを移動する                                       |
//+------------------------------------------------------------------+
bool ArrowThumbDownMove(const long   chart_ID=0,        // チャート識別子
                        const string name="ThumbDown", // オブジェクト名
                        datetime     time=0,           // アンカーポイントの時間座標
                        double       price=0)          // アンカーポイントの価格座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| サムズダウン記号のアンカーの種類を変更する                       |
//+------------------------------------------------------------------+
bool ArrowThumbDownAnchorChange(const long             chart_ID=0,        // チャート
識別子
                                const string           name="ThumbDown",  // オブジェク
ト名
```

© 2000-2025, MetaQuotes Ltd.

```
                                    const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // アンカー
の種類
  {
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーの種類を変更する
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": failed to change anchor type! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  サムズダウン記号を削除する                                      |
//+------------------------------------------------------------------+
bool ArrowThumbDownDelete(const long   chart_ID=0,       // チャート識別子
                          const string name="ThumbDown") // 記号名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 記号を削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete \"Thumbs Down\" sign! Error code =
",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  アンカーポイントの値をチェックして                              |
//|  空の物には初期値を設定する                                      |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- ポイントの時間が設定されていない場合、現在足になる
   if(!time)
      time=TimeCurrent();
//--- ポイントの価格が設定されていない場合、売値になる
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                              |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 価格配列サイズ
   int accuracy=1000;
//--- アンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
   double   price[];
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 価格配列に書き込む
//--- チャートの最高値と最安値を見つける
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 記号描画のポイントを定義する
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- チャート上にサムズダウン記号を作成する
   if(!ArrowThumbDownCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- アンカーポイントを移動し、記号との相互位置を変更する
//--- ループカウンタ
```

© 2000-2025, MetaQuotes Ltd.

```
   int h_steps=bars/4;
//--- アンカーポイントを移動する
   for(int i=0;i<h_steps;i++)
     {
      //--- 次の値を使用する
      if(d<bars-1)
         d+=1;
      //--- ポイントを移動する
      if(!ArrowThumbDownMove(0,InpName,date[d],price[p]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
      // 0.05 秒の遅れ
      Sleep(50);
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- ループカウンタ
   int v_steps=accuracy/4;
//--- アンカーポイントを移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p>1)
         p-=1;
      //--- ポイントを移動する
      if(!ArrowThumbDownMove(0,InpName,date[d],price[p]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 記号に相対したアンカーポイントの位置関係を変更する
   ArrowThumbDownAnchorChange(0,InpName,ANCHOR_TOP);
//--- チャートを再描画する
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//--- チャートから記号を削除する
   ArrowThumbDownDelete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//---
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

© 2000–2025, MetaQuotes Ltd.

OBJ_ARROW_UP

上矢印記号



注意事項

記号に相対したアンカーポイントの位置関係は ENUM_ARROW_ANCHOR 列挙から選択出来ます。

5 以上の大きな矢印は MetaEditor でコードを書く際に OBJPROP_WIDTH プロパティ値を設定してのみ作成出来ます。

例

次のスクリプトは、チャート上で上矢印記号を作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script draws \"Arrow Up\" sign."
#property description "Anchor point coordinate is set in"
#property description "percentage of the chart window size."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string             InpName="ArrowUp";     // 記号名
input int                InpDate=25;            // アンカーポイントの日付( % )
input int                InpPrice=25;           // アンカーポイントの価格( % )
input ENUM_ARROW_ANCHOR  InpAnchor=ANCHOR_TOP;  // アンカーの種類
input color              InpColor=clrRed;       // 記号の色
input ENUM_LINE_STYLE    InpStyle=STYLE_DOT;    // 境界線のスタイル
```

© 2000–2025, MetaQuotes Ltd.

```
input int                InpWidth=5;           // 記号サイズ
input bool               InpBack=false;        // 背景の記号
input bool               InpSelection=false;   // 強調表示して移動
input bool               InpHidden=true;       // オブジェクトリストに隠す
input long               InpZOrder=0;          // マウスクリックの優先順位
//+------------------------------------------------------------------+
//|  上矢印記号を作成する                                            |
//+------------------------------------------------------------------+
bool ArrowUpCreate(const long              chart_ID=0,           // チャート識別子
                   const string            name="ArrowUp",       // 記号名
                   const int               sub_window=0,         // サブウィンドウ番号
                   datetime                time=0,               // アンカーポイントの時刻
                   double                  price=0,              // アンカーポイントの価格
                   const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // アンカーの種類
                   const color             clr=clrRed,           // 記号の色
                   const ENUM_LINE_STYLE   style=STYLE_SOLID,    // 境界線のスタイル
                   const int               width=3,              // 記号サイズ
                   const bool              back=false,           // 背景で表示する
                   const bool              selection=true,       // 強調表示して移動
                   const bool              hidden=true,          // オブジェクトリストに隠す
                   const long              z_order=0)            // マウスクリックの優先順
位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeArrowEmptyPoint(time,price);
//--- エラー値をリセットする
   ResetLastError();
//--- 記号を作成する
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_UP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": failed to create \"Arrow Up\" sign! Error code = ",GetLastError());
      return(false);
     }
//--- アンカーの種類を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- 記号の色を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 境界線のスタイルを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 記号のサイズを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- マウスで記号を移動させるモードを有効(true)か無効(false)にする
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| アンカーポイントを移動する                                       |
//+------------------------------------------------------------------+
bool ArrowUpMove(const long   chart_ID=0,      // チャート識別子
                 const string name="ArrowUp", // オブジェクト名
                 datetime     time=0,         // アンカーポイントの時間座標
                 double       price=0)        // アンカーポイントの価格座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 下矢印記号のアンカーの種類を変更する                             |
//+------------------------------------------------------------------+
bool ArrowUpAnchorChange(const long              chart_ID=0,        // チャート識別子
                         const string            name="ArrowUp",    // オブジェクト名
                         const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // アンカーの種類
  {
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーポイントの位置を変更する
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": failed to change anchor type! Error code = ",GetLastError());
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 上矢印記号を削除する                                             |
//+------------------------------------------------------------------+
bool ArrowUpDelete(const long   chart_ID=0,     // チャート識別子
                   const string name="ArrowUp") // 記号名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 記号を削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete \"Arrow Up\" sign! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| アンカーポイントの値をチェックして                               |
//| 空の物には初期値を設定する                                       |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- ポイントの時間が設定されていない場合、現在足になる
   if(!time)
      time=TimeCurrent();
//--- ポイントの価格が設定されていない場合、売値になる
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 価格配列サイズ
   int accuracy=1000;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- アンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
   double   price[];
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 価格配列に書き込む
//--- チャートの最高値と最安値を見つける
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 記号描画のポイントを定義する
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- チャートに上矢印を作成する
   if(!ArrowUpCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- アンカーポイントを移動し、記号との相互位置を変更する
//--- ループカウンタ
   int v_steps=accuracy/2;
//--- アンカーポイントを移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p<accuracy-1)
         p+=1;
      //--- ポイントを移動する
      if(!ArrowUpMove(0,InpName,date[d],price[p]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- 記号に相対したアンカーポイントの位置関係を変更する
   ArrowUpAnchorChange(0,InpName,ANCHOR_BOTTOM);
//--- チャートを再描画する
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//--- チャートから記号を削除する
   ArrowUpDelete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//---
  }
```

© 2000–2025, MetaQuotes Ltd.

OBJ_ARROW_DOWN

下矢印記号



注意事項

記号に相対したアンカーポイントの位置関係は ENUM_ARROW_ANCHOR 列挙から選択出来ます。

5 以上の大きな矢印は MetaEditor でコードを書く際に OBJPROP_WIDTH プロパティ値を設定してのみ作成出来ます。

例

次のスクリプトは、チャート上で下矢印記号を作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script draws \"Arrow Down\" sign."
#property description "Anchor point coordinate is set in"
#property description "percentage of the chart window size."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string             InpName="ArrowDown";      // 記号名
input int                InpDate=75;               // アンカーポイントの日付( % )
input int                InpPrice=75;              // アンカーポイントの価格( % )
input ENUM_ARROW_ANCHOR  InpAnchor=ANCHOR_BOTTOM;  // アンカーの種類
input color              InpColor=clrRed;          // 記号の色
input ENUM_LINE_STYLE    InpStyle=STYLE_DOT;       // 境界線のスタイル
input int                InpWidth=5;               // 記号サイズ
```

© 2000–2025, MetaQuotes Ltd.

```
input bool               InpBack=false;           // 背景の記号
input bool               InpSelection=false;      // 強調表示して移動
input bool               InpHidden=true;          // オブジェクトリストに隠す
input long               InpZOrder=0;             // マウスクリックの優先順位
//+------------------------------------------------------------------+
//|  下矢印記号を作成する                                            |
//+------------------------------------------------------------------+
bool ArrowDownCreate(const long              chart_ID=0,           // チャート識別子
                     const string            name="ArrowDown",     // 記号名
                     const int               sub_window=0,         // サブウィンドウ番号
                     datetime                time=0,               // アンカーポイントの時刻
                     double                  price=0,              // アンカーポイントの価格
                     const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // アンカーの種類
                     const color             clr=clrRed,           // 記号の色
                     const ENUM_LINE_STYLE   style=STYLE_SOLID,    // 境界線のスタイル
                     const int               width=3,              // 記号サイズ
                     const bool              back=false,           // 背景で表示する
                     const bool              selection=true,       // 強調表示して移動
                     const bool              hidden=true,          // オブジェクトリストに隠す
                     const long              z_order=0)            // マウスクリックの優先順位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeArrowEmptyPoint(time,price);
//--- エラー値をリセットする
   ResetLastError();
//--- 記号を作成する
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_DOWN,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": failed to create \"Arrow Down\" sign! Error code = ",GetLastError());
      return(false);
     }
//--- アンカーの種類
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- 記号の色を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 境界線のスタイルを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 記号のサイズを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- マウスで記号を移動させるモードを有効(true)か無効(false)にする
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
```

© 2000-2025, MetaQuotes Ltd.

```
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| アンカーポイントを移動する                                       |
//+------------------------------------------------------------------+
bool ArrowDownMove(const long   chart_ID=0,        // チャート識別子
                   const string name="ArrowDown", // オブジェクト名
                   datetime     time=0,           // アンカーポイントの時間座標
                   double       price=0)          // アンカーポイントの価格座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 下矢印記号のアンカーの種類を変更する                             |
//+------------------------------------------------------------------+
bool ArrowDownAnchorChange(const long              chart_ID=0,         // チャート識別子
                           const string            name="ArrowDown",   // オブジェクト名
                           const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // アンカーの種類
  {
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーポイントの位置を変更する
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": failed to change anchor type! Error code = ",GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 下矢印記号を削除する                                             |
//+------------------------------------------------------------------+
bool ArrowDownDelete(const long   chart_ID=0,        // チャート識別子
                     const string name="ArrowDown") // 記号名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 記号を削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete \"Arrow Down\" sign! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| アンカーポイントの値をチェックして                               |
//| 空の物には初期値を設定する                                       |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- ポイントの時間が設定されていない場合、現在足になる
   if(!time)
      time=TimeCurrent();
//--- ポイントの価格が設定されていない場合、売値になる
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 価格配列サイズ
   int accuracy=1000;
//--- アンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
   double   price[];
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 価格配列に書き込む
//--- チャートの最高値と最安値を見つける
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 記号描画のポイントを定義する
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- チャートに下矢印を作成する
   if(!ArrowDownCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- アンカーポイントを移動し、記号との相互位置を変更する
//--- ループカウンタ
   int v_steps=accuracy/2;
//--- アンカーポイントを移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p>1)
         p-=1;
      //--- ポイントを移動する
      if(!ArrowDownMove(0,InpName,date[d],price[p]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
```

© 2000-2025, MetaQuotes Ltd.

```
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- 記号に相対したアンカーポイントの位置関係を変更する
   ArrowDownAnchorChange(0,InpName,ANCHOR_TOP);
//--- チャートを再描画する
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//--- チャートから記号を削除する
   ArrowDownDelete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//---
  }
```

© 2000–2025, MetaQuotes Ltd.

OBJ_ARROW_STOP

ストップサイン



注意事項

記号に相対したアンカーポイントの位置関係は ENUM_ARROW_ANCHOR 列挙から選択出来ます。

5 以上の大きな矢印は MetaEditor でコードを書く際に OBJPROP_WIDTH プロパティ値を設定してのみ作成出来ます。

例

次のスクリプトは、チャート上でストップサインを作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script draws \"Stop\" sign."
#property description "Anchor point coordinate is set in"
#property description "percentage of the chart window size."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string             InpName="ArrowStop";      // 記号名
input int                InpDate=10;               // アンカーポイントの日付( % )
input int                InpPrice=50;              // アンカーポイントの価格( % )
input ENUM_ARROW_ANCHOR  InpAnchor=ANCHOR_BOTTOM;  // アンカーの種類
input color              InpColor=clrRed;          // 記号の色
input ENUM_LINE_STYLE    InpStyle=STYLE_DOT;       // 境界線のスタイル
```

© 2000–2025, MetaQuotes Ltd.

```
input int                InpWidth=5;               // 記号サイズ
input bool               InpBack=false;            // 背景の記号
input bool               InpSelection=false;       // 強調表示して移動
input bool               InpHidden=true;           // オブジェクトリストに隠す
input long               InpZOrder=0;              // マウスクリックの優先順位
//+------------------------------------------------------------------+
//|  ストップサインを作成する                                        |
//+------------------------------------------------------------------+
bool ArrowStopCreate(const long              chart_ID=0,            // チャート識別子
                     const string            name="ArrowStop",      // 記号名
                     const int               sub_window=0,          // サブウィンドウ番号
                     datetime                time=0,                // アンカーポイントの時
刻
                     double                  price=0,               // アンカーポイントの価
格
                     const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // アンカーの種類
                     const color             clr=clrRed,            // 記号の色
                     const ENUM_LINE_STYLE   style=STYLE_SOLID,     // 境界線のスタイル
                     const int               width=3,               // 記号サイズ
                     const bool              back=false,            // 背景で表示する
                     const bool              selection=true,        // 強調表示して移動
                     const bool              hidden=true,           // オブジェクトリストに隠
す
                     const long              z_order=0)             // マウスクリックの優先
順位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeArrowEmptyPoint(time,price);
//--- エラー値をリセットする
   ResetLastError();
//--- 記号を作成する
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_STOP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": failed to create \"Stop\" sign! Error code = ",GetLastError());
      return(false);
     }
//--- アンカーの種類を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- 記号の色を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 境界線のスタイルを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 記号のサイズを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- マウスで記号を移動させるモードを有効(true)か無効(false)にする
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
```

© 2000-2025, MetaQuotes Ltd.

```
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| アンカーポイントを移動する                                       |
//+------------------------------------------------------------------+
bool ArrowStopMove(const long   chart_ID=0,        // チャート識別子
                   const string name="ArrowStop", // オブジェクト名
                   datetime     time=0,           // アンカーポイントの時間座標
                   double       price=0)          // アンカーポイントの価格座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| ストップサインのアンカーの種類を変更する                         |
//+------------------------------------------------------------------+
bool ArrowStopAnchorChange(const long              chart_ID=0,         // チャート識別子
                           const string            name="ArrowStop",   // オブジェクト名
                           const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // アンカーポイント
位置
  {
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーの種類を変更する
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print(__FUNCTION__,
            ": failed to change anchor type! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| ストップサインを削除する                                         |
//+------------------------------------------------------------------+
bool ArrowStopDelete(const long   chart_ID=0,       // チャート識別子
                     const string name="ArrowStop") // ラベル名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 記号を削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete \"Stop\" sign! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| アンカーポイントの値をチェックして                               |
//| 空の物には初期値を設定する                                       |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- ポイントの時間が設定されていない場合、現在足になる
   if(!time)
      time=TimeCurrent();
//--- ポイントの価格が設定されていない場合、売値になる
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 価格配列サイズ
   int accuracy=1000;
//--- アンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
   double   price[];
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 価格配列に書き込む
//--- チャートの最高値と最安値を見つける
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 記号描画のポイントを定義する
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- チャートにストップサインを作成する
   if(!ArrowStopCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- アンカーポイントを移動し、記号との相互位置を変更する
//--- ループカウンタ
   int h_steps=bars*2/5;
//--- アンカーポイントを移動する
   for(int i=0;i<h_steps;i++)
     {
      //--- 次の値を使用する
      if(d<bars-1)
         d+=1;
      //--- ポイントを移動する
      if(!ArrowStopMove(0,InpName,date[d],price[p]))
```

© 2000-2025, MetaQuotes Ltd.

```
            return;
         //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
         if(IsStopped())
            return;
         //--- チャートを再描画する
         ChartRedraw();
         // 0.025 秒の遅れ
         Sleep(25);
        }
//--- 記号に相対したアンカーポイントの位置関係を変更する
   ArrowStopAnchorChange(0,InpName,ANCHOR_TOP);
//--- チャートを再描画する
   ChartRedraw();
//--- ループカウンタ
   h_steps=bars*2/5;
//--- アンカーポイントを移動する
   for(int i=0;i<h_steps;i++)
     {
      //--- 次の値を使用する
      if(d<bars-1)
         d+=1;
      //--- ポイントを移動する
      if(!ArrowStopMove(0,InpName,date[d],price[p]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
      // 0.025 秒の遅れ
      Sleep(25);
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- チャートから記号を削除する
   ArrowStopDelete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_CHECK

チェックサイン



注意事項

記号に相対したアンカーポイントの位置関係は ENUM_ARROW_ANCHOR 列挙から選択出来ます。

5 以上の大きな矢印は MetaEditor でコードを書く際に OBJPROP_WIDTH プロパティ値を設定してのみ作成出来ます。

例

次のスクリプトは、チャート上でチェックサインを作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script draws \"Check\" sign."
#property description "Anchor point coordinate is set in"
#property description "percentage of the chart window size."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string             InpName="ArrowCheck"; // 記号名
input int                InpDate=10;           // アンカーポイントの日付( % )
input int                InpPrice=50;          // アンカーポイントの価格( % )
input ENUM_ARROW_ANCHOR  InpAnchor=ANCHOR_TOP; // アンカーの種類
input color              InpColor=clrRed;      // 記号の色
input ENUM_LINE_STYLE    InpStyle=STYLE_DOT;   // 境界線のスタイル
```

© 2000–2025, MetaQuotes Ltd.

```
input int                InpWidth=5;           // 記号サイズ
input bool               InpBack=false;        // 背景の記号
input bool               InpSelection=false;   // 強調表示して移動
input bool               InpHidden=true;       // オブジェクトリストに隠す
input long               InpZOrder=0;          // マウスクリックの優先順位
//+------------------------------------------------------------------+
//|  チェックサインを作成する                                          |
//+------------------------------------------------------------------+
bool ArrowCheckCreate(const long              chart_ID=0,           // チャート識別子
                      const string            name="ArrowCheck",    // 記号名
                      const int               sub_window=0,         // サブウィンドウ番号
                      datetime                time=0,               // アンカーポイントの
時刻
                      double                  price=0,              // アンカーポイントの
価格
                      const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // アンカーの種類
                      const color             clr=clrRed,           // 記号の色
                      const ENUM_LINE_STYLE   style=STYLE_SOLID,    // 境界線のスタイル
                      const int               width=3,              // 記号サイズ
                      const bool              back=false,           // 背景で表示する
                      const bool              selection=true,       // 強調表示して移動
                      const bool              hidden=true,          // オブジェクトリストに
隠す
                      const long              z_order=0)            // マウスクリックの優先
順位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeArrowEmptyPoint(time,price);
//--- エラー値をリセットする
   ResetLastError();
//--- 記号を作成する
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_CHECK,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": failed to create \"Check\" sign! Error code = ",GetLastError());
      return(false);
     }
//--- アンカーの種類を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- 記号の色を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 境界線のスタイルを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 記号のサイズを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- マウスで記号を移動させるモードを有効(true)か無効(false)にする
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
```

© 2000-2025, MetaQuotes Ltd.

```
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| アンカーポイントを移動する                                       |
//+------------------------------------------------------------------+
bool ArrowCheckMove(const long   chart_ID=0,         // チャート識別子
                    const string name="ArrowCheck", // オブジェクト名
                    datetime     time=0,            // アンカーポイントの時間座標
                    double       price=0)           // アンカーポイントの価格座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| チェックアンカーの種類を変更する                                 |
//+------------------------------------------------------------------+
bool ArrowCheckAnchorChange(const long              chart_ID=0,         // チャート識別子
                            const string            name="ArrowCheck", // オブジェクト名
                            const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // アンカーの種類
  {
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーの種類を変更する
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
```

© 2000-2025, MetaQuotes Ltd.

```
            ": failed to change anchor type! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| チェックサインを削除する                                         |
//+------------------------------------------------------------------+
bool ArrowCheckDelete(const long   chart_ID=0,        // チャート識別子
                      const string name="ArrowCheck") // 記号名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 記号を削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete \"Check\" sign! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| アンカーポイントの値をチェックして                               |
//| 空の物には初期値を設定する                                       |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- ポイントの時間が設定されていない場合、現在足になる
   if(!time)
      time=TimeCurrent();
//--- ポイントの価格が設定されていない場合、売値になる
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- チャートウィンドウで表示されているバーの数
```

© 2000-2025, MetaQuotes Ltd.

```
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 価格配列サイズ
   int accuracy=1000;
//--- アンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
   double   price[];
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 価格配列に書き込む
//--- チャートの最高値と最安値を見つける
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 記号描画のポイントを定義する
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- チャートにチェックサインを作成する
   if(!ArrowCheckCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- アンカーポイントを移動し、記号との相互位置を変更する
//--- ループカウンタ
   int h_steps=bars*2/5;
//--- アンカーポイントを移動する
   for(int i=0;i<h_steps;i++)
     {
      //--- 次の値を使用する
      if(d<bars-1)
         d+=1;
      //--- ポイントを移動する
      if(!ArrowCheckMove(0,InpName,date[d],price[p]))
         return;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
      // 0.025 秒の遅れ
      Sleep(25);
     }
//--- 記号に相対したアンカーポイントの位置関係を変更する
   ArrowCheckAnchorChange(0,InpName,ANCHOR_BOTTOM);
//--- チャートを再描画する
   ChartRedraw();
//--- ループカウンタ
   h_steps=bars*2/5;
//--- アンカーポイントを移動する
   for(int i=0;i<h_steps;i++)
     {
      //--- 次の値を使用する
      if(d<bars-1)
         d+=1;
      //--- ポイントを移動する
      if(!ArrowCheckMove(0,InpName,date[d],price[p]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
      // 0.025 秒の遅れ
      Sleep(25);
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- チャートから記号を削除する
   ArrowCheckDelete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//---
  }
```

© 2000–2025, MetaQuotes Ltd.

OBJ_ARROW_LEFT_PRICE

左プライスラベル。



例

次のスクリプトは、チャート上で左プライスラベルを作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script creates the left price label on the chart."
#property description "Anchor point coordinate is set in"
#property description "percentage of the chart window size."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string            InpName="LeftPrice";  // 価格ラベル名
input int               InpDate=100;          // アンカーポイントの日付( % )
input int               InpPrice=10;          // アンカーポイントの価格( % )
input color             InpColor=clrRed;      // 価格ラベルの色
input ENUM_LINE_STYLE   InpStyle=STYLE_SOLID; // 境界線のスタイル
input int               InpWidth=2;           // 価格ラベルのサイズ
input bool              InpBack=false;        // 背景ラベル
input bool              InpSelection=true;    // 強調表示して移動
input bool              InpHidden=true;       // オブジェクトリストに隠す
input long              InpZOrder=0;          // マウスクリックの優先順位
//+------------------------------------------------------------------+
//|  左プライスラベルを作成する                                      |
```

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool ArrowLeftPriceCreate(const long            chart_ID=0,        // チャート識別子
                          const string          name="LeftPrice",  // 価格ラベル名
                          const int             sub_window=0,      // サブウィンドウ番号
                          datetime              time=0,            // アンカーポイントの時刻
                          double                price=0,           // アンカーポイントの価格
                          const color           clr=clrRed,        // 価格ラベルの色
                          const ENUM_LINE_STYLE style=STYLE_SOLID, // 境界線のスタイル
                          const int             width=1,           // 価格ラベルのサイズ
                          const bool            back=false,        // 背景で表示する
                          const bool            selection=true,    // 強調表示して移動
                          const bool            hidden=true,       // オブジェクトリストに隠す
                          const long            z_order=0)         // マウスクリックの優先順位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeArrowEmptyPoint(time,price);
//--- エラー値をリセットする
   ResetLastError();
//--- 価格ラベルを作成する
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_LEFT_PRICE,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": failed to create the left price label! Error code = ",GetLastError());
      return(false);
     }
//--- ラベルの色を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 境界線のスタイルを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- ラベルサイズを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- マウスでラベルを移動させるモードを有効(true)か無効(false)にする
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//|  アンカーポイントを移動する                                      |
//+------------------------------------------------------------------+
bool ArrowLeftPriceMove(const long   chart_ID=0,        // チャート識別子
                        const string name="LeftPrice", // ラベル名
                        datetime     time=0,           // アンカーポイントの時間座標
                        double       price=0)          // アンカーポイントの価格座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  左プライスラベルをチャートから削除する                          |
//+------------------------------------------------------------------+
bool ArrowLeftPriceDelete(const long   chart_ID=0,        // チャート識別子
                          const string name="LeftPrice") // ラベル名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- ラベルを削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete the left price label! Error code =
",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  アンカーポイントの値をチェックして                              |
//|  空の物には初期値を設定する                                      |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- ポイントの時間が設定されていない場合、現在足になる
   if(!time)
      time=TimeCurrent();
//--- ポイントの価格が設定されていない場合、売値になる
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 価格配列サイズ
   int accuracy=1000;
//--- アンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
   double   price[];
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 価格配列に書き込む
//--- チャートの最高値と最安値を見つける
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- ラベル描画のポイントを定義する
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 左プライスラベルをチャートに作成する
   if(!ArrowLeftPriceCreate(0,InpName,0,date[d],price[p],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- アンカーポイントを移動する
//--- ループカウンタ
   int v_steps=accuracy*4/5;
//--- アンカーポイントを移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p<accuracy-1)
         p+=1;
      //--- ポイントを移動する
      if(!ArrowLeftPriceMove(0,InpName,date[d],price[p]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- チャートからラベルを削除する
   ArrowLeftPriceDelete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//---
  }
```

© 2000–2025, MetaQuotes Ltd.

OBJ_ARROW_RIGHT_PRICE

右 プライスラベル



例

次 のスクリプト は、チャート 上 で右 プライスラベルを作 成 して移 動 します。グラフィックオブジェクト のプロパティ を作 成 そして変 更 するために特 別 な関 数 が開 発 されています。これらの関 数 はお使 いのアプリケーション でそのまま使 用 することが出 来 ます。

```
//--- 説明
#property description "Script creates the right price label on the chart."
#property description "Anchor point coordinate is set in"
#property description "percentage of the chart window size."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string            InpName="RightPrice"; // 価格ラベル名
input int               InpDate=0;            // アンカーポイントの日付( % )
input int               InpPrice=90;          // アンカーポイントの価格( % )
input color             InpColor=clrRed;      // 価格ラベルの色
input ENUM_LINE_STYLE   InpStyle=STYLE_SOLID; // 境界線のスタイル
input int               InpWidth=2;           // 価格ラベルのサイズ
input bool              InpBack=false;        // 背景ラベル
input bool              InpSelection=true;    // 強調表示して移動
input bool              InpHidden=true;       // オブジェクトリストに隠す
input long              InpZOrder=0;          // マウスクリックの優先順位
//+------------------------------------------------------------------+
//|  右プライスラベルを作成する                                      |
```

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool ArrowRightPriceCreate(const long            chart_ID=0,        // チャート識別子
                           const string          name="RightPrice", // 価格ラベル名
                           const int             sub_window=0,      // サブウィンドウ番号
                           datetime              time=0,            // アンカーポイントの
時刻
                           double                price=0,           // アンカーポイントの
価格
                           const color           clr=clrRed,        // 価格ラベルの色
                           const ENUM_LINE_STYLE style=STYLE_SOLID, // 境界線のスタイル
                           const int             width=1,           // 価格ラベルのサイズ
                           const bool            back=false,        // 背景で表示する
                           const bool            selection=true,    // 強調表示して移動
                           const bool            hidden=true,       // オブジェクトリストに
隠す
                           const long            z_order=0)         // マウスクリックの優先
順位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeArrowEmptyPoint(time,price);
//--- エラー値をリセットする
   ResetLastError();
//--- 価格ラベルを作成する
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_RIGHT_PRICE,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": failed to create the right price label! Error code =
",GetLastError());
      return(false);
     }
//--- ラベルの色を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 境界線のスタイルを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- ラベルサイズを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- マウスでラベルを移動させるモードを有効(true)か無効(false)にする
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| アンカーポイントを移動する                                       |
//+------------------------------------------------------------------+
bool ArrowRightPriceMove(const long   chart_ID=0,        // チャート識別子
                         const string name="RightPrice", // ラベル名
                         datetime     time=0,            // アンカーポイントの時間座標
                         double       price=0)           // アンカーポイントの価格座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 右プライスラベルをチャートから削除する                           |
//+------------------------------------------------------------------+
bool ArrowRightPriceDelete(const long   chart_ID=0,        // チャート識別子
                           const string name="RightPrice") // ラベル名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- ラベルを削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete the right price label! Error code =
",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| アンカーポイントの値をチェックして                               |
//| 空の物には初期値を設定する                                       |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- ポイントの時間が設定されていない場合、現在足になる
   if(!time)
      time=TimeCurrent();
//--- ポイントの価格が設定されていない場合、売値になる
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 価格配列サイズ
   int accuracy=1000;
//--- アンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
   double   price[];
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 価格配列に書き込む
//--- チャートの最高値と最安値を見つける
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- ラベル描画のポイントを定義する
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
```

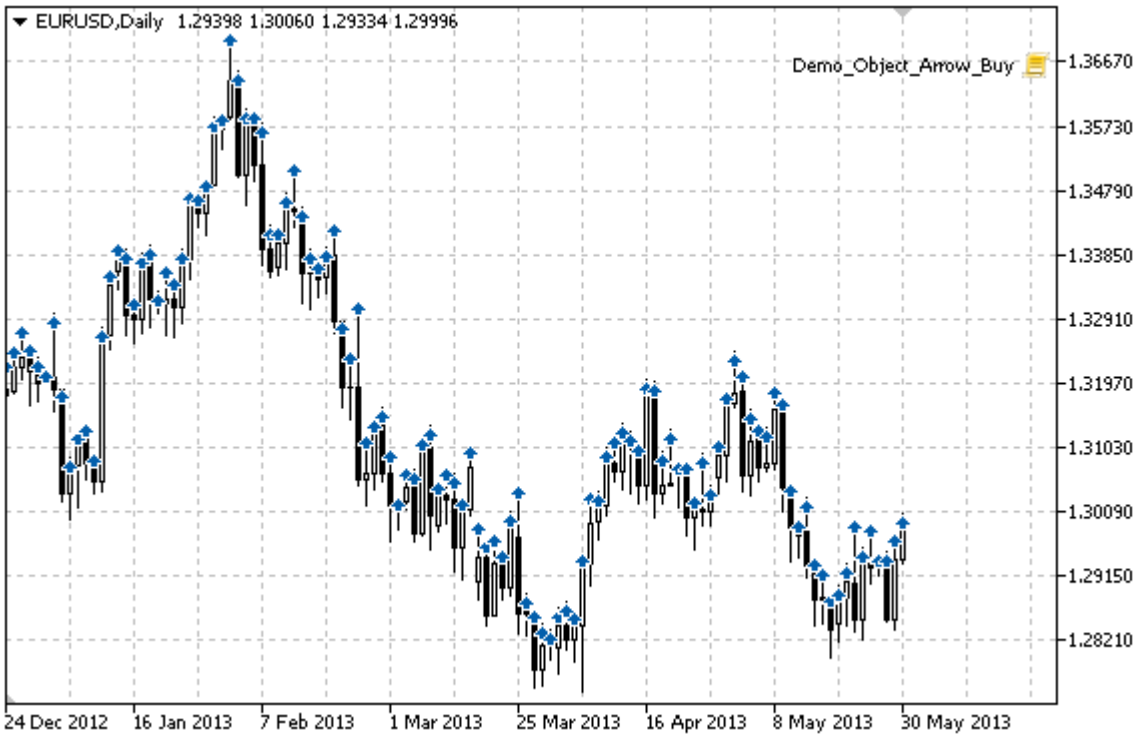
© 2000-2025, MetaQuotes Ltd.

```
//--- 右プライスラベルをチャートに作成する
   if(!ArrowRightPriceCreate(0,InpName,0,date[d],price[p],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- アンカーポイントを移動する
//--- ループカウンタ
   int v_steps=accuracy*4/5;
//--- アンカーポイントを移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 次の値を使用する
      if(p>1)
         p-=1;
      //--- ポイントを移動する
      if(!ArrowRightPriceMove(0,InpName,date[d],price[p]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- チャートからラベルを削除する
   ArrowRightPriceDelete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_BUY

買いサイン



例

次のスクリプトは、チャート上で買いサインを作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script draws \"Buy\" signs in the chart window."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input color InpColor=C'3,95,172'; // 記号の色
//+------------------------------------------------------------------+
//|  買いサインを作成する                                            |
//+------------------------------------------------------------------+
bool ArrowBuyCreate(const long            chart_ID=0,        // チャート識別子
                    const string          name="ArrowBuy",   // 記号名
                    const int             sub_window=0,      // サブウィンドウ番号
                    datetime              time=0,            // アンカーポイントの時刻
                    double                price=0,           // アンカーポイントの価格
                    const color           clr=C'3,95,172',   // 記号の色
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // 強調表示時の線のスタイル
                    const int             width=1,           // 強調表示時の線のサイズ
                    const bool            back=false,        // 背景で表示する
                    const bool            selection=false,   // 強調表示して移動
```

© 2000-2025, MetaQuotes Ltd.

```
                     const bool            hidden=true,        // オブジェクトリストに隠す
                     const long            z_order=0)          // マウスクリックの優先順位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeArrowEmptyPoint(time,price);
//--- エラー値をリセットする
   ResetLastError();
//--- 記号を作成する
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_BUY,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": failed to create \"Buy\" sign! Error code = ",GetLastError());
      return(false);
     }
//--- 記号の色を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 強調表示時の線のスタイルを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 強調表示時の線のサイズを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- マウスで記号を移動させるモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| アンカーポイントを移動する                                       |
//+------------------------------------------------------------------+
bool ArrowBuyMove(const long   chart_ID=0,       // チャート識別子
                  const string name="ArrowBuy", // オブジェクト名
                  datetime     time=0,          // アンカーポイントの時間座標
                  double       price=0)         // アンカーポイントの価格座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,0,time,price))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print(__FUNCTION__,
             ": failed to move the anchor point! Error code = ",GetLastError());
       return(false);
      }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  買いサインを削除する                                            |
//+------------------------------------------------------------------+
bool ArrowBuyDelete(const long   chart_ID=0,       // チャート識別子
                    const string name="ArrowBuy") // 記号名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 記号を削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete \"Buy\" sign! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  アンカーポイントの値をチェックして                              |
//|  空の物には初期値を設定する                                      |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- ポイントの時間が設定されていない場合、現在足になる
   if(!time)
      time=TimeCurrent();
//--- ポイントの価格が設定されていない場合、売値になる
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                              |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date[]; // 表示中のバーの日付を格納するための配列
   double   low[];  // 表示中の足の安値を格納するための配列
   double   high[]; // 表示中の足の高値を格納するための配列
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

© 2000–2025, MetaQuotes Ltd.

```
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(low,bars);
   ArrayResize(high,bars);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 安値の配列に書き込む
   if(CopyLow(Symbol(),Period(),0,bars,low)==-1)
     {
      Print("Failed to copy the values of Low prices! Error code =
",GetLastError());
      return;
     }
//--- 高値の配列に書き込む
   if(CopyHigh(Symbol(),Period(),0,bars,high)==-1)
     {
      Print("Failed to copy the values of High prices! Error code =
",GetLastError());
      return;
     }
//--- 表示された各足の低ポイントで買い記号を作成する
   for(int i=0;i<bars;i++)
     {
      if(!ArrowBuyCreate(0,"ArrowBuy_"+(string)i,0,date[i],low[i],InpColor))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
      // 0.05 秒の遅れ
      Sleep(50);
     }
//--- 表示された各足の買い記号を高ポイントに移動する
   for(int i=0;i<bars;i++)
     {
      if(!ArrowBuyMove(0,"ArrowBuy_"+(string)i,date[i],high[i]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
      // 0.05 秒の遅れ
```

© 2000–2025, MetaQuotes Ltd.

```
       Sleep(50);
      }
//--- 買いサインを削除する
   for(int i=0;i<bars;i++)
      {
       if(!ArrowBuyDelete(0,"ArrowBuy_"+(string)i))
          return;
       //--- チャートを再描画する
       ChartRedraw();
       // 0.05 秒の遅れ
       Sleep(50);
      }
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_SELL

売りサイン



例

次のスクリプトは、チャート上で売りサインを作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script draws \"Sell\" signs in the chart window."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input color InpColor=C'225,68,29'; // 記号の色
//+------------------------------------------------------------------+
//|  売りサインを作成する                                            |
//+------------------------------------------------------------------+
bool ArrowSellCreate(const long            chart_ID=0,        // チャート識別子
                     const string          name="ArrowSell",  // 記号名
                     const int             sub_window=0,      // サブウィンドウ番号
                     datetime              time=0,            // アンカーポイントの時刻
                     double                price=0,           // アンカーポイントの価格
                     const color           clr=C'225,68,29',  // 記号の色
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // 強調表示時の線のスタイル
                     const int             width=1,           // 強調表示時の線のサイズ
                     const bool            back=false,        // 背景で表示する
                     const bool            selection=false,   // 強調表示して移動
```

© 2000-2025, MetaQuotes Ltd.

```
                      const bool            hidden=true,        // オブジェクトリストに隠す
                      const long            z_order=0)          // マウスクリックの優先順位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeArrowEmptyPoint(time,price);
//--- エラー値をリセットする
   ResetLastError();
//--- 記号を作成する
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_SELL,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": failed to create \"Sell\" sign! Error code = ",GetLastError());
      return(false);
     }
//--- 記号の色を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 強調表示時の線のスタイルを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 強調表示時の線のサイズを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- マウスで記号を移動させるモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  アンカーポイントを移動する                                      |
//+------------------------------------------------------------------+
bool ArrowSellMove(const long   chart_ID=0,        // チャート識別子
                   const string name="ArrowSell", // オブジェクト名
                   datetime     time=0,           // アンカーポイントの時間座標
                   double       price=0)          // アンカーポイントの価格座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,0,time,price))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print(__FUNCTION__,
             ": failed to move the anchor point! Error code = ",GetLastError());
       return(false);
      }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  売りサインを作成する                                            |
//+------------------------------------------------------------------+
bool ArrowSellDelete(const long   chart_ID=0,       // チャート識別子
                     const string name="ArrowSell") // 記号名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 記号を削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete \"Sell\" sign! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  アンカーポイントの値をチェックして                              |
//|  空の物には初期値を設定する                                      |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- ポイントの時間が設定されていない場合、現在足になる
   if(!time)
      time=TimeCurrent();
//--- ポイントの価格が設定されていない場合、売値になる
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                              |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date[]; // 表示中のバーの日付を格納するための配列
   double   low[];  // 表示中の足の安値を格納するための配列
   double   high[]; // 表示中の足の高値を格納するための配列
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

© 2000–2025, MetaQuotes Ltd.

```
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(low,bars);
   ArrayResize(high,bars);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 安値の配列に書き込む
   if(CopyLow(Symbol(),Period(),0,bars,low)==-1)
     {
      Print("Failed to copy the values of Low prices! Error code =
",GetLastError());
      return;
     }
//--- 高値の配列に書き込む
   if(CopyHigh(Symbol(),Period(),0,bars,high)==-1)
     {
      Print("Failed to copy the values of High prices! Error code =
",GetLastError());
      return;
     }
//--- 表示された各足の高ポイントで売り記号を作成する
   for(int i=0;i<bars;i++)
     {
      if(!ArrowSellCreate(0,"ArrowSell_"+(string)i,0,date[i],high[i],InpColor))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
      // 0.05 秒の遅れ
      Sleep(50);
     }
//--- 表示された各足の売り記号を低ポイントに移動する
   for(int i=0;i<bars;i++)
     {
      if(!ArrowSellMove(0,"ArrowSell_"+(string)i,date[i],low[i]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
      // 0.05 秒の遅れ
```

© 2000-2025, MetaQuotes Ltd.

```
       Sleep(50);
      }
//--- 売りサインを削除する
   for(int i=0;i<bars;i++)
      {
       if(!ArrowSellDelete(0,"ArrowSell_"+(string)i))
          return;
       //--- チャートを再描画する
       ChartRedraw();
       // 0.05 秒の遅れ
       Sleep(50);
      }
//---
  }
```

© 2000–2025, MetaQuotes Ltd.

OBJ_ARROW

矢印オブジェクト



注意事項

オブジェクトのアンカーポイントの位置関係はENUM_ARROW_ANCHORから選択することが出来ます。

5以上の大きな矢印はMetaEditorでコードを書く際にOBJPROP_WIDTHプロパティ値を設定してのみ作成出来ます。

必要な矢印タイプはワインディングフォントの記号コードのいずれかを設定して選択出来ます。

例

次のスクリプトは、チャート上で矢印オブジェクトを作成し、その型を変更します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script creates a random arrow in the chart window."
#property description "Anchor point coordinate is set in"
#property description "percentage of the chart window size."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string             InpName="Arrow";         // 矢印名
input int                InpDate=50;              // アンカーポイントの日付( % )
input int                InpPrice=50;             // アンカーポイントの価格( % )
input ENUM_ARROW_ANCHOR  InpAnchor=ANCHOR_TOP;    // アンカーの種類
```

© 2000-2025, MetaQuotes Ltd.

```
input color             InpColor=clrDodgerBlue; // 矢印の色
input ENUM_LINE_STYLE   InpStyle=STYLE_SOLID;   // 境界線のスタイル
input int               InpWidth=10;            // 矢印のサイズ
input bool              InpBack=false;          // 背景の→
input bool              InpSelection=false;     // 強調表示して移動
input bool              InpHidden=true;         // オブジェクトリストに隠す
input long              InpZOrder=0;            // マウスクリックの優先順位
//+------------------------------------------------------------------+
//| 矢印の作成                                                       |
//+------------------------------------------------------------------+
bool ArrowCreate(const long              chart_ID=0,           // チャート識別子
                 const string            name="Arrow",         // 矢印名
                 const int               sub_window=0,         // サブウィンドウ番号
                 datetime                time=0,               // アンカーポイントの時刻
                 double                  price=0,              // アンカーポイントの価格
                 const uchar             arrow_code=252,       // 矢印コード
                 const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // アンカーポイント位置
                 const color             clr=clrRed,           // 矢印の色
                 const ENUM_LINE_STYLE   style=STYLE_SOLID,    // 境界線のスタイル
                 const int               width=3,              // 矢印のサイズ
                 const bool              back=false,           // 背景で表示する
                 const bool              selection=true,       // 強調表示して移動
                 const bool              hidden=true,          // オブジェクトリストに隠す
                 const long              z_order=0)            // マウスクリックの優先順位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeArrowEmptyPoint(time,price);
//--- エラー値をリセットする
   ResetLastError();
//--- 矢印を作成
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": failed to create an arrow! Error code = ",GetLastError());
      return(false);
     }
//--- 矢印コードを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_ARROWCODE,arrow_code);
//--- アンカーの種類を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- 矢印の色を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 境界線のスタイルを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 矢印のサイズを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- マウスで矢印を移動させるモードを有効(true)か無効(false)にする
```

© 2000-2025, MetaQuotes Ltd.

```
//--- ObjectCreate 関数を使用してグラフィックオブジェクトを作成する際、オブジェクトは
//--- デフォルトではハイライトされたり動かされたり出来ない。このメソッド内では、選択パラメータは
//--- デフォルトでは true でハイライトと移動を可能にする。
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| アンカーポイントを移動する                                       |
//+------------------------------------------------------------------+
bool ArrowMove(const long   chart_ID=0,   // チャート識別子
               const string name="Arrow", // オブジェクト名
               datetime     time=0,       // アンカーポイントの時間座標
               double       price=0)      // アンカーポイントの価格座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 矢印コードを変更                                                 |
//+------------------------------------------------------------------+
bool ArrowCodeChange(const long   chart_ID=0,   // チャート識別子
                     const string name="Arrow", // オブジェクト名
                     const uchar  code=252)     // 矢印コード
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 矢印コードを変更する
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ARROWCODE,code))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print(__FUNCTION__,
            ": failed to change the arrow code! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| アンカーの種類を変更する                                         |
//+------------------------------------------------------------------+
bool ArrowAnchorChange(const long              chart_ID=0,        // チャート識別子
                       const string            name="Arrow",      // オブジェクト名
                       const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // アンカーの種類
  {
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーの種類を変更する
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": failed to change anchor type! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 矢印を削除する                                                   |
//+------------------------------------------------------------------+
bool ArrowDelete(const long   chart_ID=0,   // チャート識別子
                 const string name="Arrow") // 矢印名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 矢印を削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete an arrow! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| アンカーポイントの値をチェックして                               |
//| 空の物には初期値を設定する                                       |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- ポイントの時間が設定されていない場合、現在足になる
   if(!time)
      time=TimeCurrent();
//--- ポイントの価格が設定されていない場合、売値になる
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 価格配列サイズ
   int accuracy=1000;
//--- アンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
   double   price[];
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 価格配列に書き込む
//--- チャートの最高値と最安値を見つける
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 価格の変更ステップを定義し、配列に書き込む
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 矢印描画のポイントを定義する
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- チャートに矢印を作成する
```

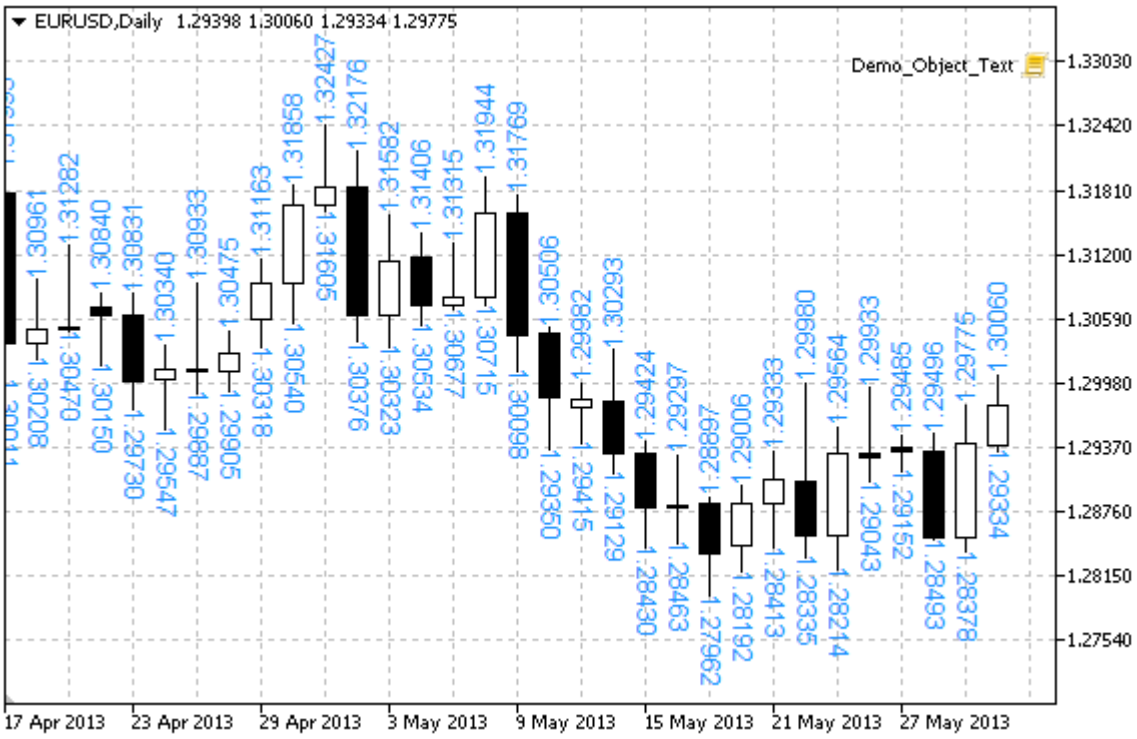
© 2000-2025, MetaQuotes Ltd.

```
   if(!ArrowCreate(0,InpName,0,date[d],price[p],32,InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画する
   ChartRedraw();
//--- ループ内での矢印作成の全てのケースを検討する
   for(int i=33;i<256;i++)
     {
      if(!ArrowCodeChange(0,InpName,(uchar)i))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
      // 半秒の遅れ
      Sleep(500);
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- チャートから矢印を削除する
   ArrowDelete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//---
  }
```

© 2000–2025, MetaQuotes Ltd.

OBJ_TEXT

テキストオブジェクト



注意事項

テキストに相対したアンカーポイントの位置関係は ENUM_ANCHOR_POINT 列挙から選択出来ます。また、テキストの傾斜角度は OBJPROP_ANGLE プロパティで変更出来ます。

例

次のスクリプトは、チャート上で複数のテキストオブジェクトを作成します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script creates \"Text\" graphical object."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string            InpFont="Arial";          // フォント
input int               InpFontSize=10;           // フォントサイズ
input color             InpColor=clrRed;          // 色
input double            InpAngle=90.0;            // 傾斜角度
input ENUM_ANCHOR_POINT InpAnchor=ANCHOR_LEFT;    // アンカーの種類
input bool              InpBack=false;            // 背景オブジェクト
input bool              InpSelection=false;       // 強調表示して移動
input bool              InpHidden=true;           // オブジェクトリストに隠す
input long              InpZOrder=0;              // マウスクリックの優先順位
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  テキストオブジェクトを作成する                                  |
//+------------------------------------------------------------------+
bool TextCreate(const long              chart_ID=0,                 // チャート識別子
                const string            name="Text",                // オブジェクト名
                const int               sub_window=0,               // サブウィンドウ番号
                datetime                time=0,                     // アンカーポイントの時刻
                double                  price=0,                    // アンカーポイントの価格
                const string            text="Text",                // テキスト
                const string            font="Arial",               // フォント
                const int               font_size=10,               // フォントサイズ
                const color             clr=clrRed,                 // 色
                const double            angle=0.0,                  // テキストの傾斜
                const ENUM_ANCHOR_POINT anchor=ANCHOR_LEFT_UPPER,   // アンカーの種類
                const bool              back=false,                 // 背景で表示する
                const bool              selection=false,            // 強調表示して移動
                const bool              hidden=true,                // ブジェクトリストに隠れて
いる
                const long              z_order=0)                  // マウスクリックの優先順
位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeTextEmptyPoint(time,price);
//--- エラー値をリセットする
   ResetLastError();
//--- テキストオブジェクトを作成する
   if(!ObjectCreate(chart_ID,name,OBJ_TEXT,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": failed to create \"Text\" object! Error code = ",GetLastError());
      return(false);
     }
//--- テキストを設定する
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- テキストフォントを設定する
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
//--- フォントサイズを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- テキストの傾斜を設定する
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- アンカーの種類を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- 色を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- マウスでオブジェクトを移動させるモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| アンカーポイントを移動する                                       |
//+------------------------------------------------------------------+
bool TextMove(const long   chart_ID=0,  // チャート識別子
              const string name="Text", // オブジェクト名
              datetime     time=0,      // アンカーポイントの時間座標
              double       price=0)     // アンカーポイントの価格座標
  {
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| オブジェクトのテキストを変更する                                 |
//+------------------------------------------------------------------+
bool TextChange(const long   chart_ID=0,  // チャート識別子
                const string name="Text", // オブジェクト名
                const string text="Text") // テキスト
  {
//--- エラー値をリセットする
   ResetLastError();
//--- オブジェクトのテキストを変更する
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": failed to change the text! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| テキストオブジェクトを削除する                                   |
//+------------------------------------------------------------------+
bool TextDelete(const long   chart_ID=0,  // チャート識別子
                const string name="Text") // オブジェクト名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- オブジェクトを削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete \"Text\" object! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| アンカーポイントの値をチェックして                               |
//| 空の物には初期値を設定する                                       |
//+------------------------------------------------------------------+
void ChangeTextEmptyPoint(datetime &time,double &price)
  {
//--- ポイントの時間が設定されていない場合、現在足になる
   if(!time)
      time=TimeCurrent();
//--- ポイントの価格が設定されていない場合、売値になる
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                               |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date[]; // 表示中のバーの日付を格納するための配列
   double   low[];  // 表示中の足の安値を格納するための配列
   double   high[]; // 表示中の足の高値を格納するための配列
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(low,bars);
   ArrayResize(high,bars);
//--- 日付配列に書き込む
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 安値の配列に書き込む
   if(CopyLow(Symbol(),Period(),0,bars,low)==-1)
     {
      Print("Failed to copy the values of Low prices! Error code =
",GetLastError());
      return;
     }
//--- 高値の配列に書き込む
   if(CopyHigh(Symbol(),Period(),0,bars,high)==-1)
     {
      Print("Failed to copy the values of High prices! Error code =
",GetLastError());
      return;
     }
//--- テキスト表示の頻度を定義する
   int scale=(int)ChartGetInteger(0,CHART_SCALE);
//--- ステップを定義する
   int step=1;
   switch(scale)
     {
      case 0:
         step=12;
         break;
      case 1:
         step=6;
         break;
      case 2:
         step=4;
         break;
      case 3:
         step=2;
         break;
     }
//--- 高値と安値のバーの値のテキストを作成する(ギャップで)
   for(int i=0;i<bars;i+=step)
     {
      //--- テキストを作成する
      if(!TextCreate(0,"TextHigh_"+(string)
i,0,date[i],high[i],DoubleToString(high[i],5),InpFont,InpFontSize,
         InpColor,InpAngle,InpAnchor,InpBack,InpSelection,InpHidden,InpZOrder))
        {
         return;
        }
```

© 2000–2025, MetaQuotes Ltd.

```
      if(!TextCreate(0,"TextLow_"+(string)
i,0,date[i],low[i],DoubleToString(low[i],5),InpFont,InpFontSize,
          InpColor,-InpAngle,InpAnchor,InpBack,InpSelection,InpHidden,InpZOrder))
         {
          return;
         }
       //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
       if(IsStopped())
          return;
       //--- チャートを再描画する
       ChartRedraw();
       // 0.05 秒の遅れ
       Sleep(50);
      }
//--- 半秒の遅れ
   Sleep(500);
//--- テキストを削除する
   for(int i=0;i<bars;i+=step)
     {
      if(!TextDelete(0,"TextHigh_"+(string)i))
         return;
      if(!TextDelete(0,"TextLow_"+(string)i))
         return;
      //--- チャートを再描画する
      ChartRedraw();
      // 0.05 秒の遅れ
      Sleep(50);
     }
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_LABEL

ラベルオブジェクト



注意事項

ラベルに相対したアンカーポイントの位置関係は ENUM_ANCHOR_POINT 列挙から選択出来ます。アンカーポイントの座標はピクセルで設定されています。

ラベルのアンカーポイントとして使用される隅は ENUM_BASE_CORNER 列挙から選択出来ます。

例

次のスクリプトは、チャート上で編集オブジェクトを作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script creates \"Label\" graphical object."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string             InpName="Label";          // ラベル名
input int                InpX=150;                 // X 軸距離
input int                InpY=150;                 // Y 軸距離
input string             InpFont="Arial";          // フォント
input int                InpFontSize=14;           // フォントサイズ
input color              InpColor=clrRed;          // 色
input double             InpAngle=0.0;             // 傾斜角度
input ENUM_ANCHOR_POINT InpAnchor=ANCHOR_CENTER; // アンカーの種類
```

© 2000-2025, MetaQuotes Ltd.

```
input bool               InpBack=false;           // 背景オブジェクト
input bool               InpSelection=true;       // 強調表示して移動
input bool               InpHidden=true;          // オブジェクトリストに隠す
input long               InpZOrder=0;             // マウスクリックの優先順位
//+------------------------------------------------------------------+
//| ラベルを作成する                                                 |
//+------------------------------------------------------------------+
bool LabelCreate(const long              chart_ID=0,               // チャート識別子
                 const string            name="Label",             // ラベル名
                 const int               sub_window=0,             // サブウィンドウ番号
                 const int               x=0,                      // X 座標
                 const int               y=0,                      // Y 座標
                 const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // アンカーに使用される
チャートのコーナー
                 const string            text="Label",             // テキスト
                 const string            font="Arial",             // フォント
                 const int               font_size=10,             // フォントサイズ
                 const color             clr=clrRed,               // 色
                 const double            angle=0.0,                // テキストの傾斜
                 const ENUM_ANCHOR_POINT anchor=ANCHOR_LEFT_UPPER, // アンカーの種類
                 const bool              back=false,               // 背景で表示する
                 const bool              selection=false,          // 強調表示して移動
                 const bool              hidden=true,              // オブジェクトリストに隠
す
                 const long              z_order=0)                // マウスクリックの優先
順位
  {
//--- エラー値をリセットする
   ResetLastError();
//--- ラベルを作成する
   if(!ObjectCreate(chart_ID,name,OBJ_LABEL,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": failed to create text label! Error code = ",GetLastError());
      return(false);
     }
//--- ラベル座標を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- ポイント座標が相対的に定義されているチャートのコーナーを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- テキストを設定する
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- テキストフォントを設定する
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
//--- フォントサイズを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- テキストの傾斜を設定する
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
```

© 2000–2025, MetaQuotes Ltd.

```
//--- アンカーの種類を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- 色を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- マウスでラベルを移動させるモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| ラベルを移動する                                                 |
//+------------------------------------------------------------------+
bool LabelMove(const long   chart_ID=0,   // チャート識別子
               const string name="Label", // ラベル名
               const int    x=0,          // X 座標
               const int    y=0)          // Y 座標
  {
//--- エラー値をリセットする
   ResetLastError();
//--- ラベルを移動する
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": failed to move X coordinate of the label! Error code =
",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": failed to move Y coordinate of the label! Error code =
",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| ラベル結合に使用されるチャートのコーナーを変更する               |
//+------------------------------------------------------------------+
bool LabelChangeCorner(const long             chart_ID=0,              // チャート識別
子
```

© 2000-2025, MetaQuotes Ltd.

```
                        const string           name="Label",              // ラベル名
                        const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER) // アンカーに使
用されるチャートのコーナー
  {
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーに使用される隅を変更する
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner))
     {
      Print(__FUNCTION__,
            ": failed to change the anchor corner! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| ラベルテキストを変更する                                         |
//+------------------------------------------------------------------+
bool LabelTextChange(const long   chart_ID=0,   // チャート識別子
                     const string name="Label", // オブジェクト名
                     const string text="Text")  // テキスト
  {
//--- エラー値をリセットする
   ResetLastError();
//--- オブジェクトのテキストを変更する
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": failed to change the text! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| ラベルを削除する                                                 |
//+------------------------------------------------------------------+
bool LabelDelete(const long   chart_ID=0,   // チャート識別子
                 const string name="Label") // ラベル名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- ラベルを削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete a text label! Error code = ",GetLastError());
      return(false);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- ラベルの座標をローカル変数に収納する
   int x=InpX;
   int y=InpY;
//--- チャートウィンドウサイズ
   long x_distance;
   long y_distance;
//--- ウィンドウサイズを設定する
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Failed to get the chart width! Error code = ",GetLastError());
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("Failed to get the chart height! Error code = ",GetLastError());
      return;
     }
//--- 入力パラメータの正しさをチェックする
   if(InpX<0 || InpX>x_distance-1 || InpY<0 || InpY>y_distance-1)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- ラベルの初期テキストを準備する
   string text;
   StringConcatenate(text,"Upper left corner: ",x,",",y);
//--- チャートにラベルを作成する
   if(!LabelCreate(0,InpName,0,InpX,InpY,CORNER_LEFT_UPPER,text,InpFont,InpFontSize,
      InpColor,InpAngle,InpAnchor,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して1 秒待つ
   ChartRedraw();
   Sleep(500);
//--- ラベルを移動すると同時に、そのテキストを変更する
//--- 軸による反復回数
   int h_steps=(int)(x_distance/2-InpX);
   int v_steps=(int)(y_distance/2-InpY);
//--- ラベルを下に移動する
```

© 2000–2025, MetaQuotes Ltd.

```
   for(int i=0;i<v_steps;i++)
     {
      //--- 座標を変更する
      y+=2;
      //--- ラベルを移動してテキストを変更する
      MoveAndTextChange(x,y,"Upper left corner: ");
     }
//--- 半秒の遅れ
   Sleep(500);
//--- ラベルを右に移動する
   for(int i=0;i<h_steps;i++)
     {
      //--- 座標を変更する
      x+=2;
      //--- ラベルを移動してテキストを変更する
      MoveAndTextChange(x,y,"Upper left corner: ");
     }
//--- 半秒の遅れ
   Sleep(500);
//--- ラベルを上に移動する
   for(int i=0;i<v_steps;i++)
     {
      //--- 座標を変更する
      y-=2;
      //--- ラベルを移動してテキストを変更する
      MoveAndTextChange(x,y,"Upper left corner: ");
     }
//--- 半秒の遅れ
   Sleep(500);
//--- ラベルを左に移動する
   for(int i=0;i<h_steps;i++)
     {
      //--- 座標を変更する
      x-=2;
      //--- ラベルを移動してテキストを変更する
      MoveAndTextChange(x,y,"Upper left corner: ");
     }
//--- 半秒の遅れ
   Sleep(500);
//--- アンカーに使用される隅を変更してポイントを移動する
//--- 左下の隅に移動する
   if(!LabelChangeCorner(0,InpName,CORNER_LEFT_LOWER))
      return;
//--- ラベルテキストを変更する
   StringConcatenate(text,"Lower left corner: ",x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return;
//--- チャートを再描画して 2 秒待つ
   ChartRedraw();
```

© 2000-2025, MetaQuotes Ltd.

```
   Sleep(2000);
//--- 右下の隅に移動する
   if(!LabelChangeCorner(0,InpName,CORNER_RIGHT_LOWER))
      return;
//--- ラベルテキストを変更する
   StringConcatenate(text,"Lower right corner: ",x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return;
//--- チャートを再描画して 2 秒待つ
   ChartRedraw();
   Sleep(2000);
//--- 右上の隅に移動する
   if(!LabelChangeCorner(0,InpName,CORNER_RIGHT_UPPER))
      return;
//--- ラベルテキストを変更する
   StringConcatenate(text,"Upper right corner: ",x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return;
//--- チャートを再描画して 2 秒待つ
   ChartRedraw();
   Sleep(2000);
//--- 左上の隅に移動する
   if(!LabelChangeCorner(0,InpName,CORNER_LEFT_UPPER))
      return;
//--- ラベルテキストを変更する
   StringConcatenate(text,"Upper left corner: ",x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return;
//--- チャートを再描画して 2 秒待つ
   ChartRedraw();
   Sleep(2000);
//--- ラベルを削除する
   LabelDelete(0,InpName);
//--- チャートを再描画して1 秒待つ
   ChartRedraw();
   Sleep(500);
//---
  }
//+------------------------------------------------------------------+
//| この関数はオブジェクトを移動してそのテキストを変更する                |
//+------------------------------------------------------------------+
bool MoveAndTextChange(const int x,const int y,string text)
  {
//--- ラベルを移動する
   if(!LabelMove(0,InpName,x,y))
      return(false);
//--- ラベルテキストを変更する
   StringConcatenate(text,text,x,",",y);
   if(!LabelTextChange(0,InpName,text))
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
//--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
   if(IsStopped())
      return(false);
//--- チャートを再描画する
   ChartRedraw();
// 0.01 秒の遅れ
   Sleep(10);
//--- 関数を終了する
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_BUTTON

ボタンオブジェクト



注意事項

アンカーポイントの座標はピクセルで設定されています。ボタンオブジェクトのアンカーポイントとして使用される隅は ENUM_BASE_CORNER 列挙から選択出来ます。

例

次のスクリプトは、チャート上でボタンオブジェクトを作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script creates the button on the chart."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string           InpName="Button";           // ボタン名
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // アンカーに使用されるチャートのコーナー
input string           InpFont="Arial";            // フォント
input int              InpFontSize=14;             // フォントサイズ
input color            InpColor=clrBlack;          // テキストの色
input color            InpBackColor=C'236,233,216'; // 背景色
input color            InpBorderColor=clrNONE;     // 境界線の色
input bool             InpState=false;             // 押される/放される
input bool             InpBack=false;              // 背景オブジェクト
input bool             InpSelection=false;         // 強調表示して移動
```

© 2000–2025, MetaQuotes Ltd.

```
input bool              InpHidden=true;              // オブジェクトリストに隠す
input long              InpZOrder=0;                 // マウスクリックの優先順位
//+------------------------------------------------------------------+
//| ボタンを作成する                                                 |
//+------------------------------------------------------------------+
bool ButtonCreate(const long              chart_ID=0,               // チャート識別子
                  const string            name="Button",            // ボタン名
                  const int               sub_window=0,             // サブウィンドウ番号
                  const int               x=0,                      // X 座標
                  const int               y=0,                      // Y 座標
                  const int               width=50,                 // ボタンの幅
                  const int               height=18,                // ボタンの高さ
                  const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // アンカーに使用されるチャートのコーナー
                  const string            text="Button",            // テキスト
                  const string            font="Arial",             // フォント
                  const int               font_size=10,             // フォントサイズ
                  const color             clr=clrBlack,             // テキストの色
                  const color             back_clr=C'236,233,216',  // 背景色
                  const color             border_clr=clrNONE,       // 境界線の色
                  const bool              state=false,              // 押される/放される
                  const bool              back=false,               // 背景で表示する
                  const bool              selection=false,          // 強調表示して移動
                  const bool              hidden=true,              // オブジェクトリストに隠す
                  const long              z_order=0)                // マウスクリックの優先順位
  {
//--- エラー値をリセットする
   ResetLastError();
//--- ボタンを作成する
   if(!ObjectCreate(chart_ID,name,OBJ_BUTTON,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": failed to create the button! Error code = ",GetLastError());
      return(false);
     }
//--- ボタン座標を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- ボタンサイズを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- ポイント座標が相対的に定義されているチャートのコーナーを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- テキストを設定する
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- テキストフォントを設定する
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- フォントサイズを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- テキストの色を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 背景色を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_BGCOLOR,back_clr);
//--- 境界線の色を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_COLOR,border_clr);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- ボタンの状態を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_STATE,state);
//--- マウスでのボタンを移動させるモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| ボタンを移動する                                                 |
//+------------------------------------------------------------------+
bool ButtonMove(const long   chart_ID=0,     // チャート識別子
                const string name="Button", // ボタン名
                const int    x=0,           // X 座標
                const int    y=0)           // Y 座標
  {
//--- エラー値をリセットする
   ResetLastError();
//--- ボタンを移動する
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": failed to move X coordinate of the button! Error code =
",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": failed to move Y coordinate of the button! Error code =
",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| ボタンサイズを変更する                                           |
//+------------------------------------------------------------------+
bool ButtonChangeSize(const long   chart_ID=0,     // チャート識別子
                      const string name="Button", // ボタン名
                      const int    width=50,      // ボタンの幅
                      const int    height=18)     // ボタンの高さ
  {
//--- エラー値をリセットする
   ResetLastError();
//--- ボタンサイズを変更する
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": failed to change the button width! Error code = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": failed to change the button height! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| ボタン結合に使用されるチャートのコーナーを変更する               |
//+------------------------------------------------------------------+
bool ButtonChangeCorner(const long             chart_ID=0,               // チャート識
別子
                        const string           name="Button",            // ボタン名
                        const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER) // アンカーに使
用されるチャートのコーナー
  {
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーに使用される隅を変更する
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner))
     {
      Print(__FUNCTION__,
            ": failed to change the anchor corner! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| ボタンテキストを変更する                                                  |
//+------------------------------------------------------------------+
bool ButtonTextChange(const long   chart_ID=0,    // チャート識別子
                      const string name="Button", // ボタン名
                      const string text="Text")   // テキスト
  {
//--- エラー値をリセットする
   ResetLastError();
//--- オブジェクトのテキストを変更する
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": failed to change the text! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| ボタンを削除する                                                  |
//+------------------------------------------------------------------+
bool ButtonDelete(const long   chart_ID=0,    // チャート識別子
                  const string name="Button") // ボタン名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- ボタンを削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete the button! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- チャートウィンドウサイズ
   long x_distance;
   long y_distance;
//--- ウィンドウサイズを設定する
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Failed to get the chart width! Error code = ",GetLastError());
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
      {
       Print("Failed to get the chart height! Error code = ",GetLastError());
       return;
      }
//--- ボタンサイズ変更のステップを定義する
   int x_step=(int)x_distance/32;
   int y_step=(int)y_distance/32;
//--- ボタン座標とサイズを設定する
   int x=(int)x_distance/32;
   int y=(int)y_distance/32;
   int x_size=(int)x_distance*15/16;
   int y_size=(int)y_distance*15/16;
//--- ボタンを作成する
   if(!ButtonCreate(0,InpName,0,x,y,x_size,y_size,InpCorner
,"Press",InpFont,InpFontSize,
       InpColor
,InpBackColor,InpBorderColor,InpState,InpBack,InpSelection,InpHidden,InpZOrder))
      {
       return;
      }
//--- チャートを再描画する
   ChartRedraw();
//--- ループ内でボタンを減らす
   int i=0;
   while(i<13)
      {
       //--- 半秒の遅れ
       Sleep(500);
       //--- ボタンを押された状態に切り替える
       ObjectSetInteger(0,InpName,OBJPROP_STATE,true);
       //--- チャートを再描画して 0.2 秒待つ
       ChartRedraw();
       Sleep(200);
       //--- 座標とボタンサイズを再定義する
       x+=x_step;
       y+=y_step;
       x_size-=x_step*2;
       y_size-=y_step*2;
       //--- ボタンを減らす
       ButtonMove(0,InpName,x,y);
       ButtonChangeSize(0,InpName,x_size,y_size);
       //--- ボタンを押されていない状態に返す
       ObjectSetInteger(0,InpName,OBJPROP_STATE,false);
       //--- チャートを再描画する
       ChartRedraw();
       //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
       if(IsStopped())
```

© 2000-2025, MetaQuotes Ltd.

```
            return;
        //--- ループカウンタを増加する
        i++;
       }
//--- 半秒の遅れ
    Sleep(500);
//--- ボタンを削除する
    ButtonDelete(0,InpName);
    ChartRedraw();
//---1 秒待つ
    Sleep(1000);
//---
  }
```

© 2000–2025, MetaQuotes Ltd.

OBJ_CHART

チャート オブジェクト



注意事項

アンカーポイントの座標はピクセルで設定されています。アンカーポイントとして使用される隅は ENUM_BASE_CORNER 列挙から選択出来ます。

チャート オブジェクトでは、シンボル、期間と規模を選択することが出来ます。価格スケールと日付表示モードは、有効/無効にすることも出来ます。

例

次のスクリプトは、チャート上でチャート オブジェクトを作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script creates \"Chart\" object."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string           InpName="Chart";              // オブジェクト名
input string           InpSymbol="EURUSD";           // シンボル
input ENUM_TIMEFRAMES  InpPeriod=PERIOD_H1;          // 期間
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // アンカーに使用される隅
input int              InpScale=2;                   // 規模
input bool             InpDateScale=true;            // 時間スケール表示
input bool             InpPriceScale=true;           // 価格スケール表示
```

© 2000-2025, MetaQuotes Ltd.

```
input color             InpColor=clrRed;              // 強調表示時の境界線の色
input ENUM_LINE_STYLE   InpStyle=STYLE_DASHDOTDOT;    // 強調表示時の線のスタイル
input int               InpPointWidth=1;              // 移動のポイントサイズ
input bool              InpBack=false;                // 背景オブジェクト
input bool              InpSelection=true;            // 強調表示して移動
input bool              InpHidden=true;               // オブジェクトリストに隠す
input long              InpZOrder=0;                  // マウスクリックの優先順位
//+------------------------------------------------------------------+
//| チャートオブジェクトを作成する                                   |
//+------------------------------------------------------------------+
bool ObjectChartCreate(const long              chart_ID=0,               // チャート識別子
                       const string            name="Chart",             // オブジェクト名
                       const int               sub_window=0,             // サブウィンドウ番号
                       const string            symbol="EURUSD",          // シンボル
                       const ENUM_TIMEFRAMES   period=PERIOD_H1,         // 期間
                       const int               x=0,                      // X 座標
                       const int               y=0,                      // Y 座標
                       const int               width=300,                // 幅
                       const int               height=200,               // 高さ
                       const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // アンカーに使用される隅
                       const int               scale=2,                  // 規模
                       const bool              date_scale=true,          // 時間スケール表示
                       const bool              price_scale=true,         // 価格スケール表示
                       const color             clr=clrRed,               // 強調表示時の境界線の色
                       const ENUM_LINE_STYLE   style=STYLE_SOLID,        // 強調表示時の線のスタイル
                       const int               point_width=1,            // 移動ポイントサイズ
                       const bool              back=false,               // 背景で表示する
                       const bool              selection=false,          // 強調表示して移動
                       const bool              hidden=true,              // オブジェクトリストに隠す
                       const long              z_order=0)                // マウスクリックの優先順位
  {
//--- エラー値をリセットする
   ResetLastError();
//--- チャートオブジェクトを作成する
   if(!ObjectCreate(chart_ID,name,OBJ_CHART,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": failed to create \"Chart\" object! Error code = ",GetLastError());
```

© 2000–2025, MetaQuotes Ltd.

```
      return(false);
     }
//--- オブジェクト座標を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- オブジェクトサイズを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- ポイント座標が相対的に定義されているチャートのコーナーを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- シンボルを設定する
   ObjectSetString(chart_ID,name,OBJPROP_SYMBOL,symbol);
//--- 期間を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_PERIOD,period);
//--- 規模を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_CHART_SCALE,scale);
//--- 時間スケールを表示(true)または非表示(false)する
   ObjectSetInteger(chart_ID,name,OBJPROP_DATE_SCALE,date_scale);
//--- 価格スケールを表示(true)または非表示(false)する
   ObjectSetInteger(chart_ID,name,OBJPROP_PRICE_SCALE,price_scale);
//--- オブジェクトのハイライト表示モードが有効になっている際の境界線の色を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- オブジェクトのハイライト表示モードが有効になっている際の境界線のスタイルを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- オブジェクトを移動するためのアンカーポイントのサイズを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,point_width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- マウスでラベルを移動させるモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  チャートオブジェクトのシンボルと時間軸を設定する                |
//+------------------------------------------------------------------+
bool ObjectChartSetSymbolAndPeriod(const long            chart_ID=0,        // チャート
識別子(チャートオブジェクト識別子ではない)
                                   const string          name="Chart",      // オブジェク
ト名
                                   const string          symbol="EURUSD",   // シンボル
                                   const ENUM_TIMEFRAMES period=PERIOD_H1) // 時間軸
  {
//--- エラー値をリセットする
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
//--- チャートオブジェクトのシンボルと時間軸を設定する
   if(!ObjectSetString(chart_ID,name,OBJPROP_SYMBOL,symbol))
     {
      Print(__FUNCTION__,
            ": failed to set a symbol for \"Chart\" object! Error code =
",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_PERIOD,period))
     {
      Print(__FUNCTION__,
            ": failed to set a period for \"Chart\" object! Error code =
",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  チャートオブジェクトを移動する                                  |
//+------------------------------------------------------------------+
bool ObjectChartMove(const long   chart_ID=0,   // チャート識別子（チャートオブジェクト ＩＤ ではない）
                     const string name="Chart", // オブジェクト名
                     const int    x=0,          // X 座標
                     const int    y=0)          // Y 座標
  {
//--- エラー値をリセットする
   ResetLastError();
//--- オブジェクトを移動する
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": failed to move X coordinate of \"Chart\" object! Error code =
",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": failed to move Y coordinate of \"Chart\" object! Error code =
",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  チャートオブジェクトサイズを変更する                            |
```

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool ObjectChartChangeSize(const long   chart_ID=0,   // チャート識別子（チャートオブジェクト ＩＤ ではない）
                           const string name="Chart", // オブジェクト名
                           const int    width=300,    // 幅
                           const int    height=200)   // 高さ
  {
//--- エラー値をリセットする
   ResetLastError();
//--- オブジェクトサイズを変更する
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": failed to change the width of \"Chart\" object! Error code = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": failed to change the height of \"Chart\" object! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| チャートオブジェクト ＩＤ を返す                                 |
//+------------------------------------------------------------------+
long ObjectChartGetID(const long   chart_ID=0,   // チャート識別子（チャートオブジェクト ＩＤ ではない）
                      const string name="Chart") // オブジェクト名
  {
//--- チャートオブジェクト ＩＤ 取得の変数を準備する
   long id=-1;
//--- エラー値をリセットする
   ResetLastError();
//--- ＩＤ 取得
   if(!ObjectGetInteger(chart_ID,name,OBJPROP_CHART_ID,0,id))
     {
      Print(__FUNCTION__,
            ": failed to get \"Chart\" object's ID! Error code = ",GetLastError());
     }
//--- 結果を返す
   return(id);
  }
//+------------------------------------------------------------------+
//| チャートオブジェクトを削除する                                   |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool ObjectChartDelete(const long   chart_ID=0,   // チャート識別子（チャートオブジェクト ＩＤ ではない）
                       const string name="Chart") // オブジェクト名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- ボタンを削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete \"Chart\" object! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 「気配値表示」でのシンボル数を取得
   int  symbols=SymbolsTotal(true);
//--- 指定された名称のシンボルがシンボルのリストに存在するかどうかをチェックする
   bool exist=false;
   for(int i=0;i<symbols;i++)
      if(InpSymbol==SymbolName(i,true))
        {
         exist=true;
         break;
        }
   if(!exist)
     {
      Print("Error!",InpSymbol," symbol is not present in \"「気配値表示」¥"!");
      return;
     }
//--- 入力パラメータの有効性をチェックする
   if(InpScale<0 || InpScale>5)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }

//--- チャートウィンドウサイズ
   long x_distance;
   long y_distance;
//--- ウィンドウサイズを設定する
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
```

© 2000–2025, MetaQuotes Ltd.

```
      {
       Print("Failed to get the chart width! Error code = ",GetLastError());
       return;
      }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
      {
       Print("Failed to get the chart height! Error code = ",GetLastError());
       return;
      }
//--- チャートオブジェクトの座標とサイズを設定する
   int x=(int)x_distance/16;
   int y=(int)y_distance/16;
   int x_size=(int)x_distance*7/16;
   int y_size=(int)y_distance*7/16;
//--- チャートオブジェクトを作成する
   if(!ObjectChartCreate(0,InpName,0,InpSymbol
,InpPeriod,x,y,x_size,y_size,InpCorner,InpScale,InpDateScale,
       InpPriceScale
,InpColor,InpStyle,InpPointWidth,InpBack,InpSelection,InpHidden,InpZOrder))
      {
       return;
      }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- チャートオブジェクトをスケッチする
   int steps=(int)MathMin(x_distance*7/16,y_distance*7/16);
   for(int i=0;i<steps;i++)
      {
       //--- サイズ変更
       x_size+=1;
       y_size+=1;
       if(!ObjectChartChangeSize(0,InpName,x_size,y_size))
          return;
       //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
       if(IsStopped())
          return;
       //--- チャートを再描画して 0.01 秒待つ
       ChartRedraw();
       Sleep(10);
      }
//--- 半秒の遅れ
   Sleep(500);
//--- チャートの時間軸を変更する
   if(!ObjectChartSetSymbolAndPeriod(0,InpName,InpSymbol,PERIOD_M1))
      return;
   ChartRedraw();
//--- 3 秒の遅れ
   Sleep(3000);
```

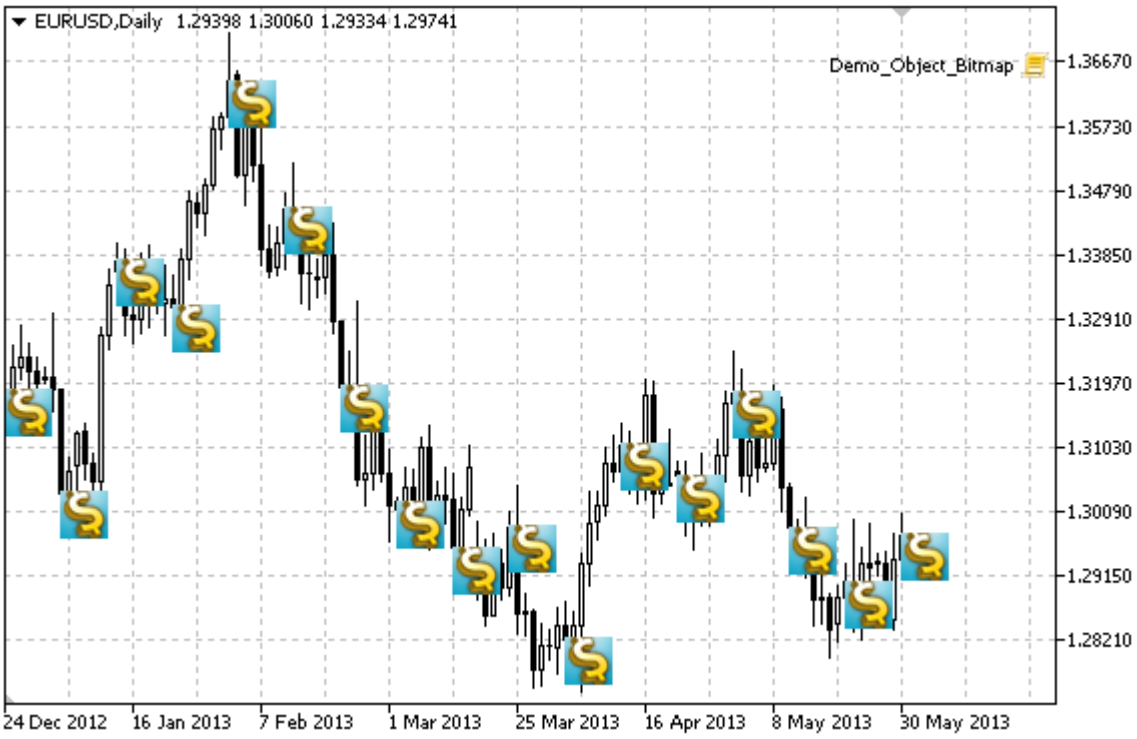
© 2000–2025, MetaQuotes Ltd.

```
//--- オブジェクトを削除する
   ObjectChartDelete(0,InpName);
   ChartRedraw();
//---1 秒待つ
   Sleep(1000);
//---
  }
```

© 2000–2025, MetaQuotes Ltd.

OBJ_BITMAP

ビットマップオブジェクト



注意事項

ビットマップオブジェクトでは、画像の表示範囲を選択出来ます。

例

次のスクリプトは、チャート上で複数のビットマップオブジェクトを作成します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script creates a bitmap in the chart window."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string           InpFile="\\Images\\dollar.bmp"; // ビットマップファイル名
input int              InpWidth=24;                    // 表示範囲の X 座標
input int              InpHeight=24;                   // 表示範囲の Y 座標
input int              InpXOffset=4;                   // 表示範囲を X 軸に沿ってシフト
input int              InpYOffset=4;                   // 表示範囲を Y 軸に沿ってシフト
input color            InpColor=clrRed;                // 強調表示時の境界線の色
input ENUM_LINE_STYLE InpStyle=STYLE_SOLID;           // 強調表示時の線のスタイル
input int              InpPointWidth=1;                // 移動のポイントサイズ
input bool             InpBack=false;                  // 背景オブジェクト
input bool             InpSelection=false;             // 強調表示して移動
```

© 2000–2025, MetaQuotes Ltd.

```
input bool            InpHidden=true;                  // オブジェクトリストに隠す
input long            InpZOrder=0;                     // マウスクリックの優先順位
//+------------------------------------------------------------------+
//| チャートウィンドウでビットマップを作成する                                |
//+------------------------------------------------------------------+
bool BitmapCreate(const long            chart_ID=0,         // チャート識別子
                  const string          name="Bitmap",      // ビットマップ名
                  const int             sub_window=0,       // サブウィンドウ番号
                  datetime              time=0,             // アンカーポイントの時刻
                  double                price=0,            // アンカーポイントの価格
                  const string          file="",            // ビットマップファイル名
                  const int             width=10,           // 表示範囲の X 座標
                  const int             height=10,          // 表示範囲の Y 座標
                  const int             x_offset=0,         // 表示範囲を X 軸に沿ってシフト
                  const int             y_offset=0,         // 表示範囲を Y 軸に沿ってシフト
                  const color           clr=clrRed,         // 強調表示時の境界線の色
                  const ENUM_LINE_STYLE style=STYLE_SOLID,  // 強調表示時の線のスタイル
                  const int             point_width=1,      // 移動ポイントサイズ
                  const bool            back=false,         // 背景で表示する
                  const bool            selection=false,    // 強調表示して移動
                  const bool            hidden=true,        // オブジェクトリストに隠す
                  const long            z_order=0)          // マウスクリックの優先順位
  {
//--- 設定されてない場合アンカーポイントの座標を設定する
   ChangeBitmapEmptyPoint(time,price);
//--- エラー値をリセットする
   ResetLastError();
//--- ビットマップを作成する
   if(!ObjectCreate(chart_ID,name,OBJ_BITMAP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": failed to create a bitmap in the chart window! Error code =
",GetLastError());
      return(false);
     }
//--- 画像へのパスを設定する
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,file))
     {
      Print(__FUNCTION__,
            ": failed to load the image! Error code = ",GetLastError());
      return(false);
     }
//--- 画像の表示範囲を設定する。もし幅または高さの値が
//--- それぞれソース画像の幅と高さを超えた場合、
//--- 描画されない。反対の場合
//--- これらの値に対応する部分が描画される。
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- 表示範囲に表示される画像の部分を設定する
```

© 2000-2025, MetaQuotes Ltd.

```
//--- デフォルトの部分は、画像の左上の領域
//--- 値は、この領域からのシフトして画像の別の部分の表示を可能にする
   ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset);
   ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset);
//--- オブジェクトのハイライト表示モードが有効になっている際の境界線の色を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- オブジェクトのハイライト表示モードが有効になっている際の境界線のスタイルを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- オブジェクトを移動するためのアンカーポイントのサイズを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,point_width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- マウスでラベルを移動させるモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| ビットマップに新しい画像を設定する                                |
//+------------------------------------------------------------------+
bool BitmapSetImage(const long   chart_ID=0,     // チャート識別子
                    const string name="Bitmap", // ビットマップ名
                    const string file="")       // ファイルパス
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 画像へのパスを設定する
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,file))
     {
      Print(__FUNCTION__,
            ": failed to load the image! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| チャートのビットマップを移動する                                  |
//+------------------------------------------------------------------+
bool BitmapMove(const long   chart_ID=0,     // チャート識別子
                const string name="Bitmap", // ビットマップ名
                datetime     time=0,        // アンカーポイントの時刻
                double       price=0)       // アンカーポイントの価格
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- ポイントの位置が設定されていない場合、売値を有する現在足に移動する
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- エラー値をリセットする
   ResetLastError();
//--- アンカーポイントを移動する
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": failed to move the anchor point! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| ビットマップの表示範囲のサイズを変更する                                  |
//+------------------------------------------------------------------+
bool BitmapChangeSize(const long   chart_ID=0,     // チャート識別子
                      const string name="Bitmap", // ビットマップ名
                      const int    width=0,       // ビットマップの幅
                      const int    height=0)      // ビットマップの高さ
  {
//--- エラー値をリセットする
   ResetLastError();
//--- ビットマップサイズを変更する
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": failed to change the bitmap width! Error code = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": failed to change the bitmap height! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 表示範囲の左上隅のy座標を変更する                                      |
//+------------------------------------------------------------------+
bool BitmapMoveVisibleArea(const long   chart_ID=0,     // チャート識別子
                           const string name="Bitmap", // ビットマップ名
                           const int    x_offset=0,    // 表示範囲の X 座標
```

© 2000-2025, MetaQuotes Ltd.

```
                              const int     y_offset=0)     // 表示範囲の Y 座標
  {
//--- エラー値をリセットする
   ResetLastError();
//--- ビットマップの表示範囲の座標を変更する
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset))
     {
      Print(__FUNCTION__,
            ": failed to change X coordinate of the visibility scope! Error code = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset))
     {
      Print(__FUNCTION__,
            ": failed to change Y coordinate of the visibility scope! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| ビットマップを削除する                                           |
//+------------------------------------------------------------------+
bool BitmapDelete(const long   chart_ID=0,     // チャート識別子
                  const string name="Bitmap") // ビットマップ名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- ラベルを削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete a bitmap! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| アンカーポイントの値をチェックして                               |
//| 空の物には初期値を設定する                                       |
//+------------------------------------------------------------------+
void ChangeBitmapEmptyPoint(datetime &time,double &price)
  {
//--- ポイントの時間が設定されていない場合、現在足になる
   if(!time)
      time=TimeCurrent();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- ポイントの価格が設定されていない場合、売値になる
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date[];  // 表示されるバーを格納するための配列
   double   close[]; // 終値を格納するための配列
//--- ビットマップファイル名
   string   file="\\Images\\dollar.bmp";
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- メモリ割り当て
   ArrayResize(date,bars);
   ArrayResize(close,bars);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- 終値の配列に書き込む
   if(CopyClose(Symbol(),Period(),0,bars,close)==-1)
     {
      Print("Failed to copy the values of Close prices! Error code =
",GetLastError());
      return;
     }
//--- 画像表示の頻度を定義する
   int scale=(int)ChartGetInteger(0,CHART_SCALE);
//--- ステップを定義する
   int step=1;
   switch(scale)
     {
      case 0:
         step=27;
         break;
      case 1:
         step=14;
         break;
      case 2:
         step=7;
         break;
      case 3:
         step=4;
```

© 2000-2025, MetaQuotes Ltd.

```
         break;
      case 4:
         step=2;
         break;
     }
//--- 高値と安値のバーの値でビットマップを作成する(ギャップで)
   for(int i=0;i<bars;i+=step)
     {
      //--- ビットマップを作成する
      if(!BitmapCreate(0,"Bitmap_"+(string)
i,0,date[i],close[i],InpFile,InpWidth,InpHeight,InpXOffset,
         InpYOffset
,InpColor,InpStyle,InpPointWidth,InpBack,InpSelection,InpHidden,InpZOrder))
        {
         return;
        }
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
      // 0.05 秒の遅れ
      Sleep(50);
     }
//--- 半秒の遅れ
   Sleep(500);
//--- 売りサインを削除する
   for(int i=0;i<bars;i+=step)
     {
      if(!BitmapDelete(0,"Bitmap_"+(string)i))
         return;
      if(!BitmapDelete(0,"Bitmap_"+(string)i))
         return;
      //--- チャートを再描画する
      ChartRedraw();
      // 0.05 秒の遅れ
      Sleep(50);
     }
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_BITMAP_LABEL

ビットマップラベルオブジェクト



注意事項

ラベルに相対したアンカーポイントの位置関係は ENUM_ANCHOR_POINT 列挙から選択出来ます。アンカーポイントの座標はピクセルで設定されています。

ビットマップのアンカーポイントとして使用される隅は ENUM_BASE_CORNER 列挙から選択出来ます。

ビットマップラベルでは、画像の表示範囲を選択出来ます。

例

次のスクリプトは、チャート上で複数のビットマップオブジェクトを作成します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script creates \"Bitmap Label\" object."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string             InpName="BmpLabel";                // ラベル名
input string             InpFileOn="\\Images\\dollar.bmp"; // オンモードでのファイル名
input string             InpFileOff="\\Images\\euro.bmp";  // オフモードでのファイル名
input bool               InpState=false;                    // ラベルが押される/解放される
input ENUM_BASE_CORNER   InpCorner=CORNER_LEFT_UPPER;       // アンカーに使用されるチャートのコーナー
input ENUM_ANCHOR_POINT InpAnchor=ANCHOR_CENTER;           // アンカーの種類
```

© 2000–2025, MetaQuotes Ltd.

```
input color             InpColor=clrRed;                  // 強調表示時の境界線の色
input ENUM_LINE_STYLE   InpStyle=STYLE_SOLID;             // 強調表示時の線のスタイル
input int               InpPointWidth=1;                  // 移動のポイントサイズ
input bool              InpBack=false;                    // 背景オブジェクト
input bool              InpSelection=false;               // 強調表示して移動
input bool              InpHidden=true;                   // オブジェクトリストに隠す
input long              InpZOrder=0;                      // マウスクリックの優先順位
//+------------------------------------------------------------------+
//| ビットマップラベルオブジェクトを作成する                                |
//+------------------------------------------------------------------+
bool BitmapLabelCreate(const long              chart_ID=0,               // チャート識別子
                       const string            name="BmpLabel",          // ラベル名
                       const int               sub_window=0,             // サブウィンドウ番号
                       const int               x=0,                      // X 座標
                       const int               y=0,                      // Y 座標
                       const string            file_on="",               // オンモードでの画像
                       const string            file_off="",              // オフモードでの画像
                       const int               width=0,                  // 表示範囲のX 座標
                       const int               height=0,                 // 表示範囲のY 座標
                       const int               x_offset=10,              // 表示範囲をX 軸に沿ってシフト
                       const int               y_offset=10,              // 表示範囲をY 軸に沿ってシフト
                       const bool              state=false,              // 押される/解放される
                       const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // アンカーに使用されるチャートのコーナー
                       const ENUM_ANCHOR_POINT anchor=ANCHOR_LEFT_UPPER, // アンカーの種類
                       const color             clr=clrRed,               // 強調表示時の境界線の色
                       const ENUM_LINE_STYLE   style=STYLE_SOLID,        // 強調表示時の線のスタイル
                       const int               point_width=1,            // 移動ポイントサイズ
                       const bool              back=false,               // 背景で表示する
                       const bool              selection=false,          // 強調表示して移動
                       const bool              hidden=true,              // オブジェクトリストに隠す
                       const long              z_order=0)                // マウスクリックの優先順位
  {
//--- エラー値をリセットする
   ResetLastError();
```

© 2000–2025, MetaQuotes Ltd.

```
//--- ビットマップラベルを作成する
   if(!ObjectCreate(chart_ID,name,OBJ_BITMAP_LABEL,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": failed to create \"Bitmap Label\" object! Error code =
",GetLastError());
      return(false);
     }
//--- オンとオフモードでの画像を設定する
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,0,file_on))
     {
      Print(__FUNCTION__,
            ": failed to load the image for On mode! Error code = ",GetLastError());
      return(false);
     }
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,1,file_off))
     {
      Print(__FUNCTION__,
            ": failed to load the image for Off mode! Error code =
",GetLastError());
      return(false);
     }
//--- ラベル座標を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- 画像の表示範囲を設定する。もし幅または高さの値が
//--- それぞれソース画像の幅と高さを超えた場合、
//--- 描画されない。反対の場合
//--- これらの値に対応する部分が描画される。
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- 表示範囲に表示される画像の部分を設定する
//--- デフォルトの部分は、画像の左上の領域
//--- 値は、この領域からのシフトして画像の別の部分の表示を可能にする
   ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset);
   ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset);
//--- ラベルの状態（キーの押または離し）を定義する
   ObjectSetInteger(chart_ID,name,OBJPROP_STATE,state);
//--- ポイント座標が相対的に定義されているチャートのコーナーを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- アンカーの種類を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- オブジェクトのハイライト表示モードが有効になっている際の境界線の色を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- オブジェクトのハイライト表示モードが有効になっている際の境界線のスタイルを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- オブジェクトを移動するためのアンカーポイントのサイズを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,point_width);
//--- 前景（false）または背景（true）に表示
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- マウスでラベルを移動させるモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| ビットマップラベルオブジェクトに新しい画像を設定する                                  |
//+------------------------------------------------------------------+
bool BitmapLabelSetImage(const long   chart_ID=0,      // チャート識別子
                         const string name="BmpLabel", // ラベル名e
                         const int    on_off=0,        // 修飾子(オンまたはオフ)
                         const string file="")         // ファイルパス
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 画像へのパスを設定する
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,on_off,file))
     {
      Print(__FUNCTION__,
            ": failed to load the image! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| ビットマップラベルオブジェクトを移動する                                          |
//+------------------------------------------------------------------+
bool BitmapLabelMove(const long   chart_ID=0,      // チャート識別子
                     const string name="BmpLabel", // ラベル名
                     const int    x=0,             // X 座標
                     const int    y=0)             // Y 座標
  {
//--- エラー値をリセットする
   ResetLastError();
//--- オブジェクトを移動する
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": failed to move X coordinate of the object! Error code =
",GetLastError());
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": failed to move Y coordinate of the object! Error code =
",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| ビットマップの表示範囲のサイズを変更する                                  |
//+------------------------------------------------------------------+
bool BitmapLabelChangeSize(const long   chart_ID=0,       // チャート識別子
                           const string name="BmpLabel", // ラベル名
                           const int    width=0,         // ラベル幅
                           const int    height=0)        // ラベルの高さ
  {
//--- エラー値をリセットする
   ResetLastError();
//--- オブジェクトサイズを変更する
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": failed to change the object width! Error code = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": failed to change the object height! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 表示範囲の左上隅のy座標を変更する                                      |
//+------------------------------------------------------------------+
bool BitmapLabelMoveVisibleArea(const long   chart_ID=0,       // チャート識別子
                                const string name="BmpLabel", // ラベル名
                                const int    x_offset=0,      // 表示範囲の X 座標
                                const int    y_offset=0)      //表示範囲の Y 座標
  {
//--- エラー値をリセットする
   ResetLastError();
//--- オブジェクトの表示範囲の座標を変更する
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print(__FUNCTION__,
            ": failed to change X coordinate of the visibility scope! Error code =
",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset))
     {
      Print(__FUNCTION__,
            ": failed to change Y coordinate of the visibility scope! Error code =
",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| ビットマップラベルオブジェクトを削除する                                  |
//+------------------------------------------------------------------+
bool BitmapLabelDelete(const long   chart_ID=0,      // チャート識別子
                       const string name="BmpLabel") // ラベル名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- ラベルを削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete \"Bitmap label\" object! Error code =
",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- チャートウィンドウサイズ
   long x_distance;
   long y_distance;
//--- ウィンドウサイズを設定する
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Failed to get the chart width! Error code = ",GetLastError());
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
```

© 2000–2025, MetaQuotes Ltd.

```
      {
       Print("Failed to get the chart height! Error code = ",GetLastError());
       return;
      }
//--- ビットマップラベル座標を定義する
   int x=(int)x_distance/2;
   int y=(int)y_distance/2;
//--- ラベルのサイズと表示範囲座標を設定する
   int width=32;
   int height=32;
   int x_offset=0;
   int y_offset=0;
//--- ビットマップラベルをウィンドウの中央に配置する
   if(!BitmapLabelCreate(0,InpName,0,x,y,InpFileOn
,InpFileOff,width,height,x_offset,y_offset,InpState,
       InpCorner
,InpAnchor
,InpColor,InpStyle,InpPointWidth,InpBack,InpSelection,InpHidden,InpZOrder))
      {
       return;
      }
//--- チャートを再描画して1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- ループ内でラベルの表示範囲サイズを変更する
   for(int i=0;i<6;i++)
     {
      //--- 表示範囲サイズを変更する
      width--;
      height--;
      if(!BitmapLabelChangeSize(0,InpName,width,height))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
      // 0.3 秒の遅れ
      Sleep(300);
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- ループ内でラベルの表示範囲座標を変更する
   for(int i=0;i<2;i++)
     {
      //--- 表示範囲座標を変更する
      x_offset++;
      y_offset++;
      if(!BitmapLabelMoveVisibleArea(0,InpName,x_offset,y_offset))
```

© 2000-2025, MetaQuotes Ltd.

```
            return;
        //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
        if(IsStopped())
            return;
        //--- チャートを再描画する
        ChartRedraw();
        // 0.3 秒の遅れ
        Sleep(300);
       }
//--- 1 秒の遅れ
    Sleep(1000);
//--- ラベルを削除する
    BitmapLabelDelete(0,InpName);
    ChartRedraw();
//--- 1 秒の遅れ
    Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_EDIT

編集オブジェクト



注意事項

アンカーポイントの座標はピクセルで設定されています。編集オブジェクトに使用される隅は ENUM_BASE_CORNER 列挙から選択出来ます。

また、編集オブジェクトではテキストの整列の種類を ENUM_ALIGN_MODE 列挙から選択することが出来ます。

例

次のスクリプトは、チャート上で編集オブジェクトを作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script creates \"Edit\" object."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string            InpName="Edit";              // オブジェクト名
input string            InpText="Text";              // オブジェクトテキスト
input string            InpFont="Arial";             // フォント
input int               InpFontSize=14;              // フォントサイズ
input ENUM_ALIGN_MODE   InpAlign=ALIGN_CENTER;       // テキスト整列の種類
input bool              InpReadOnly=false;           // 編集許可
input ENUM_BASE_CORNER  InpCorner=CORNER_LEFT_UPPER; // アンカーに使用されるチャートのコーナー
input color             InpColor=clrBlack;           // テキストの色
```

© 2000–2025, MetaQuotes Ltd.

```
input color             InpBackColor=clrWhite;      // 背景色
input color             InpBorderColor=clrBlack;    // 境界線の色
input bool              InpBack=false;              // 背景オブジェクト
input bool              InpSelection=false;         // 強調表示して移動
input bool              InpHidden=true;             // オブジェクトリストに隠す
input long              InpZOrder=0;                // マウスクリックの優先順位
//+------------------------------------------------------------------+
//| 編集オブジェクトを作成する                                       |
//+------------------------------------------------------------------+
bool EditCreate(const long             chart_ID=0,               // チャート識別子
                const string           name="Edit",              // オブジェクト名
                const int              sub_window=0,             // サブウィンドウ番号
                const int              x=0,                      // X 座標
                const int              y=0,                      // Y 座標
                const int              width=50,                 // 幅
                const int              height=18,                // 高さ
                const string           text="Text",              // テキスト
                const string           font="Arial",             // フォント
                const int              font_size=10,             // フォントサイズ
                const ENUM_ALIGN_MODE  align=ALIGN_CENTER,       // 整列の種類
                const bool             read_only=false,          // 編集する能力
                const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER, // アンカーに使用される
チャートのコーナー
                const color            clr=clrBlack,             // テキストの色
                const color            back_clr=clrWhite,        // 背景色r
                const color            border_clr=clrNONE,       // 境界線の色
                const bool             back=false,               // 背景で表示する
                const bool             selection=false,          // 強調表示して移動
                const bool             hidden=true,              // オブジェクトリストに隠す
                const long             z_order=0)                // マウスクリックの優先順
位
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 編集フィールドを作成する
   if(!ObjectCreate(chart_ID,name,OBJ_EDIT,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": failed to create \"Edit\" object! Error code = ",GetLastError());
      return(false);
     }
//--- オブジェクト座標を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- オブジェクトサイズを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- テキストを設定する
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
```

© 2000–2025, MetaQuotes Ltd.

```
//--- テキストフォントを設定する
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
//--- フォントサイズを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- オブジェクトでのテキスト配置のタイプを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ALIGN,align);
//--- 読み込み専用モードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_READONLY,read_only);
//--- オブジェクト座標が相対的に定義されているチャートのコーナーを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- テキストの色を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 背景色を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_BGCOLOR,back_clr);
//--- 境界線の色を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_COLOR,border_clr);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- マウスでラベルを移動させるモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 編集オブジェクトを移動する                                       |
//+------------------------------------------------------------------+
bool EditMove(const long   chart_ID=0,  // チャート識別子
              const string name="Edit", // オブジェクト名
              const int    x=0,         // X 座標
              const int    y=0)         // Y 座標
  {
//--- エラー値をリセットする
   ResetLastError();
//--- オブジェクトを移動する
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": failed to move X coordinate of the object! Error code =
",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
```

© 2000-2025, MetaQuotes Ltd.

```
            ": failed to move Y coordinate of the object! Error code =
",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 編集オブジェクトのサイズを変更する                                |
//+------------------------------------------------------------------+
bool EditChangeSize(const long   chart_ID=0,  // チャート識別子
                    const string name="Edit", // オブジェクト名
                    const int    width=0,     // 幅
                    const int    height=0)    // 高さ
  {
//--- エラー値をリセットする
   ResetLastError();
//--- オブジェクトサイズを変更する
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": failed to change the object width! Error code = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": failed to change the object height! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 編集オブジェクトのテキストを変更する                              |
//+------------------------------------------------------------------+
bool EditTextChange(const long   chart_ID=0,  // チャート識別子
                    const string name="Edit", // オブジェクト名
                    const string text="Text") // テキスト
  {
//--- エラー値をリセットする
   ResetLastError();
//--- オブジェクトのテキストを変更する
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": failed to change the text! Error code = ",GetLastError());
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 編集オブジェクトのテキストを返す                                 |
//+------------------------------------------------------------------+
bool EditTextGet(string      &text,         // テキスト
                 const long   chart_ID=0,  // チャート識別子
                 const string name="Edit") // オブジェクト名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- オブジェクトのテキストを取得する
   if(!ObjectGetString(chart_ID,name,OBJPROP_TEXT,0,text))
     {
      Print(__FUNCTION__,
            ": failed to get the text! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 編集オブジェクトを削除する                                       |
//+------------------------------------------------------------------+
bool EditDelete(const long   chart_ID=0,  // チャート識別子
                const string name="Edit") // オブジェクト名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- ラベルを削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete \"Edit\" object! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- チャートウィンドウサイズ
   long x_distance;
   long y_distance;
//--- ウィンドウサイズを設定する
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Failed to get the chart width! Error code = ",GetLastError());
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("Failed to get the chart height! Error code = ",GetLastError());
      return;
     }
//--- 編集フィールド変更のステップを定義する
   int x_step=(int)x_distance/64;
//--- 編集フィールド座標とサイズを設定する
   int x=(int)x_distance/8;
   int y=(int)y_distance/2;
   int x_size=(int)x_distance/8;
   int y_size=InpFontSize*2;
//--- テキストをローカル変数に収納する
   string text=InpText;
//--- 編集フィールドを作成する
   if(!EditCreate(0,InpName,0,x,y,x_size,y_size,InpText
,InpFont,InpFontSize,InpAlign,InpReadOnly,
      InpCorner
,InpColor,InpBackColor,InpBorderColor,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- 編集フィールドを伸ばす
   while(x_size-x<x_distance*5/8)
     {
      //--- 編集フィールドの幅を増やす
      x_size+=x_step;
      if(!EditChangeSize(0,InpName,x_size,y_size))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画して 0.05 秒待つ
      ChartRedraw();
      Sleep(50);
     }
//--- 半秒の遅れ
   Sleep(500);
//--- テキストを変更する
   for(int i=0;i<20;i++)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 「+」を始めと終わりに足す
      text="+"+text+"+";
      if(!EditTextChange(0,InpName,text))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画して 0.1 秒待つ
      ChartRedraw();
      Sleep(100);
     }
//--- 半秒の遅れ
   Sleep(500);
//--- 編集フィールドを削除する
   EditDelete(0,InpName);
   ChartRedraw();
//---1 秒待つ
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_EVENT

イベント オブジェクト



注 意 事 項

イベント にマウスをホーバーすると、テキスト が表 示 されます。

例

次 のスクリプト は、チャート 上 でイベント オブジェクト を作 成 して移 動 します。グラフィックオブジェクト のプロパティを作 成 そして変 更 するために特 別 な関 数 が開 発 されています。これらの関 数 はお使 いのアプリケーションでそのまま使 用 することが出 来 ます。

```
//--- 説明
#property description "Script draws \"Event\" graphical object."
#property description "Anchor point date is set in percentage of"
#property description "the chart window width in bars."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string           InpName="Event";     // イベント名
input int              InpDate=25;          // イベントの日付( % )
input string           InpText="Text";      // イベントテキスト
input color            InpColor=clrRed;     // イベントの色
input int              InpWidth=1;          // 強調表示時のポイントサイズ
input bool             InpBack=false;       // 背景イベント
input bool             InpSelection=false; // 強調表示して移動
input bool             InpHidden=true;      // オブジェクトリストに隠す
```

© 2000–2025, MetaQuotes Ltd.

```
input long              InpZOrder=0;         // マウスクリックの優先順位
//+------------------------------------------------------------------+
//| チャートにイベントオブジェクトを作成する                         |
//+------------------------------------------------------------------+
bool EventCreate(const long            chart_ID=0,       // チャート識別子
                 const string          name="Event",     // イベント名
                 const int             sub_window=0,     // サブウィンドウ番号
                 const string          text="Text",      // イベントテキスト
                 datetime              time=0,           // 時間
                 const color           clr=clrRed,       // 色
                 const int             width=1,          // 強調表示時のポイントの幅
                 const bool            back=false,       // 背景で表示する
                 const bool            selection=false,  // 強調表示して移動
                 const bool            hidden=true,      // オブジェクトリストに隠す
                 const long            z_order=0)        // マウスクリックの優先順位
  {
//--- 時間が設定されていない場合、最後のバーにオブジェクトを作成する
   if(!time)
      time=TimeCurrent();
//--- エラー値をリセットする
   ResetLastError();
//--- イベントオブジェクトを作成する
   if(!ObjectCreate(chart_ID,name,OBJ_EVENT,sub_window,time,0))
     {
      Print(__FUNCTION__,
            ": failed to create \"Event\" object! Error code = ",GetLastError());
      return(false);
     }
//--- イベントテキストを設定
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- 色を設定
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- オブジェクトがハイライトされた時のアンカーポイントの幅を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- マウスでイベントを移動させるモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| イベントオブジェクトのテキストを変更する                         |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool EventTextChange(const long   chart_ID=0,   // チャート識別子
                     const string name="Event", // イベント名
                     const string text="Text")  // テキスト
  {
//--- エラー値をリセットする
   ResetLastError();
//--- オブジェクトのテキストを変更する
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": failed to change the text! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| イベントオブジェクトを移動する                                   |
//+------------------------------------------------------------------+
bool EventMove(const long   chart_ID=0,   // チャート識別子
               const string name="Event", // イベント名
               datetime     time=0)       // 時間
  {
//--- 時間が設定されていない場合、イベントを最後のバーに移動する
   if(!time)
      time=TimeCurrent();
//--- エラー値をリセットする
   ResetLastError();
//--- オブジェクトを移動する
   if(!ObjectMove(chart_ID,name,0,time,0))
     {
      Print(__FUNCTION__,
            ": failed to move \"Event\" object! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| イベントオブジェクトを削除する                                   |
//+------------------------------------------------------------------+
bool EventDelete(const long   chart_ID=0,   // チャート識別子
                 const string name="Event") // イベント名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- オブジェクトを削除する
   if(!ObjectDelete(chart_ID,name))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print(__FUNCTION__,
            ": failed to delete \"Event\" object! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力パラメータの正しさをチェックする
   if(InpDate<0 || InpDate>100)
     {
      Print("Error! Incorrect values of input parameters!");
      return;
     }
//--- チャートウィンドウで表示されているバーの数
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- イベントアンカーポイントの座標の設定と変更に使用される日付と
//--- 価格の値を格納するための配列
   datetime date[];
//--- メモリ割り当て
   ArrayResize(date,bars);
//--- 日付配列に書き込む
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Failed to copy time values! Error code = ",GetLastError());
      return;
     }
//--- オブジェクト作成のポイントを定義する
   int d=InpDate*(bars-1)/100;
//--- イベントオブジェクトを作成する
   if(!EventCreate(0,InpName,0,InpText,date[d],InpColor,InpWidth,
      InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- オブジェクトを移動する
//--- ループカウンタ
   int h_steps=bars/2;
//--- オブジェクトを移動する
   for(int i=0;i<h_steps;i++)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 次の値を使用する
      if(d<bars-1)
         d+=1;
      //--- ポイントを移動する
      if(!EventMove(0,InpName,date[d]))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画する
      ChartRedraw();
      // 0.05 秒の遅れ
      Sleep(50);
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- チャートからチャンネルを削除する
   EventDelete(0,InpName);
   ChartRedraw();
//--- 1 秒の遅れ
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_RECTANGLE_LABEL

四角形ラベルオブジェクト



注意事項

アンカーポイントの座標はピクセルで設定されています。四角形ラベルのアンカーポイントとして使用される隅は ENUM_BASE_CORNER 列挙から選択出来ます。四角形ラベルの境界線の種類は ENUM_BORDER_TYPE 列挙から選択出来ます。

このオブジェクトは、カスタムグラフィカルインタフェースの作成と設計に使用されます。

例

次のスクリプトは、チャート上で四角形ラベルオブジェクトを作成して移動します。グラフィックオブジェクトのプロパティを作成そして変更するために特別な関数が開発されています。これらの関数はお使いのアプリケーションでそのまま使用することが出来ます。

```
//--- 説明
#property description "Script creates \"Rectangle Label\" graphical object."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- スクリプト入力パラメータ
input string           InpName="RectLabel";          // ラベル名
input color            InpBackColor=clrSkyBlue;      // 背景色
input ENUM_BORDER_TYPE InpBorder=BORDER_FLAT;        // 境界線の種類
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // アンカーに使用されるチャートのコーナー
input color            InpColor=clrDarkBlue;         // フラット境界線の色（Flat）
input ENUM_LINE_STYLE  InpStyle=STYLE_SOLID;         // フラット境界スタイル（Flat）
input int              InpLineWidth=3;               // フラット境界幅（Flat）
```

© 2000-2025, MetaQuotes Ltd.

```
input bool              InpBack=false;              // 背景オブジェクト
input bool              InpSelection=true;          // 強調表示して移動
input bool              InpHidden=true;             // オブジェクトリストに隠す
input long              InpZOrder=0;                // マウスクリックの優先順位
//+------------------------------------------------------------------+
//|  四角形ラベルを作成する                                          |
//+------------------------------------------------------------------+
bool RectLabelCreate(const long             chart_ID=0,               // チャート識別子
                     const string           name="RectLabel",         // ラベル名
                     const int              sub_window=0,             // サブウィンドウ番
号
                     const int              x=0,                      // X 座標
                     const int              y=0,                      // Y 座標
                     const int              width=50,                 // 幅
                     const int              height=18,                // 高さ
                     const color            back_clr=C'236,233,216',  // 背景色
                     const ENUM_BORDER_TYPE border=BORDER_SUNKEN,     // 境界線の種類
                     const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER, // アンカーに使用さ
れるチャートのコーナー
                     const color            clr=clrRed,               // フラット境界線の
色（Flat）
                     const ENUM_LINE_STYLE  style=STYLE_SOLID,        // フラット境界スタ
イル
                     const int              line_width=1,             //フラット境界幅
                     const bool             back=false,               // 背景で表示する
                     const bool             selection=false,          // 強調表示して移
動
                     const bool             hidden=true,              // オブジェクトリスト
に隠す
                     const long             z_order=0)                // マウスクリックの優
先順位
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 四角形ラベルを作成する
   if(!ObjectCreate(chart_ID,name,OBJ_RECTANGLE_LABEL,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": failed to create a rectangle label! Error code = ",GetLastError());
      return(false);
     }
//--- ラベル座標を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- ラベルサイズを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- 背景色を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_BGCOLOR,back_clr);
//--- 境界線を設定する
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_TYPE,border);
//--- ポイント座標が相対的に定義されているチャートのコーナーを設定
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- フラット境界線色を設定する(Flat モード)
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- フラット境界線スタイルを設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- フラット境界線幅を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,line_width);
//--- 前景(false)または背景(true)に表示
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- マウスでラベルを移動させるモードを有効(true)か無効(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- オブジェクトリストのグラフィックオブジェクトを非表示(true)か表示(false)にする
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- チャートのマウスクリックのイベントを受信するための優先順位を設定する
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 四角形ラベルを移動する                                           |
//+------------------------------------------------------------------+
bool RectLabelMove(const long   chart_ID=0,        // チャート識別子
                   const string name="RectLabel", // ラベル名
                   const int    x=0,              // X 座標
                   const int    y=0)              // Y 座標
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 四角形ラベルを移動する
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": failed to move X coordinate of the label! Error code =
",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": failed to move Y coordinate of the label! Error code =
",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  四角形ラベルのサイズを変更する                                  |
//+------------------------------------------------------------------+
bool RectLabelChangeSize(const long   chart_ID=0,        // チャート識別子
                         const string name="RectLabel", // ラベル名
                         const int    width=50,          // ラベル幅
                         const int    height=18)         // ラベルの高さ
  {
//--- エラー値をリセットする
   ResetLastError();
//--- ラベルサイズを変更する
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": failed to change the label's width! Error code = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": failed to change the label's height! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  四角形ラベルの境界線の種類を変更する                            |
//+------------------------------------------------------------------+
bool RectLabelChangeBorderType(const long             chart_ID=0,            // チャート識別子
                               const string           name="RectLabel",      // ラベル名
                               const ENUM_BORDER_TYPE border=BORDER_SUNKEN) // 境界線の種類
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 境界線の種類を変更する
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_TYPE,border))
     {
      Print(__FUNCTION__,
            ": failed to change the border type! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000–2025, MetaQuotes Ltd.

```
//|  四角形ラベルを削除する                                                  |
//+------------------------------------------------------------------+
bool RectLabelDelete(const long   chart_ID=0,        // チャート識別子
                     const string name="RectLabel") // ラベル名
  {
//--- エラー値をリセットする
   ResetLastError();
//--- ラベルを削除する
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete a rectangle label! Error code = ",GetLastError());
      return(false);
     }
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- チャートウィンドウサイズ
   long x_distance;
   long y_distance;
//--- ウィンドウサイズを設定する
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Failed to get the chart width! Error code = ",GetLastError());
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("Failed to get the chart height! Error code = ",GetLastError());
      return;
     }
//--- 四角形ラベル座標を定義する
   int x=(int)x_distance/4;
   int y=(int)y_distance/4;
//--- ラベルサイズを設定する
   int width=(int)x_distance/4;
   int height=(int)y_distance/4;
//--- 四角形ラベルを作成する
   if(!RectLabelCreate(0,InpName,0,x,y,width,height,InpBackColor
,InpBorder,InpCorner,
      InpColor,InpStyle,InpLineWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
```

© 2000–2025, MetaQuotes Ltd.

```
//--- チャートを再描画して1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- 四角形ラベルのサイズを変更する
   int steps=(int)MathMin(x_distance/4,y_distance/4);
   for(int i=0;i<steps;i++)
     {
      //--- サイズ変更
      width+=1;
      height+=1;
      if(!RectLabelChangeSize(0,InpName,width,height))
         return;
      //--- スクリプトの動作が強制的に無効にされているかどうかをチェックする
      if(IsStopped())
         return;
      //--- チャートを再描画して 0.01 秒待つ
      ChartRedraw();
      Sleep(10);
     }
//--- 1 秒の遅れ
   Sleep(1000);
//--- 境界線の種類を変更する
   if(!RectLabelChangeBorderType(0,InpName,BORDER_RAISED))
      return;
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- 境界線の種類を変更する
   if(!RectLabelChangeBorderType(0,InpName,BORDER_SUNKEN))
      return;
//--- チャートを再描画して 1 秒待つ
   ChartRedraw();
   Sleep(1000);
//--- ラベルを削除する
   RectLabelDelete(0,InpName);
   ChartRedraw();
//---1 秒待つ
   Sleep(1000);
//---
  }
```

© 2000–2025, MetaQuotes Ltd.

オブジェクトプロパティ

グラフィックオブジェクトは、オブジェクトタイプに応じて、多数のプロパティを有することができます。オブジェクトプロパティの値の設定と取得は、グラフィックオブジェクトを使う作業の関連する関数によって行われます。

テクニカル分析で使われる全てのオブジェクトは、価格と時間の座標ごとのチャート上にアンカーを持っています。(トレンドライン、チャンネル、フィボナッチツールなど)しかし、インターフェイスの改善の為に設計されたサポートオブジェクト群があり、これらは常にチャートの可視部分にアンカーを持っています。(メインチャートウィンドウまたはインディケータのサブウィンドウ)

オブジェクト	インディケータ	X/Y	Width/Height	Date/Price	OBJPROP_CORNER	OBJPROP_ANCHOR	OBJPROP_ANGLE
Text	OBJ_TEXT	–	–	はい	–	はい	はい
Label	OBJ_LABEL	はい	はい(読み取り専用)	–	はい	はい	はい
Button	OBJ_BUTTON	はい	はい	–	はい	–	–
Bitmap	OBJ_BITMAP	–	はい(読み取り専用)	はい	–	はい	–
Bitmap Label	OBJ_BITMAP_LABEL	はい	はい(読み取り専用)	–	はい	はい	–
Edit	OBJ_EDIT	はい	はい	–	はい	–	–
Rectangle Label	OBJ_RECTANGLE_LABEL	はい	はい	–	はい	–	–

表には次の名称を使用しています。

- **X/Y**—チャートの隅の1つに対してピクセル単位でアンカーポイントが指定されます。
- **Width/Height**—オブジェクトの幅と高さ。『読み取り専用』が指定されている場合、チャート上にオブジェクトが描画された後にのみ、幅と高さの値が計算されることを意味しています。
- **Date/Price**—アンカーポイントが日付と価格のペアで指定されます。
- **OBJPROP_CORNER**—チャートの隅を指定し、それに対してアンカーポイントが指定されます。ENUM_BASE_CORNERリストの4つの値のうちの1つになる場合があります。
- **OBJPROP_ANCHOR**—オブジェクトでのアンカーポイントの位置を指定します。ENUM_ANCHOR_POINTリストの9つの値のうちの1つになる場合があります。このポイントから選択したチャートの隅まで、ピクセル単位で座標が表示されます。
- **OBJPROP_ANGLE**—反時計回りにオブジェクトの回転角度を指定します。

© 2000–2025, MetaQuotes Ltd.

グラフィックオブジェクトのプロパティを定義する関数、またチャートでオブジェクトを作成して移動する ObjectCreate() 及び ObjectMove() は実際にチャートにコマンドを送るのに使用されます。これらの関数が正常に実行された場合、コマンドがチャートイベントの共用キューに追加されます。グラフィックオブジェクトのプロパティの視覚変更は、チャートイベントのキューを処理する際に実装されています。

従って、これらの関数を呼び出した後、チャートの視覚更新が即時行われると期待してはいけません。一般的には、チャートは、新たな相場の到着やチャートウィンドウのサイズ変更などの変更イベントの後に、端末によって自動更新されます。強制的にグラフィックオブジェクトを更新するには ChartRedraw() 関数を使用します。

ObjectSetInteger() 及び ObjectGetInteger() 関数

ENUM_OBJECT_PROPERTY_INTEGER

識別子	説明	プロパティ型
OBJPROP_COLOR	色。	color
OBJPROP_STYLE	スタイル。	ENUM_LINE_STYLE
OBJPROP_WIDTH	線の太さ。	int
OBJPROP_BACK	背景のオブジェクト。	bool
OBJPROP_ZORDER	チャート上のクリックイベント（CHARTEVENT_CLICK）を受け取る	long

© 2000–2025, MetaQuotes Ltd.

識別子	説明	プロパティ型
	ためのグラフィカルオブジェクトの優先順位。優先順位はオブジェクトの作成時にデフォルトのゼロに設定され、必要に応じて高く出来ます。オブジェクトが重なっ	

© 2000–2025, MetaQuotes Ltd.

識別子	説明	プロパティ型
	た場合、最も優先順位の高いオブジェクトのみがCHARTEVENT_CLICKイベントを受け取ります。	
OBJPROP_FILL	色でオブジェクトを塗りつぶします（OBJ_RECTANGLE、OBJ_TRIANGLE、OBJ	bool

© 2000–2025, MetaQuotes Ltd.

識別子	説明	プロパティ型
	_ELLIPSE, OBJ_CHANNEL、OBJ_STDDEVCHANNEL、OBJ_REGRESSION）。	
OBJPROP_HIDDEN	端末の「チャート設定」->「オブジェクト」->「オブジェクト表示リスト」メニューで、オブジェクトリス	bool

© 2000–2025, MetaQuotes Ltd.

識別子	説明	プロパティ型
	トのグラフィカルオブジェクトの名称を非表示にします。trueは、リストからのオブジェクトを非表示にします。MQL5 プログラムで作成されたカレンダーイベントや取引履歴を	

© 2000–2025, MetaQuotes Ltd.

識別子	説明	プロパティ型
	表示するオブジェクトではデフォルトはtrueです。そのようなグラフィックオブジェクトを表示してプロパティにアクセスするには、「オブジェクト表示リスト」ウィンドウの「すべて」ボタ	

© 2000–2025, MetaQuotes Ltd.

識別子	説明	プロパティ型
	ンをクリックします。	
OBJPROP_SELECTED	オブジェクトが選択されています。	bool
OBJPROP_READONLY	編集オブジェクトでテキストを編集する能力。	bool
OBJPROP_TYPE	オブジェクトの型。	ENUM_OBJECT r/o
OBJPROP_TIME	時間座標。	datetime 修飾子=アンカーポイントの数
OBJPROP_SELECTABLE	オブジェクトの可用性。	bool

© 2000–2025, MetaQuotes Ltd.

識別子	説明	プロパティ型
OBJPROP_CREATETIME	オブジェクト作成時刻。	datetime r/o
OBJPROP_LEVELS	レベルの数。	int
OBJPROP_LEVELCOLOR	ラインレベルの色。	color 修飾子=レベル番号
OBJPROP_LEVELSTYLE	ラインレベルのスタイル。	ENUM_LINE_STYLE modifier=level number
OBJPROP_LEVELWIDTH	ラインレベルの太さ。	int 修飾子=レベル番号
OBJPROP_ALIGN	編集オブジェクト（OBJ_EDIT）の水平テキストの配置。	ENUM_ALIGN_MODE
OBJPROP_FONTSIZE	フォントサイズ。	int

© 2000–2025, MetaQuotes Ltd.

識別子	説明	プロパティ型
OBJPROP_RAY_LEFT	レイは左に行きます。	bool
OBJPROP_RAY_RIGHT	レイは右に行きます。	bool
OBJPROP_RAY	縦線はチャートの全てのウィンドウを通過します。	bool
OBJPROP_ELLIPSE	フィボナッチアークオブジェクトの完全な楕円形の表	bool

© 2000–2025, MetaQuotes Ltd.

識別子	説明	プロパティ型
	示 (OBJ_FIBOARC)。	
OBJPROP_ARROWCODE	矢印オブジェクトの矢印コード。	uchar
OBJPROP_TIMEFRAMES	時間軸でのオブジェクトの可視性。	フラグセット flags
OBJPROP_ANCHOR	グラフィックオブジェクトのアンカーポイントの位置。	ENUM_ARROW_ANCHOR (for OBJ_ARROW), ENUM_ANCHOR_POINT (for OBJ_LABEL, OBJ_BITMAP_LABEL and OBJ_TEXT)
OBJPROP_XDISTANCE	結合角からX軸に	int

© 2000–2025, MetaQuotes Ltd.

識別子	説明	プロパティ型
	沿ったピクセル単位での距離（注意事項を参照）。	
OBJPROP_YDISTANCE	結合角からY軸に沿ったピクセル単位での距離（注意事項を参照）。	int
OBJPROP_DIRECTION	ギャンオブジェクトのトレンド。	ENUM_GANN_DIRECTION

© 2000–2025, MetaQuotes Ltd.

識別子	説明	プロパティ型
OBJPROP_DEGREE	エリオット波動マーキングのレベル。	ENUM_ELLIOT_WAVE_DEGREE
OBJPROP_DRAWLINES	エリオット波動をマークするための線の表示。	bool
OBJPROP_STATE	ボタンの状態（押された/押されていない）。	bool
OBJPROP_CHART_ID	チャートオブジェクト（OBJ_CHART）の識別子。	long r/o

© 2000–2025, MetaQuotes Ltd.

識別子	説明	プロパティ型
	チャート操作で説明される関数を使用すると、通常のチャートと同様にこのオブジェクトのプロパティの操作が可能になりますが例外もあります。	
OBJPROP_XSIZE	X軸に沿ったピクセ	int

© 2000–2025, MetaQuotes Ltd.

識別子	説明	プロパティ型
	ル単位でのオブジェクトの幅。OBJ_LABEL（読み込みのみ）、OBJ_BUTTON, OBJ_CHART、OBJ_BITMAP、OBJ_BITMAP_LABEL、OBJ_EDIT、OBJ_RECTANGLE_LABEL オブジェ	

© 2000–2025, MetaQuotes Ltd.

識別子	説明	プロパティ型
	クトに指定されます。	
OBJPROP_YSIZE	Y軸に沿ったオブジェクトの高さ(ピクセル単位)。OBJ_LABEL(読み込みのみ)、OBJ_BUTTON, OBJ_CHART、OBJ_BITMAP、OBJ_BITMAP_LABEL、OBJ	int

© 2000–2025, MetaQuotes Ltd.

識別子	説明	プロパティ型
	_EDIT、OBJ_RECTANGLE_LABEL オブジェクトに指定されます。	
OBJPROP_XOFFSET	ビットマップラベル（OBJ_BITMAP_LABEL）及びビットマップ（OBJ_BITMAP）グラフィックオブジェクトの四角	int

© 2000–2025, MetaQuotes Ltd.

識別子	説明	プロパティ型
	形の表示領域の左上隅のX座標。値は、元の画像の左上隅を基準にピクセル単位で設定されています。	
OBJPROP_YOFFSET	ビットマップラベル（OBJ_BITMAP_LABEL）及	int

© 2000–2025, MetaQuotes Ltd.

識別子	説明	プロパティ型
	びビットマップ（OBJ_BITMAP）グラフィックオブジェクトの四角形の表示領域の左上隅のY座標。値は、元の画像の左上隅を基準にピクセル単	

© 2000–2025, MetaQuotes Ltd.

識別子	説明	プロパティ型
	位で設定されています。	
OBJPROP_PERIOD	チャートオブジェクトの時間軸。	ENUM_TIMEFRAMES
OBJPROP_DATE_SCALE	グラフオブジェクトの時間スケールの表示。	bool
OBJPROP_PRICE_SCALE	チャートオブジェクトの価格スケールの表示。	bool
OBJPROP_CHART_SCALE	チャートオブジェクトのス	int　0～5

© 2000–2025, MetaQuotes Ltd.

識別子	説明	プロパティ型
	ケール。	
OBJPROP_BGCOLOR	OBJ_EDIT、OBJ_BUTTON、OBJ_RECTANGLE_LABEL の背景色。	color
OBJPROP_CORNER	グラフィカルオブジェクトをリンクするチャートのコーナー。	ENUM_BASE_CORNER
OBJPROP_BORDER_TYPE	「四角形ラベル」オブジェクトの境界線の	ENUM_BORDER_TYPE

© 2000–2025, MetaQuotes Ltd.

識別子	説明	プロパティ型
	種類。	
OBJPROP_BORDER_COLOR	OBJ_EDIT 及び OBJ_BUTTON オブジェクトの境界線の色。	color

チャートオブジェクト（OBJ_CHART）のチャート操作の際、下記の制限が適用されます。

- ChartClose() を使用して閉じるのは不可能
- シンボルや期間の ChartSetSymbolPeriod() 関数での変更は不可能
- CHART_SCALE、CHART_BRING_TO_TOP、CHART_SHOW_DATE_SCALE 及び CHART_SHOW_PRICE_SCALE（ENUM_CHART_PROPERTY_INTEGER）プロパティは無効

OBJ_BITMAP_LABEL 及び OBJ_BITMAP オブジェクトの特別画像表示モードが設定出来ます。このモードでは、元の画像（矩形可視領域が適用）の一部のみが表示され、画像の残りの部分は見えなくなります。この領域の大きさは、OBJPROP_XSIZE とOBJPROP_YSIZE プロパティを使用して設定される必要があります。表示領域は OBJPROP_XOFFSET 及び OBJPROP_YOFFSET プロパティを使用して「移動する」ことが出来ます。

固定サイズのオブジェクトの場合 rOBJ_BUTTON、OBJ_RECTANGLE_LABEL、OBJ_EDIT 及び OBJ_CHART と OBJPROP_XDISTANCE 及び OBJPROP_YDISTANCE プロパティがピクセル単位の X とY 座標がゼロとなるチャートのコーナー（OBJPROP_CORNER）を基準にしたオブジェクトの左上の点の位置を設定します。

ObjectSetDouble() 及び ObjectGetDouble() 関数

ENUM_OBJECT_PROPERTY_DOUBLE

© 2000–2025, MetaQuotes Ltd.

識別子	説明	プロパティ型
OBJPROP_PRICE	価格座標。	double 修飾子 = アンカーポイントの数
OBJPROP_LEVELVALUE	レベル値。	double 修飾子 = レベル番号
OBJPROP_SCALE	スケール（ギャンオブジェクトとフィボナッチアークのプロパティ）。	double
OBJPROP_ANGLE	角度。角度を指定されずにプログラムから作成されたオブジェクトの場合、値は	double

© 2000–2025, MetaQuotes Ltd.

識別子	説明	プロパティ型
	EMPTY_VALUE。	
OBJPROP_DEVIATION	標準偏差チャンネルの偏差。	double

ObjectSetString() 及び ObjectGetString(). 関数

ENUM_OBJECT_PROPERTY_STRING

識別子	説明	プロパティ型
OBJPROP_NAME	オブジェクト名。	string
OBJPROP_TEXT	オブジェクトの説明（オブジェクトに含まれるテキスト）。	string
OBJPROP_TOOLTIP	ツールヒント	string

© 2000–2025, MetaQuotes Ltd.

識別子	説明	プロパティ型
	のテキスト。プロパティが設定されていない場合は、端末によって自動的に生成されたツールヒントが表示されます。ツールヒントは「¥n」（改行）の値を設定すること	

© 2000–2025, MetaQuotes Ltd.

識別子	説明	プロパティ型
	によって無効に出来ます。	
OBJPROP_LEVELTEXT	レベルの説明。	string 修飾子 = レベル番号
OBJPROP_FONT	フォント。	string
OBJPROP_BMPFILE	ビットマップラベルのBMPファイルの名称。リソースも参照。	string 修飾子：0–ON、1–OFF
OBJPROP_SYMBOL	チャートオブジェクトの記号。	string

OBJ_RECTANGLE_LABEL オブジェクト（「四角形ラベル」）では、次の ENUM_BORDER_TYPE の値に一致するように、3 つの設計モードを設定することが出来ます。

© 2000–2025, MetaQuotes Ltd.

ENUM_BORDER_TYPE

識別子	説明
BORDER_FLAT	フラット フォーム。
BORDER_RAISED	プロミネント フォーム。
BORDER_SUNKEN	凹フォーム。

OBJ_EDIT オブジェクト（「編集」）及び ChartScreenShot() 関数では、ENUM_ALIGN_MODE 列挙体の値を使用して、水平配向型を指定することが出来ます。

ENUM_ALIGN_MODE

識別子	説明
ALIGN_LEFT	左揃え。
ALIGN_CENTER	中央揃え（編集オブジェクトのみ）。
ALIGN_RIGHT	右揃え。

例：

```
#define  UP          "\x0431"
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   string label_name="my_OBJ_LABEL_object";
   if(ObjectFind(0,label_name)<0)
     {
      Print("Object ",label_name," not found. Error code = ",GetLastError());
      //--- ラベルオブジェクトを作成する
      ObjectCreate(0,label_name,OBJ_LABEL,0,0,0);
      //--- X 座標を設定する
      ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,200);
      //--- Y 座標を設定する
      ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,300);
      //--- テキストの色を定義する
      ObjectSetInteger(0,label_name,OBJPROP_COLOR,clrWhite);
      //--- オブジェクトラベルのテキストを定義する
      ObjectSetString(0,label_name,OBJPROP_TEXT,UP);
      //--- フォントを定義する
      ObjectSetString(0,label_name,OBJPROP_FONT,"Wingdings");
      //--- フォントサイズを定義する
      ObjectSetInteger(0,label_name,OBJPROP_FONTSIZE,10);
      //--- 45 度の時計回り回転
```

© 2000-2025, MetaQuotes Ltd.

```
      ObjectSetDouble(0,label_name,OBJPROP_ANGLE,-45);
      //--- マウスでの選択を無効にする
      ObjectSetInteger(0,label_name,OBJPROP_SELECTABLE,false);
      //--- チャートで描画する
      ChartRedraw(0);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

オブジェクト結合の方法

グラフィックオブジェクト Text、Label、Bitmap、そしてBitmap Label（OBJ_TEXT、OBJ_LABEL、OBJ_BITMAP、OBJ_BITMAP_LABEL）は、OBJPROP_ANCHORのプロパティで設定した、異なる9つの方法から1つの結合座標を持つことができます。

オブジェクト	インディケータ	X/Y	Width/Height	Date/Price	OBJPROP_CORNER	OBJPROP_ANCHOR	OBJPROP_ANGLE
Text	OBJ_TEXT	—	—	はい	—	はい	はい
Label	OBJ_LABEL	はい	はい（読み取り専用）	—	はい	はい	はい
Button	OBJ_BUTTON	はい	はい	—	はい	—	—
Bitmap	OBJ_BITMAP	—	はい（読み取り専用）	はい	—	はい	—
Bitmap Label	OBJ_BITMAP_LABEL	はい	はい（読み取り専用）	—	はい	はい	—
Edit	OBJ_EDIT	はい	はい	—	はい	—	—
Rectangle Label	OBJ_RECTANGLE_LABEL	はい	はい	—	はい	—	—

表には次の名称を使用しています。

- **X/Y**—チャートの隅の1つに対してピクセル単位でアンカーポイントが指定されます。
- **Width/Height**—オブジェクトの幅と高さ。『読み取り専用』が指定されている場合、チャート上にオブジェクトが描画された後にのみ、幅と高さの値が計算されることを意味しています。
- **Date/Price**—アンカーポイントが日付と価格のペアで指定されます。
- **OBJPROP_CORNER**—チャートの隅を指定し、それに対してアンカーポイントが指定されます。ENUM_BASE_CORNERリストの4つの値のうちの1つになる場合があります。
- **OBJPROP_ANCHOR**—オブジェクトでのアンカーポイントの位置を指定します。ENUM_ANCHOR_POINTリストの9つの値のうちの1つになる場合があります。このポイントから選択したチャートの隅まで、ピクセル単位で座標が表示されます。
- **OBJPROP_ANGLE**—反時計回りにオブジェクトの回転角度を指定します。

必要なバージョンは ObjectSetInteger(chart_handle, object_name, OBJPROP_ANCHOR, anchor_point_mode) 関数で指定出来ます。ここで anchor_point_mode は ENUM_ANCHOR_POINT 値のいずれかです。

ENUM_ANCHOR_POINT

© 2000–2025, MetaQuotes Ltd.

ID	説明
ANCHOR_LEFT_UPPER	左上の隅のアンカーポイント。
ANCHOR_LEFT	中心の左側のアンカーポイント。
ANCHOR_LEFT_LOWER	左下の隅のアンカーポイント。
ANCHOR_LOWER	中心の下部のアンカーポイント。
ANCHOR_RIGHT_LOWER	右下の隅のアンカーポイント。
ANCHOR_RIGHT	中心の右側のアンカーポイント。
ANCHOR_RIGHT_UPPER	右上の隅のアンカーポイント。
ANCHOR_UPPER	中心の上のアンカーポイント。
ANCHOR_CENTER	厳密にオブジェクトの中心にあるアンカーポイント。

OBJ_BUTTON、OBJ_RECTANGLE_LABEL、OBJ_EDIT 及び OBJ_CHART オブジェクトは、左上隅（ANCHOR_LEFT_UPPER）に固定アンカーポイントを持っています。

例：

```
string text_name="my_OBJ_TEXT_object";
if(ObjectFind(0,text_name)<0)
  {
   Print("Object ",text_name," not found. Error code = ",GetLastError());
   //--- チャートの最高値を取得する
   double chart_max_price=ChartGetDouble(0,CHART_PRICE_MAX,0);
   //--- オブジェクトラベルを作成する
   ObjectCreate(0,text_name,OBJ_TEXT,0,TimeCurrent(),chart_max_price);
   //--- テキストの色を設定する
   ObjectSetInteger(0,text_name,OBJPROP_COLOR,clrWhite);
   //--- 背景色を設定する
   ObjectSetInteger(0,text_name,OBJPROP_BGCOLOR,clrGreen);
   //--- ラベルオブジェクトのテキストを設定する
   ObjectSetString(0,text_name,OBJPROP_TEXT,TimeToString(TimeCurrent()));
   //--- フォントを設定する
   ObjectSetString(0,text_name,OBJPROP_FONT,"Trebuchet MS");
   //--- フォントサイズを設定する
   ObjectSetInteger(0,text_name,OBJPROP_FONTSIZE,10);
   //--- 右上隅に結合する
   ObjectSetInteger(0,text_name,OBJPROP_ANCHOR,ANCHOR_RIGHT_UPPER);
   //--- 反時計回りに 90 度回転
   ObjectSetDouble(0,text_name,OBJPROP_ANGLE,90);
   //--- マウスでの選択を禁止する
   ObjectSetInteger(0,text_name,OBJPROP_SELECTABLE,false);
   //--- オブジェクトの再描画
   ChartRedraw(0);
  }
```

© 2000-2025, MetaQuotes Ltd.

矢印グラフィカルオブジェクト（OBJ_ARROW）の座標を結ぶ方法は 2 つのみです。識別子は ENUM_ARROW_ANCHOR でみられます。

ENUM_ARROW_ANCHOR

ID	説明
ANCHOR_TOP	上面にアンカー
ANCHOR_BOTTOM	下面にアンカー

例：

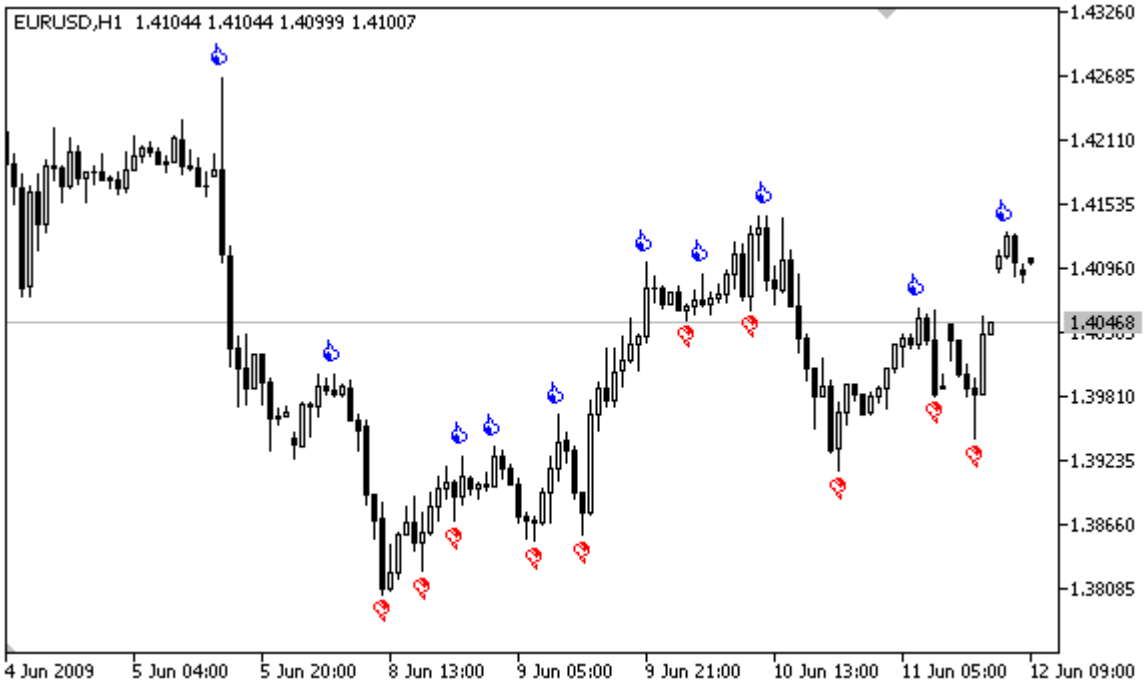
```
void OnStart()
  {
//--- 補助配列
   double Ups[],Downs[];
   datetime Time[];
//--- 配列を時系列として設定する
   ArraySetAsSeries(Ups,true);
   ArraySetAsSeries(Downs,true);
   ArraySetAsSeries(Time,true);
//--- 指標フラクタルのハンドルを作成する
   int FractalsHandle=iFractals(NULL,0);
   Print("FractalsHandle = ",FractalsHandle);
//--- 直近エラー値をゼロに設定する
   ResetLastError();
//--- 指標値のコピーを試みる
   int copied=CopyBuffer(FractalsHandle,0,0,1000,Ups);
   if(copied<=0)
     {
      Print("Unable to copy the upper fractals. Error = ",GetLastError());
      return;
     }

   ResetLastError();
//--- 指標値のコピーを試みる
   copied=CopyBuffer(FractalsHandle,1,0,1000,Downs);
   if(copied<=0)
     {
      Print("Unable to copy the bottom fractals. Error = ",GetLastError());
      return;
     }

   ResetLastError();
//--- 最後の 1000 のバーオープニングを含む時系列を複製する
   copied=CopyTime(NULL,0,0,1000,Time);
   if(copied<=0)
     {
      Print("Unable to copy the Opening Time of the last 1000 bars");
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }

   int upcounter=0,downcounter=0; // 矢印の数を数える
   bool created;// オブジェクト作成の試みの結果を受け取る
   for(int i=2;i<copied;i++)// 指標 iFractals の値をみる
     {
      if(Ups[i]!=EMPTY_VALUE)// 上部のフラクタルを発見
        {
         if(upcounter<10)// 10 以上の上向き矢印は作成しない
           {
            //--- 「Up」オブジェクトの作成を試みる
            created=ObjectCreate(0,string(Time[i]),OBJ_ARROW_THUMB_UP,0,Time[i],Ups[
i]);
            if(created)// もし作成されたなら、調整する
              {
               //--- アンカーポイントは、バーに重ならないように下にする
               ObjectSetInteger(0,string(Time[i]),OBJPROP_ANCHOR,ANCHOR_BOTTOM);
               //--- 最後の仕上げとして描画する
               ObjectSetInteger(0,string(Time[i]),OBJPROP_COLOR,clrBlue);
               upcounter++;
              }
           }
        }
      if(Downs[i]!=EMPTY_VALUE)// 下部のフラクタルを発見
        {
         if(downcounter<10)// 10 以上の下向き矢印は作成しない
           {
            //--- 「Down」オブジェクトの作成を試みる
            created=ObjectCreate(0,string(Time[i]),OBJ_ARROW_THUMB_DOWN,0,Time[i],Do
wns[i]);
            if(created)// もし作成されたなら、調整する
              {
               //--- アンカーポイントは、バーに重ならないように上にする
               ObjectSetInteger(0,string(Time[i]),OBJPROP_ANCHOR,ANCHOR_TOP);
               //--- 最後の仕上げとして描画する
               ObjectSetInteger(0,string(Time[i]),OBJPROP_COLOR,clrRed);
               downcounter++;
              }
           }
        }
     }
  }
```

スクリプトの実行後チャートはこの図のようになります。

© 2000-2025, MetaQuotes Ltd.



© 2000–2025, MetaQuotes Ltd.

オブジェクトが接続されているチャートのコーナー

グラフィックオブジェクトの系列があり、グラフィックオブジェクトの為に、それに対してピクセルで座標を表示するチャートの隅を指定することができます。これは次のオブジェクトタイプです。(括弧内にオブジェクトタイプの識別子が書かれています)

- Label (OBJ_LABEL); テキストラベル
- Button (OBJ_BUTTON); ボタン
- Bitmap Label (OBJ_BITMAP_LABEL); グラフィックラベル
- Edit (OBJ_EDIT); 入力欄
- Rectangle Label (OBJ_RECTANGLE_LABEL). 長方形ラベル

オブジェクト	インディケータ	X/Y	Width/Height	Date/Price	OBJPROP_CORNER	OBJPROP_ANCHOR	OBJPROP_ANGLE
Text	OBJ_TEXT	–	–	はい	–	はい	はい
Label	OBJ_LABEL	はい	はい(読み取り専用)	–	**はい**	はい	はい
Button	OBJ_BUTTON	はい	はい	–	**はい**	–	–
Bitmap	OBJ_BITMAP	–	はい(読み取り専用)	はい	–	はい	–
Bitmap Label	OBJ_BITMAP_LABEL	はい	はい(読み取り専用)	–	**はい**	はい	–
Edit	OBJ_EDIT	はい	はい	–	**はい**	–	–
Rectangle Label	OBJ_RECTANGLE_LABEL	はい	はい	–	**はい**	–	–

表には次の名称を使用しています。

- **X/Y**—チャートの隅の1つに対してピクセル単位でアンカーポイントが指定されます。
- **Width/Height**—オブジェクトの幅と高さ。『読み取り専用』が指定されている場合、チャート上にオブジェクトが描画された後にのみ、幅と高さの値が計算されることを意味しています。
- **Date/Price**—アンカーポイントが日付と価格のペアで指定されます。
- **OBJPROP_CORNER**—チャートの隅を指定し、それに対してアンカーポイントが指定されます。ENUM_BASE_CORNERリストの4つの値のうちの1つになる場合があります。
- **OBJPROP_ANCHOR**—オブジェクトでのアンカーポイントの位置を指定します。ENUM_ANCHOR_POINTリストの9つの値のうちの1つになる場合があります。このポイントから選択したチャートの隅まで、ピクセル単位で座標が表示されます。
- **OBJPROP_ANGLE**—反時計回りにオブジェクトの回転角度を指定します。

© 2000–2025, MetaQuotes Ltd.

X とY 座標がピクセル単位で測定されるチャートのコーナーを指定するには ObjectSetInteger(chartID, name, OBJPROP_CORNER、**chart_corner**) を使用します。

- chartID – チャート識別子
- name – グラフィックオブジェクトの名称
- OBJPROP_CORNER – 0 座標の隅を指定するプロパティ識別子
- **chart_corner** – 希望するチャートのコーナーで ENUM_BASE_CORNER 列挙体の値のいずれか

ENUM_BASE_CORNER

ID	説明
CORNER_LEFT_UPPER	座標の中心がチャートの右上隅。
CORNER_LEFT_LOWER	座標の中心がチャートの左下隅。
CORNER_RIGHT_LOWER	座標の中心がチャートの右下隅。
CORNER_RIGHT_UPPER	座標の中心がチャートの右上隅。

例:

```
void CreateLabel(long   chart_id,
                 string name,
                 int    chart_corner,
                 int    anchor_point,
                 string text_label,
                 int    x_ord,
                 int    y_ord)
  {
//---
   if(ObjectCreate(chart_id,name,OBJ_LABEL,0,0,0))
     {
      ObjectSetInteger(chart_id,name,OBJPROP_CORNER,chart_corner);
      ObjectSetInteger(chart_id,name,OBJPROP_ANCHOR,anchor_point);
      ObjectSetInteger(chart_id,name,OBJPROP_XDISTANCE,x_ord);
      ObjectSetInteger(chart_id,name,OBJPROP_YDISTANCE,y_ord);
      ObjectSetString(chart_id,name,OBJPROP_TEXT,text_label);
     }
   else
      Print("オブジェクトOBJ_LABELの作成に失敗しました",name,", エラーコード = ", GetLastError());
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int height=(int)ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0);
   int width=(int)ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0);
   string arrows[4]={"LEFT_UPPER","RIGHT_UPPER","RIGHT_LOWER","LEFT_LOWER"};
```

© 2000–2025, MetaQuotes Ltd.

```
   CreateLabel(0,arrows[0],CORNER_LEFT_UPPER,ANCHOR_LEFT_UPPER,arrows[0],50,50);
   CreateLabel(0,arrows[1],CORNER_RIGHT_UPPER,ANCHOR_RIGHT_UPPER,arrows[1],50,50);
   CreateLabel(0,arrows[2],CORNER_RIGHT_LOWER,ANCHOR_RIGHT_LOWER,arrows[2],50,50);
   CreateLabel(0,arrows[3],CORNER_LEFT_LOWER,ANCHOR_LEFT_LOWER,arrows[3],50,50);
  }
```

© 2000–2025, MetaQuotes Ltd.

オブジェクトの可視性

オブジェクトの可視性フラグの組み合わせは、オブジェクトが表示されるチャートの時間軸を決定します。OBJPROP_TIMEFRAMES プロパティ値を設定または取得するには ObjectSetInteger()/ObjectGetInteger() 関数を使用します。

ID	値	説明
OBJ_NO_PERIODS	0	オブジェクトは全ての時間軸には描画されていません。
OBJ_PERIOD_M1	0x00000001	オブジェクトは 1 分チャートに描画されています。
OBJ_PERIOD_M2	0x00000002	オブジェクトは 2 分チャートに描画されています。
OBJ_PERIOD_M3	0x00000004	オブジェクトは 3 分チャートに描画されています。
OBJ_PERIOD_M4	0x00000008	オブジェクトは 4 分チャートに描画されています。
OBJ_PERIOD_M5	0x00000010	オブジェクトは 5 分チャートに描画されています。
OBJ_PERIOD_M6	0x00000020	オブジェクトは 6 分チャートに描画されています。
OBJ_PERIOD_M10	0x00000040	オブジェクトは 10 分チャートに描画されています。
OBJ_PERIOD_M12	0x00000080	オブジェクトは 12 分チャートに描画されています。
OBJ_PERIOD_M15	0x00000100	オブジェクトは 15 分チャートに描画されています。
OBJ_PERIOD_M20	0x00000200	オブジェクトは 20 分チャートに描画されています。
OBJ_PERIOD_M30	0x00000400	オブジェクトは 30 分チャートに描画されています。
OBJ_PERIOD_H1	0x00000800	オブジェクトは 1 時間チャートに描画されています。
OBJ_PERIOD_H2	0x00001000	オブジェクトは 2 時間チャートに描画されています。
OBJ_PERIOD_H3	0x00002000	オブジェクトは 3 時間チャートに描画されています。
OBJ_PERIOD_H4	0x00004000	オブジェクトは 4 時間チャートに描画されています。

© 2000–2025, MetaQuotes Ltd.

ID	値	説明
OBJ_PERIOD_H6	0x00008000	オブジェクトは 6 時間チャートに描画されています。
OBJ_PERIOD_H8	0x00010000	オブジェクトは 8 時間チャートに描画されています。
OBJ_PERIOD_H12	0x00020000	オブジェクトは 12 時間チャートに描画されています。
OBJ_PERIOD_D1	0x00040000	オブジェクトは 1 日チャートに描画されています。
OBJ_PERIOD_W1	0x00080000	オブジェクトは 1 週間チャートに描画されています。
OBJ_PERIOD_MN1	0x00100000	オブジェクトは 1 ヶ月チャートに描画されています。
OBJ_ALL_PERIODS	0x001fffff	オブジェクトは全ての時間軸に描画されています。

可視性フラグは「 | 」シンボルを使用して組み合わせることが出来ます。例えば OBJ_PERIOD_M10|OBJ_PERIOD_H4 はオブジェクトが 10 分と4 時間時間軸で表示されることを意味します。

例:

```
void OnStart()
  {
//---
   string highlevel="PreviousDayHigh";
   string lowlevel="PreviousDayLow";
   double prevHigh;            // 昨日の高値
   double prevLow;             // 昨日の安値
   double highs[],lows[];      // 高安値の配列

//--- 最終エラーをリセットする
   ResetLastError();
//--- 毎日の時間軸の最後の 2 つの高値を取得する
   int highsgot=CopyHigh(Symbol(),PERIOD_D1,0,2,highs);
   if(highsgot>0) // 複製が成功
     {
      Print("High prices for the last 2 days were obtained successfully");
      prevHigh=highs[0]; // 昨日の高値
      Print("prevHigh = ",prevHigh);
      if(ObjectFind(0,highlevel)<0) // highlevel オブジェクトが見つからない
        {
         ObjectCreate(0,highlevel,OBJ_HLINE,0,0,0); // 水平線オブジェクトを作成する
        }
      //--- highlevel 線の価格レベルの値を設定する
      ObjectSetDouble(0,highlevel,OBJPROP_PRICE,0,prevHigh);
      //--- 可視を PERIOD_M10 及び PERIOD_H4 に設定する
```

© 2000-2025, MetaQuotes Ltd.

```
      ObjectSetInteger(0,highlevel,OBJPROP_TIMEFRAMES,OBJ_PERIOD_M10|OBJ_PERIOD_H4);
     }
   else
     {
      Print("Could not get High prices over the past 2 days, Error =
",GetLastError());
     }

//--- 最終エラーをリセットする
   ResetLastError();
//--- 1 日時間軸の 2 日の安値を取得する
   int lowsgot=CopyLow(Symbol(),PERIOD_D1,0,2,lows);
   if(lowsgot>0) // 複製が成功
     {
      Print("Low prices for the last 2 days were obtained successfully");
      prevLow=lows[0]; // 昨日の安値
      Print("prevLow = ",prevLow);
      if(ObjectFind(0,lowlevel)<0) // lowlevel オブジェクトが見つからない
        {
         ObjectCreate(0,lowlevel,OBJ_HLINE,0,0,0); // 水平線オブジェクトを作成する
        }
      //--- lowlevel 線の価格レベルの値を設定する
      ObjectSetDouble(0,lowlevel,OBJPROP_PRICE,0,prevLow);
      //--- 可視を PERIOD_M10 及び PERIOD_H4 に設定する
      ObjectSetInteger(0,lowlevel,OBJPROP_TIMEFRAMES,OBJ_PERIOD_M10|OBJ_PERIOD_H4);
     }
   else Print("Could not get Low prices for the last 2 days, Error =
",GetLastError());

   ChartRedraw(0); // チャートの再描画を強制する
  }
```

参照

PeriodSeconds、期間、チャート時間軸、日付と時刻

© 2000-2025, MetaQuotes Ltd.

エリオット波動のレベル

エリオット波動は OBJ_ELLIOTWAVE 5 及び OBJ_ELLIOTWAVE 3 型のグラフィックオブジェクトで表されます。波の大きさ(波の指標法)を設定するには OBJPROP_DEGREE プロパティが使用され、ENUM_ELLIOT_WAVE_DEGREE 列挙体の値のいずれかを割り当てることが出来ます。

ENUM_ELLIOT_WAVE_DEGREE

ID	説明
ELLIOTT_GRAND_SUPERCYCLE	グランドスーパーサイクル。
ELLIOTT_SUPERCYCLE	スーパーサイクル。
ELLIOTT_CYCLE	サイクル。
ELLIOTT_PRIMARY	プライマリ。
ELLIOTT_INTERMEDIATE	インターメディエイト。
ELLIOTT_MINOR	マイナー。
ELLIOTT_MINUTE	ミニュート。
ELLIOTT_MINUETTE	ミヌエット。
ELLIOTT_SUBMINUETTE	サブミヌエット。

例:

```
   for(int i=0;i<ObjectsTotal(0);i++)
     {
      string currobj=ObjectName(0,i);
      if((ObjectGetInteger(0,currobj,OBJPROP_TYPE)==OBJ_ELLIOTWAVE3) ||
          ((ObjectGetInteger(0,currobj,OBJPROP_TYPE)==OBJ_ELLIOTWAVE5)))
        {
         //--- マーキングレベルを INTERMEDIATE に設定
         ObjectSetInteger(0,currobj,OBJPROP_DEGREE,ELLIOTT_INTERMEDIATE);
         //--- 波の頂部との間の線を表す
         ObjectSetInteger(0,currobj,OBJPROP_DRAWLINES,true);
         //--- 線の色を設定する
         ObjectSetInteger(0,currobj,OBJPROP_COLOR,clrBlue);
         //--- 線の幅を設定する
         ObjectSetInteger(0,currobj,OBJPROP_WIDTH,5);
         //--- 記述を設定する
         ObjectSetString(0,currobj,OBJPROP_TEXT,"test script");
        }
     }
```

© 2000-2025, MetaQuotes Ltd.

ギャンオブジェクト

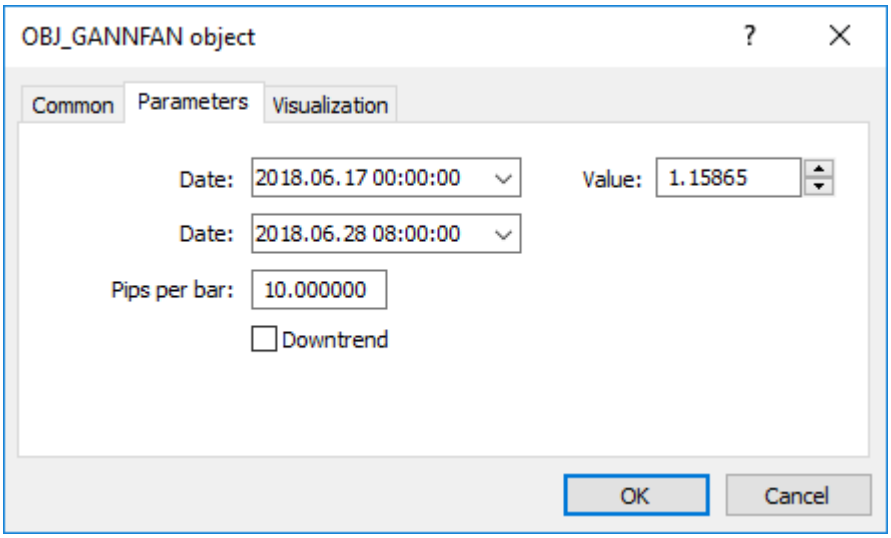
ギャンファン（OBJ_GANNFAN）とギャングリッド（OBJ_GANNGRID）オブジェクトでは、トレンドの方向を表すENUM_GANN_DIRECTION列挙を指定出来ます。

ENUM_GANN_DIRECTION

ID	説明
GANN_UP_TREND	上昇トレンドに対応する線。
GANN_DOWN_TREND	下落トレンドに対応する線。

主要線のスケールを 1x1 に設定するには、ObjectSetDouble(chart_handle, gann_object_name, OBJPROP_SCALE, scale) 関数が使用されます。

- chart_handle – オブジェクトが配置されているチャートウィンドウ
- gann_object_name – オブジェクト名
- OBJPROP_SCALE – 「スケール」プロパティの識別子
- scale – バーあたりのピップ数（Pips/Bar）単位の必要なスケール



ギャンファン作成の例

```
void OnStart()
  {
//---
   string my_gann="OBJ_GANNFAN object";
   if(ObjectFind(0,my_gann)<0)// オブジェクトが見つからない
     {
      //--- 失敗を通知する
      Print("Object ",my_gann," not found. Error code = ",GetLastError());
      //--- チャートの最高値を取得する
      double chart_max_price=ChartGetDouble(0,CHART_PRICE_MAX,0);
      //--- チャートの最安値を取得する
      double chart_min_price=ChartGetDouble(0,CHART_PRICE_MIN,0);
      //--- チャート上のバーの数
```

© 2000–2025, MetaQuotes Ltd.

```
      int bars_on_chart=ChartGetInteger(0,CHART_VISIBLE_BARS);
      //--- 各足のオープン時間を書き込むための配列を作成する
      datetime Time[];
      //--- 配列へのアクセスを時系列としてアレンジ
      ArraySetAsSeries(Time,true);
      //--- チャートで表示されているバーのデータをこの配列にコピー
      int times=CopyTime(NULL,0,0,bars_on_chart,Time);
      if(times<=0)
        {
         Print("Could not copy the array with the open time!");
         return;
        }
      //--- 予備の準備完了

      //--- チャートの中央のバーのインデックス
      int center_bar=bars_on_chart/2;
      //--- チャートの赤道 - 最大値と最小値の間
      double mean=(chart_max_price+chart_min_price)/2.0;
      //--- 最初のアンカーポイントの座標を中央に設定
      ObjectCreate(0,my_gann,OBJ_GANNFAN,0,Time[center_bar],mean,
                   //--- 2 番目のアンカーポイントを右に設定
                   Time[center_bar/2],(mean+chart_min_price)/2.0);
      Print("Time[center_bar] = "+(string)Time[center_bar]+"  Time[center_bar/2] =
"+(string)Time[center_bar/2]);
      //Print("Time[center_bar]/="+Time[center_bar]+"  Time[center_bar/2]
="+Time[center_bar/2]);
      //--- スケールを Pips / Bar 単位で設定
      ObjectSetDouble(0,my_gann,OBJPROP_SCALE,10);
      //--- 線のトレンドを設定
      ObjectSetInteger(0,my_gann,OBJPROP_DIRECTION,GANN_UP_TREND);
      //--- 線の幅を設定
      ObjectSetInteger(0,my_gann,OBJPROP_WIDTH,1);
      //--- 線のスタイルを定義
      ObjectSetInteger(0,my_gann,OBJPROP_STYLE,STYLE_DASHDOT);
      //--- 線の色を設定
      ObjectSetInteger(0,my_gann,OBJPROP_COLOR,clrYellowGreen);
      //--- ユーザのオブジェクト選択を許可
      ObjectSetInteger(0,my_gann,OBJPROP_SELECTABLE,true);
      //--- 自分で選ぶ
      ObjectSetInteger(0,my_gann,OBJPROP_SELECTED,true);
      //--- チャートに描画する
      ChartRedraw(0);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

ウェブカラー

下記のカラー定数は color 型で定義されています。

clrBlack	clrDarkGreen	clrDarkSlateGray	clrOlive	clrGreen	clrTeal	clrNavy	clrPurple
clrMaroon	clrIndigo	clrMidnightBlue	clrDarkBlue	clrDarkOliveGreen	clrSaddleBrown	clrForestGreen	clrOliveDrab
clrSeaGreen	clrDarkGoldenrod	clrDarkSlateBlue	clrSienna	clrMediumBlue	clrBrown	clrDarkTurquoise	clrDimGray
clrLightSeaGreen	clrDarkViolet	clrFireBrick	clrMediumVioletRed	clrMediumSeaGreen	clrChocolate	clrCrimson	clrSteelBlue
clrGoldenrod	clrMediumSpringGreen	clrLawnGreen	clrCadetBlue	clrDarkOrchid	clrYellowGreen	clrLimeGreen	clrOrangeRed
clrDarkOrange	clrOrange	clrGold	clrYellow	clrChartreuse	clrLime	clrSpringGreen	clrAqua
clrDeepSkyBlue	clrBlue	clrMagenta	clrRed	clrGray	clrSlateGray	clrPeru	clrBlueViolet
clrLightSlateGray	clrDeepPink	clrMediumTurquoise	clrDodgerBlue	clrTurquoise	clrRoyalBlue	clrSlateBlue	clrDarkKhaki
clrIndianRed	clrMediumOrchid	clrGreenYellow	clrMediumAquamarine	clrDarkSeaGreen	clrTomato	clrRosyBrown	clrOrchid
clrMediumPurple	clrPaleVioletRed	clrCoral	clrCornflowerBlue	clrDarkGray	clrSandyBrown	clrMediumSlateBlue	clrTan
clrDarkSalmon	clrBurlyWood	clrHotPink	clrSalmon	clrViolet	clrLightCoral	clrSkyBlue	clrLightSalmon
clrPlum	clrKhaki	clrLightGreen	clrAquamarine	clrSilver	clrLightSkyBlue	clrLightSteelBlue	clrLightBlue
clrPaleGreen	clrThistle	clrPowderBlue	clrPaleGoldenrod	clrPaleTurquoise	clrLightGray	clrWheat	clrNavajoWhite
clrMoccasin	clrLightPink	clrGainsboro	clrPeachPuff	clrPink	clrBisque	clrLightGoldenrod	clrBlanchedAlmond
clrLemonChiffon	clrBeige	clrAntiqueWhite	clrPapayaWhip	clrCornsilk	clrLightYellow	clrLightCyan	clrLinen
clrLavender	clrMistyRose	clrOldLace	clrWhiteSmoke	clrSeashell	clrIvory	clrHoneydew	clrAliceBlue
clrLavenderBlush	clrMintCream	clrSnow	clrWhite				

オブジェクトの色は ObjectSetInteger() 関数を使用して設定出来ます。カスタム指標の色を設定するには PlotIndexSetInteger() 関数が使用されます。カラー値の取得には ObjectGetInteger() 及び PlotIndexGetInteger() 関数があります。

© 2000–2025, MetaQuotes Ltd.

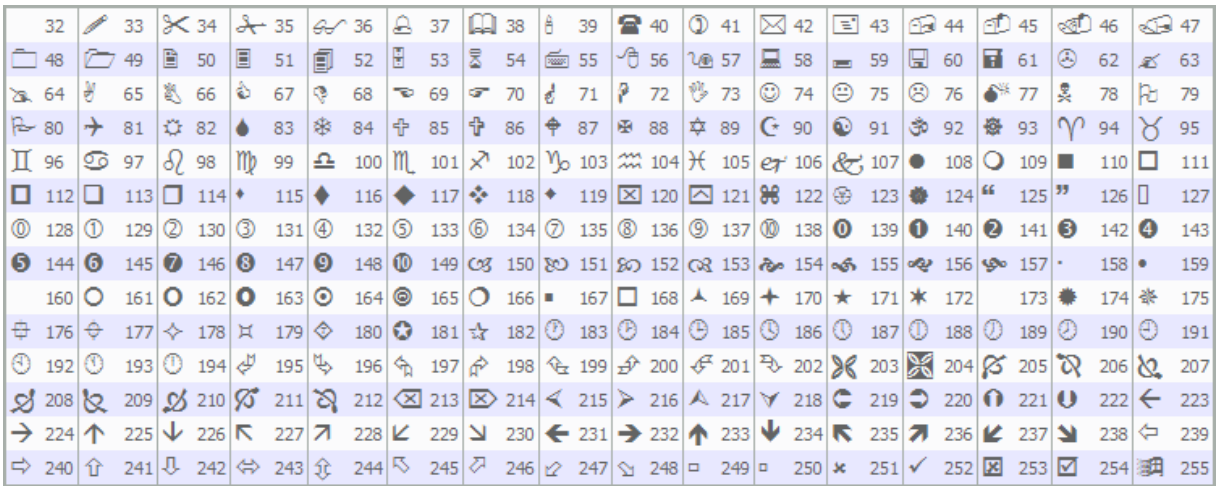
例：

```
//---- 指標の設定
#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   3
#property indicator_type1   DRAW_LINE
#property indicator_type2   DRAW_LINE
#property indicator_type3   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_color2  clrRed
#property indicator_color3  clrLime
```

© 2000–2025, MetaQuotes Ltd.

特殊文字

OBJ_ARROW オブジェクトと使用される特殊文字



必要な文字は ObjectSetInteger() 関数を使用して設定出来ます。

例:

```
void OnStart()
  {
//---
   string up_arrow="up_arrow";
   datetime time=TimeCurrent();
   double lastClose[1];
   int close=CopyClose(Symbol(),Period(),0,1,lastClose);      // 終値を取得
//--- 価格が取得された
   if(close>0)
     {
      ObjectCreate(0,up_arrow,OBJ_ARROW,0,0,0,0,0);           // 矢印を作成
      ObjectSetInteger(0,up_arrow,OBJPROP_ARROWCODE,241);     // 矢印のコードを作成
      ObjectSetInteger(0,up_arrow,OBJPROP_TIME,time);         // 時間を設定
      ObjectSetDouble(0,up_arrow,OBJPROP_PRICE,lastClose[0]);// 価格を設定
      ChartRedraw(0);                                         // 矢印を描画
     }
   else
      Print("Unable to get the latest Close price!");
  }
```

© 2000–2025, MetaQuotes Ltd.

指標定数

MQL5 言語で書かれたプログラムで使用出来る事前定義されたテクニカル指標は 37 あります。また iCustom() 関数を使用してのカスタム指標の作成も可能です。そのために必要な全ての定数は、5 つのグループに分かれています。

- 価格定数は指標が計算された価格またはボリュームの種類の選択に使用されます。
- 平滑化メソッドは指標で使用される内蔵された平滑化メソッドです。
- 指標線は CopyBuffer() で指標の値をアクセスする際に使用される指標バッファの識別子です。
- 描画スタイルは線の描画に使用される18種類の描画タイプです。
- カスタム指標プロパティはカスタム指標の操作のために関数で使用されます。
- 指標の種類は IndicatorCreate() を使用してハンドルを作成する際のテクニカル指標の種類を指定するために使用されます。
- データ型の識別子は MqlParam 型の配列で IndicatorCreate() 関数に受け渡すデータの型を指定するために使用されます。

© 2000–2025, MetaQuotes Ltd.

価格定数

テクニカル指標の計算には、価格値及び/または計算が実行されたボリュームの値が必要です。計算に使用する価格ベースを指定するための ENUM_APPLIED_PRICE 列挙には 7 つの事前定義された識別子があります。

ENUM_APPLIED_PRICE

ID	説明
PRICE_CLOSE	終値。
PRICE_OPEN	始値。
PRICE_HIGH	期間の高値。
PRICE_LOW	期間の安値。
PRICE_MEDIAN	中間価格(高 + 低)÷ 2。
PRICE_TYPICAL	典型的価格(高 + 低 + 終)÷ 3。
PRICE_WEIGHTED	平均価格(高 + 低 + 終 + 終)÷ 4。

ボリュームが計算に使用された場合、ENUM_APPLIED_VOLUME 列挙の 2 つの値のいずれかを指定する必要があります。

ENUM_APPLIED_VOLUME

ID	説明
VOLUME_TICK	ティックボリューム。
VOLUME_REAL	取引高。

iStochastic() テクニカル指標の計算法は 2 つあります。

- 終値のみ
- 高値と安値

計算に必要なバージョンを選択するには、ENUM_STO_PRICE 列挙体の値のいずれかを指定します。

ENUM_STO_PRICE

ID	説明
STO_LOWHIGH	計算は、低/高価格に基づいています。
STO_CLOSECLOSE	計算は、クローズ(ローソク足の形成が終わった時点での価格)/終値に基づいています。

テクニカル指標が ENUM_APPLIED_PRICE によって設定されている種類の価格データを計算に使用する場合、(端末機に内蔵あるいはユーザに書かれた)指標ハンドルが入力価格シリーズとして使用可能になります。この場合には、指標のゼロバッファの値が計算に使用されます。これは、他の指標の値を使用しての指標値の構築を容易にします。カスタム指標ハンドルは iCustom() 関数の呼び出しで作成されます。

例:

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   2
//--- 入力パラメータ
input int      RSIperiod=14;          // RSI を計算するための期間
input int      Smooth=8;              // 平滑化期間 RSI
input ENUM_MA_METHOD meth=MODE_SMMA; // 平滑化の方法
//---- RSI をプロットする
#property indicator_label1  "RSI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//---- RSI_Smoothed をプロットする
#property indicator_label2  "RSI_Smoothed"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrNavy
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- 指標バッファ
double         RSIBuffer[];           // RSI 値を収納
double         RSI_SmoothedBuffer[]; // スムーズにされたRSI 値
int            RSIhandle;             // RSI 指標ハンドル
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- 指標バッファマッピング
   SetIndexBuffer(0,RSIBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,RSI_SmoothedBuffer,INDICATOR_DATA);
   IndicatorSetString(INDICATOR_SHORTNAME,"iRSI");
   IndicatorSetInteger(INDICATOR_DIGITS,2);
//---
   RSIhandle=iRSI(NULL,0,RSIperiod,PRICE_CLOSE);
//---
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double &price[]
               )

  {
//--- 直近エラーの値をリセットする
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- RSI Buffer[] 配列でRSI の指標データを取得する
   int copied=CopyBuffer(RSIhandle,0,0,rates_total,RSIBuffer);
   if(copied<=0)
     {
      Print("Unable to copy the values of the indicator RSI. Error = ",
            GetLastError(),",  copied =",copied);
      return(0);
     }
//--- RSI  値を用いて平均値の指標を作成する
   int RSI_MA_handle=iMA(NULL,0,Smooth,0,meth,RSIhandle);
   copied=CopyBuffer(RSI_MA_handle,0,0,rates_total,RSI_SmoothedBuffer);
   if(copied<=0)
     {
      Print("Unable to copy the smoothed indicator of RSI. Error = ",
            GetLastError(),",  copied =",copied);
      return(0);
     }
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
```

© 2000–2025, MetaQuotes Ltd.

平滑化の方法

多くのテクニカル指標は、様々な価格シリーズ平滑化の方法に基づいています。いくつかの標準的なテクニカル指標は、入力パラメータとして平滑化の方法の指定を要します。使用される平滑化の方法を指定するためは ENUM_MA_METHOD 列挙体にリストされている識別子が使用されます。

ENUM_MA_METHOD

ID	説明
MODE_SMA	単純平均。
MODE_EMA	指数移動平均。
MODE_SMMA	平滑平均。
MODE_LWMA	線形加重移動平均。

例：

```
double ExtJaws[];
double ExtTeeth[];
double ExtLips[];
//---- 平均移動線を取り扱う
int    ExtJawsHandle;
int    ExtTeethHandle;
int    ExtLipsHandle;
//--- MA のハンドルを取得する
ExtJawsHandle=iMA(NULL,0,JawsPeriod,0,MODE_SMMA,PRICE_MEDIAN);
ExtTeethHandle=iMA(NULL,0,TeethPeriod,0,MODE_SMMA,PRICE_MEDIAN);
ExtLipsHandle=iMA(NULL,0,LipsPeriod,0,MODE_SMMA,PRICE_MEDIAN);
```

© 2000–2025, MetaQuotes Ltd.

指標線

テクニカル指標のいくつかはチャートに複数のバッファを描画します。指標バッファの番号付けは0から始まります。CopyBuffer() 関数を使用して指標値を double 型の配列に複製する時、指標によっては、番号の代わりに複製されたバッファの識別子を指定することが出来ます。

iMACD()、iRVI() 及び iStochastic() の値の複製時に使用出来る指標線の識別子

定数	値	説明
MAIN_LINE	0	正中線。
SIGNAL_LINE	1	シグナル線。

ADX() 及び ADXW() の値の複製時に使用出来る指標線の識別子

定数	値	説明
MAIN_LINE	0	正中線。
PLUSDI_LINE	1	線 +DI。
MINUSDI_LINE	2	線 -DI。

iBands() の値の複製時に使用出来る指標線の識別子

定数	値	説明
BASE_LINE	0	正中線。
UPPER_BAND	1	上限。
LOWER_BAND	2	下限。

iEnvelopes() 及び iFractals() の値の複製時に使用出来る指標線の識別子

定数	値	説明
UPPER_LINE	0	上部線。
LOWER_LINE	1	下部線。

iGator() の値の複製時に使用出来る指標線の識別子

定数	値	説明
UPPER_HISTOGRAM	0	上部ヒストグラム。
LOWER_HISTOGRAM	2	下部ヒストグラム。

iAlligator() の値の複製時に使用出来る指標線の識別子

定数	値	説明
GATORJAW_LINE	0	顎線。

© 2000-2025, MetaQuotes Ltd.

定数	値	説明
GATORTEETH_LINE	1	歯線。
GATORLIPS_LINE	2	口線。

iIchimoku() の値の複製時に使用出来る指標線の識別子

定数	値	説明
TENKANSEN_LINE	0	転換線。
KIJUNSEN_LINE	1	基準線。
SENKOUSPANA_LINE	2	先行スパンA線。
SENKOUSPANB_LINE	3	先行スパンB線。
CHIKOUSPAN_LINE	4	遅行スパンライン。

© 2000–2025, MetaQuotes Ltd.

描画スタイル

カスタム指標の作成時には、ENUM_DRAW_TYPE列挙体で指定されている(メインチャートウィンドウまたはグラフのサブウィンドウに表示される)18種類のグラフィックプロットのいずれかを指定することが出来ます。

1つのカスタム指標でどの指標作成・描画タイプを選ぶことも出来ます各構造タイプは、描画に必要なデータを格納するための1～5 グローバル配列 の仕様を必要とします。これらのデータ配列は、SetIndexBuffer() 関数を使用して指標バッファと結合する必要があります。各バッファにENUM_INDEXBUFFER_TYPE からのデータの種類を指定する必要があります。

描画スタイルに応じて、1～4 の値バッファが必要な場合があります(INDICATOR_DATAとしてマーク)。スタイルが動的な色の塗り替えを含む場合は(指標型 INDICATOR_COLOR_INDEX)もう1 つのバッファが必要になります(全てのスタイルの名称には COLOR が含まれいます)。色のバッファは、常にスタイルに対応するバッファの後に結合されています。

ENUM_DRAW_TYPE

ID	説明	データバッファ	色バッファ
DRAW_NONE	描かれません。	1	0
DRAW_LINE	線。	1	0
DRAW_SECTION	セクション。	1	0
DRAW_HISTOGRAM	ゼロラインからのヒストグラム。	1	0
DRAW_HISTOGRAM2	2 つの指標バッファのヒストグラム。	2	0
DRAW_ARROW	矢印の描画。	1	0
DRAW_ZIGZAG	ジグザグスタイルは、バーの垂直断面を可能にします。	2	0
DRAW_FILLING	2 レベル間の色の塗りつぶし。	2	0
DRAW_BARS	バーの列として表示。	4	0

© 2000-2025, MetaQuotes Ltd.

ID	説明	データバッファ	色バッファ
DRAW_CANDLES	ローソク足の列として表示。	4	0
DRAW_COLOR_LINE	多色のライン。	1	1
DRAW_COLOR_SECTION	多色のセクション。	1	1
DRAW_COLOR_HISTOGRAM	ゼロラインからの多色ヒストグラム。	1	1
DRAW_COLOR_HISTOGRAM2	2 つの指標バッファの多色ヒストグラム。	2	1
DRAW_COLOR_ARROW	多色の矢印を描画。	1	1
DRAW_COLOR_ZIGZAG	多色のジグザグ。	2	1
DRAW_COLOR_BARS	多色バー。	4	1
DRAW_COLOR_CANDLES	多色ローソク足。	4	1

選択された描画の型の表示を改良するために、ENUM_PLOT_PROPERTY に記載されている識別子が使用されます。

PlotIndexSetInteger() 及び PlotIndexGetInteger() 関数には

ENUM_PLOT_PROPERTY_INTEGER

ID	説明	プロパティ型
PLOT_ARROW	DRAW_ARROW スタイルの矢印コード。	uchar
PLOT_ARROW_SHIFT	DRAW_ARROW スタイルの矢印の垂直シフト。	int

© 2000–2025, MetaQuotes Ltd.

ID	説明	プロパティ型
PLOT_DRAW_BEGIN	データウィンドウでの描画前の初期のバーの数や値。	int
PLOT_DRAW_TYPE	グラフィック構成のタイプ。	ENUM_DRAW_TYPE
PLOT_SHOW_DATA	データウィンドウでの建設値の表示のサイン。	bool
PLOT_SHIFT	バーでの時間軸に沿っての指標のプロットシフト。	int
PLOT_LINE_STYLE	線の描画スタイル。	ENUM_LINE_STYLE
PLOT_LINE_WIDTH	描画線の太さ。	int
PLOT_COLOR_INDEXES	色の数。	int
PLOT_LINE_COLOR	描画の色を含むバッファのインデックス。	color modifier = index number of colors

PlotIndexSetDouble() 関数

ENUM_PLOT_PROPERTY_DOUBLE

ID	説明	プロパティ型
PLOT_EMPTY_VALUE	プロットに使用される空の値で、何も描画しません。	double

PlotIndexSetString() 関数

ENUM_PLOT_PROPERTY_STRING

ID	説明	プロパティ型
PLOT_LABEL	データウィンドウに表示する指標グラフィカルシリーズの名称。表示のためにいくつかの指標バッファを必要とする複雑なグラフィカルなスタイルで作業する場合、各バッファの名称は「;」セパレータを	string

© 2000-2025, MetaQuotes Ltd.

ID	説明	プロパティ型
	使用して指定することが出来ます。サンプルコードは DRAW_CANDLES にあります。	

カスタム指標の線を描画するために 5 つのスタイルを使用することが出来ます。これらは線の太さが 0 または 1 の時のみ有効です。

ENUM_LINE_STYLE

ID	説明
STYLE_SOLID	実線。
STYLE_DASH	途切れ線。
STYLE_DOT	点線。
STYLE_DASHDOT	一点鎖線。
STYLE_DASHDOTDOT	二点鎖線。

線の描画スタイルと図面の種類を設定するには PlotIndexSetInteger() 関数が使用されます。フィボナッチの拡張機能では、レベルの厚さと描画スタイルは、ObjectSetInteger() 関数を使用して示すことが出来ます。

例:

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- 指標バッファ
double          MABuffer[];
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- 配列をインデックス0で指標バッファと結合する
   SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
//--- 線の描画を設定
   PlotIndexSetInteger(0,PLOT_DRAW_TYPE,DRAW_LINE);
//--- 線のスタイルを設定
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,STYLE_DOT);
//--- 線の色を設定
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrRed);
//--- 線の太さを設定
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,1);
//--- 線のラベルを設定
```

© 2000–2025, MetaQuotes Ltd.

```
   PlotIndexSetString(0,PLOT_LABEL,"Moving Average");
//---
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---
   for(int i=prev_calculated;i<rates_total;i++)
     {
      MABuffer[i]=close[i];
     }
//--- 次の呼び出しのためにprev_calculated の値を返す
   return(rates_total);
  }
```

© 2000–2025, MetaQuotes Ltd.

カスタム指標プロパティ

カスタム指標で使用出来る指標バッファの数に制限はありません。しかし SetIndexBuffer() 関数を使用して、指標バッファとして指定される配列のそれぞれに格納されるデータ型を指定する必要があります。値は ENUM_INDEXBUFFER_TYPE 列挙体の値のうちのいずれかです。

ENUM_INDEXBUFFER_TYPE

ID	説明
INDICATOR_DATA	描画するデータ。
INDICATOR_COLOR_INDEX	色。
INDICATOR_CALCULATIONS	中間計算のための補助バッファ

カスタム指標には表示に活用できる多数の設定があります。これらの設定は、IndicatorSetDouble()、IndicatorSetInteger() 及び IndicatorSetString() 関数を使用して対応する指標のプロパティに行われます。指標のプロパティの識別子は ENUM_CUSTOMIND_PROPERTY 列挙にリストされています。

ENUM_CUSTOMIND_PROPERTY_INTEGER

ID	説明	プロパティ型
INDICATOR_DIGITS	指標値描画の精度。	int
INDICATOR_HEIGHT	指標ウィンドウの固定された高さ（プリプロセッサコマンド #property	int

© 2000–2025, MetaQuotes Ltd.

ID	説明	プロパティ型
	indicator_height)。	
INDICATOR_LEVELS	指標ウィンドウのレベル数。	int
INDICATOR_LEVELCOLOR	レベルの線の色。	color modifier = level number
INDICATOR_LEVELSTYLE	レベルの線のスタイル。	ENUM_LINE_STYLE modifier = level number
INDICATOR_LEVELWIDTH	レベルの線の太さ。	int 修飾子 = レベル番号
INDICATOR_FIXED_MINIMUM	indicator windowの固定最小値を。このプロパ	bool

© 2000–2025, MetaQuotes Ltd.

ID	説明	プロパティ型
	ティは。IndicatorSetInteger() 関数でのみ書き入れることができます	
INDICATOR_FIXED_MAXIMUM	indicator window の固定最小値。このプロパティは。IndicatorSetInteger() 関数でのみ書き入れることがで	bool

© 2000–2025, MetaQuotes Ltd.

ID	説明	プロパティ型
	きます	

ENUM_CUSTOMIND_PROPERTY_DOUBLE

ID	説明	プロパティ型
INDICATOR_MINIMUM	指標ウィンドウの最小サイズ。	double
INDICATOR_MAXIMUM	指標ウィンドウの最大サイズ。	double
INDICATOR_LEVELVALUE	レベル値。	double 修飾子 = レベル番号

ENUM_CUSTOMIND_PROPERTY_STRING

ID	説明	プロパティ型
INDICATOR_SHORTNAME	指標短縮名。	string
INDICATOR_LEVELTEXT	レベルの説明。	string modifier = level number

例：

```
//--- 指標の設定
```

© 2000–2025, MetaQuotes Ltd.

```
#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   2
#property indicator_type1   DRAW_LINE
#property indicator_type2   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_color2  clrRed
//--- 入力パラメータ
extern int KPeriod=5;
extern int DPeriod=3;
extern int Slowing=3;
//--- 指標バッファ
double MainBuffer[];
double SignalBuffer[];
double HighesBuffer[];
double LowesBuffer[];
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- 指標バッファマッピング
   SetIndexBuffer(0,MainBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,HighesBuffer,INDICATOR_CALCULATIONS);
   SetIndexBuffer(3,LowesBuffer,INDICATOR_CALCULATIONS);
//--- 精度を設定
   IndicatorSetInteger(INDICATOR_DIGITS,2);
//--- レベルを設定
   IndicatorSetInteger(INDICATOR_LEVELS,2);
   IndicatorSetDouble(INDICATOR_LEVELVALUE,0,20);
   IndicatorSetDouble(INDICATOR_LEVELVALUE,1,80);
//--- サブウィンドウの最大値と最小値を設定する
   IndicatorSetDouble(INDICATOR_MINIMUM,0);
   IndicatorSetDouble(INDICATOR_MAXIMUM,100);
//--- インデックスが描画される最初のバーを設定する
   PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,KPeriod+Slowing-2);
   PlotIndexSetInteger(1,PLOT_DRAW_BEGIN,KPeriod+Slowing+DPeriod);
//--- 2 番目の線に STYLE_DOT スタイルを設定する
   PlotIndexSetInteger(1,PLOT_LINE_STYLE,STYLE_DOT);
//--- データウィンドウ の名称と指標サブウィンドウラベル
   IndicatorSetString
(INDICATOR_SHORTNAME,"Stoch("+KPeriod+","+DPeriod+","+Slowing+")");
   PlotIndexSetString(0,PLOT_LABEL,"Main");
   PlotIndexSetString(1,PLOT_LABEL,"Signal");
//--- 線の描画を空の値に設定する
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0.0);
   PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,0.0);
//--- 初期化完了
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

© 2000–2025, MetaQuotes Ltd.

テクニカル指標の種類

値にアクセスするために指標ハンドルを作成する方法は 2 つあります。最初の方法は、テクニカル指標のリストから直接関数名を指定することです。2 番目の方法は IndicatorCreate() を使用して ENUM_INDICATOR 列挙から識別子を割り当てて、任意の指標ハンドルを作成することです。ハンドル作成の両方法は同等なので、MQL5 でプログラムを書く時に、特定の場合に最も便利なものを使用することが出来ます。

IND_CUSTOM 型の指標の作成時には 入力パラメータ MqlParam の最初の要素の *type* フィールドは ENUM_DATATYPE 列挙の TYPE_STRING 値を持たなければならず、最初の要素の *string_value* フィールドはカスタム指標の名称を含まなければなりません。

ENUM_INDICATOR

識別子	指標
IND_AC	ACオシレーター。
IND_AD	蓄積/配信（Accumulation/Distribution）。
IND_ADX	平均方向性指数（Average Directional Index）。
IND_ADXW	ウェルズ·ワイルダーのADX。
IND_ALLIGATOR	アリゲーター。
IND_AMA	適応型移動平均（Adaptive Moving Average）。
IND_AO	オーサムオシレーター。
IND_ATR	ATR（Average True Range）。
IND_BANDS	ボリンジャーバンド®（Bollinger Bands®）。
IND_BEARS	ベアパワー（Bears Power）。
IND_BULLS	ブルパワー（Bulls Power）。
IND_BWMFI	マーケットファシリテーションインデックス。
IND_CCI	コモディティチャンネルインデックス（Commodity Channel Index）。
IND_CHAIKIN	チャイキンオシレーター（Chaikin Oscillator）。
IND_CUSTOM	カスタム指標。
IND_DEMA	2 重指数移動平均（Double Exponential Moving Average）。
IND_DEMARKER	デマーカー（DeMarker）。
IND_ENVELOPES	エンベロープ（Envelopes）。
IND_FORCE	勢力指数（Force Index）。
IND_FRACTALS	フラクタル。
IND_FRAMA	フラクタル適応型移動平均（Fractal Adaptive Moving Average）。
IND_GATOR	ゲーターオシレーター。

© 2000–2025, MetaQuotes Ltd.

識別子	指標
IND_ICHIMOKU	一目均衡表（Ichimoku Kinko Hyo）。
IND_MA	移動平均（Moving Average）。
IND_MACD	MACD
IND_MFI	マネーフローインデックス（Money Flow Index）。
IND_MOMENTUM	モメンタム（Momentum）。
IND_OBV	オンバランスボリューム（On Balance Volume）。
IND_OSMA	OsMA
IND_RSI	相対力指数（Relative Strength Index）。
IND_RVI	相対活力指数（Relative Vigor Index）。
IND_SAR	パラボリック SAR。
IND_STDDEV	標準偏差（Standard Deviation）。
IND_STOCHASTIC	ストキャスティックス（Stochastic Oscillator）。
IND_TEMA	3 重指数移動平均（Triple Exponential Moving Average）。
IND_TRIX	3 重指数移動平均オシレーター（Triple Exponential Moving Averages Oscillator）。
IND_VIDYA	可変インデックス動的平均（Variable Index Dynamic Average）。
IND_VOLUMES	ボリューム。
IND_WPR	ウィリアムパーセントレンジ（Williams' Percent Range）。

© 2000–2025, MetaQuotes Ltd.

データ型識別子

IndicatorCreate() 関数を使用して指標ハンドルを作成する際には、最終パラメータとして MqlParam 型の配列が指定されなければなりません。従って、指標を記述する MqlParam 構造体は特別な *type* フィールドを含みます。このフィールドは配列の特定の要素によって渡されるデータ型（real、integer または string 型）の情報を含みます。この MqlParam 構造体のフィールドの値は ENUM_DATATYPE の値の 1 つです。

ENUM_DATATYPE

識別子	データ型
TYPE_BOOL	bool
TYPE_CHAR	char
TYPE_UCHAR	uchar
TYPE_SHORT	short
TYPE_USHORT	ushort
TYPE_COLOR	color
TYPE_INT	int
TYPE_UINT	uint
TYPE_DATETIME	datetime
TYPE_LONG	long
TYPE_ULONG	ulong
TYPE_FLOAT	float
TYPE_DOUBLE	double
TYPE_STRING	string

配列の各要素は、作成されたテクニカル指標の対応する入力パラメータを記述するので、配列内の要素の型と順序は厳密に記述に従って維持されなければなりません。

© 2000–2025, MetaQuotes Ltd.

環境状態

MQL5 プログラムの現在の実行環境を記述する定数は下記のグループに分けられます。

- クライアント端末プロパティ – クライアント端末の情報
- 実行中の MQL5 プログラムのプロパティ – 実行を制御するのに役立つMQL5 プログラムのプロパティ
- シンボルプロパティ – シンボルの情報取得
- 口座プロパティ – 現在の口座の情報
- テスト統計 – エキスパートアドバイザーのテスト結果

© 2000–2025, MetaQuotes Ltd.

クライアント端末プロパティ

クライアント端末の情報は TerminalInfoInteger() 及び TerminalInfoString() の 2 つの関数で取得出来ます。パラメータにはそれぞれ ENUM_TERMINAL_INFO_INTEGER 及び ENUM_TERMINAL_INFO_STRING の値が使用されます。

ENUM_TERMINAL_INFO_INTEGER

識別子	説明	型
TERMINAL_BUILD	クライアント端末ビルド番号。	int
TERMINAL_COMMUNITY_ACCOUNT	フラグは、端末内の MQL5.community 認証データの存在を示します。	bool
TERMINAL_COMMUNITY_CONNECTION	MQL5.community への接続。	bool
TERMINAL_CONNECTED	取引サーバへの接続。	bool
TERMINAL_DLLS_ALLOWED	DLLを使用する許可。	bool
TERMINAL_TRADE_ALLOWED	約定許可。	bool
TERMINAL_EMAIL_ENABLED	端末の設定で指定された SMTP サーバへの電子メールの送信及びログインの許可。	bool
TERMINAL_FTP_ENABLED	端末の設定で指定された FTP サーバへのレポート送信及びログインの許可。	bool
TERMINAL_NOTIFICATIONS_ENABLED	スマートフォンに通知を送信するためのアクセス許可。	bool
TERMINAL_MAXBARS	チャートのバーの最大数。	int
TERMINAL_MQID	このフラグはプッシュ通知の MetaQuotes ID データが存在することを示します。	bool
TERMINAL_CODEPAGE	クライアント端末にインストールされた言語のコードページ。	int
TERMINAL_CPU_CORES	システム CPU コアの数。	int
TERMINAL_DISK_SPACE	端末（エージェント）のMQL5¥ Files フォルダの空きディスク容量（MB 単位）。	int
TERMINAL_MEMORY_PHYSICAL	システムの物理メモリ（MB 単位）。	int
TERMINAL_MEMORY_TOTAL	端末のプロセス（エージェント）に使用可能なメモリ（MB 単位）。	int
TERMINAL_MEMORY_AVAILABLE	端末（エージェント）プロセスの空きメモリ（MB 単位）。	int

© 2000–2025, MetaQuotes Ltd.

識別子	説明	型
TERMINAL_MEMORY_USED	端末（エージェント）で使用されるメモリ（MB単位）。	int
TERMINAL_X64	「64ビット端末」の表示。	bool
TERMINAL_OPENCL_SUPPORT	0x00010002 = 1.2 形式でのサポートされている OpenCL のバージョン。「0」は OpenCL をサポートがないことを意味します。	int
TERMINAL_SCREEN_DPI	画像の情報出力解像度は、DPIで測定されます。 このパラメータを知ることで、異なる解像度のモニターで同じように表示されるように、グラフィックオブジェクトのサイズを指定することができます。	int
TERMINAL_SCREEN_LEFT	仮想画面の左座標です。仮想画面はすべてのモニターをカバーする四角形です。システムに右から左に並べた2つのモニターがある場合、仮想スクリーンの左の座標は2つのモニターの境界になります。	int
TERMINAL_SCREEN_TOP	仮想画面の上部座標です。	int
TERMINAL_SCREEN_WIDTH	端末幅	int
TERMINAL_SCREEN_HEIGHT	端末の高さ	int
TERMINAL_LEFT	仮想画面に相対した端末の左座標です。	int
TERMINAL_TOP	仮想画面に相対した端末の上部座標です。	int
TERMINAL_RIGHT	仮想画面に相対した端末の右座標です。	int
TERMINAL_BOTTOM	仮想画面に相対した端末の下部座標です。	int
TERMINAL_PING_LAST	取引サーバまでの既知のping値は、マイクロ秒単位です。1秒は100万マイクロ秒です	int
TERMINAL_VPS	Indication that the terminal is launched on the MetaTrader Virtual Hosting server (MetaTrader VPS)	bool
キー識別子	説明	
TERMINAL_KEYSTATE_LEFT	「左矢印」キーの状態	int
TERMINAL_KEYSTATE_UP	「上矢印」キーの状態	int

© 2000–2025, MetaQuotes Ltd.

識別子	説明	型
TERMINAL_KEYSTATE_RIGHT	「右矢印」キーの状態	int
TERMINAL_KEYSTATE_DOWN	「下矢印」キーの状態	int
TERMINAL_KEYSTATE_SHIFT	「Shift」キーの状態	int
TERMINAL_KEYSTATE_CONTROL	「Ctrl」キーの状態	int
TERMINAL_KEYSTATE_MENU	「Windows」キーの状態	int
TERMINAL_KEYSTATE_CAPSLOCK	「CapsLock」キーの状態	int
TERMINAL_KEYSTATE_NUMLOCK	「NumLock」キーの状態	int
TERMINAL_KEYSTATE_SCRLOCK	「ScrollLock」キーの状態	int
TERMINAL_KEYSTATE_ENTER	「Enter」キーの状態	int
TERMINAL_KEYSTATE_INSERT	「Insert」キーの状態	int
TERMINAL_KEYSTATE_DELETE	「Delete」キーの状態	int
TERMINAL_KEYSTATE_HOME	「Home」キーの状態	int
TERMINAL_KEYSTATE_END	「End」キーの状態	int
TERMINAL_KEYSTATE_TAB	「Tab」キーの状態	int
TERMINAL_KEYSTATE_PAGEUP	「PageUp」キーの状態	int
TERMINAL_KEYSTATE_PAGEDOWN	「PageDown」キーの状態	int
TERMINAL_KEYSTATE_ESCAPE	「Escape」キーの状態	int

TerminalInfoInteger(TERMINAL_KEYSTATE_XXX)の呼び出しは、GetKeyState()関数と同じキーの状態を示すコードを返します。

例 スケーリング係数を計算します:

```
//--- 画面に1.5インチの幅のボタンを作成します
int screen_dpi = TerminalInfoInteger(TERMINAL_SCREEN_DPI); // ユーザーのモニターのDPIを取得します
int base_width = 144;                                      // DPI=96の標準モニターの画面のドットの基本の幅
int width      = (button_width * screen_dpi) / 96;         // ユーザーモニター(DPIを含む)のボタンの幅を計算します
...

//--- 百分率でのスケーリング係数の計算
int scale_factor=(TerminalInfoInteger(TERMINAL_SCREEN_DPI) * 100) / 96;
//--- スケーリング係数の計算
width=(base_width * scale_factor) / 100;
```

© 2000-2025, MetaQuotes Ltd.

こういった場合、グラフィックリソースは、異なる解像度のモニター上で見た目が同じサイズになります。その際、制御要素のサイズ(ボタン、ダイアログボックスなど)は個々の設定に相応します。

ENUM_TERMINAL_INFO_DOUBLE

識別子	説明	型
TERMINAL_COMMUNITY_BALANCE	MQL5.community の残高。	double
TERMINAL_RETRANSMISSION	指定されたコンピュータ上のすべての実行中のアプリケーションおよびサービスのTCP/IPプロトコルにおけるネットワークパケットの再再送信率。パケットロスは、最速かつ正しく設定されたネットワークでも発生し、受信者と送信者との間のパケット配信の確認がないため、失われたパケットは再送信されます。 これは特定の端末と取引サーバ間の接続品質を示すものではありません。これは、システムおよびバックグラウンド・アクティビティを含むネットワーク・アクティビティ全体でパーセンテージが計算されるためです。 TERMINAL_RETRANSMISSION値は、オペレーティングシステムから1分に1回要求されます。ターミナル自体はこの値を計算しません。	double

ファイル操作は 2 つのディレクトリでのみ実行でき、パスは TERMINAL_DATA_PATH 及び TERMINAL_COMMONDATA_PATH プロパティをリクエストして見つけられます。

ENUM_TERMINAL_INFO_STRING

識別子	説明	型
TERMINAL_LANGUAGE	端末の言語。	string
TERMINAL_COMPANY	会社名。	string
TERMINAL_NAME	端末名。	string
TERMINAL_PATH	端末が起動されるフォルダ。	string
TERMINAL_DATA_PATH	端末データが格納されるフォルダ。	string
TERMINAL_COMMONDATA_PATH	コンピュータにインストールされている全ての端末の共通パス。	string

© 2000-2025, MetaQuotes Ltd.

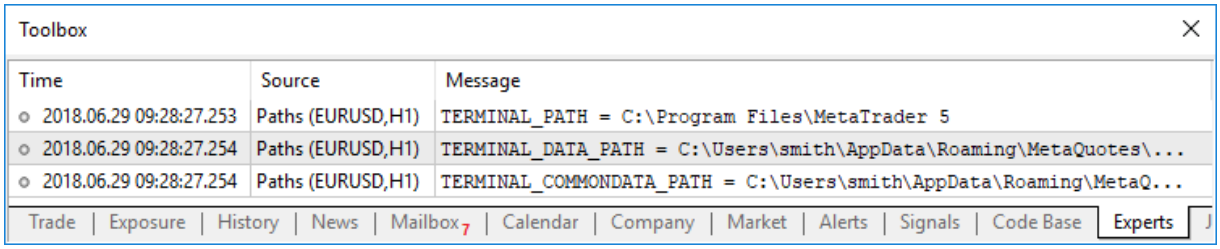
識別子	説明	型
TERMINAL_CPU_NAME	CPU名	string
TERMINAL_CPU_ARCHITECTURE	CPUアーキテクチャ	string
TERMINAL_OS_VERSION	ユーザーのOS名	string

TERMINAL_PATH、TERMINAL_DATA_PATH 及び TERMINAL_COMMONDATA_PATH パラメータに格納されているパスをより良く理解するために、お使いのコンピュータにインストールされた、クライアント端末の現在のコピーに属する値を返すスクリプトを実行することをお勧めします。

この例のスクリプトはクライアント端末のパス情報を返します。

```
//+------------------------------------------------------------------+
//|                                          Check_TerminalPaths.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   Print("TERMINAL_PATH = ",TerminalInfoString(TERMINAL_PATH));
   Print("TERMINAL_DATA_PATH = ",TerminalInfoString(TERMINAL_DATA_PATH));
   Print("TERMINAL_COMMONDATA_PATH =
",TerminalInfoString(TERMINAL_COMMONDATA_PATH));
  }
```

スクリプトの実行の結果、エキスパート操作ログに次のようなメッセージが表示されます。



© 2000–2025, MetaQuotes Ltd.

MQL5 プログラム実行プロパティ

実行中の MQL5 プログラムの情報取得には ENUM_MQL_INFO_INTEGER 及び ENUM_MQL_INFO_STRING からの定数が使用されます。

MQLInfoInteger 関数

ENUM_MQL_INFO_INTEGER

識別子	説明	Type
MQL_HANDLES_USED	現在アクティブなオブジェクトハンドルの数。動的(new で作成)オブジェクトと非動的オブジェクトの両方、グローバル/ローカル変数、またはクラスメンバーが含まれます。プログラムが使用するハンドルが多いほど、プログラムが消費するリソースも多くなります。	int
MQL_MEMORY_LIMIT	MQL5 プログラムの最大動的メモリ量(MB 単位)。	int
MQL_MEMORY_USED	MQL5 プログラムのメモリ使用量(MB 単位)。	int
MQL_PROGRAM_TYPE	MQL5 プログラムの種類。	ENUM_PROGRAM_TYPE
MQL_DLLS_ALLOWED	指定された実行プログラムで DLL を使用する許可。	bool
MQL_TRADE_ALLOWED	指定された実行プログラムでの取引許可。	bool
MQL_SIGNALS_ALLOWED	指定された実行プログラムでシグナルを変更する許可。	bool
MQL_DEBUG	デバグモードを示すフラグ。	bool

© 2000–2025, MetaQuotes Ltd.

識別子	説明	Type
MQL_PROFILER	プログラムがコードプロファイリングモードで動作することを示すフラグ。	bool
MQL_TESTER	テスタプロセスを示すフラグ。	bool
MQL_FORWARD	プログラムがフォワードテストプロセスで実行されていることを示すフラグ	bool
MQL_OPTIMIZATION	最適化プロセスを示すフラグ。	bool
MQL_VISUAL_MODE	ビジュアルテスタプロセスを示すフラグ。	bool
MQL_FRAME_MODE	エキスパートアドバイザーが最適化結果のフレーム収集モードで動作されるというフラグ。	bool
MQL_LICENSE_TYPE	EX5 モジュールのライセンスの種類。ライセンスは MQLInfoInteger（MQL_LICENSE_TYPEを使用してリクエストされた EX5 モジュールを指します。	ENUM_LICENSE_TYPE
MQL_STARTED_FROM_CONFIG	スクリプト/EAが設定ファイルの StartUpセクションから起動された場合、trueを返します。これは、このスクリプト/EAがターミナルの起動に使用された設定ファイルに登録されたことを意味します。	bool

MQLInfoString 関数

ENUM_MQL_INFO_STRING

© 2000–2025, MetaQuotes Ltd.

識別子	説明	Type
MQL_PROGRAM_NAME	実行される MQL5 プログラムの名称。	string
MQL5_PROGRAM_PATH	指定されたプログラムのパス。	string

実行中のプログラムの情報には、ENUM_PROGRAM_TYPE の値が使用されます。

ENUM_PROGRAM_TYPE

識別子	説明
PROGRAM_SCRIPT	スクリプト。
PROGRAM_EXPERT	エキスパート。
PROGRAM_INDICATOR	指標。
PROGRAM_SERVICE	サービス。

ENUM_LICENSE_TYPE

識別子	説明
LICENSE_FREE	無制限の無料版。
LICENSE_DEMO	マーケットから購入した製品の試用版。ストラテジーテスター内のみで使用出来ます。
LICENSE_FULL	購入したライセンス版では、少なくとも 5 アクティベーションが可能です。アクティベーションの数は、売主によって指定されます。売主は、アクティベーションの許容数を増加させることが出来ます。
LICENSE_TIME	期間限定ライセンス版。

例:

```
   ENUM_PROGRAM_TYPE mql_program=(ENUM_PROGRAM_TYPE)
MQLInfoInteger(MQL_PROGRAM_TYPE);
   switch(mql_program)
     {
      case PROGRAM_SCRIPT:
        {
         Print(__FILE__+" is script");
         break;
        }
      case PROGRAM_EXPERT:
        {
```

© 2000–2025, MetaQuotes Ltd.

```
         Print(__FILE__+" is Expert Advisor");
         break;
        }
      case PROGRAM_INDICATOR:
        {
         Print(__FILE__+" is custom indicator");
         break;
        }
      default:Print("MQL5 type value is ",mql_program);
   //---
   Print("MQLInfoInteger(MQL_MEMORY_LIMIT)=", MQLInfoInteger(MQL_MEMORY_LIMIT), "
MB");
   Print("MQLInfoInteger(MQL_MEMORY_USED)=", MQLInfoInteger(MQL_MEMORY_USED), "
MB");
   Print("MQLInfoInteger(MQL_HANDLES_USED)=", MQLInfoInteger(MQL_HANDLES_USED), "
handles");
```

© 2000–2025, MetaQuotes Ltd.

シンボルプロパティ

現在の銘柄に関する情報の取得には SymbolInfoInteger()、SymbolInfoDouble() 及び SymbolInfoString() の複数の関数を使用できます。1 番目のパラメータは銘柄名で、2 番目のパラメータの値には ENUM_SYMBOL_INFO_INTEGER、ENUM_SYMBOL_INFO_DOUBLE 及び ENUM_SYMBOL_INFO_STRING の識別子が使用出来ます。

SymbolInfoInteger() 関数には

ENUM_SYMBOL_INFO_INTEGER

識別子	説明	種類
SYMBOL_SECTOR	資産が属する経済部門	ENUM_SYMBOL_SECTOR
SYMBOL_INDUSTRY	銘柄が属する業界または経済部門	ENUM_SYMBOL_INDUSTRY
SYMBOL_CUSTOM	カスタムシンボルです。シンボルは板情報および/また	bool

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
	は外部データソースからの他のシンボルに基づいて総合的に作成されています。	
SYMBOL_BACKGROUND_COLOR	「気配値表示」のシンボルに使用されている背景色。	color
SYMBOL_CHART_MODE	シンボルバーを生成するために	ENUM_SYMBOL_CHART_MODE

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
	使用される価格の種類（BidまたはLast）。	
SYMBOL_EXIST	Symbol with this name exists	bool
SYMBOL_SELECT	「気配値表示」でシンボルが選択されています。	bool
SYMBOL_VISIBLE	シンボルは気配値表示で表示さ	bool

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
	れます。 一部のシンボル（主に、証拠金の計算や預金通貨の利益の計算に必要なクロスレート）は自動的に選択されますが、一般的に	

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
	気配値表示には表示されません。そのようなシンボルを表示するには、それらを明示的に選択する必要があります。	
SYMBOL_SESSION_DEALS	現在のセッションの約定数。	long
SYMBOL_SESSION_BUY_ORDERS	現時点	long

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
	での買い注文の数。	
SYMBOL_SESSION_SELL_ORDERS	現時点での売り注文の数。	long
SYMBOL_VOLUME	最終約定ボリューム。	long
SYMBOL_VOLUMEHIGH	一日の最大ボリューム。	long
SYMBOL_VOLUMELOW	一日の最小ボリューム。	long
SYMBOL_TIME	最終の	datetime

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
	相場の時刻。	
SYMBOL_TIME_MSC	最後の相場の1970.01.01から経過した時間（ミリ秒）	long
SYMBOL_DIGITS	小数点以下の桁数。	int
SYMBOL_SPREAD_FLOAT	変動スプレッドの表示。	bool
SYMBOL_SPREAD	ポイント単位でのスプレッド値。	int

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
SYMBOL_TICKS_BOOKDEPTH	板情報で表示されるリクエストの最高数。要求のないキューを持っていないシンボルの場合、値はゼロに等しいです。	int
SYMBOL_TRADE_CALC_MODE	契約価格計算モード。	ENUM_SYMBOL_CALC_MODE
SYMBOL_TRADE_MODE	注文実行の	ENUM_SYMBOL_TRADE_MODE

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
	種類。	
SYMBOL_START_TIME	シンボル取引開始の日（通常は先物取引に使用）	datetime
SYMBOL_EXPIRATION_TIME	シンボル取引終了の日（通常は先物取引に使用）。	datetime
SYMBOL_TRADE_STOPS_LEVEL	現在の終値から逆	int

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
	指値注文を配置する場合の最少のインデント（ポイント単位）。	
SYMBOL_TRADE_FREEZE_LEVEL	取引業務を凍結までの距離（ポイント単位）。	int
SYMBOL_TRADE_EXEMODE	約定実行モード。	ENUM_SYMBOL_TRADE_EXECUTION
SYMBOL_SWAP_MODE	スワップ計算モデル。	ENUM_SYMBOL_SWAP_MODE

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
SYMBOL_SWAP_ROLLOVER3DAYS	トリプルスワップが加算される曜日。	ENUM_DAY_OF_WEEK
SYMBOL_MARGIN_HEDGED_USE_LEG	より大きな脚を使用したヘッジ証拠金の計算（売りまたは買い）。	bool
SYMBOL_EXPIRATION_MODE	注文期限切れモードの可能なフラグ。	int
SYMBOL_FILLING_MODE	注文	int

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
	充填モードの可能なフラグ。	
SYMBOL_ORDER_MODE	注文の種類に使用可能なフラグ。	int
SYMBOL_ORDER_GTC_MODE	SYMBOL_EXPIRATION_MODE=SYMBOL_EXPIRATION_GTC（キャンセルされるまで有効）の場	ENUM_SYMBOL_ORDER_GTC_MODE

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
	合の、決済逆指値及び決済指値の期限</t2>。	
SYMBOL_OPTION_MODE	オプションの種類。	ENUM_SYMBOL_OPTION_MODE
SYMBOL_OPTION_RIGHT	オプション特権（買い/売り）。	ENUM_SYMBOL_OPTION_RIGHT

SymbolInfoDouble() 関数では

ENUM_SYMBOL_INFO_DOUBLE

識別子	説明	種類
SYMBOL_BID	売値 – ベストセルオファー。	double
SYMBOL_BIDHIGH	一日の最高売値。	double

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
SYMBOL_BIDLOW	一日の最低売値。	double
SYMBOL_ASK	買値 – ベストバイオファー。	double
SYMBOL_ASKHIGH	一日の最高買値。	double
SYMBOL_ASKLOW	一日の最低買値。	double
SYMBOL_LAST	直近約定値。	double
SYMBOL_LASTHIGH	一日の最高直近価格。	double
SYMBOL_LASTLOW	一日の最低直近価格。	double
SYMBOL_VOLUME_REAL	最終約定ボリューム。	double
SYMBOL_VOLUMEHIGH_REAL	一日の最大ボリューム。	double
SYMBOL_VOLUMELOW_REAL	一日の最小ボリューム。	double
SYMBOL_OPTION_STRIKE	オプションの行使価格。オプションの買い手が原資産を(コールオプ	double

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
	ションで）購入したり、（プットオプションで）販売することが出来、オプションの売り手が原資産の適切な量を売買する義務がある価格。	
SYMBOL_POINT	シンボルのポイント値。	double
SYMBOL_TRADE_TICK_VALUE	SYMBOL_TRADE_TICK_VALUE_PROFITの値。	double
SYMBOL_TRADE_TICK_VALUE_PROFIT	収益性の高いポジションの計算されたティック価格。	double
SYMBOL_TRADE_TICK_VALUE_LOSS	負けポジションの計算されたティック価格。	double
SYMBOL_TRADE_TICK_SIZE	最少の価格変化。	double

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
SYMBOL_TRADE_CONTRACT_SIZE	取引契約サイズ。	double
SYMBOL_TRADE_ACCRUED_INTEREST	未収利息。- 累積されたクーポンの利子、すなわち、クーポン債の発行または最後のクーポンの支払からの日数に比例して計算されたクーポンの利子の一部。	double
SYMBOL_TRADE_FACE_VALUE	Face value - 発行者が設定した初期債券価値。	double
SYMBOL_TRADE_LIQUIDITY_RATE	流動性率とは、証拠金に使用できる資産のシェアで。	double
SYMBOL_VOLUME_MIN	約定の最小ボリューム。	double
SYMBOL_VOLUME_MAX	約定の最大ボ	double

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
	リューム。	
SYMBOL_VOLUME_STEP	約定実行のための最小限のボリューム変化のステップ。	double
SYMBOL_VOLUME_LIMIT	シンボルの片方向（売または買）のオープンポジションと未決注文の最大許容総額。例えば、制限が5 ロットの場合、5 ロットのボリュームでオープン買いポジションを持つことができ、5 ロットのボリュームで、売り指値注文をすることが出来ます。しかし、この場合には、1 方向の	double

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
	総ボリュームが制限を超えてしまうため、買い指値注文をすることや 5 ロット以上の売り指値注文をすることは出来ません。	
SYMBOL_SWAP_LONG	長期スワップ値。	double
SYMBOL_SWAP_SHORT	短期スワップ値。	double
SYMBOL_SWAP_SUNDAY	日曜日から翌日にロールオーバーされたオーバーナイトポジションのスワップ計算比率（SYMBOL_SWAP_LONG または SYMBOL_SWAP_SHORT）。次の値がサポートされています	double

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
	• 0 - スワップは請求されない • 1 - シングルスワップ • 3 - トリプルスワップ	
SYMBOL_SWAP_MONDAY	月曜日から火曜日にロールオーバーされたオーバーナイトポジションのスワップ計算比率（SYMBOL_SWAP_LONG	double

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
	またはSYMBOL_SWAP_SHORT）	
SYMBOL_SWAP_TUESDAY	火曜日から水曜日にロールオーバーされたオーバーナイトポジションのスワップ計算比率（SYMBOL_SWAP_LONGまたはSYMBOL_SWAP_SHORT）	double
SYMBOL_SWAP_WEDNESDAY	水曜日から木曜日にロールオーバーされたオーバーナイトポジションのスワップ計算比率（SYMBOL_SWAP_LONGまたはSYMBOL_SWAP_SHORT）	double
SYMBOL_SWAP_THURSDAY	木曜日から金曜日にロールオーバーさ	double

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
	れたオーバーナイトポジションのスワップ計算比率（SYMBOL_SWAP_LONGまたはSYMBOL_SWAP_SHORT）	
SYMBOL_SWAP_FRIDAY	金曜日から土曜日にロールオーバーされたオーバーナイトポジションのスワップ計算比率（SYMBOL_SWAP_LONGまたはSYMBOL_SWAP_SHORT）	double
SYMBOL_SWAP_SATURDAY	土曜日から日曜日にロールオーバーされたオーバーナイトポジションのスワップ計算比率（SYMBOL_SWAP_LONG	double

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
	または SYMBOL_SWAP_SHORT）	
SYMBOL_MARGIN_INITIAL	当初証拠金とは、1ロットのボリュームで注文を出すのに必要な証拠金の証拠金通貨での量です。これは、クライアントが市場に入った時に、資産をチェックするために使用されます。 SymbolInfoMarginRate() 関数は、注文の種類と方向に応じて必要な証拠金額に関するデータを提供します。	double
SYMBOL_MARGIN_MAINTENANCE	維持証拠金。	double

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
	設定されている場合、1ロットから課金される、シンボルの証拠金通貨での証拠金量を設定します。これは、クライアントの口座状態が変わった時に、資産をチェックするために使用されます。維持証拠金が 0 に等しい場合、当初証拠金が使用されます。 SymbolInfoMarginRate() 関数は、注文の種類と方向に応じて必要な証拠金額に関するデータ	

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
	を提供します。	
SYMBOL_SESSION_VOLUME	現在のセッション約定ボリュームの概要。	double
SYMBOL_SESSION_TURNOVER	現在のセッションの概要売上高。	double
SYMBOL_SESSION_INTEREST	建玉概要。	double
SYMBOL_SESSION_BUY_ORDERS_VOLUME	現在の買い注文のボリューム。	double
SYMBOL_SESSION_SELL_ORDERS_VOLUME	現在の売り注文のボリューム。	double
SYMBOL_SESSION_OPEN	現在のセッションの始値。	double
SYMBOL_SESSION_CLOSE	現在のセッションの終値。	double
SYMBOL_SESSION_AW	現在のセッションの加重平均価格。	double
SYMBOL_SESSION_PRICE_SETTLEMENT	現在のセッションの決済価格。	double

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
SYMBOL_SESSION_PRICE_LIMIT_MIN	現在のセッションの最安値。	double
SYMBOL_SESSION_PRICE_LIMIT_MAX	現在のセッションの最高値。	double
SYMBOL_MARGIN_HEDGED	ヘッジポジション（1つのシンボルでの両建て）の1ロットの約定または証拠金の額。ヘッジポジションの証拠金計算の方法は2つあります。計算方法はブローカーによって決められます。 基本的な計算： • 商品に対し初期証拠金(SYMBOL_MARGIN_INITIAL)が設定さ	double

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
	れている場合、ヘッジされた証拠金は絶対値（金額の）として表示されます。 • 初期証拠金が設定されていない（0と同等）場合、SYMBOL_MARGIN_HEDGEDには、取引商品の種別に応じた式(SYMBOL_TRADE_CALC_MODE)による証拠金計算に	

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
	使用される約定額が表示されます。 最大ポジションによる計算： • SYMBOL_MARGIN_HEDGEDの値は考慮されません。 • 商品によって全てのロング/ショートポジションのボリュームが算出されます。 • 各側に発注の平均加重価格や、証拠金通貨への交換の平均	

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
	加重価格が計算されます。 • 商品種別による式 (SYMBOL_TRADE_CALC_MODE) で、ロング/ショートポジションの証拠金が計算されます。 • 最も大きい数値が、最終値として使用されます。	
SYMBOL_PRICE_CHANGE	前の取引日の終了に対する現在の価格の変化(%)	double
SYMBOL_PRICE_VOLATILITY	価格変動(%)	double

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
SYMBOL_PRICE_THEORETICAL	理論上のオプション価格	double
SYMBOL_PRICE_DELTA	オプション/ワラントデルタは、基礎となる資産価格が1変化したときに、オプション価格が変化する値を示します	double
SYMBOL_PRICE_THETA	オプション/ワラントシータは、一時的な崩壊(有効期限が近づいた)によってオプション価格が毎日失うポイント数を示します	double
SYMBOL_PRICE_GAMMA	オプション/ワラントガンマは、デルタの変化率、つまりオプションプレミアム	double

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
	の変化の速さを示します	
SYMBOL_PRICE_VEGA	オプション/ワラントベガは、ボラティリティが1%変化したときにオプション価格が変化するポイント数を示します	double
SYMBOL_PRICE_RHO	オプション/ワラントローは、理論上のオプション価格の1%変化に対する金利への感応度を反映しています	double
SYMBOL_PRICE_OMEGA	オプション/ワラントオメガ。オプションの弾力性は、基になる資産価格の変化率によるオプション価	double

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
	格の相対的な変化率を示します	
SYMBOL_PRICE_SENSITIVITY	オプション/ワラントの感応度は、オプション価格が1ポイント変化するためにオプションの原資産の価格が何ポイント変化するかを示します	double

SymbolInfoString() 関数では

ENUM_SYMBOL_INFO_STRING

識別子	説明	種類
SYMBOL_BASIS	デリバティブの原資産。	string
SYMBOL_CATEGORY	取引シンボルが属するセクターまたはカテゴリの名前。	string
SYMBOL_COUNTRY	金融銘柄が属する国	string
SYMBOL_SECTOR_NAME	金融銘柄が属する経済部門	string

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
SYMBOL_INDUSTRY_NAME	金融シンボルが属する業界のブランチまたは業界	string
SYMBOL_CURRENCY_BASE	シンボルの基本通貨。	string
SYMBOL_CURRENCY_PROFIT	利益通貨。	string
SYMBOL_CURRENCY_MARGIN	証拠金通貨。	string
SYMBOL_BANK	現在の相場のフィーダー。	string
SYMBOL_DESCRIPTION	シンボル説明。	string
SYMBOL_EXCHANGE	金融シンボルが取引されている取引所の名前。	string
SYMBOL_FORMULA	カスタムシンボルプライシングに使用される公式。数式で使用される金融シンボルの名前が数字で始まるか特殊文字(">" "、"。"、"-"、"&"、"#"ит.д.)を含む場合、引用符が シンボル名の周りで使	string

© 2000-2025, MetaQuotes Ltd.

識別子	説明	種類
	用されます。 • 複合シンボル："@ESU19"/EURCADカレンダースプレッド："Si-9.13"-"Si-	

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
	6.13" • ユーロインデックス：34.38805726*pow(EURUSD,0.31555)*pow	

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
	(EURGBP,0.30566)*pow(EURJPY,0.18911)*pow(EURCHF,0.1	

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
	113)*pow(EURSEK,0.07855)	
SYMBOL_ISIN	ISIN システム（International Securities Identification Number）での銘柄名。International Securities Identification Number はセキュリティを識別する一意の 12 桁の英数字コードです。このシンボルプロパティの存在	string

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
	は取引サーバ側で決定されます。	
SYMBOL_PAGE	シンボル情報を含むWebページのアドレスで、端末でシンボルプロパティを表示するときにリンクとして表示されます。	string
SYMBOL_PATH	シンボルツリーパス。	string

シンボル価格表は、Bid価格またはLast価格に基づくことができます。シンボルチャートのために選択された価格は、端末のバーの生成および表示にも影響します。SYMBOL_CHART_MODEプロパティの値は、ENUM_SYMBOL_CHART_MODEに記述されています。

ENUM_SYMBOL_CHART_MODE

識別子	説明
SYMBOL_CHART_MODE_BID	バーがBid価格に基づいています。
SYMBOL_CHART_MODE_LAST	バーがLast価格に基づいています。

各シンボルで、未決注文の期限のモードを指定することが出来ます。フラグが各モードに一致します。フラグは SYMBOL_EXPIRATION_GTC| SYMBOL_EXPIRATION_SPECIFIED のように、論理 **OR**(|)を持って結合することが出来ます。モードがシンボルで許可されているかどうかの確認には、論理 **AND**(&)を使用してのフラグとの比較が使用されます。.

シンボルに SYMBOL_EXPIRATION_SPECIFIED フラグが指定されている場合、送信時に未決注文の期限を指定することが出来ます。

識別子	値	説明
SYMBOL_EXPIRATION_GTC	1	注文は明示的にキャンセルされるまで、無制限の期間中有効です。
SYMBOL_EXPIRATION_DAY	2	注文は日の終わりまで有効です。

© 2000–2025, MetaQuotes Ltd.

識別子	値	説明
SYMBOL_EXPIRATION_SPECIFIED	4	有効期限の時刻が注文に指定されています。
SYMBOL_EXPIRATION_SPECIFIED_DAY	8	有効期限の日付が注文に指定されています。

例：

```
//+------------------------------------------------------------------+
//|  指定された有効期限のモードが許可されているかをチェックする                          |
//+------------------------------------------------------------------+
bool IsExpirationTypeAllowed(string symbol,int exp_type)
  {
//--- 許可された有効期限のモードについて説明したプロパティの値を取得する
   int expiration=(int)SymbolInfoInteger(symbol,SYMBOL_EXPIRATION_MODE);
//--- exp_type モードが許可されている場合はtrue を返す
   return((expiration&exp_type)==exp_type);
  }
```

SYMBOL_EXPIRATION_MODEプロパティがSYMBOL_EXPIRATION_GTC（キャンセルするまで）に設定されている場合、未決注文の有効期限と決済逆指注文及び決済指値注文は、
ENUM_SYMBOL_ORDER_GTC_MODE列挙を使用して追加設定する必要があります。

ENUM_SYMBOL_ORDER_GTC_MODE

識別子	説明
SYMBOL_ORDERS_GTC	未決注文と決済逆指値及び決済指値は、明示的なキャンセルが行われるまで無期限に有効です。
SYMBOL_ORDERS_DAILY	注文は1営業日のみ有効です。一日の終わりに、未決注文や決済逆指値及び決済指値はすべて削除されます。
SYMBOL_ORDERS_DAILY_EXCLUDING_STOPS	取引日が変更すると、未決注文だけが削除され、決済逆指値及び決済指値は保持されます。

注文を送信するときに、注文に設定されたボリュームの注文ポリシーを指定できます。各銘柄の可能なボリュームベースの注文実行オプションは、表に指定されています。フラグの組み合わせにより、商品ごとに複数のモードを設定することができます。フラグの組み合わせは、論理的なOR (|)演算で表されます(例:SYMBOL_FILLING_FOK|SYMBOL_FILLING_IOC)。商品に特定のモードが許可されているかどうかを確認するには、論理的なAND (&)の結果をモードフラグと比較します - 例

実行ポリシー	ID	値	説明
FOK	SYMBOL_FILLING_FOK	1	注文

© 2000-2025, MetaQuotes Ltd.

実行ポリシー	ID	値	説明
			は指定されたボリュームでのみ実行できます。必要な量の金融商品が現在市場で入手できない場合、注文

© 2000–2025, MetaQuotes Ltd.

実行ポリシー	ID	値	説明
			は実行されません。必要なボリュームは、いくつかの利用可能なオファーで構成できます。注文を送信す

© 2000–2025, MetaQuotes Ltd.

実行ポリシー	ID	値	説明
			るときに、このポリシーにORDER_FILLING_FOK注文タイプを指定できます。FOK注文

© 2000–2025, MetaQuotes Ltd.

実行ポリシー	ID	値	説明
			を使用する可能性は、取引サーバで決定されます。
即時またはキャンセル	SYMBOL_FILLING_IOC	2	トレーダーは、注文に示された量の範囲内で市場で最大

© 2000–2025, MetaQuotes Ltd.

実行ポリシー	ID	値	説明
			限に利用可能な量で取引を実行することに同意します。リクエストを完全に満たすことができない場合、利

© 2000–2025, MetaQuotes Ltd.

実行ポリシー	ID	値	説明
			用可能なボリュームでの注文が実行され、残りのボリュームはキャンセルされます。注文を送信するとき

© 2000–2025, MetaQuotes Ltd.

実行ポリシー	ID	値	説明
			に、このポリシーにORDER_FILLING_IOC実行タイプを指定できます。IOC注文を使用

© 2000–2025, MetaQuotes Ltd.

実行ポリシー	ID	値	説明
			する可能性は、取引サーバで決定されます。
パッシブ	SYMBOL_FILLING_BOC	4	TheBOC(Book-or-Cancel)ポリシーは、

© 2000–2025, MetaQuotes Ltd.

実行ポリシー	ID	値	説明
			注文が板情報でのみ発注でき、すぐに実行できないことを前提としています。発注時にすぐに約定できる場

© 2000–2025, MetaQuotes Ltd.

実行ポリシー	ID	値	説明
			合、注文はキャンセルされます。 I実際、BOCのポリシーは、発注された注文の価格が現在の市場より

© 2000–2025, MetaQuotes Ltd.

実行ポリシー	ID	値	説明
			も悪くなる場合のみに指定できます。BOC注文はパッシブ取引を実装するために使用されるため、注文が

© 2000–2025, MetaQuotes Ltd.

実行ポリシー	ID	値	説明
			出されてもすぐには実行されず、現在の流動性には影響しません。指値注文と逆指値注文のみがサポート

© 2000–2025, MetaQuotes Ltd.

実行ポリシー	ID	値	説明
			されています（SYMBOL_ORDER_MODEフラグにはSYMBOL_ORDER_LIMITお

© 2000–2025, MetaQuotes Ltd.

実行ポリシー	ID	値	説明
			よび/またはSYMBOL_ORDER_STOP_LIMIT野値が含まれます）。
リターン	識別子なし		部分的な実行の場合

© 2000–2025, MetaQuotes Ltd.

実行ポリシー	ID	値	説明
			、残りのボリュームのある注文はキャンセルされず、さらに処理されます。注文を送信するときに、この

© 2000–2025, MetaQuotes Ltd.

実行ポリシー	ID	値	説明
			ポリシーにORDER_FILLING_RETURN実行タイプを指定できます。成行実行モード（成

© 2000–2025, MetaQuotes Ltd.

実行ポリシー	ID	値	説明
			行実行－SYMBOL－TRADE－EXECUTION－MARKET）では、リターン注文は許可され

© 2000–2025, MetaQuotes Ltd.

実行ポリシー	ID	値	説明
			ていません。

OrderSend()関数を使用して取引リクエストを送信する場合、必要なボリューム実行ポリシーをtype_fillingフィールド(特別なMqlTradeRequest構造体)で設定できます。 ENUM_ORDER_TYPE_FILLING列挙からの値が利用可能です。注文タイプが指定されていない場合、取引リクエストでORDER_FILLING_RETURNが自動的に設定されます。ORDER_FILLING_RETURN実行タイプは、「成行実行」(SYMBOL_TRADE_EXECUTION_MARKET)を除くすべての実行モードで有効になります。

現在の時刻で実行のための取引リクエストの送信中(有効時間)、金融市場では、リクエストされたボリューム全体が特定の金融商品で希望の価格で利用できるという保証はありません。したがって、リアルタイムの取引操作は、価格およびボリューム実行モードを使用して規制されます。モードまたは実行ポリシーは、価格が変更された場合、またはリクエストされたボリュームを**現時点**で完全に満たすことができない場合のルールを定義します。

実行モード	説明	ENUM_SYMBOL_TRADE_EXECUTIONの値
実行モード (リクエスト実行)	以前に証券会社から受け取った価格で成行注文を実行します。 特定の成行注文の価格は、注文が送信される前に証券会社にリクエストされます。価格を受け取ると、指定さ	SYMBOL_TRADE_EXECUTION_REQUEST

© 2000–2025, MetaQuotes Ltd.

実行モード	説明	ENUM_SYMBOL_TRADE_EXECUTIONの値
	れた価格での注文の実行を確認または拒否できます。	
即時実行 （即時実行）	指定された価格ですぐに成行注文を実行します。 実行する取引リクエストを送信すると、プラットフォームは自動的に現在の価格を注文に追加します。 • 証券会社が価格を受け入れると、注	SYMBOL_TRADE_EXECUTION_INSTANT

© 2000–2025, MetaQuotes Ltd.

実行モード	説明	ENUM_SYMBOL_TRADE_EXECUTIONの値
	文が実行されます。 • 証券会社がリクエストされた価格を受け入れない場合、「リクオート」が送信され、証券	

© 2000–2025, MetaQuotes Ltd.

実行モード	説明	ENUM_SYMBOL_TRADE_EXECUTIONの値
	会社はこの注文を実行できる価格を返します。	
成行執行 （成行執行）	証券会社は、トレーダーとの追加の話し合いなしに、注文執行価格について決定を下します。 このようなモードで注文を送信することは、この価格での実行に事前に	SYMBOL_TRADE_EXECUTION_MARKET

© 2000–2025, MetaQuotes Ltd.

実行モード	説明	ENUM_SYMBOL_TRADE_EXECUTIONの値
	同意することを意味します。	
取引所実行 (取引所実行)	取引操作は、現在の市場オファーの価格で実行されます。	SYMBOL_TRADE_EXECUTION_EXCHANGE

現在の実行時間で注文を送信する前に、ORDER_TYPE_FILLING値(ボリューム実行タイプ)を正しく設定するには、各金融商品でSYMBOL_FILLING_MODEプロパティ値を取得するのにSymbolInfoInteger()関数を使用できます。これは銘柄で許可されたボリューム実行タイプをフラグの組み合わせとして示します。ORDER_FILLING_RETURN注文タイプは、「成行実行」モード(SYMBOL_TRADE_EXECUTION_MARKET)を除いて常に有効です。

実行モードに応じた注文タイプの使用法を次の表に示します。

実行の種類¥実行ポリシー	FOK(FOK ORDER_FILLING_FOK)	即時またはキャンセル(IOC ORDER_FILLING_IOC)	リターン(リターン ORDER_FILLING_RETURN)
即時実行 (SYMBOL_TRADE_EXECUTION_INSTANT)	+(銘柄の設定に関係なく)	+(銘柄の設定に関係なく)	+(常に)
リクエスト実行 SYMBOL_TRADE_EXECUTION_REQUEST	+(銘柄の設定に関係なく)	+(銘柄の設定に関係なく)	+(常に)
成行執行 SYMBOL_TRADE_EXECUTION_MARKET	+(銘柄設定で設定)	+(銘柄設定で設定)	-(銘柄設定に関係なく無効)
取引所実行 SYMBOL_TRADE_EXECUTION_EXCHANGE	+(銘柄設定で設定)	+(銘柄設定で設定)	+(常に)

未決注文の場合、そのような注文は送信時点での実行を目的としていないため、実行タイプに関係なくORDER_FILLING_RETURN注文タイプ(SYMBOL_TRADE_EXEMODE)を使用する必要があります。未決注文を使用する場合、トレーダーは、この注文の取引条件が満たされたときに、証券会社が取引所でサポートされている注文タイプを使用することに事前に同意します。

例:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  特定の実行モードが許可されているかどうかを確認する                |
//+------------------------------------------------------------------+
bool IsFillingTypeAllowed(string symbol,int fill_type)
  {
//--- 実行モードを説明するプロパティの値を取得する
   int filling=(int)SymbolInfoInteger(symbol,SYMBOL_FILLING_MODE);
//--- fill_typeモードが許可されている場合は「true」を返す
   return((filling&fill_type)==fill_type);
  }
```

OrderSend() 関数を使用して取引リクエストを送る場合、操作によっては注文の種類をENUM_ORDER_TYPE 列挙から指定することが必要です。シンボルには全ての注文の種類が許可されていない場合があります。SYMBOL_ORDER_MODE プロパティは許可されている注文種類のフラグを記述します。

識別子	値	説明
SYMBOL_ORDER_MARKET	1	成行注文は売買とも使用可能です。
SYMBOL_ORDER_LIMIT	2	指値注文（買い指値と売り指値）は使用可能です。
SYMBOL_ORDER_STOP	4	逆指注文（売買）は使用可能です。
SYMBOL_ORDER_STOP_LIMIT	8	指値つきストップ注文は売買とも使用可能です。
SYMBOL_ORDER_SL	16	逆指注文は可能です。
SYMBOL_ORDER_TP	32	決済指は使用可能です。
SYMBOL_ORDER_CLOSEBY	64	Close By操作の許可、つまり、同じ製品の反対方向のポジションによる決済です。このプロパティは、ヘッジ会計システム（ACCOUNT_MARGIN_MODE_RETAIL_HEDGING）を持つ口座で設定されます。

例：

```
//+------------------------------------------------------------------+
//|  この関数は、シンボルに許可されている注文の種類を出力する          |
//+------------------------------------------------------------------+
void Check_SYMBOL_ORDER_MODE(string symbol)
  {
//--- 許可された注文の種類について説明したプロパティの値を取得する
   int symbol_order_mode=(int)SymbolInfoInteger(symbol,SYMBOL_ORDER_MODE);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 成行注文をチェックする(マーケット実行)
   if((SYMBOL_ORDER_MARKET&symbol_order_mode)==SYMBOL_ORDER_MARKET)
      Print(symbol+": Market orders are allowed (Buy and Sell)");
//--- リミット注文をチェックする
   if((SYMBOL_ORDER_LIMIT&symbol_order_mode)==SYMBOL_ORDER_LIMIT)
      Print(symbol+": Buy Limit and Sell Limit orders are allowed");
//--- ストップ注文をチェックする
   if((SYMBOL_ORDER_STOP&symbol_order_mode)==SYMBOL_ORDER_STOP)
      Print(symbol+": Buy Stop and Sell Stop orders are allowed");
//--- ストップリミット注文をチェックする
   if((SYMBOL_ORDER_STOP_LIMIT&symbol_order_mode)==SYMBOL_ORDER_STOP_LIMIT)
      Print(symbol+": Buy Stop Limit and Sell Stop Limit orders are allowed");
//--- 逆指注文が許可されているかをチェックする
   if((SYMBOL_ORDER_SL&symbol_order_mode)==SYMBOL_ORDER_SL)
      Print(symbol+": Stop Loss orders are allowed");
//--- 指値注文が許可されているかをチェックする
   if((SYMBOL_ORDER_TP&symbol_order_mode)==SYMBOL_ORDER_TP)
      Print(symbol+": Take Profit orders are allowed");
//--- 反対のポジションでポジションを決済することが許可されているかどうかを確認する
   if((SYMBOL_ORDER_TP&symbol_order_mode)==SYMBOL_ORDER_CLOSEBY)
      Print(symbol+": Close by allowed");
//---
  }
```

ENUM_SYMBOL_CALC_MODE 列挙は、シンボルの証拠金要件を計算する方法についての情報を得るために使用されます。

ENUM_SYMBOL_CALC_MODE

識別子	説明	方式
SYMBOL_CALC_MODE_FOREX	外為モード－外国為替の利益と証拠金の計算。	Margin: Lots * Contract_Size / Leverage * Margin_Rate 利益:(close_price − open_price) * Contract_Size*Lots
SYMBOL_CALC_MODE_FOREX_NO_LEVERAGE	レバレッジなし外為モ	証拠金:Lots＊Contract_Size＊Margin_Rate 利益:(close_price − open_price)＊Contract_Size＊Lots

© 2000−2025, MetaQuotes Ltd.

識別子	説明	方式
	ード－レバレッジを考慮しない外国為替銘柄の利益と証拠金の計算	
SYMBOL_CALC_MODE_FUTURES	先物モード－先物の証拠金と利益の計算。	証拠金：Lots＊InitialMargin＊Margin_Rate 利益：（close_price － open_price）＊ TickPrice / TickSize*Lots
SYMBOL_CALC_MODE_CFD	CFDモード－CFDの証拠金と利益計算。	証拠金：Lots＊ContractSize＊MarketPrice＊Margin_Rate 利益：（close_price－open_price）＊Contract_Size＊Lots
SYMBOL_CALC_MODE_CFDINDEX	CFDインデックスモード－インデックスを使用したCFD	証拠金：（Lots＊ContractSize＊MarketPrice）＊TickPrice / TickSize * Margin_Rate 利益：（close_price－open_price）＊Contract_Size＊Lots

© 2000–2025, MetaQuotes Ltd.

識別子	説明	方式
	の証拠金と利益計算	
SYMBOL_CALC_MODE_CFDLEVERAGE	CFD レバレッジモード - レバレッジ取引での CFD の証拠金と利益計算	証拠金：(Lots * ContractSize * MarketPrice) / Leverage * Margin_Rate 利益：(close_price-open_price) * Contract_Size * Lots
SYMBOL_CALC_MODE_EXCH_STOCKS	交換モード - 証券取引所で有価証券を取引するための証拠金と利益計算。	証拠金：Lots * ContractSize * OpenPrice * Margin_Rate 利益：(close_price - open_price) * Contract_Size * Lots
SYMBOL_CALC_MODE_EXCH_FUTURES	先物モード - 証券取引所での取	証拠金：Lots * InitialMargin * Margin_Rate или Lots * MaintenanceMargin * Margin_Rate 利益：(close_price - open_price) * Lots * TickPrice / TickSize

© 2000-2025, MetaQuotes Ltd.

識別子	説明	方式
	引先物契約の証拠金と利益計算。	
SYMBOL_CALC_MODE_EXCH_FUTURES_FORTS	FORTS先物モード－FORTSでの取引先物契約の証拠金と利益計算。証拠金は、次の規則に従ってMarginDiscountのずれ量によって減少させることが出来ます。1. ロング	証拠金：Lots＊InitialMargin＊Margin_Rate или Lots * MaintenanceMargin * Margin_Rate 利益：(close_price－open_price)＊Lots＊TickPrice / TickSize

© 2000–2025, MetaQuotes Ltd.

識別子	説明	方式
	ポジション（買い注文）の価格が見積価格よりも低い場合、証拠金割引 = ロット数 * ((Settle価格 - Order価格) * ティック価格 / ティックサイズ) 2. ショートポジション（売り注文）の価格が予定価格を超えた場合、証拠金割	

© 2000–2025, MetaQuotes Ltd.

識別子	説明	方式
	引 = ロット数 * ((Order価格 – Settle価格) * ティック価格 / ティックサイズ) ここで ○ Settle価格 - 前のセッションの推定（クリア）価格 ○ Order	

© 2000–2025, MetaQuotes Ltd.

識別子	説明	方式
	価格 - 加重平均位置の価格や注文（リクエスト）に設定された始値。○ティック価格 - ティック値（一ホイン	

© 2000–2025, MetaQuotes Ltd.

識別子	説明	方式
	トの価格変動によるコスト）○ ティックサイズ - ティックサイズ（最低価格変更ステップ）	
SYMBOL_CALC_MODE_EXCH_BONDS	Exchange Bonds mode - calculatio	Margin: Lots * ContractSize * FaceValue * open_price * /100 Profit: Lots * close_price * FaceValue * Contract_Size + AccruedInterest * Lots * ContractSize

© 2000–2025, MetaQuotes Ltd.

識別子	説明	方式
	n of margin and profit for trading bonds on a stock exchange	
SYMBOL_CALC_MODE_EXCH_STOCKS_MOEX	Exchange MOEX Stocks mode - calculation of margin and profit for trading securities on MOEX	Margin: Lots * ContractSize * LastPrice * Margin_Rate Profit: (close_price - open_price) * Contract_Size * Lots
SYMBOL_CALC_MODE_EXCH_BONDS_MOEX	Exchange MOEX Bonds mode - calculatio	Margin: Lots * ContractSize * FaceValue * open_price * /100 Profit: Lots * close_price * FaceValue * Contract_Size + AccruedInterest * Lots * ContractSize

© 2000–2025, MetaQuotes Ltd.

識別子	説明	方式
	n of margin and profit for trading bonds on MOEX	
SYMBOL_CALC_MODE_SERV_COLLATERAL	担保モード – シンボルは、取引口座の取引不可能の資産として使用されています。ポジションの市場価値は、ボリューム、現在の市場価格、契約の大きさと流動	証拠金 : なし 利益 : なし 市場価格ロット数 * 契約サイズ * マーケット価格 * LiqudityRate

© 2000–2025, MetaQuotes Ltd.

識別子	説明	方式
	比率に基づいて算出されます。値は、株式に追加された資産の中に含まれます。この様なシンボルのポジションは余剰証拠金を増加し、ポジションの追加証拠金（担保）として使用されます。	

© 2000–2025, MetaQuotes Ltd.

シンボル取引モードはいくつかあります。特定のシンボルの取引モードの詳細は、ENUM_SYMBOL_TRADE_MODE 列挙の値に反映されます。

ENUM_SYMBOL_TRADE_MODE

識別子	説明
SYMBOL_TRADE_MODE_DISABLED	シンボルの取引が無効。
SYMBOL_TRADE_MODE_LONGONLY	ロング（買い）ポジションのみ許可。
SYMBOL_TRADE_MODE_SHORTONLY	ショート（売り）ポジションのみ許可。
SYMBOL_TRADE_MODE_CLOSEONLY	ポジション決済操作のみを許可。
SYMBOL_TRADE_MODE_FULL	取引制限なし。

可能なシンボルの約定実行モードは ENUM_SYMBOL_TRADE_EXECUTION で定義されています。

ENUM_SYMBOL_TRADE_EXECUTION

識別子	説明
SYMBOL_TRADE_EXECUTION_REQUEST	リクエストによる実行。
SYMBOL_TRADE_EXECUTION_INSTANT	即時実行。
SYMBOL_TRADE_EXECUTION_MARKET	マーケット実行。
SYMBOL_TRADE_EXECUTION_EXCHANGE	エクスチェンジ実行。

ポジション転送時のスワップ計算の方法は ENUM_SYMBOL_SWAP_MODE 列挙で指定されています。スワップ計算の方法は SYMBOL_SWAP_LONG 及び SYMBOL_SWAP_SHORT パラメータの測定単位を決定します。例えば、スワップがクライアントの預金通貨で請求された場合、それらのパラメータの値は、クライアントの預金通貨での金額として指定されます。

ENUM_SYMBOL_SWAP_MODE

識別子	説明
SYMBOL_SWAP_MODE_DISABLED	スワップ無効。
SYMBOL_SWAP_MODE_POINTS	スワップはポイントとして請求されます。
SYMBOL_SWAP_MODE_CURRENCY_SYMBOL	スワップはシンボルの基本通貨で払われます。
SYMBOL_SWAP_MODE_CURRENCY_MARGIN	スワップはシンボルの証拠金通貨で払われます。
SYMBOL_SWAP_MODE_CURRENCY_DEPOSIT	スワップは、クライアントの預金通貨で請求されます。
SYMBOL_SWAP_MODE_CURRENCY_PROFIT	スワップは利益計算通貨で請求されます

© 2000–2025, MetaQuotes Ltd.

識別子	説明
SYMBOL_SWAP_MODE_INTEREST_CURRENT	スワップはスワップの計算の商品価格の指定年利として請求されます（標準銀行年は360日です）。
SYMBOL_SWAP_MODE_INTEREST_OPEN	スワップはポジションの始値から指定年利として請求されます（標準銀行年は360日です）。
SYMBOL_SWAP_MODE_REOPEN_CURRENT	スワップはポジションを再オープンすることによって請求されます。ポジションは取引日の終わりに閉じます。次の日に、終値 +/- 指定されたポイントの数（パラメータ SYMBOL_SWAP_LONG 及び SYMBOL_SWAP_SHORT）で開けられます。
SYMBOL_SWAP_MODE_REOPEN_BID	スワップはポジションを再オープンすることによって請求されます。ポジションは取引日の終わりに閉じます。次の日に、現在の売値+/- 指定されたポイントの数（SYMBOL_SWAP_LONG 及び SYMBOL_SWAP_SHORT パラメータ）で開けられます。

ENUM_DAY_OF_WEEK 列挙の値は曜日の指定に使用されます。

ENUM_DAY_OF_WEEK

識別子	説明
SUNDAY	日曜。
MONDAY	月曜。
TUESDAY	火曜。
WEDNESDAY	水曜。
THURSDAY	木曜。
FRIDAY	金曜。
SATURDAY	土曜。

オプションとは、特定の日またはそれ以前に特定の価格で原資産（商品、株式、先物、など）を購入したり、販売する権利（義務ではない）を与える契約です。以下の列挙はオプションの種類とそれから生じる権利を含むオプションのプロパティを記述します。

ENUM_SYMBOL_OPTION_RIGHT

識別子	説明
SYMBOL_OPTION_RIGHT_CALL	コールオプションを使用すると、指定された価格で資産を買うことが出来ます。
SYMBOL_OPTION_RIGHT_PUT	プット·オプションを使用すると、指定された価格で資産を売ることが出来ます。

© 2000-2025, MetaQuotes Ltd.

ENUM_SYMBOL_OPTION_MODE

識別子	説明
SYMBOL_OPTION_MODE_EUROPEAN	ヨーロッパのオプションは、指定された日付（有効期限、実行日、配達日）にのみ行使出来ます。
SYMBOL_OPTION_MODE_AMERICAN	アメリカのオプションは満了前の任意の取引日に行使することが出来ます。買い手がオプションを行使することが出来る期間は指定されています。

金融商品は、経済部門によって分類されています。経済部門は特定の特性、経済目標、機能、行動を持つ経済活動の一部であり、この部門を経済の他の部分から分離することができます。
ENUM_ SYMBOL_ SECTORは、取引商品が属することができる経済部門をリストします。

ENUM_SYMBOL_SECTOR

ID	説明
SECTOR_UNDEFINED	未定義
SECTOR_BASIC_MATERIALS	基本材料
SECTOR_COMMUNICATION_SERVICES	通信サービス
SECTOR_CONSUMER_CYCLICAL	消費者向けサイクル
SECTOR_CONSUMER_DEFENSIVE	消費者防御
SECTOR_CURRENCY	通貨
SECTOR_CURRENCY_CRYPTO	暗号通貨
SECTOR_ENERGY	エネルギー
SECTOR_FINANCIAL	ファイナンス
SECTOR_HEALTHCARE	ヘルスケア
SECTOR_INDUSTRIALS	産業
SECTOR_REAL_ESTATE	不動産
SECTOR_TECHNOLOGY	技術
SECTOR_UTILITIES	ユーティリティ

各金融商品は、特定の種類の業界または経済部門に割り当てることができます。業界とは、密接に関連する一連の原材料、商品、またはサービスを生産する経済の一分野です。ENUM_ SYMBOL_ INDUSTRYは、取引商品が属することができる業界をリストします。

ENUM_SYMBOL_INDUSTRY

© 2000–2025, MetaQuotes Ltd.

ID	説明
INDUSTRY_UNDEFINED	未定義
基本材料	
INDUSTRY_AGRICULTURAL_INPUTS	農業投入
INDUSTRY_ALUMINIUM	アルミ
INDUSTRY_BUILDING_MATERIALS	建材
INDUSTRY_CHEMICALS	化学薬品
INDUSTRY_COKING_COAL	原料炭
INDUSTRY_COPPER	銅
INDUSTRY_GOLD	金
INDUSTRY_LUMBER_WOOD	木材と木材生産
INDUSTRY_INDUSTRIAL_METALS	その他の産業用金属および鉱業
INDUSTRY_PRECIOUS_METALS	その他の貴金属と鉱業
INDUSTRY_PAPER	紙と紙製品
INDUSTRY_SILVER	銀
INDUSTRY_SPECIALTY_CHEMICALS	特殊化学品
INDUSTRY_STEEL	鋼
通信サービス	
INDUSTRY_ADVERTISING	広告代理店
INDUSTRY_BROADCASTING	放送
INDUSTRY_GAMING_MULTIMEDIA	電子ゲームとマルチメディア
INDUSTRY_ENTERTAINMENT	エンターテインメント
INDUSTRY_INTERNET_CONTENT	インターネットコンテンツと情報
INDUSTRY_PUBLISHING	出版
INDUSTRY_TELECOM	電気通信サービス
消費者向けサイクル	
INDUSTRY_APPAREL_MANUFACTURING	アパレル製造
INDUSTRY_APPAREL_RETAIL	アパレル小売
INDUSTRY_AUTO_MANUFACTURERS	自動車メーカー
INDUSTRY_AUTO_PARTS	自動車部品
INDUSTRY_AUTO_DEALERSHIP	自動車およびトラック販売店

© 2000–2025, MetaQuotes Ltd.

ID	説明
INDUSTRY_DEPARTMENT_STORES	デパート
INDUSTRY_FOOTWEAR_ACCESSORIES	履物とアクセサリー
INDUSTRY_FURNISHINGS	家具、備品、電化製品
INDUSTRY_GAMBLING	ギャンブル
INDUSTRY_HOME_IMPROV_RETAIL	ホームセンター小売
INDUSTRY_INTERNET_RETAIL	インターネット小売
INDUSTRY_LEISURE	余暇
INDUSTRY_LODGING	宿泊
INDUSTRY_LUXURY_GOODS	高級品
INDUSTRY_PACKAGING_CONTAINERS	包装とコンテナ
INDUSTRY_PERSONAL_SERVICES	パーソナルサービス
INDUSTRY_RECREATIONAL_VEHICLES	Recreational vehicles
INDUSTRY_RESIDENT_CONSTRUCTION	住宅建設
INDUSTRY_RESORTS_CASINOS	リゾートとカジノ
INDUSTRY_RESTAURANTS	レストラン
INDUSTRY_SPECIALTY_RETAIL	専門小売
INDUSTRY_TEXTILE_MANUFACTURING	繊維製造
INDUSTRY_TRAVEL_SERVICES	旅行サービス
消費者防御	
INDUSTRY_BEVERAGES_BREWERS	飲料–ビール
INDUSTRY_BEVERAGES_NON_ALCO	飲料–ノンアルコール
INDUSTRY_BEVERAGES_WINERIES	飲料–ワイナリー＆蒸留所
INDUSTRY_CONFECTIONERS	菓子類
INDUSTRY_DISCOUNT_STORES	ディスカウントストア
INDUSTRY_EDUCATION_TRAINIG	教育およびトレーニングサービス
INDUSTRY_FARM_PRODUCTS	農産物
INDUSTRY_FOOD_DISTRIBUTION	食品流通
INDUSTRY_GROCERY_STORES	食料品店
INDUSTRY_HOUSEHOLD_PRODUCTS	家庭用品および個人用品
INDUSTRY_PACKAGED_FOODS	包装食品

© 2000–2025, MetaQuotes Ltd.

ID	説明
INDUSTRY_TOBACCO	タバコ
エネルギー	
INDUSTRY_OIL_GAS_DRILLING	石油/ガス掘削
INDUSTRY_OIL_GAS_EP	石油/ガス抽出および処理
INDUSTRY_OIL_GAS_EQUIPMENT	石油/ガス機器およびサービス
INDUSTRY_OIL_GAS_INTEGRATED	石油/ガス統合
INDUSTRY_OIL_GAS_MIDSTREAM	石油/ガス中流
INDUSTRY_OIL_GAS_REFINING	石油/ガス精製/マーケティング
INDUSTRY_THERMAL_COAL	火力炭
INDUSTRY_URANIUM	ウラン
ファイナンス	
INDUSTRY_EXCHANGE_TRADED_FUND	取引所上場ファンド
INDUSTRY_ASSETS_MANAGEMENT	資産管理
INDUSTRY_BANKS_DIVERSIFIED	銀行-多様化
INDUSTRY_BANKS_REGIONAL	銀行-地域
INDUSTRY_CAPITAL_MARKETS	資本市場
INDUSTRY_CLOSE_END_FUND_DEBT	クローズドエンド型ファンド-負債
INDUSTRY_CLOSE_END_FUND_EQUITY	クローズドエンド型ファンド-株式
INDUSTRY_CLOSE_END_FUND_FOREIGN	クローズドエンド型ファンド-外国
INDUSTRY_CREDIT_SERVICES	信用サービス
INDUSTRY_FINANCIAL_CONGLOMERATE	金融コングロマリット
INDUSTRY_FINANCIAL_DATA_EXCHANGE	財務データと証券取引所
INDUSTRY_INSURANCE_BROKERS	保険ブローカー
INDUSTRY_INSURANCE_DIVERSIFIED	保険-多様化
INDUSTRY_INSURANCE_LIFE	保険-生命
INDUSTRY_INSURANCE_PROPERTY	保険-損害保険
INDUSTRY_INSURANCE_REINSURANCE	保険-再保険
INDUSTRY_INSURANCE_SPECIALTY	保険-専門
INDUSTRY_MORTGAGE_FINANCE	住宅ローンファイナンス
INDUSTRY_SHELL_COMPANIES	シェル会社

© 2000-2025, MetaQuotes Ltd.

ID	説明
ヘルスケア	
INDUSTRY_BIOTECHNOLOGY	バイオテクノロジー
INDUSTRY_DIAGNOSTICS_RESEARCH	診断と研究
INDUSTRY_DRUGS_MANUFACTURERS	医薬品メーカー—一般
INDUSTRY_DRUGS_MANUFACTURERS_SPEC	医薬品メーカー-専門＆ジェネリック
INDUSTRY_HEALTHCARE_PLANS	医療計画
INDUSTRY_HEALTH_INFORMATION	健康情報サービス
INDUSTRY_MEDICAL_FACILITIES	医療施設
INDUSTRY_MEDICAL_DEVICES	医療機器
INDUSTRY_MEDICAL_DISTRIBUTION	医療流通
INDUSTRY_MEDICAL_INSTRUMENTS	医療機器および消耗品
INDUSTRY_PHARM_RETAILERS	製薬小売業者
産業	
INDUSTRY_AEROSPACE_DEFENSE	航空宇宙および防衛 Aerospace and defense
INDUSTRY_AIRLINES	航空会社
INDUSTRY_AIRPORTS_SERVICES	空港および航空サービス
INDUSTRY_BUILDING_PRODUCTS	建材・設備
INDUSTRY_BUSINESS_EQUIPMENT	事務機器および消耗品
INDUSTRY_CONGLOMERATES	コングロマリット
INDUSTRY_CONSULTING_SERVICES	コンサルティングサービス
INDUSTRY_ELECTRICAL_EQUIPMENT	電気機器および部品
INDUSTRY_ENGINEERING_CONSTRUCTION	エンジニアリング＆建設
INDUSTRY_FARM_HEAVY_MACHINERY	農機および重建設機械
INDUSTRY_INDUSTRIAL_DISTRIBUTION	産業流通
INDUSTRY_INFRASTRUCTURE_OPERATIONS	インフラストラクチャオペレーション
INDUSTRY_FREIGHT_LOGISTICS	統合された貨物およびロジスティクス
INDUSTRY_MARINE_SHIPPING	海運
INDUSTRY_METAL_FABRICATION	金属加工
INDUSTRY_POLLUTION_CONTROL	汚染と処理の管理
INDUSTRY_RAILROADS	鉄道

© 2000–2025, MetaQuotes Ltd.

ID	説明
INDUSTRY_RENTAL_LEASING	レンタル＆リースサービス
INDUSTRY_SECURITY_PROTECTION	セキュリティと保護サービス
INDUSTRY_SPEALITY_BUSINESS_SERVICES	専門業務サービス
INDUSTRY_SPEALITY_MACHINERY	特殊産業機械
INDUSTRY_STUFFING_EMPLOYMENT	スタッフィングおよび雇用サービス
INDUSTRY_TOOLS_ACCESSORIES	ツールとアクセサリー
INDUSTRY_TRUCKING	トラック輸送
INDUSTRY_WASTE_MANAGEMENT	廃棄物管理
不動産	
INDUSTRY_REAL_ESTATE_DEVELOPMENT	不動産–開発
INDUSTRY_REAL_ESTATE_DIVERSIFIED	不動産–多様化
INDUSTRY_REAL_ESTATE_SERVICES	不動産サービス
INDUSTRY_REIT_DIVERSIFIED	REIT – 多様化
INDUSTRY_REIT_HEALTCARE	REIT – ヘルスケア施設
INDUSTRY_REIT_HOTEL_MOTEL	REIT – ホテル/宿泊施設
INDUSTRY_REIT_INDUSTRIAL	REIT – 産業
INDUSTRY_REIT_MORTAGE	REIT – 住宅ローン
INDUSTRY_REIT_OFFICE	REIT – オフィス
INDUSTRY_REIT_RESIDENTAL	REIT – 住宅
INDUSTRY_REIT_RETAIL	REIT – 小売り
INDUSTRY_REIT_SPECIALITY	REIT – 専門
技術	
INDUSTRY_COMMUNICATION_EQUIPMENT	通信機器
INDUSTRY_COMPUTER_HARDWARE	コンピューターハードウェア
INDUSTRY_CONSUMER_ELECTRONICS	家電
INDUSTRY_ELECTRONIC_COMPONENTS	電子部品
INDUSTRY_ELECTRONIC_DISTRIBUTION	エレクトロニクスとコンピュータの配布
INDUSTRY_IT_SERVICES	情報技術サービス
INDUSTRY_SCIENTIFIC_INSTRUMENTS	科学および技術機器
INDUSTRY_SEMICONDUCTOR_EQUIPMENT	半導体装置および材料

© 2000–2025, MetaQuotes Ltd.

ID	説明
INDUSTRY_SEMICONDUCTORS	半導体
INDUSTRY_SOFTWARE_APPLICATION	ソフトウェア－アプリケーション
INDUSTRY_SOFTWARE_INFRASTRUCTURE	ソフトウェア－インフラストラクチャ
INDUSTRY_SOLAR	ソーラー
ユーティリティ	
INDUSTRY_UTILITIES_DIVERSIFIED	ユーティリティ－多様化
INDUSTRY_UTILITIES_POWERPRODUCERS	ユーティリティ－独立した電力生産者
INDUSTRY_UTILITIES_RENEWABLE	ユーティリティ-再生可能
INDUSTRY_UTILITIES_REGULATED_ELECTRIC	ユーティリティ- 規制電気
INDUSTRY_UTILITIES_REGULATED_GAS	ユーティリティ- 規制ガス
INDUSTRY_UTILITIES_REGULATED_WATER	ユーティリティ－規制水
INDUSTRY_UTILITIES_FIRST	ユーティリティサービスタイプの列挙の開始。INDUSTRY_ UTILITIES_ DIVERSIFIEDに対応します。
INDUSTRY_UTILITIES_LAST	ユーティリティサービスタイプの列挙の終了。INDUSTRY_ UTILITIES_ REGULATED_ WATERに対応します。

© 2000–2025, MetaQuotes Ltd.

口座プロパティ

現在の口座に関する情報を取得するには AccountInfoInteger()、AccountInfoDouble() 及び AccountInfoString() の複数の関数があります。関数のパラメータは、対応するENUM_ACCOUNT_INFO の列挙からの値を受け入れることが出来ます。

AccountInfoInteger() 関数では

ENUM_ACCOUNT_INFO_INTEGER

識別子	説明	種類
ACCOUNT_LOGIN	口座番号。	long
ACCOUNT_TRADE_MODE	口座取引モード。	ENUM_ACCOUNT_TRADE_MODE
ACCOUNT_LEVERAGE	口座レバレッジ。	long
ACCOUNT_LIMIT_ORDERS	アクティブな未決注文の最大許容数。	int
ACCOUNT_MARGIN_SO_MODE	許容された最小証拠金を設定するモード。	ENUM_ACCOUNT_STOPOUT_MODE
ACCOUNT_TRADE_ALLOWED	現在の口座で許可された取引。	bool
ACCOUNT_TRADE_EXPERT	エキスパートアドバイザーで許可された取引。	bool
ACCOUNT_MARGIN_MODE	証拠金計算モード	ENUM_ACCOUNT_MARGIN_MODE

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
ACCOUNT_CURRENCY_DIGITS	取引結果を正確に表示するために必要な口座通貨の小数点以下の桁数	int
ACCOUNT_FIFO_CLOSE	FIFOルールによってのみポジションを決済できることを示します。プロパティ値がtrueに設定されている場合、各シンボルのポジションは、最も古いものから順に同じ順序で決済されます。別の注文でポジションを決済しようとした場合、適切なエラーが発生します。	bool

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
	ヘッジなしポジション会計モード(ACCOUNT_MARGIN_MODE!=ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)では、プロパティ値は常にfalseです。	

AccountInfoDouble() 関数では

ENUM_ACCOUNT_INFO_DOUBLE

識別子	説明	種類
ACCOUNT_BALANCE	預金通貨での勘定残高。	double
ACCOUNT_CREDIT	預金通貨での口座のクレジット。	double
ACCOUNT_PROFIT	預金通貨での現在の利益。	double
ACCOUNT_EQUITY	預金通貨の口座株式。	double
ACCOUNT_MARGIN	預金通貨での使用された口座の証拠金。	double

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
ACCOUNT_MARGIN_FREE	預金通貨での口座の余剰証拠金。	double
ACCOUNT_MARGIN_LEVEL	百分率で表した口座証拠金レベル。	double
ACCOUNT_MARGIN_SO_CALL	証拠金コールレベル。ACCOUNT_MARGIN_SO_MODE の設定によってパーセントまたは預金通貨で表されます。	double
ACCOUNT_MARGIN_SO_SO	証拠金ストップアウトレベル。ACCOUNT_MARGIN_SO_MODE の設定によってパーセントまたは預金通貨で表されます。	double
ACCOUNT_MARGIN_INITIAL	当初証拠金。全ての未決注文の証拠金をカバーするために口座内でリザーブされた額。	double
ACCOUNT_MARGIN_MAINTENANCE	維持証拠金。全ての未決済ポジションの最小額を	double

© 2000–2025, MetaQuotes Ltd.

識別子	説明	種類
	カバーするために口座内でリザーブされた最低資本金。	
ACCOUNT_ASSETS	口座の流動資産。	double
ACCOUNT_LIABILITIES	口座の流動負債。	double
ACCOUNT_COMMISSION_BLOCKED	口座の現在のブロックされた手数料の額。	double

AccountInfoString() 関数では

ENUM_ACCOUNT_INFO_STRING

識別子	説明	種類
ACCOUNT_NAME	クライアント名。	string
ACCOUNT_SERVER	取引サーバ名。	string
ACCOUNT_CURRENCY	口座通貨。	string
ACCOUNT_COMPANY	口座を提供する会社の名称。	string

取引サーバ上で開くことが出来る口座にはいくつかの種類があります。MQL5 が実行されている口座の種類は ENUM_ ACCOUNT_ TRADE_ MODE 列挙を使用して見つけることが出来ます。

ENUM_ACCOUNT_TRADE_MODE

識別子	説明
ACCOUNT_TRADE_MODE_DEMO	デモ口座。
ACCOUNT_TRADE_MODE_CONTEST	コンテスト口座。
ACCOUNT_TRADE_MODE_REAL	リアル口座。

© 2000–2025, MetaQuotes Ltd.

資本が保有中ポジションを維持するために不充分な場合には、ストップアウト、すなわち強制閉鎖が発生します。ストップアウトが起こる最小証拠金レベルは百分率または金銭で設定することが出来ます。口座がどちらのモードを使用するかは ENUM_ACCOUNT_STOPOUT_MODE 列挙で見つけられます。

ENUM_ACCOUNT_STOPOUT_MODE

識別子	説明
ACCOUNT_STOPOUT_MODE_PERCENT	口座ストップアウトの百分率モード。
ACCOUNT_STOPOUT_MODE_MONEY	口座ストップアウトの通貨モード。

ENUM_ACCOUNT_MARGIN_MODE

識別子	説明
ACCOUNT_MARGIN_MODE_RETAIL_NETTING	『ネッティング』モード（1つのシンボルに対し、1つのポジションのみ）におけるポジション計算時に市場外市場の為に使用されます。商品種別をベースに証拠金の計算が行われます(SYMBOL_TRADE_CALC_MODE)。
ACCOUNT_MARGIN_MODE_EXCHANGE	株式市場で使用されます。商品設定で指定された割引に基づいて証拠金が計算されます。割引はブローカーによって設定されますが、取引所が決めた額よりも低くなることはありません。
ACCOUNT_MARGIN_MODE_RETAIL_HEDGING	株式市場外の独立したポジション計算時（『ヘッジング』は一つのシンボルに対し複数のポジションを保有することができる）に使用されます。商品種別(SYMBOL_TRADE_CALC_MODE)に基づき、ヘッジ対象証拠金のサイズを考慮して(SYMBOL_MARGIN_HEDGED)証拠金の計算が行われます。

下記は簡単な口座情報を出力するスクリプトの例です。

```
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 会社名
   string company=AccountInfoString(ACCOUNT_COMPANY);
//--- クライアント名
   string name=AccountInfoString(ACCOUNT_NAME);
//--- 口座番号
   long login=AccountInfoInteger(ACCOUNT_LOGIN);
//--- サーバ名
   string server=AccountInfoString(ACCOUNT_SERVER);
//--- 口座通貨
   string currency=AccountInfoString(ACCOUNT_CURRENCY);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- デモ、コンテスト、またはリアル(実際の)口座
   ENUM_ACCOUNT_TRADE_MODE account_type=(ENUM_ACCOUNT_TRADE_MODE)
AccountInfoInteger(ACCOUNT_TRADE_MODE);
//--- 列挙値を理解出来る形式に変換する
   string trade_mode;
   switch(account_type)
     {
      case  ACCOUNT_TRADE_MODE_DEMO:
         trade_mode="demo";
         break;
      case  ACCOUNT_TRADE_MODE_CONTEST:
         trade_mode="contest";
         break;
      default:
         trade_mode="real";
         break;
     }
//--- ストップアウトは百分率または通貨で設定される
   ENUM_ACCOUNT_STOPOUT_MODE stop_out_mode=(ENUM_ACCOUNT_STOPOUT_MODE)
AccountInfoInteger(ACCOUNT_MARGIN_SO_MODE);
//--- 証拠金コールとストップアウトが発生した時のレベルの値を取得する
   double margin_call=AccountInfoDouble(ACCOUNT_MARGIN_SO_CALL);
   double stop_out=AccountInfoDouble(ACCOUNT_MARGIN_SO_SO);
//--- 簡単な口座情報を表示する
   PrintFormat("The account of the client '%s' #%d %s opened in '%s' on the server
'%s'",
               name,login,trade_mode,company,server);
   PrintFormat("Account currency - %s, MarginCall and StopOut levels are set in %s",
               currency,(stop_out_mode==ACCOUNT_STOPOUT_MODE_PERCENT)?"percentage":"
money");
   PrintFormat("MarginCall=%G, StopOut=%G",margin_call,stop_out);
  }
```

© 2000-2025, MetaQuotes Ltd.

テスト統計

テスト終了後には、取引結果統計のいろいろなパラメータが計算されます。パラメータ値は ENUM_STATISTICS 列挙からパラメータ ID を指定して TesterStatistics() 関数で取得されます。

統計の計算には 2 種類のパラメータ(int とdouble)が使用されますが、関数は全ての値を double で返します。double 型の統計値は、特に指定しない限りデフォルトでは預金通貨で表現されています。

ENUM_STATISTICS

ID	統計パラメータの説明	Type
STAT_INITIAL_DEPOSIT	頭金の値。	double
STAT_WITHDRAWAL	口座から引き落とされたお金。	double
STAT_PROFIT	テスト後の純利益で、STAT_GROSS_PROFIT と STAT_GROSS_LOSS の合計（STAT_GROSS_LOSS は常にゼロか負）。	double
STAT_GROSS_PROFIT	総利益（全ての収益性（正）の取引の合計）。値はゼロ以上です。	double
STAT_GROSS_LOSS	総損失（全ての負の取引の合計）。値はゼロ以下です。	double
STAT_MAX_PROFITTRADE	最大の利益 – 全ての勝ち取引の最大値。値はゼロ以上です。	double
STAT_MAX_LOSSTRADE	最大損失 – 全ての負け取引の最安値。値はゼロ以下です。	double
STAT_CONPROFITMAX	一連の勝ち取引での最大利益。値はゼロ以上です。	double
STAT_CONPROFITMAX_TRADES	STAT_CONPROFITMAX を形成している取引の数（一連の勝ち取引の内の、最大の利益）。	int
STAT_MAX_CONWINS	勝ち取引の最も長い一連の利益総額。	double
STAT_MAX_CONPROFIT_TRADES	勝ち取引の最長シリーズの取引回数	int

© 2000–2025, MetaQuotes Ltd.

ID	統計パラメータの説明	Type
	STAT_MAX_CONWINS。	
STAT_CONLOSSMAX	一連の負け取引での最大損失。値はゼロ以下です。	double
STAT_CONLOSSMAX_TRADES	STAT_CONLOSSMAX を形成している取引の数（一連の負け取引での最大損失）。	int
STAT_MAX_CONLOSSES	負け取引の最長シリーズでの損失総額。	double
STAT_MAX_CONLOSS_TRADES	負け取引の最長シリーズの取引回数（STAT_MAX_CONLOSSES）。	int
STAT_BALANCEMIN	最低残高値。	double
STAT_BALANCE_DD	金額での最大残高ドローダウン。取引過程では多数の残高ドローダウンがあります。ここで最大値が取得されます。	double
STAT_BALANCEDD_PERCENT	最大ドローダウンバランスの瞬間に記録された金額の百分率でのバランスドローダウン（STAT_BALANCE_DD）。	double
STAT_BALANCE_DDREL_PERCENT	最大残高ドローダウン（%）。取引過程では多数のドローダウンがあり、各ドローダウンで百分率での相対ドローダウン値が算出されます。最高値が返されます。	double
STAT_BALANCE_DD_RELATIVE	最高残高ドローダウンの際に記録された、金額の百分率で表された残高ドローダウン（STAT_BALANCE_DDREL_PERCENT）。	double
STAT_EQUITYMIN	最低株式値。	double
STAT_EQUITY_DD	金額での最大株式ドローダウン。取引過程で	double

© 2000–2025, MetaQuotes Ltd.

ID	統計パラメータの説明	Type
	は多数のドローダウンがあります。ここで最大値が取得されます。	
STAT_EQUITYDD_PERCENT	最高残高ドローダウンの際に記録された、金額の百分率で表された残高ドローダウン（STAT_EQUITY_DD）。	double
STAT_EQUITY_DDREL_PERCENT	最大株式ドローダウン（%）。取引過程では多数の株式ドローダウンがあり、各ドローダウンで百分率での相対ドローダウン値が算出されます。最高値が返されます。	double
STAT_EQUITY_DD_RELATIVE	最高株式ドローダウンの際に記録された、金額の百分率で表された株式ドローダウン（STAT_EQUITY_DDREL_PERCENT）。	double
STAT_EXPECTED_PAYOFF	期待利得。	double
STAT_PROFIT_FACTOR	STAT_GROSS_PROFIT/STAT_GROSS_LOSS の比に等しい利益率。STAT_GROSS_LOSS=0 の場合、利益率は DBL_MAX。	double
STAT_RECOVERY_FACTOR	STAT_PROFIT/STAT_BALANCE_DD の比に等しい回復率。	double
STAT_SHARPE_RATIO	シャープレシオ。	double
STAT_MIN_MARGINLEVEL	証拠金レベルの最小値。	double
STAT_CUSTOM_ONTESTER	OnTester() 関数が返したカスタムの最適化基準の計算値。	double
STAT_DEALS	約定数。	int
STAT_TRADES	取引数。	int
STAT_PROFIT_TRADES	勝ち取引。	int

© 2000–2025, MetaQuotes Ltd.

ID	統計パラメータの説明	Type
STAT_LOSS_TRADES	負け取引。	int
STAT_SHORT_TRADES	短期取引。	int
STAT_LONG_TRADES	長期取引。	int
STAT_PROFIT_SHORTTRADES	収益性の高い短期取引。	int
STAT_PROFIT_LONGTRADES	収益性の高い長期取引。	int
STAT_PROFITTRADES_AVGCON	一連の収益性の高い取引の平均の長さ。	int
STAT_LOSSTRADES_AVGCON	一連の負け取引の平均の長さ。	int
STAT_COMPLEX_CRITERION	複雑な最適化基準	

© 2000–2025, MetaQuotes Ltd.

取引定数

取引ストラテジープログラミングのために使用される各種定数は、次のグループに分類されます。

- 履歴データベースプロパティ – シンボルの一般的な情報の受け取り
- 注文プロパティ – 取引注文の情報の受け取り
- ポジションプロパティ – 現在ポジションの情報の受け取り
- 約定プロパティ – 取引の情報の受け取り
- 取引操作の種類 – 利用可能な取引操作の記述
- 取引トランザクションの種類 – 利用可能な取引トランザクションの記述
- DOM での取引注文 – リクエストされた操作の方向に応じての注文の分離

© 2000–2025, MetaQuotes Ltd.

履歴データベースプロパティ

時系列アクセスでは追加のシンボル情報取得に SeriesInfoInteger() が使用されます。必要なプロパティの識別子は、関数のパラメータとして渡されます。識別子は ENUM_SERIES_INFO_INTEGER の値のいずれかになります。

ENUM_SERIES_INFO_INTEGER

識別子	説明	Type
SERIES_BARS_COUNT	現在のシンボル期間のバーの数。	long
SERIES_FIRSTDATE	現在のシンボル期間の初日。	datetime
SERIES_LASTBAR_DATE	現在のシンボル期間の最終バーの開始時刻。	datetime
SERIES_SERVER_FIRSTDATE	時間軸に関係ない、サーバ上でのシンボル履歴の最初の日付。	datetime
SERIES_TERMINAL_FIRSTDATE	時間軸に関係ない、クライアント端末でのシンボル履歴の最初の日付。	datetime
SERIES_SYNCHRONIZED	現時点の銘柄/期間のデータ同期フラグ。	bool

© 2000–2025, MetaQuotes Ltd.

注文プロパティ

取引操作の実行リクエストは、注文として定式化されます。各注文は、読み込みのための様々なプロパティを持っています。それらの情報は OrderGet...() 及び HistoryOrderGet...() 関数で得られます。

OrderGetInteger() 及び HistoryOrderGetInteger() 関数

ENUM_ORDER_PROPERTY_INTEGER

識別子	説明	Type
ORDER_TICKET	注文チケット。各注文に割り当てられる固有番号。	long
ORDER_TIME_SETUP	注文設定時刻。	datetime
ORDER_TYPE	注文の種類。	ENUM_ORDER_TYPE
ORDER_STATE	注文状態。	ENUM_ORDER_STATE
ORDER_TIME_EXPIRATION	注文の期限。	datetime
ORDER_TIME_DONE	注文の実行及びキャンセル時刻。	datetime
ORDER_TIME_SETUP_MSC	01.01.1970 から経過したミリ秒数で表された注文の実行が出された時刻。	long
ORDER_TIME_DONE_MSC	01.01.1970 から経ったミリ秒で表された注文の実行/キャンセル時刻。	long
ORDER_TYPE_FILLING	注文充填タイプ。	ENUM_ORDER_TYPE_FILLING
ORDER_TYPE_TIME	注文ライフタイム。	ENUM_ORDER_TYPE_TIME
ORDER_MAGIC	注文を出したエキスパートアドバイザーのID（各エキスパートアドバイザーは、独自のユニークな番号	long

© 2000–2025, MetaQuotes Ltd.

識別子	説明	Type
	を作成するように設計されています）。	
ORDER_REASON	注文の理由またはソース。	ENUM_ORDER_REASON
ORDER_POSITION_ID	実行後すぐに注文に設定されるポジション識別。注文の実行は注文を出すか、既存のポジション変更する約定となります。このポジションの識別子がこの時点で実行される注文のために設定されています。	long
ORDER_POSITION_BY_ID	ORDER_TYPE_CLOSE_BY型の注文の為の反対ポジションの識別子。	long

OrderGetDouble() 及び HistoryOrderGetDouble() 関数

ENUM_ORDER_PROPERTY_DOUBLE

識別子	説明	Type
ORDER_VOLUME_INITIAL	注文の初期ボリューム。	double
ORDER_VOLUME_CURRENT	注文の現在ボリューム。	double
ORDER_PRICE_OPEN	注文で指定された価格。	double
ORDER_SL	決済逆指値。	double
ORDER_TP	決済指値。	double
ORDER_PRICE_CURRENT	注文シンボルの現在の価格。	double

© 2000–2025, MetaQuotes Ltd.

識別子	説明	Type
ORDER_PRICE_STOPLIMIT	ストップリミット注文の指値注文価格。	double

OrderGetString() 及び HistoryOrderGetString() 関数

ENUM_ORDER_PROPERTY_STRING

識別子	説明	Type
ORDER_SYMBOL	注文シンボル。	string
ORDER_COMMENT	注文コメント。	string

OrderSend() 関数を使用して取引リクエストを送る場合、操作によっては注文の種類の指定が必要です。注文の種類は特別な構造体 MqlTradeRequestの *type* フィールドに含まれ、ENUM_ORDER_TYPE 列挙の値が使用出来ます。

ENUM_ORDER_TYPE

識別子	説明
ORDER_TYPE_BUY	成行買い注文。
ORDER_TYPE_SELL	成行売り注文。
ORDER_TYPE_BUY_LIMIT	買い指値注文。
ORDER_TYPE_SELL_LIMIT	売り指値注文。
ORDER_TYPE_BUY_STOP	買い逆指値注文。
ORDER_TYPE_SELL_STOP	売り逆指値注文。
ORDER_TYPE_BUY_STOP_LIMIT	注文価格に達すると、未決の買い指値注文はストップリミット価格で出されます。
ORDER_TYPE_SELL_STOP_LIMIT	注文価格に達すると、未決の売り指値注文はストップリミット価格で出されます。
ORDER_TYPE_CLOSE_BY	ポジションを反対のポジションで決済するための注文。

各注文には、その状態を説明するステータスがあります。情報を取得するには ORDER_STATE 修飾子と OrderGetInteger() または HistoryOrderGetInteger() 関数を使用します。使用可能な値は ENUM_ORDER_STATE 列挙に記載されています。

ENUM_ORDER_STATE

© 2000–2025, MetaQuotes Ltd.

識別子	説明
ORDER_STATE_STARTED	注文がチェック済み、ブローカによっての受け入れがまだ。
ORDER_STATE_PLACED	受付済みの注文。
ORDER_STATE_CANCELED	クライアントによってキャンセルされた注文。
ORDER_STATE_PARTIAL	実行中の注文。
ORDER_STATE_FILLED	実行済みの注文。
ORDER_STATE_REJECTED	拒絶された注文。
ORDER_STATE_EXPIRED	期限切れの注文。
ORDER_STATE_REQUEST_ADD	注文が登録中（取引システムに配置）。
ORDER_STATE_REQUEST_MODIFY	注文が更新中（パラメータの変更）。
ORDER_STATE_REQUEST_CANCEL	注文が削除中（取引システムからの削除）。

現在の時刻（有効時間）で実行のための取引リクエストを送信するとき、価格と必要な売買量を指定する必要があります。また、金融市場では、リクエストされたボリューム全体が特定の金融商品で希望の価格で利用できるという保証はありません。したがって、リアルタイムの取引操作は、価格およびボリューム実行モードを使用して規制されます。モードまたは実行ポリシーは、価格が変更された場合、またはリクエストされたボリュームを現時点で完全に満たすことができない場合のルールを定義します。

価格実行モードは、ENUM_SYMBOL_TRADE_EXECUTION列挙のフラグの組み合わせを含むSYMBOL_TRADE_EXEMODE銘柄プロパティから取得できます。

実行モード	説明	ENUM_SYMBOL_TRADE_EXECUTIONの値
実行モード （リクエスト実行）	以前に証券会社から受け取った価格で成行注文を実行します。 特定の成行注文の価格は、注文が送信される前に証券会社に	SYMBOL_TRADE_EXECUTION_REQUEST

© 2000–2025, MetaQuotes Ltd.

実行モード	説明	ENUM_SYMBOL_TRADE_EXECUTIONの値
	リクエストされます。価格を受け取ると、指定された価格での注文の実行を確認または拒否できます。	
即時実行 （即時実行）	指定された価格ですぐに成行注文を実行します。 実行する取引リクエストを送信すると、プラットフォームは自動的に現在の価格を注文に追加します。 • 証券会社が価格を受	SYMBOL_TRADE_EXECUTION_INSTANT

© 2000–2025, MetaQuotes Ltd.

実行モード	説明	ENUM_SYMBOL_TRADE_EXECUTIONの値
	け入れると、注文が実行されます。 • 証券会社がリクエストされた価格を受け入れない場合、「リクオート」が	

© 2000–2025, MetaQuotes Ltd.

実行モード	説明	ENUM_SYMBOL_TRADE_EXECUTIONの値
	送信され、証券会社はこの注文を実行できる価格を返します。	
成行執行 （成行執行）	証券会社は、トレーダーとの追加の話し合いなしに、注文執行価格について決定を下します。 このようなモードで注	SYMBOL_TRADE_EXECUTION_MARKET

© 2000–2025, MetaQuotes Ltd.

実行モード	説明	ENUM_SYMBOL_TRADE_EXECUTIONの値
	文を送信することは、この価格での実行に事前に同意することを意味します。	
取引所実行 （取引所実行）	取引操作は、現在の市場オファーの価格で実行されます。	SYMBOL_TRADE_EXECUTION_EXCHANGE

ボリューム実行ポリシーはORDER_TYPE_FILLING注文プロパティで指定され、
ENUM_ORDER_TYPE_FILLING列挙からの値のみを含めることができます

実行ポリシー	説明	ENUM_ORDER_TYPE_FILLINGの値
FOK	注文は指定されたボリュームでのみ実行できます。 必要な量の金融商品が現在市場で入手できない場合、注文は実行され	ORDER_FILLING_FOK

© 2000–2025, MetaQuotes Ltd.

実行ポリシー	説明	ENUM_ORDER_TYPE_FILLINGの値
	ません。 必要なボリュームは、いくつかの利用可能なオファーで構成できます。 FOK注文を使用する可能性は、取引サーバで決定されます。	
即時またはキャンセル	トレーダーは、注文に示された量の範囲内で市場で最大限に利用可能な量で取引を実行することに同意します。 リクエストを完全に満たすこ	ORDER_FILLING_IOC

© 2000–2025, MetaQuotes Ltd.

実行ポリシー	説明	ENUM_ORDER_TYPE_FILLINGの値
	とができない場合、利用可能なボリュームでの注文が実行され、残りのボリュームはキャンセルされます。 IOC注文を使用する可能性は、取引サーバで決定されます。	
パッシブ(BOC)	BoC注文は、注文が板情報でのみ発注でき、すぐに実行できないことを前提としています。発注時にすぐに約定できる場	ORDER_FILLING_BOC

© 2000-2025, MetaQuotes Ltd.

実行ポリシー	説明	ENUM_ORDER_TYPE_FILLINGの値
	合、注文はキャンセルされます。 実際、BOCのポリシーは、発注された注文の価格が現在の市場よりも悪くなることを保証しています。 BoC注文はパッシブ取引を実装するために使用されるため、注文が出されてもすぐには実行されず、現在の流動性には影響しません。 指値注文と逆指値注	

© 2000–2025, MetaQuotes Ltd.

実行ポリシー	説明	ENUM_ORDER_TYPE_FILLINGの値
	文のみがサポートされています(ORDER_TYPE_BUY_LIMIT、ORDER_TYPE_SELL_LIMIT、ORDER_TYPE_BUY_STOP_LIMIT、ORDER_TYPE_SELL_STOP_LIMIT)。	
リターン	部分的な実行の場合、残りのボリュームのある注文はキャンセルされず、さらに処理されます。 成行実行モード(成行実行ー	ORDER_FILLING_RETURN

© 2000-2025, MetaQuotes Ltd.

実行ポリシー	説明	ENUM_ORDER_TYPE_FILLINGの値
	SYMBOL_TRADE_EXECUTION_MARKET)では、リターン注文は許可されていません。	

OrderSend()関数を使用して取引リクエストを送信する場合、必要なボリューム実行ポリシーを*type_filling*フィールド(特別なMqlTradeRequest構造体)で設定できます。 ENUM_ORDER_TYPE_FILLING列挙からの値が利用可能です。特定のアクティブ/完了した注文でプロパティ値を取得するには、OrderGetInteger()またはHistoryOrderGetInteger()関数とORDER_TYPE_FILLING修飾子を使用します。

現在の実行時間で注文を送信する前に、ORDER_TYPE_FILLING値(ボリューム実行タイプ)を正しく設定するには、各金融商品でSYMBOL_FILLING_MODEプロパティ値を取得するのにSymbolInfoInteger()関数を使用できます。これは銘柄で許可されたボリューム実行タイプをフラグの組み合わせとして示します。ORDER_FILLING_RETURN注文タイプは、「成行実行」モード(SYMBOL_TRADE_EXECUTION_MARKET)を除いて常に有効です。

実行モードに応じた注文タイプの使用法を次の表に示します。

実行の種類¥実行ポリシー	FOK(FOK ORDER_FILLING_FOK)	即時またはキャンセル(IOC ORDER_FILLING_IOC)	リターン(リターン ORDER_FILLING_RETURN)
即時実行 (SYMBOL_TRADE_EXECUTION_INSTANT)	+(銘柄の設定に関係なく)	+(銘柄の設定に関係なく)	+(常に)
リクエスト実行 SYMBOL_TRADE_EXECUTION_REQUEST	+(銘柄の設定に関係なく)	+(銘柄の設定に関係なく)	+(常に)
成行執行 SYMBOL_TRADE_EXECUTION_MARKET	+(銘柄設定で設定)	+(銘柄設定で設定)	-(銘柄設定に関係なく無効)
取引所実行 SYMBOL_TRADE_EXECUTION_EXCHANGE	+(銘柄設定で設定)	+(銘柄設定で設定)	+(常に)

未決注文の場合、そのような注文は送信時点での実行を目的としていないため、実行タイプに関係なくORDER_FILLING_RETURN注文タイプ(SYMBOL_TRADE_EXEMODE)を使用する必要があります。未決注文

© 2000-2025, MetaQuotes Ltd.

を使用する場合、トレーダーは、この注文の取引条件が満たされたときに、証券会社が取引所でサポートされている注文タイプを使用することに事前に同意します。

OrderSend() 関数で取引リクエストを送る際に注文の有効期間は特別な構造体 MqlTradeRequest の *type_time* フィールドで設定できます。ENUM_ORDER_TYPE_TIME 列挙値が使用されます。このプロパティの値を取得するには OrderGetInteger() または HistoryOrderGetInteger() 関数を ORDER_TYPE_TIME 修飾子と使用します。

ENUM_ORDER_TYPE_TIME

識別子	説明
ORDER_TIME_GTC	GTC注文（キャンセルするまで有効）。
ORDER_TIME_DAY	現在の取引日のみ有効である注文。
ORDER_TIME_SPECIFIED	有効期限まで有効な注文。
ORDER_TIME_SPECIFIED_DAY	注文が指定された日の 23:59:59 まで有効となります。この時刻が取引セッション外である場合は、注文は最も近い取引時間中に満了します。

ORDER_REASONプロパティには注文を出した理由が含まれます。注文は、MQL5プログラムやモバイルアプリケーションから出されたりストップアウトなどの結果として出されたりします。ORDER_REASONの可能な値は、ENUM_ORDER_REASON列挙体に記述されています。

ENUM_ORDER_REASON

識別子	説明
ORDER_REASON_CLIENT	注文はデスクトップターミナルから出された
ORDER_REASON_MOBILE	注文はモバイルアプリケーションから出された
ORDER_REASON_WEB	注文はwebプラットフォームから出された
ORDER_REASON_EXPERT	注文はMQL5プログラム（エキスパートアドバイザーまたはスクリプト）から出された
ORDER_REASON_SL	注文はストップロスのアクティブ化の結果として出された
ORDER_REASON_TP	注文はテイクプロフィットのアクティブ化の結果として出された
ORDER_REASON_SO	注文はストップアウトイベントの結果として出された

© 2000–2025, MetaQuotes Ltd.

ポジションプロパティ

取引操作の実行は、注文を出すこと、ボリューム及び/または方向の変更、または解除につながります。取引操作は OrderSend() 関数で取引リクエストとして送られた注文を通して行われます。各有価証券（シンボル）では、ポジションは 1 つだけ可能です。ポジションには PositionGet...() 関数での読み込みが可能なプロパティセットがあります。

PositionGetInteger() 関数では

ENUM_POSITION_PROPERTY_INTEGER

識別子	説明	Type
POSITION_TICKET	ポジションチケット。各保有ポジションに新たに割り当てられた固有の番号。通常、その結果としてポジションを保有する、注文のチケットに対応します（サーバー上で行われた操作の結果、チケットが変更された場合を除く）。例えば、ポジションの再保有によるスワップの加算です。ポジションを保有することになった注文を見つけるには、POSITION_IDENTIFIERプロパティを使用する必要があります。 POSITION_TICKET値は MqlTradeRequest::positionと一致します。	long
POSITION_TIME	注文が出された時刻。	datetime
POSITION_TIME_MSC	01.01.1970 から経ったミリ秒で表された注文が出された時刻。	long

© 2000-2025, MetaQuotes Ltd.

識別子	説明	Type
POSITION_TIME_UPDATE	01.01.1970 から経った秒数で表されたポジション変更時刻。	datetime
POSITION_TIME_UPDATE_MSC	01.01.1970 から経ったミリ秒で表されたポジション変更時刻。	long
POSITION_TYPE	ポジションの種類。	ENUM_POSITION_TYPE
POSITION_MAGIC	ポジションマジックナンバー（ORDER_MAGICを参照）。	long
POSITION_IDENTIFIER	ポジション識別子は、全ての新しくオープンしたポジションに割り当てられ、ポジションのライフタイムに一貫する固有の番号です。識別子はポジションのターンオーバによって変更されません。ポジション識別子は、ポジションを開く、変更する、または決済するために使用される各注文（ORDER_POSITION_ID）と約定（DEAL_POSITION_ID）で指定されています。ポジションに関連した注文や約定を検索するにはこのプロパティを使用します。ネッティングモード（単一のイン/アウト取引を使用）でポジションが反転された場合、POSITION_IDENTIFIERは変更されま	long

© 2000-2025, MetaQuotes Ltd.

識別子	説明	Type
	せん。 ただし、POSITION_TICKETは、その反転につながった注文のチケットに置き換えられます。ポジション反転はヘッジモードでは提供されません。	
POSITION_REASON	ポジションを開く理由。	ENUM_POSITION_REASON

PositionGetDouble() 関数

ENUM_POSITION_PROPERTY_DOUBLE

識別子	説明	Type
POSITION_VOLUME	ポジションボリューム。	double
POSITION_PRICE_OPEN	ポジションの始値。	double
POSITION_SL	未決済ポジションの決済逆指レベル。	double
POSITION_TP	未決済ポジションの決済指値。	double
POSITION_PRICE_CURRENT	ポジションシンボルの現在価格。	double
POSITION_SWAP	累積スワップ。	double
POSITION_PROFIT	現在の利益。	double

PositionGetString() 関数では

ENUM_POSITION_PROPERTY_STRING

識別子	説明	Type
POSITION_SYMBOL	ポジションのシンボル。	string
POSITION_COMMENT	ポジションコメント。	string
POSITION_EXTERNAL_ID	ポジション識別子が外部取引システム(取引所)です。	string

© 2000–2025, MetaQuotes Ltd.

ポジションの方向（売買）は ENUM_POSITION_TYPE 列挙から値によって定義されています。ポジションの種類を取得するには PositionGetInteger() 関数を POSITION_TYPE 修飾子と使用します。

ENUM_POSITION_TYPE

識別子	説明
POSITION_TYPE_BUY	買い。
POSITION_TYPE_SELL	売り。

POSITION_REASONプロパティにはポジションを開いた理由が含まれます。ポジションは、デスクトップターミナル、モバイルアプリケーション、エキスパートアドバイザーなどから出された注文のアクティブ化の結果として開かれます。POSITION_REASONの可能な値は、ENUM_POSITION_REASON列挙体に記述されています。

ENUM_POSITION_REASON

識別子	説明
POSITION_REASON_CLIENT	ポジションは、デスクトップターミナルから出された注文のアクティブ化の結果として開かれた
POSITION_REASON_MOBILE	ポジションは、モバイルアプリケーションから出された注文のアクティブ化の結果として開かれた
POSITION_REASON_WEB	ポジションは、webプラットフォームから出された注文のアクティブ化の結果として開かれた
POSITION_REASON_EXPERT	ポジションは、MQL5プログラム（エキスパートアドバイザーまたはスクリプト）から出された注文のアクティブ化の結果として開かれた

© 2000–2025, MetaQuotes Ltd.

約定プロパティ

約定は取引リクエストを含む注文に基づいた取引操作の実行の事実を反映しています。 各取引は、関係した情報取得を可能にするプロパティによって記述されます。プロパティ値の読み込みには対応する列挙からの値を返す HistoryDealGet...() 型の関数が使用されます。

HistoryDealGetInteger() 関数

ENUM_DEAL_PROPERTY_INTEGER

識別子	説明	Type
DEAL_TICKET	取引チケット。各取引に割り当てられる固有の番号です。	long
DEAL_ORDER	約定の注文番号。	long
DEAL_TIME	約定時刻。	datetime
DEAL_TIME_MSC	01.01.1970 から経過したミリ秒数で表された約定実行時刻。	long
DEAL_TYPE	約定の種類。	ENUM_DEAL_TYPE
DEAL_ENTRY	約定エントリー（エントリーイン、エントリーアウト、リバース）。	ENUM_DEAL_ENTRY
DEAL_MAGIC	約定マジックナンバー（ORDER_MAGIC を参照）。	long
DEAL_REASON	約定実行のソースまたは理由。	ENUM_DEAL_REASON
DEAL_POSITION_ID	約定の開始、変更や変化が起きたポジションのポジション識別子。ポジションは、ライフタイムを通して、シンボルに実行された全ての約定に割り当てられている固有の識別子を持ちます。	long

HistoryDealGetDouble()関数には

ENUM_DEAL_PROPERTY_DOUBLE

識別子	説明	Type
DEAL_VOLUME	約定量。	double
DEAL_PRICE	約定値。	double
DEAL_COMMISSION	約定手数料。	double
DEAL_SWAP	終了時の累積スワップ。	double
DEAL_PROFIT	約定の利益。	double
DEAL_FEE	約定直後に請求される約定料金	double
DEAL_SL	決済逆指値レベル	double

© 2000–2025, MetaQuotes Ltd.

識別子	説明	Type
	• エントリおよびリバースの約定は、ポジションが開かれたまたは反転された元の注文の決済逆指値を使用します。 • エグジットの約定はポジション決済時のようにポジションの決済逆指値を使用します。	
DEAL_TP	決済指値レベル • エントリおよびリバースの約定は、ポジションが開かれたまたは反転された元の注文の済逆指値を使用します。 • エグジットの約定はポジション決済時のようにポジションの決済逆指値を使用します。	double

HistoryDealGetString() 関数には

ENUM_DEAL_PROPERTY_STRING

識別子	説明	Type
DEAL_SYMBOL	約定シンボル。	string
DEAL_COMMENT	約定コメント。	string

各 約定は種類によって特徴づけされ、可能な値は ENUM_DEAL_TYPE にみられます。約定の種類について情報を取得するには HistoryDealGetInteger() を DEAL_TYPE 修飾子で使用します。

ENUM_DEAL_TYPE

識別子	説明
DEAL_TYPE_BUY	買い。
DEAL_TYPE_SELL	売り。
DEAL_TYPE_BALANCE	残高。
DEAL_TYPE_CREDIT	クレジット。
DEAL_TYPE_CHARGE	追加料金。
DEAL_TYPE_CORRECTION	修正。
DEAL_TYPE_BONUS	ボーナス。
DEAL_TYPE_COMMISSION	追加手数料。
DEAL_TYPE_COMMISSION_DAILY	日ごとの手数料。

© 2000–2025, MetaQuotes Ltd.

識別子	説明
DEAL_TYPE_COMMISSION_MONTHLY	月ごとの手数料。
DEAL_TYPE_COMMISSION_AGENT_DAILY	日ごとの仲介手数料。
DEAL_TYPE_COMMISSION_AGENT_MONTHLY	月ごとの仲介手数料。
DEAL_TYPE_INTEREST	金利。
DEAL_TYPE_BUY_CANCELED	取り消しされた買い。以前に実行された買い約定が取り消しされる場合があります。この場合には、以前に実行された約定(DEAL_TYPE_BUY)が DEAL_TYPE_BUY_CANCELED に変更され、その利益/損失はゼロ化されます。先に取得された利益/損失は、別の残高操作で実行されます。
DEAL_TYPE_SELL_CANCELED	取り消しされた売り。以前に実行された売り約定が取り消しされる場合があります。この場合には、以前に実行された約定(DEAL_TYPE_SELL)が DEAL_TYPE_SELL_CANCELED に変更され、その利益/損失はゼロ化されます。先に取得された利益/損失は、別の残高操作で実行されます。
DEAL_DIVIDEND	配当操作
DEAL_DIVIDEND_FRANKED	フランク(非課税)配当操作
DEAL_TAX	税

約定は ENUM_DEAL_TYPE に設定された種類のみではなく、ポジション変更方法においても異なります。これは単純なポジションオープン、または既にオープンされたポジションの蓄積(市場に参入)、同じボリュームの反対方向の約定によるポジション決済(市場から退出)、または、反対方向約定が建玉の蓄積をカバーする場合のポジションの逆転になりえます。

これらの状況の全ては ENUM_DEAL_ENTRY 列挙からの値によって記述されています。約定について情報を取得するには HistoryDealGetInteger()関数を DEAL_ENTRY 修飾子で使用します。

ENUM_DEAL_ENTRY

識別子	説明
DEAL_ENTRY_IN	エントリーイン。
DEAL_ENTRY_OUT	エントリーアウト。
DEAL_ENTRY_INOUT	リバース。
DEAL_ENTRY_OUT_BY	反対ポジションの決済。

DEAL_REASONプロパティには約定実行の理由が含まれます。約定は、モバイルアプリケーションまたはMQL5プログラムから出された注文のトリガ、ストップアウトイベント、変動証拠金計算の結果として実行されます。DEAL_REASONの可能な値は、ENUM_DEAL_REASON列挙体に記述されています。バランス、クレジット、手数料などの取引以外の約定理由はDEAL_REASON_CLIENTとして表示されます。

© 2000–2025, MetaQuotes Ltd.

ENUM_DEAL_REASON

識別子	説明
DEAL_REASON_CLIENT	約定は、デスクトップターミナルから出された注文のアクティブ化の結果として実行された
DEAL_REASON_MOBILE	約定は、モバイルアプリケーションから出された注文のアクティブ化の結果として実行された
DEAL_REASON_WEB	約定は、webプラットフォームから出された注文のアクティブ化の結果として実行された
DEAL_REASON_EXPERT	約定は、MQL5プログラム（エキスパートアドバイザーまたはスクリプト）から出された注文のアクティブ化の結果として実行された
DEAL_REASON_SL	約定は、ストップロスのアクティブ化の結果として実行された
DEAL_REASON_TP	約定は、テイクプロフィットのアクティブ化の結果として実行された
DEAL_REASON_SO	約定は、ストップアウトイベントの結果として実行された
DEAL_REASON_ROLLOVER	約定は、ロールオーバーの結果として実行された
DEAL_REASON_VMARGIN	約定は、変動証拠金の課金の後で実行された
DEAL_REASON_SPLIT	約定は、スプリット（価格低下）の発表時にポジションを持っていた製品のスプリットによって実行された
DEAL_REASON_CORPORATE_ACTION	この取引は、証券の統合または名前変更、顧客の別の口座への移管などの企業行動の結果として実行されました。

© 2000–2025, MetaQuotes Ltd.

取引操作の種類

取引は OrderSend() 関数を使用して建玉に注文を送ったり、未決注文を配置、変更、または解除することです。各取引注文はリクエストされた操作の種類を示します。取引操作は ENUM_TRADE_REQUEST_ACTIONS 列挙で記述されます。

ENUM_TRADE_REQUEST_ACTIONS

識別子	説明
TRADE_ACTION_DEAL	指定されたパラメータ(成行注文)の即時実行のために約定注文を出します。
TRADE_ACTION_PENDING	指定された条件(未決注文)で実行するために取引注文を出します。
TRADE_ACTION_SLTP	保有中ポジションの決済逆指値及び決済指値を変更します。
TRADE_ACTION_MODIFY	以前の注文のパラメータを変更します。
TRADE_ACTION_REMOVE	以前の未決注文を削除します。
TRADE_ACTION_CLOSE_BY	反対ポジションの決済。

買いポジションを持つ 取引操作TRADE_ACTION_DEALの例:

```
#define EXPERT_MAGIC 123456   // エキスパートアドバイザのMagicNumber
//+------------------------------------------------------------------+
//|  買いポジションの発注                                            |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- リクエストと結果の宣言と初期化
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- リクエストのパラメータ
   request.action   =TRADE_ACTION_DEAL;                     // 取引操作タイプ
   request.symbol   =Symbol();                              // シンボル
   request.volume   =0.1;                                   // 0.1ロットのボリューム
   request.type     =ORDER_TYPE_BUY;                        // 注文タイプ
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_ASK); // 発注価格
   request.deviation=5;                                     // 価格からの許容偏差
   request.magic    =EXPERT_MAGIC;                          // 注文のMagicNumber
//--- リクエストの送信
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());     // リクエストの送信が失敗した場
合、エラーコードを出力する
//--- 操作に関する情報
   PrintFormat("retcode=%u  deal=%I64u  order=%
I64u",result.retcode,result.deal,result.order);
  }
//+------------------------------------------------------------------+
```

売りポジションを持つ 取引操作TRADE_ACTION_DEALの例:

© 2000-2025, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456   // エキスパートアドバイザのMagicNumber
//+------------------------------------------------------------------+
//| 売りポジションを持つ                                             |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- リクエストと結果の宣言と初期化
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- リクエストのパラメータ
   request.action   =TRADE_ACTION_DEAL;                     // 取引操作タイプ
   request.symbol   =Symbol();                              // シンボル
   request.volume   =0.2;                                   // 0.2ロットのボリューム
   request.type     =ORDER_TYPE_SELL;                       // 注文タイプ
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_BID); // 発注価格
   request.deviation=5;                                     // 価格からの許容偏差
   request.magic    =EXPERT_MAGIC;                          // 注文のMagicNumber
//--- リクエストの送信
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());     // リクエストの送信が失敗した場
合、エラーコードを出力する
//--- 操作についての情報
   PrintFormat("retcode=%u  deal=%I64u  order=%
I64u",result.retcode,result.deal,result.order);
  }
//+------------------------------------------------------------------+
```

ポジションを決済する取引操作TRADE_ACTION_DEALの例：

© 2000-2025, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456   // エキスパートアドバイザのMagicNumber
//+------------------------------------------------------------------+
//| 全てのポジションを決済                                           |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 結果とリクエストの宣言
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // 保有ポジション数
//--- 全ての保有ポジションの取捨
   for(int i=total-1; i>=0; i--)
     {
      //--- 注文のパラメータ
      ulong
 position_ticket=PositionGetTicket(i);                                  // ポジショ
ンチケット
      string
position_symbol=PositionGetString(POSITION_SYMBOL);                    // シンボ
ル
      int    digits=(int)
SymbolInfoInteger(position_symbol,SYMBOL_DIGITS);          // 小数点以下の桁数
      ulong
 magic=PositionGetInteger(POSITION_MAGIC);                              // ポジショ
ンのMagicNumber
      double
volume=PositionGetDouble(POSITION_VOLUME);                             // ポジショ
ンボリューム
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)
PositionGetInteger(POSITION_TYPE);    // ポジションタイプ
      //--- ポジション情報の出力
      PrintFormat("#%I64u %s  %s  %.2f  %s [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  magic);
      //--- MagicNumberが一致している場合
      if(magic==EXPERT_MAGIC)
        {
         //--- リクエストと結果の値のゼロ化
         ZeroMemory(request);
         ZeroMemory(result);
         //--- 操作パラメータの設定
         request.action   =TRADE_ACTION_DEAL;        // 取引操作タイプ
         request.position =position_ticket;          // ポジションチケット
         request.symbol   =position_symbol;          // シンボル
         request.volume   =volume;                   // ポジションボリューム
         request.deviation=5;                        // 価格からの許容偏差
         request.magic    =EXPERT_MAGIC;             // ポジションのMagicNumber
         //--- ポジションタイプによる注文タイプと価格の設定
         if(type==POSITION_TYPE_BUY)
           {
            request.price=SymbolInfoDouble(position_symbol,SYMBOL_BID);
            request.type =ORDER_TYPE_SELL;
           }
         else
           {
            request.price=SymbolInfoDouble(position_symbol,SYMBOL_ASK);
            request.type =ORDER_TYPE_BUY;
```

© 2000-2025, MetaQuotes Ltd.

```
            }
         //--- 決済情報の出力
         PrintFormat("Close #%I64d %s %
s",position_ticket,position_symbol,EnumToString(type));
         //--- リクエストの送信
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // リクエストの送信に失敗した
場合、エラーコードを出力
         //--- 操作情報
         PrintFormat("retcode=%u  deal=%I64u  order=%
I64u",result.retcode,result.deal,result.order);
         //---
        }
     }
  }
//+------------------------------------------------------------------+
```

未決注文を設定する 取引操作TRADE_ACTION_PENDINGの例：

© 2000–2025, MetaQuotes Ltd.

```
#property description "未決注文設定の例"
#property script_show_inputs
#define EXPERT_MAGIC 123456                                    // エキスパートアドバイザの
MagicNumber
input ENUM_ORDER_TYPE orderType=ORDER_TYPE_BUY_LIMIT;   // 注文タイプ
//+------------------------------------------------------------------+
//| 未決注文の設定                                                   |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- リクエストと結果の宣言と初期化
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- 未決注文の設定の為のパラメータ
   request.action   =TRADE_ACTION_PENDING;                         // 取引操作タイ
プ
   request.symbol   =Symbol();                                     // シンボル
   request.volume   =0.1;                                          // 0.1ロットのボ
リューム
   request.deviation=2;                                            // 価格からの許
容偏差
   request.magic    =EXPERT_MAGIC;                                 // 注文の
MagicNumber
   int offset = 50;                                                // 注文設定の為
の現在の価格からのオフセット(ポイント)
   double price;                                                   // 注文の発動価
格
   double point=SymbolInfoDouble(_Symbol,SYMBOL_POINT);            // ポイントサイズ
   int digits=SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);            // 小数点以下の
桁数(精度)
   //--- 操作タイプのチェック
   if(orderType==ORDER_TYPE_BUY_LIMIT)
     {
      request.type     =ORDER_TYPE_BUY_LIMIT;                     // 注文タイプ
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;   // 発注価格
      request.price    =NormalizeDouble(price,digits);            // 正規化された
発注価格
     }
   else if(orderType==ORDER_TYPE_SELL_LIMIT)
     {
      request.type     =ORDER_TYPE_SELL_LIMIT;                    // 注文タイプ
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK)+offset*point;   // 発注価格
      request.price    =NormalizeDouble(price,digits);            // 正規化された
発注価格
     }
   else if(orderType==ORDER_TYPE_BUY_STOP)
     {
      request.type =ORDER_TYPE_BUY_STOP;                          // 注文タイプ
      price        =SymbolInfoDouble(Symbol(),SYMBOL_ASK)+offset*point; // 発注価格
      request.price=NormalizeDouble(price,digits);                // 正規化された
発注価格
     }
   else if(orderType==ORDER_TYPE_SELL_STOP)
     {
      request.type     =ORDER_TYPE_SELL_STOP;                     // 注文タイプ
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;   // 発注価格
      request.price    =NormalizeDouble(price,digits);            // 正規化された
発注価格
     }
   else Alert("この例は未決注文の設定のみを対象にしています");    // もし未決注文ではないものが選択されて
いる場合
```

© 2000-2025, MetaQuotes Ltd.

```
//--- リクエストの送信
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());                    // もしリクエスト
の送信に失敗した場合、エラーコードを出力する
//--- 操作情報
   PrintFormat("retcode=%u  deal=%I64u  order=%
I64u",result.retcode,result.deal,result.order);
  }
//+------------------------------------------------------------------+
```

保有ポジションのストップロスとテイクプロフィットの値を変更する 取引操作 TRADE_ACTION_SLTPの例：

```
#define EXPERT_MAGIC 123456  // エキスパートアドバイザのMagicNumber
//+------------------------------------------------------------------+
//| ポジションのストップロスとテイクプロフィットを変更する                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- リクエストと結果の宣言
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // 保有ポジション数
//--- 全ての保有ポジションを取捨
   for(int i=0; i<total; i++)
     {
      //--- 注文パラメータ
      ulong  position_ticket=PositionGetTicket(i);// ポジションチケット
      string position_symbol=PositionGetString(POSITION_SYMBOL); // シンボル
      int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS); // 小数点
以下の桁数
      ulong  magic=PositionGetInteger(POSITION_MAGIC); // ポジションのMagicNumber
      double volume=PositionGetDouble(POSITION_VOLUME);     // ポジションボリューム
      double sl=PositionGetDouble(POSITION_SL);  // ポジションのStop Loss
      double tp=PositionGetDouble(POSITION_TP);  // ポジションのTake Profit
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)
PositionGetInteger(POSITION_TYPE);  // ポジションタイプ
      //--- ポジション情報の出力
      PrintFormat("#%I64u %s  %s  %.2f  %s  sl: %s  tp: %s  [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  DoubleToString(sl,digits),
                  DoubleToString(tp,digits),
                  magic);
      //--- MagicNumberが一致し、ストップロスとテイクプロフィットが指定されていない場合
      if(magic==EXPERT_MAGIC && sl==0 && tp==0)
        {
```

© 2000-2025, MetaQuotes Ltd.

```
         //--- 現在の価格レートの計算
         double price=PositionGetDouble(POSITION_PRICE_OPEN);
         double bid=SymbolInfoDouble(position_symbol,SYMBOL_BID);
         double ask=SymbolInfoDouble(position_symbol,SYMBOL_ASK);
         int    stop_level=(int)
SymbolInfoInteger(position_symbol,SYMBOL_TRADE_STOPS_LEVEL);
         double price_level;
         //--- 現在の決済価格からの最小許容オフセットが指定されていない場合
         if(stop_level<=0)
            stop_level=150; // 現在の決済価格から150ポイントのオフセットを設定する
         else
            stop_level+=50; // 念の為オフセットレベルは(SYMBOL_TRADE_STOPS_LEVEL + 50)ポイント
と同じにする

         //--- Stop LossとTake Profitの値の計算と四捨五入
         price_level=stop_level*SymbolInfoDouble(position_symbol,SYMBOL_POINT);
         if(type==POSITION_TYPE_BUY)
           {
            sl=NormalizeDouble(bid-price_level,digits);
            tp=NormalizeDouble(ask+price_level,digits);
           }
         else
           {
            sl=NormalizeDouble(ask+price_level,digits);
            tp=NormalizeDouble(bid-price_level,digits);
           }
         //--- リクエストと結果の値のゼロ化
         ZeroMemory(request);
         ZeroMemory(result);
         //--- 操作パラメータの設定
         request.action  =TRADE_ACTION_SLTP; // 取引操作タイプ
         request.position=position_ticket;   // ポジションシンボル
         request.symbol=position_symbol;     // シンボル
         request.sl      =sl;                // ポジションのStop Loss
         request.tp      =tp;                // ポジションのTake Profit
         request.magic=EXPERT_MAGIC;         // MagicNumber позиции
         //--- 変更情報の出力
         PrintFormat("Modify #%I64d %s %
s",position_ticket,position_symbol,EnumToString(type));
         //--- リクエストの送信
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // リクエストの送信に失敗した
場合、エラーコードを出力する
         //--- 操作情報
         PrintFormat("retcode=%u  deal=%I64u  order=%
I64u",result.retcode,result.deal,result.order);
        }
     }
  }
//+------------------------------------------------------------------+
```

未決注文の価格レベルを変更する 取引操作TRADE_ACTION_MODIFYの例：

© 2000-2025, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456  // エキスパートアドバイザのMagicNumber
//+------------------------------------------------------------------+
//| 未決注文の変更                                                   |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- リクエストと結果の宣言と初期化
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   int total=OrdersTotal(); // 保有未決注文数
//--- 全ての保有未決注文を取捨
   for(int i=0; i<total; i++)
     {
      //--- 注文パラメータ
      ulong  order_ticket=OrderGetTicket(i);                                  //注文チケット
      string order_symbol=Symbol();                                           // シンボル
      int    digits=(int)SymbolInfoInteger(order_symbol,SYMBOL_DIGITS);       // 小数点以下の桁数
      ulong  magic=OrderGetInteger(ORDER_MAGIC);                              // 注文のMagicNumber
      double volume=OrderGetDouble(ORDER_VOLUME_CURRENT);                     // 現在の注文量
      double sl=OrderGetDouble(ORDER_SL);                                     // 現在の注文のStop Loss
      double tp=OrderGetDouble(ORDER_TP);                                     // 現在の注文のTake Profit
      ENUM_ORDER_TYPE type=(ENUM_ORDER_TYPE)OrderGetInteger(ORDER_TYPE);      // 注文タイプ
      int offset = 50;                                                        // 発注の為の現在価格からのオフセット（ポイント）
      double price;                                                           // 注文発動価格
      double point=SymbolInfoDouble(order_symbol,SYMBOL_POINT);               // ポイントサイズ
      //--- 注文情報の出力
      PrintFormat("#%I64u %s  %s  %.2f  %s  sl: %s  tp: %s  [%I64d]",
                  order_ticket,
                  order_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  DoubleToString(sl,digits),
                  DoubleToString(tp,digits),
                  magic);
      //--- MagicNumberが一致し、Stop LossとTake Profitが設定されていない場合
      if(magic==EXPERT_MAGIC && sl==0 && tp==0)
        {
         request.action=TRADE_ACTION_MODIFY;                                  // 取引操作タイプ
         request.order = OrderGetTicket(i);                                   // 注文チケット
         request.symbol    =Symbol();                                         // シンボル
         request.deviation=5;                                                 // 価格からの許容オフセット
        //--- 価格レベルの設定、タイプに合わせた注文のTake ProfitとStop Loss
         if(type==ORDER_TYPE_BUY_LIMIT)
           {
            price = SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;
            request.tp = NormalizeDouble(price+offset*point,digits);
            request.sl = NormalizeDouble(price-offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                  // 正規化された発注価格
           }
```

© 2000-2025, MetaQuotes Ltd.

```
         else if(type==ORDER_TYPE_SELL_LIMIT)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_BID)+offset*point;
            request.tp = NormalizeDouble(price-offset*point,digits);
            request.sl = NormalizeDouble(price+offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                  // 正規
化された発注価格
           }
         else if(type==ORDER_TYPE_BUY_STOP)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_BID)+offset*point;
            request.tp = NormalizeDouble(price+offset*point,digits);
            request.sl = NormalizeDouble(price-offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                  // 正規
化された発注価格
           }
         else if(type==ORDER_TYPE_SELL_STOP)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;
            request.tp = NormalizeDouble(price-offset*point,digits);
            request.sl = NormalizeDouble(price+offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                  // 正規
化された発注価格
           }
         //---リクエストの送信
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // リクエストの送信に失敗した
場合、エラーコードを出力する
         //--- 操作情報
         PrintFormat("retcode=%u  deal=%I64u  order=%
I64u",result.retcode,result.deal,result.order);
         //--- リクエストと結果の値のゼロ化
         ZeroMemory(request);
         ZeroMemory(result);
        }
     }
  }
//+------------------------------------------------------------------+
```

未決注文の削除をする取引操作TRADE_ACTION_REMOVEの例：

```
#define EXPERT_MAGIC 123456  // エキスパートアドバイザのMagicNumber
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  未決注文の削除                                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- リクエストと結果の宣言と初期化
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   int total=OrdersTotal(); // 保有未決注文数
//--- 全ての保有未決注文を取捨
   for(int i=total-1; i>=0; i--)
     {
      ulong  order_ticket=OrderGetTicket(i);                   // 注文チケット
      ulong  magic=OrderGetInteger(ORDER_MAGIC);               // 注文のMagicNumber
      //--- MagicNumberが一致している場合
      if(magic==EXPERT_MAGIC)
        {
         //--- リクエストと結果の値のゼロ化
         ZeroMemory(request);
         ZeroMemory(result);
         //--- 操作パラメータの設定
         request.action=TRADE_ACTION_REMOVE;                   // 取引操作タイプ
         request.order = order_ticket;                         // 注文チケット
         //--- リクエストの送信
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // リクエストの送信に失敗した
場合、エラーコードを出力する
         //--- 操作情報
         PrintFormat("retcode=%u  deal=%I64u  order=%
I64u",result.retcode,result.deal,result.order);
        }
     }
  }
//+------------------------------------------------------------------+
```

反対ポジションを決済する取引操作TRADE_ACTION_CLOSE_BYの例：

© 2000-2025, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456  // エキスパートアドバイザのMagicNumber
//+------------------------------------------------------------------+
//| 全ての反対ポジションの決済                                       |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- リクエストと結果の宣言
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // 保有ポジション数
//--- 全ての保有ポジションの取捨
   for(int i=total-1; i>=0; i--)
     {
      //--- 注文パラメータ
      ulong
 position_ticket=PositionGetTicket(i);                                  // ポジション
チケット
      string
position_symbol=PositionGetString(POSITION_SYMBOL);                    // シンボル
      int    digits=(int)
SymbolInfoInteger(position_symbol,SYMBOL_DIGITS);          // 小数点以下の桁数
      ulong
 magic=PositionGetInteger(POSITION_MAGIC);                              // ポジション
のMagicNumber
      double
volume=PositionGetDouble(POSITION_VOLUME);                             // ポジションボ
リューム
      double
sl=PositionGetDouble(POSITION_SL);                                     // ポジションの
Stop Loss
      double
tp=PositionGetDouble(POSITION_TP);                                     // ポジションの
Take Profit
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)
PositionGetInteger(POSITION_TYPE);  // ポジションタイプ
      //--- ポジション情報の出力
      PrintFormat("#%I64u %s  %s  %.2f  %s  sl: %s  tp: %s  [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  DoubleToString(sl,digits),
                  DoubleToString(tp,digits),
                  magic);
      //--- MagicNumberが一致している場合
      if(magic==EXPERT_MAGIC)
        {
         for(int j=0; j<i; j++)
           {
            string symbol=PositionGetSymbol(j); // 反対ポジションのシンボル
            //--- 反対と未知のポジションのシンボルが一致している場合
            if(symbol==position_symbol && PositionGetInteger(POSITION_MAGIC)
==EXPERT_MAGIC)
              {
               //--- 反対ポジションのタイプの設定
               ENUM_POSITION_TYPE type_by=(ENUM_POSITION_TYPE)
PositionGetInteger(POSITION_TYPE);
               //--- 元のポジションと反対ポジションのタイプが一致する場合終了する
               if(type==type_by)
                  continue;
```

© 2000-2025, MetaQuotes Ltd.

```
                  //--- リクエストと結果の値のゼロ化
                  ZeroMemory(request);
                  ZeroMemory(result);
                  //--- 操作パラメータの設定
                  request.action=TRADE_ACTION_CLOSE_BY;                    //取引操
作タイプ
                  request.position=position_ticket;                        // ポジ
ションチケット
                  request.position_by=PositionGetInteger(POSITION_TICKET);       // 反対
ポジションのチケット
                  //request.symbol     =position_symbol;
                  request.magic=EXPERT_MAGIC;                              // ポジ
ションのMagicNumber
                  //--- 反対ポジションの決済情報の出力
                  PrintFormat("Close #%I64d %s %s by #%
I64d",position_ticket,position_symbol,EnumToString(type),request.position_by);
                  //--- リクエストの送信
                  if(!OrderSend(request,result))
                     PrintFormat("OrderSend error %d",GetLastError()); // リクエストの送信に
失敗した場合、エラーコードを出力する

                  //--- 操作情報
                  PrintFormat("retcode=%u  deal=%I64u  order=%
I64u",result.retcode,result.deal,result.order);
               }
            }
         }
      }
   }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

取引トランザクションの種類

取引口座にいくつかの明確なアクションを実行すると、状態が変更されます。アクションの例は下記です。

- クライアント端末内の任意の MQL5 アプリケーションから OrderSend と OrderSendAsync 関数を使用して取引リクエストを送信し、取引が実行される。
- 端末のグラフィカルインターフェースを使用して取引リクエストを送信し、取引が実行される。
- サーバ上で未決注文と逆指値注文が執行される。
- 取引サーバ側で操作を行う。

これらのアクションの結果として以下の取引トランザクションが実行されます。

- 取引リクエストの処理
- 未執行注文の変更
- 注文履歴の変更
- 約定履歴の変更
- ポジションの変更

例えば、市場の買い注文を送信する際に、注文が処理され、買い注文が口座に記録されます。その後注文が実行されオープン注文の表から削除されて注文履歴に追加されます。約定履歴に追加され新しいポジションが作成されます。これらのアクションは全て取引トランザクションです。

プログラマによる取引口座に関連して実行されるアクション追跡が可能なように OnTradeTransaction 関数が提供されています。このハンドラは MQL5 アプリケーションの口座に適用される取引トランザクションを取得することが出来ます。取引トランザクションの記述は MqlTradeTransaction 構造体を使用して OnTradeTransaction の1番目のパラメータとして渡されます。

取引トランザクションの種類は MqlTradeTransaction 構造体型のパラメータで提出されます。取引トランザクションの種類は、以下の列挙に記載されています。

ENUM_TRADE_TRANSACTION_TYPE

識別子	説明
TRADE_TRANSACTION_ORDER_ADD	新規の未執行注文の追加。
TRADE_TRANSACTION_ORDER_UPDATE	未執行注文の更新。アップデートは、クライアント端末や取引サーバ側からの明らかな変更だけでなく、設定時の注文状態の変化を含みます（例えば ORDER_STATE_STARTED から ORDER_STATE_PLACED へや ORDER_STATE_PLACED から ORDER_STATE_PARTIALへの変化など）。
TRADE_TRANSACTION_ORDER_DELETE	注文の未処理注文リストからの削除。注文は、適切なリクエストの設定または実行（充填）後、履歴に移動された結果として、未処理の注文のリストから削除することが出来ます。
TRADE_TRANSACTION_DEAL_ADD	約定を履歴に追加。このアクションは注文実行や口座残高の操作の結果として実行されます。
TRADE_TRANSACTION_DEAL_UPDATE	約定を履歴での更新。以前に実行された約定がサーバで更新される場合があります。例えば、約定が以前にブ

© 2000–2025, MetaQuotes Ltd.

識別子	説明
	ブローカによって転送された外部の取引システム（exchange）で更新される場合です。
TRADE_TRANSACTION_DEAL_DELETE	約定履歴の削除。以前に実行された約定がサーバから削除される場合があります。例えば、約定が以前にブローカによって転送された外部の取引システム（exchange）で削除されている場合です。
TRADE_TRANSACTION_HISTORY_ADD	実行またはキャンセルの結果として履歴に注文を追加します。
TRADE_TRANSACTION_HISTORY_UPDATE	注文履歴内の注文を変更します。この型は、取引サーバ側の機能性を高めるために設けられています。
TRADE_TRANSACTION_HISTORY_DELETE	注文を履歴から削除します。この型は、取引サーバ側の機能性を高めるために設けられています。
TRADE_TRANSACTION_POSITION	取引の実行に関連していないポジションを変更します。このトランザクション型はポジションが取引サーバ側で変更されたことを示しています。ポジションボリューム、始値、決済逆指及び決済指レベルの変更が可能です。返られたデータは OnTradeTransaction ハンドラで MqlTradeTransaction 構造体に提示されます。約定実行によるポジションの変化（追加、変更、閉鎖）は TRADE_TRANSACTION_POSITION トランザクションの発生にはつながりません。
TRADE_TRANSACTION_REQUEST	取引リクエストがサーバと処理され結果が受け取られたことの通知。そのような取引では MqlTradeTransaction 構造体の type フィールド（取引タイプ）の分析のみが必要です。OnTradeTransaction 関数の 2、3 番目のパラメータ（request と result）は追加のデータを得るために分析されなければいけません。

取引トランザクションの種類によって、各種パラメータが MqlTradeTransaction 構造体に書き入れられます。引き渡されたデータの詳細な説明は「取引トランザクション構造体」でご覧下さい。

参照

取引トランザクション構造体、OnTradeTransaction

© 2000–2025, MetaQuotes Ltd.

板情報の取引注文

持分証券では、現在の売買注文を含む板情報が利用可能です。希望された取引操作の方向、必要な量とリクエストされた価格は、注文ごとに指定されています。

MQL5 で現在の DOM の情報を取得するには MqlBookInfo 構造体の配列に DOM「スクリーンショット」を配置するMarketBookGet() 関数が使用されます。*type* フィールドの配列の各要素は ENUM_BOOK_TYPE 列挙値での注文方向の情報を含みます。

ENUM_BOOK_TYPE

識別子	説明
BOOK_TYPE_SELL	売り注文（オファー）。
BOOK_TYPE_BUY	買い注文（ビッド）。
BOOK_TYPE_SELL_MARKET	売りの成行注文。
BOOK_TYPE_BUY_MARKET	買いの成行注文。

参照

構造体とクラス、DOM 構造体、取引操作の種類、市場情報

© 2000–2025, MetaQuotes Ltd.

シグナルプロパティ

売買シグナルとシグナルコピーの設定を使用する際は、次の列挙が使用されています。

売買シグナルの double 型プロパティの列挙

ENUM_SIGNAL_BASE_DOUBLE

ID	説明
SIGNAL_BASE_BALANCE	口座残高。
SIGNAL_BASE_EQUITY	口座株式。
SIGNAL_BASE_GAIN	口座ゲイン。
SIGNAL_BASE_MAX_DRAWDOWN	口座最大ドローダウン。
SIGNAL_BASE_PRICE	シグナルサブスクリプション価格。
SIGNAL_BASE_ROI	投資収益（%）。

売買シグナルの integer 型プロパティの列挙

ENUM_SIGNAL_BASE_INTEGER

ID	説明
SIGNAL_BASE_DATE_PUBLISHED	公開日（サブスクリプションが可能になった日付）。
SIGNAL_BASE_DATE_STARTED	モニターの開始日。
SIGNAL_BASE_DATE_UPDATED	シグナルの取引統計の最終更新日
SIGNAL_BASE_ID	シグナル ID。
SIGNAL_BASE_LEVERAGE	口座レバレッジ。
SIGNAL_BASE_PIPS	利益（ピップス単位）。
SIGNAL_BASE_RATING	建て格付けでのポジション。
SIGNAL_BASE_SUBSCRIBERS	加入者数。
SIGNAL_BASE_TRADES	取引数。
SIGNAL_BASE_TRADE_MODE	口座の種類（ 0 – 実際、1 – デモ、2 – コンテスト）。

売買シグナルの string 型プロパティの列挙

ENUM_SIGNAL_BASE_STRING

ID	説明
SIGNAL_BASE_AUTHOR_LOGIN	著者のログイン。
SIGNAL_BASE_BROKER	ブローカー名（会社）。

© 2000–2025, MetaQuotes Ltd.

ID	説明
SIGNAL_BASE_BROKER_SERVER	ブローカーサーバ。
SIGNAL_BASE_NAME	シグナル名。
SIGNAL_BASE_CURRENCY	シグナル基準通貨。

シグナルコピー設定の double 型プロパティの列挙

ENUM_SIGNAL_INFO_DOUBLE

ID	説明
SIGNAL_INFO_EQUITY_LIMIT	株式の制限。
SIGNAL_INFO_SLIPPAGE	スリッページ（ポジションと同期して成行注文をする時と取引の取引の複製時に使用）。
SIGNAL_INFO_VOLUME_PERCENT	預金使用の最大率（%）（読み込みのみ）。

シグナルコピー設定の integer 型プロパティの列挙

ENUM_SIGNAL_INFO_INTEGER

ID	説明
SIGNAL_INFO_CONFIRMATIONS_DISABLED	フラグは、確認ダイアログなしでの同期を可能にします。
SIGNAL_INFO_COPY_SLTP	決済逆指値及び決済指値フラグを複製。
SIGNAL_INFO_DEPOSIT_PERCENT	預金比率（%）。
SIGNAL_INFO_ID	シグナル ID（読み込みのみ）。
SIGNAL_INFO_SUBSCRIPTION_ENABLED	「サブスクリプションによっての取引コピー」許可フラグ。
SIGNAL_INFO_TERMS_AGREE	「シグナルサービスの利用規約に同意」フラグ（読み込みのみ）。

シグナルコピー設定の string 型プロパティの列挙

ENUM_SIGNAL_INFO_STRING

ID	説明
SIGNAL_INFO_NAME	シグナル名（読み込みのみ）。

参照

売買シグナル

© 2000–2025, MetaQuotes Ltd.

名前付き定数

MQL5 で使用される全ての定数は、以下のグループに分けることが出来ます。

- 定義済みマクロ代入 – 値はコンパイル時に代入されます。
- 数学定数 – 数式の値。
- 数値型定数 – 基本データ型の制限の一部。
- 初期化解除の理由のコード – 初期化解除の理由の説明。
- オブジェクトポインタのチェック - CheckPointer() 関数で返されたポインタ型の列挙。
- その他の定数 – その他の全ての定数。

© 2000–2025, MetaQuotes Ltd.

定義済みマクロ代入

デバッグプロセスを簡素化し、MQL5 プログラムの動作に関する情報を取得するには、コンパイルの時点で値が設定された特別なマクロ定数があります。これらの定数を使用する最も簡単な方法は、Print() 関数による値出力です。

定数	説明
__CPU_ARCHITECTURE__	アーキテクチャ（コマンドのセット）用にコンパイルされたEX5ファイルの名前
__DATE__	時刻を含まないファイルコンパイルの日付（時間、分、秒は0）。
__DATETIME__	ファイルコンパイルの日付と時刻。
__LINE__	マクロが配置されているソースコードの行番号。
__FILE__	現在コンパイルされているファイルの名称。
__PATH__	現在コンパイルされているファイルへの絶対パス。
__FUNCTION__	本体にマクロが配置されている関数の名称。
__FUNCSIG__	本体にマクロが配置されている関数のシグネチャ。関数の完全な記述のロギングはオーバーロード関数の識別に役立つことがあります。
__MQLBUILD__, __MQL5BUILD__	コンパイラのビルド番号。
__COUNTER__	コンパイラは、検出された各__COUNTER__宣言を0からN-1までのカウンター値で置き換えます。ここで、Nはコードでの使用回数です。__COUNTER__の順序はソースコードを再コンパイルしても変わらないことが保証されています。 __COUNTER__値は、次の方法で計算されます。 • 初期カウンタ値は0です。 • 各カウンタの使用後、その値は1ずつ増加します。 • まず、コンパイラはすべてのマクロとテンプレートをオンサイトでソースコードに展開します。 • テンプレート関数の特殊化ごとに個別のコードが作成されます。 • テンプレートクラス/構造体の特殊化ごとに個別のコードが作成されます。 • 次に、コンパイラは取得したソースコードを定義された順序で渡し、検出された__COUNTER__の各使用を現在のカウンタ値に置き換えます。 以下の例は、コンパイラがソースコードを処理し、検出された__COUNTER__のすべてのインスタンスを順次増加する値に置き換える方法を示しています。
__RANDOM__	コンパイラは、__RANDOM__宣言ごとにランダムなulong値を挿入します。

例：

```
#property copyright "Copyright © 2009, MetaQuotes Software Corp."
#property link      "https://www.metaquotes.net"
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  エキスパート初期化に使用される関数                                |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- エキスパートアドバイザーの初期化情報出力の例
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
//--- タイマーイベントの間隔を設定する
   EventSetTimer(5);
//---
  }
//+------------------------------------------------------------------+
//|  エキスパート初期化解除に使用される関数                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- エキスパートアドバイザーの初期化解除情報出力の例
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
//---
  }
//+------------------------------------------------------------------+
//|  エキスパートティック関数                                          |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- ティック受け取りの情報出力
   Print(" __MQLBUILD__ = ",__MQLBUILD__,"  __FILE__ = ",__FILE__);
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
   test1(__FUNCTION__);
   test2();
//---
  }
//+------------------------------------------------------------------+
//|  テスト1                                                         |
//+------------------------------------------------------------------+
void test1(string par)
  {
//--- 関数内での情報出力
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__," par=",par);
  }
//+------------------------------------------------------------------+
//|  テスト2                                                         |
//+------------------------------------------------------------------+
void test2()
  {
//--- 関数内での情報出力
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| OnTimer イベントハンドラ                                          |
//+------------------------------------------------------------------+
void OnTimer()
  {
//---
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
   test1(__FUNCTION__);
  }
```

__COUNTER__マクロの使用例

```
//--- 式とその値を操作ログにすばやく表示するためのマクロを作成する
#define print(expr) Print(#expr,"=",expr)

//--- 事前定義された__COUNTER__マクロを介してMACRO_COUNTERカスタムマクロを定義する
#define MACRO_COUNTER __COUNTER__

//--- __COUNTER__マクロを使用して変数の入力値を設定する
input int InpVariable = __COUNTER__;

//--- 関数を定義する前に、__COUNTER__マクロを使用してグローバル変数の値を設定する
int ExtVariable = __COUNTER__;

//+------------------------------------------------------------------+
//| 関数は__COUNTER__値を戻す                                        |
//+------------------------------------------------------------------+
int GlobalFunc(void)
  {
   return(__COUNTER__);
  }
//+------------------------------------------------------------------+
//| テンプレート関数は__COUNTER__値を戻す                            |
//+------------------------------------------------------------------+
template<typename T>
int GlobalTemplateFunc(void)
  {
   return(__COUNTER__);
  }
//+------------------------------------------------------------------+
//| __COUNTER__を戻すメソッドを持つ構造体                            |
//+------------------------------------------------------------------+
struct A
  {
   int               dummy;  // 未使用

   int               Method(void)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       return(__COUNTER__);
      }
  };
//+------------------------------------------------------------------+
//| __COUNTER__を戻すテンプレートメソッドを持つ構造体                |
//+------------------------------------------------------------------+
template<typename T>
struct B
  {
   int               dummy;  // 未使用

   int               Method(void)
     {
      return(__COUNTER__);
     }
  };
//+------------------------------------------------------------------+
//| __COUNTER__を返すテンプレートメソッドを持つ構造体                |
//+------------------------------------------------------------------+
struct C
  {
   int               dummy;  // 未使用

   template<typename T>
   int               Method(void)
     {
      return(__COUNTER__);
     }
  };
//+------------------------------------------------------------------+
//| 関数#2は__COUNTER__値を戻す                                      |
//+------------------------------------------------------------------+
int GlobalFunc2(void)
  {
   return(__COUNTER__);
  }
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart(void)
  {
// マクロと変数での__COUNTER__
   print(MACRO_COUNTER);
   print(InpVariable);
   print(ExtVariable);

//--- 関数での__COUNTER__
   print(GlobalFunc());
```

© 2000-2025, MetaQuotes Ltd.

```
   print(GlobalFunc());                  // 値は不変
   print(GlobalTemplateFunc<int>());
   print(GlobalTemplateFunc<int>());     // 値は不変
   print(GlobalTemplateFunc<double>());// 値が変更されている
   print(GlobalFunc2());
   print(GlobalFunc2());                 // 値は不変

// 構造体での__COUNTER__
   A a1, a2;
   print(a1.Method());
   print(a2.Method());                   // 値は不変

// テンプレート構造体での__COUNTER__
   B<int> b1, b2;
   B<double> b3;
   print(b1.Method());
   print(b2.Method());                   // 値は不変
   print(b3.Method());                   // 値が変更されている

// テンプレート関数を持つ構造体での__COUNTER__
   C c1, c2;
   print(c1.Method<int>());
   print(c1.Method<double>());           // 値が変更されている
   print(c2.Method<int>());              // the same value as during the first
c1.Method<int>() call

//--- マクロとグローバル変数の__COUNTER__をもう一度見てみる
   print(MACRO_COUNTER);  // 値が変更されている
   print(ExtGlobal2);
  }
//--- 関数を定義した後、__COUNTER__マクロを使用してグローバル変数の値を設定する
int ExtGlobal2 = __COUNTER__;
//+------------------------------------------------------------------+

/* 結果
   __COUNTER__=3
   InpVariable=0
   ExtVariable=1
   GlobalFunc()=5
   GlobalFunc()=5
   GlobalTemplateFunc<int>()=8
   GlobalTemplateFunc<int>()=8
   GlobalTemplateFunc<double>()=9
   GlobalFunc2()=7
   GlobalFunc2()=7
   a1.Method()=6
   a2.Method()=6
   b1.Method()=10
   b2.Method()=10
```

© 2000-2025, MetaQuotes Ltd.

```
   b3.Method()=11
   c1.Method<int>()=12
   c1.Method<double>()=13
   c2.Method<int>()=12
   __COUNTER__=4
   ExtGlobal2=2

*/
```

© 2000–2025, MetaQuotes Ltd.

数学定数

特別な定数群の値は、数学的な表現のために予約されていますこれらの定数は数学関数を使用して値を計算する代わりにプログラムで使用することが出来ます。

定数	説明	値
M_E	e	2.71828182845904523536
M_LOG2E	log2(e)	1.44269504088896340736
M_LOG10E	log10(e)	0.434294481903251827651
M_LN2	ln(2)	0.693147180559945309417
M_LN10	ln(10)	2.30258509299404568402
M_PI	pi	3.14159265358979323846
M_PI_2	pi/2	1.57079632679489661923
M_PI_4	pi/4	0.785398163397448309616
M_1_PI	1/pi	0.318309886183790671538
M_2_PI	2/pi	0.636619772367581343076
M_2_SQRTPI	2/sqrt(pi)	1.12837916709551257390
M_SQRT2	sqrt(2)	1.41421356237309504880
M_SQRT1_2	1/sqrt(2)	0.70710678118654524401

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 定数値を出力する
   Print("M_E = ",DoubleToString(M_E,16));
   Print("M_LOG2E = ",DoubleToString(M_LOG2E,16));
   Print("M_LOG10E = ",DoubleToString(M_LOG10E,16));
   Print("M_LN2 = ",DoubleToString(M_LN2,16));
   Print("M_LN10 = ",DoubleToString(M_LN10,16));
   Print("M_PI = ",DoubleToString(M_PI,16));
   Print("M_PI_2 = ",DoubleToString(M_PI_2,16));
   Print("M_PI_4 = ",DoubleToString(M_PI_4,16));
   Print("M_1_PI = ",DoubleToString(M_1_PI,16));
   Print("M_2_PI = ",DoubleToString(M_2_PI,16));
   Print("M_2_SQRTPI = ",DoubleToString(M_2_SQRTPI,16));
   Print("M_SQRT2 = ",DoubleToString(M_SQRT2,16));
   Print("M_SQRT1_2 = ",DoubleToString(M_SQRT1_2,16));
  }
```

© 2000-2025, MetaQuotes Ltd.

数値定数

単純な数値型は、それぞれ特定の種類のタスクのために意図され、正しく使用した場合 MQL5 プログラムの動作の最適化を可能にします。コードをより読みやすくして計算結果を正しく処理するために、いくつかの基本データ型に設定された制限に関する情報を受信出来る定数があります。

定数	説明	値
CHAR_MIN	char 型で表すことが出来る最小値。。	-128
CHAR_MAX	char 型で表すことが出来る最大値。。	127
UCHAR_MAX	uchar 型で表すことが出来る最大値。。	255
SHORT_MIN	short 型で表すことが出来る最小値。。	-32768
SHORT_MAX	short 型で表すことが出来る最大値。。	32767
USHORT_MAX	ushort 型で表すことが出来る最大値。。	65535
INT_MIN	int 型で表すことが出来る最小値。。	-2147483648
INT_MAX	int 型で表すことが出来る最大値。。	2147483647
UINT_MAX	uint 型で表すことが出来る最大値。。	4294967295
LONG_MIN	long 型で表すことが出来る最小値。。	-9223372036854775808
LONG_MAX	long 型で表すことが出来る最大値。。	9223372036854775807
ULONG_MAX	ulong 型で表すことが出来る最大値。。	18446744073709551615
DBL_MIN	double 型で表すことが出来る最小の正の値。	2.2250738585072014e-308
DBL_MAX	double 型で表すことが出来る最大値。。	1.7976931348623158e+308
DBL_EPSILON	下記の条件を満たす最小値。 1.0+DBL_EPSILON != 1.0（double 型）	2.2204460492503131e-016
DBL_DIG	double 型の小数点以下の桁数。	15
DBL_MANT_DIG	double 型の仮数部のビット数。	53
DBL_MAX_10_EXP	double 型の指数度の 10 進数での最大値。	308
DBL_MAX_EXP	double 型の指数度の 2 進数での最大値。	1024
DBL_MIN_10_EXP	double 型の指数度の 10 進数での最小値。	(-307)
DBL_MIN_EXP	double 型の指数度の 2 進数での最小値。	(-1021)
FLT_MIN	float 型で表すことが出来る最小の正の値。	1.175494351e-38
FLT_MAX	float 型で表すことが出来る最大値。。	3.402823466e+38
FLT_EPSILON	下記の条件を満たす最小値。 1.0+DBL_EPSILON != 1.0（float 型）	1.192092896e-07

© 2000–2025, MetaQuotes Ltd.

定数	説明	値
FLT_DIG	float 型の小数点以下の桁数。	6
FLT_MANT_DIG	float 型の仮数部のビット数。	24
FLT_MAX_10_EXP	float 型の指数度の 10 進数での最大値。	38
FLT_MAX_EXP	float 型の指数度の 2 進数での最大値。	128
FLT_MIN_10_EXP	float 型の指数度の 10 進数での最小値。	-37
FLT_MIN_EXP	float 型の指数度の 2 進数での最小値。	(-125)

例：

```
void OnStart()
  {
//--- 定数値を出力する
   printf("CHAR_MIN = %d",CHAR_MIN);
   printf("CHAR_MAX = %d",CHAR_MAX);
   printf("UCHAR_MAX = %d",UCHAR_MAX);
   printf("SHORT_MIN = %d",SHORT_MIN);
   printf("SHORT_MAX = %d",SHORT_MAX);
   printf("USHORT_MAX = %d",USHORT_MAX);
   printf("INT_MIN = %d",INT_MIN);
   printf("INT_MAX = %d",INT_MAX);
   printf("UINT_MAX = %u",UINT_MAX);
   printf("LONG_MIN = %I64d",LONG_MIN);
   printf("LONG_MAX = %I64d",LONG_MAX);
   printf("ULONG_MAX = %I64u",ULONG_MAX);
   printf("EMPTY_VALUE = %.16e",EMPTY_VALUE);
   printf("DBL_MIN = %.16e",DBL_MIN);
   printf("DBL_MAX = %.16e",DBL_MAX);
   printf("DBL_EPSILON = %.16e",DBL_EPSILON);
   printf("DBL_DIG = %d",DBL_DIG);
   printf("DBL_MANT_DIG = %d",DBL_MANT_DIG);
   printf("DBL_MAX_10_EXP =  %d",DBL_MAX_10_EXP);
   printf("DBL_MAX_EXP = %d",DBL_MAX_EXP);
   printf("DBL_MIN_10_EXP = %d",DBL_MIN_10_EXP);
   printf("DBL_MIN_EXP = %d",DBL_MIN_EXP);
   printf("FLT_MIN = %.8e",FLT_MIN);
   printf("FLT_MAX = %.8e",FLT_MAX);
   printf("FLT_EPSILON = %.8e",FLT_EPSILON);
/*
   CHAR_MIN = -128
   CHAR_MAX = 127
   UCHAR_MAX = 255
   SHORT_MIN = -32768
   SHORT_MAX = 32767
```

© 2000-2025, MetaQuotes Ltd.

```
   USHORT_MAX = 65535
   INT_MIN = -2147483648
   INT_MAX = 2147483647
   UINT_MAX = 4294967295
   LONG_MIN = -9223372036854775808
   LONG_MAX = 9223372036854775807
   ULONG_MAX = 18446744073709551615
   EMPTY_VALUE = 1.7976931348623157e+308
   DBL_MIN = 2.2250738585072014e-308
   DBL_MAX = 1.7976931348623157e+308
   DBL_EPSILON = 2.2204460492503131e-16
   DBL_DIG = 15
   DBL_MANT_DIG = 53
   DBL_MAX_10_EXP =  308
   DBL_MAX_EXP = 1024
   DBL_MIN_10_EXP = -307
   DBL_MIN_EXP = -1021
   FLT_MIN = 1.17549435e-38
   FLT_MAX = 3.40282347e+38
   FLT_EPSILON = 1.19209290e-07
*/
  }
```

© 2000–2025, MetaQuotes Ltd.

初期化解除の理由のコード

初期化解除の理由のコードは UninitializeReason() 関数から返されます。可能な値は下記の通りです。

定数	値	説明
REASON_PROGRAM	0	エキスパートアドバイザーが ExpertRemove() 関数を呼び出して操作を終了しました。
REASON_REMOVE	1	プログラムがチャートから削除されました。
REASON_RECOMPILE	2	プログラムが再コンパイルされました。
REASON_CHARTCHANGE	3	シンボルかチャート期間が変更されました。
REASON_CHARTCLOSE	4	チャートが閉じられました。
REASON_PARAMETERS	5	入力パラメータがユーザーによって変更されました。
REASON_ACCOUNT	6	別のアカウントが有効化されるか、アカウントの設定変更によって取引サーバへの再接続が発生しました。
REASON_TEMPLATE	7	新しいテンプレートが適用されました。
REASON_INITFAILED	8	この値は OnInit() ハンドラがゼロ以外の値を返したことを示します。
REASON_CLOSE	9	端末が閉じられました。

初期化解除理由のコードは、また、所定の関数 OnDeinit(const int reason) のパラメータとしても渡されます。

例：

```
//+------------------------------------------------------------------+
//|  テキスト記述を取得                                              |
//+------------------------------------------------------------------+
string getUninitReasonText(int reasonCode)
  {
   string text="";
//---
   switch(reasonCode)
     {
      case REASON_ACCOUNT:
         text="Account was changed";break;
      case REASON_CHARTCHANGE:
         text="Symbol or timeframe was changed";break;
      case REASON_CHARTCLOSE:
         text="Chart was closed";break;
      case REASON_PARAMETERS:
         text="Input-parameter was changed";break;
      case REASON_RECOMPILE:
         text="Program "+__FILE__+" was recompiled";break;
      case REASON_REMOVE:
```

© 2000-2025, MetaQuotes Ltd.

```
         text="Program "+__FILE__+" was removed from chart";break;
      case REASON_TEMPLATE:
         text="New template was applied to chart";break;
      default:text="Another reason";
     }
//---
   return text;
  }
//+------------------------------------------------------------------+
//| エキスパート初期化解除に使用される関数                                |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- 初期化解除の理由のコード取得の方法その 1
   Print(__FUNCTION__,"_Uninitalization reason code = ",reason);
    //--- 初期化解除の理由のコード取得の方法その 2
   Print(__FUNCTION__,"_UninitReason = ",getUninitReasonText(_UninitReason));
  }
```

© 2000-2025, MetaQuotes Ltd.

オブジェクト ポインタのチェック

CheckPointer() 関数はオブジェクト ポインタの型をチェックするのに使用されます。この関数は ENUM_POINTER_TYPE 列挙のいずれかの値を返します。不正なポインタが使用されると、プログラムの実行が直ちに終了されます。

new() 演算子で作成されたオブジェクトは POINTER_DYNAMIC 型です。この様なポインタとは delete() 演算子 のみが使用されるべきです。

他の全てのポインタは、オブジェクトが MQL5 プログラム環境によって自動的に作成されたことを意味する POINTER_AUTOMATIC 型のものです。このようなオブジェクトは使用後に自動的に削除されます。

ENUM_POINTER_TYPE

定数	説明
POINTER_INVALID	不正ポインタ。
POINTER_DYNAMIC	new() 演算子で作成されたオブジェクトのポインタ。
POINTER_AUTOMATIC	自動的に(new() を使用せずに)作成されたオブジェクトのポインタ。

参照

ランタイムエラー、delete オブジェクト解除演算子、CheckPointer

© 2000–2025, MetaQuotes Ltd.

その他の定数

CLR_NONE　定数は、色が存在しないことを概説するために使用され、指標のグラフィックオブジェクトやグラフィックシリーズはプロットされません。この定数はウェブカラー定数リストには含まれていませんが、カラー引数が必要なところならどこにでも適用することが出来ます。

INVALID_HANDLE 定数はハンドラのチェックに使用されます（FileOpen() 及び FileFindFirst() を参照）。

定数	説明	値
CHARTS_MAX	端末内に同時に開けるチャートの最大数。	100
clrNONE	無色。	-1
EMPTY_VALUE	指標バファーの空値。	DBL_MAX
INVALID_HANDLE	不正なハンドル。	-1
IS_DEBUG_MODE	MQ5 プログラムをデバグモードで動作するフラグ。	デバグモードではゼロ以外、それ以外はゼロ
IS_PROFILE_MODE	MQ5　プログラムをプロファイルモードで動作するフラグ。	プロファイルモードではゼロ以外、それ以外はゼロ
NULL	すべの型でゼロ。	0
WHOLE_ARRAY	配列の最後までに残っている要素の数を表します。配列全体が処理されます。	-1
WRONG_VALUE	定数は任意の列挙型に暗黙的にキャストされることが出来ます。	-1

EMPTY_VALUE 定数は、通常、チャートには示されていない指標値に相当します。例えば、内蔵された 20 時間軸の指標標準偏差では、履歴内の最初の19のバーは、チャートに示されていません。この指標ハンドルを iStdDev() 関数で作成し、指標値の配列を CopyBuffer(), でコピーした場合、値は EMPTY_VALUE に等しくなります。

指標がチャートに表示されてはいけない場合、カスタム指標に独自の空値を指定することが出来ます。この際、PlotIndexSetDouble() 関数を PLOT_EMPTY_VALUE 修飾子と使用します。

。NULL 定数は、基本データ型の変数やオブジェクトの構造やクラスのポインタに割り当てることが出来ます。文字列変数への NULL の割り当てはこの変数の完全な初期化解除を意味します。

WRONG_VALUE　定数は、不正の列挙の値を返す時に必要です。例えば、戻り値がこの列挙の値であることを知らせる必要がある場合です。例として、名称で指定されたオブジェクトの線のスタイルを返す CheckLineStyle()　をみてみましょう。ObjectGetInteger()　によるスタイルチェックの結果が　true　の場合、ENUM_LINE_STYLE からの値が返されます。その他の場合　WRONG_VALUE が返されます。

```
void OnStart()
  {
   if(CheckLineStyle("MyChartObject")==WRONG_VALUE)
      printf("Error line style getting.");
```

© 2000–2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//|  名称で指定されたオブジェクトの線のスタイルを返す                |
//+------------------------------------------------------------------+
ENUM_LINE_STYLE CheckLineStyle(string name)
  {
   long style;
//---
   if(ObjectGetInteger(0,name,OBJPROP_STYLE,0,style))
      return((ENUM_LINE_STYLE)style);
   else
      return(WRONG_VALUE);
  }
```

WHOLE_ARRAY 定数は、処理された配列の要素数を指定することを必要とする関数を対象としています。

- ArrayCopy();
- ArrayMinimum();
- ArrayMaximum();
- FileReadArray();
- FileWriteArray().

指定された位置から配列の最後までの全ての値が処理されなければならないことを指定する場合は、WHOLE_ARRAY 値を指定する必要があります。

IS_PROFILE_MODE 定数は、プロファイルモードでデータが正しく収集されるようにプログラムの動作を変更することが出来ます。プロファイリングは個々のプログラムフラグメント（通常、関数を含む）の実行時間を測定し、ならびに呼び出しの数を計算することが出来ます。この例のように、プロファイリングモードで実行時間を決定するためにはSleep() 関数の呼び出しは無効にすることが出来ます。

```
//--- Sleep はプロファイリング結果に大影響（変化）を与えることがある
if(!IS_PROFILE_MODE) Sleep(100); // プロファイリングモードで Sleep() を無効にする
```

IS_PROFILE_MODE は従来のモードではゼロに設定される定数値で、コンパイル時にコンパイラによって設定されます。プロファイリングモードでプログラムを起動すると、特別なコンパイルが行われ、IS_PROFILE_MODE はゼロ以外の値に置き換えられます。

IS_DEBUG_MODE 定数はデバッグモードでの MQL5 プログラムの動作をわずかに変更する必要がある場合に役立ちます。例えば、デバッグモードでは、端末ログに追加のデバッグ情報を表示したり、チャートに追加のグラフィックオブジェクトを作成する必要があります。

次の例は、ラベルオブジェクトを作成し、スクリプトの実行モードに応じてその記述や色を設定します。MetaEditor でスクリプトをデバッグモードで実行するには、F5 を押します。端末でブラウザウィンドウからスクリプトを実行すると、オブジェクトラベルの色とテキストが異なります。

例：

```
//+------------------------------------------------------------------+
//|                                              Check_DEBUG_MODE.mq5 |
//|                        Copyright © 2009, MetaQuotes Software Corp. |
```

© 2000-2025, MetaQuotes Ltd.

```
//|                                             https://www.metaquotes.net |
//+------------------------------------------------------------------------+
#property copyright "Copyright © 2009, MetaQuotes Software Corp."
#property link      "https://www.metaquotes.net"
//+------------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                       |
//+------------------------------------------------------------------------+
void OnStart()
  {
//---
   string label_name="invisible_label";
   if(ObjectFind(0,label_name)<0)
     {
      Print("Object",label_name,"not found. Error code = ",GetLastError());
      //--- ラベルを作成する
      ObjectCreate(0,label_name,OBJ_LABEL,0,0,0);
      //--- X 座標を設定する
      ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,200);
      //--- Y 座標を設定する
      ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,300);
      ResetLastError();
      if(IS_DEBUG_MODE) // デバグモード
        {
         //--- スクリプト実行モードについてのメッセージを表示する
         ObjectSetString(0,label_name,OBJPROP_TEXT,"DEBUG MODE");
         //--- テキストの色を赤にする
         if(!ObjectSetInteger(0,label_name,OBJPROP_COLOR,clrRed))
            Print("Unable to set the color. Error",GetLastError());
        }
      else               // 操作モード
        {
         ObjectSetString(0,label_name,OBJPROP_TEXT,"RELEASE MODE");
         //--- テキストの色を不可視にする
         if(!ObjectSetInteger(0,label_name,OBJPROP_COLOR,CLR_NONE))
            Print("Unable to set the color. Error ",GetLastError());
        }
      ChartRedraw();
      DebugBreak();     // デバッグモードにでは、ここで終了する
     }
  }
```

クリプトメソッド

ENUM_CRYPT_METHOD 列挙は、データ変換方法を指定するために使用され CryptEncode() 及び CryptDecode() 関数で使用されます。

ENUM_CRYPT_METHOD

© 2000-2025, MetaQuotes Ltd.

定数	説明
CRYPT_BASE64	BASE64
CRYPT_AES128	128ビット（16バイト）キーでの AES 暗号化。
CRYPT_AES256	256 ビット（32 バイト）キーでの AES 暗号化。
CRYPT_DES	56ビット（7バイト）キーでの DES 暗号化。
CRYPT_HASH_SHA1	SHA1 HASH の計算。
CRYPT_HASH_SHA256	SHA256 HASH の計算。
CRYPT_HASH_MD5	MD5 HASH の計算。
CRYPT_ARCH_ZIP	ZIP アーカイブ。

参照

参照

DebugBreak、MQL5 プログラム実行プロパティ、CryptEncode()、CryptDecode()

© 2000–2025, MetaQuotes Ltd.

データ構造体

MQL5 言語は、12 つの構造体を事前定義します。

- MqlDateTime は日付と時刻を操作します。
- MqlParamは IndicatorCreate() 関数を使用して指標ハンドルを作成する際に入力パラメータを送信します。
- MqlRates は価格、ボリュームとスプレッドに関する情報を含み、履歴データの操作に使用されます。
- MqlBookInfo は板情報の取得に使用されます。
- MqlTradeRequest は取引操作のリクエスト作成に使用されます。
- MqlTradeCheckResult は用意された取引リクエストを送信する前にチェックするのに使用されます。
- MqlTradeResult はOrderSend() 関数で送られた取引リクエストへのサーバの応答を含みます。
- MqlTradeTransaction は取引トランザクションの記述を含みます。
- MqlTick は現在価格についての最も要求される情報を高速で取得するために設計されています。
- Economic calendar structures are used to obtain data on the economic calendar events sent to the MetaTrader 5 platform in real time. Economic calendar functions allow analyzing macroeconomic parameters immediately after new reports are released, since relevant values are broadcast directly from the source with no delay.

© 2000–2025, MetaQuotes Ltd.

MqlDateTime

このデータ型構造体は 9 個の int 型のフィールドを含みます。

```
struct MqlDateTime
  {
   int year;            // 年
   int mon;             // 月
   int day;             // 日
   int hour;            // 時間
   int min;             // 分
   int sec;             // 秒
   int day_of_week;     // 曜日(日曜が 0、月曜が 1、... 土曜が 6 )
   int day_of_year;     // 年の日番号(1月1日には、ゼロの数の値が割り当てられる)
  };
```

注意事項

3 月以降の DAY_OF_YEAR は、うるう年と通常の年とでは異なります。

例:

```
void OnStart()
  {
//---
   datetime date1=D'2008.03.01';
   datetime date2=D'2009.03.01';

   MqlDateTime str1,str2;
   TimeToStruct(date1,str1);
   TimeToStruct(date2,str2);
   printf("%02d.%02d.%4d, day of year = %d",str1.day,str1.mon,
          str1.year,str1.day_of_year);
   printf("%02d.%02d.%4d, day of year = %d",str2.day,str2.mon,
          str2.year,str2.day_of_year);
  }
/*  結果
   01.03.2008, day of year = 60
   01.03.2009, day of year = 59
*/
```

参照

TimeToStruct、構造体とクラス

© 2000-2025, MetaQuotes Ltd.

指標入力パラメータの構造体（MqlParam）

MqlParam 構造体は IndicatorCreate() 関数を使用してテクニカル指標ハンドルを作成する際の入力パラメータを提供するためにデザインされています。

```
struct MqlParam
  {
   ENUM_DATATYPE     type;                // 入力パラメータの型。値は ENUM_DATATYPE
   long              integer_value;       // 整数型を格納するフィールド
   double            double_value;        // double 型を格納するフィールド
   string            string_value;        // string 型を格納するフィールド
  };
```

全ての指標入力パラメータは、MqlParam 型の配列の形で送信され、この配列の各要素の *type* フィールドは、要素によって送信されるデータの型を指定します。指標値は、初めに、*type* フィールドに ENUM_DATATYPE のどちらの列挙値があるかによって各要素に適切なフィールドに配置する必要があります（*integer_value*、*double_value* 及び *string_value*）。

指標のタイプとして IND_CUSTOM 値が IndicatorCreate() 関数の第三パラメータで受け渡される場合、入力パラメータ配列の最初の要素の *type* フィールドは ENUM_DATATYPE 列挙の TYPE_STRING 値を含み、*string_value* フィールドはカスタム指標の名称を含まなければなりません。

© 2000–2025, MetaQuotes Ltd.

MqlRates

この構造体は、価格、ボリュームとスプレッドの情報を提供します。

```
struct MqlRates
  {
   datetime time;          // 期間開始時間
   double   open;          // 始値
   double   high;          // 期間中の最高値
   double   low;           // 期間中の最安値
   double   close;         // 終値
   long     tick_volume;   // ティックボリューム
   int      spread;        // スプレッド
   long     real_volume;   // 取引高
  };
```

例:

```
void OnStart()
  {
   MqlRates rates[];
   int copied=CopyRates(NULL,0,0,100,rates);
   if(copied<=0)
      Print("Error copying price data ",GetLastError());
   else Print("Copied ",ArraySize(rates)," bars");
  }
```

参照

CopyRates、時系列へのアクセス

© 2000-2025, MetaQuotes Ltd.

MqlBookInfo

板情報についての情報を提供します。

```
struct MqlBookInfo
  {
   ENUM_BOOK_TYPE    type;           // ENUM_BOOK_TYPE 列挙での注文の種類
   double            price;          // 価格
   long              volume;         // ボリューム
   double            volume_real;    // より正確なボリューム
  };
```

注意事項

MqlBookInfo 構造体は事前定義されているので、宣言と説明を必要としません。この構造体を使用するには、その型の変数を宣言するだけです。

DOM は一部のシンボルのみで使用出来ます。

例:

```
   MqlBookInfo priceArray[];
   bool getBook=MarketBookGet(NULL,priceArray);
   if(getBook)
     {
      int size=ArraySize(priceArray);
      Print("MarketBookInfo about ",Symbol());
     }
   else
     {
      Print("Failed to receive DOM for the symbol ",Symbol());
     }
```

参照

MarketBookAdd、MarketBookRelease、MarketBookGet、DOM での取引注文、データ型

© 2000–2025, MetaQuotes Ltd.

取引リクエスト構造体（MqlTradeRequest）

クライアント端末と注文確定操作を実行する取引サーバとの間の相互作用は、取引リクエストを用いて行われます。取引リクエストは取引を実行するために必要な全てのフィールドが含まれている MqlTradeRequest 構造体で表されます。リクエスト処理の結果は MqlTradeResult 型の構造体で表されます。

```
struct MqlTradeRequest
  {
   ENUM_TRADE_REQUEST_ACTIONS    action;          // 取引の種類
   ulong                         magic;           // エキスパートアドバイザーＩＤ(マジックナンバー)
   ulong                         order;           // 注文チケット
   string                        symbol;          // 取引シンボル
   double                        volume;          // 約定のための要求されたボリューム(ロット単位)
   double                        price;           // 価格
   double                        stoplimit;       // 注文のストップリミットレベル
   double                        sl;              // 注文の決済逆指値レベル
   double                        tp;              // 注文の決済指値レベル
   ulong                         deviation;       // リクエストされた価格からの可能な最大偏差
   ENUM_ORDER_TYPE               type;            // 注文の種類
   ENUM_ORDER_TYPE_FILLING       type_filling;    // 注文実行の種類
   ENUM_ORDER_TYPE_TIME          type_time;       // 注文期限切れの種類
   datetime                      expiration;      // 注文期限切れの時刻
(ORDER_TIME_SPECIFIED 型の注文)
   string                        comment;         // 注文コメント
   ulong                         position;        // Position ticket
   ulong                         position_by;     // The ticket of an opposite position
  };
```

フィールドの説明

フィールド	説明
action	取引操作の種類値は ENUM_TRADE_REQUEST_ACTIONS 列挙値の 1 つです。
magic	エキスパートアドバイザー ID。取引注文の分析処理の整理を可能にします。取引リクエストを送信する際に、各エキスパートアドバイザーには、固有の ID を設定することが出来ます。
order	注文チケット。未決注文を変更するために使用されます。
symbol	注文シンボル。注文の変更とポジション決済操作には必要ありません。
volume	リクエストされた注文ボリューム(ロット単位)。約定の出来高は注文実行の種類によって異なります。
price	注文が実行されなければならない値に達している価格。「成行」実行タイプのシンボルの成行注文(TRADE_ACTION_DEALのSYMBOL_TRADE_EXECUTION_MARKET)では価格の指定は必要ありません。

© 2000-2025, MetaQuotes Ltd.

フィールド	説明
stoplimit	Limit pending 注文が置かれる*価格* 値（この条件は必須です）。それまで未決注文は出されません。
sl	不利な値動きの際の決済逆指値。
tp	有利な値動きの際の決済指値。
deviation	ポイントで指定された最大価格偏差。
type	注文の種類。ENUM_ORDER_TYPE 列挙値の 1 つ。
type_filling	注文実行の種類。ENUM_ORDER_TYPE_FILLING 列挙値の 1 つ。
type_time	注文期限切れの種類。ENUM_ORDER_TYPE_TIME 列挙値の 1 つ。
expiration	注文期限切れの時刻（ORDER_TIME_SPECIFIED 型）。
comment	注文コメント。
position	ポジションチケット。ポジションの決済や変更をするときに、ポジションの一意義の認証の為に埋める必要があります。原則として、その結果としてポジションを保有する、注文のチケットに対応します。
position_by	反対ポジションのチケット。反対ポジション（同じ商品であるが反対側に保有するポジション）の決済時に使用されます。

ヘッジシステムでのポジションの変更または決済の時には、そのチケット（MqlTradeRequest::position）を必ず指定してください。ネッティングシステムでもチケットを指定する必要がありますが、ポジションの識別はシンボル名で行われます。

取引操作を実行する注文を送信するには、OrderSend() 関数の使用が必要です。各取引操作のためには、必須項目を指定する必要があります。任意のフィールドを指定することも出来ます。取引注文を送信するには 7 つの可能なケースがあります。

Request Execution

これは、「リクエスト実行モード」で注文を出すための取引注文です（リクエストされた価格での取引）。以下の 9 のフィールドを指定する必要があります。

- action
- symbol
- volume
- price
- sl
- tp
- deviation
- type
- type_filling

「magic」及び「comment」フィールドの指定も可能です。

即時実行

© 2000–2025, MetaQuotes Ltd.

これは、「即時実行モード」で注文を出すための取引注文です（現在価格での取引）。以下の 9 のフィールドを指定する必要があります。

- action
- symbol
- volume
- price
- sl
- tp
- deviation
- type
- type_filling

「magic」及び「comment」フィールドの指定も可能です。

マーケット実行

これは、「マーケット実行モード」で注文を出すための取引注文です。以下の 5 つのフィールドを指定する必要があります。

- action
- symbol
- volume
- type
- type_filling

「magic」及び「comment」フィールドの指定も可能です。

エクスチェンジ実行

これは、「エクスチェンジ実行モード」で注文を出すための取引注文です。以下の 5 つのフィールドを指定する必要があります。

- action
- symbol
- volume
- type
- type_filling

「magic」及び「comment」フィールドの指定も可能です。

買いポジションを持つ 取引操作TRADE_ACTION_DEALの例：

© 2000–2025, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456   // エキスパートアドバイザのMagicNumber
//+------------------------------------------------------------------+
//|  買いポジションの発注                                            |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- リクエストと結果の宣言と初期化
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- リクエストのパラメータ
   request.action   =TRADE_ACTION_DEAL;                  // 取引操作タイプ
   request.symbol   =Symbol();                           // シンボル
   request.volume   =0.1;                                // 0.1ロットのボリューム
   request.type     =ORDER_TYPE_BUY;                     // 注文タイプ
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_ASK); // 発注価格
   request.deviation=5;                                  // 価格からの許容偏差
   request.magic    =EXPERT_MAGIC;                       // 注文のMagicNumber
//--- リクエストの送信
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());     // リクエストの送信が失敗した場
合、エラーコードを出力する
//--- 操作に関する情報
   PrintFormat("retcode=%u  deal=%I64u  order=%
I64u",result.retcode,result.deal,result.order);
  }
//+------------------------------------------------------------------+
```

売りポジションを持つ 取引操作TRADE_ACTION_DEALの例：

```
#define EXPERT_MAGIC 123456   // エキスパートアドバイザのMagicNumber
//+------------------------------------------------------------------+
//|  売りポジションを持つ                                            |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- リクエストと結果の宣言と初期化
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- リクエストのパラメータ
   request.action   =TRADE_ACTION_DEAL;                  // 取引操作タイプ
   request.symbol   =Symbol();                           // シンボル
   request.volume   =0.2;                                // 0.2ロットのボリューム
   request.type     =ORDER_TYPE_SELL;                    // 注文タイプ
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_BID); // 発注価格
   request.deviation=5;                                  // 価格からの許容偏差
   request.magic    =EXPERT_MAGIC;                       // 注文のMagicNumber
//--- リクエストの送信
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());     // リクエストの送信が失敗した場
合、エラーコードを出力する
//--- 操作についての情報
   PrintFormat("retcode=%u  deal=%I64u  order=%
I64u",result.retcode,result.deal,result.order);
  }
//+------------------------------------------------------------------+
```

ポジションを決済する取引操作TRADE_ACTION_DEALの例：

© 2000-2025, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456   // エキスパートアドバイザのMagicNumber
//+------------------------------------------------------------------+
//| 全てのポジションを決済                                           |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 結果とリクエストの宣言
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // 保有ポジション数
//--- 全ての保有ポジションの取捨
   for(int i=total-1; i>=0; i--)
     {
      //--- 注文のパラメータ
      ulong
 position_ticket=PositionGetTicket(i);                                  // ポジショ
ンチケット
      string
position_symbol=PositionGetString(POSITION_SYMBOL);                    // シンボ
ル
      int    digits=(int)
SymbolInfoInteger(position_symbol,SYMBOL_DIGITS);          // 小数点以下の桁数
      ulong
 magic=PositionGetInteger(POSITION_MAGIC);                              // ポジショ
ンのMagicNumber
      double
volume=PositionGetDouble(POSITION_VOLUME);                             // ポジショ
ンボリューム
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)
PositionGetInteger(POSITION_TYPE);    // ポジションタイプ
      //--- ポジション情報の出力
      PrintFormat("#%I64u %s  %s  %.2f  %s [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  magic);
      //--- MagicNumberが一致している場合
      if(magic==EXPERT_MAGIC)
        {
         //--- リクエストと結果の値のゼロ化
         ZeroMemory(request);
         ZeroMemory(result);
         //--- 操作パラメータの設定
         request.action   =TRADE_ACTION_DEAL;        // 取引操作タイプ
         request.position =position_ticket;          // ポジションチケット
         request.symbol   =position_symbol;          // シンボル
         request.volume   =volume;                   // ポジションボリューム
         request.deviation=5;                        // 価格からの許容偏差
         request.magic    =EXPERT_MAGIC;             // ポジションのMagicNumber
         //--- ポジションタイプによる注文タイプと価格の設定
         if(type==POSITION_TYPE_BUY)
           {
            request.price=SymbolInfoDouble(position_symbol,SYMBOL_BID);
            request.type =ORDER_TYPE_SELL;
           }
         else
           {
            request.price=SymbolInfoDouble(position_symbol,SYMBOL_ASK);
            request.type =ORDER_TYPE_BUY;
```

© 2000-2025, MetaQuotes Ltd.

```
            }
         //--- 決済情報の出力
         PrintFormat("Close #%I64d %s %
s",position_ticket,position_symbol,EnumToString(type));
         //--- リクエストの送信
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // リクエストの送信に失敗した
場合、エラーコードを出力
         //--- 操作情報
         PrintFormat("retcode=%u  deal=%I64u  order=%
I64u",result.retcode,result.deal,result.order);
         //---
        }
     }
  }
//+------------------------------------------------------------------+
```

SL & TP 更新

決済逆指値及び/または決済指値のレベルを変更する取引注文です。以下の 4 つのフィールドを指定する必要があります。

- action
- symbol
- sl
- tp

保有ポジションのストップロスとテイクプロフィットの値を変更する 取引操作 TRADE_ACTION_SLTPの例：

© 2000–2025, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456  // エキスパートアドバイザのMagicNumber
//+------------------------------------------------------------------+
//| ポジションのストップロスとテイクプロフィットを変更する                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- リクエストと結果の宣言
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // 保有ポジション数
//--- 全ての保有ポジションを取捨
   for(int i=0; i<total; i++)
     {
      //--- 注文パラメータ
      ulong  position_ticket=PositionGetTicket(i);// ポジションチケット
      string position_symbol=PositionGetString(POSITION_SYMBOL); // シンボル
      int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS); // 小数点
以下の桁数
      ulong  magic=PositionGetInteger(POSITION_MAGIC); // ポジションのMagicNumber
      double volume=PositionGetDouble(POSITION_VOLUME);     // ポジションボリューム
      double sl=PositionGetDouble(POSITION_SL);  // ポジションのStop Loss
      double tp=PositionGetDouble(POSITION_TP);  // ポジションのTake Profit
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)
PositionGetInteger(POSITION_TYPE);  // ポジションタイプ
      //--- ポジション情報の出力
      PrintFormat("#%I64u %s  %s  %.2f  %s  sl: %s  tp: %s  [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  DoubleToString(sl,digits),
                  DoubleToString(tp,digits),
                  magic);
      //--- MagicNumberが一致し、ストップロスとテイクプロフィットが指定されていない場合
      if(magic==EXPERT_MAGIC && sl==0 && tp==0)
        {
```

© 2000-2025, MetaQuotes Ltd.

```
         //--- 現在の価格レートの計算
         double price=PositionGetDouble(POSITION_PRICE_OPEN);
         double bid=SymbolInfoDouble(position_symbol,SYMBOL_BID);
         double ask=SymbolInfoDouble(position_symbol,SYMBOL_ASK);
         int    stop_level=(int)
SymbolInfoInteger(position_symbol,SYMBOL_TRADE_STOPS_LEVEL);
         double price_level;
         //--- 現在の決済価格からの最小許容オフセットが指定されていない場合
         if(stop_level<=0)
            stop_level=150; // 現在の決済価格から150ポイントのオフセットを設定する
         else
            stop_level+=50; // 念の為オフセットレベルは(SYMBOL_TRADE_STOPS_LEVEL + 50)ポイント
と同じにする

         //--- Stop LossとTake Profitの値の計算と四捨五入
         price_level=stop_level*SymbolInfoDouble(position_symbol,SYMBOL_POINT);
          if(type==POSITION_TYPE_BUY)
            {
             sl=NormalizeDouble(bid-price_level,digits);
             tp=NormalizeDouble(bid+price_level,digits);
            }
          else
            {
             sl=NormalizeDouble(ask+price_level,digits);
             tp=NormalizeDouble(ask-price_level,digits);
            }
         //--- リクエストと結果の値のゼロ化
         ZeroMemory(request);
         ZeroMemory(result);
         //--- 操作パラメータの設定
         request.action  =TRADE_ACTION_SLTP; // 取引操作タイプ
         request.position=position_ticket;   // ポジションシンボル
         request.symbol=position_symbol;     // シンボル
         request.sl      =sl;                // ポジションのStop Loss
         request.tp      =tp;                // ポジションのTake Profit
         request.magic=EXPERT_MAGIC;         // MagicNumber позиции
         //--- 変更情報の出力
         PrintFormat("Modify #%I64d %s %
s",position_ticket,position_symbol,EnumToString(type));
         //--- リクエストの送信
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // リクエストの送信に失敗した
場合、エラーコードを出力する
         //--- 操作情報
         PrintFormat("retcode=%u  deal=%I64u  order=%
I64u",result.retcode,result.deal,result.order);
        }
     }
  }
//+------------------------------------------------------------------+
```

未決注文

未決注文をするための取引注文です。以下の 11 のフィールドを指定する必要があります。

- action
- symbol
- volume
- price

© 2000-2025, MetaQuotes Ltd.

- stoplimit
- sl
- tp
- type
- type_filling
- type_time
- expiration

「magic」及び「comment」フィールドの指定も可能です。

未決注文を設定する 取引操作TRADE_ACTION_PENDINGの例：

© 2000–2025, MetaQuotes Ltd.

```
#property description "未決注文設定の例"
#property script_show_inputs
#define EXPERT_MAGIC 123456                                    // エキスパートアドバイザの
MagicNumber
input ENUM_ORDER_TYPE orderType=ORDER_TYPE_BUY_LIMIT;   // 注文タイプ
//+------------------------------------------------------------------+
//|  未決注文の設定                                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- リクエストと結果の宣言と初期化
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- 未決注文の設定の為のパラメータ
   request.action   =TRADE_ACTION_PENDING;                                 // 取引操作タイ
プ
   request.symbol   =Symbol();                                             // シンボル
   request.volume   =0.1;                                                  // 0.1ロットのボ
リューム
   request.deviation=2;                                                    // 価格からの許
容偏差
   request.magic    =EXPERT_MAGIC;                                         // 注文の
MagicNumber
   int offset = 50;                                                        // 注文設定の為
の現在の価格からのオフセット(ポイント)
   double price;                                                           // 注文の発動価
格
   double point=SymbolInfoDouble(_Symbol,SYMBOL_POINT);                    // ポイントサイズ
   int digits=SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);                    // 小数点以下の
桁数(精度)
   //--- 操作タイプのチェック
   if(orderType==ORDER_TYPE_BUY_LIMIT)
     {
      request.type     =ORDER_TYPE_BUY_LIMIT;                             // 注文タイプ
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;           // 発注価格
      request.price    =NormalizeDouble(price,digits);                    // 正規化された
発注価格
     }
   else if(orderType==ORDER_TYPE_SELL_LIMIT)
     {
      request.type     =ORDER_TYPE_SELL_LIMIT;                            // 注文タイプ
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID)+offset*point;           // 発注価格
      request.price    =NormalizeDouble(price,digits);                    // 正規化された
発注価格
     }
   else if(orderType==ORDER_TYPE_BUY_STOP)
     {
      request.type =ORDER_TYPE_BUY_STOP;                                  // 注文タイプ
      price        =SymbolInfoDouble(Symbol(),SYMBOL_ASK)+offset*point;   // 発注価格
      request.price=NormalizeDouble(price,digits);                        // 正規化された
発注価格
     }
   else if(orderType==ORDER_TYPE_SELL_STOP)
     {
      request.type     =ORDER_TYPE_SELL_STOP;                             // 注文タイプ
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID)-offset*point;           // 発注価格
      request.price    =NormalizeDouble(price,digits);                    // 正規化された
発注価格
     }
   else Alert("この例は未決注文の設定のみを対象にしています");    // もし未決注文ではないものが選択されて
いる場合
```

© 2000-2025, MetaQuotes Ltd.

```
//--- リクエストの送信
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());                    // もしリクエスト
の送信に失敗した場合、エラーコードを出力する
//--- 操作情報
   PrintFormat("retcode=%u  deal=%I64u  order=%
I64u",result.retcode,result.deal,result.order);
  }
//+------------------------------------------------------------------+
```

未決注文の変更

未決注文を変更する取引注文です。以下の 7 つのフィールドを指定する必要があります

- action
- order
- price
- sl
- tp
- type_time
- expiration

未決注文の価格レベルを変更する取引操作TRADE_ACTION_MODIFYの例：

© 2000–2025, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456  // エキスパートアドバイザのMagicNumber
//+------------------------------------------------------------------+
//| 未決注文の変更                                                   |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- リクエストと結果の宣言と初期化
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   int total=OrdersTotal(); // 保有未決注文数
//--- 全ての保有未決注文を取捨
   for(int i=0; i<total; i++)
     {
      //--- 注文パラメータ
      ulong  order_ticket=OrderGetTicket(i);                                  //注文チケット
      string order_symbol=Symbol();                                           // シンボル
      int    digits=(int)SymbolInfoInteger(order_symbol,SYMBOL_DIGITS);       // 小数点以下の桁数
      ulong  magic=OrderGetInteger(ORDER_MAGIC);                              // 注文のMagicNumber
      double volume=OrderGetDouble(ORDER_VOLUME_CURRENT);                     // 現在の注文量
      double sl=OrderGetDouble(ORDER_SL);                                     // 現在の注文のStop Loss
      double tp=OrderGetDouble(ORDER_TP);                                     // 現在の注文のTake Profit
      ENUM_ORDER_TYPE type=(ENUM_ORDER_TYPE)OrderGetInteger(ORDER_TYPE); // 注文タイプ
      int offset = 50;                                                        // 発注の為の現在価格からのオフセット（ポイント）
      double price;                                                           // 注文発動価格
      double point=SymbolInfoDouble(order_symbol,SYMBOL_POINT);               // ポイントサイズ
      //--- 注文情報の出力
      PrintFormat("#%I64u %s  %s  %.2f  %s  sl: %s  tp: %s  [%I64d]",
                  order_ticket,
                  order_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  DoubleToString(sl,digits),
                  DoubleToString(tp,digits),
                  magic);
      //--- MagicNumberが一致し、Stop LossとTake Profitが設定されていない場合
      if(magic==EXPERT_MAGIC && sl==0 && tp==0)
        {
         request.action=TRADE_ACTION_MODIFY;                                  // 取引操作タイプ
         request.order = OrderGetTicket(i);                                   // 注文チケット
         request.symbol    =Symbol();                                         // シンボル
         request.deviation=5;                                                 // 価格からの許容オフセット
        //--- 価格レベルの設定、タイプに合わせた注文のTake ProfitとStop Loss
         if(type==ORDER_TYPE_BUY_LIMIT)
           {
            price = SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;
            request.tp = NormalizeDouble(price+offset*point,digits);
            request.sl = NormalizeDouble(price-offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                 // 正規化された発注価格
           }
```

© 2000-2025, MetaQuotes Ltd.

```
         else if(type==ORDER_TYPE_SELL_LIMIT)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_BID)+offset*point;
            request.tp = NormalizeDouble(price-offset*point,digits);
            request.sl = NormalizeDouble(price+offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                  // 正規
化された発注価格
           }
         else if(type==ORDER_TYPE_BUY_STOP)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_ASK)+offset*point;
            request.tp = NormalizeDouble(price+offset*point,digits);
            request.sl = NormalizeDouble(price-offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                  // 正規
化された発注価格
           }
         else if(type==ORDER_TYPE_SELL_STOP)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_BID)-offset*point;
            request.tp = NormalizeDouble(price-offset*point,digits);
            request.sl = NormalizeDouble(price+offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                  // 正規
化された発注価格
           }
         //---リクエストの送信
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // リクエストの送信に失敗した
場合、エラーコードを出力する
         //--- 操作情報
         PrintFormat("retcode=%u  deal=%I64u  order=%
I64u",result.retcode,result.deal,result.order);
         //--- リクエストと結果の値のゼロ化
         ZeroMemory(request);
         ZeroMemory(result);
        }
     }
  }
//+------------------------------------------------------------------+
```

未決注文の削除

未決注文を削除するための取引注文です。以下の 2 つのフィールドを指定する必要があります。

- action
- order

未決注文の削除をする取引操作TRADE_ACTION_REMOVEの例：

```
#define EXPERT_MAGIC 123456  // エキスパートアドバイザのMagicNumber
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  未決注文の削除                                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- リクエストと結果の宣言と初期化
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   int total=OrdersTotal(); // 保有未決注文数
//--- 全ての保有未決注文を取捨
   for(int i=total-1; i>=0; i--)
     {
      ulong  order_ticket=OrderGetTicket(i);                   // 注文チケット
      ulong  magic=OrderGetInteger(ORDER_MAGIC);               // 注文のMagicNumber
      //--- MagicNumberが一致している場合
      if(magic==EXPERT_MAGIC)
        {
         //--- リクエストと結果の値のゼロ化
         ZeroMemory(request);
         ZeroMemory(result);
         //--- 操作パラメータの設定
         request.action=TRADE_ACTION_REMOVE;                   // 取引操作タイプ
         request.order = order_ticket;                         // 注文チケット
         //--- リクエストの送信
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // リクエストの送信に失敗した
場合、エラーコードを出力する
         //--- 操作情報
         PrintFormat("retcode=%u  deal=%I64u  order=%
I64u",result.retcode,result.deal,result.order);
        }
     }
  }
//+------------------------------------------------------------------+
```

参照

構造体とクラス、取引の関数、注文プロパティ

© 2000-2025, MetaQuotes Ltd.

小切手リクエストの結果の構造体（MqlTradeCheckResult）

取引操作リクエストを取引サーバに送信する前に、チェックすることが推奨されます。このチェックはOrderCheck() 関数に、チェックするリクエストと MqlTradeCheckResult 構造体型の変数を渡すことによってなされます。チェックの結果はこの変数に書かれます。

```
struct MqlTradeCheckResult
  {
   uint          retcode;          // 返信コード
   double        balance;          // 約定実行後の残高
   double        equity;           // 約定実行後の株式
   double        profit;           // 変動利益
   double        margin;           // 証拠金必要条件
   double        margin_free;      // 余剰証拠金
   double        margin_level;     // 証拠金レベル
   string        comment;          // 返信コードコメント（エラーの説明）
  };
```

フィールドの説明

フィールド	説明
retcode	リターンコード。
balance	取引操作実行後の残高値。
equity	取引操作実行後の株式値。
profit	取引操作実行後の変動利益。
margin	取引操作に必要な証拠金。
margin_free	取引操作実行後に残される余剰証拠金。
margin_level	取引操作実行後に設定される証拠金レベル。
comment	返信コードコメント（エラーの説明）。

参照

取引リクエスト構造体、現在価格の構造体、OrderSend、OrderCheck

© 2000-2025, MetaQuotes Ltd.

取引リクエスト結果の構造体（MqlTradeResult）

取引リクエストの結果として、取引サーバは MqlTradeResult 型の特殊な事前定義された構造と取引リクエストの処理結果のデータを返します。

```
struct MqlTradeResult
  {
   uint     retcode;          // 操作のリターンコード
   ulong    deal;             // 実行された場合の 約定チケット
   ulong    order;            // 注文された場合のチケット
   double   volume;           // ブローカーによって確認された約定ボリューム
   double   price;            // ブローカーによって確認された約定価格
   double   bid;              // 現在の売値
   double   ask;              // 現在の買値
   string   comment;          // 操作に対するブローカーコメント（デフォルトは取引サーバの返したコードの記述）
   uint     request_id;       // ディスパッチの際に、端末によって設定されたリクエストID
   int      retcode_external; // 外部取引システムのリターンコード
  };
```

フィールドの説明

フィールド	説明
retcode	取引サーバのリターンコード。
deal	約定チケット（注文がなされた場合）。TRADE_ACTION_DEAL 型の取引操作で使用可能です。
order	注文チケット（注文された場合）。TRADE_ACTION_PENDING 型の取引操作で使用可能です。
volume	ブローカーによって確認された約定ボリューム。注文充填タイプによって変化します。
price	ブローカーによって確認された約定価格取引リクエストまたは取引操作の *deviation フィールドに依存します。*
bid	現在のマーケット売値（リクオート価格）。
ask	現在のマーケット買値（リクオート価格）。
comment	操作に対するブローカーコメント（デフォルトは取引サーバの返したコードの記述）。
request_id	取引サーバへの送信時に端末によって設定されたリクエストID。
retcode_external	外部取引システムが返したエラーコード。これらのエラーの種類と使用は取引操作が送信される外部取引システムとブローカーに依存します。

取引操作の結果は取引操作を実行するためにOrderSend()に 2 番目のパラメータとして渡される MqlTradeResult type の変数に返されます。

© 2000–2025, MetaQuotes Ltd.

端末は OrdersSend() または OrderSendAsync()関数を使用しての取引サーバへの送信時に request_id フィールドのリクエスト ID を変更します。端末は、取引サーバから実行されるトランザクションに関するメッセージを受信しOnTradeTransaction()関数による処理のためにパラメータとして以下の成分を含んで提出します。

- MqlTradeTransaction 構造体内での取引トランザクションの記述。
- OrderSend() または OrdersSendAsync() 関数から送信された取引リクエストの記述 。リクエスト ID は端末によって取引サーバに送信され、リクエストとその request_id は端末メモリに保存されます。
- 取引リクエストの実行結果は、リクエストのIDを request_id フィールドに含む MqlTradeResult 構造になります。

OnTradeTransaction() 関数は、3 つの入力パラメータを受信しますが、最後の 2 つは TRADE_TRANSACTION_REQUEST 型のトランザクションのみで分析されるべきです。他の全ての場合では、取引リクエストとその実行結果のデータは満たされません。パラメータ分析の例は取引リクエスト構造体をご参照ください。

サーバ送信時の端末による request_id の設定は主に OrderSendAsync() 非同期関数を操作するために導入されました。この識別子は、アクション(OrderSend または OrderSendAsync 関数の呼び出し)とOnTradeTransaction()に送信されたアクションの結果を関連付けます。

例:

```
//+------------------------------------------------------------------+
//|  結果処理と取引リクエストを送信する                                |
//+------------------------------------------------------------------+
bool MyOrderSend(MqlTradeRequest request,MqlTradeResult result)
  {
//--- 最後のエラーコードをゼロにリセットする
   ResetLastError();
//--- リクエストを送信する
   bool success=OrderSend(request,result);
//--- 失敗したら、理由を見つける
   if(!success)
     {
      int answer=result.retcode;
      Print("TradeLog: Trade request failed. Error = ",GetLastError());
      switch(answer)
        {
         //--- リクオート
         case 10004:
           {
            Print("TRADE_RETCODE_REQUOTE");
            Print("request.price = ",request.price,"   result.ask = ",
                  result.ask," result.bid = ",result.bid);
            break;
           }
         //--- 注文がサーバに受け入れられない
         case 10006:
           {
            Print("TRADE_RETCODE_REJECT");
```

© 2000-2025, MetaQuotes Ltd.

```
            Print("request.price = ",request.price,"   result.ask = ",
                  result.ask," result.bid = ",result.bid);
            break;
           }
         //--- 無効な価格
         case 10015:
           {
            Print("TRADE_RETCODE_INVALID_PRICE");
            Print("request.price = ",request.price,"   result.ask = ",
                  result.ask," result.bid = ",result.bid);
            break;
           }
         //--- 無効な SL 及び/または TP
         case 10016:
           {
            Print("TRADE_RETCODE_INVALID_STOPS");
            Print("request.sl = ",request.sl," request.tp = ",request.tp);
            Print("result.ask = ",result.ask," result.bid = ",result.bid);
            break;
           }
         //--- 無効なボリューム
         case 10014:
           {
            Print("TRADE_RETCODE_INVALID_VOLUME");
            Print("request.volume = ",request.volume,"   result.volume = ",
                  result.volume);
            break;
           }
         //--- 取引操作に不充分なメモリ
         case 10019:
           {
            Print("TRADE_RETCODE_NO_MONEY");
            Print("request.volume = ",request.volume,"   result.volume = ",
                  result.volume,"   result.comment = ",result.comment);
            break;
           }
         //--- 他の理由。サーバ応答コードを出力する
         default:
           {
            Print("Other answer = ",answer);
           }
        }
      //--- false を返して、取引リクエストの失敗結果を通知する
      return(false);
     }
//--- OrderSend() が true を返す。- 答えを繰り返す。
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

取引処理構造体（MqlTradeTransaction）

取引口座にいくつかの明確なアクションを実行すると、状態が変更されます。アクションの例は下記です。

- クライアント端末内の任意の MQL5 アプリケーションから OrderSend と OrderSendAsync 関数を使用して取引リクエストを送信し、取引が実行される。
- 端末のグラフィカルインターフェースを使用して取引リクエストを送信し、取引が実行される。
- サーバ上で未決注文と逆指値注文が執行される。
- 取引サーバ側で操作を行う。

これらのアクションの結果として以下の取引トランザクションが実行されます。

- 取引リクエストの処理
- 未執行注文の変更
- 注文履歴の変更
- 約定履歴の変更
- ポジションの変更

例えば、市場の買い注文を送信する際に、注文が処理され、買い注文が口座に記録されます。その後注文が実行されオープン注文の表から削除されて注文履歴に追加されます。約定履歴に追加され新しいポジションが作成されます。これらのアクションは全て取引トランザクションです。

取引処理の口座への適応のために、MQL5 では特別な OnTradeTransaction() ハンドラーが提供されています。ハンドラーの1 番目のパラメータは取引処理を記述する MqlTradeTransaction 構造体を取得します。

```
struct MqlTradeTransaction
  {
   ulong                     deal;             // 約定チケット
   ulong                     order;            // 注文チケット
   string                    symbol;           // 取引銘柄名
   ENUM_TRADE_TRANSACTION_TYPE type;           // 取引トランザクションの種類
   ENUM_ORDER_TYPE           order_type;       // 注文の種類
   ENUM_ORDER_STATE          order_state;      // 注文状態
   ENUM_DEAL_TYPE            deal_type;        // 約定の種類
   ENUM_ORDER_TYPE_TIME      time_type;        // 作用時間の 注文種類
   datetime                  time_expiration;  // 注文の期限
   double                    price;            // 価格
   double                    price_trigger;    // ストップリミット注文アクティベーションの価格
   double                    price_sl;         // 決済逆指値レベル
   double                    price_tp;         // 決済指値レベル
   double                    volume;           // ボリューム(ロット単位)
   ulong                     position;         // ポジションチケット
   ulong                     position_by;      // 反対ポジションのチケット
  };
```

フィールドの説明

© 2000–2025, MetaQuotes Ltd.

フィールド	説明
deal	約定チケット。
order	注文チケット。
symbol	トランザクションが実行される、取引シンボルの名称。
type	取引トランザクションの種類。値は ENUM_TRADE_TRANSACTION_TYPE 列挙値のいずれかです。
order_type	取引注文の種類。値は ENUM_ORDER_TYPE 列挙値のいずれかです。
order_state	取引注文の状態。値は ENUM_ORDER_STATE 列挙値のいずれかです。
deal_type	約定の種類。値は ENUM_DEAL_TYPE 列挙値のいずれかです。
time_type	期限切れ後の注文の種類。値は ENUM_ORDER_TYPE_TIME 列挙値のいずれかです。
time_expiration	未決注文の期限切れ日時（ORDER_TIME_SPECIFIED と ORDER_TIME_SPECIFIED_DAY 型の注文）。
price	価格。取引トランザクションの種類によって、注文価格、約定またはポジションの価格。
price_trigger	指値つきストップ注文（活性化）価格（ORDER_TYPE_BUY_STOP_LIMIT と ORDER_TYPE_SELL_STOP_LIMIT）。
price_sl	決済逆指値。取引トランザクションの種類によって、注文、約定、またはポジション。
price_tp	決済指値。取引トランザクションの種類によって、注文、約定、またはポジション。
ボリューム	ロットでのボリューム。取引トランザクションの種類によって、注文、約定、またはポジションの現在のボリューム。

position	トランザクションが影響を与えたポジションチケット。
position_by	反対ポジションのチケット。反対ポジション（同じ商品であるが反対側に保有するポジション）の決済時に使用されます。

受信したトランザクション分析の必須パラメータは**type** フィールドで指定された型です。例えば、トランザクションが TRADE_TRANSACTION_REQUEST タイプ（サーバによる取引リクエスト処理の結果が受信された）の場合、構造体の**type**フィールド のみが書き入れられています。他のフィールドは分析されません。この場合、下記のように、OnTradeTransaction() ハンドラーに渡された**request** 及び **result** 追加パラメータの分析が可能です。

取引操作タイプのデータによってで、取引口座の注文、ポジションや 約定の現在の状態を分析することが出来ます。サーバに送信された 1 つの取引リクエストが複数の新しいトランザクションを生成することがあることは、留意されるべきです。端末におけるこれらのトランザクションの到着の優先順位は保証されません。

MqlTradeTransaction 構造は、取引トランザクションの種類に応じて異なる方法で書き入れられています（ENUM_TRADE_TRANSACTION_TYPE）。

© 2000–2025, MetaQuotes Ltd.

TRADE_TRANSACTION_ORDER_* 及び TRADE_TRANSACTION_HISTORY_*

MqlTradeTransaction 構造内の次のフィールドは、オープン注文の処理に関連する取引（TRADE_TRANSACTION_ORDER_ADD、TRADE_TRANSACTION_ORDER_UPDATE 及び TRADE_TRANSACTION_ORDER_DELETE）と注文履歴（TRADE_TRANSACTION_HISTORY_ADD、TRADE_TRANSACTION_HISTORY_UPDATE、TRADE_TRANSACTION_HISTORY_DELETE）のために書き入れられています。

- order – 注文チケット。
- symbol – 注文銘柄名。
- type – 取引トランザクションの種類。
- order_type – 注文の種類。
- orders_state – 注文の現状態。
- time_type – 注文期限切れの種類。
- time_expiration – 注文期限切れの時刻（ORDER_TIME_SPECIFIED 及び ORDER_TIME_SPECIFIED_DAY 期限切れ型を持つ注文）。
- price – クライアントによって指定された注文価格。
- price_trigger – 指値つきストップ注文のストップ価格（ORDER_TYPE_BUY_STOP_LIMIT 及び ORDER_TYPE_SELL_STOP_LIMIT のみ）。
- price_sl – 決済逆指値（注文で指定されている場合）。
- price_tp – 決済指値（注文で指定されている場合）。
- volume – 現在の注文のボリューム。初回注文ボリュームは、HistoryOrders* 関数を使用して注文履歴で見つけることが出来ます。
- position – 注文実行の結果として保有、変更、決済されたポジションのチケット。成行注文の為にのみ記入されます。TRADE_TRANSACTION_ORDER_ADDの為には記入されません。
- position_by – 反対ポジションのチケット。反対ポジションを決済する注文(close by)の為にのみ記入されます。

TRADE_TRANSACTION_DEAL_*

MqlTradeTransaction 構造内の次のフィールドは取引ハンドルに関するトランザクションのために書き入れられています（TRADE_TRANSACTION_DEAL_ADD、TRADE_TRANSACTION_DEAL_UPDATE 及び TRADE_TRANSACTION_DEAL_DELETE）。

- deal – 約定チケット
- order – 約定が基づいている注文チケット
- symbol – 約定銘柄名
- type – 取引トランザクションの種類。
- deal_type – 約定の種類
- price – 約定価格
- price_sl – 決済逆指値（約定が基づいている注文で指定された場合に書き入れられる）
- price_tp – 決済指値（約定が基づいている注文で指定された場合に書き入れられる）
- volume – 約定ボリューム(ロット単位)
- position – 取引実行の結果、保有、変更、決済されたポジションのチケット。
- position_by – 反対ポジションのチケット。反対ポジション決済の取引(out by)の為にのみ記入されます。

TRADE_TRANSACTION_POSITION

以下の MqlTradeTransaction 構造内の次のフィールドは、取引の実行に関連しないポジション変更に関連する取引トランザクションのために満たされています（TRADE_TRANSACTION_POSITION）

- symbol – ポジション銘柄名

© 2000–2025, MetaQuotes Ltd.

- type – 取引トランザクションの種類。
- deal_type – ポジションの種類（DEAL_TYPE_BUY または DEAL_TYPE_SELL）
- price – 加重平均ポジションの始値
- price_sl – 決済逆指値
- price_tp – 決済指値
- volume – 変更された場合のポジションボリューム（ロット単位）
- position – ポジションチケット。

> 約定実行の結果としてのポジション変更（追加、変更または閉鎖）は、
> TRADE_TRANSACTION_POSITION トランザクションの発生にはつながりません。

TRADE_TRANSACTION_REQUEST

取引トランザクションの場合、取引リクエストがサーバに処理され結果が受け取られたという事実を説明するために、MqlTradeTransaction の 1 つのフィールド（TRADE_TRANSACTION_REQUEST）のみが書き込まれます。

- type – 取引トランザクションの種類。

> この様なトランザクションでは type フィールド（取引トランザクションの種類）のみを分析する必要があります。OnTradeTransaction 関数の第 2 また第 3 パラメータ（request と result）も追加データのために分析する必要があります。

例：

```
input int MagicNumber=1234567;

//--- CTrade 取引クラスを有効にしてクラス変数を宣言する
#include <Trade\Trade.mqh>
CTrade trade;
//--- 未決注文インストールと削除のフラグ
bool pending_done=false;
bool pending_deleted=false;
//--- 未決注文チケットはここに保存される
ulong order_ticket;
//+------------------------------------------------------------------+
//| エキスパート初期化に使用される関数                                |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- MagicNumber を設定して全ての注文をマークする
   trade.SetExpertMagicNumber(MagicNumber);
//--- 取引リクエストは OrderSendAsync() 関数を使用して非同期モードで送信される
   trade.SetAsyncMode(true);
//--- 変数をゼロで初期化する
   order_ticket=0;
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| エキスパートティック関数                                          |
```

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnTick()
  {
//---未決注文のインストール
   if(!pending_done)
     {
      double ask=SymbolInfoDouble(_Symbol,SYMBOL_ASK);
      double buy_stop_price=NormalizeDouble(ask+1000*_Point,(int)
SymbolInfoInteger(_Symbol,SYMBOL_DIGITS));
      bool res=trade.BuyStop(0.1,buy_stop_price,_Symbol);
      //--- BuyStop() 関数が成功
      if(res)
        {
         pending_done=true;
         //--- ctrade から送信リクエストした結果の取得
         MqlTradeResult trade_result;
         trade.Result(trade_result);
         //--- 送信されたリクエストの request_id を取得
         uint request_id=trade_result.request_id;
         Print("Request has been sent to set a pending order.
Request_ID=",request_id);
         //--- 注文チケットを保存する(CTrade への送信に非同期モードを使用する場合はゼロ)
         order_ticket=trade_result.order;
         //--- 完成。 OnTick() ハンドラーを早期終了する。
         return;
        }
     }
//--- 未決注文の削除
   if(!pending_deleted)
      //--- 追加のチェック
      if(pending_done && (order_ticket!=0))
        {
         //--- 未決注文の削除を試みる
         bool res=trade.OrderDelete(order_ticket);
         Print("OrderDelete=",res);
         //--- 削除リクエスト送信に成功
         if(res)
           {
            pending_deleted=true;
            //--- リクエストの実行結果を取得
            MqlTradeResult trade_result;
            trade.Result(trade_result);
            //--- 結果から request ＩＤ と取得
            uint request_id=trade_result.request_id;
            //--- 操作ログに表示
            Print("The request has been sent to delete a pending order
#",order_ticket,
                  ". Request_ID=",request_id,
                  "\r\n");
```

© 2000-2025, MetaQuotes Ltd.

```
            //--- リクエスト結果から注文チケットを修正
            order_ticket=trade_result.order;
           }
        }
//---
  }
//+------------------------------------------------------------------+
//| TradeTransaction 関数                                            |
//+------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction &trans,
                        const MqlTradeRequest &request,
                        const MqlTradeResult &result)
  {
//--- トランザクション型を列挙値として取得する
   ENUM_TRADE_TRANSACTION_TYPE type=(ENUM_TRADE_TRANSACTION_TYPE)trans.type;
//--- トランザクションが結果処理のリクエストの場合は、名称のみを表示する
   if(type==TRADE_TRANSACTION_REQUEST)
     {
      Print(EnumToString(type));
      //--- 取り扱つかわれたリクエストの文字列名を表示する
      Print("------------RequestDescription\r\n",RequestDescription(request));
      //--- リクエスト結果の記述を表示する
      Print("------------ResultDescription\r\n",TradeResultDescription(result));
      //--- OnTick() の次のハンドルで削除されるよう、チケットを保存する
      if(result.order!=0)
        {
         //--- 次の OnTick() の呼び出しでこの注文をチケットによって削除する
         order_ticket=result.order;
         Print(" Pending order ticket ",order_ticket,"\r\n");
        }
     }
   else // 別の種類のトランザクションの完全記述を表示する
//--- 受け取ったトランザクションの記述を操作ログに表示する
      Print("------------TransactionDescription\r\n",TransactionDescription(trans));

//---
  }
//+------------------------------------------------------------------+
//| トランザクションのテキスト形式の記述を返す                       |
//+------------------------------------------------------------------+
string TransactionDescription(const MqlTradeTransaction &trans)
  {
//---
   string desc=EnumToString(trans.type)+"\r\n";
   desc+="Symbol: "+trans.symbol+"\r\n";
   desc+="Deal ticket: "+(string)trans.deal+"\r\n";
   desc+="Deal type: "+EnumToString(trans.deal_type)+"\r\n";
   desc+="Order ticket: "+(string)trans.order+"\r\n";
   desc+="Order type: "+EnumToString(trans.order_type)+"\r\n";
```

© 2000-2025, MetaQuotes Ltd.

```
   desc+="Order state: "+EnumToString(trans.order_state)+"\r\n";
   desc+="Order time type: "+EnumToString(trans.time_type)+"\r\n";
   desc+="Order expiration: "+TimeToString(trans.time_expiration)+"\r\n";
   desc+="Price: "+StringFormat("%G",trans.price)+"\r\n";
   desc+="Price trigger: "+StringFormat("%G",trans.price_trigger)+"\r\n";
   desc+="Stop Loss: "+StringFormat("%G",trans.price_sl)+"\r\n";
   desc+="Take Profit: "+StringFormat("%G",trans.price_tp)+"\r\n";
   desc+="Volume: "+StringFormat("%G",trans.volume)+"\r\n";
//--- 取得した文字列を返す
   return desc;
  }
//+------------------------------------------------------------------+
//| 取引リクエストのテキスト形式の記述を返す                                  |
//+------------------------------------------------------------------+
string RequestDescription(const MqlTradeRequest &request)
  {
//---
   string desc=EnumToString(request.action)+"\r\n";
   desc+="Symbol: "+request.symbol+"\r\n";
   desc+="Magic Number: "+StringFormat("%d",request.magic)+"\r\n";
   desc+="Order ticket: "+(string)request.order+"\r\n";
   desc+="Order type: "+EnumToString(request.type)+"\r\n";
   desc+="Order filling: "+EnumToString(request.type_filling)+"\r\n";
   desc+="Order time type: "+EnumToString(request.type_time)+"\r\n";
   desc+="Order expiration: "+TimeToString(request.expiration)+"\r\n";
   desc+="Price: "+StringFormat("%G",request.price)+"\r\n";
   desc+="Deviation points: "+StringFormat("%G",request.deviation)+"\r\n";
   desc+="Stop Loss: "+StringFormat("%G",request.sl)+"\r\n";
   desc+="Take Profit: "+StringFormat("%G",request.tp)+"\r\n";
   desc+="Stop Limit: "+StringFormat("%G",request.stoplimit)+"\r\n";
   desc+="Volume: "+StringFormat("%G",request.volume)+"\r\n";
   desc+="Comment: "+request.comment+"\r\n";
//--- 取得した文字列を返す
   return desc;
  }
//+------------------------------------------------------------------+
//| リクエスト処理結果の記述ををテキスト形式で返す                              |
//+------------------------------------------------------------------+
string TradeResultDescription(const MqlTradeResult &result)
  {
//---
   string desc="Retcode "+(string)result.retcode+"\r\n";
   desc+="Request ID: "+StringFormat("%d",result.request_id)+"\r\n";
   desc+="Order ticket: "+(string)result.order+"\r\n";
   desc+="Deal ticket: "+(string)result.deal+"\r\n";
   desc+="Volume: "+StringFormat("%G",result.volume)+"\r\n";
   desc+="Price: "+StringFormat("%G",result.price)+"\r\n";
   desc+="Ask: "+StringFormat("%G",result.ask)+"\r\n";
   desc+="Bid: "+StringFormat("%G",result.bid)+"\r\n";
```

© 2000-2025, MetaQuotes Ltd.

```
   desc+="Comment: "+result.comment+"\r\n";
//--- 取得した文字列を返す
   return desc;
  }
```

参照

取引処理の種類、OnTradeTransaction()

© 2000-2025, MetaQuotes Ltd.

現在の価格を取得する構造体(MqlTick)

銘柄ごとの最新価格を保存する構造体です。現在の価格についての最もニーズのある情報を取得する為のものです。

```
struct MqlTick
  {
   datetime     time;          // 最新の価格更新の時間
   double       bid;           // 現在のBid価格
   double       ask;           // 現在のAsk価格
   double       last;          // 最後の取引の現在価格(Last)
   ulong        volume;        // 現在のLast価格の数量
   long         time_msc;      // ミリ秒単位の最新の価格更新の時間
   uint         flags;         // ティックフラグ
   double       volume_real;   // 現在のLast価格のより正確な数量
  };
```

MqlTick型の変数によって一回のSymbolInfoTick()変数の呼び出しでAsk、Bid、Last、Volumeの値を取得することができます。

各ティックは、前回のティックと比べデータが変わったかどうかに関わらず、いつも全てのパラメータが埋められます。ティックヒストリーで前回の数値を検索することなく、あらゆる瞬間の現在の価格状態をいつも見ることができます。例えば、ティックと共にBid価格だけ変わったが、構造体の中では新しい価格の他に、前回のAsk価格や数量など、他のパラメータも表示されます。

現在のティックのデータとどう変わったかを知るには、そのフラグを分析してください。

- TICK_FLAG_BID - ティックがBid価格を変更した
- TICK_FLAG_ASK - ティックがAsk価格を変更した
- TICK_FLAG_LAST - ティックが最終取引価格を変更した
- TICK_FLAG_VOLUME - ティックが数量を変更した
- TICK_FLAG_BUY - 買い取引の結果ティックが発生した
- TICK_FLAG_SELL - 売り取引の結果ティックが発生した

例：

```
void OnTick()
  {
   MqlTick last_tick;
//---
   if(SymbolInfoTick(Symbol(),last_tick))
     {
      Print(last_tick.time,": Bid = ",last_tick.bid,
            " Ask = ",last_tick.ask,"  Volume = ",last_tick.volume);
     }
   else Print("SymbolInfoTick() failed, error = ",GetLastError());
//---
  }
```

こちらもご参照ください

構造体とクラス、CopyTicks()、SymbolInfoTick()

© 2000-2025, MetaQuotes Ltd.

経済指標カレンダー構造体

このセクションでは、MetaTraderプラットフォームで直接利用可能な経済指標カレンダーを使用するための構造体について説明します。経済指標カレンダーは、マクロ経済指標、その発表日および重要度の説明を含む既製の「百科事典」です。関連するマクロ経済指標の値は、発行直後にメタトレーダープラットフォームに送信され、国、通貨、重要度ごとに必要な指標を視覚的に追跡できるタグとしてチャートに表示されます。

経済指標カレンダー関数を使用すると、必要な国/通貨ペアの観点から、カスタム重要度基準に従って、着信したイベントの自動分析を実行できます。

国の説明はMqlCalendarCountry構造体で設定されます。これはCalendarCountryById()およびCalendarCountries()関数で使用されます。

```
struct MqlCalendarCountry
  {
   ulong                id;              // 国ID(ISO 3166-1)
   string               name;            // 国名(現在のターミナルエンコーディング)
   string               code;            // 国コード名(ISO 3166-1 alpha-2)
   string               currency;        // 国の通貨コード
   string               currency_symbol; // 国の通貨の銘柄
   string               url_name;        // mql5.comウェブサイトURLに使用される国名
  };
```

イベントの説明はMqlCalendarEvent構造体で設定されます。これはCalendarEventById()、CalendarEventByCountry()、CalendarEventByCurrency()関数で使用されます。

```
struct MqlCalendarEvent
  {
   ulong                            id;           // イベントID
   ENUM_CALENDAR_EVENT_TYPE         type;         // ENUM_CALENDAR_EVENT_TYPE列挙対からのイベントタイプ
   ENUM_CALENDAR_EVENT_SECTOR       sector;       // イベントが関連する部門
   ENUM_CALENDAR_EVENT_FREQUENCY    frequency;    // イベントの頻度
   ENUM_CALENDAR_EVENT_TIMEMODE     time_mode;    // イベント時間モード
   ulong                            country_id;   // 国ID
   ENUM_CALENDAR_EVENT_UNIT         unit;         // 経済指標値の単位
   ENUM_CALENDAR_EVENT_IMPORTANCE   importance;   // イベントの重要度
   ENUM_CALENDAR_EVENT_MULTIPLIER   multiplier;   // 経済指標値の乗数
   uint                             digits;       // 小数点以下の桁数
   string                           source_url;   // イベント発表源のURL
   string                           event_code;   // イベントコード
   string                           name;         // イベント名(現在のターミナルエンコーディング)
  };
```

© 2000-2025, MetaQuotes Ltd.

イベント値はMqlCalendarValue構造体で設定されます。これはCalendarValueById()、CalendarValueHistoryByEvent()、CalendarValueHistory()、CalendarValueLastByEvent()、CalendarValueLast()関数で使用されます。

```
struct MqlCalendarValue
  {
   ulong                                id;                  // 値ID
   ulong                                event_id;            // イベントID
   datetime                             time;                // イベントの日時
   datetime                             period;              // イベント報告期間
   int                                  revision;            // 報告期間に関連して発表された指標の改訂
   long                                 actual_value;        // ppmでの実際の値(設定されていない場合はLONG_MIN)
   long                                 prev_value;          // ppmでの前の値(設定されていない場合はLONG_MIN)
   long                                 revised_prev_value;  // ppmでの改訂された前の値(設定されていない場合はLONG_MIN)
   long                                 forecast_value;      // ppmでの予測値(設定されていない場合はLONG_MIN)
   ENUM_CALENDAR_EVENT_IMPACT           impact_type;         // 為替レートへの潜在的影響
  //--- 値を確認する関数
   bool                              HasActualValue(void) const;    // actual_valueが設定されている場合はrueを返す
   bool                              HasPreviousValue(void) const;  // prev_valueが設定されている場合はrueを返す
   bool                              HasRevisedValue(void) const;   // revised_prev_value が設定されている場合はrueを返す
   bool                              HasForecastValue(void) const;  // forecast_valueが設定されている場合はrueを返す
  //--- 値を受け取る関数
   double                            GetActualValue(void) const;    // actual_valueを返す(値が設定されていない場合はnan)
   double                            GetPreviousValue(void) const;  // prev_valueを返す(値が設定されていない場合はnan)
   double                            GetRevisedValue(void) const;   // revised_prev_valueを返す(値が設定されていない場合はnan)
   double                            GetForecastValue(void) const;  // forecast_valueを返す(値が設定されていない場合はnan)
  };
```

MqlCalendarValue構造体は*actual_value*、*forecast_value*、*prev_value*、*revised_prev_value*フィールドの値を確認/設定するメソッドを備えています。値が指定されていない場合、フィールドにはLONG_MIN (-9223372036854775808)が格納されます。

これらのフィールドに格納されている値は100万倍されることに注意してください。CalendarValueById、CalendarValueHistoryByEvent、CalendarValueHistory、CalendarValueLastByEvent、CalendarValueLastのいずれかの関数を使用してMqlCalendarValueで値を受け取る場合、フィールド値がLONG_MINに等しいかどうかを確認する必要があります。フィールドに値が指定されている場合、値を取得するには、値を100万で

© 2000-2025, MetaQuotes Ltd.

除算する必要があります。値を取得するもう1つの方法は、MqlCalendarValue構造体の関数を使用して値を確認および取得することです。

カレンダーイベント処理の例：

```
//--- 整数ではなく実数値でカレンダーイベントを格納する構造体を作成する
struct AdjustedCalendarValue
  {
   ulong                                id;                 // 値ID
   ulong                                event_id;           // イベントID
   datetime                             time;               // イベントの日時
   datetime                             period;             // イベント報告期間
   int                                  revision;           // 報告期間に関連して発表された指標の改訂
   double                               actual_value;       // 実際の値
   double                               prev_value;         // 前の値
   double                               revised_prev_value; // 改訂された前の値
   double                               forecast_value;     // 予測値
   ENUM_CALENDAR_EVENT_IMPACT           impact_type;        // 為替レートへの潜在的影響
  };
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
//--- EUの国コード（ISO 3166-1 Alpha-2）
   string EU_code="EU";
//--- EUイベント値を取得する
   MqlCalendarValue values[];
//--- イベントを取得する期間の境界を設定する
   datetime date_from=D'01.01.2021';  // 2021のすべてのイベント
   datetime date_to=0;                // 0はまだ発生していないイベントを含む、すべての既知のイベントを意味する
//--- 2021以来のEUイベント履歴をリクエストする
   if(!CalendarValueHistory(values, date_from, date_to, EU_code))
     {
      PrintFormat("Error!Failed to get events for country_code=%s", EU_code);
      PrintFormat("Error code: %d", GetLastError());
      return;
     }
   else
      PrintFormat("Received event values for country_code=%s: %d",
                  EU_code, ArraySize(values));
//--- 操作ログに出力する配列サイズを減らす
   if(ArraySize(values)>5)
      ArrayResize(values, 5);
//--- 実際の値をチェックしたり変換したりせずに、イベント値をそのまま操作ログに出力する
   Print("Output calendar values as they are");
   ArrayPrint(values);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- フィールド値を確認し、実際の値に変換する
//--- 値を確認して取得するオプション1
   AdjustedCalendarValue values_adjusted_1[];
   int total=ArraySize(values);
   ArrayResize(values_adjusted_1, total);
//--- 確認と調整を行って値をコピーする
   for(int i=0; i<total; i++)
     {
      values_adjusted_1[i].id=values[i].id;
      values_adjusted_1[i].event_id=values[i].event_id;
      values_adjusted_1[i].time=values[i].time;
      values_adjusted_1[i].period=values[i].period;
      values_adjusted_1[i].revision=values[i].revision;
      values_adjusted_1[i].impact_type=values[i].impact_type;
      //--- 値を確認し、1, 000, 000で徐算する
      if(values[i].actual_value==LONG_MIN)
         values_adjusted_1[i].actual_value=double("nan");
      else
         values_adjusted_1[i].actual_value=values[i].actual_value/1000000.;

      if(values[i].prev_value==LONG_MIN)
         values_adjusted_1[i].prev_value=double("nan");
      else
         values_adjusted_1[i].prev_value=values[i].prev_value/1000000.;

      if(values[i].revised_prev_value==LONG_MIN)
         values_adjusted_1[i].revised_prev_value=double("nan");
      else
         values_adjusted_1
[i].revised_prev_value=values[i].revised_prev_value/1000000.;

      if(values[i].forecast_value==LONG_MIN)
         values_adjusted_1[i].forecast_value=double("nan");
      else
         values_adjusted_1[i].forecast_value=values[i].forecast_value/1000000.;
     }
   Print("The first method to check and get calendar values");
   ArrayPrint(values_adjusted_1);

//--- 値を確認して取得するオプション2
   AdjustedCalendarValue values_adjusted_2[];
   ArrayResize(values_adjusted_2, total);
//--- 確認と調整を行って値をコピーする
   for(int i=0; i<total; i++)
     {
      values_adjusted_2[i].id=values[i].id;
      values_adjusted_2[i].event_id=values[i].event_id;
      values_adjusted_2[i].time=values[i].time;
```

© 2000-2025, MetaQuotes Ltd.

```
      values_adjusted_2[i].period=values[i].period;
      values_adjusted_2[i].revision=values[i].revision;
      values_adjusted_2[i].impact_type=values[i].impact_type;
      //--- 値を確認して取得する
      if(values[i].HasActualValue())
         values_adjusted_2[i].actual_value=values[i].GetActualValue();
      else
         values_adjusted_2[i].actual_value=double("nan");

      if(values[i].HasPreviousValue())
         values_adjusted_2[i].prev_value=values[i].GetPreviousValue();
      else
         values_adjusted_2[i].prev_value=double("nan");

      if(values[i].HasRevisedValue())
         values_adjusted_2[i].revised_prev_value=values[i].GetRevisedValue();
      else
         values_adjusted_2[i].revised_prev_value=double("nan");

      if(values[i].HasForecastValue())
         values_adjusted_2[i].forecast_value=values[i].GetForecastValue();
      else
         values_adjusted_2[i].forecast_value=double("nan");
     }
   Print("The second method to check and get calendar values");
   ArrayPrint(values_adjusted_2);

//--- option 3 to get the values - without checks
   AdjustedCalendarValue values_adjusted_3[];
   ArrayResize(values_adjusted_3, total);
//--- 確認と調整を行って値をコピーする
   for(int i=0; i<total; i++)
     {
      values_adjusted_3[i].id=values[i].id;
      values_adjusted_3[i].event_id=values[i].event_id;
      values_adjusted_3[i].time=values[i].time;
      values_adjusted_3[i].period=values[i].period;
      values_adjusted_3[i].revision=values[i].revision;
      values_adjusted_3[i].impact_type=values[i].impact_type;
      //--- 値を確認せずに取得する
      values_adjusted_3[i].actual_value=values[i].GetActualValue();
      values_adjusted_3[i].prev_value=values[i].GetPreviousValue();
      values_adjusted_3[i].revised_prev_value=values[i].GetRevisedValue();
      values_adjusted_3[i].forecast_value=values[i].GetForecastValue();
     }
   Print("The third method to get calendar values - without checks");
   ArrayPrint(values_adjusted_3);
  }
/*
```

© 2000–2025, MetaQuotes Ltd.

```
   We have received event values for country_code=EU: 1051
  Output the calendar values as they are
         [id] [event_id]              [time]             [period] [revision]
[actual_value]         [prev_value] [revised_prev_value]     [forecast_value]
[impact_type] [reserved]
   [0] 144520  999500001 2021.01.04 12:00:00 2020.12.01 00:00:00          3
   55200000             55500000 -9223372036854775808             55500000
  2        ...
   [1] 144338  999520001 2021.01.04 23:30:00 2020.12.29 00:00:00          0
  143100000            143900000 -9223372036854775808 -9223372036854775808
  0        ...
   [2] 147462  999010020 2021.01.04 23:45:00 1970.01.01 00:00:00          0 -
9223372036854775808 -9223372036854775808 -9223372036854775808 -9223372036854775808
          0        ...
   [3] 111618  999010018 2021.01.05 12:00:00 2020.11.01 00:00:00          0
   11000000             10500000 -9223372036854775808             11000000
  0        ...
   [4] 111619  999010019 2021.01.05 12:00:00 2020.11.01 00:00:00          0
    3100000              3100000              3200000              3100000
  0        ...
  The first method to check and get calendar values
         [id] [event_id]              [time]             [period] [revision]
[actual_value] [prev_value] [revised_prev_value] [forecast_value] [impact_type]
   [0] 144520  999500001 2021.01.04 12:00:00 2020.12.01 00:00:00          3
55.20000     55.50000                  nan         55.50000             2
   [1] 144338  999520001 2021.01.04 23:30:00 2020.12.29 00:00:00          0
143.10000    143.90000                  nan             nan              0
   [2] 147462  999010020 2021.01.04 23:45:00 1970.01.01 00:00:00          0
  nan          nan                   nan             nan             0
   [3] 111618  999010018 2021.01.05 12:00:00 2020.11.01 00:00:00          0
11.00000     10.50000                  nan         11.00000             0
   [4] 111619  999010019 2021.01.05 12:00:00 2020.11.01 00:00:00          0
3.10000      3.10000              3.20000          3.10000             0
  The second method to check and get calendar values
         [id] [event_id]              [time]             [period] [revision]
[actual_value] [prev_value] [revised_prev_value] [forecast_value] [impact_type]
   [0] 144520  999500001 2021.01.04 12:00:00 2020.12.01 00:00:00          3
55.20000     55.50000                  nan         55.50000             2
   [1] 144338  999520001 2021.01.04 23:30:00 2020.12.29 00:00:00          0
143.10000    143.90000                  nan             nan              0
   [2] 147462  999010020 2021.01.04 23:45:00 1970.01.01 00:00:00          0
  nan          nan                   nan             nan             0
   [3] 111618  999010018 2021.01.05 12:00:00 2020.11.01 00:00:00          0
11.00000     10.50000                  nan         11.00000             0
   [4] 111619  999010019 2021.01.05 12:00:00 2020.11.01 00:00:00          0
3.10000      3.10000              3.20000          3.10000             0
  The third method to get calendar values - without checks
         [id] [event_id]              [time]             [period] [revision]
[actual_value] [prev_value] [revised_prev_value] [forecast_value] [impact_type]
   [0] 144520  999500001 2021.01.04 12:00:00 2020.12.01 00:00:00          3
55.20000     55.50000                  nan         55.50000             2
   [1] 144338  999520001 2021.01.04 23:30:00 2020.12.29 00:00:00          0
143.10000    143.90000                  nan             nan              0
```

© 2000–2025, MetaQuotes Ltd.

```
   [2] 147462  999010020 2021.01.04 23:45:00 1970.01.01 00:00:00          0
  nan          nan                  nan              nan             0
   [3] 111618  999010018 2021.01.05 12:00:00 2020.11.01 00:00:00          0
11.00000     10.50000                nan         11.00000             0
   [4] 111619  999010019 2021.01.05 12:00:00 2020.11.01 00:00:00          0
3.10000      3.10000              3.20000          3.10000             0
*/
```

イベント頻度はMqlCalendarEvent構造体で設定されます。可能な値はリストで設定されます。
ENUM_CALENDAR_EVENT_FREQUENCY

ID	説明
CALENDAR_FREQUENCY_NONE	Release frequency is not set
CALENDAR_FREQUENCY_WEEK	Released once a week
CALENDAR_FREQUENCY_MONTH	Released once a month
CALENDAR_FREQUENCY_QUARTER	Released once a quarter
CALENDAR_FREQUENCY_YEAR	Released once a year
CALENDAR_FREQUENCY_DAY	Released once a day

Event type is specified in the MqlCalendarEvent structure. 可能な値はリストで設定されます。
ENUM_CALENDAR_EVENT_TYPE

ID	説明
CALENDAR_TYPE_EVENT	Event (meeting, speech, etc.)
CALENDAR_TYPE_INDICATOR	Indicator
CALENDAR_TYPE_HOLIDAY	Holiday

A sector of the economy an event is related to is specified in the MqlCalendarEvent structure. 可能な値はリストで設定されます。**ENUM_CALENDAR_EVENT_SECTOR**

ID	説明
CALENDAR_SECTOR_NONE	Sector is not set
CALENDAR_SECTOR_MARKET	Market, exchange
CALENDAR_SECTOR_GDP	国内総生産（Gross Domestic Product、GDP）
CALENDAR_SECTOR_JOBS	Labor market
CALENDAR_SECTOR_PRICES	Prices
CALENDAR_SECTOR_MONEY	Money

© 2000–2025, MetaQuotes Ltd.

ID	説明
CALENDAR_SECTOR_TRADE	取引
CALENDAR_SECTOR_GOVERNMENT	Government
CALENDAR_SECTOR_BUSINESS	Business
CALENDAR_SECTOR_CONSUMER	Consumption
CALENDAR_SECTOR_HOUSING	Housing
CALENDAR_SECTOR_TAXES	Taxes
CALENDAR_SECTOR_HOLIDAYS	Holidays

Event importance is specified in the MqlCalendarEvent structure. 可能な値はリストで設定されます。**ENUM_CALENDAR_EVENT_IMPORTANCE**

ID	説明
CALENDAR_IMPORTANCE_NONE	Importance is not set
CALENDAR_IMPORTANCE_LOW	Low importance
CALENDAR_IMPORTANCE_MODERATE	Medium importance
CALENDAR_IMPORTANCE_HIGH	High importance

Measurement unit type used in displaying event values is specified in the MqlCalendarEvent structure. 可能な値はリストで設定されます。**ENUM_CALENDAR_EVENT_UNIT**

ID	説明
CALENDAR_UNIT_NONE	Measurement unit is not set
CALENDAR_UNIT_PERCENT	Percentage
CALENDAR_UNIT_CURRENCY	National currency
CALENDAR_UNIT_HOUR	Hours
CALENDAR_UNIT_JOB	Jobs
CALENDAR_UNIT_RIG	Drilling rigs
CALENDAR_UNIT_USD	USD
CALENDAR_UNIT_PEOPLE	People
CALENDAR_UNIT_MORTGAGE	Mortgage loans
CALENDAR_UNIT_VOTE	Votes
CALENDAR_UNIT_BARREL	Barrels

© 2000–2025, MetaQuotes Ltd.

ID	説明
CALENDAR_UNIT_CUBICFEET	Cubic feet
CALENDAR_UNIT_POSITION	Non-commercial net positions
CALENDAR_UNIT_BUILDING	Buildings

In some cases, economic parameter values require a multiplier set in the MqlCalendarEvent structure. Possible multiplier values are set in the listing **ENUM_CALENDAR_EVENT_MULTIPLIER**

ID	説明
CALENDAR_MULTIPLIER_NONE	Multiplier is not set
CALENDAR_MULTIPLIER_THOUSANDS	Thousands
CALENDAR_MULTIPLIER_MILLIONS	Millions
CALENDAR_MULTIPLIER_BILLIONS	Billions
CALENDAR_MULTIPLIER_TRILLIONS	Trillions

Event's potential impact on a national currency rate is indicated in the MqlCalendarValue structure. 可能な値はリストで設定されます。**ENUM_CALENDAR_EVENT_IMPACT**

ID	説明
CALENDAR_IMPACT_NA	Impact is not set
CALENDAR_IMPACT_POSITIVE	Positive impact
CALENDAR_IMPACT_NEGATIVE	Negative impact

Event time is specified in the MqlCalendarEvent structure. 可能な値はリストで設定されます。**ENUM_CALENDAR_EVENT_TIMEMODE**

ID	説明
CALENDAR_TIMEMODE_DATETIME	ソースはイベントの正確な時間を公開します
CALENDAR_TIMEMODE_DATE	終日イベント
CALENDAR_TIMEMODE_NOTIME	ソースはイベントの時間を公開しない
CALENDAR_TIMEMODE_TENTATIVE	ソースはイベントの日を公開するが時間を公開しません。時間はイベントの発生時に指定されます。

参照

© 2000–2025, MetaQuotes Ltd.

経済指標カレンダー

© 2000–2025, MetaQuotes Ltd.

エラーコードと警告コード

このセクションは、以下の説明を含みます。

- 取引サーバのリターンコード - OrderSend() 関数で送られた取引リクエストを分析
- コンパイラ警告 – コンパイル時に表示される（エラーでない）警告メッセージのコード
- コンパイルエラー – 失敗したコンパイル試行でのエラーメッセージ
- ランタイムエラー - GetLastError() 関数で取得出来る、MQL5 プログラム実行中のエラーコード

© 2000–2025, MetaQuotes Ltd.

取引サーバのリターンコード

取引操作実行リクエストの全ては OrderSend() 関数によって MqlTradeRequest 取引リクエスト構造体として送信されます。関数の実行結果は MqlTradeResult 構造体に収められ *retcode* フィールドは取引サーバのリターンコードを含みます。

コード	定数	説明
10004	TRADE_RETCODE_REQUOTE	リクオート。
10006	TRADE_RETCODE_REJECT	リクエストの拒否。
10007	TRADE_RETCODE_CANCEL	トレーダーによるリクエストのキャンセル。
10008	TRADE_RETCODE_PLACED	注文が出されました。
10009	TRADE_RETCODE_DONE	リクエスト完了。
10010	TRADE_RETCODE_DONE_PARTIAL	リクエストが一部のみ完了。
10011	TRADE_RETCODE_ERROR	リクエスト処理エラー。
10012	TRADE_RETCODE_TIMEOUT	リクエストが時間切れでキャンセル。
10013	TRADE_RETCODE_INVALID	無効なリクエスト。
10014	TRADE_RETCODE_INVALID_VOLUME	リクエスト内の無効なボリューム。
10015	TRADE_RETCODE_INVALID_PRICE	リクエスト内の無効な価格。
10016	TRADE_RETCODE_INVALID_STOPS	リクエスト内の無効なストップ。
10017	TRADE_RETCODE_TRADE_DISABLED	取引が無効化されています。
10018	TRADE_RETCODE_MARKET_CLOSED	市場が閉鎖中。
10019	TRADE_RETCODE_NO_MONEY	リクエストを完了するのに資金が不充分。
10020	TRADE_RETCODE_PRICE_CHANGED	価格変更。
10021	TRADE_RETCODE_PRICE_OFF	リクエスト処理に必要な相場が不在。
10022	TRADE_RETCODE_INVALID_EXPIRATION	リクエスト内の無効な注文有効期限。
10023	TRADE_RETCODE_ORDER_CHANGED	注文状態の変化。
10024	TRADE_RETCODE_TOO_MANY_REQUESTS	頻繁過ぎるリクエスト。
10025	TRADE_RETCODE_NO_CHANGES	リクエストに変更なし。
10026	TRADE_RETCODE_SERVER_DISABLES_AT	サーバが自動取引を無効化。
10027	TRADE_RETCODE_CLIENT_DISABLES_AT	クライアント端末が自動取引を無効化。
10028	TRADE_RETCODE_LOCKED	リクエストが処理のためにロック中。
10029	TRADE_RETCODE_FROZEN	注文やポジションが凍結。
10030	TRADE_RETCODE_INVALID_FILL	無効な注文充填タイプ。
10031	TRADE_RETCODE_CONNECTION	取引サーバに未接続。

© 2000–2025, MetaQuotes Ltd.

コード	定数	説明
10032	TRADE_RETCODE_ONLY_REAL	操作は、ライブ口座のみで許可。
10033	TRADE_RETCODE_LIMIT_ORDERS	未決注文の数が上限に達しました。
10034	TRADE_RETCODE_LIMIT_VOLUME	シンボルの注文やポジションのボリュームが限界に達しました。
10035	TRADE_RETCODE_INVALID_ORDER	不正または禁止された注文の種類。
10036	TRADE_RETCODE_POSITION_CLOSED	指定されたPOSITION_IDENTIFIER を持つポジションがすでに閉鎖。
10038	TRADE_RETCODE_INVALID_CLOSE_VOLUME	決済ボリュームが現在のポジションのボリュームを超過。
10039	TRADE_RETCODE_CLOSE_ORDER_EXIST	指定されたポジションの決済注文が既存。これは、ヘッジシステムでの作業中に発生する可能性があります。 • 反対のポジションを決済しようとしているときにそのポジションの決済注文が既に存在している場合 • ポジションを完全または部分的に決済しようとしているときに既存する決済注文と新しく出された決済注文の合計が現在のポジションボリュームを超えている場合
10040	TRADE_RETCODE_LIMIT_POSITIONS	アカウントに同時に存在するポジションの数は、サーバー設定によって制限されます。限度に達すると、サーバーは出された注文を処理するときにTRADE_RETCODE_LIMIT_POSITIONSエラーを返します。 これは、ポジション会計タイプによって異なる動作につながります。 • ネッティング – ポジションの数が考慮されます。 限度に達すると、プラットフォームはその実行によってポジションの数が増加する可能性がある新しい注文の発注を無効にします。実際には、プラットホームは、既にポジションを有する銘柄についてのみの発注を可能にします。 現在の未決注文は、実行によって現在のポジションの変更につながる可能性がありますがその数を増やすことはできないので考慮されません。 • ヘッジング – 未決注文のアクティブ化によって常に新しいポジションが開かれるため、未決注文はポジションとともに考慮されます。限度に達すると、プラットフォームは、成行注文と未決注文の両方での新しい発注を無効にします。

© 2000–2025, MetaQuotes Ltd.

コード	定数	説明
10041	TRADE_RETCODE_REJECT_CANCEL	未決注文アクティベーションリクエストは却下され、注文はキャンセルされます。
10042	TRADE_RETCODE_LONG_ONLY	銘柄に"Only long positions are allowed（買いポジションのみ）" (POSITION_TYPE_BUY)のルールが設定されているため、リクエストは却下されます。
10043	TRADE_RETCODE_SHORT_ONLY	銘柄に"Only short positions are allowed（売りポジションのみ）" (POSITION_TYPE_SELL)のルールが設定されているため、リクエストは却下されます。
10044	TRADE_RETCODE_CLOSE_ONLY	銘柄に"Only position closing is allowed（ポジション決済のみ）"のルールが設定されているため、リクエストは却下されます。
10045	TRADE_RETCODE_FIFO_CLOSE	取引口座に"Position closing is allowed only by FIFO rule（FIFOによるポジション決済のみ）"(ACCOUNT_FIFO_CLOSE=true)のフラグが設定されているため、リクエストは却下されます
10046	TRADE_RETCODE_HEDGE_PROHIBITED	口座で「単一の銘柄の反対のポジションは無効にする」ルールが設定されているため、リクエストが拒否されます。たとえば、銘柄に買いポジションがある場合、売りポジションを開いたり、売り指値注文を出すことはできません。このルールは口座がヘッジ勘定の場合(ACCOUNT_MARGIN_MODE=ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)のみ適用されます。

© 2000–2025, MetaQuotes Ltd.

コンパイラ警告

コンパイラ警告は、情報提供のみを目的に示され、エラーメッセージではありません。

コード	説明
21	datetime 文字列の日付が不完全です。
22	datetime 文字列内の日の数が誤っています。要件： 年 1970 <= X <= 3000 月 0 < X <= 12 日 0 < X <= 31/30/28 (29)... .
23	datetime 文字列内の時の数が誤っています。要件： 時 0 <= X <24 分 0 <= X <60
24	不正なRGB形式の色：RGB成分の 1 つが 0 未満または 255 を超えています。
25	エスケープシーケンスでの不明な文字。 知られた文字：¥n ¥r ¥t ¥¥ ¥' ¥" ¥X ¥x
26	関数のローカル変数が多すぎます(>512KB)。減らして下さい。
29	列挙がすでに定義されているので(複製)メンバは、既存の定義に追加されます。
30	マクロのオーバーライド。
31	変数が宣言されても使用されていません。
32	コンストラクタは void 型である必要があります。
33	デストラクタは void 型である必要があります。
34	定数が整数の範囲(X > _UI64_MAX || X <_I64_MIN)に収まらないので double 型に変換されます。
35	長すぎるHEX－16文字を超えており上桁のニブルが切り捨てられます。
36	HEX 文字列 "0x" にニブルがありません。
37	関数がなく何も実行されません。
38	初期化されなかった変数が使用されています。
41	関数には本体がなく呼ばれません。
43	型キャストでのデータ損失の可能性。例： int x = (double) z;
44	定数の変換時の(データの)精度損失。例： int x = M_PI
45	比較操作でのオペランドの符号の違い。例：(char) c1> (uchar) c2
46	関数のインポートの問題 － #importの宣言が必要です。または関数のインポートが閉じています。
47	記述が大きすぎます。余分な文字は、実行可能ファイルに含まれません。
48	宣言された指標バッファの数が必要以上に小さいです。

© 2000–2025, MetaQuotes Ltd.

コード	説明
49	指標でグラフィクシリーズをプロットする色がありません。
50	指標を描くグラフィクシリーズがありません。
51	「OnStart」処理関数がスクリプトにありません。
52	「OnStart」処理関数が不正なパラメータで定義されています。
53	「OnStart」関数はスクリプトのみで定義されます。
54	「OnInit」関数が不正なパラメータで定義されています。
55	「OnInit」関数はスクリプトでは使用されません。
56	「OnDeinit」関数が不正なパラメータで定義されています。
57	「OnDeinit」関数はスクリプトでは使用されません。
58	2つの「OnCalculate」関数が定義されています。1つの価格配列の OnCalculate() が使用されます。
59	複雑な整数定数を計算する際に過充填が検出されました。
60	変数が多分初期化されていません。.
61	この宣言は、指定された行で宣言されたローカル変数の参照を不可能にします。
62	この宣言は、指定された行で宣言されたグローバル変数の参照を不可能にします。
63	静的に割り当てられた配列では使用出来ません。
64	この変数の宣言は定義済み変数を隠します。
65	式の値は常に true/false です。
66	数学演算で変数及び bool 型の式を利用するのは安全でありません。
67	符号なしの ulong 型に単項マイナス演算子を適用した結果は定義されていません。
68	#property version プロパティで指定されたバージョンが Market で受け入れられません。セクション。#property バージョン ID の正しい形式は「XXX.YYY」です。
69	空の制御ステートメントが見つかりました。
70	イベント処理関数の宣言時の、無効な関数の戻り値の型または無効なパラメータ。
71	1つの型への暗黙的な構造体のキャストが必要です。
72	この宣言は指定された文字列で宣言されたクラスメンバへの直接アクセスを不可能にします。アクセスはスコープ定義演算子 :: でのみ可能になります。
73	バイナリ定数が大きすぎます。上位桁が切り捨てられます。
74	継承したクラスのメソッドのパラメータが異なる const 修飾子を持ち、派生関数が親関数をオーバーロードしました。
75	ビット単位のシフト演算でシフトが負か多すぎます。実行結果は定義されていません。
76	関数は値を返す事が必要です。

© 2000–2025, MetaQuotes Ltd.

コード	説明
77	void 関数が値を返します。
78	値を返さない制御パスがあります。
79	グローバルスコープでの式は禁止です。
80	エラーを見つけるには演算子の優先順位をチェックします。　優先順位を明確にするために括弧を使用します。
81	2 つの OnCalCulate() が定義されています。OHLC バージョンが使用されます。
82	構造体にメンバがなく、サイズが 1 バイトとされます。
83	関数の戻り値をチェックするべきです。
84	リソース指標は、デバッグ用にコンパイルされます。これでパフォーマンスが低下します。パフォーマンスを向上するためには指標を再コンパイルしてください。
85	文字列内に大きすぎる文字コードがあります。コードは 0～65535 の範囲でなければなりません。
86	文字列に認識出来ない文字があります。
87	指標ウィンドウプロパティが定義されていません（表示をメインウィンドウまたはサブウィンドウへ設定）。プロパティ #property indicator_chart_window が適用されます。

© 2000–2025, MetaQuotes Ltd.

コンパイルエラー

MetaEdtior 5 は、コンパイル時に内蔵のコンパイラによって検出されたプログラムエラーに関するエラーメッセージを表示します。エラーリストは下記の表にあります。ソースコードを実行可能なバイナリにコンパイルするには、F7 を押します。エラーを含むプログラムは、コンパイラによって識別されるエラーが解消されるまでコンパイル出来ません。

コード	説明
100	ファイル読み込みエラー。
101	*.EX5 を書き込みのために開く際のエラー。
103	コンパイル完了にメモリが不充分。
104	コンパイラが認識出来ない空の構文単位。
105	#include での不正なファイル名。
106	#include でのファイルアクセスエラー（おそらくファイルが存在しない）。
108	#define での不正な識別子。
109	プリプロセッサの不明なコマンド（有効なものは #include、#define、#property、#import）。
110	コンパイラへ未知のシンボル。
111	実装されていない関数（宣言は存在するが、本体がない）。
112	二重引用符（"）の省略。
113	オープニングアングルブラケット（<）または二重引用符（"）の省略。
114	シングルクォート（'）の省略。
115	クロージングアングルブラケット（>）の省略。
116	型が宣言で指定されていません。
117	return 演算子が不在、または return が実装のいくつかのブランチで不在。
118	コールパラメータのオープニングブラケットが期待されています。
119	EX5 書き込みエラー。
120	配列への不正アクセス。
121	この関数は void 型でないので return 演算子が値を返さなければなりません。
122	不正なデストラクタの宣言。
123	コロン（:）が欠落。
124	変数がすでに宣言されています。
125	同一の識別子を持つ変数がすでに宣言されています。
126	長すぎる変数名（>250文字）。
127	同一の識別子を持つ構造体がすでに宣言されています。

© 2000-2025, MetaQuotes Ltd.

コード	説明
128	構造体が定義されていません。
129	同名の構造体メンバがすでに定義されています。
130	そのような構造体メンバがありません。
131	ブラケットが正しくペアされていません。
132	オープニングブラケット（ (）が期待されています。
133	片ちんばな中括弧（ { ）が不在）。
134	コンパイルが困難（あまりにも多くの分岐、内部スタックレベルが過剰に満たされています）。
135	ファイルを読まれるために開く際にエラー発生。
136	メモリにソースファイルをダウンロードするために不充分なメモリ。
137	変数が期待されています。
138	リファレンスの初期化は不可能。
140	代入が期待されています（宣言に表示されます）。
141	オープニングブレース（ { ）が期待されています。
142	動的配列のパラメータのみが可能です。
143	void 型の使用は不可能。
144	片ちんばの「)」または「]」（「(」または「[」が不在）。
145	片ちんばの「(」または「 [」（「)」または「[] 」が不在）。
146	不正な配列サイズ。
147	パラメーターが多すぎます（>64）。
149	このトークンは、ここでは期待されていません。
150	操作の使用が不正（無効な演算子）。
151	void 型の式は不可能。
152	演算子が期待されています。
153	break の誤用。
154	セミコロン（ ; ）が期待されています。
155	コンマ（ , ）が期待されています。
156	構造体ではなくクラス型が必要。
157	式が期待されています。
158	HEXに「非HEX文字」が存在するか、または、数が長すぎます（>511桁の数）。
159	文字列定数に65,534以上の文字が含まれています。

© 2000–2025, MetaQuotes Ltd.

コード	説明
160	関数定義はここでは受け入れ不能。
161	プログラムの予期しない終了。
162	構造体の前方宣言は禁止されています。
163	この名称の関数は、すでに定義され、別型の戻り値を持っています。
164	この名称の関数は、すでに定義され、異なるパラメータセットを持っています。
165	この名称の関数は、すでに定義され、実装されています。
166	このコールのための関数オーバーロードが見つかりませんでした。
167	void 型の戻り値を持つ関数は値を返すことは出来ません。
168	関数が定義されていません。
170	値が期待されています。
171	*case* 式内では整数定数のみが有効。
172	この *case* の値はこの *switch* で既に使用されています。
173	整数が期待されています。
174	#import ではファイル名が期待されています。
175	式は、グローバルレベルでは許可されていません。
176	「;」の前の「)」が省略されています。
177	= の左側には変数が期待されています。
178	式の結果が不使用。
179	*case* 内での変数の宣言は不可能。
180	文字列から数への暗黙的な変換。
181	数から文字列への暗黙的な変換。
182	オーバーロードされた関数のあいまいな呼び出し（複数の候補）。
183	*if* なしの不正な *else* 。
184	*switch* なしの不正な *case* または *default* 。
185	省略記号の不適切な使用。
186	初期化シーケンスが初期化された変数より多くの要素を持っています。
187	*case* の定数が期待されています。
188	定数式が必要。
189	定数変数の変更は不可能。
190	クロージングブラケットまたはコンマが期待されています（配列メンバの宣言）。

© 2000–2025, MetaQuotes Ltd.

コード	説明
191	列挙子識別子は、すでに定義されています。
192	列挙は、アクセス修飾子を持つことは出来ません（const、extern、static）。
193	列挙型メンバが、すでに別の値を使用して宣言されています。
194	同名で定義された変数があります。
195	同名で定義された構造体があります。
196	列挙型メンバの名称が期待されています。
197	整数式が期待されています。
198	定数式でのゼロによる除算。
199	関数のパラメータの数が間違っています。
200	参照によるパラメータは変数でなければなりません。
201	参照の引き渡しには同じ型の変数が期待されます。
202	定数変数は、非定数参照で引渡しすることは出来ません。
203	正の整数定数が必要です。
204	protected クラスメンバへのアクセスに失敗しました。
205	Import は、すでに別の方法で定義されています。
208	実行可能のファイルが作成されていません。
209	指標の「OnCalculate」エントリポイントが見つかりません。
210	continue 演算子はループ内のみで使用可能です。
211	private（closed）クラスメンバへのアクセスエラー。
213	構造体またはクラスのメソッドが宣言されていません。
214	private（closed）クラスメソッドへのアクセスエラー。
216	オブジェクトを持つ構造体のコピーは許可されていません。
218	配列の範囲外のインデックス。
219	配列は構造体またはクラス宣言での初期化出来ません。
220	クラスコンストラクタはパラメータを持つことが出来ません。
221	クラスデストラクタはパラメータを持つことが出来ません。
222	同じ名称とパラメータを持つクラスメソッドまたは構造が、すでに宣言されています。
223	演算子が期待されています。
224	同名のクラスメソッドまたは構造体が存在するが、パラメータが異なります（宣言と実装が異なります）。
225	インポートされた関数が記載されていません。

© 2000–2025, MetaQuotes Ltd.

コード	説明
226	ZeroMemory() は protected メンバや継承を持つクラスでは使用できません。
227	オーバーロードされた関数のあいまいな呼び出し（複数の候補）。
228	変数名が期待されています。
229	この場所での参照の宣言は不可能です。
230	すでに列挙名として使用されています。
232	クラスまたは構造体が期待されています。
235	配列の削除には「delete」演算子の使用は不可能です。
236	「while」演算子が期待されています。
237	「delete」演算子はポインタが必要です。
238	この「switch」にはすでに「default」があります。
239	構文エラー。
240	エスケープシーケンスは、文字列でのみ発生出来ます（「¥」で始まる）。
241	配列が必要です。角括弧'['配列には適用されない、または非配列は配列パラメータとして渡されます。
242	初期化シーケンスでの初期化は不可能です。
243	Import が定義されていません。
244	構文木のオプティマイザエラー。
245	非常に多くの構造体が宣言されています（プログラムを簡略化してください）。
246	パラメータの変換は許可されていません。
247	「delete」演算子の不正な使用。
248	参照へのポインタの宣言は許可されていません。
249	参照への参照の宣言は許可されていません。
250	ポインタへのポインタの宣言は許可されていません。
251	パラメータリストで構成体の宣言は許可されていません。
252	型キャストの無効な操作。
253	ポインタは、クラスまたは構造体のみに宣言することが出来ます。
256	宣言されていない識別子。
257	実行可能なコード最適化エラー。
258	実行可能なコード生成エラー。
260	「スイッチ」演算子の無効な式。
261	文字列定数のプールが溢れたので、プログラムを簡素化してください。

© 2000–2025, MetaQuotes Ltd.

コード	説明
262	列挙体への変換は不可能。
263	データ(クラスまたは構造体のメンバ)に「 virtual 」は使用しないでください。
264	クラスの protected メソッドの呼び出しは不可能です。
265	オーバーライドされた仮想関数が、別の型を返します。
266	クラスの構造体からの継承は出来ません。
267	構造体のクラスからの継承は出来ません。
268	コンストラクタは virtual になれません(*virtual* 指定子は使用できません)。
269	構造体は、仮想メソッドを持つことが出来ません。
270	関数には本体が必要です。
271	システム関数(端末関数)のオーバーロードは禁止されています。
272	*Const* 指定子はクラスまたは構造体メンバ関数以外の関数では無効です。
274	constant メソッドのクラスメンバの変更は不可能です。
276	不適切な初期化シーケンス。
277	パラメータ初期値が不在(デフォルトパラメータの特定の宣言)。
278	デフォルトパラメータのオーバーライド(宣言と実装で異なる値)。
279	constant オブジェクトの非 constant メソッド呼び出しは禁止されています。
280	メンバアクセスにオブジェクトが必要(非クラス/構造にはポインタが設定されています)。
281	すでに宣言された構造体の名称は、宣言で使用することは出来ません。
284	(閉鎖した継承時の)不正な変換。
285	構造体と配列は、入力変数として使用することは出来ません。
286	*Const* 指定子は、コンストラクタ/デストラクタでは無効です。
287	datetime での不正な文字列式。
288	未知のプロパティ(#property)。
289	不正なプロパティ値。
290	#property での無効なプロパティインデックス。
291	Call のパラメータの省略 -<func(x,)>。
293	オブジェクトは参照によって渡される必要があります。
294	配列は参照によって渡される必要があります。
295	関数がエクスポート可能として宣言されました。
296	関数がエクスポート可能として宣言されませんでした。

© 2000-2025, MetaQuotes Ltd.

コード	説明
297	インポートされた関数のエクスポートは禁止されています。
298	インポートされた関数はこのパラメータを持つことが出来ません（ポインタ及び動的配列、ポインタ、クラスなどを含むクラスまたは構造体を渡すことは禁止）。
299	クラスが必要です。
300	#import が閉じられませんでした。
302	型の不一致。
303	Extern 変数は既に初期化せれています。
304	エクスポートされた関数またはエントリーポイント が見つかりません。
305	明示的なコンストラクタの呼び出しは禁止です。
306	メソッドは constant として宣言されました。
307	メソッドは constant として宣言されませんでした。
308	リソースファイルの不正なサイズ。
309	不正なリソース名。
310	リソースファイルのオープンエラー。
311	リソースファイルの読み込みエラー。
312	未知のリソースタイプ。
313	リソースファイルへの不正なパス。
314	指定されたリソース名がすでに使用されています。
315	関数のようなマクロには引数が期待されます。
316	マクロ定義での予期しないシンボル。
317	マクロの仮パラメータでのエラー。
318	マクロのパラメータ数が無効。
319	マクロのパラメータが多すぎます。
320	マクロが複雑過ぎるので簡素化してください。
321	EnumToString() のパラメータは列挙のみです。
322	長すぎるリソース名。
323	サポートされていない画像形式（24ビットまたは32ビットの色深度を持つBMPのみがサポートされています）。
324	配列の演算子での宣言は出来ません。
325	関数はグローバルスコープなどで宣言出来ます。
326	現在のスコープでの宣言は出来ません。

© 2000–2025, MetaQuotes Ltd.

コード	説明
327	ローカル変数の値を持つ静的変数の初期化は禁止です。
328	デフォルトコンストラクタを持たないオブジェクトの配列の不正宣言。
329	初期化リストはコンストラクタのみで使用可能です。
330	初期化リストの後に関数定義がありません。
331	空の初期化リスト。
332	コンストラクタでの配列の初期化は禁止です。
333	初期化リストでの親クラスのメンバの初期化は禁止です。
334	integer 型の式が期待されています。
335	配列に必要なメモリが最大値を超えます。
336	構造体に必要なメモリが最大値を超えます。
337	グローバルレベルで宣言された変数に必要なメモリが最大値を超えます。
338	ローカル変数に必要なメモリが最大値を超えます。
339	コンストラクタが定義されていません。
340	無効なアイコンファイル名。
341	指定されたパスでアイコンファイルが開けません。
342	アイコンファイルが不正でICOフォーマットでありません。
343	初期化リストを使用したクラス/構造体のコンストラクタでメンバの再初期化。
344	コンストラクタの初期化リストでの static メンバの初期化は禁止です。
345	グローバルレベルでのクラス/構造体の非静的 メンバは 禁止です。
346	クラス/構造メソッドの名称が、以前に宣言されたメンバの名称に一致します。
347	クラス/構造メソッドの名称が、以前に宣言されたメソッドの名称に一致します。
348	Virtual 関数は staticとして宣言出来ません。
349	const 修飾子は static 関数には使用できません。
350	コンストラクタ及びデストラクタは static になれません。
351	クラスまたは構造体の非静的 メンバ及びメソッドは static 関数からアクセス出来ません。
352	オーバーロード演算子（+、-、[]、++、— 等）は operator キーワードの後ろに期待されます。
353	MQL5 ではオーバーロード出来ない演算子もあります。
354	定義と宣言が一致しません。
355	演算子に無効なパラメータ数が指定されています。

© 2000–2025, MetaQuotes Ltd.

コード	説明
356	イベント処理の関数が見つかりません。
357	メソッドはエキスポート出来ません。
358	constant オブジェクトへのポインタは、非定数オブジェクトによって正規化出来ません。
359	クラステンプレートはまだサポートされていません。
360	関数テンプレートオーバーロードはまだサポートされていません。
361	関数テンプレートは適用出来ません。
362	関数テンプレートでのあいまいなパラメータ（複数のパラメータ型の適用が可能）。
363	関数テンプレートの引数を正規化するパラメータの型を決定することが出来ません。
364	関数テンプレート内の不正なパラメータ数。
365	関数テンプレートは virtualではいけません。
366	関数テンプレートはエキスポート出来ません。
367	関数テンプレートはインポート出来ません。
368	構造体はオブジェクトを含めません。
369	文字列配列や構造体はオブジェクトを含めません。
370	静的クラス/構造体のメンバは明示的に初期化される必要があります。
371	コンパイラの制限： 文字列は、65,535 を超える文字を含むことは出来ません。
372	一貫性の欠けた #ifdef/#endif。
373	コピーコンストラクタが見つからないので、クラスオブジェクトを返すことは出来ません。
374	非静的メンバ及びメソッドを使用することが出来ません。
375	OnTesterInit() は OnTesterDeinit() なしでは利用が不可能です。
376	仮パラメータの再定義 '%s'。
377	__FUNCSIG__ 及び __FUNCTION__マクロは関数の本体外では使用できません。
378	不正な戻り型（構造体やポインタを返すDLLからインポートされた関数などに生成されます）。
379	テンプレート使用時のエラーです
380	使用していません
381	純粋仮想関数の宣言に使用できない構文です。"=NULL"もしくは"=0"が許可されています
382	仮想関数のみ純粋仮想関数の指定子（"=NULL"または"=0"）を伴う宣言を行うことができます
383	抽象クラスのインスタンスは作成できません

© 2000–2025, MetaQuotes Ltd.

コード	説明
384	dynamic_castオペレーターを使用した動的動作の為には、タイプ定義はカスタムタイプのポインタである必要があります。
385	『関数へのポインタ』タイプを待っています
386	メソッドへのポインタはサポートされていません
387	エラー。関数へのポインタタイプを判別することができません
388	継承が閉じられている為にタイプの動作が利用できません
389	修飾子を持つ変数constは宣言時に初期化する必要があります
393	インターフェースではパブリックアクセスを持つメソッドのみが宣言できます
394	インターフェース内での不正なインターフェースのネスティング
395	インターフェースは他のインターフェースからのみ派生できます
396	インターフェースが期待されています
397	インターフェースはパブリック継承のみをサポートします。
398	インターフェースはメンバを含むことができません
399	インターフェースオブジェクトは直接作成できず、継承のみが使われます
400	指定子は前方宣言では使用できません
401	final指定子で宣言されているため、クラスからの継承は不可能です
402	final指定子で宣言されているメソッドは再定義できません
403	final指定子は仮想関数にのみ適用できます
404	override指定子でマークされたメソッドは実際には基本クラス関数をオーバーライドしません
405	指定子は関数の宣言のみで使用できますが、定義では使用できません
406	指定された型に型をキャストできません
407	型をresource変数に使用できません
408	プロジェクトファイルにエラーがあります
409	unionメンバとしては使用できません
410	名前の選択があいまいです。使用コンテキストは明示的に定義する必要があります
411	DLLから構造体を使用できません
412	delete指定しでマークされた関数は呼び出しできません
413	MQL4はサポートされていません。プログラムをコンパイルするには、MetaTrader 4ターミナルインストールフォルダーのMetaEditorを使用してください。

© 2000-2025, MetaQuotes Ltd.

ランタイムエラー

GetLastError() は事前に定義された変数 _LastError に格納された最後のエラーコードを返す関数です。値は ResetLastError() 関数を使用してゼロにリセットすることが出来ます。

定数	コード	説明
ERR_SUCCESS	0	操作が正常に完了しました。
ERR_INTERNAL_ERROR	4001	予期しない内部エラー。
ERR_WRONG_INTERNAL_PARAMETER	4002	クライアント端末関数の内部呼び出しでの不正なパラメータ。
ERR_INVALID_PARAMETER	4003	システム関数呼び出し時の不正なパラメータ。
ERR_NOT_ENOUGH_MEMORY	4004	システム関数の実行に不充分なメモリ。
ERR_STRUCT_WITHOBJECTS_ORCLASS	4005	構造体に、文字列、及び/または動的な配列及び/またはそのようなオブジェクトの構造体及び/またはクラスが含まれます。
ERR_INVALID_ARRAY	4006	型やサイズが不正な配列、または動的配列内に損傷を受けたオブジェクト。
ERR_ARRAY_RESIZE_ERROR	4007	配列の移転に不充分なメモリ、または静的配列のサイズを変更する試み。
ERR_STRING_RESIZE_ERROR	4008	文字列の再配置に不充分なメモリ。
ERR_NOTINITIALIZED_STRING	4009	初期化されていない文字列。
ERR_INVALID_DATETIME	4010	無効な日付/時刻。
ERR_ARRAY_BAD_SIZE	4011	配列の要素数の合計は2,147,483,647を超えることは出来ません。
ERR_INVALID_POINTER	4012	不正なポインタ。
ERR_INVALID_POINTER_TYPE	4013	不正なポインタ型。
ERR_FUNCTION_NOT_ALLOWED	4014	関数呼び出しの許可がありません。
ERR_RESOURCE_NAME_DUPLICATED	4015	同一の動的及び静的リソース名。
ERR_RESOURCE_NOT_FOUND	4016	この名称のリソースがEX5で見つかりません。

© 2000-2025, MetaQuotes Ltd.

定数	コード	説明
ERR_RESOURCE_UNSUPPORTED_TYPE	4017	リソースタイプがサポートされていないかサイズが 16 MB を超えます。
ERR_RESOURCE_NAME_IS_TOO_LONG	4018	リソース名が 63 字を超えます。
ERR_MATH_OVERFLOW	4019	数学関数の計算時にオーバーフローが発生
ERR_SLEEP_ERROR	4020	Sleep()呼び出しが終了日より後です。
ERR_PROGRAM_STOPPED	4022	テストは外部から強制停止されました。例は、最適化の中断、視覚的テストウィンドウの終了、テストエージェントの停止です。
チャート		
ERR_CHART_WRONG_ID	4101	不正なチャート識別子。
ERR_CHART_NO_REPLY	4102	チャートが応答しません。
ERR_CHART_NOT_FOUND	4103	チャートが見つかりません。
ERR_CHART_NO_EXPERT	4104	イベントを処理出来るエキスパートアドバイザーがチャートに不在。
ERR_CHART_CANNOT_OPEN	4105	チャートオープンエラー。
ERR_CHART_CANNOT_CHANGE	4106	銘柄と期間の変更に失敗。
ERR_CHART_WRONG_PARAMETER	4107	チャート操作の関数のパラメータ値のエラー。
ERR_CHART_CANNOT_CREATE_TIMER	4108	タイマー作成に失敗。
ERR_CHART_WRONG_PROPERTY	4109	不正なチャートプロパティ識別子。
ERR_CHART_SCREENSHOT_FAILED	4110	スクリーンショット作成エラー。
ERR_CHART_NAVIGATE_FAILED	4111	チャートナビゲートエラー。
ERR_CHART_TEMPLATE_FAILED	4112	テンプレート適用エラー。
ERR_CHART_WINDOW_NOT_FOUND	4113	指標を含むサブウィンドウが不在。
ERR_CHART_INDICATOR_CANNOT_ADD	4114	指標をチャートに追加するのに失敗。
ERR_CHART_INDICATOR_CANNOT_DEL	4115	指標をチャートから削除するのに失敗。
ERR_CHART_INDICATOR_NOT_FOUND	4116	指標が指定されたチャートに不在。
グラフィックオブジェクト		
ERR_OBJECT_ERROR	4201	グラフィックオブジェクト操作エラー。

© 2000–2025, MetaQuotes Ltd.

定数	コード	説明
ERR_OBJECT_NOT_FOUND	4202	グラフィックオブジェクトが見つかりません。
ERR_OBJECT_WRONG_PROPERTY	4203	グラフィックオブジェクトプロパティの不正なID。
ERR_OBJECT_GETDATE_FAILED	4204	値に対応する日付の取得が不可能。
ERR_OBJECT_GETVALUE_FAILED	4205	日付に対応する値の取得が不可能。
マーケット情報		
ERR_MARKET_UNKNOWN_SYMBOL	4301	未知のシンボル。
ERR_MARKET_NOT_SELECTED	4302	「気配値表示」でシンボルが未選択。
ERR_MARKET_WRONG_PROPERTY	4303	シンボルプロパティの不正な識別子。
ERR_MARKET_LASTTIME_UNKNOWN	4304	最後のティックの時間が未知（ティック不在）。
ERR_MARKET_SELECT_ERROR	4305	「気配値表示」のシンボル追加・削除に失敗。
ERR_MARKET_SELECT_LIMIT	4306	気配値表示で選択した銘柄の制限を超えました
ERR_MARKET_SESSION_INDEX	4307	SymbolInfoSessionQuote/SymbolInfoSessionTrade関数呼び出し時にセッションIDが間違っています
履歴アクセス		
ERR_HISTORY_NOT_FOUND	4401	リクエストされた履歴が見つかりません。
ERR_HISTORY_WRONG_PROPERTY	4402	不正な履歴プロパティ識別子。
ERR_HISTORY_TIMEOUT	4403	履歴リクエストがタイムアウトしました。
ERR_HISTORY_BARS_LIMIT	4404	要求されたバーの数が端末の設定によって制限されています。
ERR_HISTORY_LOAD_ERRORS	4405	履歴を読み込む際に複数のエラーが生成されました。
ERR_HISTORY_SMALL_BUFFER	4407	受信する配列が小さすぎて、リクエストされたすべてのデータを格納できません。
Global_Variables		

© 2000–2025, MetaQuotes Ltd.

定数	コード	説明
ERR_GLOBALVARIABLE_NOT_FOUND	4501	クライアント端末のグローバル変数が見つかりません。
ERR_GLOBALVARIABLE_EXISTS	4502	同名のクライアント端末のグローバル変数が既存。
ERR_GLOBALVARIABLE_NOT_MODIFIED	4503	グローバル変数が変更されていない
ERR_GLOBALVARIABLE_CANNOTREAD	4504	グローバル変数値を持つファイルの読み取りが不可能
ERR_GLOBALVARIABLE_CANNOTWRITE	4505	グローバル変数値を持つファイルの書き込みが不可能
ERR_MAIL_SEND_FAILED	4510	メール送信に失敗。
ERR_PLAY_SOUND_FAILED	4511	音の再生に失敗。
ERR_MQL5_WRONG_PROPERTY	4512	プログラムプロパティの不正な識別子。
ERR_TERMINAL_WRONG_PROPERTY	4513	端末プロパティの不正な識別子。
ERR_FTP_SEND_FAILED	4514	FTPでのファイル送信に失敗。
ERR_NOTIFICATION_SEND_FAILED	4515	通知送信に失敗。
ERR_NOTIFICATION_WRONG_PARAMETER	4516	通知送信に無効なパラメータ(空の文字列か NULL)が SendNotification() 関数に渡されました。
ERR_NOTIFICATION_WRONG_SETTINGS	4517	端末内の不正な通知の設定(IDの未指定か、許可の無設定)。
ERR_NOTIFICATION_TOO_FREQUENT	4518	頻繁過ぎる通知の送信。
ERR_FTP_NOSERVER	4519	FTPサーバが指定されていません。
ERR_FTP_NOLOGIN	4520	FTPログインが指定されていません。
ERR_FTP_FILE_ERROR	4521	FTPサーバに送信するファイルがMQL5¥Filesディレクトリで見つかりません。
ERR_FTP_CONNECT_FAILED	4522	FTP接続に失敗しました。
ERR_FTP_CHANGEDIR	4523	サーバでFTPパスが見つかりません。
カスタム指標バファー		
ERR_BUFFERS_NO_MEMORY	4601	指標バッファ配布に不充分なメモリ。
ERR_BUFFERS_WRONG_INDEX	4602	不正な指標バッファインデックス。

© 2000-2025, MetaQuotes Ltd.

定数	コード	説明
カスタム指標プロパティ		
ERR_CUSTOM_WRONG_PROPERTY	4603	不正なカスタム指標プロパティ識別子。
口座		
ERR_ACCOUNT_WRONG_PROPERTY	4701	不正な口座プロパティ識別子。
ERR_TRADE_WRONG_PROPERTY	4751	不正な取引プロパティ識別子。
ERR_TRADE_DISABLED	4752	エキスパートアドバイザーでの取引が許可されていません。
ERR_TRADE_POSITION_NOT_FOUND	4753	ポジションが見つかりません。
ERR_TRADE_ORDER_NOT_FOUND	4754	注文が見つかりません。
ERR_TRADE_DEAL_NOT_FOUND	4755	約定が見つかりません。
ERR_TRADE_SEND_FAILED	4756	取引リクエスト送信に失敗。
ERR_TRADE_CALC_FAILED	4758	利益または証拠金の計算に失敗
指標		
ERR_INDICATOR_UNKNOWN_SYMBOL	4801	未知のシンボル。
ERR_INDICATOR_CANNOT_CREATE	4802	指標作成が不可。
ERR_INDICATOR_NO_MEMORY	4803	指標追加に不充分なメモリ。
ERR_INDICATOR_CANNOT_APPLY	4804	指標の他の指標への適用は不可。
ERR_INDICATOR_CANNOT_ADD	4805	指標をチャートに適用するのに失敗。
ERR_INDICATOR_DATA_NOT_FOUND	4806	リクエストされたデータが見つかりません。
ERR_INDICATOR_WRONG_HANDLE	4807	不正な指標ハンドル。
ERR_INDICATOR_WRONG_PARAMETERS	4808	指標作成時の不正なパラメータ数。
ERR_INDICATOR_PARAMETERS_MISSING	4809	指標作成時にパラメータ型が不在。
ERR_INDICATOR_CUSTOM_NAME	4810	配列の最初のパラメータは、カスタム指標の名称でなければなりません。
ERR_INDICATOR_PARAMETER_TYPE	4811	指標作成時の配列内の無効なパラメータ型。
ERR_INDICATOR_WRONG_INDEX	4812	リクエストされた指標バッファの不正なインデックス。

© 2000–2025, MetaQuotes Ltd.

定数	コード	説明
板情報		
ERR_BOOKS_CANNOT_ADD	4901	板情報の追加が不可能。
ERR_BOOKS_CANNOT_DELETE	4902	板情報の削除が不可能。
ERR_BOOKS_CANNOT_GET	4903	板情報データ取得が不可能。
ERR_BOOKS_CANNOT_SUBSCRIBE	4904	板情報から新規データを取得するのにサブスクライブ中にエラーが発生。
ファイル操作		
ERR_TOO_MANY_FILES	5001	64 を超えるファイルを同時に開く事は不可能。
ERR_WRONG_FILENAME	5002	無効なファイル名。
ERR_TOO_LONG_FILENAME	5003	長すぎるファイル名。
ERR_CANNOT_OPEN_FILE	5004	ファイルオープンエラー。
ERR_FILE_CACHEBUFFER_ERROR	5005	読み込みのためにキャッシュに出来るメモリが不足。
ERR_CANNOT_DELETE_FILE	5006	ファイル削除エラー。
ERR_INVALID_FILEHANDLE	5007	このハンドルを使用したファイルは閉じられた、または、初めから開けられませんでした。
ERR_WRONG_FILEHANDLE	5008	不正なファイルハンドル。
ERR_FILE_NOTTOWRITE	5009	ファイルは書き込むために開かれる必要があります。
ERR_FILE_NOTTOREAD	5010	ファイルは読み込むために開かれる必要があります。
ERR_FILE_NOTBIN	5011	ファイルはバイナリとして開かれる必要があります。
ERR_FILE_NOTTXT	5012	ファイルはテキストとして開かれる必要があります。
ERR_FILE_NOTTXTORCSV	5013	ファイルはテキストまたは CSV として開かれる必要があります。
ERR_FILE_NOTCSV	5014	ファイルは CSV として開かれる必要があります。
ERR_FILE_READERROR	5015	ファイル読み込みエラー。
ERR_FILE_BINSTRINGSIZE	5016	ファイルがバイナリとしてオープンされたため、文字列のサイズ指定が必要。

© 2000–2025, MetaQuotes Ltd.

定数	コード	説明
ERR_INCOMPATIBLE_FILE	5017	文字列配列はテキストファイル、他の配列は　バイナリファイルでなければいけません。
ERR_FILE_IS_DIRECTORY	5018	これはファイルではなくディレクトリです。
ERR_FILE_NOT_EXIST	5019	ファイルが不在。
ERR_FILE_CANNOT_REWRITE	5020	ファイルの書き換えが不可。
ERR_WRONG_DIRECTORYNAME	5021	不正なディレクトリ名。
ERR_DIRECTORY_NOT_EXIST	5022	ディレクトリ不在。
ERR_FILE_ISNOT_DIRECTORY	5023	これはディレクトリではなくファイルです。
ERR_CANNOT_DELETE_DIRECTORY	5024	ディレクトリを削除に失敗。
ERR_CANNOT_CLEAN_DIRECTORY	5025	ディレクトリのクリアに失敗（おそらく1つ以上のファイルがブロックされ、除去操作が失敗）。
ERR_FILE_WRITEERROR	5026	ファイルへのリソースの書き込みに失敗。
ERR_FILE_ENDOFFILE	5027	ファイルの終わりに達する為、CSVファイル(FileReadString、FileReadNumber、FileReadDatetime、FileReadBool)から次のデータを読み取ることができませんでした。
文字列のキャスト		
ERR_NO_STRING_DATE	5030	文字列内に日付なし。
ERR_WRONG_STRING_DATE	5031	文字列で不正な日付。
ERR_WRONG_STRING_TIME	5032	文字列内の不正な時刻。
ERR_STRING_TIME_ERROR	5033	文字列から日付への変換エラー。
ERR_STRING_OUT_OF_MEMORY	5034	文字列に不充分なメモリ。
ERR_STRING_SMALL_LEN	5035	予想より短い文字列。
ERR_STRING_TOO_BIGNUMBER	5036	大きすぎて　ULONG_MAX　を超える数。
ERR_WRONG_FORMATSTRING	5037	無効なフォーマットストリング。
ERR_TOO_MANY_FORMATTERS	5038	フォーマット指定子数がパラメータ数を超過。

© 2000–2025, MetaQuotes Ltd.

定数	コード	説明
ERR_TOO_MANY_PARAMETERS	5039	パラメータ数がフォーマット指定子数を超過。
ERR_WRONG_STRING_PARAMETER	5040	損傷した文字列型のパラメータ。
ERR_STRINGPOS_OUTOFRANGE	5041	位置が文字列の範囲外。
ERR_STRING_ZEROADDED	5042	文字列の末尾に 0 が追加されました(無効な操作)。
ERR_STRING_UNKNOWNTYPE	5043	文字列への変換時の未知なデータ型。
ERR_WRONG_STRING_OBJECT	5044	破損した文字列オブジェクト。
配列操作		
ERR_INCOMPATIBLE_ARRAYS	5050	互換性のない配列の複製。文字列配列は文字列配列、数値配列は数値配列のみに複製することが出来ます。
ERR_SMALL_ASSERIES_ARRAY	5051	受け取り側の配列は AS_SERIES として宣言されていて、サイズが不充分です。
ERR_SMALL_ARRAY	5052	配列が小さすぎ、開始位置が配列の外側にあります。
ERR_ZEROSIZE_ARRAY	5053	長さゼロの配列。
ERR_NUMBER_ARRAYS_ONLY	5054	数値配列のみ可。
ERR_ONEDIM_ARRAYS_ONLY	5055	1 次元配列のみが可。
ERR_SERIES_ARRAY	5056	時系列は使用不可。
ERR_DOUBLE_ARRAY_ONLY	5057	double 型の配列のみ可。
ERR_FLOAT_ARRAY_ONLY	5058	float 型の配列のみ可。
ERR_LONG_ARRAY_ONLY	5059	long 型の配列のみ可。
ERR_INT_ARRAY_ONLY	5060	int 型の配列のみ可。
ERR_SHORT_ARRAY_ONLY	5061	short 型の配列のみ可。
ERR_CHAR_ARRAY_ONLY	5062	char 型の配列のみ可。
ERR_STRING_ARRAY_ONLY	5063	文字列配列のみ
OpenCL の操作		
ERR_OPENCL_NOT_SUPPORTED	5100	OpenCL 関数がこのコンピューターでサポートされていません。
ERR_OPENCL_INTERNAL	5101	OpenCL実行中に内部エラー発生。

© 2000–2025, MetaQuotes Ltd.

定数	コード	説明
ERR_OPENCL_INVALID_HANDLE	5102	無効なOpenCL ハンドル。
ERR_OPENCL_CONTEXT_CREATE	5103	OpenCL コンテキスト作成に失敗。
ERR_OPENCL_QUEUE_CREATE	5104	OpenCL　の実行キューの作成に失敗。
ERR_OPENCL_PROGRAM_CREATE	5105	OpenCL プログラムのコンパイル中にエラー発生。
ERR_OPENCL_TOO_LONG_KERNEL_NAME	5106	カーネル名が長すぎます(OpenCL カーネル)。
ERR_OPENCL_KERNEL_CREATE	5107	OpenCL カーネル作成エラー。
ERR_OPENCL_SET_KERNEL_PARAMETER	5108	OpenCL　カーネルパラメータ設定中にエラー発生。
ERR_OPENCL_EXECUTE	5109	OpenCL プログラムランタイムエラー。
ERR_OPENCL_WRONG_BUFFER_SIZE	5110	OpenCL バッファの無効なサイズ。
ERR_OPENCL_WRONG_BUFFER_OFFSET	5111	OpenCL バッファ内の無効なオフセット。
ERR_OPENCL_BUFFER_CREATE	5112	OpenCL バッファの作成に失敗。
ERR_OPENCL_TOO_MANY_OBJECTS	5113	OpenCLオブジェクトが多すぎる
ERR_OPENCL_SELECTDEVICE	5114	OpenCLデバイス選択エラー
データベース操作		
ERR_DATABASE_INTERNAL	5120	内部データベースエラー
ERR_DATABASE_INVALID_HANDLE	5121	無効なデータベースハンドル
ERR_DATABASE_TOO_MANY_OBJECTS	5122	Databaseオブジェクトの最大数を超えました
ERR_DATABASE_CONNECT	5123	データベース接続エラー
ERR_DATABASE_EXECUTE	5124	要求実行エラー
ERR_DATABASE_PREPARE	5125	要求生成エラー
ERR_DATABASE_NO_MORE_DATA	5126	読み込むデータがありません
ERR_DATABASE_STEP	5127	次の要求エントリに移動できませんでした
ERR_DATABASE_NOT_READY	5128	要求結果を読み取るためのデータはまだ準備できていません
ERR_DATABASE_BIND_PARAMETERS	5129	SQL要求へのパラメーターの自動置換に失敗しました
WebRequest の操作		

© 2000–2025, MetaQuotes Ltd.

定数	コード	説明
ERR_WEBREQUEST_INVALID_ADDRESS	5200	無効な URL。
ERR_WEBREQUEST_CONNECT_FAILED	5201	指定された URL への接続に失敗。
ERR_WEBREQUEST_TIMEOUT	5202	タイムアウトを超過。
ERR_WEBREQUEST_REQUEST_FAILED	5203	HTTP リクエストに失敗。
ネットワーク操作（ソケット）		
ERR_NETSOCKET_INVALIDHANDLE	5270	関数に無効なソケットハンドルが渡されました。
ERR_NETSOCKET_TOO_MANY_OPENED	5271	開いているソケットが多すぎます（最高 128）。
ERR_NETSOCKET_CANNOT_CONNECT	5272	遠隔ホストに接続できません。
ERR_NETSOCKET_IO_ERROR	5273	ソケットからデータを送受信できません。
ERR_NETSOCKET_HANDSHAKE_FAILED	5274	安全な接続を確立できません（TLS ハンドシェイク）。
ERR_NETSOCKET_NO_CERTIFICATE	5275	接続を保護する証明書に関するデータがありません。
カスタムシンボル		
ERR_NOT_CUSTOM_SYMBOL	5300	カスタムシンボルが指定されていません。
ERR_CUSTOM_SYMBOL_WRONG_NAME	5301	カスタムシンボル名が無効です。シンボル名には、句読点、スペース、特殊文字以外のラテン文字を含めることができます（"."、"_"、"&"、"#"は可）。<、>、:、"、/、¥、|、?、*の使用は推奨されていません。
ERR_CUSTOM_SYMBOL_NAME_LONG	5302	カスタムシンボル名が長すぎます。シンボル名の長さは、末尾の0文字を含んで32文字を超えてはなりません。
ERR_CUSTOM_SYMBOL_PATH_LONG	5303	カスタムシンボルのパスが長すぎます。パスの長さは"Custom¥"、シンボル名、グループセパレータ、および末尾の0を含んで128文字を超えてはなりません。
ERR_CUSTOM_SYMBOL_EXIST	5304	同じ名称のカスタムシンボルが既存。

© 2000–2025, MetaQuotes Ltd.

定数	コード	説明
ERR_CUSTOM_SYMBOL_ERROR	5305	カスタムシンボルの作成、削除または変更中にエラーが発生。
ERR_CUSTOM_SYMBOL_SELECTED	5306	板情報で選択されたカスタムシンボルを削除しようとしています。
ERR_CUSTOM_SYMBOL_PROPERTY_WRONG	5307	カスタムシンボルプロパティが無効。
ERR_CUSTOM_SYMBOL_PARAMETER_ERROR	5308	カスタムシンボルのプロパティを設定している際のパラメータが不正。
ERR_CUSTOM_SYMBOL_PARAMETER_LONG	5309	カスタムシンボルのプロパティを設定する際に文字列パラメータが長すぎます。
ERR_CUSTOM_TICKS_WRONG_ORDER	5310	配列内のティックが時間順でありません
経済カレンダー		
ERR_CALENDAR_MORE_DATA	5400	配列サイズがすべての値の説明を受け取るには不十分です
ERR_CALENDAR_TIMEOUT	5401	リクエストの制限時間を超えました
ERR_CALENDAR_NO_DATA	5402	国が見つかりません
データベースの操作		
ERR_DATABASE_ERROR	5601	一般的なエラー
ERR_DATABASE_LOGIC	5602	SQLite内部ロジックエラー
ERR_DATABASE_PERM	5603	アクセスが拒否されました
ERR_DATABASE_ABORT	5604	コールバックルーチンが中止を要求しました
ERR_DATABASE_BUSY	5605	データベースファイルがロックされています
ERR_DATABASE_LOCKED	5606	データベーステーブルがロックされています<分節 2781
ERR_DATABASE_NOMEM	5607	操作を完了するためのメモリが不足しています
ERR_DATABASE_READONLY	5608	読み取り専用データベースへの書き込みが試みされました
ERR_DATABASE_INTERRUPT	5609	操作がsqlite3_interrupt()によって終了されました
ERR_DATABASE_IOERR	5610	ディスクI/Oエラー
ERR_DATABASE_CORRUPT	5611	データベースディスクイメージが破損しています

© 2000–2025, MetaQuotes Ltd.

定数	コード	説明
ERR_DATABASE_NOTFOUND	5612	sqlite3_file_control()に不明な操作コードがあります
ERR_DATABASE_FULL	5613	データベースがいっぱいのため、挿入に失敗しました
ERR_DATABASE_CANTOPEN	5614	データベースファイルを開くことができません
ERR_DATABASE_PROTOCOL	5615	データベースロックプロトコルエラー
ERR_DATABASE_EMPTY	5616	内部使用のみ
ERR_DATABASE_SCHEMA	5617	データベーススキーマが変更されました
ERR_DATABASE_TOOBIG	5618	文字列またはBLOBがサイズ制限を超えています
ERR_DATABASE_CONSTRAINT	5619	制約違反による中止
ERR_DATABASE_MISMATCH	5620	データ型の不一致
ERR_DATABASE_MISUSE	5621	ライブラリが正しく使用されていません
ERR_DATABASE_NOLFS	5622	ホストでサポートされていないOS機能を使用しています
ERR_DATABASE_AUTH	5623	承認が拒否されました
ERR_DATABASE_FORMAT	5624	未使用
ERR_DATABASE_RANGE	5625	バインドパラメータエラー、不正なインデックス
ERR_DATABASE_NOTADB	5626	データベースファイルではないファイルが開かれました
ユーザ定義のエラー		
ERR_USER_ERROR_FIRST	65536	ユーザ定義エラーはこのコードで始まります。

参照

取引サーバのリターンコード

© 2000-2025, MetaQuotes Ltd.

入出力定数

定数：

- ファイルを開く際のフラグ
- ファイルプロパティ
- ファイル内の位置
- コードページの使用
- メッセージボックス

© 2000–2025, MetaQuotes Ltd.

ファイルを開く際のフラグ

ファイルを開く際のフラグはファイルのアクセスモードを指定します。フラグは下記の様に定義されます。

識別子	値	説明
FILE_READ	1	ファイルは読み込むために開かれます。フラグは FileOpen() で使用されます。ファイルを開く際には FILE_WRITE 及び/または FILE_READ 仕様が必要です。
FILE_WRITE	2	ファイルは書き込むために開かれます。フラグは FileOpen() で使用されます。ファイルを開く際には FILE_WRITE 及び/または FILE_READ 仕様が必要です。
FILE_BIN	4	(文字列への変換文字列を含まない)バイナリ読み込み/書き込みモードです。フラグは FileOpen() で使用されます。
FILE_CSV	8	CSV ファイル(全ての要素が Unicode または ANSI の適切な型の文字列に変換されセパレータで区切られています)。フラグは FileOpen() で使用されます。
FILE_TXT	16	単純なテキストファイル(CSV ファイルと同じですが、セパレータを考慮しません)。フラグは FileOpen() で使用されます。
FILE_ANSI	32	ANSIタイプ(1 バイトのシンボル)の文字列。フラグは FileOpen() で使用されます。
FILE_UNICODE	64	UNICODE タイプ(2 バイトのシンボル)の文字列。フラグは FileOpen() で使用されます。
FILE_SHARE_READ	128	複数のプログラムからの読み込みのための共通のアクセス。フラグは FileOpen() で使用されますがファイルを開く時の FILE_WRITE 及び/または FILE_READ フラグは必要です。
FILE_SHARE_WRITE	256	複数のプログラムからの書き込みのための共通のアクセス。フラグは FileOpen() で使用されますがファイルを開く時の FILE_WRITE 及び/または FILE_READ フラグは必要です。
FILE_REWRITE	512	FileCopy() 及び FileMove() 関数でのファイル上書きを可能にします。ファイルが存在し、書き込み用に開かれる必要があります。さもなければ、ファイルは開かれません。
FILE_COMMON	4096	全てのクライアント端末の共有フォルダ内のファイルパス \Terminal\Common\Files。フラグは FileOpen()、FileCopy()、FileMove() 及び FileIsExist() 関数で使用されます。

ファイルを開く時には、1 つ以上のフラグを指定することが出来ます。これは、フラグの組み合わせです。フラグの組み合わせは、列挙されたフラグの間に配置され(|)の論理和の符号を使用して書かれます。例えば、ファイルを CSV フォーマットで読み書きするために開くには、FILE_READ| FILE_WRITE| FILE_CSV の組み合わせを指定します。

例:

```
int filehandle=FileOpen(filename,FILE_READ|FILE_WRITE|FILE_CSV);
```

© 2000-2025, MetaQuotes Ltd.

読み書きのフラグの指定には、いくつかの特定の機能があります。

- FILE_READ が指定されると、既存するファイルを開く試みがなされます。ファイルが存在しない場合、ファイルを開く操作は失敗し、新しいファイルは作成されません。
- FILE_READ|FILE_WRITE – 指定された名称のファイルが存在しない場合は、新しいファイルが作成されます。
- FILE_WRITE – ファイルは、ゼロサイズで再作成されます。

ファイルを開く際には FILE_WRITE 及び/または FILE_READ が必要です。

開いているファイルの読み込みの種類を定義するフラグには優先順位があります。最優先されるフラグは FILE_CSV、その次が FILE_BIN で、FILE_TXT の優先度は最も低いです。複数のフラグ(FILE_TXT|FILE_CSV または FILE_TXT|FILE_BIN または FILE_BIN|FILE_CSV)が同時に指定された場合、最高の優先度のフラグが使用されます。

エンコーディングの種類を定義するフラグにも優先順位があります。FILE_UNICODE の優先度は FILE_ANSI より高いです。なので、もし FILE_UNICODE|FILE_ANSI が指定された場合 FILE_UNICODE が使用されます。

FILE_UNICODE と FILE_ANSI のどちらもが指定されない場合には、FILE_UNICODE が暗示されます。FILE_CSV、FILE_BIN、FILE_TXT の 1 つも指定されない場合には、FILE_CSV が暗示されます。

もしファイルがテキストとして読まれるために開かれれ(FILE_TXT または FILE_CSV)ファイルの始めに特別な 2 バイトの印 0xff,0xfe がある場合、FILE_ANSI が指定されたとしても、エンコーディングフラグは FILE_UNICODE となります。

参照

ファイル関数

© 2000-2025, MetaQuotes Ltd.

ファイルプロパティ

FileGetInteger() 関数はファイルプロパティの取得に使用されます。ENUM_FILE_PROPERTY_INTEGER 列挙からの必要なプロパティの識別子は、呼び出し時に渡されます。

ENUM_FILE_PROPERTY_INTEGER

ID	ID 説明
FILE_EXISTS	ファイルの存在をチェック。
FILE_CREATE_DATE	作成日。
FILE_MODIFY_DATE	最終更新日。
FILE_ACCESS_DATE	最終アクセス日。
FILE_SIZE	ファイルのバイトサイズ。
FILE_POSITION	ファイル内のポインタ位置。
FILE_END	ファイル終了の記号を取得。
FILE_LINE_END	行終了の記号を取得。
FILE_IS_COMMON	ファイルを全ての端末の共有フォルダ（ FILE_COMMON を参照）で開きます。
FILE_IS_TEXT	ファイルをテキスト（ FILE_TXT を参照）として開きます。
FILE_IS_BINARY	ファイルをバイナリ（ FILE_BIN を参照）として開きます。
FILE_IS_CSV	ファイルを CSV（ FILE_CSV を参照）として開きます。
FILE_IS_ANSI	ファイルを ANSI（ FILE_ANSI を参照）として開きます。
FILE_IS_READABLE	開かれたファイルが読み込み可能（ FILE_READ を参照）。
FILE_IS_WRITABLE	開かれたファイルが書き込み可能（ FILE_WRITE を参照）。

FileGetInteger() には 2 つの呼び出し方があります。1 つ目はファイルプロパティの取得に使用され、FileOpen() 関数でファイルを開く際に取得されるハンドルが指定されます。これで、ファイルの全てのプロパティの取得が出来ます。

2 番目の呼び出しでは FileGetInteger() はファイル名でファイルのプロパティの値を返します。このオプションを使用して取得出来るのは次の一般的なプロパティのみです。

- FILE_EXISTS – 指定された名称のファイルの存在
- FILE_CREATE_DATE – 指定された名称のファイルの作成日
- FILE_MODIFY_DATE – 指定された名称のファイルの更新日
- FILE_ACCESS_DATE - 指定された名称のファイルの最終アクセス日
- FILE_SIZE – 指定された名称のファイルサイズ

上記に指定された以外のプロパティを取得しようとすると FileGetInteger() の2 番目の呼び出しオプションは、エラーを返します。

© 2000–2025, MetaQuotes Ltd.

ファイル内の位置

ファイル関数の多くは、データの読み込み/書き込み動作に関連します。同時に、FileSeek() を使用すれば、次の読み込みまたは書き込み操作が実行されるファイル内の位置にファイルポインタの位置を指定することが出来ます。ENUM_FILE_POSITION 列挙は、次の演算のためのバイトの相対的なシフトを指定出来る有効なポインタの位置を含みます。

ENUM_FILE_POSITION

識別子	説明
SEEK_SET	ファイルの始め。
SEEK_CUR	ファイルポインタの現在位置。
SEEK_END	ファイルの終わり。

参照

FileIsEnding、FileIsLineEnding

© 2000-2025, MetaQuotes Ltd.

文字列の変換操作でのコードページの使用

string 変数を char 型 の変数と相互に変換する場合、MQL5 は、デフォルトでは、Windows オペレーティングシステムの現在の ANSI エンコーディング(CP_ACP)を使用します。異なる種類のエンコーディングは CharArrayToString()、StringToCharArray() 及び FileOpen() 関数の追加パラメータとして指定することが出来ます。

表は、いくつかの最も人気のあるコードページのための内蔵された定数を示します。他のコードページは、ページに対応するコードで指定することが出来ます。

コードページの内蔵定数

定数	値	説明
CP_ACP	0	現在の Windows ANSI コードページ。
CP_OEMCP	1	現在のシステムの OEM コードページ。
CP_MACCP	2	現在のシステムの Macintosh コードページ。 注意事項：現代の Macintosh コンピュータはエンコーディングに Unicode を使用しているため、この値は、主に以前に作成したプログラムコードで使用され、今は使用されていません。
CP_THREAD_ACP	3	現在のスレッドの Windows ANSI コードページ。
CP_SYMBOL	42	シンボルコードページ。
CP_UTF7	65000	UTF-7 コードページ。
CP_UTF8	65001	UTF-8 コードページ。

参照

クライアント端末プロパティ

© 2000-2025, MetaQuotes Ltd.

メッセージボックスダイアログウィンドウ定数

このセクションは MessageBox() 関数のリターンコードを含みます。メッセージウィンドウに「キャンセル」ボタンがあって、ESCキーまたは「キャンセル」ボタンが押された場合、この関数は IDCANCEL を返します。メッセージウィンドウに「キャンセル」ボタンがない場合、ESC を押すことで変化はありません。

定数	値	説明
IDOK	1	「OK」ボタンが押されました。
IDCANCEL	2	「キャンセル」ボタンが押されました。
IDABORT	3	「中止(A)」ボタンが押されました。
IDRETRY	4	「再試行(R)」ボタンが押されました。
IDIGNORE	5	「無視(I)」ボタンが押されました。
IDYES	6	「はい」ボタンが押されました。
IDNO	7	「いいえ」ボタンが押されました。
IDTRYAGAIN	10	「再試行する」ボタンが押されました。
IDCONTINUE	11	「続行(C)」ボタンが押されました。

MessageBox() 関数のメインフラグはダイアログウィンドウの内容と動作を定義します。この値は、次のフラグのグループの組み合わせであることが出来ます。

定数	値	説明
MB_OK	0x00000000	メッセージウィンドウには、「OK」ボタンが 1 つあります。(デフォルト)。
MB_OKCANCEL	0x00000001	メッセージウィンドウには、「OK」と「キャンセル」の 2 つのボタンがあります。
MB_ABORTRETRYIGNORE	0x00000002	メッセージウィンドウには、「中止(A)」「再試行(R)」「無視(I)」の 3 つのボタンがあります。
MB_YESNOCANCEL	0x00000003	メッセージウィンドウには、「はい」、「いいえ」と「キャンセル」の 3 つのボタンがあります。
MB_YESNO	0x00000004	メッセージウィンドウには、「はい」と「いいえ」の 2 つのボタンがあります。
MB_RETRYCANCEL	0x00000005	メッセージウィンドウには、「再試行(R)」と「キャンセル」の 2 つのボタンがあります。
MB_CANCELTRYCONTINUE	0x00000006	メッセージウィンドウには、「キャンセル」「再試行(R)」「続行(C)」の 3 つのボタンがあります。

メッセージウィンドウにアイコンを表示するには、追加のフラグを指定する必要があります。

© 2000-2025, MetaQuotes Ltd.

定数	値	説明
MB_ICONSTOP, MB_ICONERROR, MB_ICONHAND	0x00000010	STOP アイコン。
MB_ICONQUESTION	0x00000020	はてなマークのアイコン。
MB_ICONEXCLAMATION, MB_ICONWARNING	0x00000030	感嘆/警告記号アイコン。
MB_ICONINFORMATION, MB_ICONASTERISK	0x00000040	丸で囲まれた i サイン。

デフォルトボタンは、次のフラグによって定義されています。

定数	値	説明
MB_DEFBUTTON1	0x00000000	MB_DEFBUTTON2、MB_DEFBUTTON3 及び MB_DEFBUTTON4 が指定されてない場合、最初のボタンである MB_DEFBUTTON1 がデフォルトです。
MB_DEFBUTTON2	0x00000100	2 番目のボタンがデフォルト。
MB_DEFBUTTON3	0x00000200	3 番目のボタンがデフォルト。
MB_DEFBUTTON4	0x00000300	4 番目のボタンがデフォルト。

© 2000-2025, MetaQuotes Ltd.

MQL5 プログラム

MQL5　プログラムは動作する前にコンパイルが必要です(「Compile」ボタンまたはF7キー)。コンパイルはエラーなしで完成しなければなりません(警告は分析する必要があります)。この過程では、ファイルと同じ名称とEX5 拡張子を持つ実行可能ファイルが terminal_dir¥MQL5¥Experts、terminal_dir¥MQL5¥Indicators もしくは terminal_dir¥MQL5¥Scripts ディレクトリに作成されます。作成されるのは実行ファイルです。

MQL5 プログラムの動作機能は、次のセクションで説明されます。

- プログラム実行 – 事前定義されたイベントハンドラの呼び出し順序
- 取引ストラテジーのテスト – ストラテジーテスターでの MQL5 プログラムの動作機能
- クライアント端末イベント – プログラムで処理出来るイベントの説明
- インポートした関数の呼び出し – インポート関数の記述順、許可されたパラメータ、検索の詳細及び呼び出し規則
- ランタイムエラー – ランタイムと重大なエラーに関する情報取得

エキスパートアドバイザー、カスタム指標及びスクリプトは、ナビゲーターウィンドウからドラッグアンドドロップで開かれたチャートの 1 つに取り付けられます。

エキスパートアドバイザーの動作を停止するには、チャートからの削除が必要です。削除には、チャートコンテキストメニューの「EA表示リスト」を選択、リストからエキスパートアドバイザーを選択して「削除」ボタンをクリックします。エキスパートアドバイザーの動作は、「自動売買」ボタンの状態にも影響されます。

カスタム指標の停止には、チャートからの削除が必要です。

カスタム指標とエキスパートアドバイザーは明示的にチャートから削除されるまで動作します。接続されたエキスパートアドバイザーと指標に関する情報は、クライアント端末のセッション間で保存されます。

スクリプトは一度だけ実行され、操作の完了または現在のチャートの状態の変化、またはクライアント端末のシャットダウン時の時に自動的に削除されます。スクリプト情報は保存されないので、クライアント端末の再起動時にはスクリプトは開始されません。

各チャートでは、エキスパートアドバイザーとスクリプトは各　1　つのみ動作できますが、動作する指標の数には制限がありません。

サービスは、機能するためにチャートにバインドされる必要はなく、補助機能を実行するように設計されています。たとえば、サービスでは、カスタム銘柄を作成してチャートを開き、ネットワークを使用して無限ループでそのデータを受け取り、常に更新することができます。

© 2000–2025, MetaQuotes Ltd.

プログラム実行

スクリプト、サービス、エキスパートアドバイザーはそれぞれ独自のスレッドで実行されます。1 つの銘柄に基づいて計算される全ての指標は、別々のチャートに接続されている場合でも同じスレッドで動作します。従って、1 つの銘柄の指標全てが1 つのスレッドのリソースを共有します。

ティック処理や履歴同期のような他の全てのシンボルに関連するアクションは、一貫して指標と同じスレッドで実行されます。これは、指標で無限のアクションが実行されている場合は、そのシンボルに関連した他のイベントは一切行われないことを意味します。

エキスパートアドバイザーを実行する際には、それが実際の取引環境を持ち、必要な銘柄と期間の履歴にアクセスでき、端末とサーバのデータ同期が出来ることをご確認ください。これら全ての手順の終了に、端末は、エキスパートアドバイザーが使用可能なデータで開始されるまでのわずか 5 秒の開始遅延を提供します。そのため、サーバへの接続がない場合、エキスパートアドバイザーの開始の遅れが発生する可能性があります。

以下の表には、MQL5 プログラムの概要が含まれています。

プログラム	実行	注意事項
サービス	別のスレッド。サービス用のスレッド数はサービス数に等しいです。	反復したサービスの実行は、他のプログラムの実行を妨げることは出来ません。
スクリプト	別々のスレッド。スクリプトのスレッド数は、スクリプトの数に等しいです。	反復したスクリプトの実行は、他のプログラムの実行を妨げることは出来ません。
エキスパートアドバイザー	別々のスレッド。エキスパートアドバイザーのスレッド数は、エキスパートアドバイザーの数に等しいです。	反復したエキスパートアドバイザーの実行は、他のプログラムの実行を妨げることが出来ません。
指標	シンボルの全ての指標に 1 つのスレッド。スレッド数は指標とシンボルの数に等しいです。	1 つの指標での無限ループはこのシンボルの他の全ての指標を停止します。

指標は、プログラムがチャートに装着された直後、グローバル変数が初期化されるとともにクライアント端末のメモリにアップロードされます。クラス型のグローバル変数がコンストラクタを持つ場合、コンストラクタはグローバル変数の初期化中に呼ばれます。

その後プログラムはクライアント端末からのイベントを待機します。MQL5 プログラムには少なくとも 1 つのイベントハンドラがあるべきです。さもないと読み込まれたプログラムが実行されません。イベントハンドラは事前定義された名称、パラメータ、及び戻し型を持っています。

Type	関数名	パラメータ	アプリケーション	コメント
int	OnInit	なし	エキスパートアドバイザーと指標	Init イベントハンドラ。void 戻り値が使用出来ます。
void	OnDeinit	const int reason	エキスパートアドバイザーと指標	Deinit イベントハンドラ。

© 2000–2025, MetaQuotes Ltd.

Type	関数名	パラメータ	アプリケーション	コメント
void	OnStart	なし	スクリプトとサービス	Start イベントハンドラ。
int	OnCalculate	const int rates_total, const int prev_calculated, const datetime &Time[], const double &Open[], const double &High[], const double &Low[], const double &Close[], const long &TickVolume[], const long &Volume[], const int &Spread[]	指標	全価格のイベントハンドラを計算します。
int	OnCalculate	const int rates_total, const int prev_calculated, const int begin, const double &price[]	指標	単一データ配列のイベントハンドラを計算します。 指標は 2 つのイベントハンドラを同時に持つことは出来ません。この場合、唯一のイベントハンドラがデータ配列上で動作します。
void	OnTick	なし	エキスパートアドバイザー	NewTick イベントハンドラ。新しいティックの受け取りイベントが処理されている場合、他の同型のイベントは受け入れられません。
void	OnTimer	なし	エキスパートアドバイザーと指標	Timer イベントハンドラ。
void	OnTrade	なし	エキスパートアドバイザー	Trade イベントハンドラ。
double	OnTester	なし	エキスパートアドバイザー	Tester イベントハンドラ。
void	OnChartEvent	const int id, const long &lparam, const double &dparam, const string &sparam	エキスパートアドバイザーと指標	ChartEvent イベントハンドラ。
void	OnBookEvent	const string &symbol_name	エキスパートアドバイザーと指標	BookEvent イベントハンドラ。

クライアント端末は、対応するオープンチャートに新しいイベントを送信します。イベントはチャート（チャートイベント）または MQL5 プログラム（カスタムイベント）によっても生成出来ます。チャート上のグラフィカルオブ

© 2000–2025, MetaQuotes Ltd.

ジェクトの作成または削除のイベントの生成は CHART_EVENT_OBJECT_CREATE 及び CHART_EVENT_OBJECT_DELETE チャートプロパティによって有効または無効に出来ます。それぞれの MQL5 プログラムとチャートは、全ての新しい受信イベントが追加される独自のイベントキューを持っています。

プログラムは、それが動作しているチャートからのイベントのみを受け取ります。全てのイベントは、受信された順に次々に処理されます。キューにすでに NewTick イベントがある場合、またはこのイベントが現在処理中の場合、新しい NewTick イベントは MQL5 プログラムのキューには追加されません。似たように ChartEvent がすでにキューに存在する場合、またはこのイベントが現在処理中の場合、同じ型のイベントはキューには追加されません。timer イベントも同じように扱われ、Timer イベントがキューに存在するか処理中の場合、新しい timer イベントはキューに追加されません。

イベントキューは限られてはいますが充分な大きさを持っています。よく書かれたプログラムでのキューのオーバーフローは起こりにくいです。オーバーフローの場合、新しいイベントはキューに追加されずに廃棄されます。

無限ループを使用してのイベント処理は推奨されません。例外は単一のStartイベントを処理するスクリプトとサービスです。

ライブラリはイベントを処理しません。

指標及びエキスパートアドバイザーで使用出来ない関数

指標、スクリプトとエキスパートアドバイザーは MQL5 で書かれた実行可能なプログラムです。これらは、タスクの種類ごとに設計されています。従って、プログラムの種類によって関数の使用に制限が見られます。下記の関数は指標では使用出来ません。

- OrderCalcMargin()
- OrderCalcProfit()
- OrderCheck()
- OrderSend()
- SendFTP()
- Sleep()
- ExpertRemove()
- MessageBox()

指標用に設計された関数の全ては、エキスパートアドバイザーやスクリプトでは使用出来ません。

- SetIndexBuffer()
- IndicatorSetDouble()
- IndicatorSetInteger()
- IndicatorSetString()
- PlotIndexSetDouble()
- PlotIndexSetInteger()
- PlotIndexSetString()
- PlotIndexGetInteger

© 2000–2025, MetaQuotes Ltd.

ライブラリは独立したプログラムではなく、それを呼び出した MQL5 プログラム（スクリプト、指標またはエキスパートアドバイザー）のコンテキストで実行されます。従って、上記の制限は呼ばれたライブラリにも適用されます。

サービスで使用できない関数

チャートにバインドされていないため、サービスはイベントを受け入れません。以下の関数はサービス内で禁止されています。

ExpertRemove();

EventSetMillisecondTimer();

EventSetTimer();

EventKillTimer();

SetIndexBuffer();

IndicatorSetDouble();

IndicatorSetInteger();

IndicatorSetString();

PlotIndexSetDouble();

PlotIndexSetInteger();

PlotIndexSetString();

PlotIndexGetInteger();

指標のロードとアンロード

指標は、次の場合に読み込まれます。

- 指標のチャートへの接続
- 端末の開始（指標が端末のシャットダウン前にチャートに接続された場合）
- テンプレートの読み込み（チャートに接続されている指標がテンプレートで指定されている場合）
- プロファイルの変更（指標がプロファイルチャートに接続されている場合）
- 指標が接続されたチャートのシンボル及び/または時間軸の変更
- 指標の再コンパイルの成功（指標がチャートに接続されている場合）
- 端末が接続された口座への接続
- 指標の入力パラメータの変更

指標は、次のような場合にアンロードされます。

- 指標のチャートからの取り外し
- 端末のシャットダウン（指標がチャートに接続されている場合）
- テンプレートの読み込み（チャートに接続されている指標がテンプレートで指定されている場合）
- 指標が接続されたチャートの閉鎖

© 2000–2025, MetaQuotes Ltd.

- プロファイルの変更（指標がプロファイルチャートに接続されている場合）
- 端末が接続された口座への接続
- 指標が接続されたチャートのシンボル及び/または時間軸の変更
- 指標の入力パラメータの変更

エキスパートアドバイザーのロードとアンロード

エキスパートアドバイザーは、次の場合に読み込まれます。

- エキスパートアドバイザーのチャートへの接続
- 端末の開始（エキスパートアドバイザーが端末のシャットダウン前にチャートに接続された場合）
- テンプレートの読み込み（チャートに接続されているエキスパートアドバイザーがテンプレートで指定されている場合）
- プロファイルの変更（エキスパートアドバイザーがプロファイルチャートに接続されている場合）
- 口座への接続（口座番号が同一の場合も含む、エキスパートアドバイザーがサーバ上の端末の許可前にチャートに接続された場合）

エキスパートアドバイザーは、次の場合にアンロードされます。

- エキスパートアドバイザーのチャートからの取り外し
- 新しいエキスパートアドバイザーのチャートへの接続（他のエキスパートアドバイザーが既に接続されている場合、このエキスパートアドバイザーはアンロードされます）
- 端末のシャットダウン（エキスパートアドバイザーがチャートに接続されている場合）
- テンプレートの読み込み（チャートに接続されているエキスパートアドバイザーがテンプレートで指定されている場合）
- 指標が接続されたチャートの閉鎖
- プロファイルの変更（エキスパートアドバイザーがプロファイルチャートに接続されている場合）
- 端末が接続された口座への接続（エキスパートアドバイザーがサーバ上の端末の許可前にチャートに接続された場合）
- ExpertRemove() 関数の呼び出し

エキスパートアドバイザーが接続されているチャートのシンボルや時間軸が変更した場合、エキスパートアドバイザーはロードやアンロードされません。この場合、クライアント端末が古い銘柄/時間軸で OnDeinit() ハンドラを呼び、新しい銘柄/時間軸が存在する場合はその値で OnInit() を呼びます。グローバル変数及び静的変数の値はリセットされません。初期化の前にエキスパートアドバイザーのために受信された全てのイベントは完了します（OnInit() 関数は抜かされます）。

スクリプトのロードとアンロード

スクリプトは、チャートへの取り付けの直後に読み込みられ、操作を完了した直後にアンロードされます。OnInit() 及び OnDeinit() はスクリプトでは呼ばれません。

クライアント端末はプログラムアンロード（チャートからの削除）時にグローバル変数の初期化解除を行い、イベントキューを削除します。この場合、初期化解除とは、全ての string 型変数のリセット、動的配列オブジェクトの割り当て解除、及び存在する場合はデストラクタの呼び出しです。

© 2000–2025, MetaQuotes Ltd.

サービスのロードとアンロード

ターミナルのシャットダウン時にサービスが起動された場合は、ターミナルの起動直後にサービスがロードされます。サービスは、作業が完了した直後にアンロードされます。

サービスには単一のOnStart()ハンドラがあります。このハンドラでは、ネットワーク機能を使用してカスタム銘柄を作成および更新するなど、無限のデータ受信および処理ループを実装できます。

エキスパートアドバイザー、指標、スクリプトとは異なり、サービスは特定のチャートに縛られていないため、それらを起動するための別のメカニズムが提供されています。[サービスの追加]コマンドを使用して、新しいサービスインスタンスをナビゲータに作成します。適切なインスタンスメニューを使用して、サービスインスタンスを起動、停止、削除できます。すべてのインスタンスを管理するには、サービスメニューを使用します。

エキスパートアドバイザー操作をより良く理解するために、以下のエキスパートアドバイザーのコードをコンパイルし、ロード/アンロード、テンプレートの変更、シンボルの変更、時間軸の変更などのアクションを実行することをお勧めします。

例:

```
//+------------------------------------------------------------------+
//|                                                   TestExpert.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

class CTestClass
  {
public:
   CTestClass() { Print("CTestClass constructor"); }
   ~CTestClass() { Print("CTestClass destructor"); }
  };
CTestClass global;
//+------------------------------------------------------------------+
//| エキスパート初期化に使用される関数                                |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   Print("Initialization");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| エキスパート初期化解除に使用される関数                            |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Print("Deinitialization with reason",reason);
  }
//+------------------------------------------------------------------+
//| エキスパートティック関数                                         |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
```

参照

クライアント端末イベント、イベントハンドラ

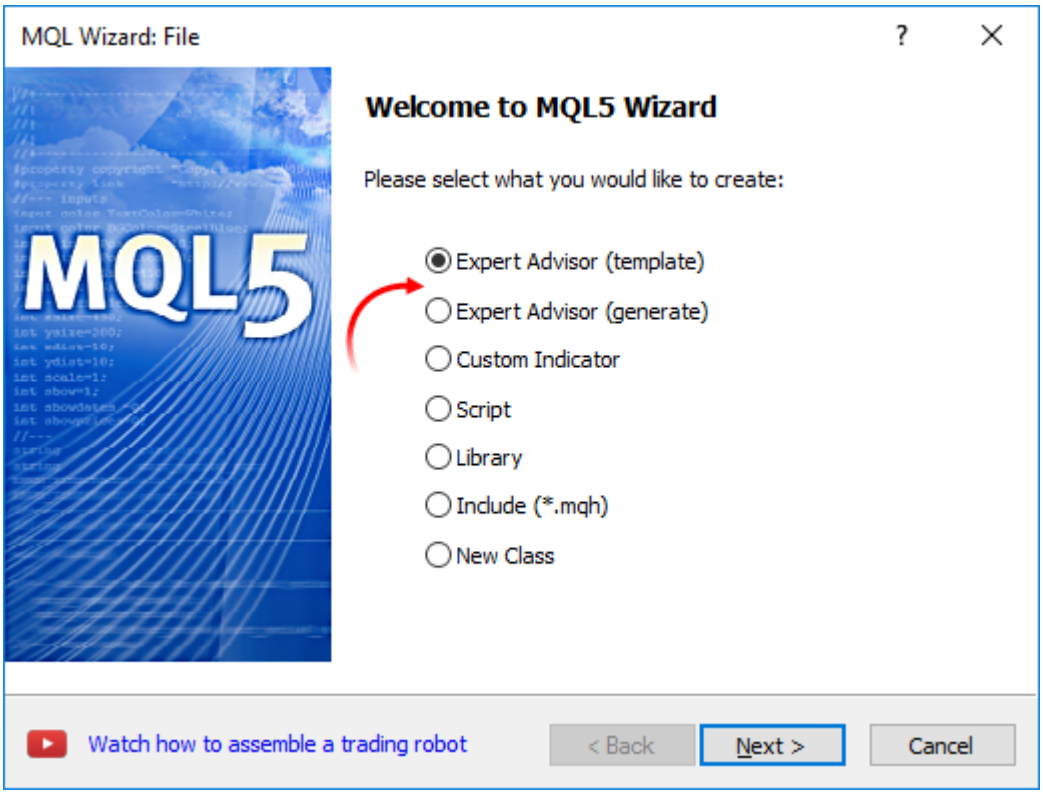
© 2000–2025, MetaQuotes Ltd.

取引許可

取引の自動化

MQL5 言語は自動売買システムの開発に意図された取引関数 を提供します。人間の介入なしの自動売買のために開発されたプログラムは「エキスパートアドバイザー(EA)」または「取引ロボット」と呼ばれます。MetaEditor でエキスパートアドバイザーを作成するには MQL5 Wizard を起動し、次の 2 つの選択肢の 1 つを選択します。

- Expert Advisor (template) – 既製のイベント処理の関数を持ち、プログラミングによって必要な関数を全て補足出来る、テンプレートを作成することが出来ます。
- Expert Advisor (generate) – 単に売買シグナルモジュール、資金管理モジュールとトレイリングストップモジュールなどの必要なモジュールを選択して本格的な取引ロボット開発 が出来ます。



取引関数はエキスパートアドバイザーと指標のみで動作出来ます。指標では取引が出来ません。

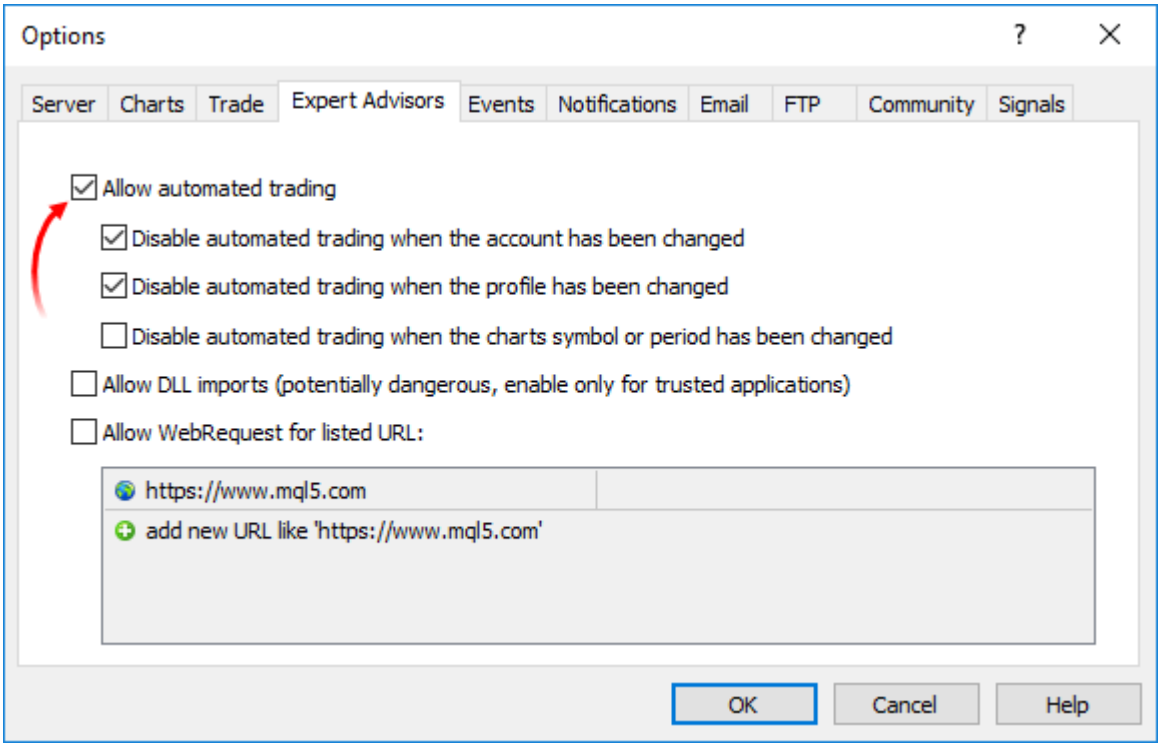
自動売買実行のための権限の確認

人間の介入なしで作業する信頼性のあるエキスパートアドバイザーを開発するためには、いくつかの重要なチェックが必要がです。まず、取引が許可されているかどうかをプログラム的にチェックする必要があります。これは、自動化システムを開発する際に不可欠である基本的なチェックです。

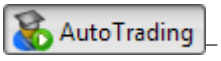
端末で自動売買が許可されているかどうかのチェック

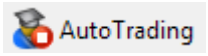
端末の設定が全てのプログラムの自動売買を許可または禁止する機能を提供します。

© 2000–2025, MetaQuotes Ltd.



自動売買オプションは端末の標準パネルで切り替えることが出来ます。

- 

– 自動売買が有効。起動したアプリケーションでの取引関数の使用が許可されています。
- 

– 自動売買が無効。実行されたアプリケーションは取引関数を実行することが出来ません。

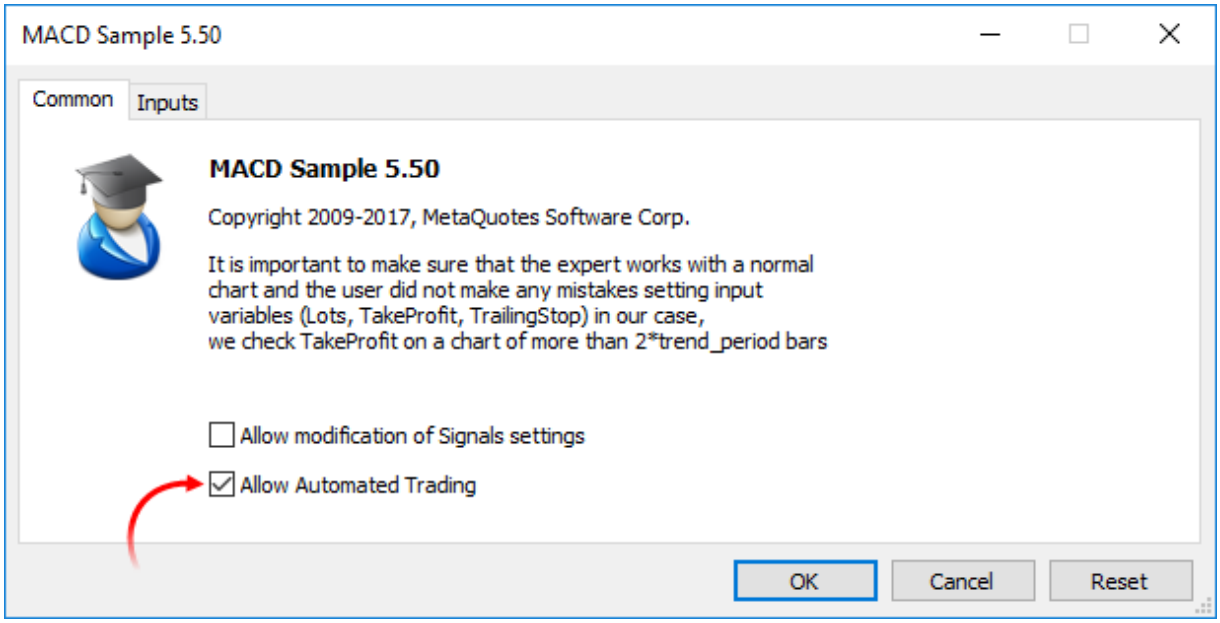
チェックの例

```
if (!TerminalInfoInteger(TERMINAL_TRADE_ALLOWED))
   Alert("Check if automated trading is allowed in the terminal settings!");
```

実行中のエキスパートアドバイザーまたはスクリプトで売買が許可されているかのチェック

特定のプログラムの自動売買を起動時に許可または禁止することが出来ます。これを行うには、プログラムのプロパティでの特殊なチェックボックスを使用します。

© 2000–2025, MetaQuotes Ltd.



チェックの例

```
   if(!TerminalInfoInteger(TERMINAL_TRADE_ALLOWED))
      Alert("Check if automated trading is allowed in the terminal settings!");
   else
     {
      if(!MQLInfoInteger(MQL_TRADE_ALLOWED))
         Alert("Automated trading is forbidden in the program settings for
",__FILE__);
     }
```

実行中のエキスパートアドバイザーまたはスクリプトで現在の口座での売買が許可されているかのチェック

自動売買はサーバ側で無効に出来ます。チェックの例

```
   if(!AccountInfoInteger(ACCOUNT_TRADE_EXPERT))
      Alert("Automated trading is forbidden for the account
",AccountInfoInteger(ACCOUNT_LOGIN),
      " at the trade server side");
```

自動売買が取引口座で無効にされている場合、エキスパートアドバイザーやスクリプトの取引操作は実行されません。

売買が現在の口座で許可されているかのチェック

特定の取引口座での取引操作が無効になっていて、手動/自動にかかわらず売買を行うことが出来ない場合があります。投資家のパスワードが取引口座への接続に使用されているかどうかをチェックする例：

```
   if(!AccountInfoInteger(ACCOUNT_TRADE_ALLOWED))
      Comment("Trading is forbidden for the account
",AccountInfoInteger(ACCOUNT_LOGIN),
```

© 2000–2025, MetaQuotes Ltd.

```
            ".\n Perhaps an investor password has been used to connect to the
trading account.",
            "\n Check the terminal journal for the following entry:",
            "\n\'",AccountInfoInteger(ACCOUNT_LOGIN),"\': trading has been disabled
- investor mode.");
```

AccountInfoInteger(ACCOUNT_ TRADE_ ALLOW ED) は次 の場 合 に **false** を返 すことがあります。

- 取 引 サーバへの接 続 がない(この状 態 は TerminalInfoInteger(TERMINAL_ CONNECTED) でチェック出 来 ます。)
- 取 引 口 座 が読 み込 み専 用 モードに切 り替 えられている(アーカイブに送 信 された)
- 口 座 での売 買 がサーバ側 で無 効 にされた
- 取 引 口 座 への接 続 が投 資 家 モードで実 行 されている

参 照

クライアント端 末 プロパティ、口 座 プロパティ、MQL5 プログラム実 行 のプロパティ

© 2000–2025, MetaQuotes Ltd.

クライアント端末イベント

Init

Init イベントは、クライアント端末プログラム(エキスパートアドバイザーまたはカスタム指標)が読み込まれてグローバル変数の初期化プロセスが開始した直後に送信され、存在する場合 OnInit() イベントハンドラで処理されます。このイベントはまた、金融商品及び/またはチャート時間軸の変更後、MetaEditor でのプログラムの 再コンパイル後、エキスパートアドバイザーの設定ウィンドウまたはカスタム指標での入力パラメータの変更後にも生成されます。エキスパートアドバイザーは口座変更後も初期化されます。Init イベントはスクリプトでは生成されません。

Deinit

グローバル変数が初期化解除されプログラム(エキスパートアドバイザーまたはカスタム指標)がアンロードされる前に、クライアント端末はプログラムに Deinit イベントを送信します。Deinit はまた、クライアント端末の閉鎖時、チャートの閉鎖時、セキュリティ及び/または時間軸の変更直前、プログラムの再コンパイル成功時、入力パラメータの変更時、及び口座変更時に生成されます。

初期化解除の理由は OnDeinit() 関数に渡されたパラメータによって取得出来ます。OnDeinit() 関数の実行は 2.5 秒に制限されています。この間に完了しない場合、この関数は強制的に終了されます。Deinit イベントはスクリプトでは生成されません。

Start

Startは、スクリプトまたはサービスを読み込んだ後に起動するための特別なイベントです。これはOnStart関数によって処理されます。Startイベントはおよびカスタム指標には渡されません。

NewTick

NewTick イベントは新しい相場が存在する時に生成され、接続されたエキスパートアドバイザーの OnTick() で処理されます。新しい相場を受信した際に前の相場のための OnTick 関数が処理中の場合、対応するイベントはキューに追加されないので、新しい相場はエキスパートアドバイザーによって無視されます。

プログラムの実行中に受信された新しい相場は全て OnTick() が終了するまで無視されます。その後、この関数は新しい相場が受信された後にのみ実行されます。NewTick イベントは自動取引が許可されているか否か(「自動売買」ボタン)に関係なく生成されます。自動売買の禁止は、エキスパートアドバイザーが動作し続ける一方でエキスパートアドバイザーからの取引リクエストの送信が許可されないことだけを意味します。

ボタンを押しての自動取引の禁止は実行中の OnTick() 関数を停止しません。

Calculate

Calculate イベントは Init イベントの送信直後と価格データの変更時に指標のみに発生します。このイベントは OnCalculate 関数で処理されます。

Timer

Timer イベントは EventSetTimer 関数でタイマーを起動したエキスパートアドバイザーのクライアント端末によって定期的に生成されます。通常この関数は OnInit で呼び出されます。Timer イベント処理は

© 2000–2025, MetaQuotes Ltd.

OnTimer 関数で行われます。エキスパートアドバイザーの操作終了後、タイマーは通常 OnDeinit 関数で呼ばれる EventKillTimer 関数を使用して破壊されるべきです。

Trade

Trade イベントは取引操作が取引サーバで完了した時に生成されます。Trade イベントは下記の取引操作時に OnTrade() 関数で処理されます。

- 未決注文の変更または削除
- 不充分な資金または期限切れによる未決注文の取り消し
- 未決注文のアクティベーション
- 注文、ポジション(またはその一部)の追加または決済
- ポジションの変更(決済逆指及び/または決済指の値の変更)

TradeTransaction

取引口座にいくつかの明確なアクションを実行すると、状態が変更されます。アクションの例は下記です。

- クライアント端末内の任意の MQL5 アプリケーションから OrderSend と OrderSendAsync 関数を使用して取引リクエストを送信し、取引が実行される。
- 端末のグラフィカルインターフェースを使用して取引リクエストを送信し、取引が実行される。
- サーバ上で未決注文と逆指値注文が執行される。
- 取引サーバ側で操作を行う。

これらのアクションの結果として以下の取引トランザクションが実行されます。

- 取引リクエストの処理
- 未執行注文の変更
- 注文履歴の変更
- 約定履歴の変更
- ポジションの変更

例えば、買いの成行注文を送信すると、注文が処理され、買い注文が口座に記録されます。その後注文が実行されオープン注文の表から削除されて注文履歴に追加されます。 約定が履歴に追加され新しいポジションが作成されます。これらのアクションは全て取引トランザクションです。このようなトランザクションの端末への到着が TradeTransaction イベントです。このイベントは OnTradeTransaction 関数で処理されます。

Tester

Tester イベントはエキスパートアドバイザーの履歴データテストの終了後に生成されます。このイベントは OnTester() 関数で処理されます。

TesterInit

TesterInit イベントは、ストラテジーテスターでの最適化の開始の際に初めの最適化パスの前に生成されます。TesterInit イベントは OnTesterInit() 関数で処理されます。

TesterPass

© 2000–2025, MetaQuotes Ltd.

TesterPass イベントは新しい データフレームが受信された際に生成されます。TesterPass イベントは OnTesterPass() 関数で処理されます。

TesterDeinit

TesterDeinit イベントはストラテジーテスターでのエキスパートアドバイザーの最適化の終了時に生成されます。TesterDeinit イベントは OnTesterDeinit() 関数で処理されます。

ChartEvent

ChartEvent はユーザがチャートを操作する時にクライアント端末によってイベント作成されます。

- チャートウィンドウがフォーカスされている時のキーストローク
- グラフィックオブジェクト作成
- グラフィックオブジェクト削除
- チャートのオブジェクトのマウスプレス
- マウスでのグラフィックオブジェクトの移動
- LabelEdit でのテキスト編集の終了

また、EventChartCustom 関数を使用して、任意の MQL5 プログラムによってエキスパートアドバイザーに送信することが出来るChartEvent カスタムイベントがあります。このイベントは OnChartEvent 関数で処理されます。

BookEvent

BookEvent は板情報の変更後にクライアント端末で生成され OnBookEvent 関数で処理されます。指定されたシンボルの BookEvent の生成を始めるには MarketBookAdd 関数を使用してシンボルをイベントにサブスクライブする必要があります。

BookEvent から指定されたシンボルのサブスクライブを停止するには MarketBookRelease 関数を呼ぶ必要があります。BookEvent イベントは、ブロードキャストイベントです。つまり、イベントに 1 つのエキスパートアドバイザーをサブスクライブすれば、OnBookEvent イベントハンドラを持つ他の全てのエキスパートアドバイザーがそれを受信出来ます。これがパラメータとしてハンドラに渡された銘柄名を分析する必要がある理由です。

参照

イベントハンドラ、プログラムの実行

© 2000–2025, MetaQuotes Ltd.

リソース

MQL5 プログラムでのグラフィック及び音声リソースの使用

MQL5 プログラムは音声及びグラフィックファイルの操作を可能にします。

- PlaySound()は音声ファイルを再生します。
- ObjectCreate() は OBJ_BITMAP 及び OBJ_BITMAP_LABEL グラフィックオブジェクトを使用してのユーザインターフェースの作成を可能にします。

PlaySound()

PlaySound() 関数呼び出しの例

```
//+------------------------------------------------------------------+
//|  標準的な OrderSend() を呼んで音を再生する                       |
//+------------------------------------------------------------------+
void OrderSendWithAudio(MqlTradeRequest  &request, MqlTradeResult &result)
  {
  //--- サーバにリクエストを送信する
   OrderSend(request,result);
   //--- リクエストが受け入れられたら Ok.wav を再生する
   if(result.retcode==TRADE_RETCODE_PLACED) PlaySound("Ok.wav");
   //--- 失敗したら timeout.wav ファイルの警笛を鳴らす
   else PlaySound("timeout.wav");
  }
```

この例は標準端末パッケージに含まれている「Ok.wav」と 「timeout.wav」の音の再生の仕方を示します。これらのファイルは terminal_directory\Sounds フォルダに位置します。ここで terminal_directory は、MetaTrader 5 クライアント端末が起動されるフォルダです。terminal_directory の位置は、MQL5 プログラムから次のように見つけることが出来ます。

```
//--- 端末データの格納フォルダ
   string terminal_path=TerminalInfoString(TERMINAL_PATH);
```

terminal_directory\Sounds フォルダだけでなくterminal_data_directory¥MQL5 のサブフォルダの音声ファイルも利用可能です。端末データディレクトリは端末メニュの「ファイル」 -> 「データフォルダを開く」またははプログラムのメソッドを使用して見つけることが出来ます。

```
//--- 端末データの格納フォルダ
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
```

例えば、Demo.wav 音声ファイルが terminal_data_directory¥MQL5¥Files に位置する場合、PlaySound() の呼び出しは次のように書かれるべきです。

```
//--- terminal_directory_data¥MQL5¥Files¥Demo.wav を再生する
   PlaySound("\\Files\\Demo.wav");
```

コメントではファイルパスが「￥」バックスラッシュで書かれているのに比べ、関数では「¥¥」が使用されているのにご注意ください。

© 2000-2025, MetaQuotes Ltd.

単一のバックスラッシュはコンパイラがプログラムソースコードの定数文字列及び文字定数扱う際の制御シンボルであるため、パスの指定時は常にセパレータとして二重のバックスラッシュを使用します。

再生を停止するには PlaySound() 関数を NULL パラメータで呼び出します。

```
//--- PlaySound() の NULL パラメータトの呼び出しは再生を停止する
   PlaySound(NULL);
```

ObjectCreate()

ObjectCreate() 関数でグラフィックレベル(OBJ_BITMAP_LABEL)を作成するエキスパートアドバイザーの例

```
string label_name="currency_label";          // OBJ_BITMAP_LABEL オブジェクトの名称
string euro       ="\\Images\\euro.bmp";     //
terminal_data_directory¥MQL5¥Images¥euro.bmp ファイルへのパス
string dollar     ="\\Images\\dollar.bmp";   //
terminal_data_directory¥MQL5¥Images¥dollar.bmp ファイルへのパス
//+------------------------------------------------------------------+
//| エキスパート初期化に使用される関数                                |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- まだ作成されていなければ OBJ_BITMAP_LABEL ボタンを作成する
   if(ObjectFind(0,label_name)<0)
     {
      //--- OBJ_BITMAP_LABEL オブジェクト作成を試みる
      bool created=ObjectCreate(0,label_name,OBJ_BITMAP_LABEL,0,0,0);
      if(created)
        {
         //--- ボタンをチャートの左上にリンクする
         ObjectSetInteger(0,label_name,OBJPROP_CORNER,CORNER_RIGHT_UPPER);
         //--- オブジェクトプロパティを設定する
         ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,100);
         ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,50);
         //--- 直近エラーの値を 0 にリセットする
         ResetLastError();
         //--- ボタンの「押された」状態を示す画像をダウンロードする
         bool set=ObjectSetString(0,label_name,OBJPROP_BMPFILE,0,euro);
         //--- 結果をテストする
         if(!set)
           {
            PrintFormat("Failed to download image from file %s. Error code %
d",euro,GetLastError());
           }
         ResetLastError();
         //--- ボタンの「押されていない」状態を示す画像をダウンロードする
         set=ObjectSetString(0,label_name,OBJPROP_BMPFILE,1,dollar);

         if(!set)
```

© 2000-2025, MetaQuotes Ltd.

```
            {
             PrintFormat("Failed to download image from file %s. Error code %
d",dollar,GetLastError());
            }
          //--- ティックがなくてもボタンが即時表示されるようにチャートにリフレッシュ指令を送る
          ChartRedraw(0);
         }
       else
         {
          //--- オブジェクト作成に失敗したので通知する
          PrintFormat("Failed to create object OBJ_BITMAP_LABEL. Error code %
d",GetLastError());
         }
      }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| エキスパート初期化解除に使用される関数                                  |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- チャートからオブジェクトを削除する
   ObjectDelete(0,label_name);
  }
```

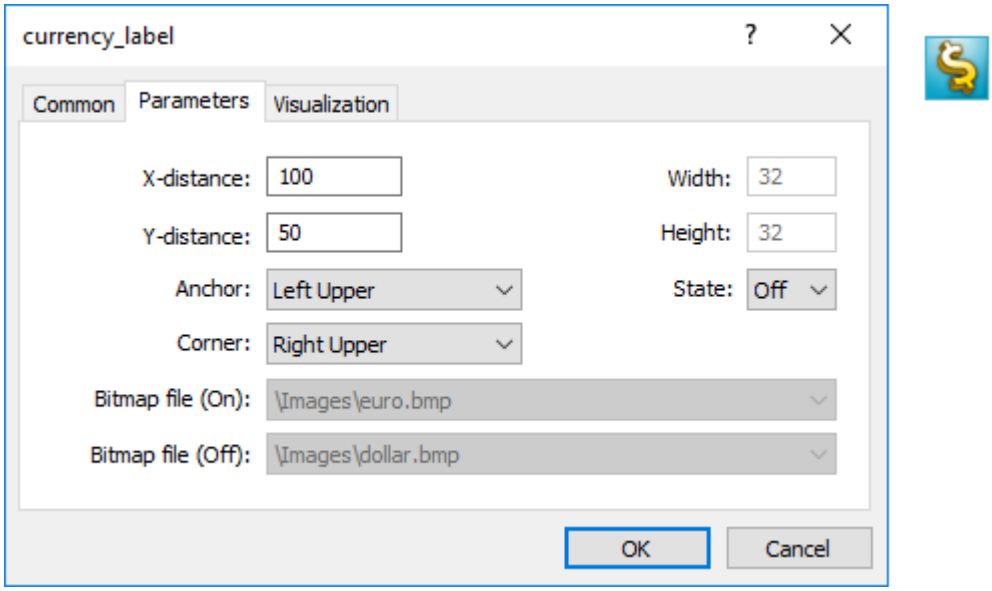
OnInit() 関数で currency_label という名称のグラフィックオブジェクトが作成/設定されます。グラフィックファイルへのパスは **euro** 及び **dollar** グローバル変数で設定され、二重のバックスラッシュが区切り文字として使用されます。

```
string euro       ="\\Images\\euro.bmp";     //
terminal_dara_directory¥MQL5¥Images¥euro.bmp ファイルへのパス
string dollar     ="\\Images\\dollar.bmp";   //
terminal_dara_directory¥MQL5¥Images¥dollar.bmp ファイルへのパス
```

ファイルは **terminal_data_directory**\MQL5¥Images フォルダに位置します。

OBJ_BITMAP_LABEL オブジェクトは、実際にはボタンの状態（押されたまたは押されていない状態）に応じて euro.bmp または dollar.bmp を表示するボタンです。

© 2000-2025, MetaQuotes Ltd.



グラフィカルインターフェースとのボタンのサイズは自動的に画像の大きさに調整されます。画像は OBJ_BITMAP_LABEL オブジェクト上のマウスの左ボタンのクリックによって変更されます（**プロパティで「選択を無効化」が設定されている必要があります**）。OBJ_BITMAP オブジェクトは同じように作成されます。このオブジェクトは背景を必要な画像と作成するのに使用されます。

OBJ_BITMAP 及び OBJ_BITMAP_LABEL オブジェクトの外観を担当するOBJPROP_BMPFILE プロパティの値は動的に変更出来ます。これによって MQL5 プログラムのための様々なインタラクティブなユーザインタフェースの作成が出来ます。

MQL5 プログラムのコンパイル時の実行可能ファイルへのリソースの埋め込み

MQL5 プログラムはダウンロード可能な多くの映像と音声ファイルの形式のリソースを必要とする場合があります。MQL5 実行ファイルの移動時にこれらのファイルを転送する必要性をなくすには、コンパイラの #resource ディレクティブが使用されるべきです。

```
#resource path_to_resource_file
```

#resource コマンドはコンパイラに指定されたパス **path_to_resource_file** でのリソースが実行可能な EX5 ファイルを埋め込むように指令します。このように、別の端末上でプログラムを実行する場合、必要な画像や音声ファイルの全ては EX5 ファイルに直接配置することが出来るので、別途に転送される必要はありません。リソースはEX5 ファイルに埋め込むことができ、任意の EX5 プログラムは別の EX5 プログラムからのリソースを使用することが出来ます。

BMP 及び WAV 形式のファイルは EX5 ファイルに埋め込まれる前に自動的に圧縮されます。これは、グラフィックと音声を使用する場合にリソースを使用すると、MQL5 プログラムの通常の書き込みの方法に比べて必要なファイルの合計サイズを小さくすることが可能であることを意味します。

リソースファイルサイズは 16 MB を超えてはいけません。

コンパイラによる指定されたリソースの検索

© 2000–2025, MetaQuotes Ltd.

リソースは #resource "< リソースファイルへのパス >" 指令を使用して挿入されます。

```
#resource "< リソースファイルへのパス >"
```

< リソースファイルへのパス > の定数文字列は 63 文字を超えてはいけません。

コンパイラはリソースを指定されたパスで次の順に探します。

- (「¥」として書かれた) バックスラッシュ「¥」がパスの始めにある場合、リソース検索は terminal_data_directory¥MQL5¥ ディレクトリに相対して行われます。
- バックスラッシュがない場合、リソースが書かれたソースファイルと相対して検索が行われます。

リソースパスは「..¥」及び「:¥」を含むことが出来ません。

リソース埋め込みの例

```
//--- 正しいリソースの指定法
#resource "\\Images\\euro.bmp" // euro.bmp は terminal_data_directory¥MQL5¥Images¥ に位置する
#resource "picture.bmp"        // picture.bmp はソースファイルと同じディレクトリに位置する
#resource "Resource\\map.bmp"  // リソースは source_file_directory¥Resource¥map.bmp

//--- 不正なリソースの指定
#resource ":picture_2.bmp"     // 「 : 」を含む
#resource "..\\picture_3.bmp"  // 「 .. 」を含む
#resource "\\Files\\Images\\Folder_First\\My_panel\\Labels\\too_long_path.bmp" // 63 記号以上
```

リソースの使用

リソース名

#resource ディレクティブでの宣言後、リソースはプログラムの任意の部分で使用出来ます。リソースの名称は、リソースへのパスを設定する行の先頭のバックスラッシュを付けないパスです。自分自身のリソースを使用するにはリソース名の前に「::」特殊記号が加えられるべきです。

例:

```
//--- リソース指定とコメント内での名称の例
#resource "\\Images\\euro.bmp"           // リソース名 - Images¥euro.bmp
#resource "picture.bmp"                  // リソース名 - picture.bmp
#resource "Resource\\map.bmp"            // リソース名 - Resource¥map.bmp
#resource "\\Files\\Pictures\\good.bmp" // リソース名 - Files¥Pictures¥good.bmp
#resource "\\Files\\Demo.wav";           // リソース名 - Files¥Demo.wav"
#resource "\\Sounds\\thrill.wav";        // リソース名 - Sounds¥thrill.wav"
...

//--- リソースの使用
ObjectSetString(0,bitmap_name,OBJPROP_BMPFILE,0,"::Images\\euro.bmp");
...
ObjectSetString(0,my_bitmap,OBJPROP_BMPFILE,0,"::picture.bmp");
```

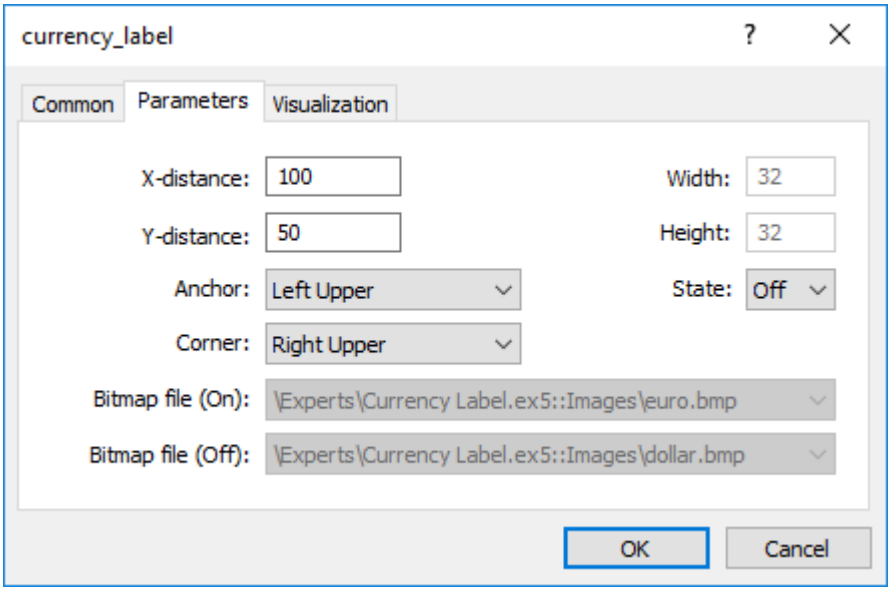
© 2000-2025, MetaQuotes Ltd.

```
...
set=ObjectSetString(0,bitmap_label,OBJPROP_BMPFILE,1,"::Files\\Pictures\\good.bmp");
...
PlaySound("::Files\\Demo.wav");
...
PlaySound("::Sounds\\thrill.wav");
```

リソースファイルの画像を OBJ_BITMAP and OBJ_BITMAP_LABEL オブジェクトに設定する場合、OBJPROP_BMPFILE プロパティ値は手動で変更出来ません。例えば OBJ_BITMAP_LABEL の作成には euro.bmp 及び dollar.bmp を使用します。

```
#resource "\\Images\\euro.bmp";     // euro.bmp は
terminal_data_directory¥MQL5¥Images¥ に位置する
#resource "\\Images\\dollar.bmp";   // dollar.bmp は
terminal_data_directory¥MQL5¥Images¥ に位置する
```

このオブジェクトのプロパティを表示する際、プロパティの「ビットマップファイル（オン）」と「ビットマップファイル（オフ）」が淡色表示されていて、手動変更が不可能だことがわかります。



他の MQL5 プログラムのリソース使用

リソース使用には、MQL5 プログラムで別の EX5 ファイルのリソースを使用出来るというもう 1 つの利点があります。従って、1 つの EX5 ファイルのリソースは多くの MQL5 プログラムで使用することが出来ます。

別のファイルからのリソース名を使用するためには <path_EX5_file_name>::<resource_name> としての指定が必要です。例として Draw_Triangles_Script.mq5 スクリプトが triangle.bmp 画像ファイルのリソースを含むとします。

```
#resource "\\Files\\triangle.bmp"
```

スクリプト自体で使用されるためにはその名称は "Files¥triangle.bmp" となり、それを使用するためには「::」がリソース名に追加される必要があります。

```
//--- スクリプトでのリソースの使用
```

© 2000-2025, MetaQuotes Ltd.

```
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"::Files\\triangle.bmp");
```

別のプログラム（例えばエキスパートアドバイザー）から同じリソースを使用する場合、リソース名に terminal_data_directory\MQL5\ に相対したEX5ファイルのパスとEX5ファイルの名称 - Draw_Triangles_Script.ex5を加える必要があります。スクリプトが terminal_data_directory\MQL5\Scripts¥ 標準フォルダに位置する場合、呼び出しは次のように書かれるべきです。

```
//--- EA でのスクリプトのリソースの使用
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"\\Scripts\
\Draw_Triangles_Script.ex5::Files\\triangle.bmp");
```

別の EX5 からのリソース呼び出しの際に実行可能ファイルへのパスが指定されていない場合は、実行可能ファイルは、リソースを呼び出すプログラムが含まれている同じフォルダで検索されます。つまり、エキスパートアドバイザーがDraw_Triangles_Script.ex5 のリソースをパスの指定なしにこのように呼び出す場合

```
//--- EA でパスの指定なしでスクリプトリソースを呼ぶ
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"Draw_Triangles_Script.ex5::Files
\\triangle.bmp");
```

エキスパートアドバイザーが terminal_data_directory\MQL5\Experts\ に位置する場合、ファイルは terminal_data_directory¥MQL5¥で検索されます。

リソースとして含まれたカスタム指標の操作

MQL5 アプリケーションの動作には 1 つまたは複数のカスタム指標が必要であるかもしれません。それらの全ては、実行可能な MQL5 プログラムのコードに含めることが出来ます。指標をリソースとして含むことで、アプリケーションの配布が簡素化します。

下記はterminal_data_folder¥MQL5¥Indicators¥ ディレクトリ位置する SampleIndicator.ex5 カスタム指標を含んで使用する例です。

```
//+------------------------------------------------------------------+
//|                                                     SampleEA.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                             https://www.MQL5.com |
//+------------------------------------------------------------------+
#resource "\\Indicators\\SampleIndicator.ex5"
int handle_ind;
//+------------------------------------------------------------------+
//| エキスパート初期化に使用される関数                                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   handle_ind=iCustom(_Symbol,_Period,"::Indicators\\SampleIndicator.ex5");
   if(handle_ind==INVALID_HANDLE)
     {
      Print("Expert: iCustom call: Error code=",GetLastError());
      return(INIT_FAILED);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- ...
   return(INIT_SUCCEEDED);
  }
```

OnInit() 関数のカスタム指標が 1 つ以上の自己コピーをする場合、特別な考慮が必要です。リソースの指定は <path_EX5_file_name>::<resource_name> としてなされることにご注意下さい。

例えば SampleIndicator.ex5 指標が SampleEA.ex5 エキスパートアドバイザーに指標として含まれていて、カスタム指標の初期化関数で iCustom() が呼ばれる場合の位置表現は "¥¥Experts¥¥SampleEA.ex5::Indicators¥¥SampleIndicator.ex5" のようになります。このパスが明示的に設定されている場合、SampleIndicator.ex5 カスタム指標は厳格にSampleEA.ex5 エキスパートアドバイザーに接続されて、独立して動作する能力を失います。

自己のパスは GetRelativeProgramPath() 関数で受け取ることが出来ます。使用の例は下記の通りです。

```
//+------------------------------------------------------------------+
//|                                                SampleIndicator.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                             https://www.MQL5.com |
//+------------------------------------------------------------------+
#property indicator_separate_window
#property indicator_plots 0
int handle;
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 自己のリンクを提供する不正な方法
//--- 文字列パス ="¥¥Experts¥¥SampleEA.ex5::Indicators¥¥SampleIndicator.ex5";
//--- 自己のリンクを受け取る正しい方法
  string path=GetRelativeProgramPath();
//--- 指標バッファマッピング
   handle=iCustom(_Symbol,_Period,path,0,0);
   if(handle==INVALID_HANDLE)
     {
      Print("Indicator: iCustom call: Error code=",GetLastError());
      return(INIT_FAILED);
     }
   else Print("Indicator handle=",handle);
//---
   return(INIT_SUCCEEDED);
  }
///....
//+------------------------------------------------------------------+
//| GetRelativeProgramPath                                           |
//+------------------------------------------------------------------+
string GetRelativeProgramPath()
  {
```

```
   int pos2;
//--- アプリケーションへの絶対パスを取得する
   string path=MQLInfoString(MQL_PROGRAM_PATH);
//--- "¥MQL5¥" サブストリングのポジションを見つける
   int    pos =StringFind(path,"\\MQL5\\");
//--- サブストリングが見つからないので、エラー
   if(pos<0)
      return(NULL);
//--- "¥MQL5" ディレクトリをとばす
   pos+=5;
//--- 余分な「 ¥ 」記号をとばす
   while(StringGetCharacter(path,pos+1)=='\\')
      pos++;
//--- これがリソースならMQL5 ディレクトリに相対したパスを返す
   if(StringFind(path,"::",pos)>=0)
      return(StringSubstr(path,pos));
//--- MQL5¥Indicators のような最初の MQL5 サブディレクトリの区切りを見つける
//--- 見つからない場合 MQL5 ディレクトリに相対したパスを返す
   if((pos2=StringFind(path,"\\",pos+1))<0)
      return(StringSubstr(path,pos));
//--- サブディレクトリ(例えば MQL5¥Indicators )に相対したパスを返す
   return(StringSubstr(path,pos2+1));
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double& price[])
  {
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
```

リソース変数

リソースは、リソース変数を使用して宣言でき、それらが適切な型の変数と同様に扱われます。宣言の形式：

```
#resource path_to_the_resource_file as resource_variable_type resource_variable_name
```

宣言の例：

```
#resource "data.bin" as int ExtData[]              // data.binファイルからのデータを含む数値配列を宣言
#resource "data.bin" as MqlRates ExtData[]         // data.binファイルからのデータを含む単純構造体配列を宣言
```

© 2000-2025, MetaQuotes Ltd.

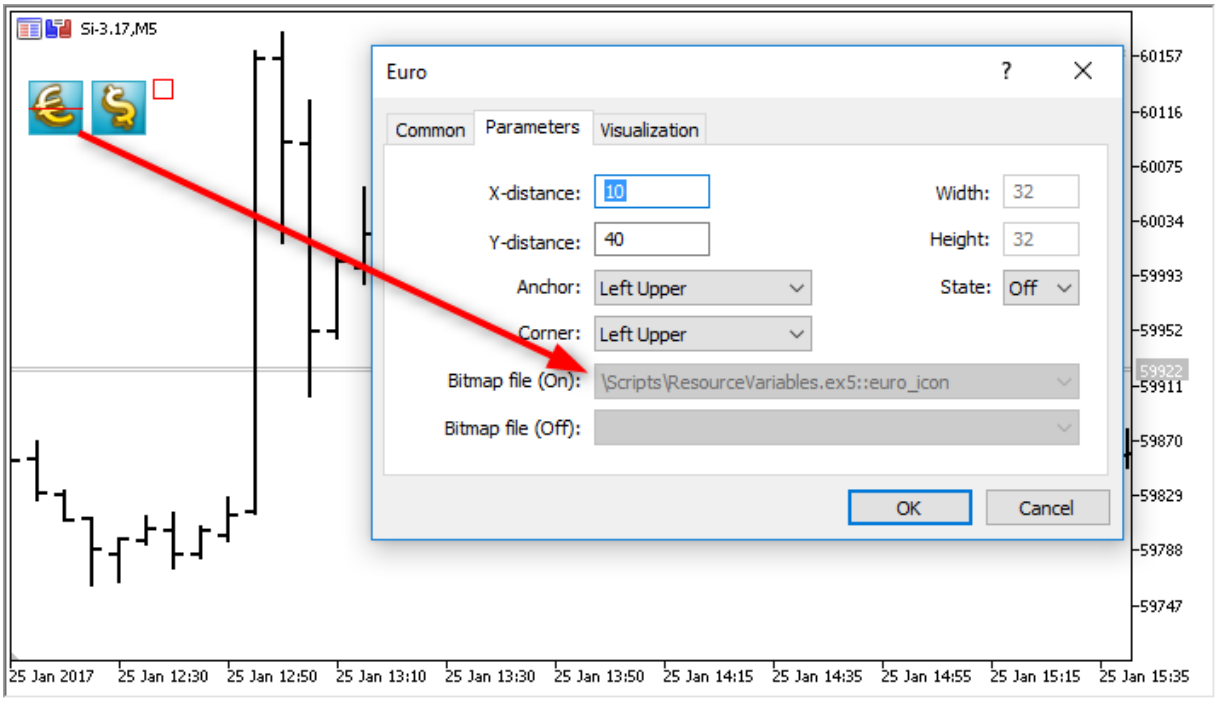
```
//--- 文字列
#resource "data.txt" as string ExtCode             // data.txt ファイルデータを含む文字列を宣言
(ANSI、UTF-8、UTF-16 エンコードがサポートされる)
//--- グラフィックリソース
#resource "image.bmp" as bitmap ExtBitmap[]        // BMPファイルからのビットマップを含む1次元配列
を宣言、array size = height * width
#resource "image.bmp" as bitmap ExtBitmap2[][]     // BMPファイルからビットマップを含む2次元配列を
宣言、array size [height][width]
```

このような宣言の場合、リソースデータは変数を介してのみのアドレス指定が可能で、"::<rsource name>"のような自動アドレス指定は動作しません。

```
#resource "\\Images\\euro.bmp" as bitmap euro[][]
#resource "\\Images\\dollar.bmp"
//+------------------------------------------------------------------+
//| リソースを使ったOBJ_BITMAP_LABELオブジェクト作成関数                |
//+------------------------------------------------------------------+
void Image(string name,string rc,int x,int y)
  {
   ObjectCreate(0,name,OBJ_BITMAP_LABEL,0,0,0);
   ObjectSetInteger(0,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(0,name,OBJPROP_YDISTANCE,y);
   ObjectSetString(0,name,OBJPROP_BMPFILE,rc);
  }
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- ユーロリソース変数に格納されている画像サイズ[幅、高さ]を出力する
   Print(ArrayRange(euro,1),", ",ArrayRange(euro,0));
//--- ユーロで画像を変更する - 中央に赤い水平ストライプを描画する
   for(int x=0;x<ArrayRange(euro,1);x++)
      euro[ArrayRange(euro,1)/2][x]=0xFFFF0000;
//--- リソース変数を使用してグラフィカルリソースを作成する
   ResourceCreate
("euro_icon",euro,ArrayRange(euro,1),ArrayRange(euro,0),0,0,ArrayRange(euro,1),COLOR
_FORMAT_ARGB_NORMALIZE);
//---euro_iconリソースからの画像が設定されるユーログラフィカルラベルオブジェクトを作成する
   Image("Euro","::euro_icon",10,40);
//--- リソースを適用するあと一つのメソッド、それへの描画は不可能
   Image("USD","::Images\\dollar.bmp",15+ArrayRange(euro,1),40);
//--- euro.bmpリソースをアドレス指定する直接的な方法は、すでにユーロリソース変数くによって宣言されているため利用
できない
   Image("E2","::Images\\euro.bmp",20+ArrayRange(euro,1)*2,40); // 実行時エラーが発生する
  }
```

スクリプト実行結果 - 3つのOBJ_BITMAP_LABELのうち2のみが作成されます。最初のオブジェクトの画像には、真ん中に赤いストライプがあります。

© 2000-2025, MetaQuotes Ltd.



リソースを適用することの重要な利点は、リソースファイルがコンパイル前に実行可能EX5ファイルに組み込まれる前に自動的に圧縮されることです。したがって、リソース変数を使用すると、実行可能なEX5ファイルにすべての必要なデータを直接入れることができ、MQL5プログラムを書く従来の方法と比較したファイルの数と合計サイズを減らすことができます。

リソース変数の使用はマーケットに製品を公開する場合に特に便利です。

機能

- 特別なbitmapリソース変数タイプは、リソースが画像であることをコンパイラに通知します。そのような変数はuint型を受け取ります。
- *bitmap* 型配列リソース変数は2次元を持つことができます。この場合、配列の大きさは [image_height] [image_width]. で定義されます。1次元配列が指定されている場合、要素の数はimage_height * image_widthに等しくなります。
- 24ビット画像のダウンロードではアルファチャネルコンポーネントは、すべての画像画素に対して255に設定されます。
- アルファチャネルなしでの32ビット画像のダウンロードでは、アルファチャネルコンポーネントもすべての画像画素に対して255に設定されます。
- アルファチャンネルを使用した32ビット画像のダウンロードでは、画素はまったく処理されません。
- リソースファイルサイズは128MBを超えてはいけません。
- 文字列ファイルに対してはBOM（ヘッダ）の存在による自動エンコード検出が実行されます。BOMが存在しない場合、エンコーディングはファイルの内容によって定義されます。ANSI、UTF-8およびUTF-16エンコーディングのファイルがサポートされています。すべての文字列は、ファイルからデータを読み込むときにUnicodeに変換されます。

OpenCLプログラム

ある種のプログラムの開発は、リソース文字列変数の使用によって大幅に容易になります。たとえばOpenCLプログラムのコードを別のCLファイルに書き込んだ後、文字列としてMQL5プログラムリソースに組み込むことが可能です。

© 2000–2025, MetaQuotes Ltd.

```
#resource "seascape.cl" as string cl_program
...
int context;
if((cl_program=CLProgramCreate(context,cl_program)!=INVALID_HANDLE)
  {
   //--- OpenCLプログラムでさらなる行動を実行する
  }
```

リソース変数が使用されない場合、この例は*cl_program*　コード全体の単一の大きな文字列としての記述が必要でした。

参照

ResourceCreate()、ResourceSave()、PlaySound()、ObjectSetInteger()、ChartApplyTemplate()、ファイル関数

© 2000–2025, MetaQuotes Ltd.

インポートした関数の呼び出し

MQL5 プログラムの実行時に関数をインポートするには、クライアント端末は事前結合を使用します。これは、プログラムがインポートされた関数を呼び出す場合は、対応するモジュール(EX5またはDLL)がプログラムロード中に読み込まれていることを意味します。MQL5 及び DLL ライブラリは呼び出しモジュールと同じスレッドで実行されます。

Drive:\Directory\FileName.Ext の様なモジュールの絶対パスの指定は推奨されていません。MQL5 ライブラリは *terminal_dir\MQL5\Libraries* フォルダから読み込まれます。ライブラリが見つからない場合、クライアント端末は *terminal_dir\experts* フォルダからの読み込みを試みます。

システムライブラリ(DLL)はオペレーティングシステムルールによって読み込みされます。ライブラリがすでに読み込みされている場合(例えば、別のエキスパートアドバイザーや更に別のクライアント端末が並列に実行されている)それは既に読み込まれたライブラリへのリクエストを使用します。それ以外の場合は、以下の順序で検索が実行されます。

1. モジュールをインポートする dll が開始されたディレクトリ(モジュールとは、エキスパートアドバイザー、スクリプト、指標または EX5 ライブラリ)
2. terminal_data_directory¥MQL5¥Libraries ディレクトリ(TERMINAL_DATA_PATH\MQL5\Libraries)
3. MetaTrader 5 クライアント端末が開始されたディレクトリ
4. システムディレクトリ
5. ウィンドウディレクトリ
6. 現在のディレクトリ
7. PATH システム変数にリストされたディレクトリ

DLL ライブラリが他の DLL を使用する場合、使用される DLL が存在しない場合、初めの DLL は読み込むことが出来ません。

エキスパートアドバイザー(スクリプト、指標)が読み込まれる前に、全ての EX5 のライブラリモジュールの共通リストが形成されます。これは、読み込まれたエキスパートアドバイザー(スクリプト、指標)とライブラリー両方から使用されます。従って、EX5 ライブラリモジュールの複数使用のために1 回のみの読み込みが必要とされています。ライブラリはエキスパートアドバイザー(スクリプト、指標)の事前定義された変数を使用します。

インポートされたライブラリ EX5 は次の順序で検索されます。

1. パスが EX5 をインポートしたエキスパートアドバイザー(スクリプト、指標)のディレクトリからの相対に設定されているディレクトリ
2. terminal_directory¥MQL5¥Libraries ディレクトリ
3. 全てのクライアント端末の共有ディレクトリ(Common¥MQL5¥Libraries)内の MQL5¥Libraries ディレクトリ

DLL を MQL5 プログラムにインポートした関数は Windows API 呼び出しルールを守らなければなりません。そのような合意を確保するには、C または C ++ で記述されたプログラムのソーステキストで Microsoft(r) コンパイラに固有のキーワード __stdcall を使用します。この合意は以下のことを特徴としています。

- 呼び出し側(MQ5 プログラム)は、スタックに正しくパラメータを組み合わせるために、(DLLからインポートされた)呼びだされる関数のプロトタイプを「見る」必要があります。
- 呼び出し側(MQ5 プログラム)は、インポートされた関数が渡されたパラメータをこの順に読み込むように、右から左の逆の順序でスタックにパラメータを置きます 。
- パラメータは、明示的に参照で渡されたもの(文字列)を除いて、値によって渡されます。
- インポートされた関数は、渡されたパラメータを読み込むことにより、独立的にスタックをクリーンします。

© 2000–2025, MetaQuotes Ltd.

インポートされた関数のプロトタイプを記述する場合、デフォルトのパラメータを使用することが出来ます。

対応するライブラリが読み込み出来ない場合、または DLL 使用が禁止されているかインポートされた関数が見つからない場合は、エキスパートアドバイザーは操作ログ(ログファイル)に「Expert Advisor stopped(エキスパートアドバイザー停止)」とのメッセージを書き入れて停止します。この場合、エキスパートアドバイザーは再初期化されるまで実行されません。エキスパートアドバイザーは、再コンパイルの結果として、またはそのプロパティの表が開かれ「OK」が押された後、再初期化することが出来ます。

パラメータの引き渡し

基本データ型のパラメータは全て明示的に参照で渡されるもの以外は値で受け渡されます。string の受け渡し時には、複製された文字列のバッファのアドレスが渡されます。文字列は参照で受け渡された場合、この文字列のバッファのアドレスが複製されずに DLL からインポートされた関数に渡されます。

動的配列、文字列、クラス、またその他の複雑な構造体を含む構造体や、列挙オブジェクトの静的または動的配列はパラメータとしてインポートされた関数に渡すことが出来ません。

配列を DLL に渡す場合 AS_SERIES フラグにかかわらず常にバッファの初めのアドレスが渡されます。DLL 内の関数は AS_SERIES フラグについては何も知らないので、渡された配列は不定のサイズ持つ静的配列となります。配列のサイズを指定するには追加のパラメータが使用されるべきです。

© 2000–2025, MetaQuotes Ltd.

ランタイムエラー

MQL5 実行中にエラーが発生した場合、クライアント端末の実行サブシステムはエラーコードを保存することが出来ます。それぞれの実行可能な MQL5 プログラムは事前定義された _LastError 変数を持っています。

_LastError 変数は OnInit 関数の開始前にリセットされます。計算中や内部関数呼び出しの過程で不正な状況が発生した場合 _LastError 変数は対応するエラーコードを受け付けます。この変数に保存された変数は GetLastError() 関数で取得出来ます。

プログラムの即時終了につながる重大なエラーはいくつかあります。

- ゼロ除算
- 配列の境界外のアクセス
- 不正なオブジェクトポインタの使用

© 2000–2025, MetaQuotes Ltd.

取引ストラテジーのテスト

自動売買のアイディアは、取引ロボットが四六時中ノンストップで働くことが出来るという事実によって魅力的です。ロボットは、心理的な問題から完全に解放されており、疲れたり、疑問を持ったり怖がったりしません。取引ルールを定式化してアルゴリズムで実装すれば、ロボットはたゆまず努力する準備ができています。しかし、初めに次の 2 つの重要な条件が満たされていることを確認する必要があります。

- エキスパートアドバイザーは取引システムの規則に従って取引操作を行う
- EA に実装された取引ストラテジーが履歴的に利益を示している

これらの質問に答えるためには MetaTrader 5 のクライアント端末に含まれるストラテジーテスターが使用されます。

このセクションでは、ストラテジーテスターでのプログラムのテストと最適化の機能について説明します。

- ストラテジーテスターでの機能の制限
- ティック生成モード
- スプレッドのシミュレーション
- テストでの実際ティックの使用
- クライアント端末のグローバル変数
- テスト中の指標の計算
- テスト中の履歴の読み込み
- 多通貨テスト
- ストラテジーテスターでの時間のシミュレーション
- テストでのグラフィックオブジェクト
- ストラテジーテスターでの OnTimer() 関数
- ストラテジーテスターでの Sleep() 関数
- 数値計算や最適化問題のためのストラテジーテスターの使用
- 「始値のみ」モードでのバーの同期
- テスタの IndicatorRelease() 関数
- テスタでのイベントハンドル
- テストエージェント
- 端末とエージェントの間でのデータ交換
- 全てのクライアント端末の共有フォルダの使用
- DLL の使用

MQL5 クラウドネットワークのメモリとディスク容量の制限

次の制限は、MQL5クラウドネットワークで実行される最適化に適用されます: エキスパートアドバイザー(EA)は、4GBを超える情報をディスクに書き込んだり、4GBを超えるRAMを使用したりしてはなりません。上限を超えると、ネットワークエージェントは計算を正しく完了できず、結果を受け取ることができなくなります。ただし、計算に費やした時間はすべて料金が発生します。

© 2000–2025, MetaQuotes Ltd.

各最適化パスから情報を取得する必要がある場合は、ディスクに書き込まずにフレームを送信します。MQL5クラウドネットワークでの計算中にEAでのファイル操作の使用を回避するには、次のチェックを使用できます。

```
int handle=INVALID_HANDLE;
bool file_operations_allowed=true;
if(MQLInfoInteger(MQL_OPTIMIZATION) || MQLInfoInteger(MQL_FORWARD))
   file_operations_allowed=false;

if(file_operations_allowed)
  {
   ...
   handle=FileOpen(...);
   ...
  }
```

ストラテジーテスターでの機能の制限

クライアント端末のストラテジーテスターの一部の関数には操作上の制限があります。

Comment()、Print() 及び PrintFormat() 関数

パフォーマンス向上のため、Comment()、Print() 及び PrintFormat() 関数は取引ロボットパラメータの最適化の際には実行されません。例外は OnInit() ハンドラでのこれらの関数の使用です。これによって、エラーが発生した時に簡単に原因を見つけることが出来ます

Alert()、MessageBox()、PlaySound()、SendFTP、SendMail()、SendNotification()、及び WebRequest() 関数

「外部世界」との相互作用のために設計されたAlert()、MessageBox()、PlaySound()、SendFTP()、SendMail()、SendNotification() 及び WebRequest() 関数はストラテジーテスターでは実行されません。

ティック生成モード

エキスパートアドバイザーは MQL5 で書かれた、外部のイベント対応するたびに実行されるプログラムです。エキスパートアドバイザーは事前定義されたイベントの 1 つに対応する関数（イベントハンドラ）を持っています。

NewTick イベント（価格変更）はエキスパートアドバイザーの主要なイベントなので、エキスパートアドバイザーのテストは連続してティックを作成することが必要です。MetaTrader 5 クライアント端末のストラテジーテスターには 3 つのティック生成のモードがあります。

- 全ティック
- 1 分足 OHLC（1 分足を持ったOHLC 価格）
- 始値のみ

© 2000–2025, MetaQuotes Ltd.

基本的で最も詳細なものは「全ティック」モードで、他の 2 つのモードは基本的なものを単純化したもので、「全ティック」モードに比較して説明されます。それらの違いを理解するために全ての 3 つのモードを考えてみましょう。

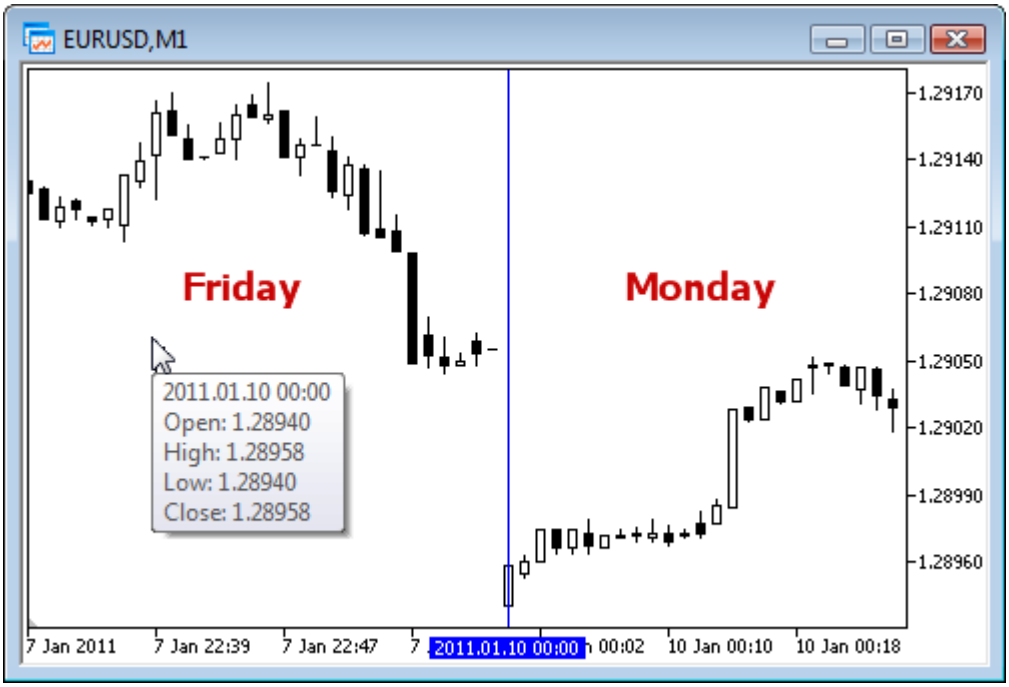
「全ティック」

金融商品の履歴相場データは、パックされた 1 分足の形で MetaTrader 5 クライアント端末に取引サーバから転送されます。リクエストの発生及び必要な時間フレームの構成に関する詳細な情報は MQL5 リファレンスのデータアクセスの整理チャプターで取得出来ます。

価格履歴の最小要素は1 分足で、価格の 4 値の情報を得ることが出来ます。

- 始値 – 1 分足が開かれた時の価格
- 高値 – 1 分足の間に達成された最大値
- 安値 – 1 分足の間に達成された最小値
- 終値 – 1 分足の終値

新しい1 分足は、新しい分が始まる(秒数が 0 に等しくなる)瞬間ではなく、価格が少なくても 1 ポイントによって変動しティックが発生した時に開かれます。この図は、2011年1月10日午前零時零分のオープン時間を持つ新たな取引週の最初の1　分足を示します。為替レートは、着信ニュースに反応して週末にでも変動するため、チャート上の金曜日と月曜日との間の価格差は一般的です。



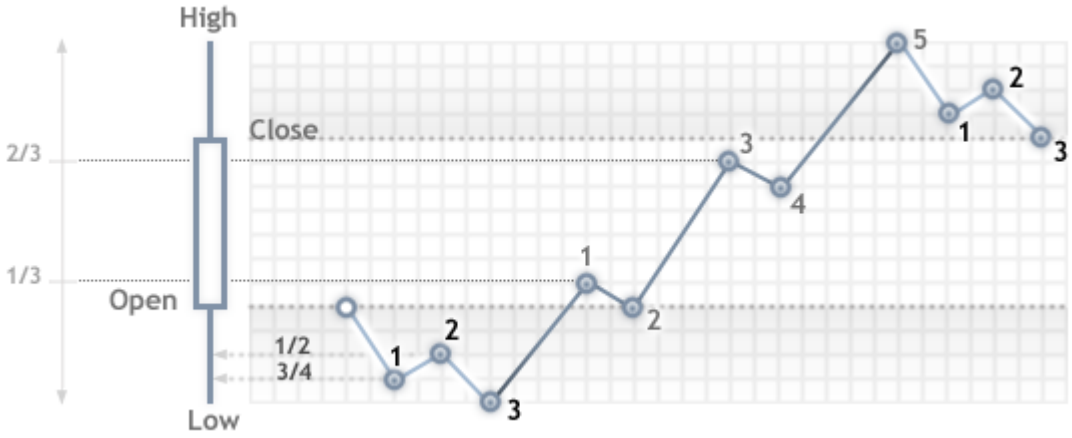
1 分足が 2011 年 1 月 10 日午前 0 時 0 分に開かれたことは分かっていますが、秒については何も分かりません。0 時 0 分 12秒 または 0 時 0 分 36秒(新しい日の始まりから 12 または 36 秒後)またはその分内のどの秒に開いたかもしれません。しかし新しい1 分足が開いた時に EURUSD の始値が 1.28940 であったことは分かります。

また、1 分足の終値に対応する2 番目のティックが受信された秒も分かりません。分かっているのは1 分足の最後の終値だけです。この分では、その価格は　1.28958　でした。高値と安値の出現時刻も不明ですが、値がそれぞれ 1.28958 と 1.28940 のレベルにあったことは分かります。

© 2000–2025, MetaQuotes Ltd.

取引ストラテジーのテストには、エキスパートアドバイザーの作業をシミュレートする連続したティックが必要です。全ての1分足での、価格が間違いなく位置した4制御点の場所は分かっています。足にティックが4つある場合、テスト実行には充分な情報ですが、ティックボリュームは通常4以上です。

従って、始値、高値、安値及び終値の間に発生したティックの追加制御点を生成する必要があります。「全ティック」の原則は、MetaTrader 5 ターミナルのストラテジーテスタ内でティック作成をするアルゴリズムに記述されており、下に図が引用されています。



「全ティック」モードのテストでは、エキスパートアドバイザーの OnTick() が全ての制御点で呼ばれます。各制御点は、生成されたシーケンスからのティックです。エキスパートアドバイザーは、オンラインで作業する場合と同じように、シミュレートされたティックの時間と価格を受信します。

> **重要事項**：「全ティック」テストモードは一番正確ですが、同時に一番時間がかかります。大部分の取引ストラテジーの初期テストとしては、通常、他の2つのテストモードのいずれかを使用することで充分です。

「1分足 OHLC」

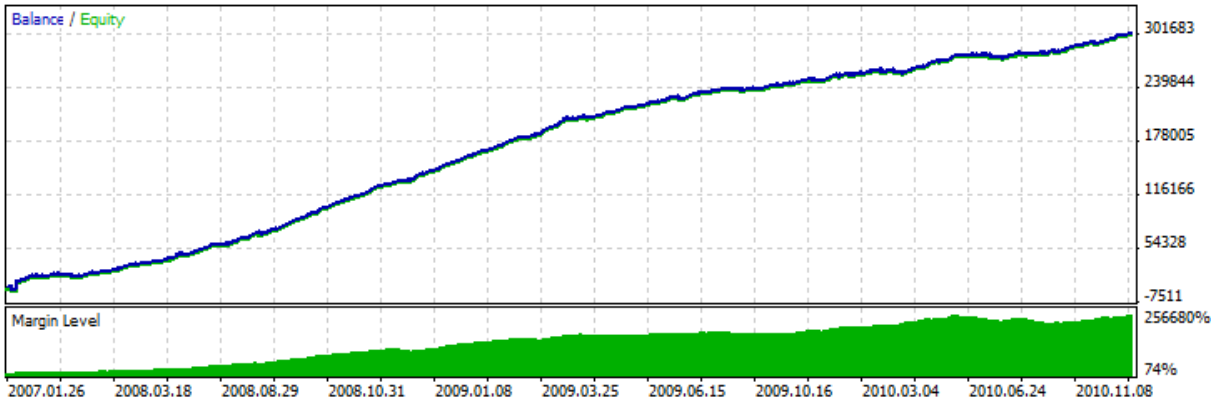
3つのモードのうちで「全ティック」モードは一番正確ですが、同時に一番低速です。ティックボリュームは非常に多くなりえるのですが、OnTick() ハンドラは全てのティックで起こります。足全体での価格変動のティックシーケンスが重要ではないストラテジーでは、より速く、より粗い「1分足 OHLC」シミュレーションモードがあります。

「1分足 OHLC」モードでは、ティックシーケンスは1分足の OHLC 価格でのみ構成され、生成された制御点の数が大幅に削減されます。従って、テスト時間も減少します。OnTick () 関数は1分足 OHLC の価格によって構築されている全ての制御点で起動されます。

始値、高値、安値及び終値以外での追加の中間ティックを生成しないことで、始値が決定された瞬間から、価格展開を決定します。これにより、テストバランスの素晴らしい上向きのグラフを示す「テストグレイル」を作成することが出来ます。

そのようなグレイルの例はコードベースの Grr-al でみられます。

© 2000–2025, MetaQuotes Ltd.



図は、このエキスパートアドバイザーテストの非常に魅力的なグラフを示しています。それはどのように取得されたのでしょうか？1分足の4価格は知られており、また、最初は始値で最後が終値であることも知られています。それらの間に高値と安値があり、発生順序は不明ですが、高値が始値以上であること（かつ安値が始値以下であること）も知られています。

始値を受信した瞬間を決定した後、現時点で持っている価格が高値か安値かを決定するために、次のティックを分析するので充分です。価格が始値を下回っている場合は、このティックでは安値と買いがあり、次のティックは、買いを閉めて売りに開く高値に対応します。終値である次のティックが最後で、ここでは売りを閉じます。

価格の後、始値よりも価格の大きいティックを受信した場合、取引の順番が逆になっています。　この「カンニング」モードで1分足を処理し、次のものを待ちます。

このようなエキスパートアドバイザーを履歴上でテストする場合全てがスムーズに行きますが、オンラインで起動すると、バランスラインが安定したままですが、下向きだという、真実が明らかになります。このトリックを公開するには、単に「全ティック」モードでエキスパートアドバイザーを実行するだけです。

> **注意事項：**　粗雑なテストモード（「1分　OHLC」と「始値のみ」）でのエキスパートアドバイザーのテスト結果があまりにも良すぎる場合は「全ティック」モードでテストしてみてください。

「始値のみ」

このモードでは、ティックはテストのために選択された時間軸の OHLC 価格に基づいて生成されます。エキスパートアドバイザーの OnTick() 関数は始値でのバーの初めでのみ実行されます。この機能により、ストップレベルや未決済は（特に高い時間軸のテストで）指定とは異なる価格でトリガされることがあります。代わりに、すぐにエキスパートアドバイザーの評価テストを実行する機会があります。

W1 及び MN1 期間は「始値のみ」ティック生成モードの例外です。これらの時間軸では、ティックは週または月の OHLC 価格ではなく、毎日の OHLC 価格に生成されます。

「始値のみ」モードでEURUSD H1 でエキスパートアドバイザーをテストするとします。この場合、ティック（制御点）の合計数はテスト間隔内の1時間のバーの数 4 倍以下です。しかしOnTick() **ハンドラは 1 時間足の開始でのみ呼ばれます。**. 正しいテストのために必要なチェックは（エキスパートアドバイザーから「隠れ」ている）ティックの残りの部分で発生します。

- 証拠金要件の計算
- 決済逆指及び決済指レベルのトリガ
- 未決注文のトリガ
- 期限切れの未決注文の削除

© 2000–2025, MetaQuotes Ltd.

ポジションや未決注文がない場合、隠れたティックのチェックは無用で、速度の増加はかなり大きいです。この「始値のみ」モードは、未決注文、決済逆指及び決済指注文を使わずにバーの開始時のみで取引を処理するストラテジーのテストに適しています。そのようなストラテジーのクラスでは、テストの必要な精度が維持されます。

どのモードでのテストも可能なエキスパートアドバイザーの一例として、標準パッケージからMoving Average（移動平均）エキスパートアドバイザーを使用してみましょう。このエキスパートアドバイザーの論理は、意思決定の全てがバーのオープン時でなされるような方法で構築されており、取引は未決を使用せずにすぐに実行されます。

エキスパートアドバイザーのテストを 2010.09.01 から 2010.12.31 までの EURUSD H1 で実行し、グラフを比べてみます。図は、3 つのモードの全てのテストレポートからのバランスグラフを示します。



標準パッケージの Moving Average.mq5 エキスパートアドバイザーのグラフは異なるテストモードで全く同じです。

「始値のみ」モードには制限があります。

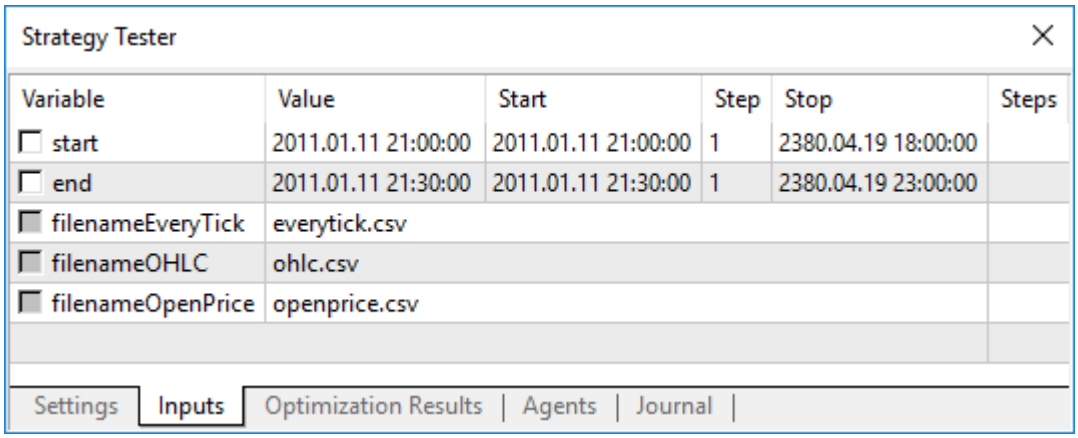
- ランダム遅延実行モードは使用出来ません。
- テストされているエキスパートアドバイザーでは、テストと最適化に使用されるより低い時間軸のデータはアクセス出来ません。例えば、H1 期間をテスト/最適化する場合、H2、H3、H4のデータにアクセスすることが出来ますが、M30、M20、M10などにはアクセス出来ません。また、アクセスされた長い時間軸は、テストの時間軸の倍数でなければなりません。例えば、M20でテストを実行する場合、M30のデータにアクセスすることは出来ません。が、H1にアクセスすることは可能です。これらの制限は、テスト/最適化中に生成されたバーのうち、下側または非多重時間軸のデータの取得が不可能なことに関係しています。
- 他の時間軸のデータへのアクセスの制限は、データエキスパートアドバイザーが使用している他のシンボルにも適用されます。この場合、各シンボルの制限は、テスト/最適化中に初めにアクセスされた時間軸によって異なります。EURUSD H1 Eのテスト中に、エキスパートアドバイザーが GBPUSD M20 のデータにアクセスすると仮定します。この場合、エキスパートアドバイザーは EURUSD H1、H2、だけでなく、GBPUSD M20、H1、H2などのデータを使用することが出来るようになります。

© 2000-2025, MetaQuotes Ltd.

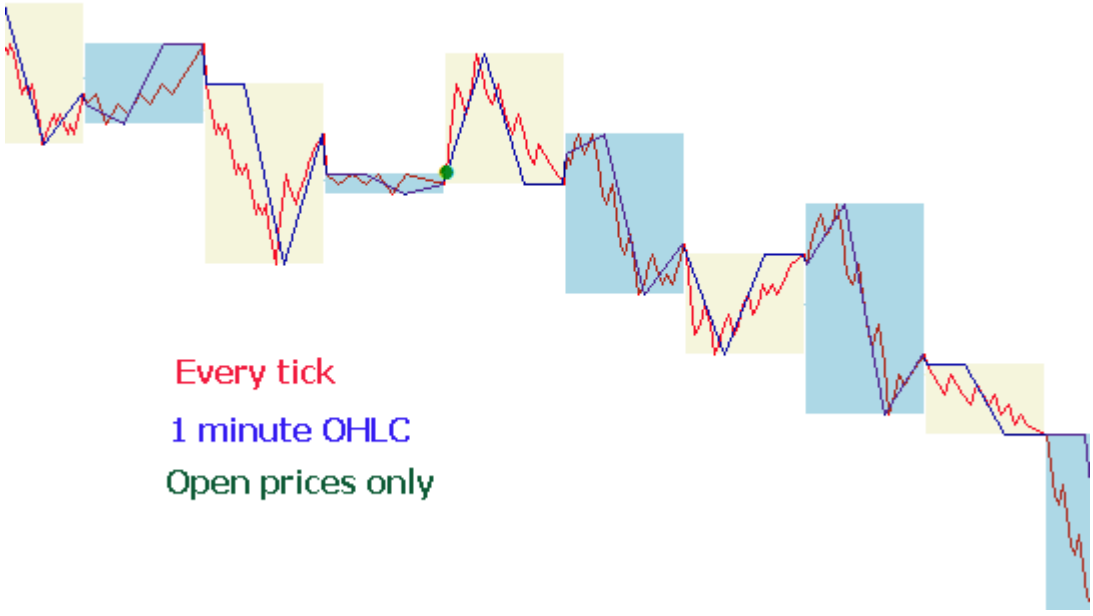
注意事項：「始値のみ」モードのテスト時間は最短ですが、このモードは取引ストラテジーの全てには適していません。取引システムの特性に基づいて、所望のテストモードを選択するべきです。

このセクションを締めくくりとして 2011.01.11 21:00:00 から 2011.01.11 21:30:00 の間隔の 2 つの M15 バーでの EURUSD での異なるティック生成モードを視覚的に比較して検討してみましょう。

ティックは WriteTicksFromTester.mq5 エキスパートアドバイザーを使用して異なるファイルに保存され、ファイル名の終わりは filenameEveryTick、filenameOHLC 及び filenameOpenPrice 入力パラメータに指定されています。



「全ティック」、「1分 OHLC」と「始値のみ」の 3 つのモードで、3ティック配列で3つのファイルを取得するために、エキスパートアドバイザーは、単一の実行で、対応するモードで 3 回起動されました。そして、この 3 つのファイルからのデータは TicksFromTester.mq5 指標を使用してチャート上に表示されました。 指標コードはこの記事に添付されています。



デフォルトでは MQL5 言語のファイル操作は「ファイルサンドボックス」内で行われ、テストではエキスパートアドバイザーはこれらの「ファイルサンドボックス」にしかアクセス出来ません。テスト中で指標とエキスパートアド

© 2000–2025, MetaQuotes Ltd.

バイザーが 1 つのフォルダのファイルと作業出来るために FILE_COMMON フラグ が使用されました。エキスパートアドバイザーからのコード例

```
//--- ファイルを開く
   file=FileOpen(filename,FILE_WRITE|FILE_CSV|FILE_COMMON,";");
//--- ファイルハンドルをチェックする
   if(file==INVALID_HANDLE)
     {
      PrintFormat("Error in opening of file %s for writing. Error code=%
d",filename,GetLastError());
      return;
     }
   else
     {
      PrintFormat("The file will be created in %s
folder",TerminalInfoString(TERMINAL_COMMONDATA_PATH));
     }
```

指標のデータを読むために FILE_COMMON フラグも使用されました。これによって、必要なファイルの別のフォルダへの手動転送を回避できました。

```
//--- ファイルを開く
   int file=FileOpen(fname,FILE_READ|FILE_CSV|FILE_COMMON,";");
//--- ファイルハンドルをチェックする
   if(file==INVALID_HANDLE)
     {
      PrintFormat("Error in open of file %s for reading. Error code=%
d",fname,GetLastError());
      return;
     }
   else
     {
      PrintFormat("File will be opened from %
s",TerminalInfoString(TERMINAL_COMMONDATA_PATH));
     }
```

スプレッドのシミュレーション

売値と買値との価格差はスプレッドと呼ばれています。テストでは、スプレッドはモデルされる代わりに履歴データからとられます。履歴データでスプレッドがゼロ以下の場合、最後の既知の(生成の瞬間での)スプレッドがテストエージェントによって使用されます。

ストラテジーテスターでは、スプレッドは常に浮動と考えられています。つまり SymbolInfoInteger(symbol, SYMBOL_SPREAD_FLOAT) は常に true を返します。

更に、履歴データはティック値と取引高を含みます。データ格納と取得のためには特別な MqlRates 構造体が使用されます。

```
struct MqlRates
  {
   datetime time;          // 期間開始時刻
   double   open;          // 始値
```

© 2000-2025, MetaQuotes Ltd.

```
   double   high;         // 期間の高値
   double   low;          // 期間の安値
   double   close;        // 終値
   long     tick_volume;  // ティックボリューム
   int      spread;       // スプレッド
   long     real_volume;  // 取引高
};
```

テストでの実際ティックの使用

実際ティックのテストと最適化は、実世界の条件をより近く再現します。分足データに基づいてティックを生成する代わりに、ブローカによって蓄積された実際のティックを使用することが可能です。これらは、取引所および流動資産プロバイダーからのティックです。

最も正確なテスト精度を保証するために、分足は実際ティックモードでも使用され、ティックデータをチェックし修正するために適用されます。これにより、テスターとクライアント端末のチャートの相違を避けることもできます。

テスターは、ティックデータと分足パラメータを比較します。ティックはバーの高/低レベルを超えてはならず、最初と最後のティックはバーの始値/終値と一致する必要があります。ボリュームも同様に比較されます。不一致が検出された場合、この分足に対応するすべてのティックは破棄され、生成ティックが代わりに使用されます(「全ティック」モード同様)。

シンボル履歴にティックデータを持たない分足が存在する場合、テスターは「全ティック」モードでティックを生成します。これにより、ブローカのティックデータが不十分な場合に、テスターに正しいチャートがプロットされます。

シンボル履歴に分足が存在しないが、その分の適切なティックデータが存在する場合、テスターはそのデータを使用します。たとえば、取引シンボルのペアは最終価格を使用して形成されたとします。最終価格が設定されていないBid /Ask価格のティックのみがサーバーから到着した場合、バーは生成されません。これらのティックデータは分データと矛盾しないので、テスターに使用されます。

ソースからクライアント端末にデータを送信する際の接続損失または他の不具合といった様々な理由により、ティックデータは分足と一致しないことがあります。テストにおいては分足データの信頼性はより高いと考えられます。

実際ティックでテストするときは、次の点にご注意ください。

- テストを開始すると、シンボルの分足データがティックデータと同期されます。
- ティックは、ストラテジーテスターのシンボル・キャッシュに格納されます。キャッシュサイズは128,000ティックを超えません。新しいティックが到着すると、最も古いデータがキャッシュから削除されます。しかし、CopyTicks機能によってキャッシュ外でティックを受け取ることが可能です(実際ティックでテストする場合のみ)。その場合、データは、対応するクライアント端末データベースと完全に類似しているテスターティックデータベースから要求されます。このベースでは分足は補正されていません。したがって、そこにあるティックは、キャッシュに格納されているティックと異なる場合があります。

クライアント端末のグローバル変数

テストではクライアント端末のグローバル変数もエミュレートされますが、これらは、端末で F3 ボタンでみられる、現在の端末のグローバル変数とは関係ありません。これは、テスト中の端末のグローバル変数の全ての操作は、クライアント端末の外で(テストエージェントで)行われることを意味します。

© 2000-2025, MetaQuotes Ltd.

テスト中の指標の計算

リアルタイムモードでは、全てのティックで指標値が計算されます。

ストラテジーテスターでは、指標はデータにアクセスされたとき、つまり指標バッファ値が要求されたときにのみ計算されます。唯一の例外は、#property tester_everytick_calculateが指定されたカスタム指標です。この場合、再計算はティックごとに行われます。

ビジュアルテストモードでは、ビジュアルテストチャートに正しく表示されるように、新しいティックが到着したときにすべての指標が無条件に再計算されます。

指標はティックごとに1回計算されます。新しいティックが到着するまでは、指標データに対する後続の要求は再計算にはつながりません。したがって、EventSetTimer()関数を介してEAでタイマーが有効になっている場合、指標データはOnTimer()ハンドラの各呼び出しの前の最後のティックから要求されます。最後のティックで指標がまだ計算されていない場合は、指標値の計算が開始されます。データがすでに準備されている場合は、新しい再計算なしで提供されます。

したがって、すべての指標の計算は、最もリソースを節約する方法で実行されます。指標が特定のティックですでに計算されている場合、そのデータは「そのまま」提供されます。再計算は開始されません。

テスト中の履歴の読み込み

テストされるシンボルの履歴は、テストプロセスの開始前に端末によって取引サーバから読み込みまた同期されます。端末は、後のリクエストを避けるために、初めに使用可能な全てのシンボル履歴を順番に読み込みます。その後は、新しいデータのみが読み込まれます。

テストエージェントは、テスト開始の直後に、テストされるシンボルの履歴を端末から受け取ります。他の金融製品のデータがテストの過程で使用されている(例えば、多通貨エキスパートアドバイザー)場合、テストエージェントは、データの最初の呼び出し時にクライアント端末から必要な履歴を要求します。履歴データが端末で利用可能な場合、それらはすぐにテストエージェントに渡されます。データが利用可能でない場合は、端末はサーバからリクエスト及びダウンロードしてからテストエージェントに渡します。

取引操作のクロスレートを計算するには、追加的な製品のデータも必要です。例えば、米ドル預金通貨での EURCHF ストラテジーをテストする際、ストラテジーにはそれらを直接使用する呼び出しが含まれないにもかかわらず、テストエージェントは、最初の取引操作を処理する前に、クライアント端末から EURUSD と USDCHF の履歴データをリクエストします。

多通貨ストラテジーをテストする前に、クライアント端末に全ての必要な履歴データをダウンロードすることをお勧めします。これによって、必要なデータのダウンロードに関連したテスト/最適化の遅れが防げます。例えば、適切なチャートを開いて履歴先頭にスクロールすることによって、履歴をダウンロードすることが出来ます。端末への履歴の強制読み込みの例は MQL5 リファレンスのデータアクセスの整理セクションでみられます。

代わりに、テストエージェントは端末からパックされた形式で履歴を受け取ります。必要なデータはテスタの前回の実行以降利用可能であるため、次のテスト中にはテスタは端末から履歴を読み込みません。

- 端末は、エージェントが初めにテストされるシンボルの履歴をリクエストする際に、取引サーバから一度だけ履歴を読み込みます。トラフィックを減らすために、履歴はパック形式で読み込まれます。
- ティックはネットワーク経由で送信されず、テストエージェント上で生成されます。

© 2000-2025, MetaQuotes Ltd.

多通貨テスト

ストラテジーテスターは、複数シンボルの売買、ストラテジーのテストを実行することが出来ます。　以前のプラットフォームではテストは単一のシンボルに対して実施されていたので、このようなエキスパートアドバイザーは従来「多通貨エキスパートアドバイザー」と呼ばれます。MetaTrader 5 端末のストラテジーテスターでは、利用可能な全てのシンボルの取引をモデル化することが出来ます。

テスタはシンボルデータの最初の呼び出し時に自動的に使用される記号の履歴を（取引サーバからではなく）**クライアント端末** から読み込みます。

テストエージェントはテスト開始時に指標の計算に必要で欠けている小幅な履歴データをダウンロードします。D1 以下の時間軸では、ダウンロードされた履歴の最小容量は1年です。

よって、2010.11.01-2010.12.01 の間隔を（各足が 15 分に等しい）M15 期間でテストした場合、端末は2010年 1 年間の履歴をリクエストします。週足では、約 2 年（1年は 52 週）である 100 足の履歴を要求します。月足でのテストのために、エージェントは 8 年の履歴（12 カ月 ×8 年 = 96 ヶ月）を要求します。

足が足りない場合、必要な足数を確保するためにテスト前にテストの開始日が過去から現在まで自動的にシフトされます。

テスト中には「気配値表示」もエミュレートされ 銘柄情報が取得出来ます。

デフォルトでは、テスト開始時にストラテジーテスターの「気配値表示」にはテストが実行されている 1 つのみの銘柄が存在します　。必要なシンボルの全ては自動的に（端末ではなく）ストラテジーテスターの「気配値表示」に参照されます。

> 多通貨エキスパートアドバイザーのテストを開始する前に、端末の「気配値表示」のテストに必要なシンボルを選択して必要なデータを読み込む必要があります。「外来」シンボルの最初の呼び出し時には、その履歴が自動的にテストエージェントとクライアント端末間で同期されます。「外来」シンボルとはテストが実行されているシンボル以外のシンボルです。

「他の」シンボルへの参照は次の場合に起こります。

- 銘柄/時間軸でのテクニカル指標関数及び IndicatorCreate() の使用時
- 「気配値表示」の他のシンボルデータのリクエスト
  1. SeriesInfoInteger
  2. Bars
  3. SymbolSelect
  4. SymbolIsSynchronized
  5. SymbolInfoDouble
  6. SymbolInfoInteger
  7. SymbolInfoString
  8. SymbolInfoTick
  9. SymbolInfoSessionQuote
  10. SymbolInfoSessionTrade
  11. MarketBookAdd
  12. MarketBookGet

© 2000–2025, MetaQuotes Ltd.

- 以下の関数を使用した銘柄/時間軸のための時系列のリクエスト
  1. CopyBuffer
  2. CopyRates
  3. CopyTime
  4. CopyOpen
  5. CopyHigh
  6. CopyLow
  7. CopyClose
  8. CopyTickVolume
  9. CopyRealVolume
  10. CopySpread

「その他」の銘柄の最初の呼び出しの時点で、テストプロセスが停止され、銘柄/時間軸の履歴が端末からテストエージェントにダウンロードされます。同時に、このシンボルのティックシーケンスが生成されます。

シンボルごとに個々のティックシーケンスが、選択されたティック生成モードに応じて生成されます。OnInit() ハンドラの SymbolSelect() の呼び出しによって、目的のシンボルの履歴を明示的にリクエストすることも出来ます。履歴ダウンロードはエキスパートアドバイザーテストの直前に行われます。

この様に MetaTrader 5 クライアント端末での多通貨テストの実行は余分な労力を必要としません。ただ、クライアント端末で適切な銘柄のチャートを開くだけです。適切なデータが含まれていれば全ての必要なシンボルの履歴は自動的に取引サーバからアップロードされます。

ストラテジーテスターでの時間のシミュレーション

テスト中、TimeLocal() ローカル時刻は常に TimeTradeServer() サーバ時刻と同じです。そして、サーバ時刻は常に TimeGMT() GMT に等しいです。こうすれば、これらの全ての関数は、テスト中に同じ時刻を表示します。

GMT、ローカル、及びストラテジーテスターでのサーバ時刻との差の欠如は、サーバへの接続がない場合には、意図的に行われます。テスト結果は接続のあるなしに関係なく同じでなければなりません。サーバ時刻に関する情報はローカル格納されておらず、サーバから取得されます。

テストでのグラフィックオブジェクト

テスト/最適化中には、グラフィックオブジェクトはプロットされません。よって、テスト/最適化の際に作成されたオブジェクトのプロパティを参照すると、エキスパートアドバイザーはゼロ値を受け取ることになります。

> この制限は、ビジュアルモードでのテストには適用されません。

ストラテジーテスターでの OnTimer() 関数

MQL5 はタイマーイベントを処理するための機会を提供します。OnTimer() ハンドラはテストモードにかかわらず呼び出されます。これは、テストが H4 期間で「始値のみ」モードを実行し、エキスパートアドバイザーが毎秒コールに設定されたタイマを有する場合、それぞれの H4 足の開始時に OnTick() ハンドラが1回呼ばれ、その後 OnTimer() ハンドラが 14,400 回（3,600 秒 ＊4 時間）呼ばれることを意味します。エキスパートアドバイザーのテスト時間が増加される量は、その論理に依存します。

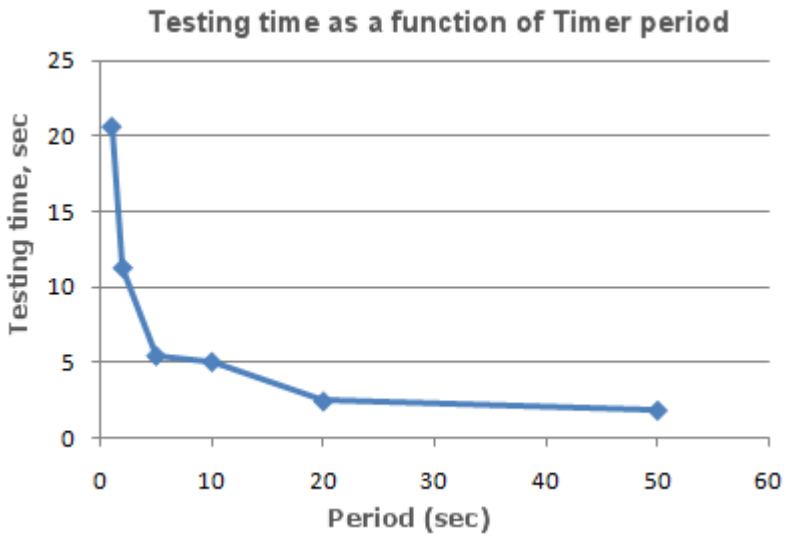
© 2000–2025, MetaQuotes Ltd.

テスト時間のタイマーに所定された頻繁度依への存性を確認するために、取引操作なしの簡単なエキスパートアドバイザーを作成しました。

```
//--- 入力パラメータ
input int      timer=1;               // 秒単位でのタイマー値
input bool     timer_switch_on=true; // タイマーがオン
//+------------------------------------------------------------------+
//| エキスパート初期化に使用される関数                                |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- timer_switch_on==true  の場合タイマーを実行
   if(timer_switch_on)
     {
      EventSetTimer(timer);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| エキスパート初期化解除に使用される関数                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- タイマーを停止する
   EventKillTimer();
  }
//+------------------------------------------------------------------+
//| Timer 関数                                                       |
//+------------------------------------------------------------------+
void OnTimer()
  {
//---
// ハンドラ本体が空なので何もしない
  }
//+------------------------------------------------------------------+
```

タイマパラメータ(タイマーイベントの時間軸)の異なる値でテスト時間の測定が取られました。取得されたデータを使用して、テスト時間をタイマー時間軸の関数としてプロットします。

© 2000-2025, MetaQuotes Ltd.



timer パラメータが小さいほど、EventSetTimer(Timer) 関数の初期化中の OnTimer() ハンドラの呼び出しの間隔が小さく、テスト時間 T が大きいことが明確です。

ストラテジーテスターでの Sleep() 関数

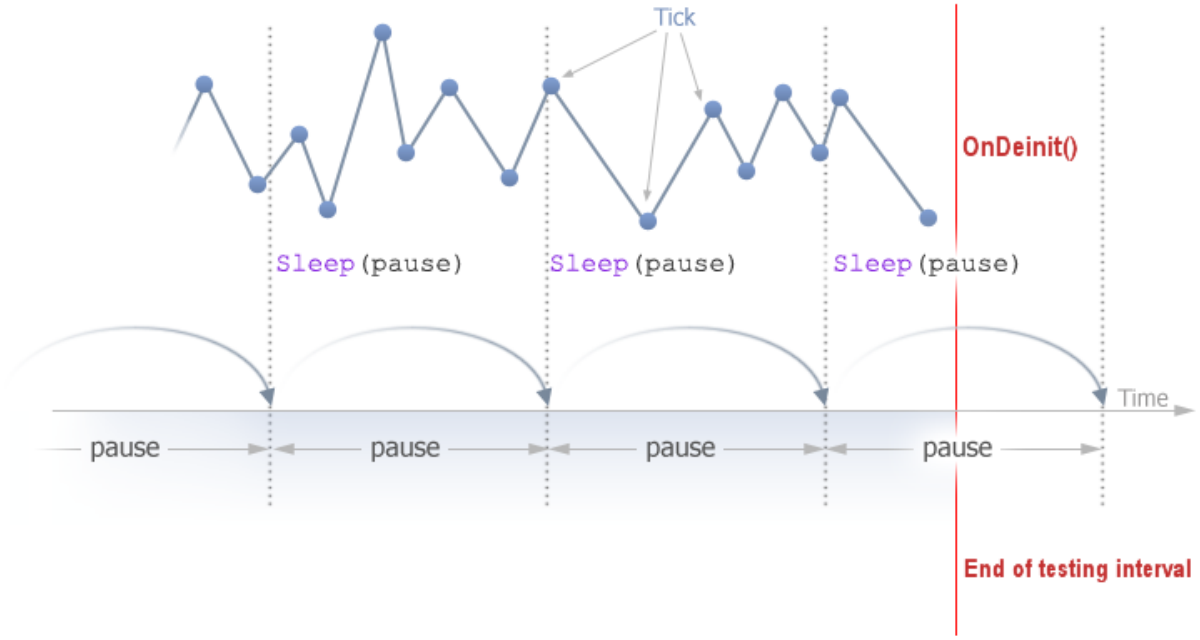
Sleep() 関数は、グラフを操作する場合、エキスパートアドバイザーまたはスクリプトがしばらくの間 MQL5 プログラムの実行を中断することを可能にします。これは、リクエスト時にデータが準備されてなく準備が出来るまで待つ必要がある場合に便利です。Sleep() 関数の繊細な使用例は データーアクセスの整理 セクションでみられます。

テストプロセスは Sleep() の呼び出しで滞ることはありません。Sleep() が呼び出されると生成されたティックは指定された遅延以内に「プレイ」され、未決注文やストップのトリガになるかもしれません。Sleep() の呼び出し後に、ストラテジーテスターでのシミュレート時刻は Sleep 関数のパラメータで指定された時間分進みます。

Sleep() 関数の実行結果として、ストラテジーテスターで現在時刻がテスト期間を超えてしまった場合は、「Infinite Sleep loop detected while testing（テスト中に無限スリープループが検出されました）」とのエラーを受け取ることになります。このエラーが受け取られた場合、テスト結果は拒否されず、全ての計算は全体（取引数、沈下など）にわたって行われ、テスト結果は端末に渡されます。

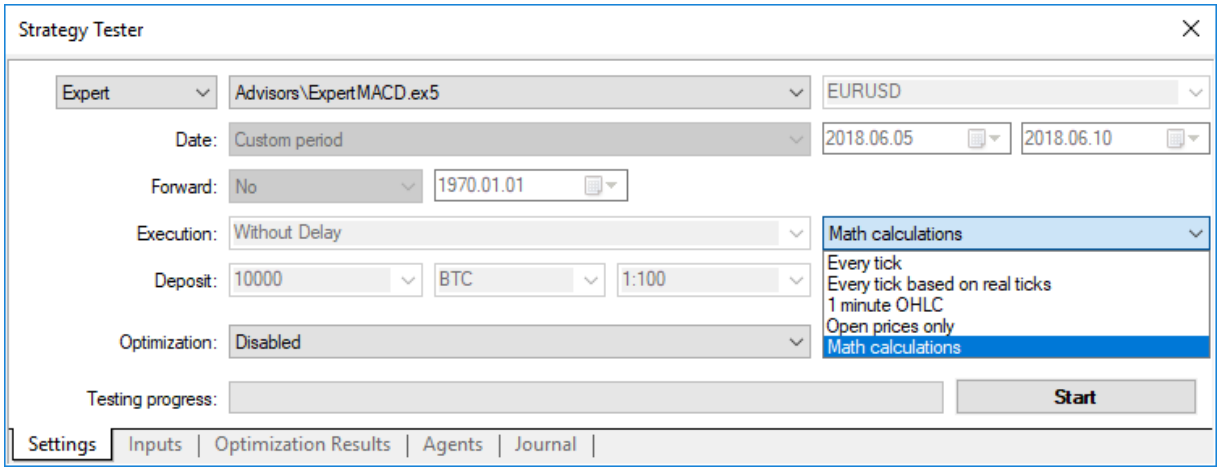
OnDeinit() の呼び出しは、テスト期間の範囲を超えることが保証されるため、Sleep() 関数は OnDeinit() では動作しません。

© 2000–2025, MetaQuotes Ltd.



数値計算や最適化問題のためのストラテジーテスター――の使用

MetaTrader 5 端末のテスタは、取引ストラテジーテストにだけでなく数学的な計算のためにも使用することが出来ます。使用するには「数値計算」モードの選択が必要です。



この場合、OnInit()、OnTester()、OnDeinit() の 3 つの関数のみが呼ばれます。「数値計算」モードでは、ストラテジーテスターは、いかなるティックも生成せず履歴もダウンロードしません。

終了日より後日の開始日を指定された場合も、ストラテジーテスター「数値計算」モードで動作されます。

数学問題の解決にテスタを使用する場合は、履歴のアップロードとティックの生成は起こりません。

多変数関数の極値を探すことは MetaTrader 5 ストラテジーテスターで解決する典型的な数学の問題です。

解決するには

- 関数値の計算は OnTester() 関数に位置するべきです。
- 関数のパラメータはエキスパートアドバイザーの入力変数として定義されるべきです。

© 2000–2025, MetaQuotes Ltd.

エキスパートアドバイザーをコンパイルして「ストラテジーテスター」ウィンドウを開きます。「パラメータ」タブで必要な入力パラメータを選択し、各関数変数のスタート、ストップ、及びステップ値を指定してパラメータセットの値を定義します。

「設定」タブの「オプティマイズ」(最適化の種類)で、「完全アルゴリズム(低速)」または「遺伝的アルゴリズム(高速)」を選びます。関数の極値の簡単な検索のためには、高速な最適化を選択した方が良いですが、変数のセット全体の値を計算する場合は、時間をかけた最適化を使用するのが最適です。

「数値計算」モードを選び「スタート」ボタンを使用して最適化の実行をします。ストラテジーテスターは最適化中に OnTester 関数の極大値を検索することにご注目ください。極小値を見つけるには OnTester 関数で計算された関数値の逆数を返します。

```
return(1/function_value);
```

ゼロ除算による重大なエラーの取得を回避するために function_value がゼロでないことを確認する必要があります。

この記事の読者によって示唆された、より便利で最適化の結果を歪めない別方法があります。

```
return(-function_value);
```

このオプションは function_value がゼロに等しいかどうかの確認を必要とせず、3次元表現における最適化の結果の面は同じ図形を有します。違いは、これがオリジナルのミラーリングということです。

例として sink() 関数を提供します。

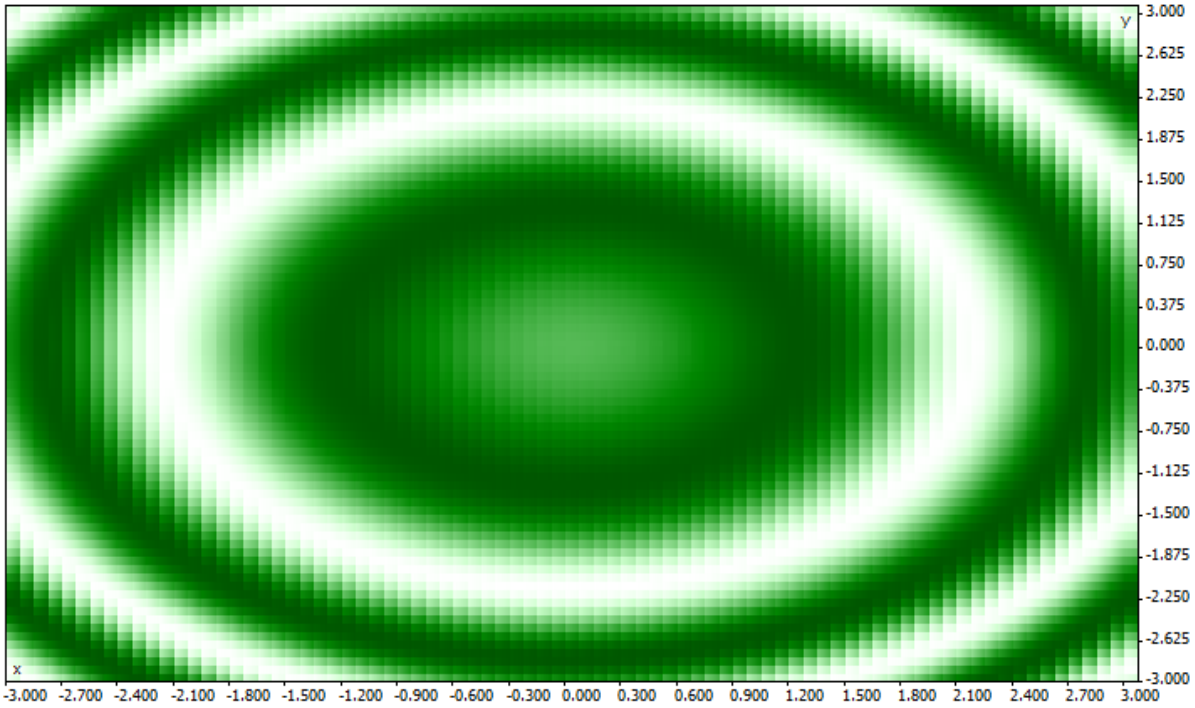
$$sink(x,y) = \sin(x^2 + y^2)$$

極値を見つけるためのエキスパートアドバイザーのコードは OnTester() に配置されます。

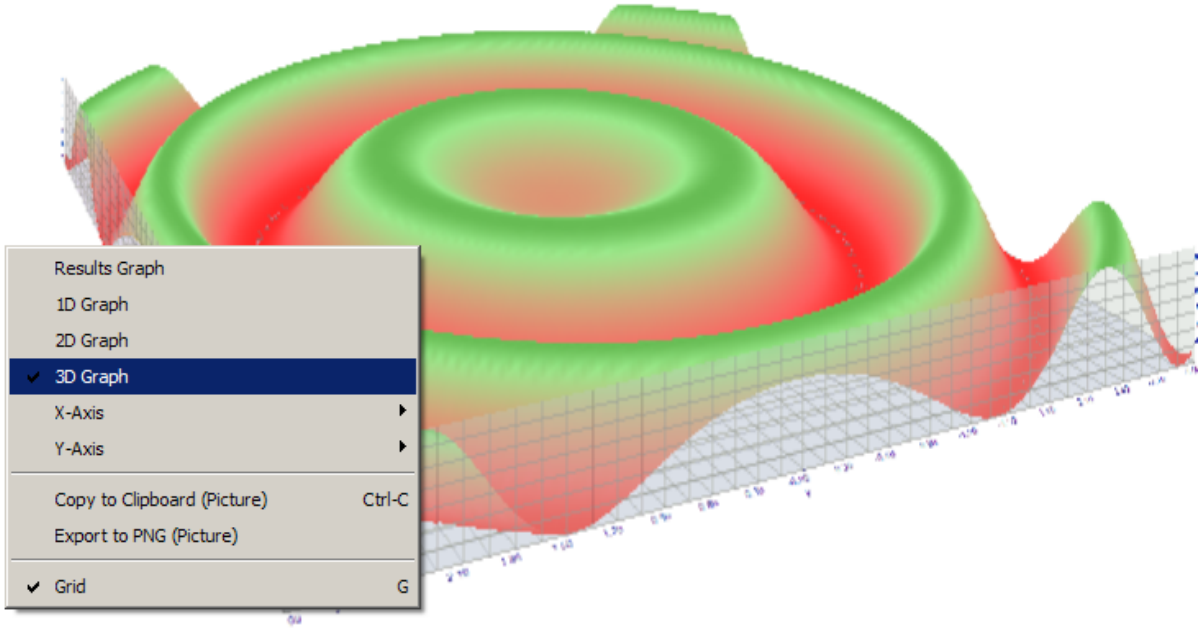
```
//+------------------------------------------------------------------+
//|                                                         Sink.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.MQL5.com"
#property version   "1.00"
//--- 入力パラメータ
input double   x=-3.0; // スタート= -3、ステップ = 0.05、ストップ = 3
input double   y=-3.0; // スタート = -3、ステップ = 0.05、ストップ = 3
//+------------------------------------------------------------------+
//| テスタ関数                                                       |
//+------------------------------------------------------------------+
double OnTester()
  {
//---
   double sink=MathSin(x*x+y*y);
//---
   return(sink);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

最適化をして、最適化の結果 を 2D グラフで見ましょう。



より良い値が与えられた(x、y)パラメータの組はより飽和した色を持ちます。sink() 式の形の観点から期待されたように、その値は(0, 0)を中心とする同心円を形成しています。関数は、単一の最大値・最小値を持っていないことは 3D グラフで見ることが出来ます。



「始値のみ」モードでのバーの同期

MetaTrader 5 クライエント端末のテスタは、いわゆる「多通貨」エキスパートアドバイザーをチェックすることが出来ます。多通貨エキスパートアドバイザーとは 2 つ以上のシンボルで売買をするエキスパートアドバイザーです。

© 2000–2025, MetaQuotes Ltd.

複数のシンボルを売買するストラテジーのテストは、テスタにいくつかの追加的な技術要件を課しています。

- シンボルのティック生成
- シンボル指標値の計算
- シンボルの証拠金要件の計算
- 全ての取引シンボルの生成されたティックシーケンスの同期

ストラテジーテスターは、ティックを生成し選択された取引モードに応じて各金融製品のためにティックシーケンスを再生します。同時に、他のシンボルにどのようにバーが開いたかに関係なく、それぞれのシンボルに新しいバーが開かれます。これは、多通貨エキスパートアドバイザーのテスト時に、ある製品では新しい足が既に開かれていても他の製品では開かれていないという状態が発生する可能性がある（実際によくあります）ことを意味します。よって、テストでも現実のように事が起こります。

テスタでのこの本格的な履歴シミュレーションは「全ティック」及び「1分足 OHLC」テストモードが使用されている限り問題を発生することはありません。これらのモードでは、1 つのローソク足に充分なティックが生成され、異なるシンボルのバーの同期まで待つことが出来ます。しかし、金融製品の取引でのバーの同期が必須である場合、多通貨ストラテジーは「始値のみ」モードではどのようにテストされるのでしょうか。このモードでは、エキスパートアドバイザーはバーのオープン時に対応する 1 つのティックのみで呼ばれています。

例を使用して説明します。EURUSD のエキスパートアドバイザーがテストされていて、1 時間足が EURUSD で開いたとします。この事実は簡単に認識出来ます。「始値のみ」モードでは NewTick イベントはテスト期間のバーのオープン時に対応します。しかし、エキスパートアドバイザーで使用されている USDJPY 上で開いたローソク足について保証するものではありません。

通常の状況では OnTick() 関数の操作を完了して次のティックでの新しい USDJPY バーの出現を確認するので充分です。しかし「始値のみ」モードでのテストでは他のティックが存在しないので、このモードは多通貨エキスパートアドバイザーのテストには向いていないように見えるかもしれません。しかし、これはそうではありません。MetaTrader 5 のテスタは現実そっくりに動作することを忘れてはいけません。Sleep() を使用して新しいバーが別のシンボルに開かれるまで待てばいいのです。

「始値のみ」モードでのバー同期化を示すエキスパートアドバイザーの Synchronize_Bars_Use_Sleep.mq5 のコード

```
//+------------------------------------------------------------------+
//|                                  Synchronize_Bars_Use_Sleep.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.MQL5.com"
#property version   "1.00"
//--- 入力パラメータ
input string   other_symbol="USDJPY";
//+------------------------------------------------------------------+
//| エキスパート初期化に使用される関数                                  |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- シンボルをチェックする
   if(_Symbol==other_symbol)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("You have to specify the other symbol in input parameters or
select other symbol in Strategy Tester!");
      //--- 強制的なテスト終了
      return(INIT_PARAMETERS_INCORRECT);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| エキスパートティック関数                                         |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- 最後のバーの時刻の格納に使用される静的変数
   static datetime last_bar_time=0;
//--- 同期化フラグ
   static bool synchonized=false;
//--- 静的変数が初期化されていない場合
   if(last_bar_time==0)
     {
      //--- 始めての呼び出し。バーを保存して終了する。
      last_bar_time=(datetime)
SeriesInfoInteger(_Symbol,Period(),SERIES_LASTBAR_DATE);
      PrintFormat("The last_bar_time variable is initialized with value %
s",TimeToString(last_bar_time));
     }
//--- チャートシンボルの最後のバーの開始時刻を取得する
   datetime curr_time=(datetime)
SeriesInfoInteger(Symbol(),Period(),SERIES_LASTBAR_DATE);
//--- 時刻が異なる場合
   if(curr_time!=last_bar_time)
     {
      //--- バーの開始時刻を静的変数に保存する
      last_bar_time=curr_time;
      //--- 同期されていない
      synchonized=false;
      //--- メッセージを出力する
      PrintFormat("A new bar has appeared on symbol %s at %
s",_Symbol,TimeToString(TimeCurrent()));
     }
//--- 他のシンボルのバーの開始時刻
   datetime other_time;
//---他のシンボルの開始時刻が curr_time になるまで反復する
   while(!(curr_time==(other_time=(datetime)
SeriesInfoInteger(other_symbol,Period(),SERIES_LASTBAR_DATE)) && !synchonized))
     {
      PrintFormat("Waiting 5 seconds..");
      //--- 5 秒待ってSeriesInfoInteger(other_symbol,Period(),SERIES_LASTBAR_DATE) を呼
び出す
      Sleep(5000);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- バーは同期されている
   synchonized=true;
   PrintFormat("Open bar time of the chart symbol %s: is %
s",_Symbol,TimeToString(last_bar_time));
   PrintFormat("Open bar time of the symbol %s: is %
s",other_symbol,TimeToString(other_time));
//--- TimeCurrent() は役立たないので TimeTradeServer() を使用する
   Print("The bars are synchronized at
",TimeToString(TimeTradeServer(),TIME_SECONDS));
  }
//+------------------------------------------------------------------+
```

同期の事実が確立された時、現在の時刻を表示するエキスパートアドバイザーの最後の行にご注目ください。

```
   Print("The bars synchronized at ",TimeToString(TimeTradeServer(),TIME_SECONDS));
```

現在時刻の表示には TimeCurrent() の代わりに TimeTradeServer() 関数を使用しました。TimeCurrent() 関数は Sleep() の使用後に変更されない最後のティックの時刻を返します。「始値のみ」モードでエキスパートアドバイザーを実行すると、バーの同期に関するメッセージが表示されます。

Core 1	2010.12.01 20:00:05	The bars are synchronized at 2010.12.01 20:00:05
Core 1	2010.12.01 20:00:05	Open bar time of the chart symbol USDJPY: 2010.12.01 20:00
Core 1	2010.12.01 20:00:05	A new bar has appeared on symbol EURUSD: 2010.12.01 20:00
Core 1	2010.12.01 20:00:00	Waiting 5 seconds...
Core 1	2010.12.01 20:00:05	A new bar has appeared on symbol EURUSD: 2010.12.01 20:00
Core 1	2010.12.01 16:00:05	The bars are synchronized at 2010.12.01 16:00:05

最後ティック到着時刻ではなく現在のサーバ時刻を取得する必要がある場合 TimeCurrent() の代わりに TimeTradeServer() 関数を使用します。

バーを同期させるためには、タイマーを使用した別方法があります。そのようなエキスパートアドバイザーの例はこの記事に添付されている Synchronize_Bars_Use_OnTimer.mq5 です。

テスタの IndicatorRelease() 関数

単一のテストの完了後、機器のチャートが自動的に開かれ、完成した約定とエキスパートアドバイザーで使用される指標を表示します。これは、視覚的にエントリーポイントと終了ポイントをチェックして指標の値と比較するのに役立ちます。

> **注意事項：** テスト終了時に自動に開くチャートに表示された指標は、テスト終了時に新たに計算されます。これらの指標がテストされたエキスパートアドバイザーで使用されていた場合にもです。

しかし、場合によって、プログラマが指標が取引アルゴリズムに関与された場合の情報を非表示にすることがあります。例えば、エキスパートアドバイザーのコードが、ソースコードなしの実行可能ファイルとしてレンタルまたは販売されている場合です。この場合 IndicatorRelease() 関数が適しています。

端末は、クライアント端末のディレクトリ/プロファイル/テンプレートでテンプレートの名称を tester.tpl と設定した場合、それが開かれたチャートに適用されます。それがない場合、デフォルトのテンプレートが適用されます。(default.tpl).

© 2000-2025, MetaQuotes Ltd.

IndicatorRelease() 関数はもともと、不要になった場合指標の演算部を解放するのに意図されています。各ティックが指標計算を呼ぶので、メモリとCPUリソースの両方を保存することが出来ます。その2つ目の目的は、単一のテストの実行後、テストチャートでの指標の表示を禁止することです。

テスト後に指標の表示を禁止するには IndicatorRelease() 関数を OnDeinit()ハンドラの指標ハンドルと呼びます。OnDeinit() 関数は、常にテスト完了後及びチャート表示前に呼び出されます。

```
//+------------------------------------------------------------------+
//|  エキスパート初期化解除に使用される関数                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   bool hidden=IndicatorRelease(handle_ind);
   if(hidden) Print("IndicatorRelease() successfully completed");
   else Print("IndicatorRelease() returned false. Error code ",GetLastError());
  }
```

テスト後に指標の表示を禁止するには OnDeinit() ハンドラで IndicatorRelease() を呼びます。

テスタでのイベントハンドル

MetaTrader 5 テスタで履歴データのテストをするにはエキスパートアドバイザーでの OnTick() ハンドラの存在は必須ではありません。エキスパートアドバイザーが下記の関数ハンドラの 1 つを持つことで充分です。

- OnTick() – 新しいティック到着のイベントハンドラ
- OnTrade() – 取引イベントハンドラ
- OnTimer() – タイマからの信号到達のイベントハンドラ
- OnChartEvent() – クライアントイベントハンドラ

エキスパートアドバイザーのテストでは、カスタムイベントは OnChartEvent() 関数で処理出来ますが、指標では、この関数はテスタで呼ぶことが出来ません。指標に OnChartEvent() イベントハンドラがあって指標がテストされたエキスパートアドバイザーに使用されていたとしても、指標自体はカスタムイベントを受け取りません。

テスト時に、指標は EventChartCustom() 関数を使用してカスタムイベントを作成することができ、エキスパートアドバイザーは個のイベントをOnChartEvent() で処理出来ます。

これらのイベントに加えて、ストラテジーテスターではテストと最適化のプロセスに関連する特別なイベントが生成されます。

- Tester – このイベントはエキスパートアドバイザーの履歴データのテストの完了時に生成されます。Tester イベントは OnTester() 関数で処理されます。この関数は、エキスパートアドバイザーのテスト時にのみ使用でき、主に入力パラメータの遺伝的最適化のためのカスタム最大基準として使用される値の計算のために意図されています。
- TesterInit – このイベントは、ストラテジーテスターでの最適化の一番初めのパスの開始時に発生します。TesterInit イベントは OnTesterInit() 関数で処理されます。最適化の開始時に、このハンドラを持つエキスパートアドバイザーは、テスタに指定されたシンボルと時間軸を持つ別の端末チャートに自動的に読み込まれ、TesterInit イベントを受け取ります。この関数は、更なる最適化結果の処理のために、最適化の開始前にエキスパートアドバイザーを開始するために使用されます。

© 2000–2025, MetaQuotes Ltd.

- TesterPass – このイベントは新しいデータ枠が受信された時に生成されます。TesterPass イベントは OnTesterPass() 関数で処理されます。このハンドラを持つエキスパートアドバイザーは、テストのために指定された銘柄/時間軸で別の端末のチャートに自動的に読み込まれ、最適化の間にフレームが受信された時にTesterPass イベントを受け取ります。この関数は、完了を待たずの「その場で」の最適化の結果の動的処理に使用されます。フレームは OnTester() ハンドラの単一パスの後で呼ばれる FrameAdd() 関数で追加されます。
- TesterDeinit – このイベントは、ストラテジーテスターにおけるエキスパートアドバイザーの最適化の終了後に生成されます。TesterDeinit イベントは OnTesterDeinit() 関数で処理されます。このハンドラを持つエキスパートアドバイザーは、最適化の開始時に自動的にチャートに読み込まれ、終了後にTesterDeinit を受け取ります。この関数は最適化の結果の全ての最終処理に使用されます。

テストエージェント

MetaTrader 5 クライアント端末でのテストはテストエージェントを使用して行われます。ローカルエージェントは自動的に作成されて有効にされます。ローカルオブジェクトのデフォルト数は、コンピュータ内のコアの数に対応しています。

それぞれのテストエージェントはクライアント端末と関係なく独自のグローバル変数のコピーを持ちます。端末自体は、ローカル及び遠隔エージェントにタスクを分散するディスパッチャです。エキスパートアドバイザーのテストの作業を実行した後、エージェントは指定されたパラメータで端末に結果を返します。単一のテストには、単一のエージェントが使用されます。

端末から受信された履歴はエージェントによって製品の名称を使用した別々のフォルダに保存されるので、EURUSD の履歴はEURUSD という名称のフォルダに保存されます。また、金融製品の履歴は、そのソースによって分類されています。履歴は以下のように保存されます。

```
tester_catalog\Agent-IPaddress-Port\bases\name_source\history\symbol_name
```

例えば MetaQuotes-Demo サーバからの EURUSD の履歴は tester_catalog¥Agent-127.0.0.1-3000¥bases¥MetaQuotes-Demo¥EURUSD fフォルダに保存されます。

ローカルエージェントは、テストの完了後に次の呼び出しのために立ち上げに時間を無駄にしないように、次のタスクを待って5分間スタンバイモードに入ります。待機期間が終わって初めてローカルエージェントはシャットダウンし CPU メモリからアンロードされます。

ユーザがテストを早期終了(「キャンセル」ボタン)した場合やクライアント端末が閉じられた場合、全てのローカルエージェントはすぐに作業を停止しメモリからアンロードされます。

端末とエージェントの間でのデータ交換

テスト実行時に、クライアント端末はエージェントに送信するパラメータブロックをいくつか用意します。

- テストの入力パラメータ(シミュレーションモード、テスト、金融製品、最適化基準等の間隔など)
- 「気配値表示」で選択されたシンボルのリスト
- シンボルテストの仕様(契約サイズ、決済逆指値及び決済指値を設定するための市場からの許容補償金)
- テストされるエキスパートアドバイザーと入力パラメータの値
- 追加的なファイルの情報(ライブラリ、指標、データファイル – # property tester_ ...)

© 2000–2025, MetaQuotes Ltd.

tester_indicator	string	「indicator_name.ex5」形式でのカスタム指標の名称テストが必要な指標は対応するパラメータが定数文字列で設定されている場合 iCustom() 関数の呼び出しで自動的に定義されます。それ以外は全て(IndicatorCreate() 関数の使用または指標銘を設定するパラメータでの非定数文字列の使用)このプロパティは必須です。
tester_file	string	テスタのファイル名。二重引用符内で拡張子の表示も含みます(文字列定数)。指定されたファイルはテスタに渡されます。テストされるべき入力ファイルがある場合、指定が必ず必要です。
tester_library	string	二重引用符内で拡張子を含むライブラリ名。ライブラリは拡張子として dll または ex5 を持つことが出来ます。テストを必要とするライブラリは自動的に定義されます。しかし、ライブラリがカスタム指標で使用された場合、このプロパティは必須です。

パラメータの各ブロックで、MD5 ハッシュ形式のデジタル指紋が作成されてエージェントに送信されます。MD5 ハッシュは、基づいて計算された情報の量よりもはるかに小さく、セットごとに一意です。

エージェントは、ブロックのハッシュを受信し、既に持っているものと比較します。与えられたパラメータブロックの指紋がエージェントに存在しない、または受信したハッシュが、既存のものと異なる場合、エージェントは、このパラメータのブロックを要求します。これによって、端末とエージェント間のトラフィックが減少します。

テスト後にエージェントは「テスト結果」と「最適化の結果」タブに表示される、受信利益、約定数、シャープ係数、OnTester() 関数の結果などの実行結果をプラットフォームに返します。

最適化中に、端末はエージェントにテスト作業を小型パッケージで渡し、それぞれのパッケージにはいくつかのタスクが含まれています(ここで、それぞれのタスクとは、入力パラメータのセットを持つ単一のテストを意味します。)。これは、端末とエージェント間の交換時間を短縮します。

エージェントは、セキュリティ上の理由のため、端末から取得した EX5 ファイル(エキスパートアドバイザー、指標、ライブラリなど)をハードディスクに記録することはありません。実行中のエージェントを持つコンピュータが送信したデータを使用することが出来ないようにです。DLL を含んだ他の全てのファイルはサンドボックスに記録されます。遠隔エージェントではDLL を使用してのエキスパートアドバイザーのテストは出来ません。

テスト結果は、必要な時に迅速にアクセス出来るように、端末によって結果の特別なキャッシュ(結果キャッシュ)に追加されます。再実行を回避するために、端末はパラメータのセットごとに以前の実行からの既に入手可能な結果を結果キャッシュ内で検索します。この様なパラメータセットで結果が見つからない場合、エージェントにテスト実行タスクが与えられます。<

端末とエージェントの間のトラフィックは全て暗号化されます。

> ティックはネットワーク経由で送信されず、テストエージェント上で生成されます。

全てのクライアント端末の共有フォルダの使用

全てのテストエージェントは、互いから、またクライアント端末から単離されています。各エージェントは、ログが記録されている独自のフォルダを持っています。また、テスト中のエージェントのファイル操作の全ては **agent_name/MQL5/Files** フォルダで行われます。しかし、ファイルのオープン時に FILE_COMMON フラグを指定された場合、クライアント端末の全てに共有フォルダを介してローカルエージェントとクライアント端末間の相互作用を実現することが出来ます。

```
//+------------------------------------------------------------------+
```

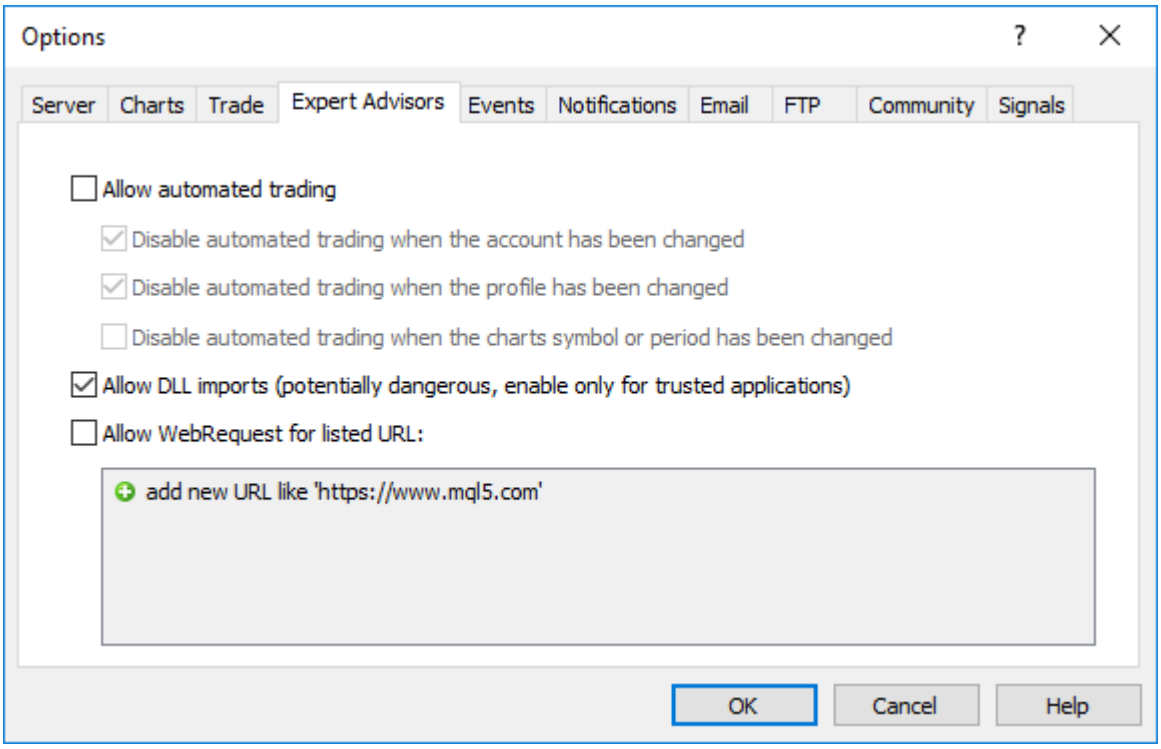
© 2000-2025, MetaQuotes Ltd.

```
//|  エキスパート初期化に使用される関数                                  |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 全てのクライアント端末の共有フォルダの使用
   common_folder=TerminalInfoString(TERMINAL_COMMONDATA_PATH);
//--- このフォルダの名称を書く
   PrintFormat("Open the file in the shared folder of the client terminals %s",
common_folder);
//--- ファイルを共通フォルダで開く( FILE_COMMON フラッグで示される)
   handle=FileOpen(filename,FILE_WRITE|FILE_READ|FILE_COMMON);
  ... 更なるアクション
//---
   return(INIT_SUCCEEDED);
  }
```

DLL の使用

最適化を速めるために、ローカルだけではなく遠隔エージェントも利用出来ます。この場合、遠隔エージェントにはいくつかの制限があります。まず第一に、遠隔エージェントはログにポジションの開閉に関する Print() 関数実行の結果を表示しません。間違って書かれたエキスパートアドバイザーが、遠隔エージェントに使用されているコンピューターをメッセージで破壊しないように、ログには最小限の情報が表示されます。

2 番目の制限はエキスパートアドバイザーのテストで DLL が使用出来ないことです。セキュリティ上の理由から、遠隔エージェントでの DLL の呼び出しは絶対禁止となっています。ローカルエージェントでは、エキスパートアドバイザーのテストでの DLL の呼び出しは、適切な「インポート DLLを許可」でのみ許可されています。



© 2000–2025, MetaQuotes Ltd.

注意事項： DLL 呼び出しを必要とするサードパーティーのエキスパートアドバイザー（スクリプト、指標）の使用にあたって、端末の設定でこのオプションを許可する場合にとるリスクには注意するべきです。エキスパートアドバイザーがどのように（テストまたはチャートでの実行）使用されるかには関係なくです。

© 2000–2025, MetaQuotes Ltd.

定義済み変数

MQL5 プログラム（エキスパートアドバイザー、スクリプトまたはカスタム指標）では、実行開始時から、現在の価格チャートの状態を反映する事前定義された変数のセットがサポートされています。

事前定義された変数の値は、MQL5 プログラムが開始される前にクライアント端末によって設定されます。事前定義された変数は一定であり、MQL5 プログラムから変更することは出来ません。例外は、ResetLastError で 0 にリセット出来る _LastError です。

Variable	値
_AppliedTo	The _AppliedTo variable allows finding out the type of data, used for indicator calculation
_Digits	小数点以下の桁数
_Point	相場通貨での現在のシンボル点のサイズ
_LastError	最終のエラーコード
_Period	現在のチャートの時間軸
_RandomSeed	擬似乱数整数の現在の状態
_StopFlag	プログラム停止フラグ
_Symbol	現在のチャートの銘柄名
_UninitReason	初期化解除の理由のコード
_IsX64	The _IsX64 variable allows finding out the bit version of the terminal, in which an MQL5 application is running

事前定義された変数は、ライブラリ内で定義されることは出来ません。ライブラリは、ライブラリが呼び出されるプログラムの中で定義される変数を使用します。

© 2000–2025, MetaQuotes Ltd.

int _AppliedTo

_AppliedTo変数を使用すると、指標計算に使用されるデータの型を見つけることができます。

データ型	意味	指標計算に使用されるデータの説明
—	0	この指標ではOnCalculate()2番目の呼び出し形式を使用し、計算のためのデータはデータ配列のバッファでは指定されません。
Close	1	終値
Open	2	始値
High	3	高値
Low	4	安値
Median Price (HL/2)	5	Median price = (高値+安値)/2
Typical Price (HLC/3)	6	Typical price = (高値+安値+終値)/3
Weighted Price (HLCC/4)	7	Weighted price = (始値+高値+安値+終値)/4
1つ前の指標データ	8	チャートでこの指標の前に起動された指標のデータ
1番目の指標データ	9	チャートで1番目に起動された指標のデータ
指標ハンドル	10+	指標ハンドルを通してiCustom()関数に渡された指標のデータ _AppliedTo値は指標ハンドルを含みます。

例:

```
//+------------------------------------------------------------------+
//|  カスタム指標初期化関数                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 指標バッファマッピング
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
// 指標計算に使用されるデータの取得
   Print("_AppliedTo=",_AppliedTo);
   Print(getIndicatorDataDescription(_AppliedTo));
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//|  指標計算に使用されるデータの説明                                |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
string getIndicatorDataDescription(int data_id)
  {
   string descr="";
   switch(data_id)
     {
      case(0):descr="It's first type of OnCalculate() - no data buffer";
         break;
      case(1):descr="Indicator calculates on Close price";
         break;
      case(2):descr="Indicator calculates on Open price";
         break;
      case(3):descr="Indicator calculates on High price";
         break;
      case(4):descr="Indicator calculates on Low price";
         break;
      case(5):descr="Indicator calculates on Median Price (HL/2)";
         break;
      case(6):descr="Indicator calculates on Typical Price (HLC/3)";
         break;
      case(7):descr="Indicator calculates on Weighted Price (HLCC/4)";
         break;
      case(8):descr="Indicator calculates Previous Indicator's data";
         break;
      case(9):descr="Indicator calculates on First Indicator's data";
         break;
      default: descr="Indicator calculates on data of indicator with
handle="+string(data_id);
         break;
     }
//---
   return descr;
  }
```

参照

ENUM_APPLIED_PRICE

© 2000–2025, MetaQuotes Ltd.

int _Digits

_Digits 変数は、現在のチャートシンボルの価格精度を定義する小数点以下の桁を保存します。

Digits() 関数も使用出来ます。

© 2000-2025, MetaQuotes Ltd.

double _Point

_Point 変数は現在の通貨/シンボルのポイントサイズを含みます。

Point() 関数も使用出来ます。

© 2000-2025, MetaQuotes Ltd.

int _LastError

_LastError 変数は MQL5 プログラム実行中に発生した最終 エラーのコードを含みます。その値は ResetLastError() によってゼロにリセット出来ます。

直近エラーのコード取得には GetLastError() 関数を使用することも出来ます。

© 2000-2025, MetaQuotes Ltd.

ENUM_TIMEFRAMES _Period

_Period 関数は現在のチャートの時間軸を含みます。

Period() 関数も使用出来ます。

参照

PeriodSeconds、チャート時間軸、日付と時刻、オブジェクトの可視性

© 2000-2025, MetaQuotes Ltd.

_RandomSeed

擬似乱数整数を作成時に現在の状態を保管する変数です。_RandomSeed は MathRand() の呼び出し時に値を変えます。必要な初期条件を設定するには MathSrand() を使用します。

x MathRand() 関数はそれぞれの呼び出しで乱数整数を次の様に計算します。

```
x=_RandomSeed*214013+2531011;
_RandomSeed=x;
x=(x>>16)&0x7FFF;
```

参照

MathRand()、MathSrand()、整数型

© 2000-2025, MetaQuotes Ltd.

int _StopFlag

_StopFlag 変数は MQL5 プログラム停止フラグを含みます。このフラグは、通常動作中には 0 に設定されていますが、クライアント端末がプログラムを停止しようとうすると、0 以外の値に設定されます。

_StopFlag の状態をチェックするには IsStopped() 関数も使用出来ます。

© 2000–2025, MetaQuotes Ltd.

string _Symbol

_Symbol は現在のチャートの銘柄名を含みます。

Symbol() 関数も使用出来ます。

© 2000–2025, MetaQuotes Ltd.

int _UninitReason

_UninitReason 関数はプログラム初期化解除の理由を含みます。

このコードは通常 UninitializeReason() 関数で取得されます。

© 2000–2025, MetaQuotes Ltd.

int _IsX64

_IsX64 変数を使用するとMQL5 アプリケーションが実行されているターミナルのビット版を見つけることができます（32 ビットは_IsX64=0 で 64 ビットは_IsX64!=0）

また、TerminalInfoInteger(TERMINAL_X64) 関数も使用できます。

例：

```
// プログラムが実行されているターミナルの確認
   Print("_IsX64=",_IsX64);
   if(_IsX64)
      Print("Program ",__FILE__," is running in the 64-bit terminal");
   else
      Print("Program ",__FILE__," is running in the 32-bit terminal");
   Print("TerminalInfoInteger(TERMINAL_X64)=",TerminalInfoInteger(TERMINAL_X64));
```

参照

MQLInfoInteger、関数のインポート（#import）

© 2000–2025, MetaQuotes Ltd.

共通関数

どんな専門グループの中にも含まれていない、多目的な関数です。

関数	アクション
Alert	別ウィンドウにメッセージを表示します。
CheckPointer	オブジェクトポインタの型を返します。
Comment	チャートの左上隅にコメントを出力します。
CryptEncode	指定されたメソッドを使用して配列からデータを変換します。
CryptDecode	配列からのデータの逆変換を行います。
DebugBreak	デバグ中のプログラムのブレークポイント。
ExpertRemove	エキスパートアドバイザーを停止しチャートからアンロードします。
GetPointer	オブジェクトポインタを返します。
GetTickCount	システムが起動してからの経過時間をミリ秒数で返します。
GetTickCount64	システムが起動してからの経過時間をミリ秒数で返します。
GetMicrosecondCount	MQL5 プログラムの開始からの経過時間マイクロ秒数で返します。
MessageBox	メッセージボックスを作成、表示及び管理します。
PeriodSeconds	期間中の秒数を返します。
PlaySound	サウンドファイルを再生します。
Print	ログにメッセージを表示します。
PrintFormat	シンボルと値のセットを事前定義された形式に従ってフォーマットし、ログファイルにプリントします。
ResetLastError	所定の変数 _LastError の値をゼロに設定します。
ResourceCreate	データセットに基づいて画像のリソースを作成します。
ResourceFree	動的に作成されたリソースを削除します（割り当てられたメモリを解放します）。
ResourceReadImage	ResourceCreate() 関数で作成されたまたはコンパイル中に EX5 ファイルに保存されたグラフィックリソースからデータを読みます。
ResourceSave	リソースを指定されたファイルに保存します。
SetUserError	事前に定義された変数 _LastError を ERR_USER_ERROR_FIRST + user_error に等しい値に設定します。
SetReturnError	操作終了時に端末プロセスを返すコードを設定します。
Sleep	指定された間隔内で現在のエキスパートアドバイザーまたはスクリプトの実行を一時停止します。
TerminalClose	端末に操作完了を命令します。

© 2000–2025, MetaQuotes Ltd.

関数	アクション
TesterHideIndicators	EAで使用される指標の非/表示モードを設定します。
TesterStatistics	テストの結果に基づいて算出された指定の統計の値を返します。
TesterStop	テスト中にプログラム操作終了コマンドを発行します。
TesterDeposit	テスト中の資金入金をエミュレートする特殊関数です。資金管理システムの一部で使用することができます。
TesterWithdrawal	この特殊関数は、テストの過程で出金の動作をエミュレートします。
TranslateKey	現在の入力言語とコントロールキーの状態を考慮して、仮想キーコードでUnicode文字を返します。
ZeroMemory	参照によって渡された変数をリセットします。変数は、クラスとコンストラクタを含む構造体を除いては任意の型のものとすることが出来ます。

© 2000-2025, MetaQuotes Ltd.

アラート

別ウィンドウにメッセージを表示します。

```
void  Alert(
   argument,      // 初めの値
   ...            // 他の値
   );
```

パラメータ

argument

[in] コンマで区切られた任意の値。情報出力を複数の行に分割するには、改行文字「 ¥n 」または「 ¥r¥n 」を使用することが出来ます。パラメータの数は 64 を超えることは出来ません。

戻り値

なし

注意事項

配列は Alert() 関数には渡せません。配列は要素ごとに出力する必要があります。double 型のデータは小数点以下8桁で出力され、float 型のデータは小数点以下 5 桁で表示されます。実数を異なる精度または科学形式で出力するには DoubleToString() 関数が使用されます。

bool 型のデータは「 true 」または 「 false 」の文字列として出力されます。日付は YYYY.MM.DD HH:MI:SS として出力されます。日付を別の形式で表示するには TimeToString() 関数が使用されます。カラー型のデータは R、G、B の文字列、または、色がカラーセットに存在する場合は色名として出力されます。

Alert() 関数はストラテジーテスター内では使用できません。

例：

```
//--- 列挙体
enum ENUM_INTERSECT_DIRECTION
  {
   INTERSECT_DIRECTION_NONE= 0,  // 交差なし
   INTERSECT_DIRECTION_UP  = 1,  // 上向きに交差
   INTERSECT_DIRECTION_DOWN=-1,  // 下向きに交差
  };

//--- 入力パラメータ
input    uint                InpPeriod = 10;             // MA期間
input    int                 InpShift  = 0;              // MAシフト
input    ENUM_MA_METHOD      InpMethod = MODE_SMA;       // MA法
input    ENUM_APPLIED_PRICE  InpPrice  = PRICE_CLOSE;    // MA適応価格

//--- グローバル変数
int      ExtMaHandle;
int      ExtMaPeriod;
```

© 2000–2025, MetaQuotes Ltd.

```
double   ExtData[2];
MqlRates ExtRates[2];

//+------------------------------------------------------------------+
//| エキスパート初期化関数                                           |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 入力パラメータにゼロが指定されている場合、移動平均を計算する期間はデフォルト値(10)と等しくなる
   ExtMaPeriod=int(InpPeriod<1 ? 10 : InpPeriod);
//--- 指定されたパラメータを使用して移動平均指標のハンドルを作成する
   ExtMaHandle=iMA(Symbol(),PERIOD_CURRENT,ExtMaPeriod,InpShift,InpMethod,InpPrice);
   ResetLastError();
   if(ExtMaHandle==INVALID_HANDLE)
     {
      PrintFormat("Failed to create iMA() handle. Error code: %d",GetLastError());
      return(INIT_FAILED);
     }

//--- 最終価格更新時刻を取得する
   datetime tick_time=TickTime();
//--- 最後の2つのバーから移動平均データと価格データを取得する
   if(GetData(ExtMaHandle,ExtData,ExtRates) && tick_time!=0)
     {
      //--- 価格がMAを上回る場合
      if(ExtRates[1].close>ExtData[1])
        {
         //--- メッセージテキストを作成してアラートを表示する
         string message=StringFormat("Bar time: %s. The price is above the moving
average",TimeToString(ExtRates[1].time));
         Alert(message+" at "+TimeToString(tick_time,TIME_DATE|TIME_MINUTES|
TIME_SECONDS));
         /*
        結果:
         Alert: Bar time: 2024.02.16 18:00. The price is above the moving average at
 2024.02.16 18:47:43
         */
        }
      else
        {
      //--- 価格がMAを下回る場合
         if(ExtRates[1].close<ExtData[1])
           {
         //--- メッセージテキストを作成してアラートを表示する
            string message=StringFormat("Bar time: %s. The price is below the moving
average.",TimeToString(ExtRates[1].time));
            Alert(message+" at "+TimeToString(tick_time,TIME_DATE|TIME_MINUTES|
TIME_SECONDS));
            /*
           結果:
```

© 2000-2025, MetaQuotes Ltd.

```
            Alert: Bar time: 2024.02.16 19:00. The price is below the moving average
 at 2024.02.16 19:33:14
            */
           }
         else
           {
         //--- メッセージテキストを作成してアラートを表示する
            string message=StringFormat("Bar time: %s. The price and moving average
are equal.",TimeToString(ExtRates[1].time));
            Alert(message+" at "+TimeToString(tick_time,TIME_DATE|TIME_MINUTES|
TIME_SECONDS));
            /*
           結果:
            Alert: Bar time: 2024.02.16 20:00. The price and moving average are
equal at 2024.02.16 20:12:22
            */
           }
        }
     }

//--- 成功
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| エキスパートティック関数                                         |
//+------------------------------------------------------------------+
void OnTick()
  {
   ResetLastError();
//--- 最後の2つのバーから移動平均データと価格データを取得する
   if(!GetData(ExtMaHandle,ExtData,ExtRates))
      return;
//--- 現在の足の移動平均を横切る価格の方向を取得する
   ENUM_INTERSECT_DIRECTION intersect=GetIntersectDirection(ExtData,ExtRates);

//--- 前のメッセージを保存するための変数
   static string message_prev="";

//--- 価格が現在のバーの移動平均を上向きに超えた場合
   if(intersect==INTERSECT_DIRECTION_UP)
     {
 //--- 交差が発生したティック時間を取得する
      datetime tick_time=TickTime();
      if(tick_time==0)
         return;
      //--- メッセージテキストを作成する
      string message=StringFormat("Bar time: %s. The price crossed the MA from
bottom to top",TimeToString(ExtRates[1].time));
 //--- 前のメッセージが現在のメッセージと異なる場合は、メッセージと時間経過を含むアラートを表示する
      if(message!=message_prev)
```

© 2000–2025, MetaQuotes Ltd.

```
         {
          Alert(message+" at "+TimeToString(tick_time,TIME_DATE|TIME_MINUTES|
TIME_SECONDS));
          message_prev=message;
          /*
          Result:\
          Alert: Bar time: 2024.02.16 09:00. The price crossed the MA from bottom to
top at 2024.02.16 09:20:35
          */
         }
      }

//--- 価格が現在のバーの移動平均を下向きに超えた場合
   if(intersect==INTERSECT_DIRECTION_DOWN)
      {
 //--- 交差が発生したティック時間を取得する
       datetime tick_time=TickTime();
       if(tick_time==0)
          return;
       //--- メッセージテキストを作成する
       string message=StringFormat("Bar time: %s. The price crossed the MA from top
to bottom",TimeToString(ExtRates[1].time));
 //--- 前のメッセージが現在のメッセージと異なる場合は、メッセージと時間経過を含むアラートを表示する
       if(message!=message_prev)
         {
          Alert(message+" at "+TimeToString(tick_time,TIME_DATE|TIME_MINUTES|
TIME_SECONDS));
          message_prev=message;
          /*
          Result:\
          Alert: Bar time: 2024.02.16 10:00. The price crossed the MA from top to
bottom at 2024.02.16 10:42:15
          */
         }
      }
  }
//+------------------------------------------------------------------+
//|  価格と移動平均データを配列に取得する                                |
//+------------------------------------------------------------------+
bool GetData(int handle,double &ma_data[],MqlRates &price_data[])
  {
   ResetLastError();
//--- 最後の2つのバーから移動平均データを取得する
   if(CopyBuffer(handle,0,0,2,ma_data)!=2)
     {
      PrintFormat("CopyBuffer() failed. Error code: %d",GetLastError());
      return(false);
     }
//--- 最後の2つのバーから価格データを取得する
   if(CopyRates(Symbol(),PERIOD_CURRENT,0,2,price_data)!=2)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       PrintFormat("CopyRates() failed. Error code: %d",GetLastError());
       return(false);
      }

   return(true);
  }
//+------------------------------------------------------------------+
//| 移動平均を横切る価格の方向を返す                                 |
//+------------------------------------------------------------------+
ENUM_INTERSECT_DIRECTION GetIntersectDirection(double &ma_data[],MqlRates
&price_data[])
  {
   double ma0=ma_data[1];
   double ma1=ma_data[0];
   double close0=price_data[1].close;
   double close1=price_data[0].close;

   if(close1<=ma1 && close0>ma0)
      return(INTERSECT_DIRECTION_UP);
   else
     {
      if(close1>=ma1 && close0<ma0)
         return(INTERSECT_DIRECTION_DOWN);
      else
         return(INTERSECT_DIRECTION_NONE);
     }
  }
//+------------------------------------------------------------------+
//| 秒単位でのティック時刻を返す                                     |
//+------------------------------------------------------------------+
datetime TickTime()
  {
   MqlTick tick={};

   ResetLastError();
   if(!SymbolInfoTick(Symbol(),tick))
     {
      PrintFormat("SymbolInfoTick() failed. Error code: %d",GetLastError());
      return(0);
     }

   return(tick.time);
  }
```

© 2000–2025, MetaQuotes Ltd.

CheckPointer

この関数はオブジェクトポインタの型を返します。

```
ENUM_POINTER_TYPE  CheckPointer(
   object* anyobject       // オブジェクトポインタ
   );
```

パラメータ

anyobject

[in] オブジェクトポインタ

戻り値

ENUM_POINTER_TYPE 列挙のいずれかの値

注意事項

不正なポインタを呼び出す試みはプログラムの緊急終了を生じます。これが、ポインタを使用する前にCheckPointer 関数を呼び出すのが必要な理由です。ポインタは、次のような場合に不正である場合があります。

- ポインタが NULL
- オブジェクトが delete 演算子で削除された

この関数は、ポインタの有効性をチェックするために使用することが出来ます。ゼロ以外の値はポインタアクセスを保証します。

ポインタをすばやく検証するにはCheckPointer関数の暗黙的な呼び出しを介してチェックする演算子「!」（例）も使用できます。

例:

```
//+------------------------------------------------------------------+
//|  要素の削除によってリストを削除する                                |
//+------------------------------------------------------------------+
void CMyList::Destroy()
  {
//--- ループの中で作業するためのサービスポインタ
   CItem* item;
//--- ループを通過し、動的なポインタの削除を試みる
   while(CheckPointer(m_items)!=POINTER_INVALID)
     {
      item=m_items;
      m_items=m_items.Next();
      if(CheckPointer(item)==POINTER_DYNAMIC)
        {
         Print("Dynamic object ",item.Identifier()," to be deleted");
         delete (item);
        }
      else Print("Non-dynamic object ",item.Identifier()," cannot be deleted");
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//---
  }
```

参照

オブジェクト ポインタ、オブジェクト ポインタのチェック、delete オブジェクト 解除 演算 子

© 2000–2025, MetaQuotes Ltd.

Comment

この関数はチャートの左上隅にコメントを出力します。

```
void  Comment(
   argument,      // 初めの値
   ...            // 次の値
   );
```

パラメータ

...

[in] コンマで区切られた任意の値。情報出力を複数の行に分割するには、改行文字「 ¥n 」または「 ¥r¥n 」を使用することが出来ます。パラメータの数は 64 を超えることは出来ません。入力コメントの合計は(目に見えないシンボルを含んで)2,045 文字を超えることは出来ません(余分な文字は切り捨てられます)。

戻り値

なし

注意事項

配列は Alert() 関数には渡せません。配列は要素ごとに渡される必要があります。

double 型のデータは小数点以下最大 16 桁の精度で出力され、どちらの表記がよりコンパクトになるかに応じて、伝統的または科学形式で出力することが出来ます。float 型のデータは小数点以下 5 桁で出力されます。実数を異なる精度または事前定義された形式で出力するには DoubleToString() 関数が使用されます。

bool 型のデータは「 true 」または 「 false 」の文字列として出力されます。日付は YYYY.MM.DD HH:MI:SS として出力されます。日付を別の形式で表示するには TimeToString() 関数が使用されます。カラー型のデータはR、G、Bの文字列、または、色がカラーセットに存在する場合は色名として出力されます。

Comment() 関数はストラテジーテスターでの最適化中には使用できません。

例:

```
void OnTick()
  {
//---
   double Ask,Bid;
   int Spread;
   Ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
   Bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   Spread=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD);
//--- 値を 3 行に出力する
   Comment(StringFormat("Show prices\nAsk = %G\nBid = %G\nSpread = %
d",Ask,Bid,Spread));
  }
```

参照

© 2000-2025, MetaQuotes Ltd.

ChartSetString、ChartGetString

© 2000–2025, MetaQuotes Ltd.

CryptEncode

指定されたメソッドを使用して配列からデータを変換します。

```
int  CryptEncode(
   ENUM_CRYPT_METHOD   method,        // 方法
   const uchar&        data[],        // ソース配列
   const uchar&        key[],         // キー
   uchar&              result[]       // ターゲットの配列
   );
```

パラメータ

method

[in] データ変換方法。ENUM_CRYPT_METHOD 列挙のいずれかです。

data[]

[in] ソース配列

key[]

[in] キー配列

result[]

[out] ターゲットの配列

戻り値

受け取り側の配列のバイト数、またはエラーの場合は 0。エラー情報を取得するにはGetLastError() 関数が呼ばれます。

例:

```
//+------------------------------------------------------------------+
//| ArrayToHex                                                       |
//+------------------------------------------------------------------+
string ArrayToHex(uchar &arr[],int count=-1)
  {
   string res="";
//--- チェック
   if(count<0 || count>ArraySize(arr))
      count=ArraySize(arr);
//--- 16 進文字列に変換する
   for(int i=0; i<count; i++)
      res+=StringFormat("%.2X",arr[i]);
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                               |
//+------------------------------------------------------------------+
void OnStart()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   string text="The quick brown fox jumps over the lazy dog";
   string keystr="ABCDEFG";
   uchar src[],dst[],key[];
//--- キーを準備する
   StringToCharArray(keystr,key);
//--- テキストを元の配列 src[] に複製する
   StringToCharArray(text,src);
//--- 初期データをプリントする
   PrintFormat("Initial data: size=%d, string='%
s'",ArraySize(src),CharArrayToString(src));
//--- src[] を key[] のDES 56-ビットキーを使用して暗号化する
   int res=CryptEncode(CRYPT_DES,src,key,dst);
//--- エラーをチェックする
   if(res>0)
     {
      //--- 暗号化されたデータをプリントする
      PrintFormat("Encoded data: size=%d %s",res,ArrayToHex(dst));
      //--- dst[] を src[] にデコードする
      res=CryptDecode(CRYPT_DES,dst,key,src);
      //--- エラーをチェックする
      if(res>0)
        {
         //--- デコードされたデータをプリントする
         PrintFormat("Decoded data: size=%d, string='%
s'",ArraySize(src),CharArrayToString(src));
        }
      else
         Print("Error in CryptDecode. Error code=",GetLastError());
     }
   else
      Print("Error in CryptEncode. Error code=",GetLastError());
  }
```

参照

配列関数、CryptDecode()

© 2000-2025, MetaQuotes Ltd.

CryptDecode

CryptEncode()で返還された配列からデータの逆変換を行います。

```
int  CryptEncode(
   ENUM_CRYPT_METHOD   method,         // 方法
   const uchar&        data[],         // ソース配列
   const uchar&        key[],          // キー
   uchar&              result[]        // ターゲットの配列
   );
```

パラメータ

method

[in] データ変換方法。ENUM_CRYPT_METHOD 列挙のいずれかです。

data[]

[in] ソース配列

key[]

[in] キー配列

result[]

[out] ターゲットの配列

戻り値

受け取り側の配列のバイト数、またはエラーの場合は 0。エラー情報を取得するにはGetLastError() 関数が呼ばれます。

例：

```
input string InpKey = "ABCDEFG";  // Encryption key

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string text="The quick brown fox jumps over the lazy dog";
   uchar src[],dst[],key[];

//--- 暗号化キーを準備する
   StringToCharArray(InpKey,key);
//--- src[]ソース配列を準備する
   StringToCharArray(text,src);
//--- ソースデータを表示する
   PrintFormat("Initial data: size=%d, string='%
s'",ArraySize(src),CharArrayToString(src));
```

© 2000-2025, MetaQuotes Ltd.

```
//--- key[] 56ビット鍵とDESメソッドを使用してsrc[]配列を暗号化する
   int res=CryptEncode(CRYPT_DES,src,key,dst);

//--- 暗号化結果を確認する
   if(res>0)
     {
      //--- 暗号化されたデータを表示する
      PrintFormat("Encoded data: size=%d %s",res,ArrayToHex(dst));
//--- key[] 56ビット鍵とDESメソッドを使用してdst[]配列を復号化する
      res=CryptDecode(CRYPT_DES,dst,key,src);
      //--- 結果を確認する
      if(res>0)
        {
      //--- 復号化されたデータを表示する
         PrintFormat("Decoded data: size=%d, string='%
s'",ArraySize(src),CharArrayToString(src));
        }
      else
         Print("CryptDecode failed. Error: ",GetLastError());
     }
   else
      Print("CryptEncode failed. Error: ",GetLastError());
  }
//+------------------------------------------------------------------+
//| ArrayToHex                                                       |
//+------------------------------------------------------------------+
string ArrayToHex(uchar &arr[],int count=-1)
  {
   string res="";

//--- サイズを確認する
   if(count<0 || count>ArraySize(arr))
      count=ArraySize(arr);

//--- 16 進文字列に変換する
   for(int i=0; i<count; i++)
      res+=StringFormat("%.2X",arr[i]);

   return(res);
  }
```

参照

配列関数、CryptEncode()

© 2000-2025, MetaQuotes Ltd.

DebugBreak

デバッグ中のプログラムのブレークポイントです

```
void  DebugBreak();
```

戻り値

なし

注意事項

MQL5　プログラムの実行はデバッグモードで起動された場合のみ中断されます。この関数は変数の値及び/またはステップ・バイ・ステップ実行の表示に使用することが出来ます。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- F5を使用してファイルをコンパイルする
//--- デバッグモードでは、i  == j の場合、DebugBreak() 文字列で停止する
   for(int i=0,j=20; i<20; i++,j--)
     {
      if(i==j)
         DebugBreak();
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

ExpertRemove

この関数はエキスパートアドバイザーを停止しチャートからアンロードします。

```
void  ExpertRemove();
```

戻り値

なし

注意事項

エキスパートアドバイザーは、ExpertRemove() の呼び出し直後に停止されるわけではありません。EA 動作を停止するフラグのみが設定されます。つまり、次のイベントは処理されず、OnDeinit() が呼び出されてエキスパートアドバイザーがアンロードされてチャートから削除されます。

ストラテジーテスターのOnInit()ハンドラでのExpertRemove()の呼び出しによって、現在のパラメータセットでのテストがキャンセルされます。このような終了は初期化エラーとみなされます。

ストラテジーテスターでEAが正常に初期化された後でExpertRemove()が呼び出されると、テストはOnDeinit()とOnTester()の呼び出しによって正常に終了します。この場合、取引統計と最適化基準値はすべて取得されますが。

例:

```
//+------------------------------------------------------------------+
//|                                              Test_ExpertRemove.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
input int ticks_to_close=20;// EA アンロードまでのティック
//+------------------------------------------------------------------+
//| エキスパート初期化解除に使用される関数                           |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Print(TimeCurrent(),": " ,__FUNCTION__," reason code = ",reason);
//--- コメントをクリアする
   Comment("");
//---
  }
//+------------------------------------------------------------------+
//| エキスパートティック関数                                         |
//+------------------------------------------------------------------+
void OnTick()
  {
   static int tick_counter=0;
//---
```

© 2000-2025, MetaQuotes Ltd.

```
   tick_counter++;
   Comment("\nBefore unloading expert advisor ",__FILE__," left",
           (ticks_to_close-tick_counter)," ticks");
//--- 前
   if(tick_counter>=ticks_to_close)
     {
      ExpertRemove();
      Print(TimeCurrent(),": ",__FUNCTION__," expert advisor will be unloaded");
     }
   Print("tick_counter =",tick_counter);
//---
  }
//+------------------------------------------------------------------+
```

参照

プログラムの実行、クライアント端末イベント

© 2000–2025, MetaQuotes Ltd.

GetPointer

この関数はオブジェクトポインタを返します。

```
void*  GetPointer(
   any_class anyobject       // 任意のクラスオブジェクト
   );
```

パラメータ

anyobject

[in] 任意のクラスオブジェクト

戻り値

オブジェクトポインタ

注意事項

クラスオブジェクトのみがポインタを持ちます。構造体インスタンスと基本データ型の変数にはポインタはありません。new() 演算子によって作成されなくても、例えばオブジェクトの配列で自動的に作成されたクラスオブジェクトにはポインタがあります。しかし、このポインタは POINTER_AUTOMATIC 型の自動型であり、delete() 演算子が適用出来ません。これ以外の点では、ポインタ型は POINTER_DYNAMIC 型の動的ポインタとは異なりません。

構造体型また基本データ型の変数はポインタを持っていないので、それらに GetPointer() 関数を適用することは禁じられています。また、ポインタを関数の引数として渡すことも禁止されています。これらの場合全てで、コンパイラはエラーを通知します。

不正なポインタを呼び出す試みはプログラムの緊急終了を生じます。これが、ポインタを使用する前に CheckPointer() 関数を呼び出すのが必要な理由です。ポインタは、次のような場合に不正であることがあります。

- ポインタが NULL
- オブジェクトが delete 演算子で削除された

この関数は、ポインタの有効性を確認するために使用することが出来ます。ゼロ以外の値はポインタアクセスを保証します。

例:

```
//+------------------------------------------------------------------+
//|                                                Check_GetPointer.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

//+------------------------------------------------------------------+
//| リスト要素を実装するクラス                                                 |
//+------------------------------------------------------------------+
```

© 2000–2025, MetaQuotes Ltd.

```
class CItem
  {
   int               m_id;
   string            m_comment;
   CItem*            m_next;
public:
                     CItem() { m_id=0; m_comment=NULL; m_next=NULL; }
                    ~CItem() { Print("Destructor of ",m_id,
                                     (CheckPointer(GetPointer(this))
==POINTER_DYNAMIC)?
                                     "dynamic":"non-dynamic"); }
   void              Initialize(int id,string comm) { m_id=id; m_comment=comm; }
   void              PrintMe() { Print(__FUNCTION__,":",m_id,m_comment); }
   int               Identifier() { return(m_id); }
   CItem*            Next() {return(m_next); }
   void              Next(CItem *item) { m_next=item; }
  };
//+------------------------------------------------------------------+
//| リストの最も単純なクラス                                         |
//+------------------------------------------------------------------+
class CMyList
  {
   CItem*            m_items;
public:
                     CMyList() { m_items=NULL; }
                    ~CMyList() { Destroy(); }
    bool             InsertToBegin(CItem* item);
    void             Destroy();
  };
//+------------------------------------------------------------------+
//| 先頭にリスト要素を挿入する                                       |
//+------------------------------------------------------------------+
bool CMyList::InsertToBegin(CItem* item)
  {
   if(CheckPointer(item)==POINTER_INVALID) return(false);
//---
   item.Next(m_items);
   m_items=item;
//---
   return(true);
  }
//+------------------------------------------------------------------+
//| 要素の削除によってリストを削除する                               |
//+------------------------------------------------------------------+
void CMyList::Destroy()
  {
//--- ループの中で作業するためのサービスポインタ
   CItem* item;
//--- ループを通過し、動的なポインタの削除を試みる
```

© 2000-2025, MetaQuotes Ltd.

```
   while(CheckPointer(m_items)!=POINTER_INVALID)
     {
      item=m_items;
      m_items=m_items.Next();
      if(CheckPointer(item)==POINTER_DYNAMIC)
        {
         Print("Dynamyc object ",item.Identifier()," to be deleted");
         delete (item);
        }
      else Print("Non-dynamic object ",item.Identifier()," cannot be deleted");
     }
//---
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
   CMyList list;
   CItem   items[10];
   CItem*  item;
//--- 作成し、リスト動的オブジェクトのポインタに追加
   item=new CItem;
   if(item!=NULL)
     {
      item.Initialize(100,"dynamic");
      item.PrintMe();
      list.InsertToBegin(item);
     }
//--- 自動ポインタをリストに追加する
   for(int i=0; i<10; i++)
     {
      items[i].Initialize(i,"automatic");
      items[i].PrintMe();
      item=GetPointer(items[i]);
      if(CheckPointer(item)!=POINTER_INVALID)
         list.InsertToBegin(item);
     }
//--- リスト先頭に動的なオブジェクトポインタをあと 1 つ追加する
   item=new CItem;
   if(item!=NULL)
     {
      item.Initialize(200,"dynamic");
      item.PrintMe();
      list.InsertToBegin(item);
     }
//--- リスト要素を全て削除する
   list.Destroy();
//--- スクリプト終了後、全てのリスト要素が削除される
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 端末内の「エキスパート」タブを参照する
  }
```

参照

オブジェクト ポインタ、オブジェクト ポインタのチェック、delete オブジェクト 解除 演算子

© 2000-2025, MetaQuotes Ltd.

GetTickCount

GetTickCount() 関数はシステムが起動してからの経過時間をミリ秒数で返します。

```
uint  GetTickCount();
```

戻り値

uint 型の値

注意事項

カウンタは、システムタイマの制約によって制限されます。時間は符号なし整数として格納されるので、コンピュータが途切れることなく動作した場合 49.7日 ごとに溢れます。

例:

```
#define MAX_SIZE 40
//+------------------------------------------------------------------+
//|  40フィボナッチ数の計算時間を測定するためのスクリプト                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 初期値を記録する
   uint start=GetTickCount();
//--- フィボナッチ数列の次の番号を取得するための変数
   long fib=0;
//--- フィボナッチ数列から番号の指定された量をループで計算する
   for(int i=0;i<MAX_SIZE;i++) fib=TestFibo(i);
//--- かかった時間をミリ秒単位で取得する
   uint time=GetTickCount()-start;
//--- メッセージをエキスパート操作ログへ出力する
   PrintFormat("Calculating %d first Fibonacci numbers took %d ms",MAX_SIZE,time);
//--- スクリプトが完了した
   return;
  }
//+------------------------------------------------------------------+
//|  フィボナッチ数をシリアル番号で取得するための関数                                    |
//+------------------------------------------------------------------+
long TestFibo(long n)
  {
//--- フィボナッチ数列の最初のメンバ
   if(n<2) return(1);
//--- 他の全てのメンバは、以下の式で計算される
   return(TestFibo(n-2)+TestFibo(n-1));
  }
```

参照

日付と時刻, EventSetMillisecondTimer, GetTickCount64, GetMicrosecondCount

© 2000-2025, MetaQuotes Ltd.

GetTickCount64

GetTickCount64() 関数はシステムが起動してからの経過時間をミリ秒数で返します。

```
ulong  GetTickCount64();
```

戻り値

ulong型の値。

注意事項

カウンタはシステムタイマーの精度に制限され、通常は10～16ミリ秒の精度で結果が返されます。uint型であって、コンピュータが継続して操作された場合には49.7日ごとに内容がオーバーフローするGetTickCountとは異なり、GetTickCount64()は、オーバーフローの影響を受けず、コンピュータ操作時間は無制限です。

例：

```
#define MAX_SIZE 40

//+------------------------------------------------------------------+
//|  40フィボナッチ数の計算時間を測定するためのスクリプト                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
   long fib_array[MAX_SIZE];

//--- 初期値を収納する
   ulong start=GetTickCount64();
//--- フィボナッチ数列から指定された数の数値を計算するループ
   for(int i=0;i<MAX_SIZE;i++)
      fib_array[i]=TestFibo(i);
//--- かかった時間をミリ秒単位で取得する
   ulong time=GetTickCount64()-start;

//--- エキスパート操作ログでエラーメッセージを表示する
   ArrayPrint(fib_array);
   PrintFormat("Calculating the first %d Fibonacci numbers took %I64u
ms",MAX_SIZE,time);
  }
//+------------------------------------------------------------------+
//|  フィボナッチ数をシリアル番号で取得するための関数                                    |
//+------------------------------------------------------------------+
long TestFibo(long n)
  {
//--- フィボナッチ数列の最初のメンバ
   if(n<2)
      return(1);
//--- 続く全てのメンバは、以下の式で計算される
```

© 2000-2025, MetaQuotes Ltd.

```
   return(TestFibo(n-2)+TestFibo(n-1));
  }
```

参照

日付と時刻、EventSetMillisecondTimer、GetTickCount、GetMicrosecondCount

© 2000-2025, MetaQuotes Ltd.

GetMicrosecondCount

GetMicrosecondCount() 関数はMQL5 プログラムの開始からの経過時間マイクロ秒数で返します。

```
ulong  GetMicrosecondCount();
```

戻り値

ulong 型の値

例:

```
//+------------------------------------------------------------------+
//|  テスト関数                                                      |
//+------------------------------------------------------------------+
void Test()
  {
   int    res_int=0;
   double res_double=0;
//---
   for(int i=0;i<10000;i++)
     {
      res_int+=i*i;
      res_int++;
      res_double+=i*i;
      res_double++;
     }
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                              |
//+------------------------------------------------------------------+
void OnStart()
  {
   uint   ui=0,ui_max=0,ui_min=INT_MAX;
   ulong  ul=0,ul_max=0,ul_min=INT_MAX;
//--- 測定値の数
   for(int count=0;count<1000;count++)
     {
      uint  ui_res=0;
      ulong ul_res=0;
      //---
      for(int n=0;n<2;n++)
        {
         //--- 測定の種類を選ぶ
         if(n==0) ui=GetTickCount();
         else     ul=GetMicrosecondCount();
         //--- コードを実行する
         Test();
         //--- 測定結果を追加する(種類に応じて)
         if(n==0) ui_res+=GetTickCount()-ui;
```

© 2000-2025, MetaQuotes Ltd.

```
         else      ul_res+=GetMicrosecondCount()-ul;
        }
      //--- 測定値の最小時間と最大時間を計算する
      if(ui_min>ui_res) ui_min=ui_res;
      if(ui_max<ui_res) ui_max=ui_res;
      if(ul_min>ul_res) ul_min=ul_res;
      if(ul_max<ul_res) ul_max=ul_res;
     }
//---
   Print("GetTickCount error(msec): ",ui_max-ui_min);
   Print("GetMicrosecondCount error(msec): ",DoubleToString((ul_max-
ul_min)/1000.0,2));
  }
```

参照

日付と時刻

© 2000-2025, MetaQuotes Ltd.

MessageBox

メッセージボックスを作成、表示及び管理します。メッセージボックスには、メッセージとヘッダ、また、事前に定義された徴候やコマンドボタンの任意の組み合わせが含まれています。

```
int  MessageBox(
   string  text,              // メッセージテキスト
   string  caption=NULL,      // ボックスヘッダ
   int     flags=0            // ボックスにボタンのセットを定義する
   );
```

パラメータ

text

[in] 出力メッセージを含むテキスト

caption=NULL

[in] ボックスヘッダに表示される任意のテキスト。パラメータが空の場合、エキスパートアドバイザ名がボックスヘッダに示されます。

flags=0

[in] メッセージボックスの外観と動作を定義する任意のフラグ。フラグは特殊なグループの組み合わせであることも出来ます。

戻り値

関数が正常に実行された場合は MessageBox() リターンコード値のいずれか

注意事項

MessageBox() を呼び出すと、ユーザーの応答を待っている間実行スレッドがずっと中断されるため、この関数をカスタム指標で使用することはできません。各銘柄のすべての指標は単一のスレッドで実行されるため、この銘柄のすべての時間枠のすべてのチャートが一時停止されます。

MessageBox() 関数はストラテジーテスター内では使用できません。

例：

```
//+------------------------------------------------------------------+
//|  EAは、一連の不採算取引が指定された数に達すると
//|  さらなる作業を求めるメッセージ ボックス ウィンドウを表示する
//|  指定されたバー数待って、作業を続行するリクエストを含むMessageBoxを
//|  表示する
//|  簡単にするために、EAはマジックナンバーによって独自の「ポジション」を
//|  制御しないため、確認するには、
//|  損失のある複数のポジションを手動で開閉する必要がある
//+------------------------------------------------------------------+

//--- 入力パラメータ
input uint InpMaxLossDeals      =  3;    // 最大損失取引
input uint InpInactivityNumBars =  5;    // EAが非アクティブなバーの数
```

© 2000-2025, MetaQuotes Ltd.

```
//--- グローバル変数
bool      ExtFirstStart=true;              // 初起動フラグ
bool      ExtFlag=true;                    // EAの動作を許可するフラグ
uint      ExtNumLoss;                      // 連続不採算取引の回数
datetime  ExtTimeLastLoss;                 // 負けポジションを決済する最後の取引の時間

//+------------------------------------------------------------------+
//|  エキスパート初期化関数                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 連続の負け取引の数と、ポジションを閉じるための最後の取引の時間を取得する
   ExtNumLoss=GetNumLosingTradesInRow(ExtTimeLastLoss);

   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//|  エキスパート初期化解除関数                                      |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   Comment("");
  }
//+------------------------------------------------------------------+
//|  エキスパートティック関数                                        |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- シリーズで最後に閉じた負けポジションから経過したバー数を決定する
   int bars_remaining=iBarShift(Symbol(),PERIOD_CURRENT,ExtTimeLastLoss);

//--- 初起動の場合
   if(ExtFirstStart)
     {
      //--- 一連の不採算ポジションの後、指定された数のバーがすでに経過した場合、EA動作フラグを設定する
      if(bars_remaining>(int)InpInactivityNumBars)
         ExtFlag=true;
      ExtFirstStart=false;
     }

//--- EA操作フラグが無効な場合
   if(!ExtFlag)
     {
      Comment(StringFormat("The advisor is stopped for %d bars. Num Loss positions:
%u, Time last loss: %s",
                           (InpInactivityNumBars-
bars_remaining),ExtNumLoss,TimeToString(ExtTimeLastLoss,TIME_DATE|TIME_MINUTES|
TIME_SECONDS)));
```

© 2000-2025, MetaQuotes Ltd.

```
 //--- 一連の不採算ポジションの後、指定された数のバーが経過した場合
      if(bars_remaining>(int)InpInactivityNumBars)
        {
 //--- 指定されたテキストとウィンドウタイトルを持つメッセージ ボックスウィンドウを表示する
 //--- リクエストウィンドウには2つのはい/いいえボタンと疑問符の付いたアイコンがある
 //--- デフォルトでは「はい」ボタンが選択されている
         string mb_text="The specified number of bars of EA inactivity have passed.
\n Continue its work?";
         string mb_caption="Please note";
         int    mb_id=MessageBox(mb_text,mb_caption,MB_YESNO|MB_ICONQUESTION|
MB_DEFBUTTON1);
 //--- メッセージボックスからのリターンコードが「はい」ボタンが押されたことを示す場合、EA操作フラグを設定する
         if(mb_id==IDYES)
           {
            ExtFlag=true;
            return;
           }
        }
      //--- EA操作フラグが無効な場合、OnTick()を終了する
      return;
     }

//--- EA操作フラグが設定されている - EAは以下のコードで提供されるように機能する
   Comment(StringFormat("The advisor is working. Num Loss positions: %u, Time last
loss: %s, Elapsed Bars: %d",
           ExtNumLoss,TimeToString(ExtTimeLastLoss,TIME_DATE|TIME_MINUTES|
TIME_SECONDS),bars_remaining));
  }
//+------------------------------------------------------------------+
//| TradeTransaction関数                                             |
//+------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction& trans,
                        const MqlTradeRequest& request,
                        const MqlTradeResult& result)
  {
//--- 取引タイプが取引を履歴に追加する場合
   if(trans.type==TRADE_TRANSACTION_DEAL_ADD)
     {
 //--- 取引チケットを取得し、チケットごとにリストから取引を選択する
      ulong deal_ticket=trans.deal;
      if(HistoryDealSelect(deal_ticket))
        {
 //--- これが市場からのエグジット取引の場合、連続の負け取引の数と最後の取引の時間を取得する
         ENUM_DEAL_ENTRY entry=(ENUM_DEAL_ENTRY)
HistoryDealGetInteger(deal_ticket,DEAL_ENTRY);
         if(entry==DEAL_ENTRY_OUT || entry==DEAL_ENTRY_INOUT ||
entry==DEAL_ENTRY_OUT_BY)
            ExtNumLoss=GetNumLosingTradesInRow(ExtTimeLastLoss);
        }
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 連続した負け取引の数が指定された値より大きく、EA操作フラグが有効な場合
   if(ExtNumLoss>=InpMaxLossDeals && ExtFlag)
     {
      //--- 指定されたテキストとウィンドウタイトルを持つメッセージ ボックスウィンドウを表示する
      //--- リクエストウィンドウには2つのはい/いいえボタンと感嘆符の付いたアイコンがある
      //--- デフォルトでは「いいえ」ボタンが選択されている
      string mb_text="The number of losing trades has reached the specified maximum.
The advisor is stopped.\n Continue its work?";
      string mb_caption="Attention!";
      int    mb_id=MessageBox(mb_text,mb_caption,MB_YESNO|MB_ICONQUESTION|
MB_DEFBUTTON2);
      //--- メッセージボックスからのリターンコードが「いいえ」ボタンが押されたことを示す場合、EA操作フラグを無効にする
      if(mb_id==IDNO)
         ExtFlag=false;
     }
  }
//+------------------------------------------------------------------+
//|  連続不採算取引の回数と                                          |
//|  損失ポジションを決済するための最後の取引の時刻を返す            |
//+------------------------------------------------------------------+
uint GetNumLosingTradesInRow(datetime &time_last_deal)
  {
//--- 履歴全部を選択する
   if(!HistorySelect(0,TimeCurrent()))
      return(0);

//--- ループ内の過去の取引のリストから次の取引チケットを取得する
   uint res=0;
   uint total=HistoryDealsTotal();
   for(int i=(int)total-1; i>=0; i--)
     {
      ulong deal_ticket=HistoryDealGetTicket(i);
      if(deal_ticket>0)
        {
 //--- ポジションを閉じるための取引ではない場合は、次の取引に進む
         ENUM_DEAL_ENTRY entry=(ENUM_DEAL_ENTRY)
HistoryDealGetInteger(deal_ticket,DEAL_ENTRY);
         if(entry!=DEAL_ENTRY_OUT && entry!=DEAL_ENTRY_OUT_BY &&
entry!=DEAL_ENTRY_INOUT)
            continue;
         //--- ポジションを閉じた結果利益が得られた場合、ループを中断する
         if(!IsClosePositionWithLoss(deal_ticket))
            break;
         //--- 連続損失取引のカウンターを増やす
         res++;
         //--- 最大取引時間を変数に書き込む(最後の取引時間を探す)
         datetime deal_time=(datetime)HistoryDealGetInteger(deal_ticket,DEAL_TIME);
         if(deal_time>time_last_deal)
```

© 2000-2025, MetaQuotes Ltd.

```
            time_last_deal=deal_time;
         }
      }

//--- 連続不採算取引の回数を返す
   return(res);
  }
//+------------------------------------------------------------------+
//|  損失でポジションを閉じるフラグを返す                                |
//+------------------------------------------------------------------+
bool IsClosePositionWithLoss(const ulong deal_ticket)
  {
//--- 取引からの利益に影響を与えるプロパティの値を取得する
   double profit=HistoryDealGetDouble(deal_ticket,DEAL_PROFIT);
   double comission=HistoryDealGetDouble(deal_ticket,DEAL_COMMISSION);
   double swap=HistoryDealGetDouble(deal_ticket,DEAL_SWAP);
   double fee=HistoryDealGetDouble(deal_ticket,DEAL_FEE);

//--- 受け取ったプロパティの合計値が負であることを示すフラグを返す
   return(profit+comission+swap+fee<0);
  }
```

© 2000-2025, MetaQuotes Ltd.

PeriodSeconds

この関数は期間中の秒数を返します。

```
int  PeriodSeconds(
   ENUM_TIMEFRAMES  period=PERIOD_CURRENT      // チャート期間
   );
```

パラメータ

period=PERIOD_CURRENT

[in] ENUM_TIMEFRAMES列挙からのチャート期間。パラメータが指定されていない場合は、プログラムが実行されている現在のチャートの期間の秒数を返します。

戻り値

選択した期間の秒数

例：

```
//--- 入力パラメータ
input ENUM_TIMEFRAMES InpPeriod1 = PERIOD_CURRENT;    // First Period
input ENUM_TIMEFRAMES InpPeriod2 = PERIOD_M1;         // Second Period

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- InpPeriod1およびInpPeriod2チャート期間の秒数を取得する
   int sec1=PeriodSeconds(InpPeriod1);
   int sec2=PeriodSeconds(InpPeriod2);
//--- 受信した値をログに表示する
   PrintFormat("Seconds in period %s: %lu, in period %s: %
lu",TimeframeDescription(InpPeriod1),sec1,TimeframeDescription(InpPeriod2),sec2);
//--- InpPeriod1のチャート期間のバーにInpPeriod2チャート期間のバーが何本含まれるかを計算する
   int res=sec1/sec2;
   if(res==0)
      res=1;
//--- 取得した値をログに表示する
   PrintFormat("One bar %s contains %d bars %
s",TimeframeDescription(InpPeriod1),res,TimeframeDescription(InpPeriod2));
   /*
  結果:
   Seconds in period M5: 300, in period M1: 60
   One bar M5 contains 5 bars M1
   */
  }
//+------------------------------------------------------------------+
//|  時間枠名を返す                                                  |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
string TimeframeDescription(const ENUM_TIMEFRAMES period)
  {
   return(StringSubstr(EnumToString(period==PERIOD_CURRENT ? Period() : period),
7));
  }
```

参照

_Period、チャート時間軸、日付と時刻、オブジェクトの可視性

© 2000–2025, MetaQuotes Ltd.

PlaySound

サウンドファイルを再生します。

```
bool  PlaySound(
   string  filename      // ファイル名
   );
```

パラメータ

filename

[in] サウンドファイルへのパスファイル名がNULLの場合は、再生が停止されます。

戻り値

ファイルが見つかった場合はtrue、それ以外の場合はfalse

注意事項

ファイルは terminal_directory¥Sounds またはそのサブディレクトリに位置しなければいけません。WAVファイルのみが再生されます。

PlaySound() のNULLパラメータでの呼び出しは、再生を停止します。

PlaySound() 関数はストラテジーテスター内では使用できません。

例：

```
#include <Trade\Trade.mqh>
#define MAGIC (123)

//--- 入力パラメータ
input string          InpFileNameOK  = "ok.wav";               // 成功音声ファイル
input string          InpFileNameErr = "timeout.wav";          // エラー音声ファイル
input ENUM_ORDER_TYPE InpOrderType   = ORDER_TYPE_BUY_LIMIT;   // 注文タイプ
input double          InpLots        = 0.1;                    // ロット

//--- グローバル変数
CTrade ExtTrade;

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 銘柄設定に従って実行によりマジックと注文のタイプを設定する
   ExtTrade.SetExpertMagicNumber(MAGIC);
   ExtTrade.SetTypeFillingBySymbol(Symbol());
//--- サウンド再生付きで注文またはポジションを開く関数を呼び出する
   OrderSendWithAudio();
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  この関数は注文を発注するかポジションを開き                      |
//|  成功またはエラーの音声を再生する                                |
//+------------------------------------------------------------------+
void OrderSendWithAudio(void)
  {
   bool     res=true;
   MqlTick tick= {};

   ResetLastError();
   if(!SymbolInfoTick(Symbol(),tick))
     {
      Print("SymbolInfoTick() failed. Error code: ",GetLastError());
      PlaySound(InpFileNameErr);
      return;
     }
//--- サーバにリクエストを送信する
   switch(InpOrderType)
     {
      case ORDER_TYPE_BUY :
         res=ExtTrade.Buy(InpLots);
         break;
      case ORDER_TYPE_BUY_LIMIT :
         res=ExtTrade.BuyLimit(InpLots,NormalizeDouble(tick.ask-
100*Point(),Digits()));
         break;
      case ORDER_TYPE_BUY_STOP :
         res
=ExtTrade.BuyStop(InpLots,NormalizeDouble(tick.ask+100*Point(),Digits()));
         break;
      case ORDER_TYPE_SELL :
         res=ExtTrade.Sell(InpLots);
         break;
      case ORDER_TYPE_SELL_LIMIT :
         res
=ExtTrade.SellLimit(InpLots,NormalizeDouble(tick.bid+100*Point(),Digits()));
         break;
      case ORDER_TYPE_SELL_STOP :
         res=ExtTrade.SellStop(InpLots,NormalizeDouble(tick.bid-
100*Point(),Digits()));
         break;
      default :
         res=false;
     }
   if(!res)
      Print("Error ",GetLastError());
   Print(ExtTrade.ResultRetcodeDescription());

//--- リクエストが受け入れられたらOk.wavを再生する
```

© 2000-2025, MetaQuotes Ltd.

```
   if(ExtTrade.ResultRetcode()==TRADE_RETCODE_DONE)
      PlaySound(InpFileNameOK);
   else
      PlaySound(InpFileNameErr);
  }
```

参照

リソース

© 2000-2025, MetaQuotes Ltd.

Print

エキスパートアドバイザーログにメッセージを入力します。パラメータは任意の型です。

```
void  Print(
   argument,      // 初めの値
   ...            // 次の値
   );
```

パラメータ

...

[in] コンマで区切られた任意の値。パラメータの数は 64 を超えることは出来ません。

注意事項

配列は Print() 関数には渡せません。配列は要素ごとに渡される必要があります。

double 型のデータは小数点以下最大 16 桁の精度で出力され、どちらの表記がよりコンパクトになるかに応じて、伝統的または科学形式で出力することが出来ます。float 型のデータは小数点以下 5 桁で出力されます。実数を異なる精度または事前定義された形式で出力するには PrintFormat()関数が使用されます。

bool 型のデータは「 true 」または「 false 」のラインとして出力されます。日付は YYYY.MM.DD HH:MI:SS として出力されます。日付を別の形式で表示するには TimeToString() 関数が使用されます。カラー型のデータはR、G、Bの文字列、または、色がカラーセットに存在する場合は色名として出力されます。

Print() 関数はストラテジーテスターでの最適化中には使用できません。

例:

```
void OnStart()
  {
//--- DBL_MAX を Print() を使用して出力する。これは PrintFormat("%%.16G",DBL_MAX) と同等
   Print("---- how DBL_MAX looks like -----");
   Print("Print(DBL_MAX)=",DBL_MAX);
//--- DBL_MAX 数を PrintFormat() を使用して出力する
   PrintFormat("PrintFormat(%%.16G,DBL_MAX)=%.16G",DBL_MAX);
//--- エキスパート操作ログへ出力する
// Print(DBL_MAX)=1.797693134862316e+308
// PrintFormat(%.16G,DBL_MAX)=1.797693134862316E+308

//--- float の出力され方を参照する
   float c=(float)M_PI; // 明示的にターゲット型にキャストする必要がある
   Print("c=",c, "    Pi=",M_PI, "    (float)M_PI=",(float)M_PI);
// c=3.14159    Pi=3.141592653589793    (float)M_PI=3.14159

//--- 実数型の算術演算で何が起こるかを表示する
   double a=7,b=200;
   Print("---- Before arithmetic operations");
   Print("a=",a,"   b=",b);
   Print("Print(DoubleToString(b,16))=",DoubleToString(b,16));
//--- a を b で割る( 7 / 200 )
```

© 2000-2025, MetaQuotes Ltd.

```
   a=a/b;
//--- B変数にて値を復元するエミュレートをする
   b=7.0/a; // b=7.0/(7.0/200.0)=>7.0/7.0*200.0=200 が期待されるが、異なる
//--- b の新たに計算された値を出力する
   Print("----- After arithmetic operations");
   Print("Print(b)=",b);
   Print("Print(DoubleToString(b,16))=",DoubleToString(b,16));
//--- エキスパート操作ログへ出力する
// Print(b)=200.0
// Print(DoubleToString(b,16))=199.9999999999999716 (b はすでに 200.0 と等しくない)

//--- 常に小さな値を作成 epsilon=1E-013
   double epsilon=1e-13;
   Print("---- Create a very small value");
   Print("epsilon=",epsilon); // epsilon=1E-013 を取得
//--- b から epsilon を引き、値を再びエキスパート操作ログへ出力する
   b=b-epsilon;
//--- 2 つの方法を使用する
   Print("---- After subtracting epsilon from the b variable");
   Print("Print(b)=",b);
   Print("Print(DoubleToString(b,16))=",DoubleToString(b,16));
//--- エキスパート操作ログへ出力する
// Print(b)=199.9999999999999  ( epsilon を差し引いた後の b の値は 200 に丸めることが出来ない)
// Print(DoubleToString(b,16))=199.9999999999998578
//     ( epsilon を差し引いた後の b の値は 200 に丸めることが出来ない)
  }
```

参照

DoubleToString、StringFormat

© 2000-2025, MetaQuotes Ltd.

PrintFormat

事前に定義された形式に従ってシンボルと値のセットをフォーマットとし、エキスパートアドバイザーのログに入力します。

```
void  PrintFormat(
   string format_string,    // フォーマットされる文字列
   ...                      // 基本データ型の値
   );
```

パラメータ

format_string

[in] フォーマット文字列は、簡単な記号で構成され、引数が続いている場合、引数内にフォーマットの仕様が含まれています。

...

[in] コンマで区切られた任意の基本データ型の値パラメータの数はフォーマットされる文字列を含んで64を超えることは出来ません。

戻り値

文字列

注意事項

PrintFormat() 関数はストラテジーテスターでの最適化中には使用できません。

パラメータの数、順序、及び型は正確に一連の修飾子と一致しなければなりません。さもないと、プリント結果は未定義になります。PrintFormat() の代わりに printf() の使用が可能です。

フォーマット文字列にパラメータが続く場合、この文字列には、これらのパラメータの出力フォーマットを示すフォーマットの仕様が含まれている必要があります。形式の仕様は、常にパーセント記号 (%) で始まります。

フォーマット文字列は左から右に読み込まれます。最初の形式仕様 (もしあれば) が満たされると、フォーマット文字列の後にくる最初のパラメータの値は、予め設定された仕様に従って変換して出力されます。2 番目の形式仕様は変換を呼び出し2 番目のパラメータを出力し、というようにフォーマット文字列が終わるまでプロセスされます。

形式仕様の形式は次の通りです。

%[フラグ][幅][.精度][{h | l | ll | I32 | I64}]型

形式仕様の各フィールドは、単純なシンボル、または単純なフォーマットオプションを表す番号のどちらかです。最も単純な形式の仕様は、パーセント記号 (%) と出力パラメータの種類 (例えば %s) を定義するシンボルが含まれています。フォーマット文字列中にパーセント記号が必要な場合は、形式仕様 %% を使用します。

フラグ

© 2000-2025, MetaQuotes Ltd.

フラグ	説明	デフォルト動作
- (minus)	セット幅内の左寄せ。	右寄せ。
+ (plus)	符号付型の値の + または − 符号を出力します。	符号は値が負の場合のみ表示されます。
0 (zero)	ゼロは事前に定義された幅内で出力値の前に追加されます。0 フラグが整数形式で指定され（ i、u、x、X、o、d）精度仕様が設定されている場合（例えば %04.d ）、0 は無視されます。	何も追加されません。
space	符号が正の値である場合、出力値の前にスペースが示されています。	スペースは挿入されません。
#	フォーマット o、x または X と使用された場合、出力値の前に 0、0x または 0X がそれぞれ追加されます。	何も追加されません。
	フォーマット e、E、a または A と使用された場合、値は常に小数点で表示されます。	小数点はゼロ以外の小数部がある場合にのみ表示されます。
	フォーマット g または G と使用された場合、フラグは、出力値の小数点の存在を定義し、先行ゼロの削除を防ぐことが出来ます。 フラグ # はフォーマット c、d、i、u 及び s と使用された場合には無視されます。	小数点はゼロ以外の小数部がある場合にのみ表示されます。先行のゼロは削除されます。

幅

フォーマットされた値の出力記号の最小数を設定する負でない 10 進数です。出力シンボルの数が指定された幅よりも小さい場合、対応する数のスペースが、アライメントに応じて左または右から追加されます（ − フラグ）。ゼロフラグ（ 0 ）が存在する場合、対応する数のゼロが出力値の前に追加されます。出力シンボルの数が指定された幅よりも大きい場合、出力値は絶対に切られません。

アスタリスク（*）が幅として指定された場合、整数型の値が渡されたパラメータのリストの対応する場所に示されている必要があります。この値は出力値の幅を指定するのに使用されます。

精度

小数点以下の桁数として出力精度を設定する、負でない小数です 。幅仕様とは異なり、精度仕様は、丸めのあるなしにかかわらず小数型の一部を切断することが出来ます。

精度仕様の使用は、フォーマッとの種類によって異なります。

種類	説明	デフォルト桁数
a, A	精度仕様が小数点以下の桁数を設定します。	6。
c, C	使用されません。	

© 2000−2025, MetaQuotes Ltd.

種類	説明	デフォルト桁数
d, i, u, o, x, X	出力桁の最小数を設定します。対応するパラメータの桁数がこの精度よりも小さい場合、ゼロが出力値の左側に追加されます。出力桁数が指定された精度よりも大きい場合、出力値は短縮されません。	1。
e, E, f	小数点以下の出力桁数を設定します。最後の数字は四捨五入されます。	6。設定精度が 0 であるか、小数部が存在しない場合には、小数点以下は表示されません。
g, G	有効数字の最大数を設定します。	6。
s	文字列の出力シンボルの数を設定します。文字列の長さが精度を超えた場合、文字列が短くされます。	全体の文字列が出力されます。

```
PrintFormat("1. %s", _Symbol);
PrintFormat("2. %.3s", _Symbol);
int length=4;
PrintFormat("3. %.*s", length, _Symbol);
/*
1. EURUSD
2. EUR
3. EURU
/
```

h | l | ll | I32 | I64

パラメータとして渡されたデータサイズの指定を指定します。

パラメータ型	使用するプレフィックス	型のジョイント指定子
int	l（小文字のL）	d、i、o、x または X
uint	l（小文字のL）	o、u、x または X
long	ll（小文字の L が二つ）<	d、i、o、x または X
short	h	d、i、o、x または X
ushort	h	o、u、x または X
int	I32	d、i、o、x または X
uint	I32	o、u、x または X
long	I64	d、i、o、x または X
ulong	I64	o、u、x または X

© 2000–2025, MetaQuotes Ltd.

型

型指定子は、フォーマットされた出力のための唯一の必須フィールドです。

Symbol	Type	出力フォーマット
c	int	short 型のシンボル（Unicode）。
C	int	char 型のシンボル（ANSI）。
d	int	符号付き 10 進整数。
i	int	符号付き 10 進整数。
o	int	符号なし 8 進整数。
u	int	符号なし 10 進整数。
x	int	「abcdef」を使用した符号なし 16 進整数。
X	int	「ABCDEF」を使用した符号なし 16 進整数。
e	double	[-] d.ddde[sign] ddd のフォーマットを持った実数で、d は 1 つの 10 進整数、dddd は 1 つ以上の 10 進整数で ddd は 3 桁の数で、指数の大きさとプラスマイナスのサインを決めます。
E	double	指数の符号が大文字で出力されている以外は e のフォーマットと似ています（ e の代わりに**E** ）。
f	double	[-] dddd.dddd のフォーマットを持った実数。ここで dddd は 1 つ以上の 10 進整数です。小数点の前の表示桁数は、数値のサイズによって異なります。小数点以下の桁数は、必要な精度に依存します。
g	double	どちらの表記がよりコンパクトになるかに応じて **f** または **e** フォーマットで出力された実数。
G	double	どちらの表記がよりコンパクトになるかに応じて **F** または **E** フォーマットで出力された実数。
a	double	[-]0xh.hhhh **p**±dd フォーマットを持った実数で、h.hhhh が「abcdef」を使用した 16 進数の仮数部で、dd が1桁以上の指数。小数点以下の桁数は、精度仕様によって決定されます。
A	double	[-]0xh.hhhh **P**±dd フォーマットを持った実数で、h.hhhh が「abcdef」を使用した 16 進数の仮数部で、dd が1桁以上の指数。小数点以下の桁数は、精度仕様によって決定されます。
s	string	文字列出力

PrintFormat() の代わりに printf() の使用が可能です。

例：

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//--- 取引サーバ名
   string server=AccountInfoString(ACCOUNT_SERVER);
//--- 口座番号
   int login=(int)AccountInfoInteger(ACCOUNT_LOGIN);
//--- long 値の出力
   long leverage=AccountInfoInteger(ACCOUNT_LEVERAGE);
   PrintFormat("%s %d: leverage = 1:%I64d",
               server,login,leverage);
//--- 口座通貨
   string currency=AccountInfoString(ACCOUNT_CURRENCY);
//--- 小数点以下 2 桁の double 値の出力
   double equity=AccountInfoDouble(ACCOUNT_EQUITY);
   PrintFormat("%s %d: account equity = %.2f %s",
               server,login,equity,currency);
//--- +/- 符号の出力を必須とした double 値の出力
   double profit=AccountInfoDouble(ACCOUNT_PROFIT);
   PrintFormat("%s %d: current result for open positions = %+.2f %s",
               server,login,profit,currency);
//--- 小数点以下の桁数が可変の double 値の出力
   double point_value=SymbolInfoDouble(_Symbol,SYMBOL_POINT);
   string format_string=StringFormat("%%s: point value  = %%.%df",_Digits);
   PrintFormat(format_string,_Symbol,point_value);
//--- 整数値の出力
   int spread=(int)SymbolInfoInteger(_Symbol,SYMBOL_SPREAD);
   PrintFormat("%s: current spread in points = %d ",
               _Symbol,spread);
//--- 小数点以下 17 桁の科学(浮動小数点)形式の double 値の出力
   PrintFormat("DBL_MAX = %.17e",DBL_MAX);
//--- 小数点以下 17 桁の科学(浮動小数点)形式の double 値の出力
   PrintFormat("EMPTY_VALUE = %.17e",EMPTY_VALUE);
//--- デフォルトの精度で PrintFormat() を使用して出力
   PrintFormat("PrintFormat(EMPTY_VALUE) = %e",EMPTY_VALUE);
//--- Print() を使用した簡単な出力
   Print("Print(EMPTY_VALUE) = ",EMPTY_VALUE);
/* execution result
   MetaQuotes-Demo 1889998: leverage = 1:100
   MetaQuotes-Demo 1889998: account equity = 22139.86 USD
   MetaQuotes-Demo 1889998: current result for open positions = +174.00 USD
   EURUSD: point value  = 0.00001
   EURUSD: current spread in points = 12
   DBL_MAX = 1.79769313486231570e+308
   EMPTY_VALUE = 1.79769313486231570e+308
   PrintFormat(EMPTY_VALUE) = 1.797693e+308
   Print(EMPTY_VALUE) = 1.797693134862316e+308
*/
  }
```

参照

StringFormat、DoubleToString、浮動小数点数型（ダブル、フロート）

© 2000-2025, MetaQuotes Ltd.

ResetLastError

所定の変数 _LastErrorの値をゼロに設定します。

```
void  ResetLastError();
```

戻り値

なし

注意事項

GetLastError() は _LastError 変数をゼロにしないことには留意すべきです。通常、関数呼び出しの前に ResetLastError() が呼び出され エラー発生がチェックされます。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 関数を呼び出す前に最後のエラーコードをリセットする
//--- さもないと、GetLastError() は前のエラーコードを返す可能性がある
   long lres=SymbolInfoInteger("123456",SYMBOL_DIGITS);
   PrintFormat("lres=%d  error=%d",lres,GetLastError());
   lres=SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);
   PrintFormat("lres=%d  error=%d",lres,GetLastError());
   ResetLastError();
   lres=SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);
   PrintFormat("lres=%d  error=%d",lres,GetLastError());
  }
```

© 2000-2025, MetaQuotes Ltd.

ResourceCreate

データセットに基づいて画像のリソースを作成します。この関数には 2 つのバージョンがあります。
ファイルに基づいてリソースの作成

```
bool  ResourceCreate(
   const string      resource_name,       // リソース名
   const string      path                 // ファイルへの相対パス
   );
```

画素の配列に基づいたリソースの作成

```
bool  ResourceCreate(
   const string      resource_name,       // リソース名
   const uint&       data[],              // 配列としてのデータセット
   uint              img_width,           // 画像リソースの幅
   uint              img_height,          // 画像リソースの高さ
   uint              data_xoffset,        // 画像の左上隅の水平右方向オフセット
   uint              data_yoffset,        // 画像の左上隅の鉛直オフセット
   uint              data_width,          // データセットに基づいた画像の全体の幅
   ENUM_COLOR_FORMAT color_format         // 色処理方法
   );
```

パラメータ

resource_name

[in] リソース名

data[][]

。[in] 完全な画像を作成するための 1 次元または 2 次元の配列。

img_width

[in] 画像としてリソースに配置される矩形の画像領域のピクセルで表された幅。*data_width* 値を超すことは出来ません。

img_height

[in] 画像としてリソースに配置される矩形の画像領域の高さ(ピクセル単位)。

data_xoffset

[in] 画像の矩形領域の水平右方向オフセット。

data_yoffset

[in] 画像の矩形領域の縦の下方のオフセット。

data_width

[in] 1 次元配列のみで必要。データセットから画像の全幅を表します。*data_width*=0 の場合、値が *img_width* と等しいとみなされます。2 次元配列の場合、パラメータは無視され、*data[]* 配列の第 2 次元に等しいと仮定されます。

color_format

[in] 色処理方法で ENUM_COLOR_FORMAT 列挙から選ばれます。

© 2000-2025, MetaQuotes Ltd.

戻り値

成功の場合は true、それ以外の場合は false 。エラー情報を取得するには、GetLastError() 関数が呼ばれます。次のエラーが発生可能です。

- 4015 – ERR_RESOURCE_NAME_DUPLICATED (同名の動的リソースと静的リソース)
- 4016 – ERR_RESOURCE_NOT_FOUND (リソースが見つからない)
- 4017 – ERR_RESOURCE_UNSUPPORTED_TYPE (この種類のリソースのサポートが不在)
- 4018 – ERR_RESOURCE_NAME_IS_TOO_LONG (リソース名が長すぎる)

注意事項

関数の 2 番目のバージョンが、異なる幅、高さ、及びシフトパラメータを使用して同じリソースを作成するために呼び出された場合は、新しいリソースは作成されず、単に既存のものが更新されます。

関数の最初のバージョンは、ファイルから画像や音声をアップロードするために使用され、2 番目のバージョンは、画像を動的に作成するために使用されます。

画像は24ビットまたは 32 ビットの色深度を持つ BMP 形式でなければなりません。サポートされているサウンド形式は WAV のみです。リソースのサイズは、16 MBを超えてはなりません。

ENUM_COLOR_FORMAT

識別子	説明
COLOR_FORMAT_XRGB_NOALPHA	アルファチャンネルの成分は無視されます。
COLOR_FORMAT_ARGB_RAW	色成分は端末によって処理されません(ユーザが正しく設定する必要があります)。
COLOR_FORMAT_ARGB_NORMALIZE	色成分は端末によって処理されます。

例：

```
//+------------------------------------------------------------------+
//|  グラフィックリソースデータを更新する                                |
//+------------------------------------------------------------------+
void Update(const string res_name,const uint &pixel_data[],const uint width,const uint height,const bool redraw)
  {
//--- ゼロ次元が渡された場合は終了する
   if(width==0 || height==0)
      return;
//--- リソース データを更新し、チャートを再描画する
   if(ResourceCreate(res_name,pixel_data,width,height,0,0,0,COLOR_FORMAT_ARGB_NORMALIZE) && redraw)
      ChartRedraw();
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                       |
//+------------------------------------------------------------------+
```

© 2000–2025, MetaQuotes Ltd.

```
void OnStart()
  {
//--- グラフィックリソースのパラメータを宣言する
   string rc_name="Resource";
   uint   rc_width=100;
   uint   rc_height=100;
   uint   rc_data[];
   uint   rc_size=rc_width*rc_height;

   ResetLastError();
//--- ピクセル配列のサイズを設定する
   if(ArrayResize(rc_data,rc_size)!=rc_size)
     {
      Print("ArrayResize() failed. Error code: ",GetLastError());
      return;
     }
//--- ピクセル配列を透明色で塗りつぶし、それに基づいてグラフィック リソースを作成する
   ArrayInitialize(rc_data,0x00FFFFFF);
   if(!ResourceCreate
(rc_name,rc_data,rc_width,rc_height,0,0,0,COLOR_FORMAT_ARGB_NORMALIZE))
     {
      Print("ResourceCreate() failed. Error code: ",GetLastError());
      return;
     }
   Print("Size of created recource array: ",rc_data.Size());

//--- 作成されたグラフィックリソースを確認する
//--- 現在の足の時間と価格のデータを取得する
   MqlTick tick={};
   if(!SymbolInfoTick(Symbol(),tick))
     {
      Print("SymbolInfoTick() failed. Error code: ",GetLastError());
      return;
     }
//--- 最後のティック価格と時刻の座標を使用してBitmapオブジェクトを作成する
   string obj_name="Bitmap";
   if(!ObjectCreate(0,obj_name,OBJ_BITMAP,0,tick.time,tick.bid))
     {
      Print("ObjectCreate() failed. Error code: ",GetLastError());
      return;
     }
//--- 作成されたビットマップ オブジェクトの幅と高さを、グラフィックリソースの幅と高さに等しく設定する
//--- オブジェクトのアンカーポイントをその中心に設定する
   ObjectSetInteger(0,obj_name,OBJPROP_XSIZE,rc_width);
   ObjectSetInteger(0,obj_name,OBJPROP_YSIZE,rc_height);
   ObjectSetInteger(0,obj_name,OBJPROP_ANCHOR,ANCHOR_CENTER);
//--- ビットマップオブジェクト用に以前に作成したグラフィックリソースを画像ファイルとして指定する
//--- この場合、使用されるグラフィックリソースの名前を示すために、その名前の前に「::」を追加する必要がある
   ObjectSetString(0,obj_name,OBJPROP_BMPFILE,"::"+rc_name);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 透明度200でDodger Blue色を設定する
   uint clr=ColorToARGB(clrDodgerBlue,200);
//--- グラフィックリソースのピクセル配列全体を設定された色で塗りつぶす
   ArrayInitialize(rc_data,clr);
//--- グラフィックリソースデータを更新する
   Update(rc_name,rc_data,rc_width,rc_height,true);
  }
```

参照

リソース、ObjectCreate()、ObjectSetString()、OBJPROP_BMPFILE

© 2000-2025, MetaQuotes Ltd.

ResourceFree

この関数は動的に作成されたリソースを削除します（割り当てられたメモリを解放します）。

```
bool  ResourceFree(
   const string  resource_name      // リソース名
   );
```

パラメータ

resource_name

[in] リソース名は "::" で始まるべきです。

戻り値

成功の場合は true、それ以外の場合は false。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

注意事項

ResourceFree() は、MQL5アプリケーション開発者がリソースを操作する場合のメモリ消費量の管理を可能にします。メモリから削除されているリソースに結合されたグラフィカルオブジェクトは、削除後でも正常に表示されます。しかし、新しく作成されたグラフィカルオブジェクト（OBJ_BITMAP と OBJ_BITMAP_LABEL）は削除されたリソースを使用することが出来ません。

この関数は、プログラムによって作成された動的リソースのみを削除します。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- グラフィックリソースのパラメータを宣言する
   string rc_name="Resource";
   uint   rc_width=100;
   uint   rc_height=100;
   uint   rc_data[];
   uint   rc_size=rc_width*rc_height;

   ResetLastError();
//--- ピクセル配列のサイズを設定する
   if(ArrayResize(rc_data,rc_size)!=rc_size)
     {
      Print("ArrayResize() failed. Error code: ",GetLastError());

     }
//--- ピクセル配列を透明色で塗りつぶし、それに基づいてグラフィック リソースを作成する
   ArrayInitialize(rc_data,0x00FFFFFF);
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ResourceCreate
(rc_name,rc_data,rc_width,rc_height,0,0,0,COLOR_FORMAT_ARGB_NORMALIZE))
     {
      Print("ResourceCreate() failed. Error code: ",GetLastError());

     }
   Print("Size of created recource array: ",rc_data.Size());

//--- 作成されたグラフィックリソースを確認する
//--- 現在の足の時間と価格のデータを取得する
   MqlTick tick={};
   if(!SymbolInfoTick(Symbol(),tick))
     {
      Print("SymbolInfoTick() failed. Error code: ",GetLastError());

     }
//--- 最後のティック価格と時刻の座標を使用してBitmapオブジェクトを作成する
   string obj_name="Bitmap";
   if(!ObjectCreate(0,obj_name,OBJ_BITMAP,0,tick.time,tick.bid))
     {
      Print("ObjectCreate() failed. Error code: ",GetLastError());

     }
//--- 作成されたビットマップ オブジェクトの幅と高さを、グラフィックリソースの幅と高さに等しく設定する
//--- オブジェクトのアンカーポイントをその中心に設定する
   ObjectSetInteger(0,obj_name,OBJPROP_XSIZE,rc_width);
   ObjectSetInteger(0,obj_name,OBJPROP_YSIZE,rc_height);
   ObjectSetInteger(0,obj_name,OBJPROP_ANCHOR,ANCHOR_CENTER);
//--- ビットマップオブジェクト用に以前に作成したグラフィックリソースを画像ファイルとして指定する
//--- この場合、使用されるグラフィックリソースの名前を示すために、その名前の前に「::」を追加する必要がある
   ObjectSetString(0,obj_name,OBJPROP_BMPFILE,"::"+rc_name);

//--- 透明度200でDodgerBlue色を設定する
   uint clr=ColorToARGB(clrDodgerBlue,200);
//--- グラフィックリソースのピクセル配列全体を設定された色で塗りつぶす
   ArrayInitialize(rc_data,clr);
//--- グラフィックリソースデータを更新する
   Update(rc_name,rc_data,rc_width,rc_height,true);

//--- 3秒待って画像の色を変更する
   Print("Wait 3 seconds before changing color");
   Sleep(3000);
//--- 透明度200でOrangeRed色を設定する
   Print("Change color");
   clr=ColorToARGB(clrOrangeRed,200);
//--- グラフィックリソースのピクセル配列全体を設定された色で塗りつぶす
   ArrayInitialize(rc_data,clr);
//--- グラフィックリソースデータを更新する
   Update(rc_name,rc_data,rc_width,rc_height,true);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 3秒待ってグラフィックリソースを解放する
   Print("Wait 3 seconds before ResourceFree()");
   Sleep(3000);
   bool res=ResourceFree("::"+rc_name);
   Print("ResourceFree: ",res);

//--- リソースを解放した後に色を変更してみる
   Print("Trying to change color to GreenYellow after ResourceFree()");
//--- 透明度200でGreenYellow色を設定する
   clr=ColorToARGB(clrGreenYellow,200);
//--- グラフィックリソースのピクセル配列全体を設定された色で塗りつぶす
   ArrayInitialize(rc_data,clr);
//--- グラフィックリソース データを更新する(画像は残るが、色は変更できない)
   Update(rc_name,rc_data,rc_width,rc_height,true);
   Print("The color has not changed because the resource has been released");

//--- 3秒待ってbitmapオブジェクトを削除する
   Print("Wait 3 seconds before deleting the Bitmap object");
   Sleep(3000);
   Print("Delete Bitmap object");
   ObjectDelete(0,obj_name);
  }
//+------------------------------------------------------------------+
//| グラフィックリソースデータを更新する                                |
//+------------------------------------------------------------------+
void Update(const string res_name,const uint &pixel_data[],const uint width,const
uint height,const bool redraw)
  {
//--- ゼロ次元が渡された場合は終了する
   if(width==0 || height==0)

//--- リソース データを更新し、チャートを再描画する
   if
(ResourceCreate(res_name,pixel_data,width,height,0,0,0,COLOR_FORMAT_ARGB_NORMALIZE)
&& redraw)
      ChartRedraw();
  }
```

参照

リソース、ObjectCreate()、PlaySound()、ObjectSetString()、OBJPROP_BMPFILE

© 2000-2025, MetaQuotes Ltd.

ResourceReadImage

ResourceCreate() 関数で作成されたまたはコンパイル中に EX5 ファイルに保存されたグラフィックリソースからデータを読みます。

```
bool  ResourceReadImage(
   const string      resource_name,        // 読まれるグラフィックリソース名
   uint&             data[],               // リソースからデータを受け取るための配列
   uint&             width,                // リソースの画像幅を受け取る
   uint&             height,               // リソースの画像の高さを受け取る
   );
```

パラメータ

resource_name

[in] 画像を含むグラフィクリソースの名称。自身のリソースにアクセスするには、「::ResourceName」が名称として使用されます。コンパイルされた EX5 ファイルからリソースをダウンロードする場合、完全な名称は「path¥¥filename.ex5::resourcename」のようにMQL5ディレクトリ、ファイル名とリソース名の相対パスが使用されるべきです 。

data[][]

[in] グラフィックリソースからデータを受け取るための 1 次元または 2 次元の配列。

img_width

[out] グラフィックリソース画像の幅（ピクセル単位）。

img_height

[out] グラフィックリソース画像の高さ（ピクセル単位）。

戻り値

成功の場合は true、それ以外の場合は false 。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

注意事項

data[] 配列がグラフィックリソースの作成に使用される場合， _COLOR_FORMAT_ARGB_NORMALIZE または COLOR_FORMAT_XRGB_NOALPHA カラーフォーマットが使用されるべきです。

data[] 配列が 2 次元で、2 番目の次元が、グラフィカルリソースサイズのX（幅）より小さい場合、ResourceReadImage() 関数はfalseを返し、読み込みが行われません。しかし、もしリソースが存在すれば、実際の画像サイズが幅と高さのパラメータに返されます。これで、リソースからのデータ受けとりをもう一度試むことが可能になります。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- グラフィックリソースのパラメータを宣言する
   string rc_name="Resource";
   uint   rc_width=100;
   uint   rc_height=100;
   uint   rc_data[];
   uint   rc_size=rc_width*rc_height;

   ResetLastError();
//--- ピクセル配列のサイズを設定する
   if(ArrayResize(rc_data,rc_size)!=rc_size)
     {
      Print("ArrayResize() failed. Error code: ",GetLastError());
      return;
     }
//--- ピクセル配列を透明色で塗りつぶし、それに基づいてグラフィック リソースを作成する
   ArrayInitialize(rc_data,0x00FFFFFF);
   if(!ResourceCreate
(rc_name,rc_data,rc_width,rc_height,0,0,0,COLOR_FORMAT_ARGB_NORMALIZE))
     {
      Print("ResourceCreate() failed. Error code: ",GetLastError());
      return;
     }
   Print("Size of created recource array: ",rc_data.Size());

//--- 作成されたグラフィックリソースを確認する
//--- 現在の足の時間と価格のデータを取得する
   MqlTick tick={};
   if(!SymbolInfoTick(Symbol(),tick))
     {
      Print("SymbolInfoTick() failed. Error code: ",GetLastError());
      return;
     }
//--- 最後のティック価格と時刻の座標を使用してBitmapオブジェクトを作成する
   string obj_name="Bitmap";
   if(!ObjectCreate(0,obj_name,OBJ_BITMAP,0,tick.time,tick.bid))
     {
      Print("ObjectCreate() failed. Error code: ",GetLastError());
      return;
     }
//--- 作成されたビットマップ オブジェクトの幅と高さを、グラフィックリソースの幅と高さに等しく設定する
//--- オブジェクトのアンカーポイントをその中心に設定する
   ObjectSetInteger(0,obj_name,OBJPROP_XSIZE,rc_width);
   ObjectSetInteger(0,obj_name,OBJPROP_YSIZE,rc_height);
   ObjectSetInteger(0,obj_name,OBJPROP_ANCHOR,ANCHOR_CENTER);
//--- ビットマップオブジェクト用に以前に作成したグラフィックリソースを画像ファイルとして指定する
//--- この場合、使用されるグラフィックリソースの名前を示すために、その名前の前に「::」を追加する必要がある
   ObjectSetString(0,obj_name,OBJPROP_BMPFILE,"::"+rc_name);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 透明度200でDodger Bl ue色を設定する
   uint clr=ColorToARGB(clrDodgerBlue,200);
//--- グラフィックリソースのピクセル配列全体を設定された色で塗りつぶす
   ArrayInitialize(rc_data,clr);
//--- グラフィックリソースデータを更新する
   Update(rc_name,rc_data,rc_width,rc_height,true);

//--- グラフィカルリソースデータを読み取る前に3秒待機する
   Print("Wait 3 seconds before ResourceReadImage()");
   Sleep(3000);
//--- 画像をリソースから新しいピクセル配列に読み込む
   uint rc_data_copy[];
   uint w=0,h=0;
   ResetLastError();
   if(!ResourceReadImage("::"+rc_name,rc_data_copy,w,h))
     {
      Print("ResourceReadImage() failed. Error code: ",GetLastError());
      return;
     }

//--- 透明度200でOr angeRed色を設定する
   clr=ColorToARGB(clrOrangeRed,200);
//--- グラフィックリソースのピクセル配列全体を設定された色で塗りつぶし、それに基づいて新しいグラフィックリソースを作成する
   ArrayInitialize(rc_data_copy,clr);
   if(!ResourceCreate
(rc_name+"Copy",rc_data_copy,rc_width,rc_height,0,0,0,COLOR_FORMAT_ARGB_NORMALIZE))
     {
      Print("New ResourceCreate() failed. Error code: ",GetLastError());
      return;
     }
   Print("Size of created new recource array: ",rc_data_copy.Size());

//--- 最後のティック価格と時刻の座標を使用して「グラフィックラベル」オブジェクトを作成する
   string obj_name2="BitmapLabel";
   if(!ObjectCreate(0,obj_name2,OBJ_BITMAP_LABEL,0,0,0))
     {
      Print("ObjectCreate() failed. Error code: ",GetLastError());
      return;
     }
//--- 以前に受け取った価格と時間を使用して画面座標を取得する
   int x=0,y=0;
   if(!ChartTimePriceToXY(0,0,tick.time,tick.bid,x,y))
     {
      Print("New ChartTimePriceToXY() failed. Error code: ",GetLastError());
      return;
     }
//--- 作成されたグラフィカルラベルオブジェクトの幅と高さを、グラフィカルリソースの幅と高さに等しく設定する
//--- オブジェクトのアンカーポイントをその中心に設定する
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(0,obj_name2,OBJPROP_XSIZE,rc_width);
   ObjectSetInteger(0,obj_name2,OBJPROP_YSIZE,rc_height);
   ObjectSetInteger(0,obj_name2,OBJPROP_ANCHOR,ANCHOR_LEFT_UPPER);
   ObjectSetInteger(0,obj_name2,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(0,obj_name2,OBJPROP_YDISTANCE,y);
//--- コピーしたグラフィックリソースをグラフィックラベルオブジェクトの画像ファイルとして設定する
//--- この場合、使用されるグラフィックリソースの名前を示すために、その名前の前に「::」を追加する必要がある
   ObjectSetString(0,obj_name2,OBJPROP_BMPFILE,"::"+rc_name+"Copy");

//--- 新しいグラフィカルラベルオブジェクトの色を変更する
   Print("Wait 3 seconds before changing color to GreenYellow");
   Sleep(3000);
//--- 透明度200でGreenYellow色を設定する
   clr=ColorToARGB(clrGreenYellow,200);
//--- 新しいグラフィックリソースのピクセル配列全体を設定された色で塗りつぶす
   ArrayInitialize(rc_data_copy,clr);
//--- グラフィックリソースデータを更新する
   Update(rc_name+"Copy",rc_data_copy,rc_width,rc_height,true);

//--- 3秒待ってリソースと両方のオブジェクトを削除する
   Print("Wait 3 seconds before deleting both objects");
   Sleep(3000);
   Print("Deleting Resource and all Bitmap objects");
   ResourceFree("::"+rc_name);
   ResourceFree("::"+rc_name+"Copy");
   ObjectDelete(0,obj_name);
   ObjectDelete(0,obj_name2);
  }
//+------------------------------------------------------------------+
//| グラフィックリソースデータを更新する                                |
//+------------------------------------------------------------------+
void Update(const string res_name,const uint &pixel_data[],const uint width,const uint height,const bool redraw)
  {
//--- ゼロ次元が渡された場合は終了する
   if(width==0 || height==0)
      return;
//--- リソース データを更新し、チャートを再描画する
   if(ResourceCreate(res_name,pixel_data,width,height,0,0,0,COLOR_FORMAT_ARGB_NORMALIZE) && redraw)
      ChartRedraw();
  }
```

参照

リソース、ObjectCreate()、ObjectSetString()、OBJPROP_BMPFILE

© 2000-2025, MetaQuotes Ltd.

ResourceSave

リソースを指定されたファイルに保存します。

```
bool  ResourceSave(
   const string  resource_name      // リソース名
   const string  file_name          // ファイル名
   );
```

パラメータ

resource_name

[in]「::」から始まるリソース名

file_name

[in] MQL5¥Files からのファイルの相対名

戻り値

成功の場合は true、それ以外の場合は fals。エラー情報を取得するには、GetLastError() が呼ばれます。

注意事項

この関数は常にファイルを上書きし、必要に応じてファイル名に必要な全ての中間ディレクトリを作成します。

例：

```
//--- グラフィックリソースパラメータ
string ExtResName="Resource";
int    ExtResWidth=100;
int    ExtResHeight=100;
uint   ExtResData[];
int    ExtResSize=ExtResWidth*ExtResHeight;

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   ResetLastError();
//--- ピクセル配列のサイズを設定する
   if(ArrayResize(ExtResData,ExtResSize)!=ExtResSize)
     {
      Print("ArrayResize() failed. Error code: ",GetLastError());
      return;
     }
//--- ピクセル配列を透明色で塗りつぶし、それに基づいてグラフィック リソースを作成する
   ArrayInitialize(ExtResData,0x00FFFFFF);
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ResourceCreate
(ExtResName,ExtResData,ExtResWidth,ExtResHeight,0,0,0,COLOR_FORMAT_ARGB_NORMALIZE))
     {
      Print("ResourceCreate() failed. Error code: ",GetLastError());
      return;
     }
   Print("Size of created recource array: ",ExtResData.Size());

//--- 作成されたグラフィックリソースを確認する
//--- 現在の足の時間と価格のデータを取得する
   MqlTick tick={};
   if(!SymbolInfoTick(Symbol(),tick))
     {
      Print("SymbolInfoTick() failed. Error code: ",GetLastError());
      return;
     }
//--- 最後のティック価格と時刻の座標を使用してBitmapオブジェクトを作成する
   string obj_name="Bitmap";
   if(!ObjectCreate(0,obj_name,OBJ_BITMAP,0,tick.time,tick.bid))
     {
      Print("ObjectCreate() failed. Error code: ",GetLastError());
      return;
     }
//--- 作成されたビットマップ オブジェクトの幅と高さを、グラフィックリソースの幅と高さに等しく設定する
//--- オブジェクトのアンカーポイントをその中心に設定する
   ObjectSetInteger(0,obj_name,OBJPROP_XSIZE,ExtResWidth);
   ObjectSetInteger(0,obj_name,OBJPROP_YSIZE,ExtResHeight);
   ObjectSetInteger(0,obj_name,OBJPROP_ANCHOR,ANCHOR_CENTER);
//--- ビットマップオブジェクト用に以前に作成したグラフィックリソースを画像ファイルとして指定する
//--- この場合、使用されるグラフィックリソースの名前を示すために、その名前の前に"::"を追加する必要がある
   ObjectSetString(0,obj_name,OBJPROP_BMPFILE,"::"+ExtResName);

//--- 透明度200でGreenYellow色を設定する
   uint clr=ColorToARGB(clrGreenYellow,200);
//--- グラフィックリソースのピクセル配列全体を設定された色で塗りつぶす
   ArrayInitialize(ExtResData,clr);
//--- DodgerBlue色を使用してグリッドを描画する
   for(int x=0;x<ExtResWidth;x+=9)
      LineVertical(x,0,ExtResHeight,ColorToARGB(clrDodgerBlue,200));
   for(int y=0;y<ExtResHeight;y+=9)
      LineHorizontal(0,ExtResWidth,y,ColorToARGB(clrDodgerBlue,200));
//--- グラフィックリソースデータを更新する
   Update(ExtResName,ExtResData,ExtResWidth,ExtResHeight,true);

//--- リソースとグラフィックオブジェクトを削除する前に3秒待つ
   Print("Wait 3 seconds before deleting the resource and graphic object");
   Sleep(3000);

//--- リソースをファイルに保存する
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
   if(!ResourceSave("::"+ExtResName,"ResourceSave\\"+ExtResName+".bmp"))
     {
      Print("ResourceSave() failed. Error code: ",GetLastError());
      return;
     }
//--- リソースとグラフィックオブジェクト画像を削除する
   if(!ResourceFree("::"+ExtResName))
     {
      Print("ResourceFree() failed. Error code: ",GetLastError());
      return;
     }
   if(!ObjectDelete(0,obj_name))
     {
      Print("ObjectDelete() failed. Error code: ",GetLastError());
      return;
     }

//--- ここで、画像の幅だけ左にシフトされた最後のティックの価格と時刻の座標を使用して、「グラフィックラベル」オブジェクトを作成する
//--- 以前に受信した最後のティック価格と時刻を使用して画面座標を取得する
   int x=0,y=0;
   if(!ChartTimePriceToXY(0,0,tick.time,tick.bid,x,y))
     {
      Print("ChartTimePriceToXY() failed. Error code: ",GetLastError());
      return;
     }
   obj_name="BitmapLabel";
   if(!ObjectCreate(0,obj_name,OBJ_BITMAP_LABEL,0,0,0))
     {
      Print("ObjectCreate() failed. Error code: ",GetLastError());
      return;
     }
//--- オブジェクトのアンカーポイントをその中心に設定する
   ObjectSetInteger(0,obj_name,OBJPROP_ANCHOR,ANCHOR_CENTER);
//--- 最後のティックの価格と時刻の座標を画像の幅だけ左にシフトして、グラフィカルラベルオブジェクトに設定する
   ObjectSetInteger(0,obj_name,OBJPROP_XDISTANCE,x-ExtResWidth);
   ObjectSetInteger(0,obj_name,OBJPROP_YDISTANCE,y);
//--- 以前にbmp画像として保存した、グラフィックラベルオブジェクトのリソースを画像ファイルとして指定する
   ObjectSetString(0,obj_name,OBJPROP_BMPFILE,"\\Files\\ResourceSave\
\"+ExtResName+".bmp");
//--- チャートを更新する
   ChartRedraw();

//--- グラフィックラベルオブジェクトを削除する前に3秒待つ
   Print("Wait 3 seconds before deleting the new graphic object");
   Sleep(3000);
   if(!ObjectDelete(0,obj_name))
      Print("ObjectDelete() failed. Error code: ",GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| グラフィックリソースデータを更新する                                |
//+------------------------------------------------------------------+
void Update(const string res_name,const uint &pixel_data[],const uint width,const
uint height,const bool redraw)
  {
//--- ゼロ次元が渡された場合は終了する
   if(width==0 || height==0)
      return;
//--- リソース データを更新し、チャートを再描画する
   if
(ResourceCreate(res_name,pixel_data,width,height,0,0,0,COLOR_FORMAT_ARGB_NORMALIZE)
&& redraw)
      ChartRedraw();
  }
//+------------------------------------------------------------------+
//| 垂直線を描画する                                                 |
//+------------------------------------------------------------------+
void LineVertical(int x,int y1,int y2,const uint clr)
  {
   int tmp;
//--- Yによって並び替える
   if(y1>y2)
     {
      tmp=y1;
      y1 =y2;
      y2 =tmp;
     }
//--- 画像境界の外側の線
   if(y2<0 || y1>=ExtResHeight || x<0 || x>=ExtResWidth)
      return;
//--- 画像の境界内にとどまる
   if(y1<0)
      y1=0;
   if(y2>=ExtResHeight)
      y2=ExtResHeight-1;
//--- 線を引く
   int index=y1*ExtResWidth+x;
   for(int i=y1; i<=y2; i++,index+=ExtResWidth)
      ExtResData[index]=clr;
  }
//+------------------------------------------------------------------+
//| 水平線を描画する                                                 |
//+------------------------------------------------------------------+
void LineHorizontal(int x1,int x2,int y,const uint clr)
  {
   int tmp;
//--- Xによって並び替える
```

© 2000-2025, MetaQuotes Ltd.

```
   if(x1>x2)
     {
      tmp=x1;
      x1 =x2;
      x2 =tmp;
     }
//--- 画像境界の外側の線
   if(x2<0 || x1>=ExtResWidth || y<0 || y>=ExtResHeight)
      return;
//--- 画像の境界内にとどまる
   if(x1<0)
      x1=0;
   if(x2>=ExtResWidth)
      x2=ExtResWidth-1;
//--- 線を引く
   ArrayFill(ExtResData,y*ExtResWidth+x1,(x2-x1)+1,clr);
  }
```

参照

リソース、ObjectCreate()、PlaySound()、ObjectSetString()、OBJPROP_BMPFILE

© 2000-2025, MetaQuotes Ltd.

SetReturnError

操作終了時に端末プロセスを返すコードを設定します。

```
void  SetReturnError(
   int ret_code      // クライアント端末完了コード
   );
```

パラメータ

ret_code

[in] 操作完了にクライアント端末が返すコード

戻り値

なし

注意事項

SetReturnError()関数を使用して指定された*ret_code*戻りコードを設定すると、コマンドラインから端末を起動する際にプログラム操作完了の理由を分析するのに有用です。

TerminalClose()と異なり、端末操作はSetReturnError()関数によっては完了されません。代わりに、完了時に端末プロセスを返すコードを設定するだけです。

SetReturnError()関数が複数回呼び出されたり、異なるMQL5プログラムから呼び出されたりすると、最後に設定された戻りコードが返されます。

設定されたコードは、以下の場合を除いて、端末プロセスの完了時に返されます。

- 実行中に重大エラーが発生した -
- TerminalClose(int ret_code)関数が呼び出されて、指定されたコードで端末操作完了コマンドを発行した。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   matrix matrix_a =
     {
        {-3.474589, 1.106384, -9.091977,-3.925227 },
        {-5.522139, 2.366887,-15.162351,-6.357512 },
        { 8.394926,-2.960067, 22.292115, 9.524129 },
        { 7.803242,-2.080287, 19.217706, 8.186645 }
     };
   matrix matrix_l(4,4);
   matrix matrix_u(4,4);

//--- LU分解
   matrix_a.LU(matrix_l,matrix_u);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- A = L * Uかどうかを確認する
   matrix matrix_lu=matrix_l.MatMul(matrix_u);
   int    compare_errors=(int)matrix_a.Compare(matrix_lu,1e-29);
   Print("MatrixCompare errors=",compare_errors);

//--- 完了すると、クライアント端末は2つの行列の比較におけるエラーの数を返す
   SetReturnError(compare_errors);
  }
```

参照

プログラム実行、実行エラー、初期化解除の理由、TerminalClose

© 2000-2025, MetaQuotes Ltd.

SetUserError

事前に定義された変数 _LastError を ERR_USER_ERROR_FIRST + user_error に等しい値に設定します。

```
void  SetUserError(
   ushort user_error,    // エラー番号
   );
```

パラメータ

user_error

[in] ユーザが設定したエラー番号

戻り値

なし

注意事項

SetUserError(user_error) 関数でエラーが設定された後、GetLastError() は ERR_USER_ERROR_FIRST + user_error に等しい値を返します。

例:

```
void OnStart()
  {
//--- エラー番号 65537=(ERR_USER_ERROR_FIRST +1) を設定
   SetUserError(1);
//--- 最後のエラーコードを取得する
   Print("GetLastError = ",GetLastError());
/*
  結果
   GetLastError = 65537
*/
  }
```

© 2000-2025, MetaQuotes Ltd.

Sleep

指定された間隔内で現在のエキスパートアドバイザーまたはスクリプトの実行を一時停止します。

```
void  Sleep(
   int  milliseconds      // 間隔
   );
```

パラメータ

milliseconds

[in] ミリ秒単位の遅延間隔

戻り値

なし

注意事項

指標はインターフェースのスレッドで実行され遅らせるべきではないので、Sleep() 関数はカスタム指標からは呼び出すことが出来ません。この関数は 0.1 秒ごとの EA 休止フラグのチェックを内蔵しています。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- チャート上のコメントに10から1までのカウントダウンを表示する
   for(int i=10; i>0 && !IsStopped(); i--)
     {
      Comment(StringFormat("Wait %u seconds",i));
      Sleep(1000);
     }
//--- 「到着」コメントにスクリプトの目的を説明するテキストを書く
   string text="This was a test showing how the Sleep() function works";
   string mess="";
   for(int i=0; i<(int)text.Length(); i++)
     {
      mess+=ShortToString(text.GetChar(i));
      Sleep(100);
      Comment(mess);
     }
//--- さよならを言う...
   Sleep(1000);
   for(int i=0; i<6; i++)
     {
      mess=(i % 2 == 0 ? "" : "  Bye!");
      Comment(mess);
```

© 2000-2025, MetaQuotes Ltd.

```
      Sleep(300);
     }
//--- チャートのテキストを削除する
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

TerminalClose

この関数は端末に操作完了を命令します。

```
bool  TerminalClose(
   int ret_code      // クライアント端末のコードを終了する
   );
```

パラメータ

ret_code

[in] 動作完了時にクライアント端末の処理によって返されたリターンコード

戻り値

成功の場合は true、それ以外の場合は false

注意事項

TerminalClose() 関数は、すぐに端末を停止せずに、動作を完了するように端末に命令します。

TerminalClose() を呼び出したエキスパートアドバイザーのコードは、即時完了のための準備を（例えば、開いているファイルは、通常モードで閉じる）終える必要があります。この関数の呼び出しは リターンオペレータ で続かれる必要があります。

ret_code パラメータは、コマンドプロンプトからの起動時に端末操作のプログラム終了の理由を分析するのに必要なリターンコードを示すことが出来ます。

例：

```
//--- 入力パラメータ
input int   tiks_before=500; // 終了までのティック数
input int   pips_to_go=15;   // ピップス（為替差益）
input int   seconds_st=50;   // エキスパートアドバイザーに与えられた秒数
//--- グローバル
datetime    launch_time;
int         tick_counter=0;
//+------------------------------------------------------------------+
//| エキスパート初期化解除に使用される関数                           |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Print(__FUNCTION__," reason code = ",reason);
   Comment("");
  }
//+------------------------------------------------------------------+
//| エキスパートティック関数                                         |
//+------------------------------------------------------------------+
void OnTick()
  {
   static double first_bid=0.0;
   MqlTick       tick;
```

© 2000-2025, MetaQuotes Ltd.

```
   double         distance;
//---
   SymbolInfoTick(_Symbol,tick);
   tick_counter++;
   if(first_bid==0.0)
     {
      launch_time=tick.time;
      first_bid=tick.bid;
      Print("first_bid =",first_bid);
      return;
     }
//--- 価格距離(ピップス単位)
   distance=(tick.bid-first_bid)/_Point;
//--- EA 動作を追跡するための通知を表示する
   string comm="From the moment of start:\r\n\x25CF elapsed seconds: "+
               IntegerToString(tick.time-launch_time)+" ;"+
               "\r\n\x25CF ticks received: "+(string)tick_counter+" ;"+
               "\r\n\x25CF price went in points: "+StringFormat("%G",distance);
   Comment(comm);
//--- 端末を閉じる状態を確認するためのセクション
   if(tick_counter>=tiks_before)
      TerminalClose(0);     // ティックカウンタによる終了
   if(distance>pips_to_go)
      TerminalClose(1);     // pips_to_go に等しいピップスの数で上がる
   if(distance<-pips_to_go)
      TerminalClose(-1);    // pips_to_go に等しいピップスの数で下げる
   if(tick.time-launch_time>seconds_st)
      TerminalClose(100);   // 時間切れによる終了
//---
  }
```

参照

プログラム実行、実行エラー、初期化解除の理由

© 2000-2025, MetaQuotes Ltd.

TesterHideIndicators

EAで使用される指標の非/表示モードを設定します。この関数は、テスト中にのみ指標の可視性を管理するためのものです。

```
void  TesterHideIndicators(
   bool      hide     // フラグ
   );
```

パラメータ

hide

[in] テスト中に指標を非表示にするフラグ(true - 非表示、false - 表示)

戻り値

なし

注意事項

デフォルトでは、テスト中のEAで作成されたすべての指標は視覚的にテストチャートに表示されます。さらに、これらの指標は、テストが完了したときに自動的に開かれるチャートに表示されます。TesterHideIndicators()関数を使用すると、開発者は使用される指標の表示を無効にすることができます。

EAをテストするときに適用された指標の表示を無効にするには、EAのハンドルを作成する前にTesterHideIndicators()をtrueで呼び出します。それ以降に作成されるすべての指標には非表示フラグが設定されます。これらの指標は、ビジュアルテスト中にも、テスト完了時に自動的に開かれるチャート上にも表示されません。

新しく作成された指標の非表示モードを無効にするには、TesterHideIndicators()をfalseで呼び出します。テスト中のEAから直接生成された指標のみをテストチャートに表示することができます。このルールは<data_folder>MQL5¥Profiles¥Templatesにテンプレートが1つもない場合にのみ適用されます。

<data_folder>MQL5¥Profiles¥Templatesディレクトリに特別なテンプレート(<EA_name>.tpl)が含まれている場合は、このテンプレートの指標のみがビジュアルテスト中とテストチャート上に表示されます。この場合、テスト中のEAに適用された指標は表示されません。この動作は、EAコードでTesterHideIndicators()がtrueで呼び出されても変わりません。

<data_folder>MQL5¥Profiles¥Templatesディレクトリにtester.tplがあるが特別な<EA_name>.tplテンプレートがない場合は、tester.tplの指標とTesterHideIndicators()関数で無効化されていないEA指標がビジュアルテストとテストチャートで表示されます。tester.tplテンプレートがない場合は、default.tplテンプレートの指標が代わりに使われます。

ストラテジーテスターが適切なテンプレート(<EA_name>.tpl、tester.tplまたはdefault.tpl)を見つけられなかった場合、EAに適用された指標の表示は完全にTesterHideIndicators()関数によって管理されます。

例:

```
bool CSampleExpert::InitIndicators(void)
  {
   TesterHideIndicators(true);
//--- MACD指標を作成する
   if(m_handle_macd==INVALID_HANDLE)
      if((m_handle_macd=iMACD(NULL,0,12,26,9,PRICE_CLOSE))==INVALID_HANDLE)
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          printf("Error creating MACD indicator");
          return(false);
         }
    TesterHideIndicators(false);
//--- EMA指標を作成してコレクションに追加する
    if(m_handle_ema==INVALID_HANDLE)
       if((m_handle_ema=iMA(NULL,0,InpMATrendPeriod,0,MODE_EMA,PRICE_CLOSE))
==INVALID_HANDLE)
         {
          printf("Error creating EMA indicator");
          return(false);
         }
//--- 成功した
    return(true);
   }
```

参照

IndicatorRelease

© 2000–2025, MetaQuotes Ltd.

TesterStatistics

この関数はテストの結果に基づいて算出された指定の統計パラメータを返します。

```
double  TesterStatistics(
   ENUM_STATISTICS statistic_id      // ID
   );
```

パラメータ

statistic_id

[in] ENUM_STATISTICS 列挙から選ばれた統計パラメータの ID

戻り値

テスト結果に基づいた統計パラメータの値

注意事項

この関数はテスタの OnTester() または OnDeinit() の中で呼び出されます。それ以外の場合、結果は定義されていません。

例：

```
// 標準の「MACD Sample.mq5」ファイルに基づく EA
// Tester イベントハンドラの Tester Statistics() の結果を表示する
#define MACD_MAGIC 1234502
//---
#include <Trade\Trade.mqh>
#include <Trade\SymbolInfo.mqh>
#include <Trade\PositionInfo.mqh>
#include <Trade\AccountInfo.mqh>
//---
input double InpLots          =0.1; // ロット
input int    InpTakeProfit    =50;  // 単位のテイクプロフィット（ピップス単位）
input int    InpTrailingStop  =30;  // トレーリングストップレベル（ピップス単位）
input int    InpMACDOpenLevel =3;   // MACDオープンレベル（ピップス単位）
input int    InpMACDCloseLevel=2;   // MACDクローズレベル（ピップス単位）
input int    InpMATrendPeriod =26;  // MAトレンド期間
//+------------------------------------------------------------------+
//| エキスパート初期化関数                                           |
//+------------------------------------------------------------------+
int OnInit(void)
  {
//--- 必要なオブジェクトをすべて作成する
   if(!ExtExpert.Init())
      return(INIT_FAILED);
//--- 正常な初期化
   return(INIT_SUCCEEDED);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  エキスパートティック処理関数                                   |
//+------------------------------------------------------------------+
void OnTick(void)
  {
   static datetime limit_time=0; // last call time considering 'timeout'
//--- 時間が指定されたlimit_time値を超えた場合
   if(TimeCurrent()>=limit_time)
     {
      //--- データを確認する
      if(Bars(Symbol(),Period())>2*InpMATrendPeriod)
        {
         //--- 成功した場合は、limit_timeを「timeout」秒だけ増やす
         if(ExtExpert.Processing())
            limit_time=TimeCurrent()+ExtTimeOut;
        }
     }
  }
//+------------------------------------------------------------------+
//|  エキスパートテスター処理関数                                    |
//+------------------------------------------------------------------+
double OnTester(void)
  {
   double ret=TesterStatistics(STAT_PROFIT_FACTOR);
   double profit=TesterStatistics(STAT_PROFIT);
   int    trades_total=(int)TesterStatistics(STAT_TRADES);
   int    profit_total=(int)TesterStatistics(STAT_PROFIT_TRADES);
   int    loss_total=(int)TesterStatistics(STAT_LOSS_TRADES);
   PrintFormat("%s: Profit = %.2f, trades total: %lu, profit trades total: %lu, loss
trades total: %lu, profit factor:
%.2f",__FUNCTION__,profit,trades_total,profit_total,loss_total,ret);
   return(ret);
   /*
  結果:
   OnTester: Profit = 209.84, trades total: 13, profit trades total: 11, loss trades
 total: 2, profit factor: 3.02
   final balance 10209.84 USD
   OnTester result 3.020606644198363
   */
  }
```

© 2000–2025, MetaQuotes Ltd.

TesterStop

テスト中にプログラム操作終了コマンドを発行します。

```
void  TesterStop();
```

戻り値

なし

注意事項

TesterStop()関数はテストエージェント（たとえば、特定の数の負け取引または事前に設定されたドローダウンレベルに達した場合）で、EAを定期的に早期にシャットダウンするように設計されています。

TesterStop()の呼び出しはテストの正常終了とみなされるため、OnTester()関数が呼び出され、累積取引統計および最適化基準の値がストラテジーテスターに送信されます。

ストラテジーテスターでのExpertRemove()の呼び出しもテストの正常終了とみなされて取引統計を取得できますが、EAはエージェントのメモリからアンロードされます。この場合、プログラムを再読み込みするためには、次のパラメータセットのパスを実行する必要があります。よって、TesterStop()はテストの早期終了のための好ましいオプションです。

例：

```
//--- define
#define BALANCE_LOSS_STOP  100.0       // value of the balance drawdown, at which
testing is stopped
#define EQUITY_LOSS_STOP   100.0       // value of the equity drawdown, at which
testing is stopped

//--- 入力パラメータ
input  double  InpLots        =  0.1;  // ロット
input  uint    InpStopLoss    =  50;   // ポイント単位のストップロス
input  uint    InpTakeProfit  =  150;  // ポイント単位のテイクプロフィット
sinput ulong   InpMagic       =  123;  // マジックナンバー
sinput ulong   InpDeviation   =  5;    // 偏差
//--- グローバル変数
CTrade      trade;                     // 取引クラスのインスタンス
CSymbolInfo symb;                      // 銘柄クラスのインスタンス
CAccountInfo account;                  // 取引口座クラスのインスタンス
...
//+------------------------------------------------------------------+
//|  エキスパート初期化関数                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
   ...
//--- 正常な初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

© 2000–2025, MetaQuotes Ltd.

```
//|  エキスパートティック関数                                                |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- 現在の相場を更新する
   if(!symb.RefreshRates())
      return;
   ...

//--- 残高または資本がBALANCE_LOSS_STOPおよびEQUITY_LOSS_STOPマクロ置換で示されている以上に減少した場合
//--- テストは失敗したとみなされ、Tester Stop() 関数が呼び出される
//--- BALANCE_LOSS_STOPを超える残高損失を確認する
   if(balance_prev!=account.Balance())
     {
      if(account.Balance()<balance_prev-BALANCE_LOSS_STOP)
        {
         PrintFormat("The initial balance of %.2f %s decreased by %.2f %s, and now has a value of %.2f %s. Stop testing.",balance_prev,account.Currency(),balance_prev-account.Balance(),account.Currency(),account.Balance(),account.Currency());
         TesterStop();
         /*
        結果:
         The initial balance of 10000.00 USD decreased by 100.10 USD, and now has a value of 9899.90 USD. Stop testing.
         TesterStop() called on 9% of testing interval
         */
        }
     }
//--- EQUITY_LOSS_STOPを超える資本の損失を確認する
   if(equity_prev!=account.Equity())
     {
      if(account.Equity()<equity_prev-EQUITY_LOSS_STOP)
        {
         PrintFormat("The initial equity of %.2f %s decreased by %.2f %s, and now has a value of %.2f %s. Stop testing.",equity_prev,account.Currency(),equity_prev-account.Equity(),account.Currency(),account.Equity(),account.Currency());
         TesterStop();
         /*
        結果:
         The initial equity of 10000.00 USD decreased by 100.10 USD, and now has a value of 9899.90 USD. Stop testing.
         TesterStop() called on 9% of testing interval
         */
        }
     }
  }
```

参照

プログラムの実行、取り引き戦略のテスト、ExpertRemove、SetReturnError

© 2000-2025, MetaQuotes Ltd.

TesterDeposit

テスト中の資金入金をエミュレートする特殊関数です。資金管理システムの一部で使用することができます。

```
bool  TesterDeposit(
   double money      // 入金額
   );
```

パラメータ

money

[in] 預金通貨での口座への入金

戻り値

成功の場合は true、それ以外の場合は false

例：

```
//--- define
#define BALANCE_LOSS_DEPOSIT  100.0    // テスターの口座に資金が入金される残高ドローダウンの値

//--- 入力パラメータ
input  double  InpLots         =  0.1;  // ロット
input  uint    InpStopLoss     =  50;   // ポイント単位のストップロス
input  uint    InpTakeProfit   =  150;  // ポイント単位のテイクプロフィット
sinput ulong   InpMagic        =  123;  // マジックナンバー
sinput ulong   InpDeviation    =  5;    // 偏差
//--- グローバル変数
CTrade      trade;                      // 取引クラスのインスタンス
CSymbolInfo symb;                       // 銘柄クラスのインスタンス
CAccountInfo account;                   // 取引口座クラスのインスタンス
...
double      balance_dep_summ;           // 残高チャージ総額
uint        balance_dep_total;          // 残高チャージ数
//+------------------------------------------------------------------+
//|  エキスパート初期化関数                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
   ...
//--- 初期バランス値を保存する
   balance_prev=account.Balance();
   balance_dep_summ=0;
   balance_dep_total=0;
//--- 正常な初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| エキスパートティック関数                                                 |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- 現在の相場を更新する
   if(!symb.RefreshRates())
      return;
   ...

//--- BALANCE_LOSS_DEPOSIT マクロ置換で示されている以上に残高が減少した場合、
//--- 口座にチャージしてTesterDeposit()関数を呼び出す必要がある
//--- BALANCE_LOSS_DEPOSIT を超える残高の損失を確認する
   if(balance_prev!=account.Balance())
     {
      if(account.Balance()<balance_prev-BALANCE_LOSS_DEPOSIT)
        {
         double loss=balance_prev-account.Balance();
         PrintFormat("The initial balance of %.2f %s decreased by %.2f %s. It is
necessary to make a deposit to the account for %.2f %
s."
,balance_prev,account.Currency(),loss,account.Currency(),loss,account.Currency());
         if(TesterDeposit(loss))
           {
            balance_dep_total++;
            balance_dep_summ+=loss;
            balance_prev=account.Balance();
            PrintFormat("Funds have been deposited into the account. Account
balance: %.2f %s.",account.Balance(),account.Currency());
            PrintFormat("Total deposits: %lu. Amount of deposits: %.2f %
s.",balance_dep_total,balance_dep_summ,account.Currency());
           }
         /*
        結果:
         The initial balance of 10000.00 USD decreased by 116.00 USD. It is
necessary to make a deposit to the account for 116.00 USD.
         deal #45 balance 116.00 [deposit] done
         Funds have been deposited into the account. Account balance: 10000.00 USD.
         Total deposits: 1. Amount of deposits: 116.00 USD.
         */
        }
     }
  }
//+------------------------------------------------------------------+
//| テスタ関数                                                       |
//+------------------------------------------------------------------+
double OnTester()
  {
//--- 最大残高ドローダウンを金額換算で出力ハンドラー値として設定する
   double ret=TesterStatistics(STAT_BALANCE_DD);
//--- ドローダウン、入金数、その合計額に関するメッセージをログに表示する
```

© 2000-2025, MetaQuotes Ltd.

```
   PrintFormat("%s: Maximum balance drawdown in money: %.2f %s. Total deposits: %lu.
Amount of deposits: %.2f %
s."
,__FUNCTION__
,ret,account.Currency(),balance_dep_total,balance_dep_summ,account.Currency());
//--- 結果を返す
   return(ret);
   /*
  結果:
   OnTester: Maximum balance drawdown in money: 5188.50 USD. Total deposits: 46.
Amount of deposits: 5128.50 USD.
   final balance 4867.50 USD
   OnTester result 5188.5
   */
  }
```

参照

TesterWithdrawal

© 2000–2025, MetaQuotes Ltd.

TesterWithdrawal

この特殊関数は、テストの過程で出金の動作をエミュレートします。資産管理システムでの使用が可能です。

```
bool  TesterWithdrawal(
   double money      // 出金の合計額
   );
```

パラメータ

money

[in]（預金通貨で）出金必要額の合計

戻り値

成功の場合は true、それ以外の場合は false

例：

```
//--- define
#define BALANCE_PROFIT_WITHDRAWAL   5  // テスターの口座から資金が引き出される残高利益の値

//--- 入力パラメータ
input  double  InpLots         =  0.1;  // ロット
input  uint    InpStopLoss     =  50;   // ポイント単位のストップロス
input  uint    InpTakeProfit   =  150;  // ポイント単位のテイクプロフィット
sinput ulong   InpMagic        =  123;  // マジックナンバー
sinput ulong   InpDeviation    =  5;    // 偏差
//--- グローバル変数
CTrade      trade;                      // 取引クラスのインスタンス
CSymbolInfo symb;                       // 銘柄クラスのインスタンス
CAccountInfo account;                   // 取引口座クラスのインスタンス
...
double      balance_op_sum;             // 残高操作の合計額
uint        balance_op_total;           // 残高操作の数
//+------------------------------------------------------------------+
//|  エキスパート初期化関数                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
   ...
//--- 初期バランス値を保存する
   balance_prev=account.Balance();
   balance_op_sum=0;
   balance_op_total=0;
//--- 正常な初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|  エキスパートティック関数                                                |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- 現在の相場を更新する
   if(!symb.RefreshRates())
      return;
   ...

//--- 残高利益が現在の残高をBALANCE_PROFIT_WITHDRAWALマクロ置換で指定された値だけ超える場合、
//--- これらの資金を口座から引き出す必要があるTesterWithdrawal() 関数を呼び出します
//--- BALANCE_PROFIT_WITHDRAWALを超える残高利益を確認する
   if(balance_prev!=account.Balance())
     {
      if(account.Balance()>balance_prev+BALANCE_PROFIT_WITHDRAWAL)
        {
         double profit=account.Balance()-balance_prev;
         PrintFormat("The account balance has been increased by %.2f %s. Need to
withdraw these funds from the account.",profit,account.Currency());
         if(TesterWithdrawal(profit))
           {
            balance_op_total++;
            balance_op_summ+=profit;
            balance_prev=account.Balance();
            PrintFormat("Funds have been withdrawn from the account. Account
balance: %.2f %s.",account.Balance(),account.Currency());
            PrintFormat("Total withdrawals: %lu. Amount of withdrawals: %.2f %
s.",balance_op_total,balance_op_summ,account.Currency());
           }
         /*
         結果:
         The account balance has been increased by 21.00 USD. Need to withdraw these
 funds from the account.
         deal #13 balance -21.00 [withdrawal] done
         Funds have been withdrawn from the account. Account balance: 10000.00 USD.
         Total withdrawals: 1. Amount of withdrawals: 21.00 USD.
         */
        }
     }
  }
//+------------------------------------------------------------------+
//|  テスタ関数                                                      |
//+------------------------------------------------------------------+
double OnTester()
  {
//--- 最大残高ドローダウンを金額換算で出力パラメータ値として設定する
   double ret=TesterStatistics(STAT_BALANCE_DD);
//--- ドローダウン、出金回数、およびその合計額に関するメッセージをログに表示する
```

© 2000-2025, MetaQuotes Ltd.

```
   PrintFormat("%s: Maximum balance drawdown in money: %.2f %s. Total withdrawals: %
lu. Amount of withdrawals: %.2f %
s."
,__FUNCTION__
,ret,account.Currency(),balance_op_total,balance_op_summ,account.Currency());
//--- 結果を返す
   return(ret);
   /*
  結果:
   OnTester: Maximum balance drawdown in money: 5188.50 USD. Total withdrawals: 2.
Amount of withdrawals: 36.00 USD.
   final balance 4867.50 USD
   OnTester result 5188.5
   */
  }
```

参照

TesterDeposit

© 2000–2025, MetaQuotes Ltd.

TranslateKey

現在の入力言語とコントロールキーの状態を考慮して、仮想キーコードでUnicode文字を返します。

```
short  TranslateKey(
   int  key_code      // Unicode文字を受け取るキーコード
   );
```

パラメータ

key_code

[in] キーコード

戻り値

変換が成功した場合はUnicode文字。エラーの場合は-1。

注意事項

この関数はToUnicodeExを使用してユーザーが押したキーをUnicode文字に変換します。ToUnicodeExがトリガされない場合（たとえば、SHIFTキー文字を受け取ろうとする場合など）にはエラーが発生することがあります。

例:

```
void OnChartEvent(const int id,const long& lparam,const double& dparam,const string&
sparam)
  {
   if(id==CHARTEVENT_KEYDOWN)
     {
      short sym=TranslateKey((int)lparam);
      //--- 入力された文字が最上にUnicodeに変換された場合
      if(sym>0)
         Print(sym,"'",ShortToString(sym),"'");
      else
         Print("Error in TranslateKey for key=",lparam);
     }
  }
```

参照

クライアント端末イベント、OnChartEvent

© 2000-2025, MetaQuotes Ltd.

ZeroMemory

この関数は参照によって渡された変数をリセットします。

```
void  ZeroMemory(
   void & variable      // 変数をリセットする
   );
```

パラメータ

variable

[in] [out] リセットされる、参照によって渡された変数（ゼロに初期化する）

戻り値

なし

注意事項

この関数の呼び出しは、引数が文字列である場合、その値として NULL を示すのと同じです。
基本データ型とその配列、また基本データ型の構造体とクラスにとっては、これはただのリセットと同じです。
文字列と動的配列を含むオブジェクトの場合、ZeroMemory() が各要素に対して呼び出されます。
const 修飾子によって保護されていない任意の配列の場合、すべての要素がゼロで初期化されます。
複合オブジェクトの配列の場合、ZeroMemory() が各要素に対して呼び出されます。

ZeroMemory() は protected メンバや継承クラスを持つクラスでは使用できません。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 文字列を宣言して初期化する
   string str="Test ZeroMemory func";
//--- ZeroMemory()を適用する前にログに行を送信する
   PrintFormat("The line before applying ZeroMemory() to it: '%s'",str);
//--- 文字列をリセットし、結果をログに送信する
   ZeroMemory(str);
   Print("The same line after applying ZeroMemory() to it: '",str,"'");
  /*
  結果:
   The line before applying ZeroMemory() to it: 'Test ZeroMemory func'
   The same line after applying ZeroMemory() to it: ''
  */

//--- int型の変数を宣言して初期化する
   int var=123;
//--- ZeroMemory()を適用する前にログに行を送信する
```

© 2000-2025, MetaQuotes Ltd.

```
   PrintFormat("\nThe integer variable before applying ZeroMemory() to it: %d",var);
//--- 変数をリセットし、結果をログに送信する
   ZeroMemory(var);
   PrintFormat("The same variable after applying ZeroMemory() to it: %d",var);
  /*
  結果:
   The integer variable before applying ZeroMemory() to it: 123
   The same variable after applying ZeroMemory() to it: 0
  */

//--- int 型の配列を宣言して初期化する
   int arr[]={0,1,2,3,4,5,6,7,8,9};
//--- ZeroMemory() を適用する前にログに配列を送信する
   Print("\nThe integer array before applying ZeroMemory() to it:");
   ArrayPrint(arr);
//--- 配列をリセットし、結果をログに送信する
   ZeroMemory(arr);
   Print("The same array after applying ZeroMemory() to it:");
   ArrayPrint(arr);
  /*
  結果:
   The integer array before applying ZeroMemory() to it:
   0 1 2 3 4 5 6 7 8 9
   The same array after applying ZeroMemory() to it:
   0 0 0 0 0 0 0 0 0 0
  */

//--- 2つのフィールド（文字列フィールドと整数フィールド）の構造体を宣言する
   struct STest
     {
      string   var_string;
      long     var_long;
     };
//--- STest 構造体型の配列を宣言して初期化する
   STest arr_struct[]={ {"0",0}, {"1",1}, {"2",2}, {"3",3} };
//--- ZeroMemory() を適用する前にログに配列を送信する
   Print("\nThe array struct before applying ZeroMemory() to it:");
   ArrayPrint(arr_struct);
//--- 構造体配列をリセットし、結果をログに送信する
   ZeroMemory(arr_struct);
   Print("The same array struct after applying ZeroMemory() to it:");
   ArrayPrint(arr_struct);
  /*
  結果:
   The array struct before applying ZeroMemory() to it:
       [var_string] [var_long]
   [0] "0"                   0
   [1] "1"                   1
   [2] "2"                   2
```

© 2000-2025, MetaQuotes Ltd.

```
   [3] "3"                   3
   The same array struct after applying ZeroMemory() to it:
       [var_string] [var_long]
   [0] null                  0
   [1] null                  0
   [2] null                  0
   [3] null                  0
  */
  }
```

© 2000–2025, MetaQuotes Ltd.

配列を使用する関数群

配列は最大 4 次元まで許可されています。それぞれの次元のインデックスは 0 から *dimension_size-1* です。例えば、50 の要素からなる1次元の配列では、最初の要素の呼び出しは array[0] となり、最後の要素は array[49] と表現されます。

関数	アクション
ArrayBsearch	初次元で最初に見つかった要素のインデックスを返します。
ArrayCopy	配列を別の配列に複製します。
ArrayCompare	基本データ型または複合型オブジェクトを使用しないカスタム構造体から出来た 2 つの配列を比較した結果を返します。
ArrayFree	動的配列のバッファを解放し、ゼロ次元のサイズを 0 に設定します。
ArrayGetAsSeries	配列の索引付けの方向をチェックします。
ArrayInitialize	数値配列の全ての要素を 1 つの値に設定します。
ArrayFill	配列を指定された値に設定します。
ArrayIsSeries	配列が時系列であるかどうかをチェックします。
ArrayIsDynamic	配列が動的であるかどうかをチェックします。
ArrayMaximum	最大値を持つ要素を検索します。
ArrayMinimum	最小値を持つ要素を検索します。
ArrayPrint	単純型または単純な構造体の配列を操作ログに出力します
ArrayRange	配列の指定された次元の要素数を返します。
ArrayResize	配列の初次元のサイズを新しく設定します。
ArrayInsert	指定されたインデックスから開始して、指定された数の要素をソース配列からターゲット配列に挿入します
ArrayRemove	指定されたインデックスから開始して、配列内の指定された数の要素を削除します
ArrayReverse	指定されたインデックスから開始して、配列内の指定された数の要素を反転します
ArraySetAsSeries	配列の索引付けの方向を設定します。
ArraySize	配列の要素数を返します。
ArraySort	数値配列を初次元によってソートします。
ArraySwap	同じ型の2つの動的配列の内容を交換します。
ArrayToFP16	float型またはdouble型の配列を、指定された形式のushort型の配列にコピーします
ArrayToFP8	float型またはdouble型の配列を、指定された形式のuchar型の配列にコピーします

© 2000–2025, MetaQuotes Ltd.

関数	アクション
ArrayFromFP16	ushort型の配列を、指定された形式のfloat型またはdouble型の配列にコピーします
ArrayFromFP8	uchar型の配列を、指定された形式のfloat型またはdouble型の配列にコピーします

© 2000-2025, MetaQuotes Ltd.

ArrayBsearch

昇順にソートされた多次元数値配列内の指定された値を検索します。検索は最初の次元の要素を介して行われます。

double 型の配列の検索には

```
int  ArrayBsearch(
   const double&    array[],   // 検索対象の配列
   double           value      // 検索される値
   );
```

float 型の配列の検索には

```
int  ArrayBsearch(
   const float&    array[],   // 検索対象の配列
   float           value      // 検索される値
   );
```

long 型の配列の検索には

```
int  ArrayBsearch(
   const long&    array[],   // 検索対象の配列
   long           value      // 検索される値
   );
```

int 型の配列の検索には

```
int  ArrayBsearch(
   const int&    array[],   // 検索対象の配列
   int           value      // 検索される値
   );
```

short 型の配列の検索には

```
int  ArrayBsearch(
   const short&    array[],   // 検索対象の配列
   short           value      // 検索される値
   );
```

char 型の配列の検索には

```
int  ArrayBsearch(
   const char&    array[],   // 検索対象の配列
   char           value      // 検索される値
   );
```

パラメータ

array[]

[in] 検索される数値配列

value

© 2000-2025, MetaQuotes Ltd.

[in] 検索される数値

戻り値

この関数は、見つかった要素のインデックスを返します。指定された値を持つ要素が配列に存在しない場合は、検索値の範囲で最も近い値を持つ要素のインデックスが返されます。

注意事項

二分探索ではソートされた配列の処理だけが可能です。数値の配列をソートするには ArraySort() 関数を使用します。

例:

```
#property description "Script based on RSI indicator data displays"
#property description "how often the market was in"
#property description "overbought and oversold areas in the specified time
interval."
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- 入力パラメータ
input int                InpMAPeriod=14;                    // 平均移動線の期間
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;       // 価格の種類
input double             InpOversoldValue=30.0;             //売りすぎレベル
input double             InpOverboughtValue=70.0;           //買いすぎレベル
input datetime           InpDateStart=D'2012.01.01 00:00';  //分析開始日
input datetime           InpDateFinish=D'2013.01.01 00:00'; // 分析終了日
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
   double rsi_buff[]; // 指標値の配列
   int    size=0;     // 配列サイズ
//--- RSI  指標ハンドルを受け取る
   ResetLastError();
   int rsi_handle=iRSI(Symbol(),Period(),InpMAPeriod,InpAppliedPrice);
   if(rsi_handle==INVALID_HANDLE)
     {
      //--- 指標ハンドルの受け取りに失敗
      PrintFormat("Indicator handle receiving error. Error code = %
d",GetLastError());
      return;
     }
//--- 指標が全ての値を計算するまでループ内にとどまる
   while(BarsCalculated(rsi_handle)==-1)
     {
      //--- 指標が強制的にスクリプトの動作を完了した場合ループから出る
      if(IsStopped())
         return;
      //--- 指標が全ての値を計算出来るように休止する
```

© 2000-2025, MetaQuotes Ltd.

```
      Sleep(10);
     }
//--- 一定期間の指標値を複製する
   ResetLastError();
   if(CopyBuffer(rsi_handle,0,InpDateStart,InpDateFinish,rsi_buff)==-1)
     {
      PrintFormat("Failed to copy the indicator values. Error code = %
d",GetLastError());
      return;
     }
//--- 配列のサイズを受け取る
   size=ArraySize(rsi_buff);
//--- 配列をソートする
   ArraySort(rsi_buff);
//--- 市場が売り過ぎの領域にあった期間を(百分率で)調べる
   double ovs=(double)ArrayBsearch(rsi_buff,InpOversoldValue)*100/(double)size;
//--- 市場が買われ過ぎの領域にあった期間を(百分率で)調べる
   double ovb=(double)(size-ArrayBsearch(rsi_buff,InpOverboughtValue))*100/(double)
size;
//--- データを表示するための文字列を形成する
   string str="From "+TimeToString(InpDateStart,TIME_DATE)+" to "
              +TimeToString(InpDateFinish,TIME_DATE)+" the market was:";
   string str_ovb="in overbought area "+DoubleToString(ovb,2)+"% of time";
   string str_ovs="in oversold area "+DoubleToString(ovs,2)+"% of time";
//--- チャートにデータを表示する
   CreateLabel("top",5,60,str,clrDodgerBlue);
   CreateLabel("overbought",5,35,str_ovb,clrDodgerBlue);
   CreateLabel("oversold",5,10,str_ovs,clrDodgerBlue);
//--- チャートを再描画する
   ChartRedraw(0);
//--- 休止する
   Sleep(10000);
  }
//+------------------------------------------------------------------+
//|  チャートの左下隅にコメントを表示する                                |
//+------------------------------------------------------------------+
void CreateLabel(const string name,const int x,const int y,
                 const string str,const color clr)
  {
//--- ラベルを作成する
   ObjectCreate(0,name,OBJ_LABEL,0,0,0);
//--- 左下隅にラベルを結合する
   ObjectSetInteger(0,name,OBJPROP_CORNER,CORNER_LEFT_LOWER);
//--- アンカーポイントの位置を変更する
   ObjectSetInteger(0,name,OBJPROP_ANCHOR,ANCHOR_LEFT_LOWER);
//--- X 方向におけるアンカーポイントからの距離
   ObjectSetInteger(0,name,OBJPROP_XDISTANCE,x);
//--- Y 方向におけるアンカーポイントからの距離
   ObjectSetInteger(0,name,OBJPROP_YDISTANCE,y);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- ラベルテキスト
   ObjectSetString(0,name,OBJPROP_TEXT,str);
//--- テキストの色
   ObjectSetInteger(0,name,OBJPROP_COLOR,clr);
//--- テキストサイズ
   ObjectSetInteger(0,name,OBJPROP_FONTSIZE,12);
  }
```

© 2000–2025, MetaQuotes Ltd.

ArrayCopy

配列を別の配列に複製します。

```
int  ArrayCopy(
   void&         dst_array[],          // 複製先の配列
   const void&   src_array[],          // ソース配列
   int           dst_start=0,          // 複製先の配列の開始インデックス
   int           src_start=0,          // ソース配列の開始インデックス
   int           count=WHOLE_ARRAY     // 要素の数
   );
```

パラメータ

dst_array[]

[out] 複製先の配列。

src_array[]

[in] ソース配列。

dst_start=0

[in] 複製先の配列の開始インデックス。デフォルトでは、開始インデックスは 0。

src_start=0

[in] ソース配列の開始インデックス。デフォルトでは、開始インデックスは 0。

count=WHOLE_ARRAY

[in] 複製する必要がある要素数。デフォルトでは配列全体（count=WHOLE_ARRAY）。

戻り値

複製された要素の数

注意事項

count<0 や count>src_size-src_start の場合、配列の全ての残り部分が複製されます。配列は、左から右に複製されます。シリーズ配列の場合、開始位置は左から右のコピーのために調整されて、正確に定義されます。

配列のタイプが異なる場合は、複製中に、複製元配列の各要素の複製先の要素タイプへの変換が試みられます。文字列配列は文字列配列のみに複製することが出来ます。初期化を必要とするオブジェクトを含むクラスと構造の配列は複製されません。構造体の配列は同じ型の配列のみに複製することが出来ます。

時系列をインデックスとする動的配列では（もしデータサイズのほうが大きければ）複製先の配列のサイズが自動的に複製されたデータサイズに合わせて増加されます。複製先の配列のサイズは自動的に減少しません。

例：

```
#property description "The indicator highlights the candlesticks that are local"
#property description "highs and lows. Interval length for finding"
#property description "extreme values should be found using an input parameters."
//--- 指標の設定
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   1
//---- プロット
#property indicator_label1  "Extremums"
#property indicator_type1   DRAW_COLOR_CANDLES
#property indicator_color1  clrLightSteelBlue,clrRed,clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 定義済み定数
#define INDICATOR_EMPTY_VALUE 0.0
//--- 入力パラメータ
input int InpNum=4; // 区間の半分の長さ
//--- 指標バッファ
double ExtOpen[];
double ExtHigh[];
double ExtLow[];
double ExtClose[];
double ExtColor[];
//--- グローバル変数
int    ExtStart=0; // 極値ではない最初のローソク足のインデックス
int    ExtCount=0; // 区間内の非極値の数
//+------------------------------------------------------------------+
//|  非極値のローソク足に値を入れる                                  |
//+------------------------------------------------------------------+
void FillCandles(const double &open[],const double &high[],
                 const double &low[],const double &close[])
  {
//--- ローソク足に値を入れる
   ArrayCopy(ExtOpen,open,ExtStart,ExtStart,ExtCount);
   ArrayCopy(ExtHigh,high,ExtStart,ExtStart,ExtCount);
   ArrayCopy(ExtLow,low,ExtStart,ExtStart,ExtCount);
   ArrayCopy(ExtClose,close,ExtStart,ExtStart,ExtCount);
  }
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 指標バッファマッピング
   SetIndexBuffer(0,ExtOpen);
   SetIndexBuffer(1,ExtHigh);
   SetIndexBuffer(2,ExtLow);
   SetIndexBuffer(3,ExtClose);
   SetIndexBuffer(4,ExtColor,INDICATOR_COLOR_INDEX);
//--- 値を指定するが表示しない
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,INDICATOR_EMPTY_VALUE);
//--- データウィンドウに表示するために指標バッファの名称を指定する
   PlotIndexSetString(0,PLOT_LABEL,"Open;High;Low;Close");
```

© 2000-2025, MetaQuotes Ltd.

```
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- 時系列にストレート索引付けを設定する
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(close,false);
//--- バー計算開始の変数
   int start=prev_calculated;
//--- 最初の InpNum*2 のバーは計算されない
   if(start==0)
     {
      start+=InpNum*2;
      ExtStart=0;
      ExtCount=0;
     }
//--- バーがちょうど形成した場合、次の潜在的な極値をチェックする
   if(rates_total-start==1)
      start--;
//--- 極値チェックが行われるバーのインデックス
   int ext;
//--- 指標算出ループ
   for(int i=start;i<rates_total-1;i++)
     {
      //--- 初めは i 番目のバーに指標は描画されない
      ExtOpen[i]=0;
      ExtHigh[i]=0;
      ExtLow[i]=0;
      ExtClose[i]=0;
      //--- チェックする極値インデックス
      ext=i-InpNum;
      //--- 極大をチェック
      if(IsMax(high,ext))
        {
```

© 2000-2025, MetaQuotes Ltd.

```
         //--- 極値であるローソク足を強調表示する
         ExtOpen[ext]=open[ext];
         ExtHigh[ext]=high[ext];
         ExtLow[ext]=low[ext];
         ExtClose[ext]=close[ext];
         ExtColor[ext]=1;
         //---極値まで他のローソク足を中間色で強調表示する
         FillCandles(open,high,low,close);
         //--- 変数の色を変更
         ExtStart=ext+1;
         ExtCount=0;
         //--- 次の反復に渡す
         continue;
        }
      //--- 極小をチェック
      if(IsMin(low,ext))
        {
         //--- 極値であるローソク足を強調表示する
         ExtOpen[ext]=open[ext];
         ExtHigh[ext]=high[ext];
         ExtLow[ext]=low[ext];
         ExtClose[ext]=close[ext];
         ExtColor[ext]=2;
         //---極値まで他のローソク足を中間色で強調表示する
         FillCandles(open,high,low,close);
         //--- 変数の値を変更
         ExtStart=ext+1;
         ExtCount=0;
         //--- 次の反復に渡す
         continue;
        }
      //--- 区間内の非極値の数を増やす
      ExtCount++;
     }
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//|  現在の配列要素が極大かどうかを確認する                          |
//+------------------------------------------------------------------+
bool IsMax(const double &price[],const int ind)
  {
//--- 区間開始の変数
   int i=ind-InpNum;
//--- 区間終了期間
   int finish=ind+InpNum+1;
//--- 区間の前半をチェックする
   for(;i<ind;i++)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      if(price[ind]<=price[i])
         return(false);
      }
//--- 区間の後半をチェックする
   for(i=ind+1;i<finish;i++)
     {
      if(price[ind]<=price[i])
         return(false);
      }
//--- 極値を発見
   return(true);
  }
//+------------------------------------------------------------------+
//| 現在の配列要素が極小かどうかを確認する                           |
//+------------------------------------------------------------------+
bool IsMin(const double &price[],const int ind)
  {
//--- 区間開始の変数
   int i=ind-InpNum;
//--- 区間終了の変数
   int finish=ind+InpNum+1;
//--- 区間の前半をチェックする
   for(;i<ind;i++)
     {
      if(price[ind]>=price[i])
         return(false);
      }
//--- 区間の後半をチェックする
   for(i=ind+1;i<finish;i++)
     {
      if(price[ind]>=price[i])
         return(false);
      }
//--- 極値を発見
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrayCompare

この関数は 2 つの同型の配列を比較した結果を返します。基本データ型 または 複合型オブジェクトを使用しないカスタム構造体、つまり文字列型、動的配列、クラスや他の複合オブジェクトで出来た構造体を含まないカスタム構造体です。

```
int  ArrayCompare(
   const void&  array1[],           // 1番目の配列
   const void&  array2[],           // 2 番目の配列
   int          start1=0,           // 1番目の配列の初期オフセット
   int          start2=0,           // 2 番目の配列の初期オフセット
   int          count=WHOLE_ARRAY   // 比較される要素の数
   );
```

パラメータ

array1[]

[in] 1番目の配列

array2[]

[in] 2 番目の配列

start1=0

[in] 比較開始点となる、1番目の配列内の要素の初期インデックス。デフォルトの開始インデックスは0です。

start2=0

[in] 比較開始点となる、2 番目の配列内の要素の初期インデックス。デフォルトの開始インデックスは0です。

count=WHOLE_ARRAY

[in] 比較される要素の数 デフォルトでは両方の配列の全ての要素が比較されます（count=WHOLE_ARRAY）。

戻り値

- -1（ array1[] が array2[] より小さい）
- 0（ array1[] とarray2[] が等しい）
- 1（ array1[] が array2[] より大きい）
- -2（両配列の型の非互換性や start1、start2 または count 値が配列の範囲外にあることによってエラーが発生）

注意事項

配列のサイズが異なり、いずれかの配列がもう 1 つの配列の忠実なサブセットであって、かつ count=WHOLE_ARRAY である場合、0 が返されず、両配列は等しいとみなされません。この場合、両配列のサイズ比較による戻り値は、array1[] のサイズが array2[] のサイズより小さい場合は -1 でそれ以外は1になります。

例：

```
//--- グローバル変数
```

© 2000-2025, MetaQuotes Ltd.

```
double   ExtArrayFirst[];
double   ExtArraySecond[];

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 配列サイズを設定する
   if(ArrayResize(ExtArrayFirst,10)!=10)
     {
      Print("ArrayResize() failed for ExtArrayFirst. Error code: ",GetLastError());
      return;
     }
   if(ArrayResize(ExtArraySecond,10)!=10)
     {
      Print("ArrayResize() failed for ExtArraySecond. Error code: ",GetLastError());
      return;
     }

//--- ループ内で配列に i, j のインデックスの値を入力する
   int total=ArraySize(ExtArrayFirst);
   for(int i=0, j=total-1; i<total; i++,j--)
     {
      //--- ExtArrayFirst配列に左から右へ入力する
      //--- ExtArraySecond配列に右から左へ入力する
      ExtArrayFirst[i]=i;
      ExtArraySecond[i]=j;
     }
//--- 配列を比較し、結果をログに出力する
   ArrayComparePrint(ExtArrayFirst,ExtArraySecond);
   /*
  結果:
   ExtArrayFirst:
   0.00000 1.00000 2.00000 3.00000 4.00000 5.00000 6.00000 7.00000 8.00000 9.00000
   ExtArraySecond:
   9.00000 8.00000 7.00000 6.00000 5.00000 4.00000 3.00000 2.00000 1.00000 0.00000
   Result ArrayCompare(): ExtArrayFirst is smaller than ExtArraySecond (result = -1)
   */

//--- 次に配列を反転する
//--- ループ内で配列に i, j のインデックスの値を入力する
   for(int i=0, j=total-1; i<total; i++,j--)
     {
      //--- ExtArrayFirst配列に右から左へ入力する
      //--- ExtArraySecond配列に左から右へ入力する
      ExtArrayFirst[i]=j;
      ExtArraySecond[i]=i;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 配列を比較し、結果をログに出力する
   ArrayComparePrint(ExtArrayFirst,ExtArraySecond);
   /*
  結果:
   ExtArrayFirst:
   9.00000 8.00000 7.00000 6.00000 5.00000 4.00000 3.00000 2.00000 1.00000 0.00000
   ExtArraySecond:
   0.00000 1.00000 2.00000 3.00000 4.00000 5.00000 6.00000 7.00000 8.00000 9.00000
   Result ArrayCompare(): ExtArrayFirst is larger than ExtArraySecond (result = 1)
   */

//--- 次に配列に1方向に入力する
//--- ループ内で配列に iのインデックスの値を入力する
   for(int i=0; i<total; i++)
     {
      //--- 両配列に左から右へ入力する
      ExtArrayFirst[i]=i;
      ExtArraySecond[i]=i;
     }
//--- 配列を比較し、結果をログに出力する
   ArrayComparePrint(ExtArrayFirst,ExtArraySecond);
   /*
  結果:
   ExtArrayFirst:
   0.00000 1.00000 2.00000 3.00000 4.00000 5.00000 6.00000 7.00000 8.00000 9.00000
   ExtArraySecond:
   0.00000 1.00000 2.00000 3.00000 4.00000 5.00000 6.00000 7.00000 8.00000 9.00000
   Result ArrayCompare(): ExtArrayFirst and ExtArraySecond are equal (result = 0)
   */
  }
//+------------------------------------------------------------------+
//|  結果を比較して表示する
//+------------------------------------------------------------------+
void ArrayComparePrint(const double &array1[], const double &array2[])
  {
   //--- 配列のヘッダとコンテンツを出力する
   Print("ExtArrayFirst:");
   ArrayPrint(array1);
   Print("ExtArraySecond:");
   ArrayPrint(array2);
   //--- 配列を比較し、比較結果を出力する
   int     res=ArrayCompare(array1,array2);
   string res_str=(res>0 ? "ExtArrayFirst is larger than ExtArraySecond" : res<0 ?
"ExtArrayFirst is smaller than ExtArraySecond" : "ExtArrayFirst and ExtArraySecond
are equal");
   PrintFormat("Result ArrayCompare(): %s (result = %d)\n",res_str,res);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

ArrayFree

動的配列のバッファを開放し、ゼロ次元のサイズを 0 に設定します。

```
void  ArrayFree(
   void&  array[]      // 配列
   );
```

パラメータ

array[]

[in] 動的配列

戻り値

なし

注意事項

全ての使用メモリは一度に解放され、また配列の主な作業は指標/バッファへのアクセスを含むことを考えると、ArrayFree() 関数を使用する必要性はあまり頻繁でありません。バッファサイズは自動的に端末のエグゼクティブサブシステムによって管理されています。

手動でアプリケーションの複雑な動的環境でメモリを管理する必要がある場合には ArrayFree() 関数は、ユーザが明示的に、すぐにすでに不要な動的配列によって占有されていたメモリを解放することを可能にします。

例:

```
#include <Controls\Dialog.mqh>
#include <Controls\Button.mqh>
#include <Controls\Label.mqh>
#include <Controls\ComboBox.mqh>
//--- 定義済み定数
#define X_START 0
#define Y_START 0
#define X_SIZE 280
#define Y_SIZE 300
//+------------------------------------------------------------------+
//| メモリを使用するための Dialog クラス                             |
//+------------------------------------------------------------------+
class CMemoryControl : public CAppDialog
  {
private:
   //--- 配列のサイズ
   int               m_arr_size;
   //--- 配列
   char              m_arr_char[];
   int               m_arr_int[];
   float             m_arr_float[];
   double            m_arr_double[];
   long              m_arr_long[];
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- ラベル
   CLabel            m_lbl_memory_physical;
   CLabel            m_lbl_memory_total;
   CLabel            m_lbl_memory_available;
   CLabel            m_lbl_memory_used;
   CLabel            m_lbl_array_size;
   CLabel            m_lbl_array_type;
   CLabel            m_lbl_error;
   CLabel            m_lbl_change_type;
   CLabel            m_lbl_add_size;
   //--- ボタン
   CButton           m_button_add;
   CButton           m_button_free;
   //--- コンボボックス
   CComboBox         m_combo_box_step;
   CComboBox         m_combo_box_type;
   //--- コンボボックスにおける配列型の現在値
   int               m_combo_box_type_value;

public:
                     CMemoryControl(void);
                    ~CMemoryControl(void);
   //--- クラスオブジェクトの作成方法
   virtual bool      Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2);
   //--- チャートイベントハンドラ
   virtual bool      OnEvent(const int id,const long &lparam,const double
&dparam,const string &sparam);

protected:
   //--- ラベルを作成する
   bool              CreateLabel(CLabel &lbl,const string name,const int x,const int
y,const string str,const int font_size,const int clr);
   //--- 制御要素を作成する
   bool              CreateButton(CButton &button,const string name,const int
x,const int y,const string str,const int font_size,const int clr);
   bool              CreateComboBoxStep(void);
   bool              CreateComboBoxType(void);
   //--- イベントハンドラ
   void              OnClickButtonAdd(void);
   void              OnClickButtonFree(void);
   void              OnChangeComboBoxType(void);
   //--- 現在の配列を使用するメソッド
   void              CurrentArrayFree(void);
   bool              CurrentArrayAdd(void);
  };
//+------------------------------------------------------------------+
//| 現在の配列のメモリを解放する                                     |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void CMemoryControl::CurrentArrayFree(void)
  {
//--- 配列サイズをリセットする
   m_arr_size=0;
//--- 配列を解放する
   if(m_combo_box_type_value==0)
      ArrayFree(m_arr_char);
   if(m_combo_box_type_value==1)
      ArrayFree(m_arr_int);
   if(m_combo_box_type_value==2)
      ArrayFree(m_arr_float);
   if(m_combo_box_type_value==3)
      ArrayFree(m_arr_double);
   if(m_combo_box_type_value==4)
      ArrayFree(m_arr_long);
  }
//+------------------------------------------------------------------+
//| 現在の配列へのメモリ追加を試みる                                 |
//+------------------------------------------------------------------+
bool CMemoryControl::CurrentArrayAdd(void)
  {
//--- 使用メモリサイズが物理メモリサイズを超えた場合終了する
   if
(TerminalInfoInteger
(TERMINAL_MEMORY_PHYSICAL)/TerminalInfoInteger(TERMINAL_MEMORY_USED)<2)
      return(false);
//--- 現在のタイプに応じてのメモリの割り当てを試みる
   if(m_combo_box_type_value==0 && ArrayResize(m_arr_char,m_arr_size)==-1)
      return(false);
   if(m_combo_box_type_value==1 && ArrayResize(m_arr_int,m_arr_size)==-1)
      return(false);
   if(m_combo_box_type_value==2 && ArrayResize(m_arr_float,m_arr_size)==-1)
      return(false);
   if(m_combo_box_type_value==3 && ArrayResize(m_arr_double,m_arr_size)==-1)
      return(false);
   if(m_combo_box_type_value==4 && ArrayResize(m_arr_long,m_arr_size)==-1)
      return(false);
//--- メモリ割り当てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//| イベントハンドル                                                 |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CMemoryControl)
ON_EVENT(ON_CLICK,m_button_add,OnClickButtonAdd)
ON_EVENT(ON_CLICK,m_button_free,OnClickButtonFree)
ON_EVENT(ON_CHANGE,m_combo_box_type,OnChangeComboBoxType)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| コンストラクタ                                                   |
//+------------------------------------------------------------------+
CMemoryControl::CMemoryControl(void)
  {
  }
//+------------------------------------------------------------------+
//| デストラクタ                                                     |
//+------------------------------------------------------------------+
CMemoryControl::~CMemoryControl(void)
  {
  }
//+------------------------------------------------------------------+
//| クラスオブジェクトの作成メソッド                                 |
//+------------------------------------------------------------------+
bool CMemoryControl::Create(const long chart,const string name,const int subwin,
                            const int x1,const int y1,const int x2,const int y2)
  {
//--- 基本クラスオブジェクトを作成する
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
ラ//--- 文字列をラベル用に準備する
   string str_physical="Memory physical = "+(string)
TerminalInfoInteger(TERMINAL_MEMORY_PHYSICAL)+" Mb";
   string str_total="Memory total = "+(string)
TerminalInfoInteger(TERMINAL_MEMORY_TOTAL)+" Mb";
   string str_available="Memory available = "+(string)
TerminalInfoInteger(TERMINAL_MEMORY_AVAILABLE)+" Mb";
   string str_used="Memory used = "+(string)
TerminalInfoInteger(TERMINAL_MEMORY_USED)+" Mb";
//--- ラベルを作成する
   if(!CreateLabel(m_lbl_memory_physical,"physical_label",X_START+10,Y_START+5,str_p
hysical,12,clrBlack))
      return(false);
   if(!CreateLabel(m_lbl_memory_total,"total_label",X_START+10,Y_START+30,str_total,
12,clrBlack))
      return(false);
   if(!CreateLabel(m_lbl_memory_available,"available_label",X_START+10,Y_START+55,st
r_available,12,clrBlack))
      return(false);
   if(!CreateLabel(m_lbl_memory_used,"used_label",X_START+10,Y_START+80,str_used,12,
clrBlack))
      return(false);
   if(!CreateLabel(m_lbl_array_type,"type_label",X_START+10,Y_START+105,"Array type
= double",12,clrBlack))
      return(false);
   if(!CreateLabel(m_lbl_array_size,"size_label",X_START+10,Y_START+130,"Array size
= 0",12,clrBlack))
      return(false);
   if(!CreateLabel(m_lbl_error,"error_label",X_START+10,Y_START+155,"",12,clrRed))
      return(false);
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!CreateLabel(m_lbl_change_type,"change_type_label",X_START+10,Y_START+185,"Cha
nge type",10,clrBlack))
      return(false);
   if(!CreateLabel(m_lbl_add_size,"add_size_label",X_START+10,Y_START+210,"Add to
array",10,clrBlack))
      return(false);
//--- 制御要素を作成する
   if(!CreateButton(m_button_add,"add_button",X_START+15,Y_START+245,"Add",12,clrBlu
e))
      return(false);
   if(!CreateButton(m_button_free,"free_button",X_START+75,Y_START+245,"Free",12,clr
Blue))
      return(false);
   if(!CreateComboBoxType())
      return(false);
   if(!CreateComboBoxStep())
      return(false);
//--- 変数を初期化する
   m_arr_size=0;
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| ボタンを作成する                                                 |
//+------------------------------------------------------------------+
bool CMemoryControl::CreateButton(CButton &button,const string name,const int x,
                                  const int y,const string str,const int font_size,
                                  const int clr)
  {
//--- ボタンを作成する
   if(!button.Create(m_chart_id,name,m_subwin,x,y,x+50,y+20))
      return(false);
//--- テキスト
   if(!button.Text(str))
      return(false);
//--- フォントサイズ
   if(!button.FontSize(font_size))
      return(false);
//--- ラベルの色
   if(!button.Color(clr))
      return(false);
//--- ボタンを制御要素に加える
   if(!Add(button))
      return(false);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 配列サイズのコンボボックスを作成する                             |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool CMemoryControl::CreateComboBoxStep(void)
  {
//--- コンボボックスを作成する
   if(!m_combo_box_step.Create(m_chart_id,"step_combobox",m_subwin,X_START+100,Y_STA
RT+185,X_START+200,Y_START+205))
      return(false);
//--- コンボボックスに要素を加える
   if(!m_combo_box_step.ItemAdd("100 000",100000))
      return(false);
   if(!m_combo_box_step.ItemAdd("1 000 000",1000000))
      return(false);
   if(!m_combo_box_step.ItemAdd("10 000 000",10000000))
      return(false);
   if(!m_combo_box_step.ItemAdd("100 000 000",100000000))
      return(false);
//--- 現在のコンボボックス要素を設定する
   if(!m_combo_box_step.SelectByValue(1000000))
      return(false);
//---コンボボックス制御要素に加える
   if(!Add(m_combo_box_step))
      return(false);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 配列型コンボボックスを作成する                                   |
//+------------------------------------------------------------------+
bool CMemoryControl::CreateComboBoxType(void)
  {
//--- コンボボックスを作成する
   if(!m_combo_box_type.Create(m_chart_id,"type_combobox",m_subwin,X_START+100,Y_STA
RT+210,X_START+200,Y_START+230))
      return(false);
//--- コンボボックスに要素を加える
   if(!m_combo_box_type.ItemAdd("char",0))
      return(false);
   if(!m_combo_box_type.ItemAdd("int",1))
      return(false);
   if(!m_combo_box_type.ItemAdd("float",2))
      return(false);
   if(!m_combo_box_type.ItemAdd("double",3))
      return(false);
   if(!m_combo_box_type.ItemAdd("long",4))
      return(false);
//--- 現在のコンボボックス要素を設定する
   if(!m_combo_box_type.SelectByValue(3))
      return(false);
//--- 現在のコンボボックス要素を格納する
```

© 2000-2025, MetaQuotes Ltd.

```
   m_combo_box_type_value=3;
//---コンボボックス制御要素に加える
   if(!Add(m_combo_box_type))
      return(false);
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//| ラベルを作成する                                                 |
//+------------------------------------------------------------------+
bool CMemoryControl::CreateLabel(CLabel &lbl,const string name,const int x,
                                 const int y,const string str,const int font_size,
                                 const int clr)
  {
//--- ラベルを作成する
   if(!lbl.Create(m_chart_id,name,m_subwin,x,y,0,0))
      return(false);
//--- テキスト
   if(!lbl.Text(str))
      return(false);
//--- フォントサイズ
   if(!lbl.FontSize(font_size))
      return(false);
//--- 色
   if(!lbl.Color(clr))
      return(false);
//--- ラベルを制御要素に加える
   if(!Add(lbl))
      return(false);
//--- 成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 「Add(追加)」ボタンクリックのイベントハンドラ                    |
//+------------------------------------------------------------------+
void CMemoryControl::OnClickButtonAdd(void)
  {
//---配列サイズを増やす
   m_arr_size+=(int)m_combo_box_step.Value();
//--- 現在の配列へのメモリ追加を試みる
   if(CurrentArrayAdd())
     {
      //--- メモリ割り当て成功し、画面上に現在のステータスを表示する
      m_lbl_memory_available.Text("Memory available = "+(string)
TerminalInfoInteger(TERMINAL_MEMORY_AVAILABLE)+" Mb");
      m_lbl_memory_used.Text("Memory used = "+(string)
TerminalInfoInteger(TERMINAL_MEMORY_USED)+" Mb");
      m_lbl_array_size.Text("Array size = "+IntegerToString(m_arr_size));
      m_lbl_error.Text("");
```

© 2000-2025, MetaQuotes Ltd.

```
      }
    else
      {
       //--- メモリ割り当て失敗し、エラーメッセージを表示する
       m_lbl_error.Text("Array is too large, error!");
       //--- 以前の配列サイズを返す
       m_arr_size-=(int)m_combo_box_step.Value();
      }
   }
//+------------------------------------------------------------------+
//| 「Free(解放)」ボタンクリックのイベントハンドラ                   |
//+------------------------------------------------------------------+
void CMemoryControl::OnClickButtonFree(void)
   {
//--- 現在の配列のメモリを解放する
    CurrentArrayFree();
//--- 画面上に現在のステータスを表示する。
    m_lbl_memory_available.Text("Memory available = "+(string)
TerminalInfoInteger(TERMINAL_MEMORY_AVAILABLE)+" Mb");
    m_lbl_memory_used.Text("Memory used = "+(string)
TerminalInfoInteger(TERMINAL_MEMORY_USED)+" Mb");
    m_lbl_array_size.Text("Array size = 0");
    m_lbl_error.Text("");
   }
//+------------------------------------------------------------------+
//| コンボボックス変更のイベントハンドラ                             |
//+------------------------------------------------------------------+
void CMemoryControl::OnChangeComboBoxType(void)
   {
//--- 配列系が変わったかをチェックする
    if(m_combo_box_type.Value()!=m_combo_box_type_value)
      {
       //--- 現在の配列のメモリを解放する
       OnClickButtonFree();
       //--- 他のタイプの配列を使用する
       m_combo_box_type_value=(int)m_combo_box_type.Value();
       //--- 新しい配列型を画面上の表示する
       if(m_combo_box_type_value==0)
          m_lbl_array_type.Text("Array type = char");
       if(m_combo_box_type_value==1)
          m_lbl_array_type.Text("Array type = int");
       if(m_combo_box_type_value==2)
          m_lbl_array_type.Text("Array type = float");
       if(m_combo_box_type_value==3)
          m_lbl_array_type.Text("Array type = double");
       if(m_combo_box_type_value==4)
          m_lbl_array_type.Text("Array type = long");
      }
   }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- CMemoryControl クラスオブジェクト
CMemoryControl ExtDialog;
//+------------------------------------------------------------------+
//| エキスパート初期化に使用される関数                                |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- ダイアログを作成する
   if(!ExtDialog.Create(0,"MemoryControl",0,X_START,Y_START,X_SIZE,Y_SIZE))
      return(INIT_FAILED);
//--- 実行
   ExtDialog.Run();
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| エキスパート初期化解除に使用される関数                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| エキスパートチャートイベント関数                                  |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrayGetAsSeries

配列の索引付けの方向をチェックします。

```
bool  ArrayGetAsSeries(
   const void&  array[]    // チェックされる配列
   );
```

パラメータ

array

[in] チェックされた配列

戻り値

指定された配列にAS_SERIESフラグが設定され、すなわち配列へのアクセスは、時系列のように前後反対に行われる場合 trueを返します。時系列と通常の配列の違いは、時系列のインデックスが後ろから前へ(最新のデータから古いデータへ)つけられていることです。

注意事項

配列が時系列であるかどうかをチェックするには ArrayIsSeries() 関数を使用します。OnCalculate() 関数に入力パラメータとして渡された価格データの配列は時系列と同じインデックス方向を持っていません。必要なインデックス方向は ArraySetAsSeries() 関数を使用して設定出来ます。

例:

```
#property description "Indicator calculates absolute values of the difference
between"
#property description "Open and Close or High and Low prices displaying them in a
separate subwindow"
#property description "as a histrogram."
//--- 指標の設定
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- プロット
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_style1  STYLE_SOLID
#property indicator_width1  3
//--- 入力パラメータ
input bool InpAsSeries=true; // 指標バッファでの索引付けの方向
input bool InpPrices=true;   // 計算価格(true - 始/終、false - 高/低)
//--- 指標バッファ
double ExtBuffer[];
//+------------------------------------------------------------------+
//|  指標値の計算                                                    |
//+------------------------------------------------------------------+
void CandleSizeOnBuffer(const int rates_total,const int prev_calculated,
                        const double &first[],const double &second[],double
&buffer[])
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- バーを計算する start 変数
   int start=prev_calculated;
//--- 指標値が前のティックで計算されている場合には最後のバーを使用する
   if(prev_calculated>0)
      start--;
//--- 配列の索引付けの方向を定義する
   bool as_series_first=ArrayGetAsSeries(first);
   bool as_series_second=ArrayGetAsSeries(second);
   bool as_series_buffer=ArrayGetAsSeries(buffer);
//--- 必要に応じて、索引付け方向を直接なものにする
   if(as_series_first)
      ArraySetAsSeries(first,false);
   if(as_series_second)
      ArraySetAsSeries(second,false);
   if(as_series_buffer)
      ArraySetAsSeries(buffer,false);
//--- 指標値を計算する
   for(int i=start;i<rates_total;i++)
      buffer[i]=MathAbs(first[i]-second[i]);
  }
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 指標バッファを結合する
   SetIndexBuffer(0,ExtBuffer);
//--- 指標バッファのインデックス要素を設定する
   ArraySetAsSeries(ExtBuffer,InpAsSeries);
//--- 指標が計算されている価格をチェックする
   if(InpPrices)
     {
      //--- 価格を開け閉めする
      PlotIndexSetString(0,PLOT_LABEL,"BodySize");
      //--- 指標の色を設定する
      PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrOrange);
     }
   else
     {
      //--- 高値と安値
      PlotIndexSetString(0,PLOT_LABEL,"ShadowSize");
      //--- 指標の色を設定する
      PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrDodgerBlue);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---フラグの値に応じた指標を計算する
   if(InpPrices)
      CandleSizeOnBuffer(rates_total,prev_calculated,open,close,ExtBuffer);
   else
      CandleSizeOnBuffer(rates_total,prev_calculated,high,low,ExtBuffer);
//--- 次の呼び出しのためにprev_calculated の値を返す
   return(rates_total);
  }
```

参照

時系列へのアクセス、ArraySetAsSeries

© 2000–2025, MetaQuotes Ltd.

ArrayInitialize

数値配列の全ての要素に同じ値を設定します。

char 型配列の初期化には

```
int  ArrayInitialize(
   char     array[],      // 初期化された配列
   char     value         // 初期化に使用される
   );
```

short 型配列の初期化には

```
int  ArrayInitialize(
   short    array[],      // 初期化された配列
   short    value         // 初期化に使用される値
   );
```

int 型配列の初期化には

```
int  ArrayInitialize(
   int      array[],      // 初期化された配列
   int      value         // 初期化に使用される値
   );
```

long 型配列の初期化には

```
int  ArrayInitialize(
   long     array[],      // 初期化された配列
   long     value         // 初期化に使用される値
   );
```

float 型配列の初期化には

```
int  ArrayInitialize(
   float    array[],      // 初期化された配列
   float    value         // 初期化に使用される値
   );
```

double 型配列の初期化には

```
int  ArrayInitialize(
   double   array[],      // 初期化された配列
   double   value         // 初期化に使用される値
   );
```

bool 型配列の初期化には

```
int  ArrayInitialize(
   bool     array[],      // 初期化された配列
   bool     value         // 初期化に使用される値
   );
```

© 2000-2025, MetaQuotes Ltd.

uint 型配列の初期化には

```
int  ArrayInitialize(
   uint     array[],      // 初期化された配列
   uint     value         // 初期化に使用される値
   );
```

パラメータ

array[]

[out] 初期化を必要とする数値配列

value

[in] 全ての配列要素に設定する新しい値

戻り値

初期化された要素数

注意事項

ArrayResize() 関数は物理的なメモリの再配置を伴わずに配列のサイズと更なる拡大のための予備を設定することが出来ます。メモリ再配置の操作低速であるために、パフォーマンス向上のために実装されています。

ArrayInitialize(array, init_val) を使用して配列を初期化した場合、予備要素の値はこちらの配列で割り当てられたものとは違います。 ArrayResize() を使用して更に*配列*が拡大された場合、新しい要素は配列の末尾に追加されます。その値は定義されず通常 *init_value* とは異なります。

例:

```
void OnStart()
  {
//--- 動的配列
   double array[];
//--- 100 要素の配列のサイズを設定し、別の 10 要素のバッファを確保
   ArrayResize(array,100,10);
//--- EMPTY_VALUE=DBL_MAX で配列要素を初期化
   ArrayInitialize(array,EMPTY_VALUE);
   Print("Values of 10 last elements after initialization");
   for(int i=90;i<100;i++) printf("array[%d] = %G",i,array[i]);
//--- 5 要素で拡張
   ArrayResize(array,105);
   Print("Values of 10 last elements after ArrayResize(array,105)");
//--- 最後の 5 要素の値は予備バッファから取得
   for(int i=95;i<105;i++) printf("array[%d] = %G",i,array[i]);
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrayFill

この関数は指定された値で配列を埋めます。

```
void  ArrayFill(
   void&  array[],        // 配列
   int    start,          // 開始インデックス
   int    count,          // 埋める要素の数
   void   value           // 値
   );
```

パラメータ

array[]

[out] 基本データ型（char、uchar、short、ushort、int、uint、long、ulong、bool、color、datetime、float、double）の配列

start

[in] 開始インデックス。この場合、指定された AS_SERIES フラグは無視されます。

count

[in] 埋められる要素の数

value

[in] 配列を埋める値

戻り値

なし

注意事項

ArrayFill() 関数が呼ばれる時は通常のインデックス方向（左から右）が常に暗黙的に指定されます。ArraySetAsSeries() 関数を使用しての配列要素へのアクセスの順序の変更は無視されるということです。

ArrayFill() 関数によって処理される時には多次元配列は 1 次元として示されています。例えば array[2][4] は array[8] として処理されます。従って、この配列を操作する時初期要素のインデックスを 5 に等しくなるように指定することが出来ます。ArrayFill(array, 5, 2, 3.14) がarray[2][4] に呼ばれたならばarray[1][1] と array[1][2] が 3.14 で埋められます。

例：

```
void OnStart()
  {
//--- 動的配列を宣言する
   int a[];
//--- サイズを定める
   ArrayResize(a,10);
//--- 最初の 5 要素に 123 を書き込む
   ArrayFill(a,0,5,123);
//--- 次の 5 要素に 456 を書き込む
   ArrayFill(a,5,5,456);
//--- 値を表示する
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0;i<ArraySize(a);i++) printf("a[%d] = %d",i,a[i]);
  }
```

© 2000–2025, MetaQuotes Ltd.

ArrayIsDynamic

この関数は配列が動的であるかどうかをチェックします。

```
bool  ArrayIsDynamic(
   const void&  array[]     // チェックされた配列
   );
```

パラメータ

array[]

[in] チェックされた配列

戻り値

選ばれた配列が動的の場合は true、それ以外の場合は false

例:

```
#property description "This indicator does not calculate values. It makes a single attempt to"
#property description "apply the call of ArrayFree() function to three arrays: dynamic one, static one and"
#property description "an indicator buffer. Results are shown in Experts journal."
//--- 指標の設定
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- グローバル変数
double ExtDynamic[];   // 動的配列
double ExtStatic[100]; // 静的配列
bool   ExtFlag=true;   // フラグ
double ExtBuff[];      // 指標バッファ
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列へのメモリ追加
   ArrayResize(ExtDynamic,100);
//--- 指標バッファマッピング
   SetIndexBuffer(0,ExtBuff);
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
```

© 2000-2025, MetaQuotes Ltd.

```
                const int begin,
                const double &price[])
  {
//--- 単一の分析を行う
   if(ExtFlag)
     {
      //--- メモリ解除を試みる
      //--- 1. 動的配列
      Print("+============================+");
      Print("1. Check dynamic array:");
      Print("Size before memory is freed = ",ArraySize(ExtDynamic));
      Print("Is this a dynamic array = ",ArrayIsDynamic(ExtDynamic) ? "Yes" : "No");
      //--- 配列メモリ解除を試みる
      ArrayFree(ExtDynamic);
      Print("Size after memory is freed = ",ArraySize(ExtDynamic));
      //--- 2. 静的配列
      Print("2. Check static array:");
      Print("Size before memory is freed = ",ArraySize(ExtStatic));
      Print("Is this a dynamic array = ",ArrayIsDynamic(ExtStatic) ? "Yes" : "No");
      //--- 配列メモリ解除を試みる
      ArrayFree(ExtStatic);
      Print("Size after memory is freed = ",ArraySize(ExtStatic));
      //--- 3. 指標バッファ
      Print("3. Check indicator buffer:");
      Print("Size before memory is freed = ",ArraySize(ExtBuff));
      Print("Is this a dynamic array = ",ArrayIsDynamic(ExtBuff) ? "Yes" : "No");
      //--- 配列メモリ解除を試みる
      ArrayFree(ExtBuff);
      Print("Size after memory is freed = ",ArraySize(ExtBuff));
      //--- フラグの値を変更する
      ExtFlag=false;
     }
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
```

参照

[時系列と指標へのアクセス]

© 2000-2025, MetaQuotes Ltd.

ArrayIsSeries

この関数は配列が時系列であるかどうかをチェックします。

```
bool  ArrayIsSeries(
   const void&  array[]     // チェックされた配列
   );
```

パラメータ

array[]

[in] チェックされた配列

戻り値

チェックされた配列が時系列の場合は true、それ以外の場合は false を返します。OnCalculate() 関数にパラメータとして渡された配列は ArrayGetAsSeries() によって要素アクセスの順序をチェックしなければいけません。

例:

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- Label 1 をプロットする
#property indicator_label1  "Label1"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 指標バッファ
double         Label1Buffer[];
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- 指標バッファマッピング
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
//---
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---
   if(ArrayIsSeries(open))
      Print("open[] is timeseries");
   else
      Print("open[] is not timeseries!!!");
//--- 次の呼び出しのためにprev_calculated の値を返す
   return(rates_total);
  }
```

参照

時系列と指標へのアクセス

© 2000-2025, MetaQuotes Ltd.

ArrayMaximum

多次元数値配列の初次元で最大の要素を検索します。

```
int  ArrayMaximum(
   const void&    array[],              // 検索される配列
   int            start=0,              // チェック開始インデックス
   int            count=WHOLE_ARRAY     // チェックされた要素数
   );
```

パラメータ

array[]

[in] 検索対象の数値配列

start=0

[in] チェック開始インデックス

count=WHOLE_ARRAY

[in] 検索される要素数。デフォルトでは配列全体が検索されます（count=WHOLE_ARRAY）。

戻り値

配列の索引付け方向を考慮して見つかった要素のインデックスを返します。失敗した場合は、-1 が返されます。

注意事項

最大値を検索している際に AS_SERIES フラグ値が考慮されます。

ArrayMaximum と ArrayMinimum はパラメータとして任意の次元の配列を受け入れます。しかし、検索は常に最初（ゼロ）の次元に適用されます。

例：

```
#property description "The indicator displays larger timeframe's candlesticks on the current one."
//--- 指標の設定
#property indicator_chart_window
#property indicator_buffers 16
#property indicator_plots   8
//---- プロット1
#property indicator_label1  "BearBody"
#property indicator_color1  clrSeaGreen,clrSeaGreen
//---- プロット 2
#property indicator_label2  "BearBodyEnd"
#property indicator_color2  clrSeaGreen,clrSeaGreen
//---- プロット 3
#property indicator_label3  "BearShadow"
#property indicator_color3  clrSalmon,clrSalmon
//---- プロット 4
#property indicator_label4  "BearShadowEnd"
#property indicator_color4  clrSalmon,clrSalmon
```

© 2000-2025, MetaQuotes Ltd.

```
//---- プロット 5
#property indicator_label5  "BullBody"
#property indicator_color5  clrOlive,clrOlive
//---- プロット 6
#property indicator_label6  "BullBodyEnd"
#property indicator_color6  clrOlive,clrOlive
//---- プロット7
#property indicator_label7  "BullShadow"
#property indicator_color7  clrSkyBlue,clrSkyBlue
//---- プロット8
#property indicator_label8  "BullShadowEnd"
#property indicator_color8  clrSkyBlue,clrSkyBlue
//--- 定義済み定数
#define INDICATOR_EMPTY_VALUE 0.0
//--- 入力パラメータ
input ENUM_TIMEFRAMES InpPeriod=PERIOD_H4;              // 指標計算のための時間軸
input datetime        InpDateStart=D'2013.01.01 00:00'; // 分析開始日
//--- 弱気ローソク足の指標バッファ
double   ExtBearBodyFirst[];
double   ExtBearBodySecond[];
double   ExtBearBodyEndFirst[];
double   ExtBearBodyEndSecond[];
double   ExtBearShadowFirst[];
double   ExtBearShadowSecond[];
double   ExtBearShadowEndFirst[];
double   ExtBearShadowEndSecond[];
//--- 強気ローソク足の指標バッファ
double   ExtBullBodyFirst[];
double   ExtBullBodySecond[];
double   ExtBullBodyEndFirst[];
double   ExtBullBodyEndSecond[];
double   ExtBullShadowFirst[];
double   ExtBullShadowSecond[];
double   ExtBullShadowEndFirst[];
double   ExtBullShadowEndSecond[];
//--- グローバル変数
datetime ExtTimeBuff[];      // 大きめの時間軸の時間バッファ
int      ExtSize=0;          // 時間バッファサイズ
int      ExtCount=0;         // 時間バッファ内のインデックス
int      ExtStartPos=0;      // 指標計算の初期位置
bool     ExtStartFlag=true;  // 初期位置を受信するための補助フラグ
datetime ExtCurrentTime[1];  // 大きめの時間軸のバーの生成の最終時間
datetime ExtLastTime;        // 計算の行われた大きめの時間軸の最終時間
bool     ExtBearFlag=true;   // 弱気指標バッファにデータを書き込む順序を定義するためのフラグ
bool     ExtBullFlag=true;   // 強気指標バッファにデータを書き込む順序を定義するためのフラグ
int      ExtIndexMax=0;      // 配列内の最大要素のインデックス
int      ExtIndexMin=0;      // 配列内の最小要素のインデックス
int      ExtDirectionFlag=0; // 現在のローソク足の価格移動方向
//--- 正しく描画するためにローソク足の始値と終値間をシフトする
```

© 2000-2025, MetaQuotes Ltd.

```
const double ExtEmptyBodySize=0.2*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//+------------------------------------------------------------------+
//|  ローソク足の基本的な部分を埋める                                |
//+------------------------------------------------------------------+
void FillCandleMain(const double &open[],const double &close[],
                    const double &high[],const double &low[],
                    const int start,const int last,const int fill_index,
                    int &index_max,int &index_min)
  {
//--- 配列内の最大と最小要素のインデックスを見つける
   index_max=ArrayMaximum(high,ExtStartPos,last-start+1); // 最高
   index_min=ArrayMinimum(low,ExtStartPos,last-start+1);  // 最低
//--- 現在の時間軸からいくつのバーを埋めるかを定義する
   int count=fill_index-start+1;
//--- 最初のバーの終値が最後のバーでの終値を超えた場合、ローソク足は弱気となる
   if(open[start]>close[last])
     {
      //--- ローソク足が以前に強気だった場合は強気指標バッファの値をクリアする
      if(ExtDirectionFlag!=-1)
         ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullSh
adowSecond,start,count);
      //--- 弱気ローソク足
      ExtDirectionFlag=-1;
      //--- ローソク足を作成する
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearSh
adowSecond,open[start],
                     close[last],high[index_max],low[index_min],start,count,ExtBearF
lag);
      //--- 関数を終了する
      return;
     }
//--- 最初のバーの終値が最後のバーでの終値より低い場合、ローソク足は強気となる
   if(open[start]<close[last])
     {
      //--- ローソク足が以前に強気だった場合は強気指標バッファの値をクリアする
      if(ExtDirectionFlag!=1)
         ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearSh
adowSecond,start,count);
      //--- 強気ローソク足
      ExtDirectionFlag=1;
      //--- ローソク足を作成する
      FormCandleMain(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullSh
adowSecond,close[last],
                     open[start],high[index_max],low[index_min],start,count,ExtBullF
lag);
      //--- 関数を終了する
      return;
     }
//--- 関数のこの部分が実行される場合、最初のバーでの始値は最後のバーで終値に等しい。
//--- このようなローソク足は弱気であると考えられる
```

© 2000-2025, MetaQuotes Ltd.

```
//--- ローソク足が以前に強気だった場合は強気指標バッファの値をクリアする
   if(ExtDirectionFlag!=-1)
      ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowSecond,start,count);
//--- 弱気ローソク足
   ExtDirectionFlag=-1;
//--- 終値と始値が等しい場合、シフトをつくって正しく描画する
   if(high[index_max]!=low[index_min])
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSecond,open[start],
                     open[start]-ExtEmptyBodySize,high[index_max],low[index_min],start,count,ExtBearFlag);
   else
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSecond,
                     open[start],open[start]-ExtEmptyBodySize,high[index_max],
                     high[index_max]-ExtEmptyBodySize,start,count,ExtBearFlag);
  }
//+------------------------------------------------------------------+
//|  ローソク足の終わりを埋める                                      |
//+------------------------------------------------------------------+
void FillCandleEnd(const double &open[],const double &close[],
                   const double &high[],const double &low[],
                   const int start,const int last,const int fill_index,
                   const int index_max,const int index_min)
  {
//--- シングルバーの場合は描画しない
   if(last-start==0)
      return;
//--- 最初のバーの終値が最後のバーでの終値を超えた場合、ローソク足は弱気となる
   if(open[start]>close[last])
     {
      //--- ローソク足の終わりを作成する
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,ExtBearShadowEndSecond,
                    open[start],close[last],high[index_max],low[index_min],fill_index,ExtBearFlag);
      //--- 関数を終了する
      return;
     }
//--- 最初のバーの終値が最後のバーでの終値より低い場合、ローソク足は強気となる
   if(open[start]<close[last])
     {
      //--- ローソク足の終わりを作成する
      FormCandleEnd(ExtBullBodyEndFirst,ExtBullBodyEndSecond,ExtBullShadowEndFirst,ExtBullShadowEndSecond,
                    close[last],open[start],high[index_max],low[index_min],fill_index,ExtBullFlag);
      //--- 関数を終了する
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- 関数のこの部分が実行される場合、最初のバーでの始値は最後のバーで終値に等しい。
//--- このようなローソク足は弱気であると考えられる
//--- ローソク足の終わりを作成する
   if(high[index_max]!=low[index_min])
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,E
xtBearShadowEndSecond,open[start],
                    open[start]-
ExtEmptyBodySize,high[index_max],low[index_min],fill_index,ExtBearFlag);
   else
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,E
xtBearShadowEndSecond,open[start],
                    open[start]-ExtEmptyBodySize,high[index_max],high[index_max]-
ExtEmptyBodySize,fill_index,ExtBearFlag);
  }
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 指標期間をチェックする
   if(!CheckPeriod((int)Period(),(int)InpPeriod))
      return(INIT_PARAMETERS_INCORRECT);
//--- 価格データ前景に表示する
   ChartSetInteger(0,CHART_FOREGROUND,0,1);
//--- 指標バッファを結合する
   SetIndexBuffer(0,ExtBearBodyFirst);
   SetIndexBuffer(1,ExtBearBodySecond);
   SetIndexBuffer(2,ExtBearBodyEndFirst);
   SetIndexBuffer(3,ExtBearBodyEndSecond);
   SetIndexBuffer(4,ExtBearShadowFirst);
   SetIndexBuffer(5,ExtBearShadowSecond);
   SetIndexBuffer(6,ExtBearShadowEndFirst);
   SetIndexBuffer(7,ExtBearShadowEndSecond);
   SetIndexBuffer(8,ExtBullBodyFirst);
   SetIndexBuffer(9,ExtBullBodySecond);
   SetIndexBuffer(10,ExtBullBodyEndFirst);
   SetIndexBuffer(11,ExtBullBodyEndSecond);
   SetIndexBuffer(12,ExtBullShadowFirst);
   SetIndexBuffer(13,ExtBullShadowSecond);
   SetIndexBuffer(14,ExtBullShadowEndFirst);
   SetIndexBuffer(15,ExtBullShadowEndSecond);
//--- 指標作成のためにいくつかのプロパティ値を設定する
   for(int i=0;i<8;i++)
     {
      PlotIndexSetInteger(i,PLOT_DRAW_TYPE,DRAW_FILLING); // グラフィカルな構成タイプ
      PlotIndexSetInteger(i,PLOT_LINE_STYLE,STYLE_SOLID); // 描画線のスタイル
      PlotIndexSetInteger(i,PLOT_LINE_WIDTH,1);           // 描画線の幅
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- 計算されたバーがまだない場合
   if(prev_calculated==0)
     {
      //--- 大きめの時間軸のバーの到着時間を受け取る
      if(!GetTimeData())
         return(0);
     }
//--- 直接のインデックス作成を設定する
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(close,false);
//--- バーを計算する start 変数
   int start=prev_calculated;
//--- バーが作成されたらその指標値を再計算する
   if(start!=0 && start==rates_total)
      start--;
//--- 指標値を計算するためのループ
   for(int i=start;i<rates_total;i++)
     {
      //--- 指標バッファの i 要素を空の値で埋める
      FillIndicatorBuffers(i);
      //--- InpDateStart 日からバーの計算を行う
      if(time[i]>=InpDateStart)
        {
         //--- 値が表示される元となる位置を初めて定義する
         if(ExtStartFlag)
           {
            //--- 初期のバーの値を収納する
            ExtStartPos=i;
            //--- 大き目な時間軸でtime[i] を超える最初の日を定義する
```

© 2000-2025, MetaQuotes Ltd.

```
            while(time[i]>=ExtTimeBuff[ExtCount])
               if(ExtCount<ExtSize-1)
                  ExtCount++;
            //--- 再びこのブロックに遭遇しないようにフラグの値を変更する
            ExtStartFlag=false;
           }
         //--- 配列にまだ要素が残っているかをチェックする
         if(ExtCount<ExtSize)
           {
            //--- 現在の時間軸の値が大きなな時間軸の値に届くまで待つ
            if(time[i]>=ExtTimeBuff[ExtCount])
              {
               //--- ( 最後と最後から2 番目のバーの間の領域を記入せずに) ローソク足の主な部分を描画する
               FillCandleMain(open,close,high,low,ExtStartPos,i-1,i-
2,ExtIndexMax,ExtIndexMin);
               //--- ローソク足の終わりの部分を埋める ( 最後と最後から2 番目のバーの間の領域)
               FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-
1,ExtIndexMax,ExtIndexMin);
               //--- 次のローソク足を描画するために初期位置を移動する
               ExtStartPos=i;
               //--- 配列カウンタを増加する
               ExtCount++;
              }
            else
               continue;
           }
         else
           {
            //--- 配列の値をリセットする
            ResetLastError();
            //--- 大き目な時間軸の最終日を受け取る
            if(CopyTime(Symbol(),InpPeriod,0,1,ExtCurrentTime)==-1)
              {
               Print("Data copy error, code = ",GetLastError());
               return(0);
              }
            //---新しいデータが後に来る場合には、ローソク足の作成を中止する
            if(ExtCurrentTime[0]>ExtLastTime)
              {
               //--- 主要指標バッファの最後と最後から2 番目のバーの間の領域をクリアする
               ClearEndOfBodyMain(i-1);
               //--- 補助指標バッファを使用して領域に記入する
               FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-
1,ExtIndexMax,ExtIndexMin);
               //--- 次のローソク足を描画するために初期位置を移動する
               ExtStartPos=i;
               //--- 価格方向フラグをリセットする
               ExtDirectionFlag=0;
               //--- 新しい最終日を格納する
```

© 2000-2025, MetaQuotes Ltd.

```
                ExtLastTime=ExtCurrentTime[0];
               }
            else
               {
                //--- ローソク足を作成する
                FillCandleMain(open,close,high,low,ExtStartPos,i,i,ExtIndexMax,ExtInd
exMin);
               }
            }
         }
      }
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| 指定された指標期間の正確さをチェックする                                |
//+------------------------------------------------------------------+
bool CheckPeriod(int current_period,int high_period)
  {
//--- 指標期間は、それを表示する時間軸を超えている必要がある
   if(current_period>=high_period)
     {
      Print("Error! The value of the indicator period should exceed the value of the
current time frame!");
      return(false);
     }
//--- 指標期間が1週間または一ヶ月の場合は、期間は正しい
   if(high_period>32768)
      return(true);
//--- 期間値を分に変換する
   if(high_period>30)
      high_period=(high_period-16384)*60;
   if(current_period>30)
      current_period=(current_period-16384)*60;
//--- 指標期間は、それを表示する時間軸の倍数である必要がある
   if(high_period%current_period!=0)
     {
      Print("Error! The value of the indicator period should be multiple of the
value of the current time frame!");
      return(false);
     }
//--- 指標期間は、それを表示する時間軸を 3 回以上越える必要がある
   if(high_period/current_period<3)
     {
      Print("Error! The indicator period should exceed the current time frame 3 or
more times!");
      return(false);
     }
//--- 現在の時間軸の指標期間は正しい
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//|  大き目な時間軸のデータを受け取る                                |
//+------------------------------------------------------------------+
bool GetTimeData(void)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- 現在時間のデータを全て複製する
   if(CopyTime(Symbol(),InpPeriod,InpDateStart,TimeCurrent(),ExtTimeBuff)==-1)
     {
      //--- エラーコードを受け取る
      int code=GetLastError();
      //--- エラーメッセージをプリントアウトする
      PrintFormat("Data copy error! %s",code==4401
                  ? "History is still being uploaded!"
                  : "Code = "+IntegerToString(code));
      //--- データダウンロードを繰り返して試みるためにfalseを返す
      return(false);
     }
//--- 配列のサイズを受け取る
   ExtSize=ArraySize(ExtTimeBuff);
//--- 配列ループインデックスをゼロにする
   ExtCount=0;
//--- 時間軸の現在のローソク足の位置をゼロにする
   ExtStartPos=0;
   ExtStartFlag=true;
//--- 大き目の時間軸からの前回値を格納する
   ExtLastTime=ExtTimeBuff[ExtSize-1];
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  この関数は、ローソク足の主要部分を構成する。フラグの値に応じて、|
//データと配列を定義して                                             |
//|  正確な描画に使用する                                            |
//+------------------------------------------------------------------+
void FormCandleMain(double &body_fst[],double &body_snd[],
                    double &shadow_fst[],double &shadow_snd[],
                    const double fst_value,const double snd_value,
                    const double fst_extremum,const double snd_extremum,
                    const int start,const int count,const bool flag)
  {
//--- フラグの値をチェックする
   if(flag)
     {
      //--- ローソク足の実体を作成する
      FormMain(body_fst,body_snd,fst_value,snd_value,start,count);
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- ローソク足の影を作成する
      FormMain(shadow_fst,shadow_snd,fst_extremum,snd_extremum,start,count);
     }
   else
     {
      //--- ローソク足の実体を作成する
      FormMain(body_fst,body_snd,snd_value,fst_value,start,count);
      //--- ローソク足の影を作成する
      FormMain(shadow_fst,shadow_snd,snd_extremum,fst_extremum,start,count);
     }
  }
//+------------------------------------------------------------------------------+
//| この関数は、ローソク足の終わりの部分を構成するフラグの値に応じて                |
//| この関数はデータと配列を定義して                                              |
//| 正確な描画に使用する                                                          |
//+------------------------------------------------------------------------------+
void FormCandleEnd(double &body_fst[],double &body_snd[],
                   double &shadow_fst[],double &shadow_snd[],
                   const double fst_value,const double snd_value,
                   const double fst_extremum,const double snd_extremum,
                   const int end,bool &flag)
  {
//--- フラグの値をチェックする
   if(flag)
     {
      //--- ローソク足の実体の終わりを作成する
      FormEnd(body_fst,body_snd,fst_value,snd_value,end);
      //--- ローソク足の影の終わりを作成する
      FormEnd(shadow_fst,shadow_snd,fst_extremum,snd_extremum,end);
      //--- フラグの値を反対に変更する
      flag=false;
     }
   else
     {
      //--- ローソク足の実体の終わりを作成する
      FormEnd(body_fst,body_snd,snd_value,fst_value,end);
      //--- ローソク足の影の終わりを作成する
      FormEnd(shadow_fst,shadow_snd,snd_extremum,fst_extremum,end);
      //--- フラグの値を反対に変更する
      flag=true;
     }
  }
//+------------------------------------------------------------------------------
+
//| ローソク足の終わりの部分をクリアする
|
//| (最後と最後から2 番目のバーの間の領域)
|
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------------------
+
void ClearEndOfBodyMain(const int ind)
  {
   ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSe
cond,ind,1);
   ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowSe
cond,ind,1);
  }
//+------------------------------------------------------------------------------+
//|  ローソク足をクリアする                                                      |
//+------------------------------------------------------------------------------+
void ClearCandle(double &body_fst[],double &body_snd[],double &shadow_fst[],
                 double &shadow_snd[],const int start,const int count)
  {
//--- チェック
   if(count!=0)
     {
      //--- 指標バッファを空の値で埋める
      ArrayFill(body_fst,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(body_snd,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(shadow_fst,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(shadow_snd,start,count,INDICATOR_EMPTY_VALUE);
     }
  }
//+------------------------------------------------------------------------------+
//|  ローソク足の実体を作成する                                                  |
//+------------------------------------------------------------------------------+
void FormMain(double &fst[],double &snd[],const double fst_value,
              const double snd_value,const int start,const int count)
  {
//--- チェック
   if(count!=0)
     {
      //--- 指標バッファに値を記入する
      ArrayFill(fst,start,count,fst_value);
      ArrayFill(snd,start,count,snd_value);
     }
  }
//+------------------------------------------------------------------------------+
//|  ローソク足の終わりを作成する                                                |
//+------------------------------------------------------------------------------+
void FormEnd(double &fst[],double &snd[],const double fst_value,
             const double snd_value,const int last)
  {
//--- 指標バッファに値を記入する
   ArrayFill(fst,last-1,2,fst_value);
   ArrayFill(snd,last-1,2,snd_value);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  指標バッファの i  要素を空の値で埋める                          |
//+------------------------------------------------------------------+
void FillIndicatorBuffers(const int i)
  {
//---  指標バッファのセルを空の値で埋める
   ExtBearBodyFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodySecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodySecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
  }
```

© 2000–2025, MetaQuotes Ltd.

ArrayMinimum

多次元数値配列の初次元で最小の要素を検索します。

```
int  ArrayMinimum(
   const void&    array[],              // 検索される配列
   int            start=0,              // チェック開始インデックス
   int            count=WHOLE_ARRAY     // チェックされた要素数
   );
```

パラメータ

array[]

[in] 検索対象の数値配列

start=0

[in] チェック開始インデックス

count=WHOLE_ARRAY

[in] 検索される要素数。デフォルトでは配列全体が検索されます（count=WHOLE_ARRAY）。

戻り値

配列の索引付け方向を考慮して見つかった要素のインデックスを返します。失敗した場合は、-1 が返されます。

注意事項

最小値を検索している際にAS_SERIES フラグ値が考慮されます。

ArrayMaximum と ArrayMinimum はパラメータとして任意の次元の配列を受け入れます。しかし、検索は常に最初（ゼロ）の次元に適用されます。

例：

```
#property description "The indicator displays larger timeframe's candlesticks on the current one."
//--- 指標の設定
#property indicator_chart_window
#property indicator_buffers 16
#property indicator_plots   8
//---- プロット1
#property indicator_label1  "BearBody"
#property indicator_color1  clrSeaGreen,clrSeaGreen
//---- プロット 2
#property indicator_label2  "BearBodyEnd"
#property indicator_color2  clrSeaGreen,clrSeaGreen
//---- プロット 3
#property indicator_label3  "BearShadow"
#property indicator_color3  clrSalmon,clrSalmon
//---- プロット 4
#property indicator_label4  "BearShadowEnd"
#property indicator_color4  clrSalmon,clrSalmon
```

© 2000-2025, MetaQuotes Ltd.

```
//---- プロット 5
#property indicator_label5  "BullBody"
#property indicator_color5  clrOlive,clrOlive
//---- プロット 6
#property indicator_label6  "BullBodyEnd"
#property indicator_color6  clrOlive,clrOlive
//---- プロット7
#property indicator_label7  "BullShadow"
#property indicator_color7  clrSkyBlue,clrSkyBlue
//---- プロット8
#property indicator_label8  "BullShadowEnd"
#property indicator_color8  clrSkyBlue,clrSkyBlue
//--- 定義済み定数
#define INDICATOR_EMPTY_VALUE 0.0
//--- 入力パラメータ
input ENUM_TIMEFRAMES InpPeriod=PERIOD_H4;              // 指標計算のための時間軸
input datetime        InpDateStart=D'2013.01.01 00:00'; // 分析開始日
//--- 弱気ローソク足の指標バッファ
double   ExtBearBodyFirst[];
double   ExtBearBodySecond[];
double   ExtBearBodyEndFirst[];
double   ExtBearBodyEndSecond[];
double   ExtBearShadowFirst[];
double   ExtBearShadowSecond[];
double   ExtBearShadowEndFirst[];
double   ExtBearShadowEndSecond[];
//--- 強気ローソク足の指標バッファ
double   ExtBullBodyFirst[];
double   ExtBullBodySecond[];
double   ExtBullBodyEndFirst[];
double   ExtBullBodyEndSecond[];
double   ExtBullShadowFirst[];
double   ExtBullShadowSecond[];
double   ExtBullShadowEndFirst[];
double   ExtBullShadowEndSecond[];
//--- グローバル変数
datetime ExtTimeBuff[];      // 大きめの時間軸の時間バッファ
int      ExtSize=0;          // 時間バッファサイズ
int      ExtCount=0;         // 時間バッファ内のインデックス
int      ExtStartPos=0;      // 指標計算の初期位置
bool     ExtStartFlag=true;  // 初期位置を受信するための補助フラグ
datetime ExtCurrentTime[1];  // 大きめの時間軸のバーの生成の最終時間
datetime ExtLastTime;        // 計算の行われた大きめの時間軸の最終時間
bool     ExtBearFlag=true;   // 弱気指標バッファにデータを書き込む順序を定義するためのフラグ
bool     ExtBullFlag=true;   // 強気指標バッファにデータを書き込む順序を定義するためのフラグ
int      ExtIndexMax=0;      // 配列内の最大要素のインデックス
int      ExtIndexMin=0;      // 配列内の最小要素のインデックス
int      ExtDirectionFlag=0; // 現在のローソク足の価格移動方向
//--- 正しく描画するためにローソク足の始値と終値間をシフトする
```

© 2000-2025, MetaQuotes Ltd.

```
const double ExtEmptyBodySize=0.2*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//+------------------------------------------------------------------+
//|  ローソク足の基本的な部分を埋める                                |
//+------------------------------------------------------------------+
void FillCandleMain(const double &open[],const double &close[],
                    const double &high[],const double &low[],
                    const int start,const int last,const int fill_index,
                    int &index_max,int &index_min)
  {
//--- 配列内の最大と最小要素のインデックスを見つける
   index_max=ArrayMaximum(high,ExtStartPos,last-start+1); // 最高
   index_min=ArrayMinimum(low,ExtStartPos,last-start+1);  // 最低
//--- 現在の時間軸からいくつのバーを埋めるかを定義する
   int count=fill_index-start+1;
//--- 最初のバーの終値が最後のバーでの終値を超えた場合、ローソク足は弱気となる
   if(open[start]>close[last])
     {
      //--- ローソク足が以前に強気だった場合は強気指標バッファの値をクリアする
      if(ExtDirectionFlag!=-1)
         ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullSh
adowSecond,start,count);
      //--- 弱気ローソク足
      ExtDirectionFlag=-1;
      //--- ローソク足を作成する
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearSh
adowSecond,open[start],
                     close[last],high[index_max],low[index_min],start,count,ExtBearF
lag);
      //--- 関数を終了する
      return;
     }
//--- 最初のバーの終値が最後のバーでの終値より低い場合、ローソク足は強気となる
   if(open[start]<close[last])
     {
      //--- ローソク足が以前に強気だった場合は強気指標バッファの値をクリアする
      if(ExtDirectionFlag!=1)
         ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearSh
adowSecond,start,count);
      //--- 強気ローソク足
      ExtDirectionFlag=1;
      //--- ローソク足を作成する
      FormCandleMain(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullSh
adowSecond,close[last],
                     open[start],high[index_max],low[index_min],start,count,ExtBullF
lag);
      //--- 関数を終了する
      return;
     }
//--- 関数のこの部分が実行される場合、最初のバーでの始値は最後のバーで終値に等しい。
//--- このようなローソク足は弱気であると考えられる
```

© 2000-2025, MetaQuotes Ltd.

```
//--- ローソク足が以前に強気だった場合は強気指標バッファの値をクリアする
   if(ExtDirectionFlag!=-1)
      ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowSecond,start,count);
//--- 弱気ローソク足
   ExtDirectionFlag=-1;
//--- 終値と始値が等しい場合、シフトをつくって正しく描画する
   if(high[index_max]!=low[index_min])
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSecond,open[start],
                     open[start]-ExtEmptyBodySize,high[index_max],low[index_min],start,count,ExtBearFlag);
   else
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSecond,
                     open[start],open[start]-ExtEmptyBodySize,high[index_max],
                     high[index_max]-ExtEmptyBodySize,start,count,ExtBearFlag);
  }
//+------------------------------------------------------------------+
//|  ローソク足の終わりを埋める                                      |
//+------------------------------------------------------------------+
void FillCandleEnd(const double &open[],const double &close[],
                   const double &high[],const double &low[],
                   const int start,const int last,const int fill_index,
                   const int index_max,const int index_min)
  {
//--- シングルバーの場合は描画しない
   if(last-start==0)
      return;
//--- 最初のバーの終値が最後のバーでの終値を超えた場合、ローソク足は弱気となる
   if(open[start]>close[last])
     {
      //--- ローソク足の終わりを作成する
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,ExtBearShadowEndSecond,
                    open[start],close[last],high[index_max],low[index_min],fill_index,ExtBearFlag);
      //--- 関数を終了する
      return;
     }
//--- 最初のバーの終値が最後のバーでの終値より低い場合、ローソク足は強気となる
   if(open[start]<close[last])
     {
      //--- ローソク足の終わりを作成する
      FormCandleEnd(ExtBullBodyEndFirst,ExtBullBodyEndSecond,ExtBullShadowEndFirst,ExtBullShadowEndSecond,
                    close[last],open[start],high[index_max],low[index_min],fill_index,ExtBullFlag);
      //--- 関数を終了する
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- 関数のこの部分が実行される場合、最初のバーでの始値は最後のバーで終値に等しい。
//--- このようなローソク足は弱気であると考えられる
//--- ローソク足の終わりを作成する
   if(high[index_max]!=low[index_min])
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,E
xtBearShadowEndSecond,open[start],
                    open[start]-
ExtEmptyBodySize,high[index_max],low[index_min],fill_index,ExtBearFlag);
   else
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,E
xtBearShadowEndSecond,open[start],
                    open[start]-ExtEmptyBodySize,high[index_max],high[index_max]-
ExtEmptyBodySize,fill_index,ExtBearFlag);
  }
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 指標期間をチェックする
   if(!CheckPeriod((int)Period(),(int)InpPeriod))
      return(INIT_PARAMETERS_INCORRECT);
//--- 価格データ前景に表示する
   ChartSetInteger(0,CHART_FOREGROUND,0,1);
//--- 指標バッファを結合する
   SetIndexBuffer(0,ExtBearBodyFirst);
   SetIndexBuffer(1,ExtBearBodySecond);
   SetIndexBuffer(2,ExtBearBodyEndFirst);
   SetIndexBuffer(3,ExtBearBodyEndSecond);
   SetIndexBuffer(4,ExtBearShadowFirst);
   SetIndexBuffer(5,ExtBearShadowSecond);
   SetIndexBuffer(6,ExtBearShadowEndFirst);
   SetIndexBuffer(7,ExtBearShadowEndSecond);
   SetIndexBuffer(8,ExtBullBodyFirst);
   SetIndexBuffer(9,ExtBullBodySecond);
   SetIndexBuffer(10,ExtBullBodyEndFirst);
   SetIndexBuffer(11,ExtBullBodyEndSecond);
   SetIndexBuffer(12,ExtBullShadowFirst);
   SetIndexBuffer(13,ExtBullShadowSecond);
   SetIndexBuffer(14,ExtBullShadowEndFirst);
   SetIndexBuffer(15,ExtBullShadowEndSecond);
//--- 指標作成のためにいくつかのプロパティ値を設定する
   for(int i=0;i<8;i++)
     {
      PlotIndexSetInteger(i,PLOT_DRAW_TYPE,DRAW_FILLING); // グラフィカルな構成タイプ
      PlotIndexSetInteger(i,PLOT_LINE_STYLE,STYLE_SOLID); // 描画線のスタイル
      PlotIndexSetInteger(i,PLOT_LINE_WIDTH,1);           // 描画線の幅
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- 計算されたバーがまだない場合
   if(prev_calculated==0)
     {
      //--- 大きめの時間軸のバーの到着時間を受け取る
      if(!GetTimeData())
         return(0);
     }
//--- 直接のインデックス作成を設定する
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(close,false);
//--- バーを計算する start 変数
   int start=prev_calculated;
//--- バーが作成されたらその指標値を再計算する
   if(start!=0 && start==rates_total)
      start--;
//--- 指標値を計算するためのループ
   for(int i=start;i<rates_total;i++)
     {
      //--- 指標バッファの i 要素を空の値で埋める
      FillIndicatorBuffers(i);
      //--- InpDateStart 日からバーの計算を行う
      if(time[i]>=InpDateStart)
        {
         //--- 値が表示される元となる位置を初めて定義する
         if(ExtStartFlag)
           {
            //--- 初期のバーの値を収納する
            ExtStartPos=i;
            //--- 大き目な時間軸でtime[i] を超える最初の日を定義する
```

© 2000-2025, MetaQuotes Ltd.

```
            while(time[i]>=ExtTimeBuff[ExtCount])
               if(ExtCount<ExtSize-1)
                  ExtCount++;
            //--- 再びこのブロックに遭遇しないようにフラグの値を変更する
            ExtStartFlag=false;
           }
         //--- 配列にまだ要素が残っているかをチェックする
         if(ExtCount<ExtSize)
           {
            //--- 現在の時間軸の値が大きなな時間軸の値に届くまで待つ
            if(time[i]>=ExtTimeBuff[ExtCount])
              {
               //--- ( 最後と最後から2 番目のバーの間の領域を記入せずに) ローソク足の主な部分を描画する
               FillCandleMain(open,close,high,low,ExtStartPos,i-1,i-
2,ExtIndexMax,ExtIndexMin);
               //--- ローソク足の終わりの部分を埋める ( 最後と最後から2 番目のバーの間の領域)
               FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-
1,ExtIndexMax,ExtIndexMin);
               //--- 次のローソク足を描画するために初期位置を移動する
               ExtStartPos=i;
               //--- 配列カウンタを増加する
               ExtCount++;
              }
            else
               continue;
           }
         else
           {
            //--- 配列の値をリセットする
            ResetLastError();
            //--- 大き目な時間軸の最終日を受け取る
            if(CopyTime(Symbol(),InpPeriod,0,1,ExtCurrentTime)==-1)
              {
               Print("Data copy error, code = ",GetLastError());
               return(0);
              }
            //---新しいデータが後に来る場合には、ローソク足の作成を中止する
            if(ExtCurrentTime[0]>ExtLastTime)
              {
               //--- 主要指標バッファの最後と最後から2 番目のバーの間の領域をクリアする
               ClearEndOfBodyMain(i-1);
               //--- 補助指標バッファを使用して領域に記入する
               FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-
1,ExtIndexMax,ExtIndexMin);
               //--- 次のローソク足を描画するために初期位置を移動する
               ExtStartPos=i;
               //--- 価格方向フラグをリセットする
               ExtDirectionFlag=0;
               //--- 新しい最終日を格納する
```

© 2000-2025, MetaQuotes Ltd.

```
                ExtLastTime=ExtCurrentTime[0];
               }
             else
               {
                //--- ローソク足を作成する
                FillCandleMain(open,close,high,low,ExtStartPos,i,i,ExtIndexMax,ExtInd
exMin);
               }
            }
         }
      }
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//|  指定された指標期間の正確さをチェックする                        |
//+------------------------------------------------------------------+
bool CheckPeriod(int current_period,int high_period)
  {
//--- 指標期間は、それを表示する時間軸を超えている必要がある
   if(current_period>=high_period)
     {
      Print("Error! The value of the indicator period should exceed the value of the
current time frame!");
      return(false);
     }
//--- 指標期間が1週間または一ヶ月の場合は、期間は正しい
   if(high_period>32768)
      return(true);
//--- 期間値を分に変換する
   if(high_period>30)
      high_period=(high_period-16384)*60;
   if(current_period>30)
      current_period=(current_period-16384)*60;
//--- 指標期間は、それを表示する時間軸の倍数である必要がある
   if(high_period%current_period!=0)
     {
      Print("Error! The value of the indicator period should be multiple of the
value of the current time frame!");
      return(false);
     }
//--- 指標期間は、それを表示する時間軸を 3 回以上越える必要がある
   if(high_period/current_period<3)
     {
      Print("Error! The indicator period should exceed the current time frame 3 or
more times!");
      return(false);
     }
//--- 現在の時間軸の指標期間は正しい
```

© 2000-2025, MetaQuotes Ltd.

```
    return(true);
   }
//+------------------------------------------------------------------+
//|  大き目な時間軸のデータを受け取る                                |
//+------------------------------------------------------------------+
bool GetTimeData(void)
   {
//--- エラー値をリセットする
    ResetLastError();
//--- 現在時間のデータを全て複製する
    if(CopyTime(Symbol(),InpPeriod,InpDateStart,TimeCurrent(),ExtTimeBuff)==-1)
      {
       //--- エラーコードを受け取る
       int code=GetLastError();
       //--- エラーメッセージをプリントアウトする
       PrintFormat("Data copy error! %s",code==4401
                   ? "History is still being uploaded!"
                   : "Code = "+IntegerToString(code));
       //--- データダウンロードを繰り返して試みるためにfalseを返す
       return(false);
      }
//--- 配列のサイズを受け取る
    ExtSize=ArraySize(ExtTimeBuff);
//--- 配列ループインデックスをゼロにする
    ExtCount=0;
//--- 時間軸の現在のローソク足の位置をゼロにする
    ExtStartPos=0;
    ExtStartFlag=true;
//--- 大き目の時間軸からの前回値を格納する
    ExtLastTime=ExtTimeBuff[ExtSize-1];
//--- 実行成功
    return(true);
   }
//+------------------------------------------------------------------------+
//|  この関数は、ローソク足の主要部分を構成する。フラグの値に応じて、|
//データと配列を定義して                                                   |
//|  正確な描画に使用する                                                  |
//+------------------------------------------------------------------------+
void FormCandleMain(double &body_fst[],double &body_snd[],
                    double &shadow_fst[],double &shadow_snd[],
                    const double fst_value,const double snd_value,
                    const double fst_extremum,const double snd_extremum,
                    const int start,const int count,const bool flag)
   {
//--- フラグの値をチェックする
    if(flag)
      {
       //--- ローソク足の実体を作成する
       FormMain(body_fst,body_snd,fst_value,snd_value,start,count);
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- ローソク足の影を作成する
      FormMain(shadow_fst,shadow_snd,fst_extremum,snd_extremum,start,count);
     }
   else
     {
      //--- ローソク足の実体を作成する
      FormMain(body_fst,body_snd,snd_value,fst_value,start,count);
      //--- ローソク足の影を作成する
      FormMain(shadow_fst,shadow_snd,snd_extremum,fst_extremum,start,count);
     }
  }
//+------------------------------------------------------------------------------+
//|  この関数は、ローソク足の終わりの部分を構成するフラグの値に応じて  |
//|  この関数はデータと配列を定義して                                  |
//|  正確な描画に使用する                                              |
//+------------------------------------------------------------------------------+
void FormCandleEnd(double &body_fst[],double &body_snd[],
                   double &shadow_fst[],double &shadow_snd[],
                   const double fst_value,const double snd_value,
                   const double fst_extremum,const double snd_extremum,
                   const int end,bool &flag)
  {
//--- フラグの値をチェックする
   if(flag)
     {
      //--- ローソク足の実体の終わりを作成する
      FormEnd(body_fst,body_snd,fst_value,snd_value,end);
      //--- ローソク足の影の終わりを作成する
      FormEnd(shadow_fst,shadow_snd,fst_extremum,snd_extremum,end);
      //--- フラグの値を反対に変更する
      flag=false;
     }
   else
     {
      //--- ローソク足の実体の終わりを作成する
      FormEnd(body_fst,body_snd,snd_value,fst_value,end);
      //--- ローソク足の影の終わりを作成する
      FormEnd(shadow_fst,shadow_snd,snd_extremum,fst_extremum,end);
      //--- フラグの値を反対に変更する
      flag=true;
     }
  }
//
+------------------------------------------------------------------------------
--+
//|  ローソク足の終わりの部分をクリアする      |
//|  ( 最後と最後から2  番目のバーの間の領域)
                                  |
```

© 2000-2025, MetaQuotes Ltd.

```
//
+------------------------------------------------------------------------------
--+
void ClearEndOfBodyMain(const int ind)
  {
   ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSe
cond,ind,1);
   ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowSe
cond,ind,1);
  }
//+------------------------------------------------------------------+
//| ローソク足をクリアする                                           |
//+------------------------------------------------------------------+
void ClearCandle(double &body_fst[],double &body_snd[],double &shadow_fst[],
                 double &shadow_snd[],const int start,const int count)
  {
//--- チェック
   if(count!=0)
     {
      //--- 指標バッファを空の値で埋める
      ArrayFill(body_fst,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(body_snd,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(shadow_fst,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(shadow_snd,start,count,INDICATOR_EMPTY_VALUE);
     }
  }
//+------------------------------------------------------------------+
//| ローソク足の実体を作成する                                       |
//+------------------------------------------------------------------+
void FormMain(double &fst[],double &snd[],const double fst_value,
              const double snd_value,const int start,const int count)
  {
//--- チェック
   if(count!=0)
     {
      //--- 指標バッファに値を記入する
      ArrayFill(fst,start,count,fst_value);
      ArrayFill(snd,start,count,snd_value);
     }
  }
//+------------------------------------------------------------------+
//| ローソク足の終わりを作成する                                     |
//+------------------------------------------------------------------+
void FormEnd(double &fst[],double &snd[],const double fst_value,
             const double snd_value,const int last)
  {
//--- 指標バッファに値を記入する
   ArrayFill(fst,last-1,2,fst_value);
   ArrayFill(snd,last-1,2,snd_value);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//|  指標バッファの i  要素を空の値で埋める                          |
//+------------------------------------------------------------------+
void FillIndicatorBuffers(const int i)
  {
//--- 指標バッファのセルを空の値で埋める
   ExtBearBodyFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodySecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodySecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrayPrint

単純型または単純な構造体の配列を操作ログに出力します。

```
void  ArrayPrint(
   const void&   array[],             // 出力された配列
   uint          digits=_Digits,      // 小数点以下の桁数
   const string  separator=NULL,      // 構造体フィールド値のセパレーター
   ulong         start=0,             // 最初に出力された要素のインデックス
   ulong         count=WHOLE_ARRAY,   // 出力された要素の数
   ulong         flags=ARRAYPRINT_HEADER|ARRAYPRINT_INDEX|ARRAYPRINT_LIMIT|
ARRAYPRINT_ALIGN
   );
```

パラメータ

array[]

[in] 単純型または単純な構造体の配列。

digits=_Digits

[in] 実数型の小数点以下の桁数。デフォルト値は_Digits.。

separator=NULL

[in] 構造体フィールド値のセパレーター。デフォルト値のNULLは空のラインを意味します。この場合スペースが区切り文字として使用されます。

start=0

[in] 最初に出力された配列要素のインデックス。デフォルトではゼロインデックスから出力されます。

count=WHOLE_ARRAY

[in] 出力される配列要素の数。デフォルトでは配列全体が表示されます（count=WHOLE_ARRAY）。

flags=ARRAYPRINT_HEADER|ARRAYPRINT_INDEX|ARRAYPRINT_LIMIT|ARRAYPRINT_ALIGN

[in] 出力モードを設定するフラグの組み合わせ。デフォルトではすべてのフラグが有効です。

- ☐ARRAYPRINT_HEADER – 構造体配列のヘッダを出力します。
- ☐ARRAYPRINT_INDEX – インデックスを左側に出力します。
- ☐ARRAYPRINT_LIMIT – 配列の最初の100および最後の100要素のみを出力します。大きな配列の一部だけを出力したい場合に使用します。
- ☐ARRAYPRINT_ALIGN – 出力された値の位置合わせを可能にします – 数は右、ラインは左整列されます。
- ☐ARRAYPRINT_DATE – datetimeを出力するときはdd.mm.yyyy形式を使います。
- ☐ARRAYPRINT_MINUTES – datetimeを出力するときはHH:MM形式を使います。
- ☐ARRAYPRINT_SECONDS – datetimeを出力するときはHH:MM:SS形式を使います。

戻り値

いいえ

注意事項

ArrayPrint() は構造体配列のフィールドを操作ログに出力するものではありません。配列とオブジェクトポインタフィールドは抜かされます。これらの列は、単により便利なプレゼンテーションのために出力されませ

© 2000–2025, MetaQuotes Ltd.

ん。すべての構造体のフィールドを出力する必要がある場合は、希望の書式を使用して独自のマス出力関数を記述する必要があります。

例：

```
//--- 最後の10バーの値を出力する
   MqlRates rates[];
   if(CopyRates(_Symbol,_Period,1,10,rates))
     {
      ArrayPrint(rates);
      Print("Check\n[time]\t[open]\t[high]\t[low]\t[close]\t[tick_volume]\t[spread]
\t[real_volume]");
      for(int i=0;i<10;i++)
        {
         PrintFormat("[%d]\t%s\t%G\t%G\t%G\t%G\t%G\t%G\t%I64d\t",i,
         TimeToString(rates[i].time,TIME_DATE|TIME_MINUTES|TIME_SECONDS),
         rates[i].open,rates[i].high,rates[i].low,rates[i].close,
         rates[i].tick_volume,rates[i].spread,rates[i].real_volume);
        }
     }
   else
      PrintFormat("CopyRates failed, error code=%d",GetLastError());
//--- 出力例
/*
                    [time]  [open]  [high]   [low] [close] [tick_volume] [spread]
[real_volume]
   [0] 2016.11.09 04:00:00 1.11242 1.12314 1.11187 1.12295         18110       10
17300175000
   [1] 2016.11.09 05:00:00 1.12296 1.12825 1.11930 1.12747         17829        9
15632176000
   [2] 2016.11.09 06:00:00 1.12747 1.12991 1.12586 1.12744         13458       10
 9593492000
   [3] 2016.11.09 07:00:00 1.12743 1.12763 1.11988 1.12194         15362        9
12352245000
   [4] 2016.11.09 08:00:00 1.12194 1.12262 1.11058 1.11172         16833        9
12961333000
   [5] 2016.11.09 09:00:00 1.11173 1.11348 1.10803 1.11052         15933        8
10720384000
   [6] 2016.11.09 10:00:00 1.11052 1.11065 1.10289 1.10528         11888        9
 8084811000
   [7] 2016.11.09 11:00:00 1.10512 1.11041 1.10472 1.10915          7284       10
 5087113000
   [8] 2016.11.09 12:00:00 1.10915 1.11079 1.10892 1.10904          8710        9
 6769629000
   [9] 2016.11.09 13:00:00 1.10904 1.10913 1.10223 1.10263          8956        7
 7192138000
  確認
   [time] [open] [high] [low] [close] [tick_volume] [spread] [real_volume]
   [0] 2016.11.09 04:00:00 1.11242 1.12314 1.11187 1.12295 18110 10 17300175000
   [1] 2016.11.09 05:00:00 1.12296 1.12825 1.1193 1.12747 17829 9 15632176000
   [2] 2016.11.09 06:00:00 1.12747 1.12991 1.12586 1.12744 13458 10 9593492000
   [3] 2016.11.09 07:00:00 1.12743 1.12763 1.11988 1.12194 15362 9 12352245000
```

© 2000-2025, MetaQuotes Ltd.

```
   [4] 2016.11.09 08:00:00 1.12194 1.12262 1.11058 1.11172 16833 9 12961333000
   [5] 2016.11.09 09:00:00 1.11173 1.11348 1.10803 1.11052 15933 8 10720384000
   [6] 2016.11.09 10:00:00 1.11052 1.11065 1.10289 1.10528 11888 9 8084811000
   [7] 2016.11.09 11:00:00 1.10512 1.11041 1.10472 1.10915 7284 10 5087113000
   [8] 2016.11.09 12:00:00 1.10915 1.11079 1.10892 1.10904 8710 9 6769629000
   [9] 2016.11.09 13:00:00 1.10904 1.10913 1.10223 1.10263 8956 7 7192138000
*/
```

参照

FileSave、FileLoad

© 2000–2025, MetaQuotes Ltd.

ArrayRange

この関数は配列の指定された次元の要素数を返します。

```
int  ArrayRange(
   const void&    array[],      // チェックされる配列
   int            rank_index    // 次元のインデックス
   );
```

パラメータ

array[]

[in] チェックされた配列

rank_index

[in] 次元のインデックス

戻り値

配列の指定された次元の要素数

注意事項

インデックスは 0 から始まるので、配列の次元の数は最後の次元のインデックスより 1 大きいです。

例:

```
void OnStart()
  {
//--- 4 次元配列を作成する
   double array[][5][2][4];
//--- 0 次元のサイズを設定する
   ArrayResize(array,10,10);
//---次元をプリントする
   int temp;
   for(int i=0;i<4;i++)
     {
      //--- i 次元のサイズを受け取る
      temp=ArrayRange(array,i);
      //--- プリント
      PrintFormat("dim = %d, range = %d",i,temp);
     }
//--- 結果
// dim = 0, range = 10
// dim = 1, range = 5
// dim = 2, range = 2
// dim = 3, range = 4
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrayResize

この関数は最初の次元のサイズを新しく設定します。

```
int  ArrayResize(
   void&  array[],               // 参照によって渡された配列
   int    new_size,              // 新しい配列サイズ
   int    reserve_size=0         // 予備サイズの値(超過)
   );
```

パラメータ

array[]

[out] サイズを変更する配列

new_size

[in] 最初の次元の新しいサイズ

reserve_size=0

[in] 予備を取得するための分散サイズ

戻り値

正常に実行された場合はサイズ変更後の配列に含まれる全ての要素の数を返します。それ以外の場合は、-1を返し、配列のサイズは変更されません。

ArrayResize()が静的配列、時系列または指標バッファに適応された場合、配列サイズは同じままです。これらの配列は再割り当てされません。この場合*new_size*<=ArraySize(*array*)であれば、この関数は*new_size*のみを返します。その他の場合は -1が返されます。

注意事項

この関数は 動的配列のみに使用出来ます。SetIndexBuffer() 関数によって指標バッファとして割り当てられた動的配列のサイズは変更出来ないので注意が必要です。指標バッファについては、サイズ変更の全ての操作は端末のランタイムサブシステムによって実行されます。

配列の要素数の合計は 2,147,483,647を超えることは出来ません。

頻繁なメモリ割り当てをする場合には、物理メモリの割り当て量を減らすために予備を設定する 3 番目のパラメータを使用することをお勧めします。ArrayResize の後続の呼び出しはメモリの物理的な再配分にはつながらず、確保したメモリ内の最初の配列の次元のサイズを変更するだけです。3 番目のパラメータは物理メモリの割り当て時にのみ使用されることを忘れてはいけません。例：

```
ArrayResize(arr,1000,1000);
for(int i=1;i<3000;i++)
   ArrayResize(arr,i,1000);
```

この場合、メモリは2 回再割り当てされます。1 回目は3000要素のループに入る前（配列のサイズを1000に設定されます）で2 回目は i が 2000の時です。3 番目のパラメータを省略した場合は、メモリの物理的な再配分が2000回起こりプログラムの実行がを遅くなります。

例：

```
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|  スクリプトプログラムを開始する関数                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//---カウンタ
   ulong start=GetTickCount();
   ulong now;
   int   count=0;
//--- 迅速なバージョンのデモ配列
   double arr[];
   ArrayResize(arr,100000,100000);
//--- 予備メモリを使用したバージョンがどんなに迅速かをチェックする
   Print("--- Test Fast: ArrayResize(arr,100000,100000)");
   for(int i=1;i<=300000;i++)
     {
      //--- 100,000 予備要素を指定して新しい配列サイズを設定する。
      ArrayResize(arr,i,100000);
      //--- ラウンド数に達した時、配列のサイズとかかった時間を表示する
      if(ArraySize(arr)%100000==0)
        {
         now=GetTickCount();
         count++;
         PrintFormat("%d. ArraySize(arr)=%d Time=%d ms",count,ArraySize(arr),(now-
start));
         start=now;
        }
     }
//--- 予備メモリを使用したバージョンがどんなに低速かを見る
   double slow[];
   ArrayResize(slow,100000,100000);
//---
   count=0;
   start=GetTickCount();
   Print("---- Test Slow: ArrayResize(slow,100000)");
//---
   for(int i=1;i<=300000;i++)
     {
      //--- 新しい配列サイズを設定するが予備を使用しない
      ArrayResize(slow,i);
      //--- ラウンド数に達した時、配列のサイズとかかった時間を表示する
      if(ArraySize(slow)%100000==0)
        {
         now=GetTickCount();
         count++;
         PrintFormat("%d. ArraySize(slow)=%d Time=%d ms",count,ArraySize(slow),(now-
start));
         start=now;
        }
     }
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//--- スクリプトのサンプル結果
/*
   Test_ArrayResize (EURUSD,H1)   --- Test Fast: ArrayResize(arr,100000,100000)
   Test_ArrayResize (EURUSD,H1)   1. ArraySize(arr)=100000 Time=0 ms
   Test_ArrayResize (EURUSD,H1)   2. ArraySize(arr)=200000 Time=0 ms
   Test_ArrayResize (EURUSD,H1)   3. ArraySize(arr)=300000 Time=0 ms
   Test_ArrayResize (EURUSD,H1)   ---- Test Slow: ArrayResize(slow,100000)
   Test_ArrayResize (EURUSD,H1)   1. ArraySize(slow)=100000 Time=0 ms
   Test_ArrayResize (EURUSD,H1)   2. ArraySize(slow)=200000 Time=0 ms
   Test_ArrayResize (EURUSD,H1)   3. ArraySize(slow)=300000 Time=228511 ms
*/
```

参照

ArrayInitialize

© 2000–2025, MetaQuotes Ltd.

ArrayInsert

指定されたインデックスから開始して、指定された数の要素をソース配列からターゲット配列に挿入します。

```
bool  ArrayInsert(
   void&         dst_array[],          // 挿入先の配列
   const void&   src_array[],          // 挿入元の配列
   uint          dst_start,            // 挿入先の配列に挿入されるインデックス
   uint          src_start=0,          // 挿入元の配列でコピーするインデックス
   uint          count=WHOLE_ARRAY     // 挿入する要素数
   );
```

パラメータ

dst_array[]

[in][out] 追加先の配列

src_array[]

[in] 追加元の配列

dst_start

[in] ソース配列から要素を挿入するためのターゲット側配列内のインデックス

src_start=0

[in] 開始位置からソース配列の要素が挿入のために取得される、ターゲット配列のインデックス。

count

[in] ソース配列から追加される要素数。WHOLE_ARRAYは指定されたインデックスから配列の最後までのすべての要素を意味します。

戻り値

成功の場合はtrue、それ以外の場合はfalse. エラー情報を取得するには、GetLastError() 関数が呼ばれます。可能なエラーは下記です。

- 5052 - ERR_SMALL_ARRAY(*start*および/または*count* パラメータは間違って設定されている、または*src_array[]*ソース配列が空である)
- 5056 - ERR_SERIES_ARRAY(配列が変更不可能、指標バッファ)
- 4006 - ERR_INVALID_ARRAY(自分自身へのコピーは許可されていない、配列の型が異なる、または、クラスオブジェクトまたはデストラクタ構造体を含む固定サイズの配列がある)
- 4005 - ERR_STRUCT_WITHOBJECTS_ORCLASS(配列にPOD構造体が含まれていないため、シンプルなコピーが不可能)。
- 配列サイズArrayRemove()関数の説明にあるターゲット配列サイズである*dst_array[]*を変更するときにエラーが発生しました。

注意事項

この関数が固定サイズの配列に使用されている場合、ターゲット側の *dst_array[]*配列自体のサイズは変わりません。Starting from the *dst_start*位置から初めて、ターゲット配列の要素は右にシフトされます。最後の*counts*要素は失われ、ソース配列からコピーされた要素がとってかわります。

SetIndexBuffer()関数で指標バッファとして指定された動的配列に要素を挿入することはできません。指標バッファの場合、すべてのサイズ変更操作は端末の実行サブシステムによって実行されます。

© 2000-2025, MetaQuotes Ltd.

ソース配列では、要素はsrc_start指数から始めてコピーされます。ソース配列サイズは変更されません。ターゲット配列に追加される要素は、ソース配列要素へのリンクではありません。つまり、要素が後に2つの配列のいずれかで変更されると、あと一つの配列には反映されません。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 固定サイズの配列を宣言し、値を埋める
   int array_dest[10];
   for(int i=0;i<10;i++)
     {
      array_dest[i]=i;
     }
   //--- ソース配列
   int array_source[10];
   for(int i=0;i<10;i++)
     {
      array_source[i]=10+i;
     }
//--- 要素を挿入する前に配列を表示する
   Print("Before calling ArrayInsert()");
   ArrayPrint(array_dest);
   ArrayPrint(array_source);
//--- ソース配列の3つの要素をターゲット配列に挿入して新しいターゲット配列を表示する
   ArrayInsert(array_dest,array_source,4,0,3);
   Print("After calling ArrayInsert()");
   ArrayPrint(array_dest);
/*
  実行結果
ArrayInsert()呼び出し前
   0 1 2 3 4 5 6 7 8 9
ArrayInsert()呼び出し後
   0 1 2 3 10 11 12 7 8 9
*/
```

参照

ArrayRemove、ArrayCopy、ArrayResize、ArrayFree

© 2000-2025, MetaQuotes Ltd.

ArrayRemove

指定されたインデックスから開始して、配列内の指定された数の要素を削除します。

```
bool  ArrayRemove(
   void&         array[],             // 任意の型の配列
   uint          start,               // 削除開始インデックス
   uint          count=WHOLE_ARRAY    // 要素数
   );
```

パラメータ

array[]

[in][out] 配列

start

[in] 配列要素の削除を始めるインデックス

count=WHOLE_ARRAY

[in] 削除される要素の数。WHOLE_ARRAYは指定されたインデックスから配列の最後までのすべての要素を意味します。

戻り値

成功の場合はtrue、それ以外の場合はfalse. エラー情報を取得するには、GetLastError() 関数が呼ばれます。可能なエラーは下記です。

- 5052 - ERR_SMALL_ARRAY(*start*値が大き過ぎる)
- 5056 - ERR_SERIES_ARRAY(配列が変更不可能、指標バッファ)
- 4003 - ERR_INVALID_PARAMETER(*count*値が大き過ぎる)
- 4005 - ERR_STRUCT_WITHOBJECTS_ORCLASS(デストラクタを持つ複雑なオブジェクトを含む固定サイズの配列)
- 4006 - ERR_INVALID_ARRAY(デストラクタを持つオブジェクトまたはクラスオブジェクトを含む固定サイズの配列)

注意事項

この関数が固定サイズの配列に使用されている場合、配列サイズは変わりません。残りの「tail」が物理的に*start*ポジションにコピーされます。関数がどのように機能するかを正確に理解するには、以下の例を参照してください。「物理的」コピーとは、コピーされたオブジェクトがコンストラクタまたはコピー演算子の呼び出しによっては作成されないことを意味します。代わりに、オブジェクトのバイナリ表現がコピーされます。このため、デストラクタを使用してArrayRemove()関数をオブジェクトを含む固定サイズの配列に適用することはできません(ERR_INVALID_ARRAYまたはERR_STRUCT_WITHOBJECTS_ORCLASSエラーが発生しています)。このようなオブジェクトを削除するときは、元のオブジェクトとそのコピーに対して、デストラクタを2回呼び出す必要があります。

SetIndexBuffer()関数で指標バッファとして指定された動的配列から要素を削除することはできません。これにより、ERR_SERIES_ARRAYエラーが発生します。指標バッファの場合、すべてのサイズ変更操作は端末の実行サブシステムによって実行されます。

例:

```
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|  スクリプトプログラムを開始する関数                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 固定サイズの配列を宣言し、値を埋める
   int array[10];
   for(int i=0;i<10;i++)
     {
      array[i]=i;
     }
//--- 要素を削除する前に配列を表示する
   Print("Before calling ArrayRemove()");
   ArrayPrint(array);
//--- 配列から2つの要素を削除してあたらしい集合を表示する
   ArrayRemove(array,4,2);
   Print("After calling ArrayRemove()");
   ArrayPrint(array);
/*
  実行結果:
  ArrayRemove() 呼び出し前
    0 1 2 3 4 5 6 7 8 9
  ArrayRemove() 呼び出し後
    0 1 2 3 6 7 8 9 8 9
*/
```

参照

ArrayInsert、ArrayCopy、ArrayResize、ArrayFree

© 2000-2025, MetaQuotes Ltd.

ArrayReverse

指定されたインデックスから開始して、配列内の指定された数の要素を反転します。

```
bool  ArrayReverse(
   void&         array[],            // 任意の型の配列
   uint          start=0,            // 配列反転開始インデックス
   uint          count=WHOLE_ARRAY   // 要素数
   );
```

パラメータ

array[]

[in][out] 配列

start=0

[in] 反転を開始する配列のインデックス

count=WHOLE_ARRAY

[in] 反転される要素の数。WHOLE_ARRAYの場合、指定された*start*インデックスから配列の最後の要素までの位置が反転されます。

戻り値

成功の場合はtrue、それ以外の場合はfalse

注意事項

ArraySetAsSeries()関数は物理的に配列要素を移動するものではありません。代わりに、要素へのアクセスを時系列同様に変更するために、インデックスの方向を逆方向に変更するだけです。
ArrayReverse()関数は、配列が「反転」されるように物理的に配列要素を移動します。

例:

```
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 固定サイズの配列を宣言し、値を埋める
   int array[10];
   for(int i=0;i<10;i++)
     {
      array[i]=i;
     }
//--- 配列を反転する前に配列を表示する
   Print("Before calling ArrayReverse()");
   ArrayPrint(array);
//--- 配列内の3つの要素を反転して新しい集合を表示する
   ArrayReverse(array,4,3);
   Print("After calling ArrayReverse()");
   ArrayPrint(array);
```

© 2000-2025, MetaQuotes Ltd.

```
/*
   実行結果:
   ArrayReverse() 呼び出し前
     0 1 2 3 4 5 6 7 8 9
   ArrayReverse() 呼び出し後
     0 1 2 3 6 5 4 7 8 9
*/
```

参照

ArrayInsert、ArrayRemove、ArrayCopy、ArrayResize、ArrayFree、ArrayGetAsSeries、ArraySetAsSeries

© 2000–2025, MetaQuotes Ltd.

ArraySetAsSeries

この関数は選択された 動的配列オブジェクトに AS_SERIES フラグを設定し 時系列のように索引付けをします。

```
bool  ArraySetAsSeries(
   const void&  array[],     // 参照配列
   bool         flag         // true はインデックスの順番が反対であることを示す
   );
```

パラメータ

array[]

[in][out] 設定される数値配列

flag

[in] 索引付けの方向

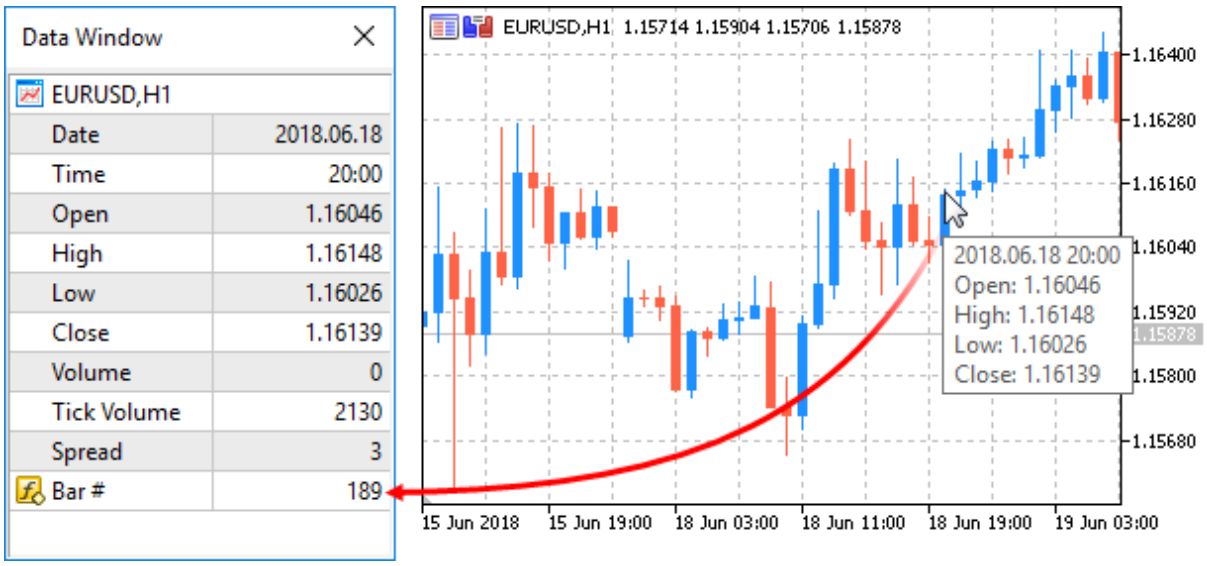
戻り値

成功の場合は true、それ以外の場合は false

注意事項

AS_SERIES フラグは 多次元配列または静的配列（サイズがコンパイル段階ですでに角括弧で定義される配列）には設定することが出来ません。時系列と通常の配列の違いは、時系列のインデックスが後ろから前へ（最新のデータから古いデータへ）つけられていることです。

例：バー番号を表示する指標




```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- プロット記数
#property indicator_label1  "Numeration"
```

© 2000–2025, MetaQuotes Ltd.

```
#property indicator_type1   DRAW_LINE
#property indicator_color1  CLR_NONE
//--- 指標バッファ
double         NumerationBuffer[];
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 指標バッファマッピング
   SetIndexBuffer(0,NumerationBuffer,INDICATOR_DATA);
//--- バッファのインデックスを時系列式に付ける
   ArraySetAsSeries(NumerationBuffer,true);
//--- データウィンドウに表示する精度を設定する
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//--- 指標配列の名称の データウィンドウでの表示の仕方
   PlotIndexSetString(0,PLOT_LABEL,"Bar #");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---  現在のゼロバー開口部の時間を保存する
   static datetime currentBarTimeOpen=0;
//--- time[] 配列へのアクセスを元に返す - 時系列のようにアクセスする
   ArraySetAsSeries(time,true);
//--- ゼロバーの時間が格納されているものと異なる場合
   if(currentBarTimeOpen!=time[0])
     {
     //--- 現在からチャートの深さまで全てのバーを列挙する
      for(int i=rates_total-1;i>=0;i--) NumerationBuffer[i]=i;
      currentBarTimeOpen=time[0];
     }
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
```

© 2000-2025, MetaQuotes Ltd.

参照

時系列へのアクセス、ArrayGetAsSeries

© 2000–2025, MetaQuotes Ltd.

ArraySize

この関数は配列の要素数を返します。

```
int  ArraySize(
   const void&  array[]     // チェックされた配列
   );
```

パラメータ

array[]

[in] 任意型の配列

戻り値

int 型の値

注意事項

1次元配列の場合 ArraySize の返す値は ArrayRange(array,0) の返す値と等しいです。

例:

```
void OnStart()
  {
//--- 配列を作成する
   double one_dim[];
   double four_dim[][10][5][2];
//--- サイズ
   int one_dim_size=25;
   int reserve=20;
   int four_dim_size=5;
//--- 補助変数
   int size;
//--- バックアップせずにメモリを割り当てる
   ArrayResize(one_dim,one_dim_size);
   ArrayResize(four_dim,four_dim_size);
//--- 1. 1 次元配列
   Print("+============================================================+");
   Print("Array sizes:");
   Print("1. One-dimensional array");
   size=ArraySize(one_dim);
   PrintFormat("Zero dimension size = %d, Array size = %d",one_dim_size,size);
//--- 2. 多次元配列
   Print("2. Multidimensional array");
   size=ArraySize(four_dim);
   PrintFormat("Zero dimension size = %d, Array size = %d",four_dim_size,size);
//--- 次元サイズ
   int d_1=ArrayRange(four_dim,1);
   int d_2=ArrayRange(four_dim,2);
   int d_3=ArrayRange(four_dim,3);
   Print("Check:");
```

© 2000-2025, MetaQuotes Ltd.

```
   Print("Zero dimension = Array size / (First dimension * Second dimension * Third
dimension)");
   PrintFormat("%d = %d / (%d * %d * %d)",size/(d_1*d_2*d_3),size,d_1,d_2,d_3);
//--- 3.  メモリバックアップと 1 次元配列
   Print("3. One-dimensional array with memory backup");
//--- 値を倍増する
   one_dim_size*=2;
//--- メモリを割り当ててバックアップする
   ArrayResize(one_dim,one_dim_size,reserve);
//--- サイズをプリントする
   size=ArraySize(one_dim);
   PrintFormat("Size with backup = %d, Actual array size = %
d",one_dim_size+reserve,size);
  }
```

© 2000–2025, MetaQuotes Ltd.

ArraySort

多次元数値配列の初次元の値を昇順にソートします。

```
bool  ArraySort(
   void&  array[]      // ソートする配列
   );
```

パラメータ

array[]

[in][out] ソートする数値配列

戻り値

成功の場合は true、それ以外の場合は false

注意事項

配列は AS_SERIES フラグ値に関係なく昇順にソートされます。

ArraySort と ArrayBSearch 関数はパラメータとして、任意の次元の配列を受け入れます。しかし、検索ソートは常に最初（ゼロ）の次元に適用されます。

例:

```
#property description "The indicator analyzes data for the last month and draws all candlesticks with small"
#property description "and large tick volumes. The tick volume array is sorted out"
#property description "to define such candlesticks. The candlesticks having the volumes comprising the first InpSmallVolume"
#property description "per cent of the array are considered small. The candlesticks having the tick volumes comprising "
#property description "the last InpBigVolume per cent of the array are considered large."
//--- 指標の設定
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   1
//--- プロット
#property indicator_label1  "VolumeFactor"
#property indicator_type1   DRAW_COLOR_CANDLES
#property indicator_color1  clrDodgerBlue,clrOrange
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- 定義済み定数
#define INDICATOR_EMPTY_VALUE 0.0
//--- 入力パラメータ
input int InpSmallVolume=15; // 小ボリューム( <50 )の百分率値
input int InpBigVolume=20;   // 大ボリューム( <50 )の百分率値
//--- 分析開始時間（移動される）
datetime ExtStartTime;
//--- 指標バッファ
```

© 2000-2025, MetaQuotes Ltd.

```
double   ExtOpenBuff[];
double   ExtHighBuff[];
double   ExtLowBuff[];
double   ExtCloseBuff[];
double   ExtColorBuff[];
//--- ローソク足を表示するためのボリュームの境界値
long     ExtLeftBorder=0;
long     ExtRightBorder=0;
//+------------------------------------------------------------------+
//|  ティックボリュームの境界値を受け取る                                |
//+------------------------------------------------------------------+
bool GetVolumeBorders(void)
  {
//--- 変数
   datetime stop_time;  // 終了時刻を複製する
   long     buff[];     // 複製に使用するバッファ
//--- 終了時刻が現在の時刻
   stop_time=TimeCurrent();
//--- start time は現在時間より1か月前である
   ExtStartTime=GetStartTime(stop_time);
//--- ティックボリュームの値を受け取る
   ResetLastError();
   if(CopyTickVolume(Symbol(),Period(),ExtStartTime,stop_time,buff)==-1)
     {
      //--- データの受信に失敗し、再計算コマンドを起動するために false を返す
      PrintFormat("Failed to receive tick volume values. Error code = %
d",GetLastError());
      return(false);
     }
//--- 配列サイズを計算する
   int size=ArraySize(buff);
//--- 配列をソートする
   ArraySort(buff);
//--- ティックボリュームの左と右の境界線の値を定義する
   ExtLeftBorder=buff[size*InpSmallVolume/100];
   ExtRightBorder=buff[(size-1)*(100-InpBigVolume)/100];
//--- 実行成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  渡されたものよりも1ヶ月前のデータを受け取る                        |
//+------------------------------------------------------------------+
datetime GetStartTime(const datetime stop_time)
  {
//--- 終了時間を MqlDateTime 型構造変数に変換する
   MqlDateTime temp;
   TimeToStruct(stop_time,temp);
//--- 1ヶ月前のデータを受け取る
   if(temp.mon>1)
```

© 2000-2025, MetaQuotes Ltd.

```
      temp.mon-=1;  // 現在の月が 1 月ではないので、前月の数が 1 小さい
   else
     {
      temp.mon=12;  // 現在の月が 1 月なので、前月の数は 12
      temp.year-=1; // 年番号が 1 少ないうち
     }
//--- 日数は 28 を超えない
   if(temp.day>28)
      temp.day=28;
//--- 取得された日付を返す
   return(StructToTime(temp));
  }
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 入力パラメータが条件を満たしているかどうかを確認する
   if(InpSmallVolume<0 || InpSmallVolume>=50 || InpBigVolume<0 || InpBigVolume>=50)
     {
      Print("Incorrect input parameters");
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- 指標バッファマッピング
   SetIndexBuffer(0,ExtOpenBuff);
   SetIndexBuffer(1,ExtHighBuff);
   SetIndexBuffer(2,ExtLowBuff);
   SetIndexBuffer(3,ExtCloseBuff);
   SetIndexBuffer(4,ExtColorBuff,INDICATOR_COLOR_INDEX);
//--- 表示されない値を設定する
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,INDICATOR_EMPTY_VALUE);
//--- 指標バッファのラベルを設定する
   PlotIndexSetString(0,PLOT_LABEL,"Open;High;Low;Close");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- 未処理のバーがまだ存在しているかどうかを確認する
   if(prev_calculated<rates_total)
     {
      //--- ボリュームの左右の境界線の新しい値を受け取る
      if(!GetVolumeBorders())
         return(0);
     }
//--- バーの計算に使用されるstart 変数
   int start=prev_calculated;
//--- 指標値が前のティックで計算されている場合には最後のバーを使用する
   if(start>0)
      start--;
//--- 時系列に直接のインデックス作成を設定する
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(close,false);
   ArraySetAsSeries(tick_volume,false);
//--- 指標値の計算ループ
   for(int i=start;i<rates_total;i++)
     {
      //--- 最初の日からローソク足に必要事項を記入する
      if(ExtStartTime<=time[i])
        {
         //--- 値が右の境界以上であれば、ローソク足に必要事項を記入する
         if(tick_volume[i]>=ExtRightBorder)
           {
            //--- ローソク足を描画するためのデータを受信する
            ExtOpenBuff[i]=open[i];
            ExtHighBuff[i]=high[i];
            ExtLowBuff[i]=low[i];
            ExtCloseBuff[i]=close[i];
            //--- DodgerBlue 色
            ExtColorBuff[i]=0;
            //--- ループを続行する
            continue;
           }
         //--- 値が左の境界線を超えていない場合にはローソク足に必要事項を記入する
         if(tick_volume[i]<=ExtLeftBorder)
           {
            //--- ローソク足を描画するためのデータを受信する
            ExtOpenBuff[i]=open[i];
            ExtHighBuff[i]=high[i];
            ExtLowBuff[i]=low[i];
            ExtCloseBuff[i]=close[i];
            //--- 橙色
            ExtColorBuff[i]=1;
```

© 2000-2025, MetaQuotes Ltd.

```
               //--- ループを続行する
               continue;
              }
           }
         //--- 計算に含まれていないバーの空の値を設定する
         ExtOpenBuff[i]=INDICATOR_EMPTY_VALUE;
         ExtHighBuff[i]=INDICATOR_EMPTY_VALUE;
         ExtLowBuff[i]=INDICATOR_EMPTY_VALUE;
         ExtCloseBuff[i]=INDICATOR_EMPTY_VALUE;
        }
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
```

参照

ArrayBsearch

© 2000-2025, MetaQuotes Ltd.

ArraySwap

同じ型の2つの動的配列の内容を交換します。多次元配列の場合、最初の次元を除くすべての次元の要素の数は一致する必要があります。

```
bool  ArraySwap(
   void&  array1[],      // 1番目の配列
   void&  array2[]       // 2番目の配列</
   );
```

パラメータ

array1[]

[in][out] 数値型配列

array2[]

[in][out] 数値型配列

戻り値

成功の場合はtrue、それ以外の場合はfalse。この場合、GetLastError()はエラーコードERR_INVALID_ARRAYを返します。

注意事項

この関数は、最初のものを除いて、同じ型および同じ次元の動的配列を受け入れます。整数型の場合、符号は無視されます（char==uchar）。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 相場を格納する配列
   double source_array[][8];
   double   dest_array[][8];
   MqlRates rates[];
//--- 現在の時間枠で最後の20本のローソク足のデータを取得する
   int copied=CopyRates(NULL,0,0,20,rates);
   if(copied<=0)
     {
      PrintFormat("CopyRates(%s,0,0,20,rates) failed, error=%d",
                  Symbol(),GetLastError());
      return;
     }
//--- コピーされたデータ量の配列サイズを設定する
   ArrayResize(source_array,copied);
//--- rate_array_1[]配列にrates[]のデータを書き入れる
   for(int i=0;i<copied;i++)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      source_array[i][0]=(double)rates[i].time;
      source_array[i][1]=rates[i].open;
      source_array[i][2]=rates[i].high;
      source_array[i][3]=rates[i].low;
      source_array[i][4]=rates[i].close;
      source_array[i][5]=(double)rates[i].tick_volume;
      source_array[i][6]=(double)rates[i].spread;
      source_array[i][7]=(double)rates[i].real_volume;
     }
//--- source_array[]とdest_array[]のデータを互いに取り替える
   if(!ArraySwap(source_array,dest_array))
     {
      PrintFormat("ArraySwap(source_array,rate_array_2) failed, error code=%
d",GetLastError());
      return;
     }
//--- スワップ後にソース配列が0になっていることを確認する
   PrintFormat("ArraySwap() done: ArraySize(source_array)=%
d",ArraySize(source_array));
//--- dest_array []出力先配列のデータを表示する
   ArrayPrint(dest_array);
  }
```

参照

ArrayCopy、ArrayFill、ArrayRange、ArrayIsDynamic

© 2000-2025, MetaQuotes Ltd.

ArrayToFP16

float型またはdouble型の配列を、指定された形式のushort型の配列にコピーします。

```
bool   ArrayToFP16(
   const ushort&         dst_array[],         // copy to
   const float&          src_array[],         // copy from
   ENUM_FLOAT16_FORMAT   fmt                  // format
   );
```

double型のオーバーロード

```
bool   ArrayToFP16(
   const ushort&         dst_array[],         // copy to
   const double&         src_array[],         // copy from
   ENUM_FLOAT16_FORMAT   fmt                  // format
   );
```

パラメータ

dst_array[]

[out] ushort型の受信側配列

src_array[]

[in] float型またはdouble型のソース配列

fmt

[in] ENUM_FLOAT16_FORMAT列挙からのコピー形式

戻り値

成功の場合はtrue、それ以外の場合はfalse

注意事項

FLOAT16およびBFLOAT16はENUM_FLOAT16_FORMAT列挙で定義されており、MQL5ではONNXモデルでの操作にのみ使用されます。

この関数は、float型またはdouble型の入力パラメータをFLOAT16型またはBFLOAT16型に変換します。これらの入力パラメータは、OnnxRun関数で使用されます。

FLOAT16は半精度浮動小数点数とも呼ばれ、16ビットを使用して浮動小数点数を表します。この形式では、精度と計算効率のバランスが提供されます。FLOAT16は、大規模なデータセットの高性能処理を必要とする深層学習アルゴリズムやニューラルネットワークで広く使用されています。この形式は、数値のサイズを小さくすることで計算を高速化します。これは、GPUでディープニューラルネットワークを訓練する場合に特に重要です。

BFLOAT16（またはBrain Floating Point 16）も16 ビットを使用しますが、形式表現のアプローチがFLOAT16とは異なります。この形式では、8ビットが指数を表すために割り当てられ、残りの7ビットが仮数を表すために使用されます。この形式は、深層学習と人工知能、特にGoogleのTensor Processing Unit

© 2000–2025, MetaQuotes Ltd.

（TPU）で使用するために開発されました。BFLOAT16は、ニューラルネットワークの訓練で優れたパフォーマンスを示し、計算を効果的に高速化できます。

例：記事の関数float16およびfloat8形式のONNXモデルの操作

```
//+------------------------------------------------------------------+
//| RunCastFloat16ToDouble                                           |
//+------------------------------------------------------------------+
bool RunCastFloat16ToDouble(long model_handle)
  {
   PrintFormat("test=%s",__FUNCTION__);
   double test_data[12]= {1,2,3,4,5,6,7,8,9,10,11,12};
   ushort data_uint16[12];
   if(!ArrayToFP16(data_uint16,test_data,FLOAT_FP16))
     {
      Print("error in ArrayToFP16. error code=",GetLastError());
      return(false);
     }
   Print("test array:");
   ArrayPrint(test_data);
   Print("ArrayToFP16:");
   ArrayPrint(data_uint16);
   U<ushort> input_float16_values[3*4];
   U<double> output_double_values[3*4];
   float test_data_float[];
   if(!ArrayFromFP16(test_data_float,data_uint16,FLOAT_FP16))
     {
      Print("error in ArrayFromFP16. error code=",GetLastError());
      return(false);
     }
   for(int i=0; i<12; i++)
     {
      input_float16_values[i].value=data_uint16[i];
      PrintFormat("%d input value =%f  Hex float16 = %s  ushort value=%
d"
,
i
,test_data_float
[i],ArrayToString(input_float16_values[i].uc),input_float16_values[i].value);
     }
   Print("ONNX input array:");
   ArrayPrint(input_float16_values);
   bool
res
=OnnxRun(model_handle,ONNX_NO_CONVERSION,input_float16_values,output_double_values);
   if(!res)
     {
      PrintFormat("error in OnnxRun. error code=%d",GetLastError());
      return(false);
```

© 2000–2025, MetaQuotes Ltd.

```
      }
   Print("ONNX output array:");
   ArrayPrint(output_double_values);
//---
   double sum_error=0.0;
   for(int i=0; i<12; i++)
     {
      double delta=test_data[i]-output_double_values[i].value;
      sum_error+=MathAbs(delta);
      PrintFormat("%d output double %f = %s  difference=%
f",i,output_double_values[i].value,ArrayToString(output_double_values[i].uc),delta);
     }
//---
   PrintFormat("test=%s   sum_error=%f",__FUNCTION__,sum_error);
//---
   return(true);
  }
```

参照

ArrayFromFP16、ArrayCopy

© 2000–2025, MetaQuotes Ltd.

ArrayToFP8

float型またはdouble型の配列を、指定された形式のuchar型の配列にコピーします。

```
bool   ArrayToFP8(
   const uchar&          dst_array[],        // copy to
   const float&          src_array[],        // copy from
   ENUM_FLOAT8_FORMAT    fmt                 // format
   );
```

double型のオーバーロード

```
bool   ArrayToFP8(
   const uchar&          dst_array[],        // copy to
   const double&         src_array[],        // copy from
   ENUM_FLOAT8_FORMAT    fmt                 // format
   );
```

パラメータ

dst_array[]

[out] uchar型の受信側配列

src_array[]

[in] float型またはdouble型のソース配列

fmt

[in] ENUM_FLOAT8_FORMAT列挙からのコピー形式

戻り値

成功の場合はtrue、それ以外の場合はfalse

注意事項

あらゆる種類のFP8形式はENUM_FLOAT8_FORMAT列挙で定義されており、MQL5ではONNXモデルでの操作にのみ使用されます。

この関数は、float型またはdouble型の入力パラメータをFP8型のいずれかに変換します。これらの入力パラメータは、OnnxRun関数で使用されます。

FP8（8ビット浮動小数点）は、浮動小数点数を表すために使用されるデータ型の1つです。FP8では、各数値は8データビットで表され、通常は符号、指数、仮数の3つの要素に分割されます。この形式は精度とストレージ効率のバランスが取れているため、メモリと計算効率を必要とするアプリケーションにとって魅力的です。

FP8はコンパクトな数値表現を採用することにより、メモリ要件を削減し、計算を高速化します。さらに、FP8は、算術計算や信号処理などの低レベルの演算を実装するのにも役立ちます。

例：記事の関数float16およびfloat8形式のONNXモデルの操作

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| RunCastFloat8Float                                               |
//+------------------------------------------------------------------+
bool RunCastFloat8ToFloat(long model_handle,const ENUM_FLOAT8_FORMAT fmt)
  {
   PrintFormat("TEST: %s(%s)",__FUNCTION__,EnumToString(fmt));
//---
   float test_data[15]   = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};
   uchar data_float8[15] = {};
   if(!ArrayToFP8(data_float8,test_data,fmt))
     {
      Print("error in ArrayToFP8. error code=",GetLastError());
      OnnxRelease(model_handle);
      return(false);
     }
   U<uchar> input_float8_values[3*5];
   U<float> output_float_values[3*5];
   float    test_data_float[];
//--- convert float8 to float
   if(!ArrayFromFP8(test_data_float,data_float8,fmt))
     {
      Print("error in ArrayFromFP8. error code=",GetLastError());
      OnnxRelease(model_handle);
      return(false);
     }
   for(uint i=0; i<data_float8.Size(); i++)
     {
      input_float8_values[i].value=data_float8[i];
      PrintFormat("%d input value =%f  Hex float8 = %s  ushort value=%
d"
,
i
,test_data_float
[i],ArrayToHexString(input_float8_values[i].uc),input_float8_values[i].value);
     }
   Print("ONNX input array: ",ArrayToString(input_float8_values));
//--- execute model (convert float8 to float using ONNX)
   if(!OnnxRun
(model_handle,ONNX_NO_CONVERSION,input_float8_values,output_float_values))
     {
      PrintFormat("error in OnnxRun. error code=%d",GetLastError());
      OnnxRelease(model_handle);
      return(false);
     }
   Print("ONNX output array: ",ArrayToString(output_float_values));
//--- calculate error (compare ONNX and ArrayFromFP8 results)
   double sum_error=0.0;
   for(uint i=0; i<test_data.Size(); i++)
     {
      double delta=test_data_float[i]-(double)output_float_values[i].value;
```

© 2000-2025, MetaQuotes Ltd.

```
      sum_error+=MathAbs(delta);
      PrintFormat("%d output float %f = %s difference=%
f"
,i,output_float_values[i].value,ArrayToHexString(output_float_values[i].uc),delta);
     }
//---
   PrintFormat("%s(%s): sum_error=%f\n",__FUNCTION__,EnumToString(fmt),sum_error);
   return(true);
  }
```

参照

ArrayFromFP8、ArrayCopy

© 2000–2025, MetaQuotes Ltd.

ArrayFromFP16

ushort型の配列を、指定された形式のfloat型またはdouble型の配列にコピーします。

```
bool   ArrayFromFP16(
   const float&          dst_array[],        // copy to
   const ushort&         src_array[],        // copy from
   ENUM_FLOAT16_FORMAT   fmt                 // format
   );
```

double型のオーバーロード

```
bool   ArrayFromFP16(
   const double&         dst_array[],        // copy to
   const ushort&         src_array[],        // copy from
   ENUM_FLOAT16_FORMAT   fmt                 // format
   );
```

パラメータ

dst_array[]

[out] float型またはdouble型の受信側配列

src_array[]

[in] ushort型のソース配列

fmt

[in] ENUM_FLOAT16_FORMAT列挙からのコピー形式

戻り値

成功の場合はtrue、それ以外の場合はfalse

注意事項

FLOAT16およびBFLOAT16はENUM_FLOAT16_FORMAT列挙で定義されており、MQL5ではONNXモデルでの操作にのみ使用されます。

OnnxRun関数の実行から取得した出力パラメータがFLOAT16型またはBFLOAT16型である場合、この関数を使用して、結果をfloatまたはdouble配列に変換できます。

FLOAT16は半精度浮動小数点数とも呼ばれ、16ビットを使用して浮動小数点数を表します。この形式では、精度と計算効率のバランスが提供されます。FLOAT16は、大規模なデータセットの高性能処理を必要とする深層学習アルゴリズムやニューラルネットワークで広く使用されています。この形式は、数値のサイズを小さくすることで計算を高速化します。これは、GPUでディープニューラルネットワークを訓練する場合に特に重要です。

BFLOAT16（またはBrain Floating Point 16）も16 ビットを使用しますが、形式表現のアプローチがFLOAT16とは異なります。この形式では、8ビットが指数を表すために割り当てられ、残りの7ビットが仮数を表すために使用されます。この形式は、深層学習と人工知能、特にGoogleのTensor Processing Unit

© 2000–2025, MetaQuotes Ltd.

（TPU）で使用するために開発されました。BFLOAT16は、ニューラルネットワークの訓練で優れたパフォーマンスを示し、計算を効果的に高速化できます。

例：記事の関数float16およびfloat8形式のONNXモデルの操作

```
//+------------------------------------------------------------------+
//| RunCastFloat16ToDouble                                           |
//+------------------------------------------------------------------+
bool RunCastFloat16ToDouble(long model_handle)
  {
   PrintFormat("test=%s",__FUNCTION__);
   double test_data[12]= {1,2,3,4,5,6,7,8,9,10,11,12};
   ushort data_uint16[12];
   if(!ArrayToFP16(data_uint16,test_data,FLOAT_FP16))
     {
      Print("error in ArrayToFP16. error code=",GetLastError());
      return(false);
     }
   Print("test array:");
   ArrayPrint(test_data);
   Print("ArrayToFP16:");
   ArrayPrint(data_uint16);
   U<ushort> input_float16_values[3*4];
   U<double> output_double_values[3*4];
   float test_data_float[];
   if(!ArrayFromFP16(test_data_float,data_uint16,FLOAT_FP16))
     {
      Print("error in ArrayFromFP16. error code=",GetLastError());
      return(false);
     }
   for(int i=0; i<12; i++)
     {
      input_float16_values[i].value=data_uint16[i];
      PrintFormat("%d input value =%f  Hex float16 = %s  ushort value=%
d"
,
i
,test_data_float
[i],ArrayToString(input_float16_values[i].uc),input_float16_values[i].value);
     }
   Print("ONNX input array:");
   ArrayPrint(input_float16_values);
   bool
res
=OnnxRun(model_handle,ONNX_NO_CONVERSION,input_float16_values,output_double_values);
   if(!res)
     {
      PrintFormat("error in OnnxRun. error code=%d",GetLastError());
      return(false);
```

© 2000–2025, MetaQuotes Ltd.

```
      }
   Print("ONNX output array:");
   ArrayPrint(output_double_values);
//---
   double sum_error=0.0;
   for(int i=0; i<12; i++)
     {
      double delta=test_data[i]-output_double_values[i].value;
      sum_error+=MathAbs(delta);
      PrintFormat("%d output double %f = %s  difference=%
f",i,output_double_values[i].value,ArrayToString(output_double_values[i].uc),delta);
     }
//---
   PrintFormat("test=%s   sum_error=%f",__FUNCTION__,sum_error);
//---
   return(true);
  }
```

参照

ArrayToFP16、ArrayCopy

© 2000–2025, MetaQuotes Ltd.

ArrayFromFP8

uchar型の配列を、指定された形式のfloat型またはdouble型の配列にコピーします。

```
bool   ArrayFromFP8(
   const float&          dst_array[],        // copy to
   const uchar&          src_array[],        // copy from
   ENUM_FLOAT8_FORMAT    fmt                 // format
   );
```

double型のオーバーロード

```
bool   ArrayFromFP8(
   const double&         dst_array[],        // copy to
   const uchar&          src_array[],        // copy from
   ENUM_FLOAT8_FORMAT    fmt                 // format
   );
```

パラメータ

dst_array[]

[out] float型またはdouble型の受信側配列

src_array[]

[in] uchar型のソース配列

fmt

[in] ENUM_FLOAT8_FORMAT列挙からのコピー形式

戻り値

成功の場合はtrue、それ以外の場合はfalse

注意事項

あらゆる種類のFP8形式はENUM_FLOAT8_FORMAT列挙で定義されており、MQL5ではONNXモデルでの操作にのみ使用されます。

OnnxRun関数の実行から取得した出力パラメータがENUM_FLOAT8_FORMAT列挙型のFP8である場合、この関数を使用して、結果をfloatまたはdouble配列に変換できます。

FP8（8ビット浮動小数点）は、浮動小数点数を表すために使用されるデータ型の1つです。FP8では、各数値は8データビットで表され、通常は符号、指数、仮数の3つの要素に分割されます。この形式は精度とストレージ効率のバランスが取れているため、メモリと計算効率を必要とするアプリケーションにとって魅力的です。

FP8はコンパクトな数値表現を採用することにより、メモリ要件を削減し、計算を高速化します。さらに、FP8は、算術計算や信号処理などの低レベルの演算を実装するのにも役立ちます。

例：記事の関数float16およびfloat8形式のONNXモデルの操作

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| RunCastFloat8Float                                               |
//+------------------------------------------------------------------+
bool RunCastFloat8ToFloat(long model_handle,const ENUM_FLOAT8_FORMAT fmt)
  {
   PrintFormat("TEST: %s(%s)",__FUNCTION__,EnumToString(fmt));
//---
   float test_data[15]   = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};
   uchar data_float8[15] = {};
   if(!ArrayToFP8(data_float8,test_data,fmt))
     {
      Print("error in ArrayToFP8. error code=",GetLastError());
      OnnxRelease(model_handle);
      return(false);
     }
   U<uchar> input_float8_values[3*5];
   U<float> output_float_values[3*5];
   float    test_data_float[];
//--- convert float8 to float
   if(!ArrayFromFP8(test_data_float,data_float8,fmt))
     {
      Print("error in ArrayFromFP8. error code=",GetLastError());
      OnnxRelease(model_handle);
      return(false);
     }
   for(uint i=0; i<data_float8.Size(); i++)
     {
      input_float8_values[i].value=data_float8[i];
      PrintFormat("%d input value =%f  Hex float8 = %s  ushort value=%
d"
,
i
,test_data_float
[i],ArrayToHexString(input_float8_values[i].uc),input_float8_values[i].value);
     }
   Print("ONNX input array: ",ArrayToString(input_float8_values));
//--- execute model (convert float8 to float using ONNX)
   if(!OnnxRun
(model_handle,ONNX_NO_CONVERSION,input_float8_values,output_float_values))
     {
      PrintFormat("error in OnnxRun. error code=%d",GetLastError());
      OnnxRelease(model_handle);
      return(false);
     }
   Print("ONNX output array: ",ArrayToString(output_float_values));
//--- calculate error (compare ONNX and ArrayFromFP8 results)
   double sum_error=0.0;
   for(uint i=0; i<test_data.Size(); i++)
     {
      double delta=test_data_float[i]-(double)output_float_values[i].value;
```

© 2000-2025, MetaQuotes Ltd.

```
      sum_error+=MathAbs(delta);
      PrintFormat("%d output float %f = %s difference=%
f"
,i,output_float_values[i].value,ArrayToHexString(output_float_values[i].uc),delta);
     }
//---
   PrintFormat("%s(%s): sum_error=%f\n",__FUNCTION__,EnumToString(fmt),sum_error);
   return(true);
  }
```

参照

ArrayToFP8、ArrayCopy

© 2000–2025, MetaQuotes Ltd.

行列とベクトル

行列とは、double、float、または複素数の2次元配列です。

ベクトルとは、double、float、または複素数の1次元配列です。ベクトルでは、垂直か水平かは示されず、これは使用状況から判断されます。たとえば、ドット積ベクトル演算では、左のベクトルが水平で、右のベクトルが垂直であると想定します。種類を示すことが必要な場合は、1行または1列の行列を使用することができますが、これは通常必要ありません。

行列とベクトルでは、データのメモリを動的に割り当てます。実際、行列とベクトルは、含まれるデータの種類や次元など、特定のプロパティを持つオブジェクトです。行列とベクトルのプロパティは、vector_a.Size()、matrix_b.Rows()、vector_c.Norm()、matrix_d.Cond()などのメソッドを使用して取得できます。任意の次元を変更することができます。

行列を作成して初期化するときには、いわゆる静的メソッドが使用されます（これらはクラスの静的メソッドのようなものです）。matrix::Eye()、matrix::Identity()、matrix::Ones()、vector::Ones()、matrix::Zeros()、vector::Zeros()、matrix::Full( )、vector::Full()、matrix::Tri()はその例です。

現時点では、この開発の方向性がまだ完了していないため、行列演算とベクトル演算においての複素数データ型の使用は示唆されていません。

MQL5は、行列とベクトルのDLLへの受け渡しをサポートしています。これにより、関連する型を使用する関数を外部変数からインポートできます。

行列とベクトルは、バッファへのポインタとしてDLLに渡されます。たとえば、float型の行列を渡すには、DLLからエクスポートされた関数の対応するパラメータが、float型のバッファポインタを受け取る必要があります。

MQL5

```
#import "mmlib.dll"
bool sgemm(uint flags, matrix<float> &C, const matrix<float> &A, const matrix<float> &B, ulong M, ulong N, ulong K, float alpha, float beta);
#import
```

C++

```
extern "C" __declspec(dllexport) bool sgemm(UINT flags, float *C, const float *A, const float *B, UINT64 M, UINT64 N, UINT64 K, float alpha, float beta)
```

正しく処理するためには、バッファに加えて行列とベクトルのサイズを渡す必要があります。

以下では、すべての行列メソッドとベクトルメソッドをアルファベット順に示します。

関数	アクション	カテゴリ
Activation	活性化関数の値を計算し、渡されたベクトル/行列に書き込みます	機械学習
ArgMax	最大値のインデックスを返します	統計
ArgMin	最小値のインデックスを返します	統計

© 2000–2025, MetaQuotes Ltd.

関数	アクション	カテゴリ
ArgSort	並び替えられたインデックスを返します	操作
Assign	自動キャストで行列、ベクトル、または配列をコピーします	初期化
Average	行列/ベクトル値の加重平均を計算します	統計
Cholesky	コレスキー分解を計算します	変換
Clip	行列/ベクトルの要素を有効な値の与えられた範囲に制限します	操作
Col	列ベクトルを返します。指定された列にベクトルを書き込みます。	操作
Cols	行列の列数を返します	特徴
Compare	指定された精度で2つの行列/ベクトルの要素を比較します	操作
CompareByDigits	2つの行列/ベクトルの要素を有効数字の精度で比較します	操作
Cond	行列の条件数を計算します	特徴
Convolve	2つのベクトルの離散線形畳み込みを返します	積
Copy	指定された行列/ベクトルのコピーを返します	操作
CopyIndicatorBuffer	指定された指標バッファのデータを指定された数量でベクトルに取得します	初期化
CopyRates	指定された量の指定された銘柄期間のMqlRates構造体の履歴シリーズを行列/ベクトルに取得します	初期化
CopyTicks	MqlTick構造体からティックを行列/ベクトルに取得します	初期化
CopyTicksRange	指定された日付範囲内のMqlTick構造体からティックを取得して行列/ベクトルにします	初期化
CorrCoef	ピアソン相関係数(線形相関係数)を計算します	積
Correlate	2つのベクトルの相互相関を計算します	積

© 2000–2025, MetaQuotes Ltd.

関数	アクション	カテゴリ
Cov	共分散行列を計算します	積
CumProd	指定された軸に沿ったものを含む、行列/ベクトル要素の累積積を返します	統計
CumSum	指定された軸に沿ったものを含む、行列/ベクトル要素の累積和を返します	統計
Derivative	活性化関数の導関数値を計算し、渡されたベクトル/行列に書き込みます	機械学習
Det	可逆正方行列の行列式を計算します	特徴
Diag	対角を抽出するか、対角行列を構築します	操作
Dot	2つのベクトルの内積	積
Eig	正方行列の固有値と右固有ベクトルを計算します	変換
EigVals	一般行列の固有値を計算します	変換
Eye	対角に1を、それ以外に0を含む行列を返します	初期化
Fill	既存の行列/ベクトルに指定された値を書き入れます	初期化
Flat	2つではなく1つのインデックスを介して行列要素にアクセスします	操作
Full	指定された値で埋められた新しい行列を作成して返します	初期化
GeMM	GeMM (General Matrix Multiply) メソッドは、2つの行列の汎用乗算を実装します	積
Hsplit	行列を複数の部分行列に水平に分割します（axis=0のSplitと同じ）	操作
Identity	指定されたサイズの単位行列を作成します	初期化
Init	行列/ベクトルの初期化	初期化
Inner	2つの行列の内積	積

© 2000–2025, MetaQuotes Ltd.

関数	アクション	カテゴリ
Inv	ジョーダン・ガウス法により正方可逆行列の乗法逆行列を計算します	解
Kron	2つの行列、行列とベクトル、ベクトルと行列、または2つのベクトルのクロネッカー積を返します	積
Loss	損失関数値を計算し、渡されたベクトル/行列に書き込みます	機械学習
LstSq	線形代数方程式の最小二乗解を返します（非正方行列または縮退行列の場合）	解
LU	行列のLU分解を実装します（下三角行列と上三角行列の積）	変換
LUP	部分置換によるLUP分解を実装します（行順列のみのLU分解：PA=LU）	変換
MatMul	2つの行列の行列積	積
Max	行列/ベクトルの最大値を返します	統計
Mean	要素値の算術平均を計算します	統計
Median	行列/ベクトル要素の中央値を計算します	統計
Min	行列/ベクトルの最小値を返します	統計
Norm	行列/ベクトルのノルムを返します	特徴
Ones	1で埋められた新しい行列を作成して返します	初期化
Outer	2つの行列または2つのベクトルの外積を計算します	積
Percentile	行列/ベクトル要素または指定された軸に沿った要素の値の指定されたパーセンタイルを返します	統計
PInv	ムーア・ペンローズ逆行列法による行列の疑似逆行列を計算します	解
Power	正方行列を整数乗します	積

© 2000–2025, MetaQuotes Ltd.

関数	アクション	カテゴリ
Prod	行列/ベクトル要素の積を返します（指定された軸に対しても実行可能）	統計
Ptp	行列/ベクトルまたは指定された行列軸の値の範囲を返します	統計
QR	行列のQR分解を計算します	変換
Quantile	行列/ベクトル要素または指定された軸に沿った要素の値の指定された分位数を返します	統計
Rank	ガウス法を使用して行列ランクを返します	特徴
RegressionMetric	指定されたデータ配列で構築された回帰直線からの偏差エラーとして回帰指標を計算します	統計
Reshape	データを変更せずに行列の形状を変更します	操作
Resize	形状とサイズが変更された新しい行列を返します	操作
Row	行ベクトルを返します。指定された行にベクトルを書き込みます	操作
Rows	行列の行数を返します	特徴
Set	指定されたインデックスのベクトル要素の値を設定します	操作
Size	ベクトルのサイズを返します	特徴
SLogDet	行列式の符号と対数を計算します	特徴
Solve	線形行列方程式または線形代数方程式系を解きます	解
Sort	場所によって並び替えます	操作
Spectrum	積AT*Aからの固有値のセットとして行列のスペクトルを計算します	特徴
Split	行列を複数の部分行列に分割します	操作
Std	行列/ベクトル要素または指定された軸に沿った要素の値の標準偏差を返します	統計
Sum	行列/ベクトル要素の合計を返します（指定された軸に対しても実	統計

© 2000–2025, MetaQuotes Ltd.

関数	アクション	カテゴリ
	行可）	
SVD	特異値分解	変換
SwapCols	行列の列を入れ替えます	操作
SwapRows	行列の行を入れ替えます	操作
Trace	行列の対角線に沿って合計を返します	特徴
Transpose	転置（軸を交換）して、修正された行列を返します	操作
Tri	指定された対角線とその下に1があり、それ以外が0の行列を作成します	初期化
TriL	k番目の対角より上の要素をゼロにした行列のコピーを返します（下三角行列）	操作
TriU	k番目の対角より下の要素を0にした行列のコピーを返します（上三角行列）	操作
Var	行列/ベクトル要素の値の分散を計算します	統計
Vsplit	行列を複数の部分行列に垂直に分割します（axis=1のSplitと同じ）	操作
Zeros	0で埋められた新しい行列を作成して返します	初期化

© 2000-2025, MetaQuotes Ltd.

行列とベクトルの種類

行列とベクトルは、線形代数操作を可能にするMQL5の特別なデータ型です。次のデータ型が存在します。

- matrix — double要素を含む行列
- matrixf — float要素を含む行列<分
- matrixc — complex要素を含む行列
- vector — double要素を含むベクトル
- vectorf — float要素を含むベクトル
- vectorc — complex要素を含むベクトル

テンプレート関数は、対応する型の代わりに、matrix<double>、matrix<float>、vector<double>、vector<float> などの表記をサポートします。

行列とベクトルの初期化メソッド

関数	アクション
Eye	対角に1を、それ以外に0を含む行列を返します
Identity	指定されたサイズの単位行列を作成します
Ones	1で埋められた新しい行列を作成して返します
Zeros	0で埋められた新しい行列を作成して返します
Full	指定された値で満たされた新しい行列を作成して返します
Tri	指定された対角線以下に1を持ち、それ以外が0の行列を作成します
Init	行列/ベクトルを初期化します
Fill	既存の行列/ベクトルに指定された値を書き入れます

© 2000–2025, MetaQuotes Ltd.

行列演算とベクトル演算の列挙

このセクションでは、さまざまな行列とベクトルのメソッドで使用される列挙について説明します。

ENUM_AVERAGE_MODE

平滑化の種類の列挙

ID	説明
AVERAGE_NONE	平均化なし。結果はラベルごとに個別に提供されます
AVERAGE_BINARY	二項分類のラベル1の結果
AVERAGE_MICRO	平均誤差行列の結果（混同行列）
AVERAGE_MACRO	各ラベルの誤差行列の結果の平均結果
AVERAGE_WEIGHTED	加重平均結果

ENUM_VECTOR_NORM

vector::Normでのベクトルノルムの列挙

ID	説明
VECTOR_NORM_INF	無限大ノルム
VECTOR_NORM_MINUS_INF	負の無限大ノルム
VECTOR_NORM_P	Pノルム

ENUM_MATRIX_NORM

matrix::Normでの行列ノルムおよびmatrix::Condで行列条件数を取得するための列挙

ID	説明
MATRIX_NORM_FROBENIUS	フロベニウスノルム
MATRIX_NORM_SPECTRAL	スペクトルノルム
MATRIX_NORM_NUCLEAR	核型ノルム
MATRIX_NORM_INF	無限大ノルム
MATRIX_NORM_P1	P1ノルム
MATRIX_NORM_P2	P2ノルム

© 2000–2025, MetaQuotes Ltd.

ID	説明
MATRIX_NORM_MINUS_INF	負の無限大ノルム
MATRIX_NORM_MINUS_P1	負のP1ノルム
MATRIX_NORM_MINUS_P2	負のP2ノルム

ENUM_VECTOR_CONVOLVE

vector::Convolveでの畳み込みおよびvector::Correlateでの相互相関の列挙

ID	説明
VECTOR_CONVOLVE_FULL	完全畳み込み
VECTOR_CONVOLVE_SAME	同じ畳み込み
VECTOR_CONVOLVE_VALID	有効な畳み込み

ENUM_REGRESSION_METRIC

vector::RegressionMetricでの回帰指標の列挙

ID	説明
REGRESSION_MAE	平均絶対誤差(MAE)
REGRESSION_MSE	平均二乗誤差(MSE(
REGRESSION_RMSE	二乗平均平方根誤差(RMSE)
REGRESSION_R2	決定係数
REGRESSION_MAPE	平均絶対誤差率(MAPE)
REGRESSION_MSPE	Mean Squared Percentage Error
REGRESSION_RMSLE	二乗平均平方根対数誤差(RMSLE)
REGRESSION_SMAPE	対称平均絶対誤差率
REGRESSION_MAXE	最大絶対誤差
REGRESSION_MEDE	中央値絶対誤差
REGRESSION_MPD	平均ポアソン偏差
REGRESSION_MGD	平均ガンマ偏差
REGRESSION_EXPV	分散説明率

© 2000–2025, MetaQuotes Ltd.

ENUM_CLASSIFICATION_METRIC

分類問題の行列の列挙

ID	説明
CLASSIFICATION_ACCURACY	すべてのクラスの予測精度に関するモデルの品質
CLASSIFICATION_AVERAGE_PRECISION	平均モデル精度
CLASSIFICATION_BALANCED_ACCURACY	バランスの取れた予測精度
CLASSIFICATION_F1	F1スコア。モデルの精度と再現率の間の調和平均
CLASSIFICATION_JACCARD	ジャカードスコア
CLASSIFICATION_PRECISION	ターゲットクラスの真陽性を予測するモデルの精度
CLASSIFICATION_RECALL	モデルの完全性
CLASSIFICATION_ROC_AUC	誤差曲線の下の面積
CLASSIFICATION_TOP_K_ACCURACY	k個の予測ラベルの先頭に表示される正しいラベルの頻度

ENUM_LOSS_FUNCTION

vector::Loss損失関数計算の列挙

ID	説明
LOSS_MSE	平均二乗誤差(MSE)
LOSS_MAE	平均絶対誤差(MAE)
LOSS_CCE	多クラス交差エントロピー
LOSS_BCE	バイナリ交差エントロピー
LOSS_MAPE	平均絶対誤差率(MAPE)
LOSS_MSLE	平均二乗対数誤差(MSLE)
LOSS_KLD	カルバック・ライブラー情報量
LOSS_COSINE	コサイン類似度/近接度
LOSS_POISSON	ポアソン
LOSS_HINGE	ヒンジ
LOSS_SQ_HINGE	二乗ヒンジ
LOSS_CAT_HINGE	多クラスヒンジ
LOSS_LOG_COSH	双曲線余弦の対数

© 2000-2025, MetaQuotes Ltd.

ID	説明
LOSS_HUBER	Huber（フーバー）

ENUM_ACTIVATION_FUNCTION

vector::Activation活性化関数およびvector::Derivative活性化関数導関数の列挙

ID	説明
AF_ELU	ELU (Exponential Linear Unit)
AF_EXP	指数
AF_GELU	GELU (Gaussian Error Linear Unit)
AF_HARD_SIGMOID	ハードシグモイド
AF_LINEAR	線形
AF_LRELU	Leaky_ReLU (Leaky Rectified Linear Unit)
AF_RELU	ReLU (REctified Linear Unit)
AF_SELU	SELU (Scaled Exponential Linear Unit)
AF_SIGMOID	シグモイド
AF_SOFTMAX	Softmax
AF_SOFTPLUS	Softplus
AF_SOFTSIGN	Softsign
AF_SWISH	Swish
AF_TANH	双曲線正接関数
AF_TRELU	Thresholded ReLU (Thresholded Rectified Linear Unit)

ENUM_SORT_MODE

Sort関数の並び替えタイプの列挙

ID	説明
SORT_ASCENDING	昇順で並べ替え
SORT_DESCENDING	降順で並べ替え

ENUM_MATRIX_AXIS

© 2000–2025, MetaQuotes Ltd.

Enumeration for specifying the axis in all statistical functions for matrices.

ID	Description
AXIS_NONE	The axis is not specified. Calculation is performed over all matrix elements, as if it were a vector (see the Flat method).
AXIS_HORZ	Horizontal axis
AXIS_VERT	Vertical axis

© 2000–2025, MetaQuotes Ltd.

初期化

行列とベクトルを宣言して初期化するには、いくつかの方法があります。

関数	アクション
Assign	自動キャストで行列、ベクトル、または配列をコピーします
CopyIndicatorBuffer	指定された指標バッファのデータを指定された数量でベクトルに取得します
CopyRates	指定された量の指定された銘柄期間のMqlRates構造体の履歴シリーズを行列/ベクトルに取得します
CopyTicks	MqlTick構造体からティックを行列/ベクトルに取得します
CopyTicksRange	指定された日付範囲内のMqlTick構造体からティックを取得して行列/ベクトルにします
Eye	対角に1を、それ以外に0を含む行列を返します
Identity	指定されたサイズの単位行列を作成します
Ones	1で埋められた新しい行列を作成して返します
Zeros	0で埋められた新しい行列を作成して返します
Full	指定された値で満たされた新しい行列を作成して返します
Tri	指定された対角線以下に1を持ち、それ以外が0の行列を作成します
Init	行列/ベクトルを初期化します
Fill	既存の行列/ベクトルに指定された値を書き入れます

サイズを指定しない宣言（データのメモリ割り当てなし）

```
matrix          matrix_a;   // double型行列
matrix<double>  matrix_a1;  // double行列を宣言する別の方法。テンプレートで使用可能
matrixf         matrix_a2;  // float行列
matrix<float>   matrix_a3;  // float行列
vector          vector_a;   // doubleベクトル
vector<double>  vector_a1;
vectorf         vector_a2;  // floatベクトル
vector<float>   vector_a3;
```

サイズを指定した宣言（データのメモリ割り当てあり、初期化なし）

```
matrix          matrix_a(128,128);          // パラメータは定数
matrix<double>  matrix_a1(InpRows,InpCols); // または変数のいずれか
matrixf         matrix_a2(1,128);           // 水平ベクトルのアナログ
matrix<float>   matrix_a3(InpRows,1);       // 垂直ベクトルのアナログ
vector          vector_a(256);
vector<double>  vector_a1(InpSize);
```

© 2000-2025, MetaQuotes Ltd.

```
  vectorf         vector_a2(SomeFunc());        // 関数SomeFuncはベクトルサイズの設定に使用されるulong型の数値を返す
  vector<float>   vector_a3(InpSize+16);        // 式はパラメータとして使用できる
```

初期化を伴う宣言（行列とベクトルのサイズは初期化シーケンスによって決定）

```
  matrix          matrix_a={{0.1,0.2,0.3},{0.4,0.5,0.6}};
  matrix<double>  matrix_a1=matrix_a;                       // 同じ型の行列が必要
  matrixf         matrix_a2={{1,0,0},{0,1,0},{0,0,1}};
  matrix<float>   matrix_a3={{1,2},{3,4}};
  vector          vector_a={-5,-4,-3,-2,-1,0,1,2,3,4,5};
  vector<double>  vector_a1={1,5,2.4,3.3};
  vectorf         vector_a2={0,1,2,3};
  vector<float>   vector_a3=vector_a2;                      // 同じ型のベクトルが必要
```

初期化を伴う宣言

```
template<typename T>
void MatrixArange(matrix<T> &mat,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<mat.Rows(); i++)
     {
      for(ulong j=0; j<mat.Cols(); j++,value+=step)
         mat[i][j]=value;
     }
  }
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
...

  matrix  matrix_a(size_m,size_k,MatrixArange,-M_PI,0.1); // 最初にサイズがsize_m x size_kの初期化されていない行列が作成され、次に初期化時に指定されたパラメータを持つ関数MatrixArangeが呼び出される
  matrixf matrix_a1(10,20,MatrixArange);                  // 行列を作成した後、デフォルトのパラメータを持つ関数MatrixArangeが呼び出される
  vector  vector_a(size,VectorArange,-10.0);              // ベクトルを作成した後、1つのパラメータを持つ関数VectorArangeが呼び出される。2 番目のパラメータはデフォルト
  vectorf vector_a1(128,VectorArange);
```

データのメモリは常に動的であるため、行列/ベクトルの次元は変更できることに注意してください。

静的メソッド

© 2000-2025, MetaQuotes Ltd.

特定の方法で初期化された、指定されたサイズの行列とベクトルを作成するための静的メソッド

```
matrix          matrix_a =matrix::Eye(4,5,1);
matrix<double> matrix_a1=matrix::Full(3,4,M_PI);
matrixf         matrix_a2=matrixf::Identity(5,5);
matrixf<float> matrix_a3=matrixf::Ones(5,5);
matrix          matrix_a4=matrix::Tri(4,5,-1);
vector          vector_a =vector::Ones(256);
vectorf         vector_a1=vector<float>::Zeros(16);
vector<float>   vector_a2=vectorf::Full(128,float_value);
```

作成済みの行列とベクトルを初期化するメソッド

```
matrix  matrix_a;
matrix_a.Init(size_m,size_k,MatrixArange,-M_PI,0.1);
matrixf matrix_a1(3,4);
matrix_a1.Init(10,20,MatrixArange);
vector  vector_a;
vector_a.Init(128,VectorArange);
vectorf vector_a1(10);
vector_a1.Init(vector_size,VectorArange,start_value,step);

matrix_a.Fill(double_value);
vector_a1.Fill(FLT_MIN);
matrix_a1.Identity();
```

© 2000–2025, MetaQuotes Ltd.

Assign

自動キャストで行列、ベクトル、または配列をコピーします。

```
bool matrix::Assign(
  const matrix<T>  &mat      // コピーされた行列
   );
bool matrix::Assign(
  const void       &array[] // コピーされた配列
   );
bool vector::Assign(
  const vector<T>  &vec      // コピーされたベクトル
   );
bool vector::Assign(
  const void       &array[] // コピーされた配列
   );
```

パラメータ

m、vまたは配列

[in] 値のコピー元の行列、ベクトル、または配列

戻り値

成功の場合はtrue、それ以外の場合はfalse

注意事項

Copyとは異なり、Assignメソッドでは配列もコピーできます。この場合、自動キャストが行われ、結果の行列/ベクトルがコピーされた配列のサイズに調整されます。

例：

```
//--- 行列をコピーする
  matrix a= {{2, 2}, {3, 3}, {4, 4}};
  matrix b=a+2;
  matrix c;
  Print("matrix a \n", a);
  Print("matrix b \n", b);
  c.Assign(b);
  Print("matrix c \n", a);

//--- 配列を行列にコピーする
  matrix double_matrix=matrix::Full(2,10,3.14);
  Print("double_matrix before Assign() \n", double_matrix);
  int int_arr[5][5]= {{1, 2}, {3, 4}, {5, 6}};
  Print("int_arr: ");
  ArrayPrint(int_arr);
  double_matrix.Assign(int_arr);
```

© 2000-2025, MetaQuotes Ltd.

```
Print("double_matrix after Assign(int_arr) \n", double_matrix);
/*
 matrix a
 [[2,2]
  [3,3]
  [4,4]]
 matrix b
 [[4,4]
  [5,5]
  [6,6]]
 matrix c
 [[2,2]
  [3,3]
  [4,4]]

 double_matrix before Assign()
 [[3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14]
  [3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14]]

 int_arr:
     [,0][,1][,2][,3][,4]
 [0,]   1   2   0   0   0
 [1,]   3   4   0   0   0
 [2,]   5   6   0   0   0
 [3,]   0   0   0   0   0
 [4,]   0   0   0   0   0

 double_matrix after Assign(int_arr)
 [[1,2,0,0,0]
  [3,4,0,0,0]
  [5,6,0,0,0]
  [0,0,0,0,0]
  [0,0,0,0,0]]

*/
```

参照

Copy

© 2000–2025, MetaQuotes Ltd.

CopyIndicatorBuffer

指定された指標バッファのデータを指定された数量でベクトルに取得します。

データは、一番古い要素がベクトルに割り当てられた物理メモリの先頭に配置されるように複製されます。3つの関数オプションがあります。

初期位置と必要な要素数によるアクセス

```
bool  vector::CopyIndicatorBuffer(
   long      indicator_handle,     // 指標ハンドル
   ulong     buffer_index,         // 指標バッファ番号
   ulong     start_pos,            // 複製開始位置
   ulong     count                 // 複製する要素の数
   );
```

開始日付と必要な要素数によるアクセス

```
bool  vector::CopyIndicatorBuffer(
   long      indicator_handle,     // 指標ハンドル
   ulong     buffer_index,         // 指標バッファ番号
   datetime  start_time,           // 複製開始日
   ulong     count                 // 複製する要素の数
   );
```

必要な時間の開始日と終了日によるアクセス

```
bool  vector::CopyIndicatorBuffer(
   long      indicator_handle,     // 指標ハンドル
   ulong     buffer_index,         // 指標バッファ番号
   datetime  start_time,           // 複製開始日
   datetime  stop_time             // 複製終了日
   );
```

パラメータ

indicator_handle

[in] 対応する指標関数によって戻された指標ハンドル

buffer_index

[in] 指標バッファの数

start_pos

[in] 最初に複製された要素のインデックス

count

[in] 複製された要素の数

start_time

[in] 最初の要素に対応するバーの時刻

stop_time

[in] 最後の要素に対応するバーの時刻

© 2000–2025, MetaQuotes Ltd.

戻り値

成功の場合はtrue、エラーが発生した場合は false

注意事項

複製されたデータの要素(インデックスbuffer_indexを持つ指標バッファ)は現在から過去までカウントダウンされるため、開始位置が0に等しい場合は、現在のバー(現在のバーの指標値)です。

未知の量のデータを複製する場合、CopyBuffer()関数は受信ベクトルのサイズを複製されたデータのサイズに分配しようとするため、サイズを指定せずに(データにメモリを割り当てずに)ベクトルを宣言する必要があります。

指標値の部分複製が必要な場合は、必要な数量が複製される中間ベクトルを使用する必要があります。この中間ベクトルから、必要な数の値をメンバーごとに受信側ベクトルの必要な場所に複製できます。

所定量のデータを複製する場合は、不必要なメモリの再割り当てを避けるために、ベクトルを事前に宣言し、そのサイズを指定することをお勧めします。

指標からデータをリクエストするとき、リクエストされた時系列がまだ構築されていない場合、または開始時にサーバからダウンロードする必要がある場合、読み込み/構築中、関数はすぐにfalseを返します。

EAまたはスクリプトからデータを要求するとき、端末が適切なデータをローカルに持っていない場合、またはデータがローカル履歴から構築できるがまだ準備ができていない場合、サーバからのダウンロードが開始されます。この関数は、タイムアウトが経過するまでに準備が整う量を返します。

参照

CopyBuffer

© 2000-2025, MetaQuotes Ltd.

CopyRates

指定された量の指定された銘柄期間のMqlRates構造体の履歴シリーズを行列/ベクトルに取得します。要素は現在から過去までカウントダウンされます。つまり、開始位置が0の場合は現在足を意味します。

最も古い要素が行列/ベクトルの先頭に配置されるように、データがコピーされます。3つの関数オプションがあります。

初期位置と必要な要素数によるアクセス

```
bool  matrix::CopyRates(
   string           symbol,        // 銘柄名
   ENUM_TIMEFRAMES  period,        // 期間
   ulong            rates_mask, // 要求されたシリーズを指定するフラグの組み合わせ
   ulong            start,         // コピーを開始する最初のバーのインデックス
   ulong            count          // コピーする数
   );
```

初期日付と必要な要素数によるアクセス

```
bool  matrix::CopyRates(
   string           symbol,        // 銘柄名
   ENUM_TIMEFRAMES  period,        // 期間
   ulong            rates_mask, // 要求されたシリーズを指定するフラグの組み合わせ
   datetime         from,          // 開始日
   ulong            count          // コピーする数
   );
```

必要な時間の開始日と終了日によるアクセス

```
bool  matrix::CopyRates(
   string           symbol,        // 銘柄名
   ENUM_TIMEFRAMES  period,        // 期間
   ulong            rates_mask, // 要求されたシリーズを指定するフラグの組み合わせ
   datetime         from,          // 開始日
   datetime         to             // 終了日
   );
```

ベクトル法

初期位置と必要な要素数によるアクセス

```
bool  vector::CopyRates(
   string           symbol,        // 銘柄名
   ENUM_TIMEFRAMES  period,        // 期間
   ulong            rates_mask, // 要求されたシリーズを指定するフラグの組み合わせ
   ulong            start,         // コピーを開始する最初のバーのインデックス
   ulong            count          // コピーする数
   );
```

初期日付と必要な要素数によるアクセス

© 2000-2025, MetaQuotes Ltd.

```
bool  vector::CopyRates(
   string           symbol,       // 銘柄名
   ENUM_TIMEFRAMES  period,       // 期間
   ulong            rates_mask, // 要求されたシリーズを指定するフラグの組み合わせ
   datetime         from,         // 開始日
   ulong            count         // コピーする数
   );
```

必要な時間の開始日と終了日によるアクセス

```
bool  vector::CopyRates(
   string           symbol,       // 銘柄名
   ENUM_TIMEFRAMES  period,       // 期間
   ulong            rates_mask, // 要求されたシリーズを指定するフラグの組み合わせ
   datetime         from,         // 開始日
   datetime         to            // 終了日
   );
```

パラメータ

symbol

[in] 銘柄

period

[in] 期間

rates_mask

[in] 要求されたシリーズのタイプを指定するフラグのENUM_COPY_RATES列挙の組み合わせ。 ベクトルにコピーする場合、ENUM_COPY_RATES列挙から1つの値のみを指定できます。そうしないと、エラーが発生します。

start

[in] 最初に複製された要素のインデックス

count

[in] 複製された要素の数

from

[in] 最初の要素に対応するバーの時刻

to

[in] 最後の要素に対応するバーの時刻

戻り値

成功の場合はtrue、エラーの場合はfalse

注意事項

要求されたデータがサーバ上で利用可能なデータに含まれていない場合、この関数はfalseを返します。TERMINAL_MAXBARS（チャートでのバーの最大数）以外のデータが要求された場合も、この関数はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

EAまたはスクリプトからデータを要求するとき、端末が適切なデータをローカルに持っていない場合、またはデータがローカル履歴から構築できるがまだ準備ができていない場合、サーバからのダウンロードが開始されます。 この関数は、タイムアウトが終わるまでに準備ができているデータの量を返します。ただし、履歴のダウンロードは続行されるため、次の同様の要求ではさらに多くのデータを返します。

開始日及び必要な要素の数によってデータが要求された場合、指定された日付以前のデータのみが戻されます。間隔が設定され秒まで数えられます。つまり、値(ボリューム、スプレッド、始値、高値、安値、終値、時刻)が返されたバーの開始日は常に指定された日付以前です。

日付の指定された範囲のデータを要求した場合、この区間からのデータのみが返されます。間隔が設定され秒まで数えられます。つまり、値(ボリューム、スプレッド、指標バッファの値、始値、高値、安値、終値、時刻)が返されたバーの開始時刻は常に指定された区間内にあります。

例えば、現在の曜日が土曜日の場合、*start_time=Last_Tuesday*と*stop_time=Last_Friday*が指定されると、関数は0を返します。週足は常に日曜日に開始され、指定された範囲内には週足が1つもないためです。

現在の未完成のバーに対応する値を返す必要がある場合、*start_pos*=0及び*count*=1を指定して1 番目の呼び出し方法を用います。

ENUM_COPY_RATES

ENUM_COPY_RATES列挙には、行列または配列に渡されるデータの型を指定するフラグが含まれています。フラグの組み合わせにより、1回の要求で履歴から複数のシリーズを取得できます。行列の行の順序は、ENUM_COPY_RATES列挙の値の順序に対応します。言い換えれば、行列内でHighデータを含む行は常にLowデータを含む行よりも高くなります。

ID	値	説明
COPY_RATES_OPEN	1	始値のシリーズ
COPY_RATES_HIGH	2	高値のシリーズ
COPY_RATES_LOW	4	低値のシリーズ
COPY_RATES_CLOSE	8	終値のシリーズ
COPY_RATES_TIME	16	時系列(バーの開く時間) ベクトルと行列(vectordとmatrixf)で floatで時間を取得すると約100秒の損失が発生します。float精度は厳しく制限されており、1<<24より大きい整数はfloatで正確に表現できないためです。
COPY_RATES_VOLUME_TICK	32	ティック数量
COPY_RATES_VOLUME_REAL	64	取引高
COPY_RATES_SPREAD	128	スプレッド
組み合わせ		
COPY_RATES_OHLC	15	始値、高値、安値、終値のシリーズ

© 2000-2025, MetaQuotes Ltd.

ID	値	説明
COPY_RATES_OHLCT	31	始値、高値、安値、終値、時間のシリーズ
データ配置		
COPY_RATES_VERTICAL	32768	受信した系列値は行列内で垂直に配置されます。最も古いデータが行列の最初の行に配置され、最新のデータが最後の行に配置されます。 デフォルトのコピーでは、系列が横軸に沿って行列に追加されます。 このフラグは、行列にコピーする場合にのみ適用されます。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- クォートを行列に取得する
  matrix matrix_rates;
  if(matrix_rates.CopyRates(Symbol(), PERIOD_CURRENT, COPY_RATES_OHLCT, 1, 10))
    Print("matrix rates: \n", matrix_rates);
  else
    Print("matrix_rates.CopyRates failed. Error ", GetLastError());
//--- 確認する
  MqlRates mql_rates[];
  if(CopyRates(Symbol(), PERIOD_CURRENT, 1, 10, mql_rates)>0)
   {
    Print("mql_rates array:");
    ArrayPrint(mql_rates);
   }
  else
    Print("CopyRates(Symbol(), PERIOD_CURRENT,1, 10, mql_rates). Error ",
GetLastError());
//--- クォートをベクトルに取得する = 無効な呼び出し
  vector vector_rates;
  if(vector_rates.CopyRates(Symbol(), PERIOD_CURRENT, COPY_RATES_OHLC, 1, 15))
    Print("vector_rates COPY_RATES_OHLC: \n", vector_rates);
  else
    Print("vector_rates.CopyRates COPY_RATES_OHLC failed. Error ", GetLastError());
//--- 終値をベクトルに取得する
  if(vector_rates.CopyRates(Symbol(), PERIOD_CURRENT, COPY_RATES_CLOSE, 1, 15))
    Print("vector_rates COPY_RATES_CLOSE: \n", vector_rates);
  else
    Print("vector_rates.CopyRates failed. Error ", GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
 };
/*
   matrix rates:
   [[
0
.99686
,0.99638,0.99588,0.99441,0.99464,0.99594,0.99698,0.99758,0.99581,0.9952800000000001]
    [
0
.99708
,0.99643,0.99591,0.9955000000000001,0.99652,0.99795,0.99865,0.99764,0.99604,0.9957]
    [
0
.9961100000000001
,0.99491,0.99426,0.99441,0.99448,0.99494,0.9964499999999999,0.99472,0.9936,0.9922]
    [
0
.99641
,0.99588,0.99441,0.99464,0.99594,0.99697,0.99758,0.99581,0.9952800000000001,0.99259]
    [1662436800
,1662440400
,1662444000
,1662447600,1662451200,1662454800,1662458400,1662462000,1662465600,1662469200]]
   mql_rates array:
                    [time]  [open]  [high]   [low] [close] [tick_volume] [spread]
[real_volume]
   [0] 2022.09.06 04:00:00 0.99686 0.99708 0.99611 0.99641          4463        0
         0
   [1] 2022.09.06 05:00:00 0.99638 0.99643 0.99491 0.99588          4519        0
         0
   [2] 2022.09.06 06:00:00 0.99588 0.99591 0.99426 0.99441          3060        0
         0
   [3] 2022.09.06 07:00:00 0.99441 0.99550 0.99441 0.99464          3867        0
         0
   [4] 2022.09.06 08:00:00 0.99464 0.99652 0.99448 0.99594          5280        0
         0
   [5] 2022.09.06 09:00:00 0.99594 0.99795 0.99494 0.99697          7227        0
         0
   [6] 2022.09.06 10:00:00 0.99698 0.99865 0.99645 0.99758         10130        0
         0
   [7] 2022.09.06 11:00:00 0.99758 0.99764 0.99472 0.99581          7012        0
         0
   [8] 2022.09.06 12:00:00 0.99581 0.99604 0.99360 0.99528          6166        0
         0
   [9] 2022.09.06 13:00:00 0.99528 0.99570 0.99220 0.99259          6950        0
         0
   vector_rates.CopyRates COPY_RATES_OHLC failed. Error 4003
   vector_rates COPY_RATES_CLOSE:
```

© 2000–2025, MetaQuotes Ltd.

```
   [
0
.9931
,
0
.99293
,
0
.99417
,
0
.99504
,
0
.9968399999999999
,
0
.99641
,0.99588,0.99441,0.99464,0.99594,0.99697,0.99758,0.99581,0.9952800000000001,0.99259]
*/
```

参照

時系列と指標へのアクセス、CopyRates

© 2000–2025, MetaQuotes Ltd.

CopyTicks

MqlTick構造体からティックを取得して行列/ベクトルにします。要素は過去から現在までカウントされます。つまり、 インデックス0のティックが最も古いティックです。ティックの分析のためには、変更内容を正確に示すflagsフィールドを確認します。

```
bool  matrix::CopyTicks(
   string            symbol,                 // 銘柄名
   ulong             flags,                  // 受け取るティックの種類を定義するフラグ
   ulong             from_msc,               // ティック要求の開始時刻
   ulong             count                   // 受信するティックの数
   );
```

ベクトルメソッド

```
bool  vector::CopyTicks(
   string            symbol,                 // 銘柄名
   ulong             flags,                  // 受け取るティックの種類を定義するフラグ
   ulong             from_msc,               // ティック要求の開始時刻
   ulong             count                   // 受信するティックの数
   );
```

パラメータ

symbol

[in] 銘柄

flags

[in] 要求されたデータの内容を示すENUM_COPY_TICKS列挙からのフラグの組み合わせ。ベクトルにコピーする場合、ENUM_COPY_TICKS 列挙から1つの値のみを指定できます。そうしないと、エラーが発生します。

from_msc

[in] ティック要求範囲の開始時刻。時刻は、1970年1月1日からのミリ秒単位で指定されます。from_msc=0の場合、countに等しい最後のティック数が返されます。

count

[in] 要求されたティックの数。from_msc及びcountパラメータが指定されない場合、2000個を超えない全てのティックが書き込まれます。

戻り値

成功の場合はtrue、エラーが発生した場合はfalse

注意事項

CopyTicks()の最初の呼び出しでは、ハードドライブに保存されている関連する銘柄のティックデータベースの同期を開始します。ローカルデータベースが要求されたすべてのティックを提供しない場合、不足しているティックは取引サーバから自動的にダウンロードされます。CopyTicks()で指定した***from_msc***から現在までのティックが同期されます。その後、この銘柄に到着するすべてのティックがティックデータベースに追加され、同期状態が維持されます。

© 2000-2025, MetaQuotes Ltd.

*from_msc*及び*count*パラメータが指定されない場合、2000個を超えない全てのティックが行列/ベクトルに書き込まれます。

指標では、CopyTicks()メソッドはすぐに結果を返します。指標から呼び出されると、CopyTick()は銘柄の利用可能なすべてのティックをすぐに返し、利用可能なデータが不十分な場合はティックデータベースの同期を開始します。 同じ銘柄のすべての指標は1つの共通スレッドで動作するため、指標は同期の完了を待つことができません。同期後、CopyTicks()は、次の呼び出し時に要求されたすべてのティックを返します。指標では、各ティックの到着後にOnCalculate()関数が呼び出されます。

エキスパートアドバイザーとスクリプトでは、CopyTicks()は最大45秒間結果を待機できます。指標とは異なり、すべてのエキスパートアドバイザーまたはスクリプトは個別のスレッドで動作するため、最大で同期が完了するまで45秒待機できます。この間に必要な量のティックが同期されなかった場合、CopyTicks()はタイムアウトまでに利用可能なティックを返し、同期を続行します。エキスパートアドバイザーのOnTick()はすべてのティックのハンドラではなく、エキスパートアドバイザーに市場の変化を通知するだけです。これは一括変更である可能性があります。端末は複数のティックを同時に受信できますが、OnTick()は1回だけ呼び出され、エキスパートアドバイザーに最新の市場状態を通知します。

データリターン率：端末は、高速アクセスキャッシュ内の各金融商品の直近の4096ティックを保存します（市場深度が実行されている銘柄の場合は65536 ティック）。このデータに関する要求は最速で実行されます。現在の取引セッションで要求されたティックがキャッシュを超えている場合、CopyTicks()は端末メモリに保存されているティックを呼び出します。これらの要求は、完了するまでにより多くの時間を必要とします。最も低速の要求は他の日のティックに対するするものです。この場合、データはドライブから読み取られます。

ENUM_COPY_TICKS

ENUM_COPY_TICKS列挙には、行列または配列に渡されるデータの型を指定するフラグが含まれています。フラグの組み合わせにより、1回の要求で履歴から複数のシリーズを取得できます。行列の行の順序は、ENUM_COPY_TICKS列挙の値の順序に対応します。言い換えれば、行列内でHighデータを含む行は常にLowデータを含む行よりも高くなります。

ID	値	説明
COPY_TICKS_INFO	1	買い気配値および/または売り気配値の変化を含むティック
COPY_TICKS_TRADE	2	Lastおよび/またはVolume価格の変化を含むティック
COPY_TICKS_ALL	3	すべてのティック
COPY_TICKS_TIME_MS	1<<8	ミリ秒単位でのティック時刻
COPY_TICKS_BID	1<<9	Bid値
COPY_TICKS_ASK	1<<10	Ask値
COPY_TICKS_LAST	1<<11	Last値（最終約定値）
COPY_TICKS_VOLUME	1<<12	Last値の数量
COPY_TICKS_FLAGS	1<<13	ティックのフラグ
データ配置		
COPY_TICKS_VERTICAL	32768	ティックは縦軸に沿って行列にコピーされます。受信したティックは行列内で垂直に配置され

© 2000–2025, MetaQuotes Ltd.

ID	値	説明
		ます。つまり、最も古いティックが行列最初の行に配置され、最新のティックが最後の行に配置されます。 デフォルトのコピーでは、ティックが横軸に沿って行列に追加されます。 このフラグは、行列にコピーする場合にのみ適用されます。

ティックフラグを分析して、どのデータが変更されたかを調べます。

- TICK_FLAG_BID – Bid値が変更
- TICK_FLAG_ASK – Ask値が変更
- TICK_FLAG_LAST – 直近約定値が変更
- TICK_FLAG_VOLUME – 取引量が変更
- TICK_FLAG_BUY – 買い取引の結果
- TICK_FLAG_SELL – 売り取引の結果

参照

時系列と指標へのアクセス、CopyTicks

© 2000–2025, MetaQuotes Ltd.

CopyTicksRange

MqlTick構造体から指定された日付範囲内のティックを取得して行列/ベクトルにします。要素は過去から現在までカウントされます。つまり、　　インデックス0のティックが最も古いティックです。ティックの分析のためには、変更内容を正確に示すflagsフィールドを確認します。

```
bool  matrix::CopyTicksRange(
   string            symbol,                 // 銘柄名
   ulong             flags,                  // 受け取るティックの種類を定義するフラグ
   ulong             from_msc,               // ティック要求の開始時刻
   ulong             to_msc                  // ティック要求の終了時刻
   );
```

ベクトルメソッド

```
bool  vector::CopyTicksRange(
   string            symbol,                 // 銘柄名
   ulong             flags,                  // 受け取るティックの種類を定義するフラグ
   ulong             from_msc,               // ティック要求の開始時刻
   ulong             to_msc                  // ティック要求の終了時刻
   );
```

パラメータ

symbol

[in] 銘柄

flags

[in] 要求されたデータの内容を示すENUM_COPY_TICKS列挙からのフラグの組み合わせ。ベクトルにコピーする場合、ENUM_COPY_TICKS　列挙から1つの値のみを指定できます。そうしないと、エラーが発生します。

from_msc

[in]　　ティック要求範囲の開始時刻。時刻は、1970年1月1日からのミリ秒単位で指定されます。from_mscパラメータが指定されていない場合、履歴の先頭からのティックが送信されます。time　　>= from_mscを持つティックが返されます。

to_msc

[in]　　ティックが要求される最後の時刻。時刻は、1970年1月1日からのミリ秒単位で指定されます。time　<=　to_mscを持つティックが返されます。to_mscパラメータが指定されていない場合、履歴の最後までのティックが送信されます。

戻り値

成功の場合は　true、エラーが発生した場合は　falseGetLastError()は下記のエラーを返す場合があります。

- ERR_HISTORY_TIMEOUT — ティック同期のタイムアウトが時間切れになり、関数が持っているものだけが返された
- ERR_HISTORY_SMALL_BUFFER — 静的バッファが小さすぎるため、配列が格納できる量だけが返された
- ERR_NOT_ENOUGH_MEMORY — 指定された範囲から動的ティック配列に履歴を受け取るためのメモリが不十分で、ティック配列に十分なメモリを割り当てられなかった

© 2000-2025, MetaQuotes Ltd.

ティックフラグを分析して、どのデータが変更されたかを調べます。

- TICK_FLAG_BID – Bid値が変更
- TICK_FLAG_ASK – Ask値が変更
- TICK_FLAG_LAST – 直近約定値が変更
- TICK_FLAG_VOLUME – 取引量が変更
- TICK_FLAG_BUY – 買い取引の結果
- TICK_FLAG_SELL – 売り取引の結果

注意事項

CopyTicksRange()メソッドは、正確に指定された範囲からのティックを要求するために使用されます。例は、履歴の特定の日のティックです。CopyTicks()では、開始日のみを指定して、月初から現在までのすべてのティックを受け取ることなどができます。

参照

時系列と指標へのアクセス、CopyTicksRange

© 2000–2025, MetaQuotes Ltd.

Eye

統計関数。指定されたサイズの、主対角が1でそれ以外が0の行列を作成します。対角に1、それ以外に0を含む行列を返します。

```
static matrix matrix::Eye(
  const ulong  rows,         // 行数
  const ulong  cols,         // 列数
  const int    ndiag=0       // 対角線の指数
   );
```

パラメータ

rows

[in] 出力の行数

cols

[in] 出力の列数

ndiag=0

[in] 対角線のインデックス。0（デフォルト）は主対角線を表し、正の値は上対角線を表し、負の値は下対角線を表します。

戻り値

値が1に等しいk番目の対角要素を除いて、すべての要素が0に等しい行列

MQL5の例

```
  matrix eye=matrix::Eye(3, 3);
  Print("eye = \n", eye);

  eye=matrix::Eye(4, 4,1);
  Print("eye = \n", eye);
  /*
   eye =
   [[1,0,0]
    [0,1,0]
    [0,0,1]]
   eye =
   [[0,1,0,0]
    [0,0,1,0]
    [0,0,0,1]
    [0,0,0,0]]
  */
```

Pythonの例

```
np.eye(3, dtype=int)
```

© 2000–2025, MetaQuotes Ltd.

```
array([[1, 0, 0],
       [0, 1, 0],
       [0, 0, 1]])


 np.eye(4, k=1)
array([[0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.],
       [0., 0., 0., 0.]])
```

© 2000–2025, MetaQuotes Ltd.

Identity

指定されたサイズ（必ずしも正方形ではない）の単位行列を作成する静的関数です。単位行列では、主対角に1が含まれ、その他の部分には0が含まれます。主対角は、[0,0]、[1,1]、[2,2] など、行インデックスと列インデックスが等しい行列要素で構成されます。新しい単位行列を作成します。

既存の行列を単位行列に変換するIdentityメソッドもあります。

```
static matrix matrix::Identity(
  const ulong  rows,         // 行数
  const ulong  cols,         // 列数
   );

void matrix::Identity();
```

パラメータ

rows

[in] n x n行列の行（と列）の数

戻り値

単位行列（主対角に 1 を持つ正方行列）

MQL5の例

```
  matrix identity=matrix::Identity(3,3);
  Print("identity = \n", identity);
/*
   identity =
   [[1,0,0]
    [0,1,0]
    [0,0,1]]
*/
  matrix identity2(3,5);
  identity2.Identity();
  Print("identity2 = \n", identity2);
/*
   identity2 =
   [[1,0,0,0,0]
    [0,1,0,0,0]
    [0,0,1,0,0]]
*/
```

Pythonの例

```
np.identity(3)
array([[1.,  0.,  0.],
       [0.,  1.,  0.],
```

© 2000–2025, MetaQuotes Ltd.

```
        [0.,  0.,  1.]])
```

© 2000–2025, MetaQuotes Ltd.

Ones

この静的関数は、1で埋められた新しい行列を作成して返します。

```
static matrix matrix::Ones(
  const ulong  rows,      // 行数
  const ulong  cols       // 列数
   );

static vector vector::Ones(
  const ulong  size,      // ベクトルサイズ
   );
```

パラメータ

rows

[in] 行数

cols

[in] 列数

戻り値

1で埋められた、指定された行と列の新しい行列

MQL5の例

```
  matrix ones=matrix::Ones(4, 4);
  Print("ones = \n", ones);
/*
ones =
   [[1,1,1,1]
    [1,1,1,1]
    [1,1,1,1]
    [1,1,1,1]]
*/
```

Pythonの例

```
np.ones((4, 1))
array([[1.],
       [1.]])
```

© 2000-2025, MetaQuotes Ltd.

Zeros

この静的関数は、0で埋められた新しい行列を作成して返します。

```
static matrix matrix::Zeros(
  const ulong  rows,      // 行数
  const ulong  cols       // 列数
   );

static vector vector::Zeros(
  const ulong  size,      // ベクトルサイズ
   );
```

パラメータ

rows

[in] 行数

cols

[in] 列数

戻り値

0で埋められた、指定された行と列の新しい行列

MQL5の例

```
  matrix zeros=matrix::Zeros(3, 4);
  Print("zeros = \n", zeros);
/*
zeros =
   [[0,0,0,0]
    [0,0,0,0]
    [0,0,0,0]]
*/
```

Pythonの例

```
np.zeros((2, 1))
array([[ 0.],
       [ 0.]])
```

© 2000–2025, MetaQuotes Ltd.

Full

この静的関数は、指定された値で埋められた新しい行列を作成して返します。

```
static matrix matrix::Full(
  const ulong   rows,       // 行数
  const ulong   cols,       // 列数
  const double  value       // 書き入れる値
   );

static vector vector::Full(
  const ulong   size,       // ベクトルサイズ
  const double  value       // 書き入れる値
   );
```

パラメータ

rows

[in] 行数

cols

[in] 列数

value

[in] すべての行列要素に書き入れる値

戻り値

指定された値で埋められた、指定された行と列の新しい行列を返します

MQL5の例

```
  matrix full=matrix::Full(3,4,10);
  Print("full = \n", full);
/*
full =
   [[10,10,10,10]
    [10,10,10,10]
    [10,10,10,10]]
*/
```

例

```
np.full((2, 2), 10)
array([[10, 10],
       [10, 10]])
```

© 2000–2025, MetaQuotes Ltd.

Tri

指定された対角線以下に1を持ち、それ以外が0の行列を作成する静的関数です。

```
static matrix matrix::Tri(
  const ulong rows,         // 行数
  const ulong cols,         // 列数
  const int   ndiag=0       // 対角線の数
   );
```

パラメータ

rows

[in] 配列の行数

cols

[in] 配列の列数

ndiag=0

[in] それ以下の行列が埋められる劣対角。k = 0は主対角で、k < 0はその下、k > 0は上です。デフォルトは0です。

戻り値

下三角行列が1で満たされ、その他の部分が0である行列

MQL5の例

```
   matrix matrix_a=matrix::Tri(3,4,1);
   Print("Tri(3,4,1)\n",matrix_a);
   matrix_a=matrix::Tri(4,3,-1);
   Print("Tri(4,3,-1)\n",matrix_a);

/*
   Tri(3,4,1)
   [[1,1,0,0]
    [1,1,1,0]
    [1,1,1,1]]
   Tri(4,3,-1)
   [[0,0,0]
    [1,0,0]
    [1,1,0]
    [1,1,1]]
*/
```

例

```
np.tri(3, 5, 2, dtype=int)
array([[1, 1, 1, 0, 0],
       [1, 1, 1, 1, 0],
       [1, 1, 1, 1, 1]])
```

© 2000–2025, MetaQuotes Ltd.

Init

行列/ベクトルを初期化します。

```
void matrix::Init(
  const ulong  rows,            // 行数
  const ulong  cols,            // 列数
  func_name    init_func=NULL,  // クラスのスコープまたは静的メソッドに配置されたinit関数
   ...         parameters
   );

void vector::Init(
  const ulong  size,            // vector size
  func_name    init_func=NULL,  // クラスのスコープまたは静的メソッドに配置されたinit関数
   ...         parameters
   );
```

パラメータ

rows

[in] 行数

cols

[in] 列数

func_name

[in] 初期化関数

...

[in] 初期化関数のパラメータ

戻り値

なし。

例

```
template<typename T>
void MatrixArange(matrix<T> &mat,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<mat.Rows(); i++)
     {
      for(ulong j=0; j<mat.Cols(); j++,value+=step)
         mat[i][j]=value;
     }
  }
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
  int size_m=3, size_k=4;
  matrix  m(size_m,size_k,MatrixArange,-2.,0.1); // 最初に、サイズがsize_m x size_kの初期化されていない行列が作成されてから、
  Print("matrix m \n",m);                        // 次に、初期化時にリストされたパラメータを使用して関数 MatrixArangeが呼び出される
  matrixf m_float(5,5,MatrixArange,-2.f,0.1f);  // float型の行列が作成された後、MatrixArange関数が呼び出される
  Print("matrix m_float \n",m_float);
  vector  v(size_k,VectorArange,-10.0);         // ベクトルが作成された後、1つのパラメータを持つVectorArangeが呼び出される。2番目のパラメータがデフォルト
  Print("vector v \n",v);
  /*
   matrix m
   [[-2,-1.9,-1.8,-1.7]
    [-1.6,-1.5,-1.399999999999999,-1.299999999999999]
    [-1.199999999999999,-1.099999999999999,-0.9999999999999992,-0.8999999999999992]]
   matrix m_float
   [[-2,-1.9,-1.8,-1.6999999,-1.5999999]
    [-1.4999999,-1.3999999,-1.2999998,-1.1999998,-1.0999998]
    [-0.99999976,-0.89999974,-0.79999971,-0.69999969,-0.59999967]
    [-0.49999967,-0.39999968,-0.29999968,-0.19999969,-0.099999689]
    [3.1292439e-07,0.10000031,0.20000032,0.30000031,0.4000003]]
   vector v
   [-10,-9,-8,-7]
  */
  }
```

© 2000-2025, MetaQuotes Ltd.

Fill

指定された値で既存の行列/ベクトルを埋めます。

```
void matrix::Fill(
  const double  value      // 書き入れる値
   );

void vector::Fill(
  const double  value      // 書き入れる値
   );
```

パラメータ

value

[in] すべての行列要素に書き入れる値

戻り値

なし。行列は指定された値で埋められます

例

```
   matrix matrix_a(2,2);
   matrix_a.Fill(10);
   Print("matrix_a\n",matrix_a);

/*
  matrix_a
  [[10,10]
   [10,10]]
*/
```

© 2000-2025, MetaQuotes Ltd.

行列とベクトルの操作

これらは、基本的な行列操作（塗りつぶし、コピー、行列の一部の取得、転置、分割、並べ替え）のメソッドです。

行列の行と列を操作するメソッドもいくつかあります。

関数	アクション
Transpose	行列の軸を反転または並べ替えて、修正された行列を返します
TriL	k番目の対角より上の要素をゼロにした行列のコピーを返します（下三角行列）
TriU	k番目の対角より下の要素をゼロにした行列のコピーを返します（上三角行列）
Diag	対角を抽出するか、対角行列を構築します
Row	行ベクトルを返します。指定された行にベクトルを書き込みます
Col	列ベクトルを返します。指定された列にベクトルを書き込みます
Copy	指定された行列/ベクトルのコピーを返します
Compare	指定された精度で2つの行列/ベクトルの要素を比較します
CompareByDigits	2つの行列/ベクトルの要素を有効数字まで比較します
Flat	2つではなく1つのインデックスを介して行列要素をアドレス指定できるようにします
Clip	行列/ベクトルの要素を有効な値の指定された範囲に制限します
Reshape	データを変更せずに行列の形状を変更します
Resize	形状とサイズが変更された新しい行列を返します
SwapRows	行列の行を入れ替えます
SwapCols	行列の列を入れ替えます
Split	行列を複数の部分行列に分割します
Hsplit	行列を複数の部分行列に水平に分割します（axis=0のSplitと同じ）
Vsplit	行列を複数の部分行列に垂直に分割します（axis=1のSplitと同じ）
ArgSort	行列/ベクトルを間接的に並べ替えます
Sort	行列/ベクトルをその場で並べ替えます

© 2000–2025, MetaQuotes Ltd.

HasNan

Return the number of NaN values in a matrix/vector.

```
ulong vector::HasNan();

ulong matrix::HasNan();
```

Return Value

The number of matrix/vector elements that contain a NaN value.

Note

When comparing the appropriate pair of elements having NaN values, the Compare and CompareByDigits methods consider these elements equal, while in case of a usual comparison of floating-point numbers NaN != NaN.

Example:

```
void OnStart(void)
  {
   double x=sqrt(-1);

   Print("single: ",x==x);

   vector<double> v1={x};
   vector<double> v2={x};

   Print("vector: ", v1.Compare(v2,0)==0);
  }

/* Result:

 single: false
 vector: true
*/
```

See also

MathClassify, Compare, CompareByDigits

© 2000–2025, MetaQuotes Ltd.

Transpose

行列の転置。行列の軸を反転または並べ替えて、修正された行列を返します。

```
matrix matrix::Transpose()
```

戻り値

転置行列

次は、MQL5での単純な行列転置アルゴリズムです。

```
matrix MatrixTranspose(const matrix& matrix_a)
  {
   matrix matrix_c(matrix_a.Cols(),matrix_a.Rows());

   for(ulong i=0; i<matrix_c.Rows(); i++)
      for(ulong j=0; j<matrix_c.Cols(); j++)
         matrix_c[i][j]=matrix_a[j][i];

   return(matrix_c);
  }
```

MQL5の例

```
  matrix a= {{0, 1, 2}, {3, 4, 5}};
  Print("matrix a \n", a);
  Print("a.Transpose() \n", a.Transpose());

  /*
  matrix a
  [[0,1,2]
   [3,4,5]]
  a.Transpose()
  [[0,3]
   [1,4]
   [2,5]]
  */
```

Pythonの例

```
import numpy as np

a = np.arange(6).reshape((2,3))
print("a \n",a)
print("np.transpose(a) \n",np.transpose(a))
```

© 2000–2025, MetaQuotes Ltd.

```
 [[0 1 2]
 [3 4 5]]
np.transpose(a)
 [[0 3]
 [1 4]
 [2 5]]
```

© 2000–2025, MetaQuotes Ltd.

TransposeConjugate

Transposing a complex matrix with conjugation. Reverse or permute the axes of a matrix by changing a sign of an imaginary part of a complex number, return the modified matrix.

```
matrix matrix::TransposeConjugate()
```

Return Value

Transposed complex conjugate matrix.

Simple algorithm of transposing a complex matrix with conjugation - method explanation

```
//--- Complex matrix transpose function with conjugation
matrix MatrixTranspose(const matrix& matrix_a) {
    //--- create a new matrix_c with dimensions inverse to matrix_a
    matrix matrix_c(matrix_a.Cols(), matrix_a.Rows());

    //--- go through all the rows of the new matrix
    for (ulong i = 0; i < matrix_c.Rows(); i++) {
        //--- go through all the columns of the new matrix
        for (ulong j = 0; j < matrix_c.Cols(); j++) {
            //--- transfer the real part of the element by transposing the index
            matrix_c[i][j].real = matrix_a[j][i].real;

            //--- transfer the imaginary part of the element by changing the sign
(conjugation)
            matrix_c[i][j].imag = -matrix_a[j][i].imag;
        }
    }

    //--- return the transposed matrix with conjugation
    return (matrix_c);
}
```

© 2000–2025, MetaQuotes Ltd.

TriL

k番目の対角より上の要素をゼロにした行列のコピーを返します（下三角行列）。

```
matrix matrix::Tril(
  const int      ndiag=0       // 対角のインデックス
   );
```

パラメータ

ndiag=0

[in] それより上の要素をゼロにする対角。ndiag = 0（デフォルト）は主対角で、ndiag < 0はその下、ndiag > 0はその上です。

戻り値

下三角行列が1で満たされ、その他の部分が0である行列

MQL5の例

```
   matrix a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   matrix b=a.TriL(-1);
   Print("matrix b \n",b);

  /*
  matrix_c
  [[0,0,0]
  [4,0,0]
  [7,8,0]
  [10,11,12]]
  */
```

Pythonの例

```
import numpy as np

a=np.tril([[1,2,3],[4,5,6],[7,8,9],[10,11,12]], -1)

[[ 0  0  0]
 [ 4  0  0]
 [ 7  8  0]
 [10 11 12]]
```

© 2000–2025, MetaQuotes Ltd.

TriU

k番目の対角より下の要素を0にした行列のコピーを返します（上三角行列）。

```
matrix matrix::Triu(
  const int      ndiag=0       // 対角のインデックス
   );
```

パラメータ

ndiag=0

[in] それより下の要素をゼロにする対角。ndiag = 0（デフォルト）は主対角で、ndiag < 0はその下、ndiag > 0はその上です。

MQL5の例

```
   matrix a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   matrix b=a.TriU(-1);
   Print("matrix b \n",b);

  /*
  matrix b
  [[1,2,3]
   [4,5,6]
   [0,8,9]
   [0,0,12]]
  */
```

Pythonの例

```
import numpy as np

a=np.triu([[1,2,3],[4,5,6],[7,8,9],[10,11,12]], -1)
print(a)

[[ 1  2  3]
 [ 4  5  6]
 [ 0  8  9]
 [ 0  0 12]]
```

© 2000–2025, MetaQuotes Ltd.

Diag

対角を抽出する、または対角行列を構築します。

```
vector matrix::Diag(
  const int      ndiag=0      // 対角の数
   );

void matrix::Diag(
  const vector   v,           // 対角ベクトル
  const int      ndiag=0      // 対角の数
   );
```

パラメータ

v

[in] 要素が対応する対角に含まれるベクトル(ndiag=0が主対角)

ndiag=0

[in] 指定された対角。デフォルトは0です。主対角より上の対角にはndiag>0を使用し、主対角より下の対角にはndiag<0を使用します。

注意事項

未割り当て行列(次元を持たない)に対角を設定できます。この場合、サイズのゼロ行列が、対角ベクトルのサイズに対応するサイズで作成され、その後、対応する対角にベクトル値が入力されます。対角が既存の行列に設定されている場合、行列の次元は変更されず、対角ベクトルの外側の行列要素の値は変更されません。

例

```
   vector v1={1,2,3};
   matrix m1;
   m1.Diag(v1);
   Print("m1\n",m1);
   matrix m2;
   m2.Diag(v1,-1);
   Print("m2\n",m2);
   matrix m3;
   m3.Diag(v1,1);
   Print("m3\n",m3);
   matrix m4=matrix::Full(4,5,9);
   m4.Diag(v1,1);
   Print("m4\n",m4);

   Print("diag -1 - ",m4.Diag(-1));
   Print("diag 0 - ",m4.Diag());
   Print("diag 1 - ",m4.Diag(1));
```

© 2000-2025, MetaQuotes Ltd.

```
/*

m1
[[1,0,0]
[0,2,0]
[0,0,3]]
m2
[[0,0,0]
[1,0,0]
[0,2,0]
[0,0,3]]
m3
[[0,1,0,0]
[0,0,2,0]
[0,0,0,3]]
m4
[[9,1,9,9,9]
[9,9,2,9,9]
[9,9,9,3,9]
[9,9,9,9,9]]
diag -1 - [9,9,9]
diag 0 - [9,9,9,9]
diag 1 - [1,2,3,9]
*/
```

© 2000–2025, MetaQuotes Ltd.

Row

行ベクトルを返します。指定された行にベクトルを書き込みます

```
vector matrix::Row(
  const ulong   nrow      // 行番号
   );

void matrix::Row(
  const vector  v,        // 行ベクトル
  const ulong   nrow      // 行番号
   );
```

パラメータ

nrow

[in] 行の番号

戻り値

ベクトル

注意事項

未割り当て行列（次元を持たない）に行を設定できます。この場合、ベクトルサイズ x 行の数 + 1のサイズのゼロ行列が作成され、その後、ベクトル要素の値が対応する列に入力されます。行が既存の行列に設定されている場合、行列の次元は変更されず、行ベクトルの外側の行列要素の値は変更されません。

例

```
   vector v1={1,2,3};
   matrix m1;
   m1.Row(v1,1);
   Print("m1\n",m1);
   matrix m2=matrix::Full(4,5,7);
   m2.Row(v1,2);
   Print("m2\n",m2);

   Print("row 1 - ",m2.Row(1));
   Print("row 2 - ",m2.Row(2));

  /*
  m1
  [[0,0,0]
  [1,2,3]]
  m2
  [[7,7,7,7,7]
```

© 2000-2025, MetaQuotes Ltd.

```
   [7,7,7,7,7]
   [1,2,3,7,7]
   [7,7,7,7,7]]
   row 1 - [7,7,7,7,7]
   row 2 - [1,2,3,7,7]
   */
```

© 2000–2025, MetaQuotes Ltd.

Col

列ベクトルを返します。指定された列にベクトルを書き込みます。

```
vector matrix::Col(
  const ulong   ncol      // 列番号
   );

void matrix::Col(
  const vector  v,        // 列ベクトル
  const ulong   ncol      // 列番号
   );
```

パラメータ

ncol

[in] 列の数

戻り値

ベクトル

注意事項

未割り当て行列（次元を持たない）に列を設定できます。この場合、ベクトルサイズ x 列の数 + 1のサイズのゼロ行列が作成され、その後、ベクトル要素の値が対応する列に入力されます。列が既存の行列に設定されている場合、行列の次元は変更されず、列ベクトルの外側の行列要素の値は変更されません。

例

```
   vector v1={1,2,3};
   matrix m1;
   m1.Col(v1,1);
   Print("m1\n",m1);
   matrix m2=matrix::Full(4,5,8);
   m2.Col(v1,2);
   Print("m2\n",m2);

   Print("col 1 - ",m2.Col(1));
   Print("col 2 - ",m2.Col(2));

  /*
  m1
  [[0,1]
  [0,2]
  [0,3]]
  m2
  [[8,8,1,8,8]
```

© 2000–2025, MetaQuotes Ltd.

```
   [8,8,2,8,8]
   [8,8,3,8,8]
   [8,8,8,8,8]]
   col 1 - [8,8,8,8]
   col 2 - [1,2,3,8]
   */
```

© 2000–2025, MetaQuotes Ltd.

Copy

指定された行列/ベクトルのコピーを作成します。

```
bool matrix::Copy(
  const matrix&   a      // コピーされた行列
   );
bool vector::Copy(
  const vector&   v      // コピーされたベクトル
   );
```

パラメータ

v

[in] コピーする行列/ベクトル

戻り値

成功の場合はtrue、それ以外の場合はfalse

MQL5の例

```
  matrix a=matrix::Eye(3, 4);
  matrix b;
  b.Copy(a);
  matrix c=a;
  Print("matrix b \n", b);
  Print("matrix_c \n", c);

  /*
  /*
  matrix b
  [[1,0,0,0]
  [0,1,0,0]
  [0,0,1,0]]
  matrix_c
  [[1,0,0,0]
  [0,1,0,0]
  [0,0,1,0]]
  */
  */
```

Pythonの例

```
import numpy as np

a = np.eye(3,4)
print('a \n',a)
b = a
```

© 2000-2025, MetaQuotes Ltd.

```
print('b \n',b)
c = np.copy(a)
print('c \n',c)


 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]]
b
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]]
c
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]]
```

© 2000–2025, MetaQuotes Ltd.

Compare

指定された精度で2つの行列/ベクトルの要素を比較します。

```
ulong vector::Compare(
  const vector& vec,          // 比較するベクトル
  const double  epsilon       // 精度
   );

ulong matrix::Compare(
  const matrix& mat,          // 比較する行列
  const double  epsilon       // 精度
   );
```

パラメータ

vector_b

[in] 比較するベクトル

epsilon

[in] 精度

戻り値

比較される行列/ベクトルの不一致要素の数（行列が等しい場合は0、そうでない場合は0より大きくなる）

注意事項

比較演算子「==」または「!=」は、正確な要素単位の比較を実行します。実数の正確な比較は用途が限られていることが知られているため、イプシロン比較法が追加されました。たとえば、1つの行列に1e-20から1e+20までの範囲の要素が含まれる場合があります。このような行列は、有効数字までの要素ごとの比較を使用して処理できます。

例

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4}};
   matrix matrix_i=matrix::Identity(3,3);
   matrix matrix_c=matrix_a.Inv();
   matrix matrix_check=matrix_a.MatMul(matrix_c);
   Print("matrix_check\n",matrix_check);

   ulong errors=matrix_check.Compare(matrix::Identity(3,3),1e-15);
   Print("errors=",errors);


  /*
```

© 2000-2025, MetaQuotes Ltd.

```
matrix_check
[[1,0,0]
[4.440892098500626e-16,1,8.881784197001252e-16]
[4.440892098500626e-16,2.220446049250313e-16,0.9999999999999996]]
errors=0

*/
```

© 2000-2025, MetaQuotes Ltd.

CompareByDigits

2つの行列/ベクトルの要素を有効数字の精度で比較します。

```
ulong vector::CompareByDigits(
  const vector& vec,           // 比較するベクトル
  const int     digits         // 有効数字の数
   );

ulong matrix::CompareByDigits(
  const matrix& mat,           // 比較する行列
  const int     digits         // 有効数字の数
   );
```

パラメータ

vector_b

[in] 比較するベクトル

digits

[in] 比較する有効数字の数

epsilon

[in] 比較の精度。2つの値の絶対値の差が指定された精度よりも小さい場合、それらは等しいと見なされます。

戻り値

比較される行列/ベクトルの不一致要素の数（行列が等しい場合は0、そうでない場合は0より大きくなる）

注意事項

比較演算子「==」または「!=」は、正確な要素単位の比較を実行します。実数の正確な比較は用途が限られていることが知られているため、イプシロン比較法が追加されました。たとえば、1つの行列に1e-20から1e+20までの範囲の要素が含まれる場合があります。このような行列は、有効数字までの要素ごとの比較を使用して処理できます。

例

```
   int    size_m=128;
   int    size_k=256;
   matrix matrix_a(size_m,size_k);
//--- 行列に書き込む
   double  value=0.0;
   for(int i=0; i<size_m; i++)
     {
      for(int j=0; j<size_k; j++)
        {
```

© 2000-2025, MetaQuotes Ltd.

```
         if(i==j)
            matrix_a[i][j]=1.0+i;
         else
           {
            value+=1.0;
            matrix_a[i][j]=value/1e+20;
           }
        }
     }
//--- あと1つの行列を取得する
   matrix matrix_c = matrix_a * -1;

   ulong errors_epsilon=matrix_a.Compare(matrix_c,1e-15);
   ulong errors_digits=matrix_a.CompareByDigits(matrix_c,15);

   printf("Compare matrix %d x %d  errors_epsilon=%I64u  errors_digits=%
I64u",size_m,size_k,errors_epsilon,errors_digits);


  /*
  Compare matrix 128 x 256  errors_epsilon=128  errors_digits=32768
  */
```

© 2000–2025, MetaQuotes Ltd.

CompareEqual

Perform an absolute comparison of two matrices by unfolding successive rows into one-dimensional vectors.

```
ulong vector::Compare(
  const vector& vec    // vector to compare
   );

ulong matrix::CompareEqual(
  const matrix& mat     // matrix to compare
   );
```

Parameters

vec

[in] Vector to compare.

mat

[in] Matrix to compare.

Method description

Let us have two matrices: matrix A the method is called for and matrix B, which is passed as a method parameter. The comparison is performed as follows:

1. Matrices are expanded into one-dimensional vectors by successive concatenation of rows.
2. Vectors are compared element by element until the first mismatched element.
3. Depending on the comparison results, one of the values described below is returned.

Return Value

-1 — if the matrix A element is less than the corresponding matrix B element.

0 — if all elements of A and B matrices are identical.

1 — if the matrix A element exceeds the corresponding matrix B element.

Note

NaN value elements are taken into account when comparing.

NaN value elements are considered equal if they are present in both matrices at corresponding positions.

The NaN sign is not taken into account in the comparison.

An element with a NaN value is considered to be less than any other numeric value.

Example

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
```

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   matrix matrix_a= {{10, 3, 2}, {1, 8, 12}, {6, 5, 4}};
   matrix matrix_i=matrix::Identity(3, 3);
   matrix matrix_c=matrix_a.Inv();
   matrix matrix_check=matrix_a.MatMul(matrix_c);
   Print("matrix_check\n", matrix_check);

   ulong errors=matrix_check.Compare(matrix::Identity(3, 3), 1e-15);
   Print("errors=", errors);
   /*
   matrix_check
   [[1,0,0]
   [4.440892098500626e-16,1,8.881784197001252e-16]
   [4.440892098500626e-16,2.220446049250313e-16,0.9999999999999996]]
   errors=0
   */

//---  absolute comparison of matrices
   matrix<double> A = matrix_a;  // Matrix A initialization
   matrix<double> B = matrix_c;  // Matrix B initialization
   int result = A.CompareEqual(B);
   switch(result)
     {
      case -1:
         Print("Matrix A is smaller than matrix B");
         break;
      case 0:
         Print("Matrices A and B are identical");
         break;
      case 1:
         Print("Matrix A is greater than matrix B");
         break;
      case -2:
         Print("Error! Matrix A is not initialized");
         break;
      case 2:
         Print("Error! Matrix B is not initialized");
         break;
      case -3:
         Print("Error! The size of matrix A is less than the size of matrix B");
         break;
      case 3:
         Print("Error! The size of matrix A is greater than the size of matrix B");
         break;
      default:
         Print("Error! Unknown error");
```

© 2000–2025, MetaQuotes Ltd.

```
            break;
        }
    }
```

© 2000–2025, MetaQuotes Ltd.

Flat

2つではなく1つのインデックスを介して行列要素をアドレス指定できます。

```
bool matrix::Flat(
  const ulong    index,      //
  const double   value       // 設定する値
   );

double matrix::Flat(
  const ulong    index,      //
   );
```

パラメータ

index

[in] フラットインデックス

value

[in] 指定されたインデックスで設定する値

戻り値

指定されたインデックスによる値

注意事項

行列mat(3,3)の場合、アクセスは次のように記述できます。

- 読み取り: x=mat.Flat(4)（x=mat[1][1]と同等）
- 書き込み: mat.Flat(5, 42)（mat[1][2]=42と同等）

例

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);
   ulong arg_max=matrix_a.ArgMax();
   Print("max_value=",matrix_a.Flat(arg_max));
   matrix_a.Flat(arg_max,0);
   arg_max=matrix_a.ArgMax();
   Print("max_value=",matrix_a.Flat(arg_max));


  /*
  matrix_a
  [[10,3,2]
   [1,8,12]
   [6,5,4]
   [7,11,9]]
```

© 2000-2025, MetaQuotes Ltd.

```
   max_value=12.0
   max_value=11.0
   */
```

© 2000–2025, MetaQuotes Ltd.

Clip

行列/ベクトルの要素を有効な値の指定された範囲に制限します。

```
bool matrix::Clip(
  const double  min_value,      // 最小値
  const double  max_value       // 最大値
   );
bool vector::Clip(
  const double  min_value,      // 最小値
  const double  max_value       // 最大値
   );
```

パラメータ

min_value

[in] 最小値

max_value

[in] 最大値

戻り値

成功の場合はtrue、それ以外の場合はfalse

注意事項

行列（またはベクトル）はその場で処理されます。コピーは作成されません。

例

```
   matrix matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   bool res=matrix_a.Clip(4,8);
   Print("matrix_a\n",matrix_a);

  /*
  matrix_a
  [[4,4,4]
   [4,5,6]
   [7,8,8]
   [8,8,8]]
  */
```

© 2000-2025, MetaQuotes Ltd.

Reshape

データを変更せずに行列の形状を変更します。

```
void  Reshape(
  const ulong  rows,      // 新しい行数
  const ulong  cols       // 新しい列数
   );
```

パラメータ

rows

[in] 新しい行数

cols

[in] 新しい列数

注意事項

行列はその場で処理されます。コピーは作成されません。任意のサイズを指定できます。つまり、rows_new*cols_new!=rows_old*cols_oldです。行列のバッファが増加すると、余分な値は定義されません。

例

```
   matrix matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};

   Print("matrix_a\n",matrix_a);
   matrix_a.Reshape(2,6);
   Print("Reshape(2,6)\n",matrix_a);
   matrix_a.Reshape(3,5);
   Print("Reshape(3,5)\n",matrix_a);
   matrix_a.Reshape(2,4);
   Print("Reshape(2,4)\n",matrix_a);

  /*
  matrix_a
  [[1,2,3]
   [4,5,6]
   [7,8,9]
   [10,11,12]]
  Reshape(2,6)
  [[1,2,3,4,5,6]
   [7,8,9,10,11,12]]
  Reshape(3,5)
  [[1,2,3,4,5]
   [6,7,8,9,10]
   [11,12,0,3,0]]
  Reshape(2,4)
  [[1,2,3,4]
```

© 2000–2025, MetaQuotes Ltd.

```
   [5,6,7,8]]
  */
```

© 2000–2025, MetaQuotes Ltd.

Resize

形状とサイズが変更された新しい行列を返します。

```
bool matrix::Resize(
  const ulong  rows,     //
  const ulong  cols,     // 新しい列数
  const ulong  reserve=0 // アイテムの予約
   );

bool vector::Resize(
  const ulong  size,     // 新しいサイズ
  const ulong  reserve=0 // アイテムの予約
   );
```

パラメータ

rows

[in] 新しい行数

cols

[in] 新しい列数

戻り値

成功の場合はtrue、それ以外の場合はfalse

注意事項

行列（またはベクトル）はその場で処理されます。コピーは作成されません。任意のサイズを指定できます。つまり、rows_new*cols_new!=rows_old*cols_oldです。Reshapeとは異なり、行列は行ごとに処理されます。列数を増やすと、余分な列の値は未定義になります。行数を増やすと、新しい行の要素の値は未定義になります。列数を減らすと、行列の各行が切り捨てられます。

例

```
   matrix matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   Print("matrix_a\n",matrix_a);
   matrix_a.Resize(2,6);
   Print("Ressize(2,6)\n",matrix_a);
   matrix_a.Resize(3,5);
   Print("Resize(3,5)\n",matrix_a);
   matrix_a.Resize(2,4);
   Print("Resize(2,4)\n",matrix_a);

   /*
   matrix_a
   [[1,2,3]
    [4,5,6]
```

© 2000–2025, MetaQuotes Ltd.

```
    [7,8,9]
    [10,11,12]]
  Ressize(2,6)
  [[1,2,3,4,5,6]
    [4,5,6,10,11,12]]
  Resize(3,5)
  [[1,2,3,4,5]
    [4,5,6,10,11]
    [11,12,3,8,8]]
  Resize(2,4)
  [[1,2,3,4]
    [4,5,6,10]]
  */
```

© 2000–2025, MetaQuotes Ltd.

Set

指定されたインデックスのベクトル要素の値を設定します。

```
bool vector::Set(
  ulong   index,      // 要素インデックス
  double  value       // 値
   );
```

パラメータ

index

[in] 値を設定する必要がある要素のインデックス

value

[in] 値

戻り値

成功の場合は true、それ以外の場合は false

注意事項

Setメソッドは、角括弧を使用した値の割り当て(*vector[index]=value*)と同等です。このメソッドは、このタイプの表記法が使用されている言語からのコードの転送を簡素化するために追加されました。以下の例は、指定されたインデックスによる値をベクトルに入力するための両方のオプションを示しています。

例：

```
void OnStart()
  {
//---
   vector v1(10, VectorAssignValues);
   Print("v1 = ", v1);

   vector v2(10, VectorSetValues);
   Print("v2 = ", v2);
  }
 /* 結果
  v1 = [1,2,4,8,16,32,64,128,256,512]
  v2 = [1,2,4,8,16,32,64,128,256,512]
  */
//+------------------------------------------------------------------+
//|  代入演算を通じてベクトルに数値の累乗を入力する |
//+------------------------------------------------------------------+
void VectorAssignValues(vector& v, double initial=1)
  {
   double value=initial;
   for(ulong k=0; k<v.Size(); k++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       v[k]=value;
       value*=2;
      }
   }
//+------------------------------------------------------------------------+
//|  Set メソッドを使用してベクトルに数値のべき乗を入力する                |
//+------------------------------------------------------------------------+
void VectorSetValues(vector& v, double initial=1)
   {
    double value=initial;
    for(ulong k=0; k<v.Size(); k++)
      {
       v.Set(k, value);
       value*=2;
      }
   }
```

© 2000-2025, MetaQuotes Ltd.

SwapRows

行列の行を入れ替えます。

```
bool matrix::SwapRows(
  const ulong  row1,      // 1番目の行のインデックス
  const ulong  row2       // 2番目の行のインデックス
   );
```

パラメータ

row1

[in] 1番目の行のインデックス

row2

[in] 2番目の行のインデックス

戻り値

成功の場合はtrue、それ以外の場合はfalse

例

```
   matrix matrix_a={{1,2,3,4},
                    {5,6,7,8},
                    {9,10,11,12},
                    {13,14,15,16}};
   matrix matrix_i=matrix::Identity(4,4);
   matrix matrix_a1=matrix_a;
   matrix_a1.SwapRows(0,3);
   Print("matrix_a1\n",matrix_a1);

   matrix matrix_p=matrix_i;
   matrix_p.SwapRows(0,3);
   matrix matrix_c1=matrix_p.MatMul(matrix_a);
   Print("matrix_c1\n",matrix_c1);

  /*
  matrix_a1
  [[13,14,15,16]
   [5,6,7,8]
   [9,10,11,12]
   [1,2,3,4]]
  matrix_c1
  [[13,14,15,16]
   [5,6,7,8]
   [9,10,11,12]
   [1,2,3,4]]
  */
```

© 2000–2025, MetaQuotes Ltd.

SwapCols

行列の列を入れ替えます。

```
bool matrix::SwapCols(
  const ulong  row1,      // 1番目の列のインデックス
  const ulong  row2       // 2番目の列のインデックス
   );
```

パラメータ

col1

[in] 1番目の列のインデックス

col2

[in] 2番目の列のインデックス

戻り値

成功の場合はtrue、それ以外の場合はfalse

例

```
   matrix matrix_a={{1,2,3,4},
                    {5,6,7,8},
                    {9,10,11,12},
                    {13,14,15,16}};
   matrix matrix_i=matrix::Identity(4,4);
   matrix matrix_a1=matrix_a;
   matrix_a1.SwapCols(0,3);
   Print("matrix_a1\n",matrix_a1);

   matrix matrix_p=matrix_i;
   matrix_p.SwapCols(0,3);
   matrix matrix_c1=matrix_a.MatMul(matrix_p);
   Print("matrix_c1\n",matrix_c1);

  /*
  matrix_a1
  [[4,2,3,1]
   [8,6,7,5]
   [12,10,11,9]
   [16,14,15,13]]
  matrix_c1
  [[4,2,3,1]
   [8,6,7,5]
   [12,10,11,9]
   [16,14,15,13]]
  */
```

© 2000–2025, MetaQuotes Ltd.

Split

行列を複数の部分行列に分割します。

```
bool matrix::Split(
  const ulong  parts,      // 部分行列の数
  const int    axis,       // 軸
  matrix&      splitted[]  // 結果の部分行列の配列
   );

void matrix::Split(
  const ulong& parts[],    // 部分行列のサイズ
  const int    axis,       // 軸
  matrix&      splitted[]  // 結果の部分行列の配列
   );
```

パラメータ

parts

[in] 行列を分割してできる部分行列の数

axis

[in] 軸。0 — 水平軸、1 — 垂直軸。

splitted

[out] 結果の部分行列の配列

戻り値

成功の場合はtrue、それ以外の場合はfalse

注意事項

部分行列の数が指定されている場合、同じサイズの部分行列が取得されます。つまり、行列のサイズ（0 – 行数、1 – 列数）はpartsで割り切れる必要があります。部分行列サイズの配列を使用して、異なるサイズの部分行列を取得できます。サイズ配列の要素は、行列全体が分割されるまで使用されます。サイズの配列が終了し、行列がまだ完全に分割されていない場合、分割されていない残りが最後の部分行列になります。

例

```
   matrix matrix_a={{ 1, 2, 3, 4, 5, 6},
                    { 7, 8, 9,10,11,12},
                    {13,14,15,16,17,18},
                    {19,20,21,22,23,24},
                    {25,26,27,28,29,30}};
   matrix splitted[];
   ulong  parts[]={2,2};
```

© 2000–2025, MetaQuotes Ltd.

```
 bool res=matrix_a.Split(2,0,splitted);
 Print(res,"  ",GetLastError());
 ResetLastError();
 for(uint i=0; i<splitted.Size(); i++)
    Print("splitted ",i,"\n",splitted[i]);

 res=matrix_a.Split(2,1,splitted);
 Print(res,"  ",GetLastError());
 for(uint i=0; i<splitted.Size(); i++)
    Print("splitted ",i,"\n",splitted[i]);

 res=matrix_a.Split(parts,0,splitted);
 Print(res,"  ",GetLastError());
 for(uint i=0; i<splitted.Size(); i++)
    Print("splitted ",i,"\n",splitted[i]);

/*
false  4003
true  0
splitted 0
[[1,2,3]
 [7,8,9]
 [13,14,15]
 [19,20,21]
 [25,26,27]]
splitted 1
[[4,5,6]
 [10,11,12]
 [16,17,18]
 [22,23,24]
 [28,29,30]]
true  0
splitted 0
[[1,2,3,4,5,6]
 [7,8,9,10,11,12]]
splitted 1
[[13,14,15,16,17,18]
 [19,20,21,22,23,24]]
splitted 2
[[25,26,27,28,29,30]]
*/
```

© 2000–2025, MetaQuotes Ltd.

Hsplit

行列を複数の部分行列に水平に分割します（axis=0のSplitと同じ）

```
bool matrix::Hsplit(
  const ulong  parts,      // 部分行列の数
  matrix&      splitted[]  // 結果の部分行列の配列
   );

void matrix::Hsplit(
  const ulong& parts[],    // 部分行列のサイズ
  matrix&      splitted[]  // 結果の部分行列の配列
   );
```

パラメータ

parts

[in] 行列を分割してできる部分行列の数

splitted

[out] 結果の部分行列の配列

戻り値

成功の場合はtrue、それ以外の場合はfalse

注意事項

部分行列の数が指定されている場合、同じサイズの部分行列が取得されます。つまり、行数はpartsで割り切れる必要があります。部分行列サイズの配列を使用して、異なるサイズの部分行列を取得できます。サイズ配列の要素は、行列全体が分割されるまで使用されます。サイズの配列が終了し、行列がまだ完全に分割されていない場合、分割されていない残りが最後の部分行列になります。

例

```
   matrix matrix_a={{ 1, 2, 3, 4, 5, 6},
                    { 7, 8, 9,10,11,12},
                    {13,14,15,16,17,18},
                    {19,20,21,22,23,24},
                    {25,26,27,28,29,30}};
   matrix splitted[];
   ulong  parts[]={2,4};

   bool res=matrix_a.Hsplit(2,splitted);
   Print(res,"  ",GetLastError());
   ResetLastError();
   for(uint i=0; i<splitted.Size(); i++)
      Print("splitted ",i,"\n",splitted[i]);
```

© 2000-2025, MetaQuotes Ltd.

```
   res=matrix_a.Hsplit(5,splitted);
   Print(res,"  ",GetLastError());
   for(uint i=0; i<splitted.Size(); i++)
      Print("splitted ",i,"\n",splitted[i]);

   res=matrix_a.Hsplit(parts,splitted);
   Print(res,"  ",GetLastError());
   for(uint i=0; i<splitted.Size(); i++)
      Print("splitted ",i,"\n",splitted[i]);


/*
false  4003
true  0
splitted 0
[[1,2,3,4,5,6]]
splitted 1
[[7,8,9,10,11,12]]
splitted 2
[[13,14,15,16,17,18]]
splitted 3
[[19,20,21,22,23,24]]
splitted 4
[[25,26,27,28,29,30]]
true  0
splitted 0
[[1,2,3,4,5,6]
[7,8,9,10,11,12]]
splitted 1
[[13,14,15,16,17,18]
[19,20,21,22,23,24]
[25,26,27,28,29,30]]
*/
```

© 2000–2025, MetaQuotes Ltd.

Vsplit

行列を複数の部分行列に垂直に分割します（axis=1のSplitと同じ）

```
bool matrix::Vsplit(
  const ulong  parts,      // 部分行列の数
  matrix&      splitted[]  // 結果の部分行列の配列
   );

void matrix::Vsplit(
  const ulong& parts[],    // 部分行列のサイズ
  matrix&      splitted[]  // 結果の部分行列の配列
   );
```

パラメータ

parts

[in] 行列を分割してできる部分行列の数

splitted

[out] 結果の部分行列の配列

戻り値

成功の場合はtrue、それ以外の場合はfalse

注意事項

部分行列の数が指定されている場合、同じサイズの部分行列が取得されます。つまり、列数はpartsで割り切れる必要があります。部分行列サイズの配列を使用して、異なるサイズの部分行列を取得できます。サイズ配列の要素は、行列全体が分割されるまで使用されます。サイズの配列が終了し、行列がまだ完全に分割されていない場合、分割されていない残りが最後の部分行列になります。

例

```
   matrix matrix_a={{ 1, 2, 3, 4, 5, 6},
                    { 7, 8, 9,10,11,12},
                    {13,14,15,16,17,18}};
   matrix splitted[];
   ulong  parts[]={2,3};

   matrix_a.Vsplit(2,splitted);
   for(uint i=0; i<splitted.Size(); i++)
      Print("splitted ",i,"\n",splitted[i]);

   matrix_a.Vsplit(3,splitted);
   for(uint i=0; i<splitted.Size(); i++)
      Print("splitted ",i,"\n",splitted[i]);

   matrix_a.Vsplit(parts,splitted);
```

© 2000-2025, MetaQuotes Ltd.

```
   for(uint i=0; i<splitted.Size(); i++)
      Print("splitted ",i,"\n",splitted[i]);


  /*
     splitted 0
     [[1,2,3]
      [7,8,9]
      [13,14,15]]
     splitted 1
     [[4,5,6]
      [10,11,12]
      [16,17,18]]

     splitted 0
     [[1,2]
      [7,8]
      [13,14]]
     splitted 1
     [[3,4]
      [9,10]
      [15,16]]
     splitted 2
     [[5,6]
      [11,12]
      [17,18]]

     splitted 0
     [[1,2]
      [7,8]
      [13,14]]
     splitted 1
     [[3,4,5]
      [9,10,11]
      [15,16,17]]
     splitted 2
     [[6]
      [12]
      [18]]

  */
```

© 2000–2025, MetaQuotes Ltd.

ArgSort

行列/ベクトルを間接的に並べ替えます。

```
vector vector::Sort(
  func_name  compare_func=NULL,  // 比較関数
  T          context             // カスタム並び替え関数のパラメータ
   );

matrix matrix::Sort(
  func_name  compare_func=NULL   // 比較関数
  T          context             // カスタム並び替え関数のパラメータ
   );

matrix matrix::Sort(
  const int  axis,               // 並べ替えの軸
  func_name  compare_func=NULL   // 比較関数
  T          context             // カスタム並び替え関数のパラメータ
   );
```

パラメータ

axis

[in] 並び替えの軸（0は水平方向、1は垂直方向）

func_name

[in] 比較関数。ENUM_SORT_MODE列挙の値の1つまたは独自の比較関数を指定できます。関数が指定されていない場合、昇順での並び替えが使用されます。

カスタム比較関数には次の2種類があります。

- int comparator(T x1,T x2)
- int comparator(T x1,T x2,TContext context)

ここで、Tは行列/ベクトルの型であり、TContextは、Sortメソッドに追加パラメータとして渡されるcontext変数の型です。

context

[in] カスタム並べ替え関数に渡すことができる追加のオプションパラメータ

戻り値

並び替えられた要素のインデックスを持つベクトルまたは行列。たとえば、結果 [4,2,0,1,3] は、位置0にインデックス4を持つ要素、位置1にインデックス2を持つ要素などがあることを示します。

© 2000-2025, MetaQuotes Ltd.

Sort

行列/ベクトルをその場で並べ替えます

```
void vector::Sort(
  func_name   compare_func=NULL,  // 比較関数
  T           context             // カスタム並び替え関数のパラメータ
   );

void matrix::Sort(
  func_name   compare_func=NULL   // 比較関数
  T           context             // カスタム並び替え関数のパラメータ
   );

void matrix::Sort(
  const int   axis,               // 並べ替えの軸
  func_name   compare_func=NULL   // 比較関数
  T           context             // カスタム並び替え関数のパラメータ
   );
```

パラメータ

axis

[in] 並び替えの軸（0は水平方向、1は垂直方向）

func_name

[in] 比較関数。ENUM_SORT_MODE列挙の値の1つまたは独自の比較関数を指定できます。関数が指定されていない場合、昇順での並び替えが使用されます。

カスタム比較関数には次の2種類があります。

- int comparator(T x1,T x2)
- int comparator(T x1,T x2,TContext context)

ここで、Tは行列/ベクトルの型であり、TContexは、Sortメソッドに追加パラメータとして渡されるcontext変数の型です。

context

[in] カスタム並べ替え関数に渡すことができる追加のオプションパラメータ

戻り値

なし。並び替えはその場で実行されます。つまり、並び替えメソッドが呼び出された行列/ベクトルのデータに適用されます。

例

```
//+------------------------------------------------------------------+
//|  並び替え関数                                                    |
//+------------------------------------------------------------------+
int MyDoubleComparator(double x1,double x2,int sort_mode=0)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   int res=x1<x2 ? -1 : (x1>x2 ? 1 : 0);
   return(sort_mode==0 ? res : -res);
  }
//+------------------------------------------------------------------+
//|  スクリプトを開始する関数                                          |
//+------------------------------------------------------------------+
void OnStart()
  {
   //--- ベクトルに書き入れる
   vector v(100);
   //--- 昇順で並べ替える
   v.Sort(MyDoubleComparator);   // an additional parameter with the default value '0' is used here
   Print(v);
   // 降順で並べ替える
   v.Sort(MyDoubleComparator,1); // ここでは、追加のパラメータ「1」がユーザーによって明示的に指定されている
   Print(v);
  }
```

© 2000–2025, MetaQuotes Ltd.

行列とベクトルによる数学演算

足し算、引き算、掛け算、割り算などの数学演算は、行列とベクトルに対して要素単位で実行できます。

数学関数は、もともとスカラー値に対して関連する操作を実行するために設計されました。ほとんどの関数は、行列とベクトルに適用できます。これには、MathAbs、MathArccos、MathArcsin、MathArctan、MathCeil、MathCos、MathExp、MathFloor、MathLog、MathLog10、MathMod、MathPow、MathRound、MathSin、MathSqrt、MathTan、MathExpm1、MathLog1p、MathArccosh、MathArcsinh、MathArctanh、MathCosh、MathSinh、MathTanhが含まれます。このような操作は、行列とベクトルの要素単位の処理を意味します。例

```
//---
  matrix a= {{1, 4}, {9, 16}};
  Print("matrix a=\n",a);
  a=MathSqrt(a);
  Print("MatrSqrt(a)=\n",a);
  /*
   matrix a=
   [[1,4]
    [9,16]]
   MatrSqrt(a)=
   [[1,2]
    [3,4]]
  */
```

MathModとMathPowの場合、2番目の要素は適切なサイズのスカラーまたは行列/ベクトルのいずれかです。

次の例は、数学関数をベクトルに適用して標準偏差を計算する方法を示しています。

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- ベクトルを埋めるために初期化関数を使用する
  vector r(10, ArrayRandom); // 0～1の乱数の配列
//--- 平均値を計算する
  double avr=r.Mean();       // 配列の平均値
  vector d=r-avr;            // 平均からの偏差の配列を計算する
  Print("avr(r)=", avr);
  Print("r=", r);
  Print("d=", d);
  vector s2=MathPow(d, 2);   // 二乗偏差の配列
  double sum=s2.Sum();       // 二乗偏差の総和
//--- 2つの異なる方法で標準偏差を計算する
  double std=MathSqrt(sum/r.Size());
  Print(" std(r)=", std);
  Print("r.Std()=", r.Std());
 }
```

© 2000-2025, MetaQuotes Ltd.

```
/*
  avr(r)=0.5300302133243813

r=[
0
.8346201971495713
,
0
.8031556138798182
,0.6696676534318063,0.05386516922513505,0.5491195410016175,0.8224433118686484,...
  d=[0.30458998382519,0.2731254005554369,0.1396374401074251,-
0.4761650440992462,0.01908932767723626,0.2924130985442671, ...
   std(r)=0.2838269732183663
  r.Std()=0.2838269732183663
*/
//+------------------------------------------------------------------+
//|  ベクトルにランダムな値を書き込む                                |
//+------------------------------------------------------------------+
void ArrayRandom(vector& v)
 {
  for(ulong i=0; i<v.Size(); i++)
    v[i]=double(MathRand())/32767.;
 }
```

© 2000–2025, MetaQuotes Ltd.

数学演算

足し算、引き算、掛け算、割り算などの数学演算は、行列とベクトルに対して要素単位で実行できます。

両方の行列または両方のベクトルは、同じ型で同じ次元でなければなりません。行列の各要素は、2番目の行列の対応する要素に作用します。

適切な型（double、float、complex）のスカラーを2番目の項（multiplier、subtrahend、または　divisor）として使用することもできます。この場合、行列/ベクトルの各メンバーは、指定されたスカラーで動作します。

```
   matrix matrix_a={{0.1,0.2,0.3},{0.4,0.5,0.6}};
   matrix matrix_b={{1,2,3},{4,5,6}};

   matrix matrix_c1=matrix_a+matrix_b;
   matrix matrix_c2=matrix_b-matrix_a;
   matrix matrix_c3=matrix_a*matrix_b;    // アダマール積です。行列積と混同しないでください! これには特別な MatMul 関数があります
   matrix matrix_c4=matrix_b/matrix_a;

   matrix_c1=matrix_a+1;
   matrix_c2=matrix_b-double_value;
   matrix_c3=matrix_a*M_PI;
   matrix_c4=matrix_b/0.1;

//--- operations in place are possible
   matrix_a+=matrix_b;
   matrix_a/=2;
```

ベクターでも同じ演算が可能です。

© 2000-2025, MetaQuotes Ltd.

数学関数

次の数学関数は、行列とベクトルに適用できます。MathAbs、MathArccos、MathArcsin、MathArctan、MathCeil、MathCos、MathExp、MathFloor、MathLog、MathLog10、MathMod、MathPow、MathRound、MathSin、MathSqrt、MathTan、MathExpm1、MathLog1p、MathArccosh、MathArcsinh、MathArctanh、MathCosh、MathSinh、MathTanh。このような操作は、行列とベクトルの要素単位の処理を意味します。

MathModとMathPowの場合、2番目の要素は適切なサイズのスカラーまたは行列/ベクトルのいずれかです。

```
   matrix<T> mat1(128,128);
   matrix<T> mat3(mat1.Rows(),mat1.Cols());
   ulong     n,size=mat1.Rows()*mat1.Cols();
...
   mat2=MathPow(mat1,(T)1.9);
   for(n=0; n<size; n++)
     {
      T res=MathPow(mat1.Flat(n),(T)1.9);
      if(res!=mat2.Flat(n))
         errors++;
     }
   mat2=MathPow(mat1,mat3);
   for(n=0; n<size; n++)
     {
      T res=MathPow(mat1.Flat(n),mat3.Flat(n));
      if(res!=mat2.Flat(n))
         errors++;
     }
...
   vector<T> vec1(16384);
   vector<T> vec3(vec1.Size());
   ulong     n,size=vec1.Size();
...
   vec2=MathPow(vec1,(T)1.9);
   for(n=0; n<size; n++)
     {
      T res=MathPow(vec1[n],(T)1.9);
      if(res!=vec2[n])
         errors++;
     }
   vec2=MathPow(vec1,vec3);
   for(n=0; n<size; n++)
     {
      T res=MathPow(vec1[n],vec3[n]);
      if(res!=vec2[n])
         errors++;
     }
```

© 2000-2025, MetaQuotes Ltd.

行列とベクトルの積

行列とベクトル積の計算には以下が含まれます。

- 行列乗算
- ベクトル乗算
- 共分散行列の計算
- 2つのベクトルの相互相関の計算
- 2つのベクトルの畳み込みの計算
- 相関係数の計算

関数	アクション
MatMul	2つの行列の行列積
GeMM	一般行列乗算(GeMM)
乗	正方行列を整数乗します
Dot	2つのベクトルの内積
Kron	2つの行列、行列とベクトル、ベクトルと行列、または2つのベクトルのクロネッカー積を返します
Inner	2つの行列の内積
Outer	2つの行列または2つのベクトルの外積を計算します
CorrCoef	ピアソン相関係数(線形相関係数)を計算します
Cov	共分散行列を計算します
Correlate	2つのベクトルの相互相関を計算します
Convolve	2つのベクトルの離散線形畳み込みを返します

© 2000–2025, MetaQuotes Ltd.

MatMul

行列とベクトルの乗算を可能にするMatMulメソッドには、いくつかのオーバーロードがあります。

Multiplying a matrix by a matrix: matrix[M][K] * matrix[K][N] = matrix[M][N]

```
matrix  matrix::MatMul(
  const matrix&  b      // 2番目の行列
   );
```

Multiplying a vector by a matrix: horizontal vector[K] * matrix[K][N] = horizontal vector[N]

```
vector  vector::MatMul(
  const matrix&  b      // 行列
   );
```

Multiplying a matrix by a vector: matrix[M][K] * vertical vector[K] = vertical vector[M]

```
vector  matrix::MatMul(
  const vector&  b      // ベクトル
   );
```

Scalar vector multiplication: horizontal vector * vertical vector = dot value

```
scalar  vector::MatMul(
  const vector&  b      // 2番目のベクトル
   );
```

パラメータ

b

[in] 行列/ベクトル

戻り値

使用するメソッドに応じて、行列、ベクトル、またはスカラー。

注意事項

行列は乗算に対応している必要があります。つまり、1番目の行列の列数は2番目の行列の行数と等しくなければなりません。行列の乗算は可換ではありません。一般に、1番目の行列に2番目の行列を掛けた結果は、2番目の行列に最初の行列を掛けた結果と等しくありません。

行列積は、最初の行列の行ベクトルと 2 番目の行列の列ベクトルのスカラー積のすべての可能な組み合わせで構成されます。

スカラー乗算では、ベクトルは同じ長さでなければなりません。

ベクトルと行列を乗算する場合、ベクトルの長さは行列の列数と正確に一致する必要があります。

次は、MQL5での単純な行列乗算アルゴリズムです。

© 2000–2025, MetaQuotes Ltd.

```
matrix MatrixProduct(const matrix& matrix_a, const matrix& matrix_b)
  {
   matrix matrix_c;

   if(matrix_a.Cols()!=matrix_b.Rows())
      return(matrix_c);

   ulong M=matrix_a.Rows();
   ulong K=matrix_a.Cols();
   ulong N=matrix_b.Cols();
   matrix_c=matrix::Zeros(M,N);

   for(ulong m=0; m<M; m++)
      for(ulong k=0; k<K; k++)
         for(ulong n=0; n<N; n++)
            matrix_c[m][n]+=matrix_a[m][k]*matrix_b[k][n];

   return(matrix_c);
  }
```

行列乗算例

```
   matrix a={{1, 0, 0},
             {0, 1, 0}};
   matrix b={{4, 1},
             {2, 2},
             {1, 3}};
   matrix c1=a.MatMul(b);
   matrix c2=b.MatMul(a);
   Print("c1 = \n", c1);
   Print("c2 = \n", c2);
/*
   c1 =
   [[4,1]
    [2,2]]
   c2 =
   [[4,1,0]
    [2,2,0]
    [1,3,0]]
*/
```

水平ベクトルに行列を乗算する例

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- 3x5行列を作成する
   matrix m35;
   m35.Init(3, 5, Arange);
//---
   vector v3 = {1, 2, 3};
   Print("Product of horizontal vector v and matrix m[3,5]");
   Print("On the left, vector v3 = ", v3);
   Print("On the right, matrix m35 = \n", m35);
   Print("v3.MatMul(m35) = horizontal vector v[5] \n", v3.MatMul(m35));

  /* 結果
    Product of horizontal vector v3 and matrix m[3,5]
    On the left, vector v3 = [1,2,3]
    On the right, matrix m35 =
    [[0,1,2,3,4]
     [5,6,7,8,9]
     [10,11,12,13,14]]
    v3.MatMul(m35) = horizontal vector v[5]
    [40,46,52,58,64]
   */
  }
//+------------------------------------------------------------------+
//|   増加する値で行列を埋める                                       |
//+------------------------------------------------------------------+
void Arange(matrix & m, double start = 0, double step = 1)
  {
//---
   ulong cols = m.Cols();
   ulong rows = m.Rows();
   double value = start;
   for(ulong r = 0; r < rows; r++)
     {
      for(ulong c = 0; c < cols; c++)
        {
         m[r][c] = value;
         value += step;
        }
     }
//---
  }
```

An example of how to multiply a matrix by a vertical vector

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
```

© 2000–2025, MetaQuotes Ltd.

```
  {
//--- 3x5行列を作成する
   matrix m35;
   m35.Init(3, 5, Arange);
//---
   Print("Product of matrix m[3,5] and vertical vector v[5]");
   vector v5 = {1,2,3,4,5};
   Print("On the left, m35 = \n",m35);
   Print("On the right v5 = ",v5);
   Print("m35.MatMul(v5) = vertical vector v[3] \n",m35.MatMul(v5));

  /* 結果
   Product of matrix m[3,5] and vertical vector v[5]
   On the left, m35 =
   [[0,1,2,3,4]
    [5,6,7,8,9]
    [10,11,12,13,14]]
   On the right, v5 = [1,2,3,4,5]
   m35.MatMul(v5) = vertical vector v[3]
   [40,115,190]
   */
  }
//+------------------------------------------------------------------+
//|   増加する値で行列を埋める                                       |
//+------------------------------------------------------------------+
void Arange(matrix & m, double start = 0, double step = 1)
  {
//---
   ulong cols = m.Cols();
   ulong rows = m.Rows();
   double value = start;
   for(ulong r = 0; r < rows; r++)
     {
      for(ulong c = 0; c < cols; c++)
        {
         m[r][c] = value;
         value += step;
        }
     }
//---
  }
```

ベクトルのスカラー（ドット）積の例

```
void OnStart()
  {
//--- 水平ベクトルと垂直ベクトルのスカラー積
   vector a= {1, 2, 3};  // 水平ベクトル
```

© 2000-2025, MetaQuotes Ltd.

```
   vector b= {4, 5, 6};  // 垂直ベクトル
   Print("a = ", a);
   Print("b = ", b);
   Print("1) a.MatMul(b) = ", a.MatMul(b));
   //--- Dot メソッドが同じ結果を生成することを確認する
   Print("2) a.Dot(b) = ", a.Dot(b));

  /* 結果
   a = [1,2,3]
   b = [4,5,6]
   1) a.MatMul(b) = 32.0
   2) a.Dot(b) = 32.0
    */
  }
```

参照

Dot、GeMM

© 2000-2025, MetaQuotes Ltd.

GeMM

GeMM (General Matrix Multiply)メソッドは、2つの行列の一般的な乗算を実装します。演算はC ← αAB + βCとして定義されます。ここで、A行列とB行列はオプションで転置できます。行列ABの通常の乗算(MatMul)では、*alpha*スカラーは1に等しいと想定され、ベータはゼロに等しくなります。

効率におけるGeMMとMatMulの主な違いは、MatMulは常に新しい行列/ベクトルオブジェクトを作成するのに対し、GeMMは既存の行列オブジェクトで動作し、再作成しないことです。したがって、GeMMを使用して対応する行列にメモリを事前に割り当てると、同じ行列サイズで作業している間、メモリの再割り当ては行われません。これは、たとえば、ストラテジーテスターで最適化を実行する場合やニューラルネットワークをトレーニングする場合など、バルクコンピューティングにおけるGeMMの重要な利点となります。

MatMulと同様に、GeMMにも4つのオーバーロードがあります。ただし、4番目のオーバーロードのセマンティクスは、垂直ベクトルと水平ベクトルの乗算を可能にするために変更されています。

既存の行列/ベクトルオブジェクトでは、メモリを事前に割り当てる必要はありません。メモリは、最初のGeMM呼び出しで割り当てられ、ゼロで埋められます。

行列と行列の乗算： `matrix C[M][N] = α * ( matrix A[M][K] * matrix B[K][N]) + β * matrix C[M][N]`

```
bool  matrix::GeMM(
  const matrix &A,     // 1番目の行列
  const matrix &B,     // 2番目の行列
  double alpha,        // 積ABのアルファ乗数
  double beta,         // 行列Cのベータ乗数
  uint   flags         // 行列A、B、Cが転置されるかどうかを決定するENUM_GEMM値の組み合わせ(ビットOR)
   );
```

ベクトルと行列の乗算： `vector C[N] = α * ( vector A[K] * matrix B[K][N]) + β * vector C[N]`

```
bool  vector::GeMM(
  const vector &A,     // 水平ベクトル
  const matrix &B,     // 行列
  double alpha,        // 積ABのアルファ乗数
  double beta,         // ベクトルCのベータ乗数
  uint   flags         // 行列Aが転置されるかどうかを決定するENUM_GEMM列挙値
   );
```

行列とベクトルの乗算： `vector C[M] = α * ( matrix A[M][K] * vector B[K] * ) + β * vector C[M]`

```
bool  vector::GeMM(
  const matrix &A,     // 行列
  const vector &B,     // 垂直ベクトル
  double alpha,        // 積ABのアルファ乗数
  double beta,         // ベクトルCのベータ乗数
  uint   flags         // 行列Bが転置されるかどうかを決定するENUM_GEMM列挙値
   );
```

© 2000-2025, MetaQuotes Ltd.

ベクトルとベクトルの乗算： matrix C[M][N] = α * (vector A[M] * vector B[N] *) + β * matrix C[M][N]. このオーバーロードは、スカラーを返すMatMulとは異なり、行列を返します。

```
bool  matrix::GeMM(
  const vector &A,     // 1番目のベクトル
  const vector &B,     // 2番目のベクトル
  double alpha,        // 積ABのアルファ乗数
  double beta,         // 行列Cのベータ
  uint   flags         // 行列Cが転置されるかどうかを決定するENUM_GEMM列挙値
   );
```

パラメータ

A

[in] 行列/ベクトル

B

[in] 行列/ベクトル

alpha

[in] AB積のアルファ乗数

beta

[in] 結果のC行列のベータ乗数

フラグ

[in] 行列A、B、Cが転置されるかどうかを決定するENUM_GEMM列挙値

戻り値

成功の場合はtrue、それ以外の場合はfalse

ENUM_GEMM

GeMMメソッドのフラグの列挙

ID	説明
TRANSP_A	転置行列Aを使用
TRANSP_B	転置行列Bを使用
TRANSP_C	転置行列Cを使用

注意事項

float、double、complex型の行列とベクトルは、パラメータAとBとして使用できます。GeMMメソッドのテンプレートバリアントは次のとおりです。

```
bool matrix<T>::GeMM(const matrix<T> &A,const matrix<T> &B,T alpha,T beta,ulong flags);
```

© 2000-2025, MetaQuotes Ltd.

```
bool matrix<T>::GeMM(const vector<T> &A,const vector<T> &B,T alpha,T beta,ulong
flags);

bool vector<T>::GeMM(const vector<T> &A,const matrix<T> &B,T alpha,T beta,ulong
flags);
bool vector<T>::GeMM(const matrix<T> &A,const vector<T> &B,T alpha,T beta,ulong
flags);
```

基本的に、一般的な行列乗算関数は次のように記述されます。

```
C[m,n] = α *Sum(A[m,k]*B[k,n]) + β*C[m,n]
```

行列AはM x K、行列BはK x N、行列CはM x Nです。

したがって、行列は乗算に対応している必要があります。つまり、1番目の行列の列数は2番目の行列の行数と等しくなければなりません。行列の乗算は可換ではありません。一般に、1番目の行列に2番目の行列を掛けた結果は、2番目の行列に最初の行列を掛けた結果と等しくありません。

例：

```
void OnStart()
  {
   vector vector_a= {1, 2, 3, 4, 5};
   vector vector_b= {4, 3, 2, 1};
   matrix matrix_c;
//--- 2つのベクトルのGeMMを計算する
   matrix_c.GeMM(vector_a, vector_b, 1, 0);
   Print("matrix_c:\n ", matrix_c, "\n");
   /*
   matrix_c:
    [[4,3,2,1]
    [8,6,4,2]
    [12,9,6,3]
    [16,12,8,4]
    [20,15,10,5]]
   */
//--- 行列をベクトルとして作成する
   matrix matrix_a(5, 1);
   matrix matrix_b(1, 4);
   matrix_a.Col(vector_a, 0);
   matrix_b.Row(vector_b, 0);
   Print("matrix_a:\n ", matrix_a);
   Print("matrix_b:\n ", matrix_b);
   /*
   matrix_a:
   [[1]
   [2]
   [3]
```

© 2000–2025, MetaQuotes Ltd.

```
    [4]
    [5]]
    matrix_b:
    [[4,3,2,1]]
    */
//-- 2つの行列のGeMMを計算し、同じ結果を得る
    matrix_c.GeMM(matrix_a, matrix_b, 1, 0);
    Print("matrix_c:\n ", matrix_c);
    /*
    matrix_c:
     [[4,3,2,1]
     [8,6,4,2]
     [12,9,6,3]
     [16,12,8,4]
     [20,15,10,5]]
    */
   }
```

参照

MatMul

© 2000-2025, MetaQuotes Ltd.

Power

正方行列を整数乗します。

```
matrix matrix::Power(
  const int  power      // 乗
   );
```

パラメータ

power

[in] 指数は、任意の整数、正、負、またはゼロにすることができます。

戻り値

行列

注意事項

結果の行列は、元の行列と同じサイズになります。0のべき乗では、単位行列が返されます。正のべき乗nは、元の行列がn倍に乗算されることを意味します。負のべき乗-nは、元の行列が最初に反転され、次に反転された行列がn回乗算されることを意味します。

次は、MQL5で行列を累乗する単純なアルゴリズムです。

```
bool MatrixPower(matrix& c, const matrix& a, const int power)
  {
//--- 正方行列が必要
   if(a.Rows()!=a.Cols())
      return(false);
//--- 結果の行列のサイズはまったく同じ
   ulong  rows=a.Rows();
   ulong  cols=a.Cols();
   matrix result(rows,cols);
//--- ゼロにべき乗された場合は単位行列を返す
   if(power==0)
      result.Identity();
   else
     {
      //--- 負の次数の場合、最初に行列を反転する
      if(power<0)
        {
         matrix inverted=a.Inv();
         result=inverted;
         for(int i=-1; i>power; i--)
            result=result.MatMul(inverted);
        }
      else
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          result=a;
          for(int i=1; i<power; i++)
             result=result.MatMul(a);
         }
      }
//---
   c=result;
   return(true);
  }
```

MQL5の例

```
  matrix i= {{0, 1}, {-1, 0}};
  Print("i:\n", i);

  Print("i.Power(3):\n", i.Power(3));

  Print("i.Power(0):\n", i.Power(0));

  Print("i.Power(-3):\n", i.Power(-3));

  /*
  i:
  [[0,1]
   [-1,0]]

  i.Power(3):
  [[0,-1]
   [1,0]]

  i.Power(0):
  [[1,0]
   [0,1]]

  i.Power(-3):
  [[0, -1]
   [1,0]]
  */
```

Pythonの例

```
import numpy as np
from numpy.linalg import matrix_power

# matrix equiv. of the imaginary unit
i = np.array([[0, 1], [-1, 0]])
```

© 2000–2025, MetaQuotes Ltd.

```
print("i:\n",i)

# should = -i
print("matrix_power(i, 3) :\n",matrix_power(i, 3) )

print("matrix_power(i, 0):\n",matrix_power(i, 0))

# should = 1/(-i) = i, but w/ f.p. elements
print("matrix_power(i, -3):\n",matrix_power(i, -3))

i:
 [[ 0  1]
 [-1  0]]

matrix_power(i, 3) :
 [[ 0 -1]
 [ 1  0]]

matrix_power(i, 0):
 [[1 0]
 [0 1]]

matrix_power(i, -3):
 [[ 0.  1.]
 [-1.  0.]]
```

© 2000–2025, MetaQuotes Ltd.

Dot

2つのベクトルの内積。

```
double vector::Dot(
  const vector&  b      // 2番目のベクトル
   );
```

パラメータ

b

[in] ベクトル

戻り値

スカラー

注意事項

2つの行列の内積は、行列積matrix::MatMul()です。

次は、MQL5でのベクトルのスカラー積の単純なアルゴリズムです。

```
double VectorDot(const vector& vector_a, const vector& vector_b)
  {
   double dot=0;

   if(vector_a.Size()==vector_b.Size())
     {
      for(ulong i=0; i<vector_a.Size(); i++)
         dot+=vector_a[i]*vector_b[i];
     }

   return(dot);
  }
```

MQL5の例

```
   for(ulong i=0; i<rows; i++)
     {
      vector v1=a.Row(i);
      for(ulong j=0; j<cols; j++)
        {
         vector v2=b.Row(j);
         result[i][j]=v1.Dot(v2);
        }
     }
```

© 2000–2025, MetaQuotes Ltd.

Pythonの例

```
import numpy as np

a = [1, 0, 0, 1]
b = [4, 1, 2, 2]
print(np.dot(a, b))

>>> 6
```

© 2000–2025, MetaQuotes Ltd.

Kron

2 つの行 列 、行 列とベクト ル、ベクト ルと行 列 、または 2つのベクト ルのクロネッカー積 を返します。

```
matrix matrix::Kron(
  const matrix&  b      // 2番目の行列
   );

matrix matrix::Kron(
  const vector&  b      // ベクトル
   );

matrix vector::Kron(
  const matrix&  b      // 行列
   );

matrix vector::Kron(
  const vector&  b      // 2番目のベクトル
   );
```

パラメータ

b

[in] 2番 目 の行 列

戻 り値

行 列

注 意 事 項

クロネッカー積 は、ブロック行 列 乗 算 とも呼 ばれます。

次 は、MQL5での2つの行 列 のクロネッカー積 の単 純 なアルゴリズムです。

```
matrix MatrixKronecker(const matrix& matrix_a,const matrix& matrix_b)
  {
   ulong  M=matrix_a.Rows();
   ulong  N=matrix_a.Cols();
   ulong  P=matrix_b.Rows();
   ulong  Q=matrix_b.Cols();
   matrix matrix_c(M*P,N*Q);

   for(ulong m=0; m<M; m++)
      for(ulong n=0; n<N; n++)
         for(ulong p=0; p<P; p++)
            for(ulong q=0; q<Q; q++)
```

© 2000–2025, MetaQuotes Ltd.

```
                matrix_c[m*P+p][n*Q+q]=matrix_a[m][n] * matrix_b[p][q];

   return(matrix_c);
  }
```

MQL5の例

```
   matrix a={{1,2,3},{4,5,6}};
   matrix b=matrix::Identity(2,2);
   vector v={1,2};

   Print(a.Kron(b));
   Print(a.Kron(v));

  /*
   [[1,0,2,0,3,0]
    [0,1,0,2,0,3]
    [4,0,5,0,6,0]
    [0,4,0,5,0,6]]

   [[1,2,2,4,3,6]
    [4,8,5,10,6,12]]
  */
```

Pythonの例

```
import numpy as np

A = np.arange(1,7).reshape(2,3)
B = np.identity(2)
V = [1,2]
print(np.kron(A, B))
print("")
print(np.kron(A, V))

[[1. 0. 2. 0. 3. 0.]
 [0. 1. 0. 2. 0. 3.]
 [4. 0. 5. 0. 6. 0.]
 [0. 4. 0. 5. 0. 6.]]

[[ 1  2  2  4  3  6]
 [ 4  8  5 10  6 12]]
```

© 2000–2025, MetaQuotes Ltd.

Inner

2つの行列の内積。

```
matrix matrix::Inner(
  const matrix&  b      // 2番目の行列
   );
```

パラメータ

b

[in] 行列

戻り値

行列

注意事項

2つのベクトルの内積は、2つのベクトルvector::Dot()のドット積です。

次は、MQL5での2つの行列の内積の単純なアルゴリズムです。

```
bool MatrixInner(matrix& c, const matrix& a, const matrix& b)
  {
//--- 列の数は等しい必要がある
   if(a.Cols()!=b.Cols())
      return(false);
//--- 結果の行列のサイズは、各行列のベクトルの数に依存する
   ulong  rows=a.Rows();
   ulong  cols=b.Rows();
   matrix result(rows,cols);
//---
   for(ulong i=0; i<rows; i++)
     {
      vector v1=a.Row(i);
      for(ulong j=0; j<cols; j++)
        {
         vector v2=b.Row(j);
         result[i][j]=v1.Dot(v2);
        }
     }
//---
   c=result;
   return(true);
  }
```

MQL5の例

© 2000–2025, MetaQuotes Ltd.

```
   matrix a={{0,1,2},{3,4,5}};
   matrix b={{0,1,2},{3,4,5},{6,7,8}};
   matrix c=a.Inner(b);
   Print(c);
   matrix a1={{0,1,2}};
   matrix c1=a1.Inner(b);
   Print(c1);

  /*
  [[5,14,23]
  [14,50,86]]
  [[5,14,23]]
  */
```

Pythonの例

```
import numpy as np

A = np.arange(6).reshape(2, 3)
B = np.arange(9).reshape(3, 3)
A1= np.arange(3)
print(np.inner(A, B))
print("");
print(np.inner(A1, B))

import numpy as np

A = np.arange(6).reshape(2, 3)
B = np.arange(9).reshape(3, 3)
A1= np.arange(3)
print(np.inner(A, B))
print("");
print(np.inner(A1, B))
```

© 2000–2025, MetaQuotes Ltd.

Outer

2つの行列または2つのベクトルの外積を計算します。

```
matrix matrix::Outer(
  const matrix&  b      // 2番目の行列
   );

matrix vector::Outer(
  const vector&  b      // 2番目のベクトル
   );
```

パラメータ

b

[in] 行列

戻り値

行列

注意事項

クロネッカー積と同様に、外積もブロック行列（およびベクトル）の乗算です。

次は、MQL5での2つの行列の外積の単純なアルゴリズムです。

```
matrix MatrixOuter(const matrix& matrix_a, const matrix& matrix_b)
  {
//--- 結果の行列のサイズは、行列サイズに依存する
   ulong  rows=matrix_a.Rows()*matrix_a.Cols();
   ulong  cols=matrix_b.Rows()*matrix_b.Cols();
   matrix matrix_c(rows,cols);
   ulong  cols_a=matrix_a.Cols();
   ulong  cols_b=matrix_b.Cols();
//---
   for(ulong i=0; i<rows; i++)
     {
      ulong row_a=i/cols_a;
      ulong col_a=i%cols_a;
      for(ulong j=0; j<cols; j++)
        {
         ulong row_b=j/cols_b;
         ulong col_b=j%cols_b;
         matrix_c[i][j]=matrix_a[row_a][col_a] * matrix_b[row_b][col_b];
        }
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//---
   return(matrix_c);
  }
```

MQL5の例

```
   vector vector_a={0,1,2,3,4,5};
   vector vector_b={0,1,2,3,4,5,6};
   Print("vector_a.Outer\n",vector_a.Outer(vector_b));
   Print("vector_a.Kron\n",vector_a.Kron(vector_b));

   matrix matrix_a={{0,1,2},{3,4,5}};
   matrix matrix_b={{0,1,2},{3,4,5},{6,7,8}};
   Print("matrix_a.Outer\n",matrix_a.Outer(matrix_b));
   Print("matrix_a.Kron\n",matrix_a.Kron(matrix_b));

  /*
   vector_a.Outer
   [[0,0,0,0,0,0,0]
    [0,1,2,3,4,5,6]
    [0,2,4,6,8,10,12]
    [0,3,6,9,12,15,18]
    [0,4,8,12,16,20,24]
    [0,5,10,15,20,25,30]]
   vector_a.Kron
   [[
0
,
0
,
0
,
0
,
0
,
0
,
0
,
0
,1,2,3,4,5,6,0,2,4,6,8,10,12,0,3,6,9,12,15,18,0,4,8,12,16,20,24,0,5,10,15,20,25,30]]
   matrix_a.Outer
   [[0,0,0,0,0,0,0,0,0]
    [0,1,2,3,4,5,6,7,8]
    [0,2,4,6,8,10,12,14,16]
    [0,3,6,9,12,15,18,21,24]
    [0,4,8,12,16,20,24,28,32]
    [0,5,10,15,20,25,30,35,40]]
   matrix_a.Kron
   [[0,0,0,0,1,2,0,2,4]
```

© 2000–2025, MetaQuotes Ltd.

```
    [0,0,0,3,4,5,6,8,10]
    [0,0,0,6,7,8,12,14,16]
    [0,3,6,0,4,8,0,5,10]
    [9,12,15,12,16,20,15,20,25]
    [18,21,24,24,28,32,30,35,40]]
   */
```

Pythonの例

```
import numpy as np

A = np.arange(6)
B = np.arange(7)
print("np.outer")
print(np.outer(A, B))
print("np.kron")
print(np.kron(A, B))

A = np.arange(6).reshape(2, 3)
B = np.arange(9).reshape(3, 3)
print("np.outer")
print(np.outer(A, B))
print("np.kron")

np.outer
[[ 0  0  0  0  0  0  0]
 [ 0  1  2  3  4  5  6]
 [ 0  2  4  6  8 10 12]
 [ 0  3  6  9 12 15 18]
 [ 0  4  8 12 16 20 24]
 [ 0  5 10 15 20 25 30]]
np.kron
[ 0  0  0  0  0  0  0  0  1  2  3  4  5  6  0  2  4  6  8 10 12  0  3  6
  9 12 15 18  0  4  8 12 16 20 24  0  5 10 15 20 25 30]
np.outer
[[ 0  0  0  0  0  0  0  0  0]
 [ 0  1  2  3  4  5  6  7  8]
 [ 0  2  4  6  8 10 12 14 16]
 [ 0  3  6  9 12 15 18 21 24]
 [ 0  4  8 12 16 20 24 28 32]
 [ 0  5 10 15 20 25 30 35 40]]
np.kron
[[ 0  0  0  0  1  2  0  2  4]
 [ 0  0  0  3  4  5  6  8 10]
 [ 0  0  0  6  7  8 12 14 16]
 [ 0  3  6  0  4  8  0  5 10]
 [ 9 12 15 12 16 20 15 20 25]
 [18 21 24 24 28 32 30 35 40]]
```

© 2000–2025, MetaQuotes Ltd.

© 2000–2025, MetaQuotes Ltd.

CorrCoef

ピアソン相関係数（線形相関係数）を計算します。

```
matrix matrix::CorrCoef(
  const bool     rowvar=true  // 観測値の行ベクトルまたは列ベクトル
);

scalar vector::CorrCoef(
  const vector&  b            // 2番目のベクトル
);
```

戻り値

ピアソンの積率相関係数

注意事項

相関係数は[-1, 1]の範囲にあります。

浮動小数点の丸めにより、結果の配列はエルミートでなく、対角要素が1でなく、要素が不等式abs(a) <= 1を満たさない場合があります。その状況を改善するために、実数部と虚数部が間隔[-1, 1]にクリップされますが、これは複雑なケースではあまり役に立ちません。

次は、MQL5を使用して2つのベクトルの相関係数を計算するための単純なアルゴリズムです。

```
double VectorCorrelation(const vector& vector_x,const vector& vector_y)
  {
   ulong n=vector_x.Size()<vector_y.Size() ? vector_x.Size() : vector_y.Size();
   if(n<=1)
      return(0);

   ulong  i;
   double xmean=0;
   double ymean=0;
   for(i=0; i<n; i++)
     {
      if(!MathIsValidNumber(vector_x[i]))
         return(0);
      if(!MathIsValidNumber(vector_y[i]))
         return(0);
      xmean+=vector_x[i];
      ymean+=vector_y[i];
     }
   xmean/=(double)n;
   ymean/=(double)n;
```

© 2000-2025, MetaQuotes Ltd.

```
   double s=0;
   double xv=0;
   double yv=0;
   double t1=0;
   double t2=0;
//--- 計算
   s=0;
   for(i=0; i<n; i++)
     {
      t1=vector_x[i]-xmean;
      t2=vector_y[i]-ymean;
      xv+=t1*t1;
      yv+=t2*t2;
      s+=t1*t2;
     }
//--- 確認する
   if(xv==0 || yv==0)
      return(0);
//--- 結果を返す
   return(s/(MathSqrt(xv)*MathSqrt(yv)));
  }
```

MQL5の例

```
   vectorf vector_a={1,2,3,4,5};
   vectorf vector_b={0,1,0.5,2,2.5};
   Print("vectors correlation ",vector_a.CorrCoef(vector_b));
//---
   matrixf matrix_a={{1,2,3,4,5},
                     {0,1,0.5,2,2.5}};
   Print("matrix rows correlation\n",matrix_a.CorrCoef());
   matrixf matrix_a2=matrix_a.Transpose();
   Print("transposed matrix cols correlation\n",matrix_a2.CorrCoef(false));
   matrixf matrix_a3={{1.0f, 2.0f, 3.0f, 4.0f, 5.0f},
                      {0.0f, 1.0f, 0.5f, 2.0f, 2.5f},
                      {0.1f, 1.0f, 2.0f, 1.0f, 0.3f}};
   Print("rows correlation\n",matrix_a3.CorrCoef());
   Print("cols correlation\n",matrix_a3.CorrCoef(false));

  /*
   vectors correlation 0.9149913787841797
   matrix rows correlation
   [[1,0.91499138]
    [0.91499138,1]]
   transposed matrix cols correlation
   [[1,0.91499138]
    [0.91499138,1]]
   rows correlation
```

© 2000–2025, MetaQuotes Ltd.

```
   [[1,0.91499138,0.08474271]
    [0.91499138,1,-0.17123166]
    [0.08474271,-0.17123166,1]]
   cols correlation
   [[1,0.99587059,0.85375023,0.91129309,0.83773589]
    [0.99587059,1,0.80295509,0.94491106,0.88385159]
    [0.85375023,0.80295509,1,0.56362146,0.43088508]
    [0.91129309,0.94491106,0.56362146,1,0.98827404]
    [0.83773589,0.88385159,0.43088508,0.98827404,1]]
   */
```

Pythonの例

```
import numpy as np
va=[1,2,3,4,5]
vb=[0,1,0.5,2,2.5]
print("vectors correlation")
print(np.corrcoef(va,vb))

ma=np.zeros((2,5))
ma[0,:]=va
ma[1,:]=vb
print("matrix rows correlation")
print(np.corrcoef(ma))
print("transposed matrix cols correlation")
print(np.corrcoef(np.transpose(ma),rowvar=False))
print("")

ma1=[[1,2,3,4,5],[0,1,0.5,2,2.5],[0.1,1,0.2,1,0.3]]
print("rows correlation\n",np.corrcoef(ma1))
print("cols correlation\n",np.corrcoef(ma1,rowvar=False))

transposed matrix cols correlation
[[1.         0.91499142]
 [0.91499142 1.        ]]

rows correlation
 [[1.         0.91499142 0.1424941 ]
 [0.91499142 1.         0.39657517]
 [0.1424941  0.39657517 1.        ]]
cols correlation
 [[1.         0.99587059 0.98226063 0.91129318 0.83773586]
 [0.99587059 1.         0.99522839 0.94491118 0.88385151]
 [0.98226063 0.99522839 1.         0.97234063 0.92527551]
 [0.91129318 0.94491118 0.97234063 1.         0.98827406]
 [0.83773586 0.88385151 0.92527551 0.98827406 1.        ]]
```

© 2000–2025, MetaQuotes Ltd.

Cov

共分散行列を計算します。

```
matrix matrix::Cov(
  const bool     rowvar=true  // 観測値の行ベクトルまたは列ベクトル
);

matrix vector::Cov(
  const vector&  b            // 2番目のベクトル
);
```

パラメータ

b

[in] 2番目のベクトル

注意事項

共分散行列を計算します。

次は、MQL5を使用して2つのベクトルの共分散行列を計算するための単純なアルゴリズムです。

```
bool VectorCovariation(const vector& vector_a,const vector& vector_b,matrix&
matrix_c)
  {
   int i,j;
   int m=2;
   int n=(int)(vector_a.Size()<vector_b.Size()?vector_a.Size():vector_b.Size());
//--- checks
   if(n<=1)
      return(false);
   for(i=0; i<n; i++)
     {
      if(!MathIsValidNumber(vector_a[i]))
         return(false);
      if(!MathIsValidNumber(vector_b[i]))
         return(false);
     }
//---
   matrix matrix_x(2,n);
   matrix_x.Row(vector_a,0);
   matrix_x.Row(vector_b,1);
   vector t=vector::Zeros(m);
//--- 計算
   for(i=0; i<m; i++)
      for(j=0; j<n; j++)
         t[i]+=matrix_x[i][j]/double(n);
   for(i=0; i<m; i++)
```

© 2000–2025, MetaQuotes Ltd.

```
      for(j=0; j<n; j++)
         matrix_x[i][j]-=t[i];
//--- syrk  C=alpha*A^H*A+beta*C(beta=0は検討しない)
   matrix_c=matrix::Zeros(m,m);
   for(i=0; i<m; i++)
     {
      for(j=0; j<n; j++)
        {
         double v=matrix_x[i][j]/(n-1);
         for(int i_=i; i_<m; i_++)
            matrix_c[i][i_]+=v*matrix_x[i_][j];
        }
     }
//--- force symmetricity
   for(i=0; i<m-1; i++)
      for(j=i+1; j<m; j++)
         matrix_c[j][i]=matrix_c[i][j];
//---
   return(true);
  }
```

MQL5の例

```
   matrix matrix_a={{3,-2.1},{1.1,-1},{0.12,4.3}};
   Print("covariation cols\n",matrix_a.Cov(false));
   Print("covariation rows\n",matrix_a.Cov());

   vector vector_a=matrix_a.Col(0);
   vector vector_b=matrix_a.Col(1);
   Print("covariation vectors\n",vector_a.Cov(vector_b));

  /*
   covariation cols
   [[2.144133333333333,-4.286]
    [-4.286,11.71]]
   covariation rows
   [[13.005,5.355,-10.659]
    [5.355,2.205,-4.389]
    [-10.659,-4.389,8.736199999999998]]
   covariation vectors
   [[2.144133333333333,-4.286]
    [-4.286,11.71]]
  */
```

Pythonの例

```
import numpy as np
```

© 2000-2025, MetaQuotes Ltd.

```
matrix_a=np.array([[3,-2.1],[1.1,-1],[0.12,4.3]])
matrix_c=np.cov(matrix_a,rowvar=False)
print("covariation cols\n",matrix_c)
matrix_c2=np.cov(matrix_a)
print("covariation rows\n",matrix_c2)

vector_a=matrix_a[:,0]
vector_b=matrix_a[:,1]
matrix_c3=np.cov(vector_a,vector_b)
print("covariation vectors\n",matrix_c3)

covariation cols
 [[ 2.14413333 -4.286     ]
 [-4.286      11.71      ]]
covariation rows
 [[ 13.005    5.355  -10.659 ]
 [  5.355    2.205   -4.389 ]
 [-10.659   -4.389    8.7362]]
covariation vectors
 [[ 2.14413333 -4.286     ]
 [-4.286      11.71      ]]
```

© 2000–2025, MetaQuotes Ltd.

Correlate

2つのベクトルの相互相関を計算します。

```
vector vector::Correlate(
  const vector&           v,        // ベクトル
  ENUM_VECTOR_CONVOLVE    mode      // モード
   );
```

パラメータ

v

[in] 2番目のベクトル

mode

[in] modeパラメータは、線形畳み込み計算モードを決定します。ENUM_VECTOR_CONVOLVE列挙体からの値。

戻り値

2つのベクトルの相互相関

注意事項

modeパラメータは、線形畳み込み計算モードを決定します。

次は、MQL5を使用して2つのベクトルの相関係数を計算するための単純なアルゴリズムです。

```
vector VectorCrossCorrelationFull(const vector& a,const vector& b)
  {
   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=m+n-1;
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
      for(int i_=i; i_<i+m; i_++)
         c[i_]+=b[n-i-1]*a[i_-i];

   return(c);
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
vector VectorCrossCorrelationSame(const vector& a,const vector& b)
  {
   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=MathMax(m,n);
   vector c=vector::Zeros(size);
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0; i<n; i++)
     {
      for(int i_=i; i_<i+m; i_++)
        {
         int k=i_-size/2+1;
         if(k>=0 && k<size)
            c[k]+=b[n-i-1]*a[i_-i];
        }
     }

   return(c);
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
vector VectorCrossCorrelationValid(const vector& a,const vector& b)
  {
   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=MathMax(m,n)-MathMin(m,n)+1;
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
     {
      for(int i_=i; i_<i+m; i_++)
        {
         int k=i_-n+1;
         if(k>=0 && k<size)
            c[k]+=b[n-i-1]*a[i_-i];
        }
     }

   return(c);
  }
```

MQL5の例

```
   vector a={1,2,3,4,5};
   vector b={0,1,0.5};

   Print("full\n",a.Correlate(b,VECTOR_CONVOLVE_FULL));
   Print("same\n",a.Correlate(b,VECTOR_CONVOLVE_SAME));
   Print("valid\n",a.Correlate(b,VECTOR_CONVOLVE_VALID));
   Print("full\n",b.Correlate(a,VECTOR_CONVOLVE_FULL));

  /*
   full
```

© 2000–2025, MetaQuotes Ltd.

```
   [0.5,2,3.5,5,6.5,5,0]
   same
   [2,3.5,5,6.5,5]
   valid
   [3.5,5,6.5]
   full
   [0,5,6.5,5,3.5,2,0.5]
  */
```

Pythonの例

```
import numpy as np
a=[1,2,3,4,5]
b=[0,1,0.5]

print("full\n",np.correlate(a,b,'full'))
print("same\n",np.correlate(a,b,'same'));
print("valid\n",np.correlate(a,b,'valid'));
print("full\n",np.correlate(b,a,'full'))

full
 [0.5 2.  3.5 5.  6.5 5.  0. ]
same
 [2.  3.5 5.  6.5 5. ]
valid
 [3.5 5.  6.5]
full
 [0.  5.  6.5 5.  3.5 2.  0.5]
```

© 2000–2025, MetaQuotes Ltd.

Convolve

2つのベクトルの離散線形畳み込みを返します

```
vector vector::Convolve(
  const vector&            v,          // ベクトル
  ENUM_VECTOR_CONVOLVE     mode        // モード
   );
```

パラメータ

v

[out] 2番目のベクトル

mode

[in] modeパラメータは、線形畳み込み計算モードENUM_VECTOR_CONVOLVEを決定します。

戻り値

2つのベクトルの離散線形畳み込み。

次は、MQL5で2つのベクトルの畳み込みを計算するための単純なアルゴリズムです。

```
vector VectorConvolutionFull(const vector& a,const vector& b)
  {
   if(a.Size()<b.Size())
      return(VectorConvolutionFull(b,a));

   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=m+n-1;
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
      for(int i_=i; i_<i+m; i_++)
         c[i_]+=b[i]*a[i_-i];

   return(c);
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
vector VectorConvolutionSame(const vector& a,const vector& b)
  {
   if(a.Size()<b.Size())
      return(VectorConvolutionSame(b,a));

   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=MathMax(m,n);
```

© 2000-2025, MetaQuotes Ltd.

```
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
     {
      for(int i_=i; i_<i+m; i_++)
        {
         int k=i_-size/2+1;
         if(k>=0 && k<size)
            c[k]+=b[i]*a[i_-i];
        }
     }

   return(c);
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
vector VectorConvolutionValid(const vector& a,const vector& b)
  {
   if(a.Size()<b.Size())
      return(VectorConvolutionValid(b,a));

   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=MathMax(m,n)-MathMin(m,n)+1;
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
     {
      for(int i_=i; i_<i+m; i_++)
        {
         int k=i_-n+1;
         if(k>=0 && k<size)
            c[k]+=b[i]*a[i_-i];
        }
     }

   return(c);
  }
```

MQL5の例

```
  vector a= {1, 2, 3, 4, 5};
  vector b= {0, 1, 0.5};

  Print("full\n", a.Convolve(b, VECTOR_CONVOLVE_FULL));
  Print("same\n", a.Convolve(b, VECTOR_CONVOLVE_SAME));
  Print("valid\n", a.Convolve(b, VECTOR_CONVOLVE_VALID));
```

© 2000–2025, MetaQuotes Ltd.

```
/*
 full
 [0,1,2.5,4,5.5,7,2.5]
 same
 [1,2.5,4,5.5,7]
 valid
 [2.5,4,5.5]
*/
```

Pythonの例

```
import numpy as np
a=[1,2,3,4,5]
b=[0,1,0.5]

print("full\n",np.convolve(a,b,'full'))
print("same\n",np.convolve(a,b,'same'));
print("valid\n",np.convolve(a,b,'valid'));


full
 [0.  1.  2.5 4.  5.5 7.  2.5]
same
 [1.  2.5 4.  5.5 7. ]
valid
 [2.5 4.  5.5]
```

© 2000–2025, MetaQuotes Ltd.

行列変換

行列分解は次の場合に使用できます。

- 連立一次方程式を解く際の中間ステップ
- 行列反転
- 行列式を計算するとき
- 行列の固有値と固有ベクトルを見つけるとき
- 行列の解析関数を計算するとき
- 最小二乗法を使用するとき
- 微分方程式の数値解法で

問題に応じて、異なる行列分解タイプが使用されます。

関数	アクション
Cholesky	コレスキー分解を計算します。
Eig	正方行列の固有値と右固有ベクトルを計算します
EigVals	一般行列の固有値を計算します
LU	下三角行列と上三角行列の積としての行列のLU分解
LUP	行順列のみのLU分解を参照する、部分ピボットによるLUP因数分解：PA=LU
QR	行列のQR分解を計算します
SVD	特異値分解

© 2000–2025, MetaQuotes Ltd.

Cholesky

コレスキー分解を計算します。

```
bool matrix::Cholesky(
  matrix&  L      // matrix
   );
```

パラメータ

L key

[out] 下三角行列

戻り値

成功の場合はtrue、それ以外の場合はfalse

注意事項

正方行列aのコレスキー分解L*L.Hを返します。ここで、Lは下三角、.Hは共役転置演算子（aが実数値の場合は通常の転置）です。aはエルミート（実数値の場合は対称）かつ正定値でなければなりません。aがエルミートであるかどうかを検証するためのチェックは実行されません。さらに、aの下三角要素と対角要素のみが使用されます。実際に返されるのはLだけです。

例

```
  matrix matrix_a= {{5.7998084, -2.1825367}, {-2.1825367, 9.85910595}};
  matrix matrix_l;
  Print("matrix_a\n", matrix_a);

  matrix_a.Cholesky(matrix_l);
  Print("matrix_l\n", matrix_l);
  Print("check\n", matrix_l.MatMul(matrix_l.Transpose()));

  /*
  matrix_a
  [[5.7998084,-2.1825367]
   [-2.1825367,9.85910595]]
  matrix_l
  [[2.408279136645086,0]
   [-0.9062640068544704,3.006291985133859]]
  check
  [[5.7998084,-2.1825367]
   [-2.1825367,9.85910595]]
  */
```

© 2000–2025, MetaQuotes Ltd.

Eig

正方行列の固有値と右固有ベクトルを計算します。

```
bool matrix::Eig(
  matrix&  eigen_vectors,          // 固有ベクトルの行列
  vector&  eigen_values            // 固有値のベクトル
   );
```

固有値と固有ベクトルの複素数解

```
bool matrix::Eig(
  matrix<complex>& eigen_vectors, // 固有ベクトルの行列
  vector<complex>& eigen_values   // 固有値のベクトル
   );
```

パラメータ

eigen_vectors

[out] 垂直固有ベクトルの行列。

eigen_values

[out] 固有値のベクトル。

戻り値

固有値の計算中に複雑な解が見つかった場合、計算は停止され、エラーコードは4019 (ERR_MATH_OVERFLOW)に設定されます。Eigメソッドの複雑なオーバーロードを使用して、複素空間で完全な解を取得します

複素固有値の虚数部がゼロに等しい場合は実数固有値です。これは以下の例で見ることができます。

例

```
void OnStart()
  {
   matrix matrix_a =
     {
        {-3.474589, 1.106384, -9.091977,-3.925227 },
        {-5.522139, 2.366887,-15.162351,-6.357512 },
        { 8.394926,-2.960067, 22.292115, 9.524129 },
        { 7.803242,-2.080287, 19.217706, 8.186645 }
     };

   matrix eigen_vectors;
   vector eigen_values;

   bool res=matrix_a.Eig(eigen_vectors, eigen_values);
   Print("res=",res,"  error=",GetLastError());
   Print("Eigen vectors:\n",eigen_vectors, "\nEigen Values:\n",eigen_values);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 正しさを確認 A * v = lambda * v
   int vectors=0;
   for(ulong n=0; n<eigen_values.Size(); n++)
     {
      vector eigen_vector=eigen_vectors.Col(n);
      vector vector_d1   =eigen_vector*eigen_values[n];
      vector vector_d2   =matrix_a.MatMul(eigen_vector);

      ulong errors=vector_d1.Compare(vector_d2,1e-13);
      if(errors==0)
         vectors++;
     }
   Print("vectors=",vectors);

//--- 複素数解
   matrix<complex> eigen_vectors_c;
   vector<complex> eigen_values_c;
   ResetLastError();
   res=matrix_a.Eig(eigen_vectors_c,eigen_values_c);
   Print("res=",res,"  error=",GetLastError());
   Print("Eigen vectors:\n",eigen_vectors_c, "\nEigen Values:\n",eigen_values_c);

//--- 正しさを確認 A * v = lambda * v
   matrixc matrix_c;
   matrix_c.Assign(matrix_a);
   vectors=0;
   for(ulong n=0; n<eigen_values_c.Size(); n++)
     {
      vectorc eigen_vector_c=eigen_vectors_c.Col(n);
      vectorc vector_c1     =eigen_vector_c*eigen_values_c[n];
      vectorc vector_c2     =matrix_c.MatMul(eigen_vector_c);

      ulong errors=vector_c1.Compare(vector_c2,1e-13);
      if(errors==0)
         vectors++;
     }
   Print("vectors=",vectors);
  }
/* Result
   res=true  error=4019
   Eigen vectors:
   [[0.2649667608713664]
    [0.4488818803991876]
    [-0.6527335897527492]
    [-0.5497604331807768]]
   Eigen Values:
   [28.94158645962942]
   vectors=1
   res=true  error=0
```

© 2000–2025, MetaQuotes Ltd.

```
   Eigen vectors:
   [[(0.2649667608713664,0),(0.2227392219204762,0.3745470492013296),
(0.403285439089771,-0.1345146135716524),(-0.2050778513598178,0)]
    [(0.4488818803991876,0),(-0.2613452530342438,-0.3685707727819327),(-
0.3968506126372395,0.5257002255533059),(0.8014703373356256,0)]
    [(-0.6527335897527492,0),(-0.3418479634708521,-0.3299830378162041),(-
0.3553021031180292,-0.03685418759727989),(-0.05618703726964112,0)]
    [(-0.5497604331807768,0),(0.523227993452828,0.3262508584080381),
(0.3512835596143666,0.3666300350184092),(0.558955624442244,0)]]
   Eigen Values:
   [(28.94158645962942,0),(0.01022501104810897,0.02954980822331488),
(0.01022501104810897,-0.02954980822331488),(0.4090215182743634,0)]
   vectors=3
*/
```

© 2000–2025, MetaQuotes Ltd.

EigVals

一般行列の固有値を計算します。

```
bool matrix::EigVals(
  vector&  eigen_values      // 固有値のベクトル
   );
```

パラメータ

eigen_values

[out] 右固有値のベクトル

戻り値

成功の場合はtrue、それ以外の場合はfalse

注意事項

EigValsとEigの唯一の違いは、EigValsは固有ベクトルを計算せず、固有値のみを計算することです。

© 2000–2025, MetaQuotes Ltd.

LU

下三角行列と上三角行列の積としての行列のLU分解。

```
bool matrix::LU(
  matrix&  L,     // 下三角行列
  matrix&  U      // 上三角行列
   );
```

パラメータ

L key

[out] 下三角行列

U

[out] 上三角行列

戻り値

成功の場合はtrue、それ以外の場合はfalse

例

```
   matrix matrix_a={{1,2,3,4},
                    {5,2,6,7},
                    {8,9,3,10},
                    {11,12,14,4}};
   matrix matrix_l,matrix_u;
//--- LU分解
   matrix_a.LU(matrix_l,matrix_u);
   Print("matrix_l\n",matrix_l);
   Print("matrix_u\n",matrix_u);
//--- A = L * Uかどうかを確認する
   Print("check\n",matrix_l.MatMul(matrix_u));


   /*
   matrix_l
   [[1,0,0,0]
    [5,1,0,0]
    [8,0.875,1,0]
    [11,1.25,0.5904761904761905,1]]
   matrix_u
   [[1,2,3,4]
    [0,-8,-9,-13]
    [0,0,-13.125,-10.625]
    [0,0,0,-17.47619047619047]]
   check
   [[1,2,3,4]
    [5,2,6,7]
```

© 2000-2025, MetaQuotes Ltd.

```
    [8,9,3,10]
    [11,12,14,4]]
  */
```

© 2000–2025, MetaQuotes Ltd.

LUP

行順列のみのLU分解を参照する、部分ピボットによるLUP因数分解：PA=LU。

```
bool  LUP(
  matrix&  L,      // 下三角行列
  matrix&  U,      // 上三角行列
  matrix&  P       // 順列行列
   );
```

パラメータ

L key

[out] 下三角行列

U

[out] 上三角行列

P

[out] 順列行列

戻り値

成功の場合はtrue、それ以外の場合はfalse

例

```
   matrix matrix_a={{1,2,3,4},
                    {5,2,6,7},
                    {8,9,3,10},
                    {11,12,14,4}};
   matrix matrix_l,matrix_u,matrix_p;
//--- LUP分解
   matrix_a.LUP(matrix_l,matrix_u,matrix_p);
   Print("matrix_l\n",matrix_l);
   Print("matrix_u\n",matrix_u);
   Print("matrix_p\n",matrix_p);
//--- P * A = L * Uかどうかを確認する
   Print("P * A\n",matrix_p.MatMul(matrix_a));
   Print("L * U\n",matrix_l.MatMul(matrix_u));

   /*
   matrix_l
   [[1,0,0,0]
    [0.4545454545454545,1,0,0]
    [0.7272727272727273,-0.07894736842105282,1,0]
    [0.09090909090909091,-0.2631578947368421,-0.2262773722627738,1]]
   matrix_u
   [[11,12,14,4]
    [0,-3.454545454545454,-0.3636363636363633,5.181818181818182]
```

© 2000-2025, MetaQuotes Ltd.

```
   [0,0,-7.210526315789473,7.500000000000001]
   [0,0,0,6.697080291970803]]
  matrix_p
  [[0,0,0,1]
   [0,1,0,0]
   [0,0,1,0]
   [1,0,0,0]]
  P * A
  [[11,12,14,4]
   [5,2,6,7]
   [8,9,3,10]
   [1,2,3,4]]
  L * U
  [[11,12,14,4]
   [5,2,6,7]
   [8,9,3.000000000000001,10]
   [1,2,3,4]]
  */
```

© 2000–2025, MetaQuotes Ltd.

QR

行列のQR分解を計算します。

```
bool  QR(
  matrix&  Q,      // 正規直交列をもつ行列
  matrix&  R       // 上三角行列
   );
```

パラメータ

Q

[out] 正規直交列を含む行列。mode = 'complete'の場合、結果はaが実数/複素数であるかどうかに応じて、直交行列/ユニタリ行列になります。その場合、行列式は+/- 1のいずれかになります。入力配列の次元数が 2 より大きい場合、上記のプロパティを持つ行列のスタックが返されます。

R key

[out] 上三角行列

戻り値

成功の場合はtrue、それ以外の場合はfalse

例

```
//---  A*x = b
  matrix A = {{0, 1}, {1, 1}, {1, 1}, {2, 1}};
  Print("A \n", A);
  vector b = {1, 2, 2, 3};
  Print("b \n", b);
//--- A = Q*R
  matrix q, r;
  A.QR(q, r);
  Print("q \n", q);
  Print("r \n", r);
  matrix qr=q.MatMul(r);
  Print("qr \n", qr);
  /*
  A
  [[0,1]
  [1,1]
  [1,1]
  [2,1]]
  b
  [1,2,2,3]
  q
  [[0.4082482904638631,-0.8164965809277259,-1.110223024625157e-16,-
0.4082482904638631]
  [0.4625425214347352,-0.03745747856526496,0.7041241452319315,0.5374574785652647]
  [-0.5374574785652648,-0.03745747856526496,0.7041241452319316,-0.4625425214347352]
```

© 2000-2025, MetaQuotes Ltd.

```
[-0.5749149571305296,-0.5749149571305299,-0.09175170953613698,0.5749149571305296]]
r
[[-1.224744871391589,-0.2415816237971962]
[-1.22474487139159,-1.466326495188786]
[1.224744871391589,1.316496580927726]
[1.224744871391589,0.2415816237971961]]
qr
[[-1.110223024625157e-16,1]
[1,0.9999999999999999]
[1,1]
[2,1]]
*/
```

© 2000–2025, MetaQuotes Ltd.

SVD

特異値分解。

```
bool matrix::SVD(
  matrix&  U,                    // ユニタリ行列
  matrix&  V,                    // ユニタリ行列
  vector&  singular_values       // 特異値ベクトル
   );
```

パラメータ

U

[out] 左特異ベクトルから構成される、次数mのユニタリ行列

V

[out] 右特異ベクトルから構成される、次数nのユニタリ行列

singular_values

[out] 特異値

戻り値

成功の場合はtrue、それ以外の場合はfalse

例

```
  matrix a= {{0, 1, 2, 3, 4, 5, 6, 7, 8}};
  a=a-4;
  Print("matrix a \n", a);
  a.Reshape(3, 3);
  matrix b=a;
  Print("matrix b \n", b);
//--- SVD分解を実行する
  matrix U, V;
  vector singular_values;
  b.SVD(U, V, singular_values);
  Print("U \n", U);
  Print("V \n", V);
  Print("singular_values = ", singular_values);

// チェックブロック
//--- U * 単対角 * V = A
  matrix matrix_s;
  matrix_s.Diag(singular_values);
  Print("matrix_s \n", matrix_s);
  matrix matrix_vt=V.Transpose();
  Print("matrix_vt \n", matrix_vt);
  matrix matrix_usvt=(U.MatMul(matrix_s)).MatMul(matrix_vt);
  Print("matrix_usvt \n", matrix_usvt);
```

© 2000-2025, MetaQuotes Ltd.

```
  ulong errors=(int)b.Compare(matrix_usvt, 1e-9);
  double res=(errors==0);
  Print("errors=", errors);

//---- 他のチェック
  matrix U_Ut=U.MatMul(U.Transpose());
  Print("U_Ut \n", U_Ut);
  Print("Ut_U \n", (U.Transpose()).MatMul(U));

  matrix vt_V=matrix_vt.MatMul(V);
  Print("vt_V \n", vt_V);
  Print("V_vt \n", V.MatMul(matrix_vt));

  /*
  matrix a
  [[-4,-3,-2,-1,0,1,2,3,4]]
  matrix b
  [[-4,-3,-2]
   [-1,0,1]
   [2,3,4]]
  U
  [[-0.7071067811865474,0.5773502691896254,0.408248290463863]
   [-6.827109697437648e-17,0.5773502691896253,-0.8164965809277256]
   [0.7071067811865472,0.5773502691896255,0.4082482904638627]]
  V
  [[0.5773502691896258,-0.7071067811865474,-0.408248290463863]
   [0.5773502691896258,1.779939029415334e-16,0.8164965809277258]
   [0.5773502691896256,0.7071067811865474,-0.408248290463863]]
  singular_values = [7.348469228349533,2.449489742783175,3.277709923350408e-17]

  matrix_s
  [[7.348469228349533,0,0]
   [0,2.449489742783175,0]
   [0,0,3.277709923350408e-17]]
  matrix_vt
  [[0.5773502691896258,0.5773502691896258,0.5773502691896256]
   [-0.7071067811865474,1.779939029415334e-16,0.7071067811865474]
   [-0.408248290463863,0.8164965809277258,-0.408248290463863]]
  matrix_usvt
  [[-3.999999999999997,-2.999999999999999,-2]
   [-0.9999999999999981,-5.977974170712231e-17,0.9999999999999974]
   [2,2.999999999999999,3.999999999999996]]
  errors=0

  U_Ut
  [[0.9999999999999993,-1.665334536937735e-16,-1.665334536937735e-16]
   [-1.665334536937735e-16,0.9999999999999987,-5.551115123125783e-17]
   [-1.665334536937735e-16,-5.551115123125783e-17,0.9999999999999999]]
```

© 2000–2025, MetaQuotes Ltd.

```
  Ut_U
  [[0.9999999999999993,-5.551115123125783e-17,-1.110223024625157e-16]
   [-5.551115123125783e-17,0.9999999999999987,2.498001805406602e-16]
   [-1.110223024625157e-16,2.498001805406602e-16,0.999999999999999]]
  vt_V
  [[1,-5.551115123125783e-17,0]
   [-5.551115123125783e-17,0.9999999999999996,1.110223024625157e-16]
   [0,1.110223024625157e-16,0.9999999999999996]]
  V_vt
  [[0.9999999999999999,1.110223024625157e-16,1.942890293094024e-16]
   [1.110223024625157e-16,0.9999999999999998,1.665334536937735e-16]
   [1.942890293094024e-16,1.665334536937735e-16,0.9999999999999996]
  */
 }
```

© 2000–2025, MetaQuotes Ltd.

統計メソッド

行列とベクトルの記述統計量を計算するメソッドです。

関数	アクション
ArgMax	最大値のインデックスを返します
ArgMin	最小値のインデックスを返します
Max	行列/ベクトルの最大値を返します
Min	行列/ベクトルの最小値を返します
Ptp	行列/ベクトルまたは指定された行列軸の値の範囲を返します
Sum	指定された軸に対しても実行できる行列/ベクトル要素の合計を返します
Prod	指定された軸に対しても実行できる行列/ベクトル要素の積を返します
CumSum	指定された軸に沿ったものを含む、行列/ベクトル要素の累積和を返します
CumProd	指定された軸に沿ったものを含む、行列/ベクトル要素の累積積を返します
Percentile	行列/ベクトル要素または指定された軸に沿った要素の値の指定されたパーセンタイルを返します
Quantile	行列/ベクトル要素または指定された軸に沿った要素の値の指定された分位数を返します
Median	行列/ベクトル要素の中央値を計算します
Mean	要素値の算術平均を計算します
Average	行列/ベクトル値の加重平均を計算します
Std	行列/ベクトル要素または指定された軸に沿った要素の値の標準偏差を返します
Var	行列/ベクトル要素の値の分散を計算します

© 2000–2025, MetaQuotes Ltd.

ArgMax

最大値のインデックスを返します。

```
ulong vector::ArgMax();

ulong matrix::ArgMax();

vector matrix::ArgMax(
  const int  axis      // 軸
   );
```

パラメータ

axis

[in] 軸。0 — 水平軸、1 — 垂直軸

戻り値

最大値のインデックス

例

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_max=matrix_a.ArgMax(0);
   vector rows_max=matrix_a.ArgMax(1);
   ulong  matrix_max=matrix_a.ArgMax();

   Print("cols_max=",cols_max);
   Print("rows_max=",rows_max);
   Print("max index ",matrix_max,"  max value ",matrix_a.Flat(matrix_max));

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_max=[0,3,1]
   rows_max=[0,2,0,1]
   max index 5  max value 12.0
   */
```

© 2000–2025, MetaQuotes Ltd.

ArgMin

最小値のインデックスを返します。

```
ulong vector::ArgMin();

ulong matrix::ArgMin();

vector matrix::ArgMin(
  const int  axis      // 軸
   );
```

パラメータ

axis

[in] 軸。0 — 水平軸、1 — 垂直軸

戻り値

最小値のインデックス

例

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_min=matrix_a.ArgMin(0);
   vector rows_min=matrix_a.ArgMin(1);
   ulong  matrix_min=matrix_a.ArgMin();

   Print("cols_min=",cols_min);
   Print("rows_min=",rows_min);
   Print("min index ",matrix_min,"  min value ",matrix_a.Flat(matrix_min));

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_min=[1,0,0]
   rows_min=[2,0,2,0]
   min index 3  min value 1.0
   */
```

© 2000–2025, MetaQuotes Ltd.

Max

行列/ベクトルの最大値を返します。

```
double vector::Max();

double matrix::Max();

vector matrix::Max(
  const int  axis      // 軸
   );
```

パラメータ

axis

[in] 軸。0 — 水平軸、1 — 垂直軸

戻り値

行列/ベクトルの最大値

例

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_max=matrix_a.Max(0);
   vector rows_max=matrix_a.Max(1);
   double matrix_max=matrix_a.Max();

   Print("cols_max=",cols_max);
   Print("rows_max=",rows_max);
   Print("max value ",matrix_max);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_max=[10,11,12]
   rows_max=[10,12,6,11]
   max value 12.0
   */
```

© 2000–2025, MetaQuotes Ltd.

Min

行列/ベクトルの最小値を返します。

```
double vector::Min();

double matrix::Min();

vector matrix::Min(
  const int  axis      // 軸
   );
```

パラメータ

axis

[in] 軸。0 — 水平軸、1 — 垂直軸

戻り値

行列/ベクトルの最小値

例

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_min=matrix_a.Min(0);
   vector rows_min=matrix_a.Min(1);
   double matrix_min=matrix_a.Min();

   Print("cols_min=",cols_min);
   Print("rows_min=",rows_min);
   Print("min value ",matrix_min);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_min=[1,3,2]
   rows_min=[2,1,4,7]
   min value 1.0
   */
```

© 2000-2025, MetaQuotes Ltd.

Ptp

Max() – Min()と同等の、行列/ベクトルまたは指定された行列軸の値の範囲を返します。Ptp – ピークツーピーク

```
double vector::Ptp();

double matrix::Ptp();

vector matrix::Ptp(
  const int  axis      // 軸
   );
```

パラメータ

axis

[in] 軸。0 – 水平軸、1 – 垂直軸

戻り値

値の範囲を持つベクトル(最大 – 最小)

例

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_ptp=matrix_a.Ptp(0);
   vector rows_ptp=matrix_a.Ptp(1);
   double matrix_ptp=matrix_a.Ptp();

   Print("cols_ptp  ",cols_ptp);
   Print("rows_ptp  ",rows_ptp);
   Print("ptp value  ",matrix_ptp);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_ptp [9,8,10]
   rows_ptp [8,11,2,4]
   ptp value  11.0
   */
```

© 2000–2025, MetaQuotes Ltd.

Sum

指定された軸に対しても実行できる行列/ベクトル要素の合計を返します

```
double vector::Sum();

double matrix::Sum();

vector matrix::Sum(
  const int  axis      // 軸
   );
```

パラメータ

axis

[in] 軸。0 — 水平軸、1 — 垂直軸

戻り値

指定された軸に対しても実行できる行列/ベクトル要素の合計

例

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_sum=matrix_a.Sum(0);
   vector rows_sum=matrix_a.Sum(1);
   double matrix_sum=matrix_a.Sum();

   Print("cols_sum=",cols_sum);
   Print("rows_sum=",rows_sum);
   Print("sum value ",matrix_sum);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_sum=[24,27,27]
   rows_sum=[15,21,15,27]
   sum value 78.0
   */
```

© 2000–2025, MetaQuotes Ltd.

Prod

指定された軸に対しても実行できる行列/ベクトル要素の積を返します。

```
double vector::Prod(
  const double  initial=1      // 初期乗数
   );

double matrix::Prod(
  const double  initial=1      // 初期乗数
   );

vector matrix::Prod(
  const int     axis,          // 軸
  const double  initial=1      // 初期乗数
   );
```

パラメータ

axis

[in] 軸。0 — 水平軸、1 — 垂直軸

initial=1

[in] 初期乗数

例

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_prod=matrix_a.Prod(0);
   vector rows_prod=matrix_a.Prod(1);
   double matrix_prod=matrix_a.Prod();

   Print("cols_prod=",cols_prod);
   cols_prod=matrix_a.Prod(0,0.1);
   Print("cols_prod=",cols_prod);
   Print("rows_prod=",rows_prod);
   Print("prod value ",matrix_prod);

  /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_prod=[420,1320,864]
   cols_prod=[42,132,86.40000000000001]
   rows_prod=[60,96,120,693]
```

© 2000–2025, MetaQuotes Ltd.

```
   prod value 479001600.0
*/
```

© 2000–2025, MetaQuotes Ltd.

CumSum

指定された軸に沿ったものを含む、行列/ベクトル要素の累積和を返します。

```
vector vector::CumSum();

vector matrix::CumSum();

matrix matrix::CumSum(
  const int  axis      // 軸
   );
```

パラメータ

axis

[in] 軸。0 — 水平軸、1 — 垂直軸

戻り値

定された軸に沿った要素の累積合計

例

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   matrix cols_cumsum=matrix_a.CumSum(0);
   matrix rows_cumsum=matrix_a.CumSum(1);
   vector cumsum_values=matrix_a.CumSum();

   Print("cols_cumsum\n",cols_cumsum);
   Print("rows_cumsum\n",rows_cumsum);
   Print("cumsum values  ",cumsum_values);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_cumsum
   [[10,3,2]
    [11,11,14]
    [17,16,18]
    [24,27,27]]
   rows_cumsum
   [[10,13,15]
    [1,9,21]
    [6,11,15]
```

© 2000-2025, MetaQuotes Ltd.

```
    [7,18,27]]
   cumsum values  [10,13,15,16,24,36,42,47,51,58,69,78]
   */
```

© 2000–2025, MetaQuotes Ltd.

CumProd

指定された軸に沿ったものを含む、行列/ベクトル要素の累積積を返します。

```
vector vector::CumProd();

vector matrix::CumProd();

matrix matrix::CumProd(
  const int  axis      // 軸
   );
```

パラメータ

axis

[in] 軸。0 — 各列の横軸（行全体）、1 — 各行の縦軸（列全体）

戻り値

指定された軸に沿った要素の累積積

例

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   matrix cols_cumprod=matrix_a.CumProd(0);
   matrix rows_cumprod=matrix_a.CumProd(1);
   vector cumprod_values=matrix_a.CumProd();

   Print("cols_cumprod\n",cols_cumprod);
   Print("rows_cumprod\n",rows_cumprod);
   Print("cumprod values  ",cumprod_values);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_cumprod
   [[10,3,2]
    [10,24,24]
    [60,120,96]
    [420,1320,864]]
   rows_cumprod
   [[10,30,60]
    [1,8,96]
    [6,30,120]
```

© 2000-2025, MetaQuotes Ltd.

```
    [7,77,693]]
  cumprod values
[10,30,60,60,480,5760,34560,172800,691200,4838400,53222400,479001600]
  */
```

© 2000–2025, MetaQuotes Ltd.

Percentile

行列/ベクトル要素または指定された軸に沿った要素の値の指定されたパーセンタイルを返します。

```
double vector::Percentile(
  const int  percent      //
   );

double matrix::Percentile(
  const int  percent      //
   );

vector matrix::Percentile(
  const int  percent,     //
  const int  axis         // 軸
   );
```

パラメータ

percent

[in] 計算するパーセンタイル。0から100までの範囲である必要がある

axis

[in] 軸。0 — 水平軸、1 — 垂直軸

戻り値

パーセンタイル: スカラーまたはベクトル

注意事項

percentパラメータの有効な値は[0, 100]の範囲です。パーセンタイルの計算には線形アルゴリズムが使用されます。パーセンタイルを正しく計算するには、シーケンスを並べ替える必要があります。

例

```
   matrixf matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   Print("matrix_a\n",matrix_a);

   vectorf cols_percentile=matrix_a.Percentile(50,0);
   vectorf rows_percentile=matrix_a.Percentile(50,1);
   float matrix_percentile=matrix_a.Percentile(50);

   Print("cols_percentile ",cols_percentile);
   Print("rows_percentile ",rows_percentile);
   Print("percentile value  ",matrix_percentile);


   /*
   matrix_a
```

© 2000-2025, MetaQuotes Ltd.

```
   [[1,2,3]
    [4,5,6]
    [7,8,9]
    [10,11,12]]
   cols_percentile [5.5,6.5,7.5]
   rows_percentile [2,5,8,11]
   percentile value  6.5
   */
```

© 2000–2025, MetaQuotes Ltd.

Quantile

行列/ベクトル要素または指定された軸に沿った要素の値の指定された分位数を返します。

```
double vector::Quantile(
  const double quantile      // 分位数
   );

double matrix::Quantile(
  const double quantile      // 分位数
   );

vector matrix::Quantile(
  const double quantile,     // 分位数
  const int    axis          // 軸
   );
```

パラメータ

quantile

[in] 計算する分位数。0から1までの間である必要がある

axis

[in] 軸。0 — 水平軸、1 — 垂直軸

戻り値

分位数：スカラーまたはベクトル

注意事項

quantileパラメータは[0, 1]の範囲の値を取ります。分位数の計算には線形アルゴリズムが使用されます。分位数を正しく計算するには、シーケンスを並べ替える必要があります。

例

```
   matrixf matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   Print("matrix_a\n",matrix_a);

   vectorf cols_quantile=matrix_a.Quantile(0.5,0);
   vectorf rows_quantile=matrix_a.Quantile(0.5,1);
   float matrix_quantile=matrix_a.Quantile(0.5);

   Print("cols_quantile ",cols_quantile);
   Print("rows_quantile ",rows_quantile);
   Print("quantile value  ",matrix_quantile);

   /*
   matrix_a
   [[1,2,3]
```

© 2000-2025, MetaQuotes Ltd.

```
    [4,5,6]
    [7,8,9]
    [10,11,12]]
  cols_quantile [5.5,6.5,7.5]
  rows_quantile [2,5,8,11]
  quantile value  6.5
  */
```

© 2000–2025, MetaQuotes Ltd.

Median

行列/ベクトル要素の中央値を計算します。

```
double vector::Median();

double matrix::Median();

vector matrix::Median(
  const int  axis      // 軸
   );
```

パラメータ

axis

[in] 軸。0 — 水平軸、1 — 垂直軸

戻り値

中央値：スカラーまたはベクトル

注意事項

中央値とは、配列/ベクトル要素の上位半分を要素の下位半分から分離する真ん中の値です。Quantile(0.5)およびPercentile(50)と同じです。パーセンタイルを正しく計算するには、シーケンスを並べ替える必要があります。

例

```
   matrixf matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   Print("matrix_a\n",matrix_a);

   vectorf cols_median=matrix_a.Median(0);
   vectorf rows_median=matrix_a.Median(1);
   float matrix_median=matrix_a.Median();

   Print("cols_median ",cols_median);
   Print("rows_median ",rows_median);
   Print("median value  ",matrix_median);


   /*
   matrix_a
   [[1,2,3]
    [4,5,6]
    [7,8,9]
    [10,11,12]]
   cols_median [5.5,6.5,7.5]
```

© 2000-2025, MetaQuotes Ltd.

```
   rows_median [2,5,8,11]
   median value  6.5
   */
```

© 2000–2025, MetaQuotes Ltd.

Mean

要素値の算術平均を計算します。

```
double vector::Mean();

double matrix::Mean();

vector matrix::Mean(
  const int  axis      // 軸
   );
```

パラメータ

axis

[in] 軸。0 — 水平軸、1 — 垂直軸

戻り値

要素値の算術平均

例

```
   matrixf matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vectorf cols_mean=matrix_a.Mean(0);
   vectorf rows_mean=matrix_a.Mean(1);
   float matrix_mean=matrix_a.Mean();

   Print("cols_mean ",cols_mean);
   Print("rows_mean ",rows_mean);
   Print("mean value  ",matrix_mean);


   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_mean [6,6.75,6.75]
   rows_mean [5,7,5,9]
   mean value  6.5
   */
```

© 2000–2025, MetaQuotes Ltd.

Average

行列/ベクトル値の加重平均を計算します。

```
double vector::Average(
  const vector& weigts      // 重みベクトル
   );

double matrix::Average(
  const matrix& weigts      // 重み行列
);

vector matrix::Average(
  const matrix& weigts,     // 重み行列
  const int     axis        // 軸
   );
```

パラメータ

axis

[in] 軸。0 — 水平軸、1 — 垂直軸

戻り値

算術平均：スカラーまたはベクトル

注意事項

重み行列/ベクトルは、メインの行列/ベクトルに関連付けられています。

例

```
   matrixf matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   matrixf matrix_w=matrixf::Ones(4,3);
   Print("matrix_a\n",matrix_a);

   vectorf cols_average=matrix_a.Average(matrix_w,0);
   vectorf rows_average=matrix_a.Average(matrix_w,1);
   float matrix_average=matrix_a.Average(matrix_w);

   Print("cols_average ",cols_average);
   Print("rows_average ",rows_average);
   Print("average value  ",matrix_average);


   /*
   matrix_a
   [[10,3,2]
```

© 2000-2025, MetaQuotes Ltd.

```
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_average [6,6.75,6.75]
   rows_average [5,7,5,9]
   average value  6.5
   */ value  6.5
```

© 2000–2025, MetaQuotes Ltd.

Std

行列/ベクトル要素または指定された軸に沿った要素の値の標準偏差を返します。

```
double vector::Std();

double matrix::Std();

vector matrix::Std(
  const int  axis      // 軸
   );
```

パラメータ

axis

[in] 軸。0 — 水平軸、1 — 垂直軸

戻り値

標準偏差：スカラーまたはベクトル

注意事項

標準偏差は、平均からの二乗偏差の平均の平方根です。e., std = sqrt(mean(x))。ここでx = abs(a - a.mean())**2です。

平均二乗偏差は通常、x.sum() / Nとして計算されます。ここで、N = len(x)です。

例

```
   matrixf matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vectorf cols_std=matrix_a.Std(0);
   vectorf rows_std=matrix_a.Std(1);
   float matrix_std=matrix_a.Std();

   Print("cols_std ",cols_std);
   Print("rows_std ",rows_std);
   Print("std value  ",matrix_std);


   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_std [3.2403703,3.0310888,3.9607449]
```

© 2000-2025, MetaQuotes Ltd.

```
   rows_std [3.5590262,4.5460606,0.81649661,1.6329932]
   std value  3.452052593231201
   */
```

© 2000–2025, MetaQuotes Ltd.

Var

行列/ベクトル要素の値の分散を計算します。

```
double vector::Var();

double matrix::Var();

vector matrix::Var(
  const int  axis      // 軸
   );
```

パラメータ

axis

[in] 軸。0 — 水平軸、1 — 垂直軸

戻り値

分散：スカラーまたはベクトル

注意事項

分散は、平均からの二乗偏差の平均です。つまり、var = mean(x)、ここで x = abs(a - a.mean())**2です。

平均は通常、x.sum() / N として計算されます。ここで、N = len(x) です。

例

```
   matrixf matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vectorf cols_var=matrix_a.Var(0);
   vectorf rows_var=matrix_a.Var(1);
   float matrix_var=matrix_a.Var();

   Print("cols_var ",cols_var);
   Print("rows_var ",rows_var);
   Print("var value  ",matrix_var);


   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_var [10.5,9.1875,15.6875]
```

© 2000–2025, MetaQuotes Ltd.

```
   rows_var [12.666667,20.666666,0.66666669,2.6666667]
   var value  11.916666984558105
   */
```

© 2000–2025, MetaQuotes Ltd.

LinearRegression

Calculate a vector/matrix with calculated linear regression values.

```
vector vector::LinearRegression();

matrix matrix::LinearRegression(
  ENUM_MATRIX_AXIS  axis=AXIS_NONE      // axis along which regression is calculated
   );
```

Parameters

axis

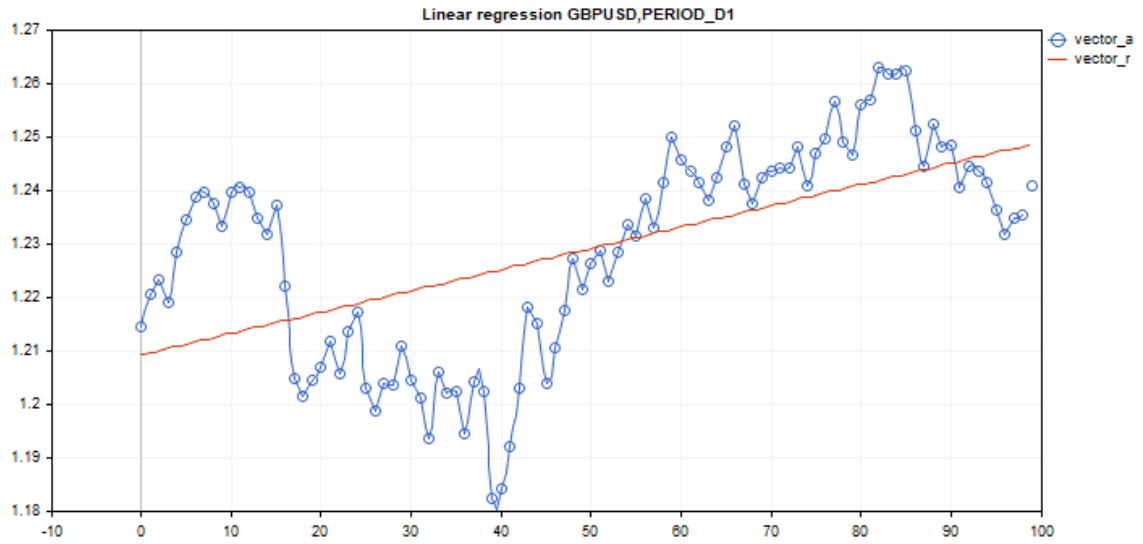
[in] Specifying the axis along which the regression is calculated. ENUM_MATRIX_AXIS enumeration value (AXIS_HORZ — horizontal axis, AXIS_VERT — vertical axis).

Return Value

Vector or matrix with calculated linear regression values.

Note

Linear regression is calculated using the standard regression equation: y (x) = b + a * x + b, where a is the line slope, while b is its Y axis shift.



Example:

```
#include <Graphics\Graphic.mqh>

#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

//+------------------------------------------------------------------+
```

© 2000–2025, MetaQuotes Ltd.

```
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector vector_a;
   vector_a.CopyRates(_Symbol,_Period,COPY_RATES_CLOSE,1,100);
   vector vector_r=vector_a.LinearRegression();

//--- switch off chart show
   ChartSetInteger(0,CHART_SHOW,false);

//--- arrays for drawing a graph
   double x[];
   double y1[];
   double y2[];
   ArrayResize(x,uint(vector_a.Size()));
   ArrayResize(y1,uint(vector_a.Size()));
   ArrayResize(y2,uint(vector_a.Size()));
   for(ulong i=0; i<vector_a.Size(); i++)
     {
      x[i]=(double)i;
      y1[i]=vector_a[i];
      y2[i]=vector_r[i];
     }

//--- graph title
   string title="Linear regression "+_Symbol+","+EnumToString(_Period);

   long   chart=0;
   string name="LinearRegression";

//--- create graph
   CGraphic graphic;
   graphic.Create(chart,name,0,0,0,GRAPH_WIDTH,GRAPH_HEIGHT);
   graphic.BackgroundMain(title);
   graphic.BackgroundMainSize(12);

//--- activation function graph
   CCurve *curvef=graphic.CurveAdd(x,y1,CURVE_POINTS_AND_LINES);
   curvef.Name("vector_a");
   curvef.LinesWidth(2);
   curvef.LinesSmooth(true);
   curvef.LinesSmoothTension(1);
   curvef.LinesSmoothStep(10);

//--- derivatives of activation function
   CCurve *curved=graphic.CurveAdd(x,y2,CURVE_LINES);
   curved.Name("vector_r");
   curved.LinesWidth(2);
```

© 2000–2025, MetaQuotes Ltd.

```
   curved.LinesSmooth(true);
   curved.LinesSmoothTension(1);
   curved.LinesSmoothStep(10);
   graphic.CurvePlotAll();
   graphic.Update();

//--- endless loop to recognize pressed keyboard buttons
   while(!IsStopped())
     {
      //--- press escape button to quit program
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
         break;
      //--- press PdDn to save graph picture
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0)
        {
         string file_names[];
         if(FileSelectDialog("Save Picture",NULL,"All files (*.*)|
*.*",FSD_WRITE_FILE,file_names,"LinearRegression.png")<1)
            continue;
         ChartScreenShot(0,file_names[0],GRAPH_WIDTH,GRAPH_HEIGHT);
        }
      Sleep(10);
     }

//--- clean up
   graphic.Destroy();
   ObjectDelete(chart,name);
   ChartSetInteger(0,CHART_SHOW,true);
  }
```

ENUM_MATRIX_AXIS

Enumeration for specifying the axis in all statistical functions for matrices.

ID	Description
AXIS_NONE	The axis is not specified. Calculation is performed over all matrix elements, as if it were a vector (see the Flat method).
AXIS_HORZ	Horizontal axis
AXIS_VERT	Vertical axis

© 2000–2025, MetaQuotes Ltd.

特徴メソッド

これらのメソッドは、次のような行列特徴の受信を可能にします。

- 行数
- 列数
- norm
- 条件番号
- 行列式
- 行列ランク
- 跡
- スペクトル

関数	アクション
Rows	行列の行数を返します
Cols	行列の列数を返します
Size	ベクトルのサイズを返します
Norm	行列/ベクトルのノルムを返します
Cond	行列の条件数を計算します
Det	可逆正方行列の行列式を計算します
SLogDet	行列式の符号と対数を計算します
Rank	ガウス法を使用して行列ランクを返します
Trace	行列の対角線に沿って合計を返します
Spectrum	積AT*Aからの固有値のセットとして行列のスペクトルを計算します

© 2000-2025, MetaQuotes Ltd.

Rows

行列の行数を返します。

```
ulong matrix::Rows()
```

戻り値

Integer.

例

```
matrix m= {{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}};
m.Reshape(3, 4);
Print("matrix m \n" , m);
Print("m.Rows()=", m.Rows());
Print("m.Cols()=", m.Cols());

/*
 matrix m
 [[1,2,3,4]
  [5,6,7,8]
  [9,10,11,12]]
 m.Rows()=3
 m.Cols()=4
*/
```

© 2000–2025, MetaQuotes Ltd.

Cols

行列の列数を返します

```
ulong matrix::Cols()
```

戻り値

Integer.

例

```
matrix m= {{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}};
m.Reshape(3, 4);
Print("matrix m \n" , m);
Print("m.Cols()=", m.Cols());
Print("m.Rows()=", m.Rows());

/*
 matrix m
 [[1,2,3,4]
  [5,6,7,8]
  [9,10,11,12]]
 m.Cols()=4
 m.Rows()=3
*/
```

© 2000–2025, MetaQuotes Ltd.

Size

ベクトルのサイズを返します。

```
ulong vector::Size()
```

戻り値

Integer.

例

```
matrix m={{1,2,3,4,5,6,7,8,9,10,11,12}};
m.Reshape(3,4);
Print("matrix m\n",m);
vector v=m.Row(1);
Print("v.Size()=",v.Size());
Print("v=",v);


/*
 matrix m
 [[1,2,3,4]
  [5,6,7,8]
  [9,10,11,12]]
 v.Size()=4
 v=[5,6,7,8]
*/
```

© 2000–2025, MetaQuotes Ltd.

Norm

行列/ベクトルのノルムを返します。

```
double vector::Norm(
  const ENUM_VECTOR_NORM  norm,      // ベクトルのノルム
  const int               norm_p=2  // VECTOR_NORM_Pの場合のpノルムの数
   );

double matrix::Norm(
  const ENUM_MATRIX_NORM  norm       // 行列ノルム
   );
```

パラメータ

norm

[in] ノルムの次元

戻り値

行列/ベクトルのノルム。

注意事項

- VECTOR_NORM_INFは、ベクトル要素の最大絶対値です。
- VECTOR_NORM_MINUS_INFは、ベクトル要素の最小絶対値です。
- VECTOR_NORM_Pベクトルのpノルムです。norm_p=0の場合、これはゼロ以外のベクトル要素の数です。norm_p=1は、ベクトル要素の絶対値の合計です。norm_p=2は、ベクトル要素値の平方和の平方根です。Pノルムは負になる場合があります。
- MATRIX_NORM_FROBENIUSは、行列要素値の平方和の平方根です。フロベニウスノルムとベクトルP2ノルムは一致しています。
- MATRIX_NORM_SPECTRALは、行列スペクトルの最大値です。
- MATRIX_NORM_NUCLEARは、行列の特異値の合計です。
- MATRIX_NORM_INFは、行列の垂直ベクトルの中で最大のベクトルP1ノルムです。行列inf-normとベクトルinf-normは一致しています。
- MATRIX_NORM_MINUS_INFは、行列の垂直ベクトルの最小ベクトルP1ノルムです。
- MATRIX_NORM_P1は、水平行列ベクトルの中で最大のベクトルP1ノルムです。
- MATRIX_NORM_MINUS_P1は、水平行列ベクトルの中で最小のベクトルP1ノルムです。
- MATRIX_NORM_P2は、行列の最大特異値です。
- MATRIX_NORM_MINUS_P2は、行列の最小特異値です。

次は、MQL5でベクトルのPノルムを計算するための単純なアルゴリズムです。

```
double VectorNormP(const vector& v,int norm_value)
  {
   ulong  i;
   double norm=0.0;
```

© 2000-2025, MetaQuotes Ltd.

```
//---
   switch(norm_value)
     {
      case 0 :
         for(i=0; i<v.Size(); i++)
            if(v[i]!=0)
               norm+=1.0;
         break;
      case 1 :
         for(i=0; i<v.Size(); i++)
            norm+=MathAbs(v[i]);
         break;
      case 2 :
         for(i=0; i<v.Size(); i++)
            norm+=v[i]*v[i];
         norm=MathSqrt(norm);
         break;
      default :
         for(i=0; i<v.Size(); i++)
            norm+=MathPow(MathAbs(v[i]),norm_value);
         norm=MathPow(norm,1.0/norm_value);
     }
//---
   return(norm);
  }
```

MQL5の例

```
  matrix a= {{0, 1, 2, 3, 4, 5, 6, 7, 8}};
  a=a-4;
  Print("matrix a \n", a);
  a.Reshape(3, 3);
  matrix b=a;
  Print("matrix b \n", b);
  Print("b.Norm(MATRIX_NORM_P2)=", b.Norm(MATRIX_NORM_FROBENIUS));
  Print("b.Norm(MATRIX_NORM_FROBENIUS)=", b.Norm(MATRIX_NORM_FROBENIUS));
  Print("b.Norm(MATRIX_NORM_INF)", b.Norm(MATRIX_NORM_INF));
  Print("b.Norm(MATRIX_NORM_MINUS_INF)", b.Norm(MATRIX_NORM_MINUS_INF));
  Print("b.Norm(MATRIX_NORM_P1)=)", b.Norm(MATRIX_NORM_P1));
  Print("b.Norm(MATRIX_NORM_MINUS_P1)=", b.Norm(MATRIX_NORM_MINUS_P1));
  Print("b.Norm(MATRIX_NORM_P2)=", b.Norm(MATRIX_NORM_P2));
  Print("b.Norm(MATRIX_NORM_MINUS_P2)=", b.Norm(MATRIX_NORM_MINUS_P2));

  /*
  matrix a
  [[-4,-3,-2,-1,0,1,2,3,4]]
  matrix b
  [[-4,-3,-2]
  [-1,0,1]
```

© 2000–2025, MetaQuotes Ltd.

```
   [2,3,4]]
   b.Norm(MATRIX_NORM_P2)=7.745966692414834
   b.Norm(MATRIX_NORM_FROBENIUS)=7.745966692414834
   b.Norm(MATRIX_NORM_INF)9.0
   b.Norm(MATRIX_NORM_MINUS_INF)2.0
   b.Norm(MATRIX_NORM_P1)=)7.0
   b.Norm(MATRIX_NORM_MINUS_P1)=6.0
   b.Norm(MATRIX_NORM_P2)=7.348469228349533
   b.Norm(MATRIX_NORM_MINUS_P2)=1.857033188519056e-16
   */
```

Pythonの例

```
import numpy as np
from numpy import linalg as LA
a = np.arange(9) - 4
print("a \n",a)
b = a.reshape((3, 3))
print("b \n",b)
print("LA.norm(b)=",LA.norm(b))
print("LA.norm(b, 'fro')=",LA.norm(b, 'fro'))
print("LA.norm(b, np.inf)=",LA.norm(b, np.inf))
print("LA.norm(b, -np.inf)=",LA.norm(b, -np.inf))
print("LA.norm(b, 1)=",LA.norm(b, 1))
print("LA.norm(b, -1)=",LA.norm(b, -1))
print("LA.norm(b, 2)=",LA.norm(b, 2))
print("LA.norm(b, -2)=",LA.norm(b, -2))


 [-4 -3 -2 -1  0  1  2  3  4]
b
 [[-4 -3 -2]
 [-1  0  1]
 [ 2  3  4]]
LA.norm(b)= 7.745966692414834
LA.norm(b, 'fro')= 7.745966692414834
LA.norm(b, np.inf)= 9.0
LA.norm(b, -np.inf)= 2.0
LA.norm(b, 1)= 7.0
LA.norm(b, -1)= 6.0
LA.norm(b, 2)= 7.3484692283495345
LA.norm(b, -2)= 1.857033188519056e-16
```

© 2000–2025, MetaQuotes Ltd.

Cond

行列の条件数を計算します。

```
double matrix::Cond(
  const ENUM_MATRIX_NORM  norm      // 行列ノルム
   );
```

パラメータ

norm

[in] ENUM_MATRIX_NORMからのノルムの順序

戻り値

行列の条件数。無限の場合もあります。

注意事項

xの条件数は、xのノルムにxの逆数のノルムを掛けたものとして定義されます [1]。ノルムは、通常のL2ノルム（二乗和の平方根）または他の多くの行列ノルムの1つにすることができます。

条件数は、行列Aノルムとその逆数の積に等しいK値です。条件数が高い行列は、悪条件と呼ばれます。条件数が低いものは良条件と呼ばれます。逆行列は、正方行列性および非特異性の条件によって制限されないように、擬似逆行列を使用して取得されます。

例外はスペクトル条件番号です。

次は、MQL5でスペクトル条件数を計算するための単純なアルゴリズムです。

```
double MatrixCondSpectral(matrix& a)
  {
   double norm=0.0;
   vector v=a.Spectrum();

   if(v.Size()>0)
     {
      double max_norm=v[0];
      double min_norm=v[0];
      for(ulong i=1; i<v.Size(); i++)
        {
         double real=MathAbs(v[i]);
         if(max_norm<real)
            max_norm=real;
         if(min_norm>real)
            min_norm=real;
        }
      max_norm=MathSqrt(max_norm);
      min_norm=MathSqrt(min_norm);
```

© 2000–2025, MetaQuotes Ltd.

```
      if(min_norm>0.0)
         norm=max_norm/min_norm;
     }

   return(norm);
  }
```

MQL5の例

```
matrix a= {{1, 0, -1}, {0, 1, 0}, { 1, 0, 1}};
Print("a.Cond(MATRIX_NORM_P2)=", a.Cond(MATRIX_NORM_P2));
Print("a.Cond(MATRIX_NORM_FROBENIUS)=", a.Cond(MATRIX_NORM_FROBENIUS));
Print("a.Cond(MATRIX_NORM_INF)=", a.Cond(MATRIX_NORM_INF));
Print("a.Cond(MATRIX_NORM_MINUS_INF)=", a.Cond(MATRIX_NORM_MINUS_INF));
Print("a.Cond(MATRIX_NORM_P1)=)", a.Cond(MATRIX_NORM_P1));
Print("a.Cond(MATRIX_NORM_MINUS_P1)=", a.Cond(MATRIX_NORM_MINUS_P1));
Print("a.Cond(MATRIX_NORM_P2)=", a.Cond(MATRIX_NORM_P2));
Print("a.Cond(MATRIX_NORM_MINUS_P2)=", a.Cond(MATRIX_NORM_MINUS_P2));

/*
matrix a
[[1,0,-1]
[0,1,0]
[1,0,1]]
a.Cond(MATRIX_NORM_P2)=1.414213562373095
a.Cond(MATRIX_NORM_FROBENIUS)=3.162277660168379
a.Cond(MATRIX_NORM_INF)=2.0
a.Cond(MATRIX_NORM_MINUS_INF)=0.9999999999999997
a.Cond(MATRIX_NORM_P1)=)2.0
a.Cond(MATRIX_NORM_MINUS_P1)=0.9999999999999998
a.Cond(MATRIX_NORM_P2)=1.414213562373095
a.Cond(MATRIX_NORM_MINUS_P2)=0.7071067811865472
*/
```

Pythonの例

```
import numpy as np
from numpy import linalg as LA
a = np.array([[1, 0, -1], [0, 1, 0], [1, 0, 1]])
print("a \n",a)
print("LA.cond(a)=",LA.cond(a))
print("LA.cond(a, 'fro')=",LA.cond(a, 'fro'))
print("LA.cond(a, np.inf)=",LA.cond(a, np.inf))
print("LA.cond(a, -np.inf)=",LA.cond(a, -np.inf))
print("LA.cond(a, 1)=",LA.cond(a, 1))
print("LA.cond(a, -1)=",LA.cond(a, -1))
print("LA.cond(a, 2)=",LA.cond(a, 2))
```

© 2000–2025, MetaQuotes Ltd.

```
print("LA.cond(a, -2)=",LA.cond(a, -2))


 [[ 1  0 -1]
 [ 0  1  0]
 [ 1  0  1]]
LA.cond(a)= 1.4142135623730951
LA.cond(a, 'fro')= 3.1622776601683795
LA.cond(a, np.inf)= 2.0
LA.cond(a, -np.inf)= 1.0
LA.cond(a, 1)= 2.0
LA.cond(a, -1)= 1.0
LA.cond(a, 2)= 1.4142135623730951
LA.cond(a, -2)= 0.7071067811865475
```

© 2000–2025, MetaQuotes Ltd.

Det

可逆正方行列の行列式を計算します。

```
double matrix::Det()
```

戻り値

行列の行列式。

注意事項

2次および3次の行列式は、サラスの方法に従って計算されます。d2=a11*a22-a12*a21;
d3=a11*a22*a33+a12*a23*a31+a13*a21*a32-a13*a22*a31-a11*a23*a32-a12*a21*a33

行列式は、行列を上三角行列形式に縮小することにより、ガウス法によって計算されます。上三角行列の行列式は、主対角要素の積に等しくなります。

少なくとも1つの行列の行または列がゼロの場合、行列式はゼロです。

2つ以上の行列の行または列が線形従属である場合、その行列式はゼロです。

行列の行列式は、その固有値の積に等しくなります。

MQL5の例

```
   matrix m={{1,2},{3,4}};
   double det=m.Det();
   Print("matrix m\n",m);
   Print("det(m)=",det);
   /*
   matrix m
   [[1,2]
    [3,4]]
   det(m)=-2.0
   */
```

Pythonの例

```
import numpy as np

a = np.array([[1, 2], [3, 4]])
print('a \n',a)
print('nnp.linalg.det(a) \n',np.linalg.det(a))



 [[1 2]
 [3 4]]
```

© 2000-2025, MetaQuotes Ltd.

```
np.linalg.det(a)
 -2.0000000000000004
```

© 2000–2025, MetaQuotes Ltd.

SLogDet

行列式の符号と対数を計算します。

```
double matrix::SLogDet(
  int&  sign      // sign
   );
```

パラメータ

sign

[out] 行列式の符号。符号が偶数の場合、行列式は正です。

戻り値

行列式の符号を表現する数。

注意事項

行列式は、行列を上三角行列形式に縮小することにより、ガウス法によって計算されます。上三角行列の行列式は、主対角要素の積に等しくなります。積の対数は対数の和に等しくなります。したがって、行列式の計算時にオーバーフローが発生した場合は、SLogDetメソッドを使用できます。

符号が偶数の場合、行列式は正です。

例

```
a = np.array([[1, 2], [3, 4]])  (sign, logdet) = np.linalg.slogdet(a)  (sign,
logdet) (-1, 0.69314718055994529) # 符号は異なる場合がある * np.exp(logdet) -2.0
```

© 2000–2025, MetaQuotes Ltd.

Rank

ガウス法を使用して行列ランクを返します。

```
int  Rank()
```

戻り値

行列のランク。

注意事項

m行n列の行列Aの行（または列）のシステムのランクは、線形に独立した行（または列）の最大数です。いくつかの行（列）は、他の行で線形に表現できない場合、線形独立と呼ばれます。行システムのランクは常に列システムのランクと同じです。この値は行列のランクと呼ばれます。

MQL5の例

```
matrix a=matrix::Eye(4, 4);;
Print("matrix a \n", a);
Print("a.Rank()=", a.Rank());

matrix I=matrix::Eye(4, 4);
I[3, 3] = 0.;     // rank deficient matrix
Print("I \n", I);
Print("I.Rank()=", I.Rank());

matrix b=matrix::Ones(1, 4);
Print("b \n", b);
Print("b.Rank()=", b.Rank());;// 1 dimension - rank 1 unless all 0

matrix  zeros=matrix::Zeros(4, 1);
Print("zeros \n", zeros);
Print("zeros.Rank()=", zeros.Rank());

/*
matrix a
[[1,0,0,0]
[0,1,0,0]
[0,0,1,0]
[0,0,0,1]]
a.Rank()=4

I
[[1,0,0,0]
[0,1,0,0]
[0,0,1,0]
[0,0,0,0]]
```

© 2000–2025, MetaQuotes Ltd.

```
  I.Rank()=3

  b
  [[1,1,1,1]]
  b.Rank()=1

  zeros
  [[0]
  [0]
  [0]
  [0]]
  zeros.Rank()=0
  */
```

Pythonの例

```
import numpy as np
from numpy.linalg import matrix_rank
a=(np.eye(4)) # Full rank matrix
print("a \n", a)
print("matrix_rank(a)=",matrix_rank(a))
I=np.eye(4)
I[-1,-1] = 0. # rank deficient matrix
print("I \n",I)
print("matrix_rank(I)=",matrix_rank(I))

b=np.ones((4,))
print("b \n",b)
print("matrix_rank(b)=",matrix_rank(b)) # 1 dimension - rank 1 unless all 0

zeros=np.zeros((4,))
print("zeroes \n",zeros)
print("matrix_rank(zeros)=",matrix_rank(zeros))


 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
matrix_rank(a)= 4

I
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 0.]]
matrix_rank(I)= 3
```

© 2000–2025, MetaQuotes Ltd.

```
b
 [1. 1. 1. 1.]
matrix_rank(b)= 1

zeroes
 [0. 0. 0. 0.]
matrix_rank(zeros)= 0
```

© 2000–2025, MetaQuotes Ltd.

Trace

行列の対角線に沿って合計を返します。

```
double matrix::Trace()
```

戻り値

対角線に沿った合計。

注意事項

行列の跡は、その固有値の和に等しくなります。

MQL5の例

```
  matrix a= {{0, 1, 2, 3, 4, 5, 6, 7, 8}};
  a.Reshape(3, 3);
  Print("matrix a \n", a);
  Print("a.Trace() \n", a.Trace());

  /*
  matrix a
  [[0,1,2]
  [3,4,5]
  [6,7,8]]
  a.Trace()
  12.0
  */
```

Pythonの例

```
a = np.arange(9).reshape((3,3))
print('a \n',a)
print('np.trace(a) \n',np.trace(a))


 [[0 1 2]
 [3 4 5]
 [6 7 8]]

np.trace(a)
 12
```

© 2000–2025, MetaQuotes Ltd.

Spectrum

積AT*Aからの固有値のセットとして行列のスペクトルを計算します。

```
vector matrix::Spectrum()
```

戻り値

行列固有値のベクトルとしての行列のスペクトル。

例

```
double MatrixCondSpectral(matrix& a)
  {
   double norm=0.0;
   vector v=a.Spectrum();

   if(v.Size()>0)
     {
      double max_norm=v[0];
      double min_norm=v[0];
      for(ulong i=1; i<v.Size(); i++)
        {
         double real=MathAbs(v[i]);
         if(max_norm<real)
            max_norm=real;
         if(min_norm>real)
            min_norm=real;
        }
      max_norm=MathSqrt(max_norm);
      min_norm=MathSqrt(min_norm);
      if(min_norm>0.0)
         norm=max_norm/min_norm;
     }

   return(norm);
  }
```

© 2000–2025, MetaQuotes Ltd.

連立一次方程式を解くための行列法

連立一次方程式を解き、逆行列を計算するメソッド。

関数	アクション
Solve	線形行列方程式または線形代数方程式系を解きます。
LstSq	線形代数方程式の最小二乗解を返します（非正方行列または縮退行列の場合）
Inv	ジョーダン・ガウス法により正方非縮退行列の（乗法）逆行列を計算します。
PInv	ムーア・ペンローズ逆行列法による行列の疑似逆行列を計算します

© 2000–2025, MetaQuotes Ltd.

Solve

線形行列方程式または線形代数方程式系を解きます。

```
vector matrix::Solve(
  const vector  b       // 縦座標または「従属変数」の値
   );
```

パラメータ

b

[in] 縦座標または「従属変数」の値（自由項のベクトル）

戻り値

系 a＊x = b の解を持つベクトル。

注意事項

行列の行または列の少なくとも 1 つがゼロの場合、システムには解がありません。

2つ以上の行列の行または列が線形従属である場合、システムには解がありません。

例

```
//--- SLAEの解
   vector_x=matrix_a.Solve(vector_b);
//--- a * x = bかどうか確認する
   result_vector=matrix_a.MatMul(vector_x);
   errors=vector_b.Compare(result_vector,1e-12);
```

© 2000–2025, MetaQuotes Ltd.

LstSq

線形代数方程式の最小二乗解を返します（非正方行列または縮退行列の場合）。

```
vector matrix::LstSq(
  const vector  b      // 縦座標または「従属変数」の値
   );
```

パラメータ

b

[in] 縦座標または「従属変数」の値（自由項のベクトル）

戻り値

系 a*x = b の解を持つベクトル。これは、正確な解を持つシステムにのみ当てはまります。

例

```
   matrix a={{3, 2},
             {4,-5},
             {3, 3}};
   vector b={7,40,3};
//---
   vector x=a.LstSq(b);
//--- 確認。[5, -4]が必要
   Print("x=", x);
//--- 確認。[7, 40, 3]が必要
   vector b1=a.MatMul(x);
   Print("b1=",b1);

/*
  x=[5.000000000000002,-4]
  b1=[7.000000000000005,40.00000000000001,3.000000000000005]
*/
```

© 2000-2025, MetaQuotes Ltd.

Inv

ジョーダン・ガウス法により正方可逆行列の乗法逆行列を計算します

```
matrix matrix::Inv()
```

戻り値

行列の乗法逆

注意事項

元の行列と逆行列の積が恒等行列です。

行列の行または列の少なくとも1つがゼロの場合、逆行列は取得できません。

2つ以上の行列の行または列が線形従属である場合、逆行列は取得できません。

例

```
int TestInverse(const int size_m)
  {
   int    i,j,errors=0;
   matrix matrix_a(size_m,size_m);
//--- 正方行列に書き込む
   MatrixTestFirst(matrix_a);
//--- マイクロ秒が測定される
   ulong t1=GetMicrosecondCount();
//--- 逆行列を取得する
   matrix inverse=matrix_a.Inv();
//--- 測定する
   ulong t2=GetMicrosecondCount();
//--- 正しさを確認する
   matrix identity=matrix_a.MatMul(inverse);
//---
   for(i=0; i<size_m; i++)
     {
      for(j=0; j<size_m; j++)
        {
         double value;
         //--- 1は対角線に沿っている必要がある
         if(i==j)
            value=1.0;
         else
            value=0.0;
         if(MathClassify(identity[i][j])>FP_ZERO)
            errors++;
         else
```

© 2000-2025, MetaQuotes Ltd.

```
            {
             if(identity[i][j]!=value)
               {
                double diff=MathAbs(identity[i][j]-value);
                //--- 掛け算と割り算が多すぎるので、チェックの精度を下げる
                if(diff>1e-9)
                   errors++;
               }
            }
         }
      }
//---
   double elapsed_time=double(t2-t1)/1000.0;
   printf("Inversion of matrix %d x %d  %s  errors=%d  time=%.3f
ms",size_m,size_m,errors==0?"passed":"failed",errors,elapsed_time);
   return(errors);
  }
```

© 2000-2025, MetaQuotes Ltd.

PInv

ムーア・ペンローズ逆行列法による行列の疑似逆行列を計算します。

```
matrix matrix::PInv()
```

戻り値

行列の疑似逆行列

例

```
int TestPseudoInverse(const int size_m, const int size_k)
  {
   matrix matrix_a(size_m,size_k);
   matrix matrix_inverted(size_k,size_m);
   matrix matrix_temp;
   matrix matrix_a2;
//--- 行列に書き込む
   MatrixTestFirst(matrix_a);
//--- 逆にする
   matrix_inverted=matrix_a.PInv();
//--- 正しさを確認する
   int errors=0;
//--- A * A+ * A = A   (A+ is a pseudo-inverse of A)
   matrix_temp=matrix_a.MatMul(matrix_inverted);
   matrix_a2=matrix_temp.MatMul(matrix_a);
   errors=(int)matrix_a.CompareByDigits(matrix_a2,10);

   printf("PseudoInversion %s matrix_size  %d x %d  errors=%
d",errors==0?"passed":"failed",size_m,size_k,errors);
//---
   return(errors);
  }
```

© 2000–2025, MetaQuotes Ltd.

機械学習

これらの方法は、機械学習で使用されます。

ニューラルネットワーク活性化関数は、重み付けされた入力の合計に応じて、ニューロンの出力値を決定します。活性化関数の選択は、ニューラルネットワークのパフォーマンスに大きな影響を与えます。異なるモデルパーツ(レイヤー)は、異なるアクティベーション関数を使用できます。

すべての既知の活性化関数に加えて、MQL5はそれらの導関数も提供します。関数導関数により、学習で受け取ったエラーに基づいてモデルパラメータを効率的に更新できます。

ニューラルネットワークは、損失関数が使用される学習のエラーを最小化するアルゴリズムを見つけることを目的としています。損失関数の値は、モデルによって予測された値が実際の値からどれだけずれているかを示します。問題に応じて異なる損失関数が使用されます。たとえば、平均二乗誤差(MSE)は回帰問題に使用され、バイナリクロスエントロピー(BCE)は二項分類に使用されます。

関数	アクション
Activation	活性化関数の値を計算し、渡されたベクトル/行列に書き込みます
Derivative	活性化関数の導関数値を計算し、渡されたベクトル/行列に書き込みます
Loss	損失関数値を計算し、渡されたベクトル/行列に書き込みます
RegressionMetric	指定されたデータ配列で構築された回帰直線からの偏差エラーとして回帰指標を計算します
ConfusionMatrix	混同行列を計算します。このメソッドは、予測値のベクトルに適用されます
ConfusionMatrixMultilabel	各ラベルの混同行列を計算します。このメソッドは、予測値のベクトルに適用されます
ClassificationMetric	分類指標を計算して、実際のデータと比較して予測データの品質を評価します。このメソッドは、予測値のベクトルに適用されます
ClassificationScore	分類指標を計算して、実際のデータと比較して予測データの品質を評価します
PrecisionRecall	値を計算して適合率–再現率曲線を構築します。ClassificationScoreと同様、このメソッドは真の値のベクトルに適用されます
ReceiverOperatingCharacteristic	値を計算して受信者操作特性(ROC)曲線を構築します。ClassificationScoreと同様、このメソッドは真の値のベクトルに適用されます

例

この例は、行列演算を使用したモデルのトレーニングを示しています。モデルは、関数 (a + b + c)^2 / (a^2 + b^2 + c^2) についてトレーニングされます。a、b、c が異なる列に含まれる初期データ行列を入力します。関数の結果は、モデルの出力で取得されます。

© 2000–2025, MetaQuotes Ltd.

```
matrix weights1, weights2, weights3;                // 重みの行列
matrix output1, output2, result;                    // ニューラル層出力の行列
input int layer1 = 200;                             // 最初の隠れ層のサイズ
input int layer2 = 200;                             // 2番目の隠れ層のサイズ
input int Epochs = 20000;                           // トレーニングエポック数
input double lr = 3e-6;                             // 学習率
input ENUM_ACTIVATION_FUNCTION ac_func = AF_SWISH; // 活性化関数
//+------------------------------------------------------------------+
//|  スクリプトを開始する関数                                          |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int train = 1000;     // トレーニングサンプルサイズ
   int test = 10;        // テストサンプルサイズ
   matrix m_data, m_target;
//--- トレーニングサンプルを生成する
   if(!CreateData(m_data, m_target, train))
      return;
//--- モデルをトレーニングする
   if(!Train(m_data, m_target, Epochs))
      return;
//--- テストサンプルを生成する
   if(!CreateData(m_data, m_target, test))
      return;
//--- モデルをテストする
   Test(m_data, m_target);
  }
//+------------------------------------------------------------------+
//|  サンプル生成メソッド                                              |
//+------------------------------------------------------------------+
bool CreateData(matrix &data, matrix &target, const int count)
  {
//--- 初期データと結果行列を初期化する
   if(!data.Init(count, 3) || !target.Init(count, 1))
      return false;
//--- 初期データ行列をランダムな値で埋める
   data.Random(-10, 10);
//--- トレーニングサンプルの目標値を計算する
   vector X1 = MathPow(data.Col(0) + data.Col(1) + data.Col(1), 2);
   vector X2 = MathPow(data.Col(0), 2) + MathPow(data.Col(1), 2) +
MathPow(data.Col(2), 2);
   if(!target.Col(X1 / X2, 0))
      return false;
//--- 結果を返す
   return true;
  }
//+------------------------------------------------------------------+
//|  モデルトレーニングメソッド                                        |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool Train(matrix &data, matrix &target, const int epochs = 10000)
  {
//--- モデルを作成する
   if(!CreateNet())
      return false;
//--- モデルをトレーニングする
   for(int ep = 0; ep < epochs; ep++)
     {
      //--- フィードフォワードパス
      if(!FeedForward(data))
         return false;
      PrintFormat("Epoch %d, loss %.5f", ep, result.Loss(target, LOSS_MSE));
      //--- 重み行列の逆伝播と更新
      if(!Backprop(data, target))
         return false;
     }
//--- 結果を返す
   return true;
  }
//+------------------------------------------------------------------+
//| モデル作成メソッド                                               |
//+------------------------------------------------------------------+
bool CreateNet()
  {
//--- 重み行列を初期化する
   if(!weights1.Init(4, layer1) || !weights2.Init(layer1 + 1, layer2) ||
!weights3.Init(layer2 + 1, 1))
      return false;
//--- 重み行列にランダムな値を書き込む
   weights1.Random(-0.1, 0.1);
   weights2.Random(-0.1, 0.1);
   weights3.Random(-0.1, 0.1);
//--- 結果を返す
   return true;
  }
//+------------------------------------------------------------------+
//| フィードフォワードメソッド                                       |
//+------------------------------------------------------------------+
bool FeedForward(matrix &data)
  {
//--- 初期データサイズを確認する
   if(data.Cols() != weights1.Rows() - 1)
      return false;
//--- 1番目のニューラル層を計算する
   matrix temp = data;
   if(!temp.Resize(temp.Rows(), weights1.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights1.Rows() - 1))
      return false;
```

© 2000-2025, MetaQuotes Ltd.

```
   output1 = temp.MatMul(weights1);
//--- 活性化関数を計算する
   if(!output1.Activation(temp, ac_func))
      return false;
//--- 2番目のニューラル層を計算する
   if(!temp.Resize(temp.Rows(), weights2.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights2.Rows() - 1))
      return false;
   output2 = temp.MatMul(weights2);
//--- 活性化関数を計算する
   if(!output2.Activation(temp, ac_func))
      return false;
//--- 3番目のニューラル層を計算する
   if(!temp.Resize(temp.Rows(), weights3.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights3.Rows() - 1))
      return false;
   result = temp.MatMul(weights3);
//--- 結果を返す
   return true;
  }
//+------------------------------------------------------------------+
//|  逆伝播メソッド                                                  |
//+------------------------------------------------------------------+
bool Backprop(matrix &data, matrix &target)
  {
//--- ターゲット値の行列のサイズを確認する
   if(target.Rows() != result.Rows() ||
      target.Cols() != result.Cols())
      return false;
//--- 目標値からの計算値の偏差を決定する
   matrix loss = (target - result) * 2;
//--- 勾配を前の層に伝播する
   matrix gradient = loss.MatMul(weights3.Transpose());
//--- 最後の層の重み行列を更新する
   matrix temp;
   if(!output2.Activation(temp, ac_func))
      return false;
   if(!temp.Resize(temp.Rows(), weights3.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights3.Rows() - 1))
      return false;
   weights3 = weights3 + temp.Transpose().MatMul(loss) * lr;
//--- 活性化関数の導関数によって誤差勾配を調整する
   if(!output2.Derivative(temp, ac_func))
      return false;
   if(!gradient.Resize(gradient.Rows(), gradient.Cols() - 1))
      return false;
   loss = gradient * temp;
//--- 勾配をより低い層に伝播する
   gradient = loss.MatMul(weights2.Transpose());
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 2番目の隠れ層の重み行列を更新する
   if(!output1.Activation(temp, ac_func))
      return false;
   if(!temp.Resize(temp.Rows(), weights2.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights2.Rows() - 1))
      return false;
   weights2 = weights2 + temp.Transpose().MatMul(loss) * lr;
//--- 活性化関数の導関数によって誤差勾配を調整する
   if(!output1.Derivative(temp, ac_func))
      return false;
   if(!gradient.Resize(gradient.Rows(), gradient.Cols() - 1))
      return false;
   loss = gradient * temp;
//--- 1番目の隠れ層の重み行列を更新する
   temp = data;
   if(!temp.Resize(temp.Rows(), weights1.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights1.Rows() - 1))
      return false;
   weights1 = weights1 + temp.Transpose().MatMul(loss) * lr;
//--- 結果を返す
   return true;
  }
//+------------------------------------------------------------------+
//| モデルテストメソッド                                             |
//+------------------------------------------------------------------+
bool Test(matrix &data, matrix &target)
  {
//--- テストデータのフィードフォワード
   if(!FeedForward(data))
      return false;
//--- モデルの計算結果と真の値を記録する
   PrintFormat("Test loss %.5f", result.Loss(target, LOSS_MSE));
   ulong total = data.Rows();
   for(ulong i = 0; i < total; i++)
      PrintFormat("(%.2f + %.2f + %.2f)^2 / (%.2f^2 + %.2f^2 + %.2f^2) =  Net %.2f,
Target %.2f", data[i, 0], data[i, 1], data[i, 2],
                  data[i, 0], data[i, 1], data[i, 2], result[i, 0], target[i, 0]);
//--- 結果を返す
   return true;
  }
//+------------------------------------------------------------------+
```

© 2000–2025, MetaQuotes Ltd.

Activation

活性化関数の値を計算し、渡されたベクトル/行列に書き込みます。

```
bool vector::Activation(
  vector&                    vect_out,      // 値を取得するベクトル
  ENUM_ACTIVATION_FUNCTION   activation,    // 活性化関数
   ...                                      // 追加のパラメータ
   );


bool matrix::Activation(
  matrix&                    matrix_out,    // 値を取得する行列
  ENUM_ACTIVATION_FUNCTION   activation      // 活性化関数
   );


bool matrix::Activation(
  matrix&                    matrix_out,    // 値を取得する行列
  ENUM_ACTIVATION_FUNCTION   activation,    // 活性化関数
  ENUM_MATRIX_AXIS           axis,          // 軸
   ...                                      // 追加のパラメータ
   );
```

パラメータ

vect_out/matrix_out

[out] 活性化関数の計算値を取得するためのベクトルまたは行列

activation

[in] ENUM_ACTIVATION_FUNCTION列挙体からの活性化関数

axis

[in] ENUM_MATRIX_AXIS列挙体から野値（AXIS_HORZ — 水平軸、AXIS_VERT — 垂直軸）

...

[in] 一部の活性化関数に必要な追加パラメータ。パラメータの指定がない場合は初期値が使用されます。

戻り値

成功の場合はtrue、それ以外の場合はfalse

追加のパラメータ

一部の活性化関数は追加パラメータを受け入れます。パラメータの指定がない場合は初期値が使用されます。

© 2000–2025, MetaQuotes Ltd.

```
AF_ELU  (Exponential Linear Unit)
  double alpha=1.0

活性化関数:if(x>=0) f(x) = x
                   else f(x) = alpha * (exp(x)-1)


AF_LINEAR
  double alpha=1.0
  double beta=0.0

活性化関数:f(x) = alpha*x + beta


AF_LRELU   (Leaky REctified Linear Unit)
  double alpha=0.3

活性化関数:if(x>=0) f(x)=x
                   else f(x) = alpha*x


AF_RELU  (REctified Linear Unit)
  double alpha=0.0
  double max_value=0.0
  double treshold=0.0

活性化関数:if(alpha==0) f(x) = max(x,0)
                   else if(x>max_value) f(x) = x
                   else f(x) = alpha*(x - treshold)


AF_SWISH
  double beta=1.0

活性化関数:f(x) = x / (1+exp(-x*beta))


AF_TRELU   (Thresholded REctified Linear Unit)
  double theta=1.0

活性化関数:if(x>theta) f(x) = x
                   else f(x) = 0


AF_PRELU   (Parametric REctified Linear Unit)
  double alpha[] - learned array of coeefficients

活性化関数:if(x[i]>=0) f(x)[i] = x[i]
                   else f(x)[i] = alpha[i] * x[i]
```

© 2000–2025, MetaQuotes Ltd.

注意事項

人工ニューラル　ネットワークでは、ニューロンの活性化関数によって出力信号が決定され、出力信号は入力信号または入力信号のセットによって定義されます。活性化関数の選択は、ニューラルネットワークのパフォーマンスに大きな影響を与えます。異なるモデルパーツ（層）では、異なる活性化関数を使用することができます。

追加のパラメータの使用例：

```
vector x={0.1, 0.4, 0.9, 2.0, -5.0, 0.0, -0.1};
vector y;

x.Activation(y,AF_ELU);
Print(y);
x.Activation(y,AF_ELU,2.0);
Print(y);

Print("");
x.Activation(y,AF_LINEAR);
Print(y);
x.Activation(y,AF_LINEAR,2.0);
Print(y);
x.Activation(y,AF_LINEAR,2.0,5.0);
Print(y);

Print("");
x.Activation(y,AF_LRELU);
Print(y);
x.Activation(y,AF_LRELU,1.0);
Print(y);
x.Activation(y,AF_LRELU,0.1);
Print(y);

Print("");
x.Activation(y,AF_RELU);
Print(y);
x.Activation(y,AF_RELU,2.0,0.5);
Print(y);
x.Activation(y,AF_RELU,2.0,0.5,1.0);
Print(y);

Print("");
x.Activation(y,AF_SWISH);
Print(y);
x.Activation(y,AF_SWISH,2.0);
```

© 2000–2025, MetaQuotes Ltd.

```
   Print(y);

   Print("");
   x.Activation(y,AF_TRELU);
   Print(y);
   x.Activation(y,AF_TRELU,0.3);
   Print(y);

   Print("");
   vector a=vector::Full(x.Size(),2.0);
   x.Activation(y,AF_PRELU,a);
   Print(y);

/* 結果
   [0.1,0.4,0.9,2,-0.993262053000915,0,-0.095162581964040]
   [0.1,0.4,0.9,2,-1.986524106001829,0,-0.190325163928081]

   [0.1,0.4,0.9,2,-5,0,-0.1]
   [0.2,0.8,1.8,4,-10,0,-0.2]
   [5.2,5.8,6.8,9,-5,5,4.8]

   [0.1,0.4,0.9,2,-1.5,0,-0.03]
   [0.1,0.4,0.9,2,-5,0,-0.1]
   [0.1,0.4,0.9,2,-0.5,0,-0.01]

   [0.1,0.4,0.9,2,0,0,0]
   [0.2,0.8,0.9,2,-10,0,-0.2]
   [-1.8,-1.2,0.9,2,-12,-2,-2.2]

   [0.052497918747894,0.239475064044981,0.6398545523625035,1.761594155955765,-
0.03346425462142428,0,-0.047502081252106]
   [0.054983399731247,0.275989792451045,0.7723340415895611,1.964027580075817,-
0.00022698934351217,0,-0.045016600268752]

   [0,0,0,2,0,0,0]
   [0,0.4,0.9,2,0,0,0]

   [0.1,0.4,0.9,2,-10,0,-0.2]
*/
```

© 2000–2025, MetaQuotes Ltd.

Derivative

活性化関数の導関数値を計算し、渡されたベクトル/行列に書き込みます

```
bool vector::Derivative(
  vector&                   vect_out,      // 値を取得するベクトル
  ENUM_ACTIVATION_FUNCTION  activation,    // 活性化関数
   ...                                     // 追加のパラメータ
   );


bool matrix::Derivative(
  matrix&                   matrix_out,    // 値を取得する行列
  ENUM_ACTIVATION_FUNCTION  activation,    // 活性化関数
   );


bool matrix::Derivative(
  matrix&                   matrix_out,    // 値を取得する行列
  ENUM_ACTIVATION_FUNCTION  activation,    // 活性化関数
  ENUM_MATRIX_AXIS          axis,          // 軸
   ...                                     // 追加のパラメータ
   );
```

パラメータ

vect_out/matrix_out

[out] 活性化関数の導関数の計算値を取得するためのベクトルまたは行列

activation

[in] ENUM_ACTIVATION_FUNCTION列挙体からの活性化関数

axis

[in] ENUM_MATRIX_AXIS列挙体から野値（AXIS_HORZ — 水平軸、AXIS_VERT — 垂直軸）

...

[in] 追加のパラメータは活性化関数のパラメータと同じです。一部の活性化関数は追加パラメータを受け入れます。パラメータの指定がない場合は初期値が使用されます。

戻り値

成功の場合はtrue、それ以外の場合はfalse

注意事項

関数導関数により、誤差逆伝播中の学習で受け取った誤差に基づいてモデルパラメータを効率的に更新できます。

© 2000–2025, MetaQuotes Ltd.

© 2000–2025, MetaQuotes Ltd.

Loss

損失関数の値を計算します。

```
double vector::Loss(
  const vector&        vect_true,     // 真の値のベクトル
  ENUM_LOSS_FUNCTION   loss,          // 損失関数
   ...                                // 追加のパラメータ
   );


double matrix::Loss(
  const matrix&        matrix_true,   // 真の値の行列
  ENUM_LOSS_FUNCTION   loss,          // 損失関数
   );


double matrix::Loss(
  const matrix&        matrix_true,   // 真の値の行列
  ENUM_LOSS_FUNCTION   loss,          // 損失関数
  ENUM_MATRIX_AXIS     axis,          // 軸
   ...                                // 追加のパラメータ
   );
```

パラメータ

vect_true/matrix_true

[in] 真の値のベクトルまたは行列

loss

[in] ENUM_LOSS_FUNCTION列挙体からの損失関数

axis

[in] ENUM_MATRIX_AXIS列挙体から野値（AXIS_HORZ — 水平軸、AXIS_VERT — 垂直軸）

...

[in] 追加パラメータ「delta」は、Huber損失関数（LOSS_HUBER）でのみ使用できます。

戻り値

double値

Huber損失関数（LOSS_HUBER）での「delta」パラメーターの使用方法

```
   double delta = 1.0;
   double error = fabs(y - x);
   if(error<delta)
      loss = 0.5 * error^2;
   else
      loss = 0.5 * delta^2 + delta * (error - delta);
```

© 2000–2025, MetaQuotes Ltd.

注意事項

ニューラルネットワークは、損失関数が使用される訓練サンプルの誤差を最小限に抑えるアルゴリズムを見つけることを目的としています。

損失関数の値は、モデルによって予測された値が実際の値からどれだけずれているかを示します。

問題に応じて異なる損失関数が使用されます。たとえば、平均二乗誤差(MSE)は回帰問題に使用され、バイナリ交差エントロピー(BCE) は二項分類に使用されます。

Huber損失関数呼び出しの例

```
   vector y_true = {0.0, 1.0, 0.0, 0.0};
   vector y_pred = {0.6, 0.4, 0.4, 0.6};
   double loss=y_pred.Loss(y_true,LOSS_HUBER);
   Print(loss);
   double loss2=y_pred.Loss(y_true,LOSS_HUBER,0.5);
   Print(loss2);

/* 結果
   0.155
   0.15125
*/
```

© 2000–2025, MetaQuotes Ltd.

LossGradient

損失関数勾配のベクトルまたは行列を計算します。

```
vector vector::LossGradient(
  const vector&        vect_true,      // 真の値のベクトル
  ENUM_LOSS_FUNCTION   loss,           // 損失関数型
   ...                                 // 追加のパラメータ
   );


matrix matrix::LossGradient(
  const matrix&        matrix_true,    // 真の値の行列
  ENUM_LOSS_FUNCTION   loss,           // 損失関数
   );


matrix matrix::LossGradient(
  const matrix&        matrix_true,    // 真の値の行列
  ENUM_LOSS_FUNCTION   loss,           // 損失関数
  ENUM_MATRIX_AXIS     axis,           // 軸
   ...                                 // 追加のパラメータ
   );
```

パラメータ

vect_true/matrix_true

[in] 真の値のベクトルまたは行列

loss

[in] ENUM_LOSS_FUNCTION列挙体からの損失関数

axis

[in] ENUM_MATRIX_AXIS列挙体から野値（AXIS_HORZ ― 水平軸、AXIS_VERT ― 垂直軸）

...

[in] 追加パラメータ「delta」は、Huber損失関数（LOSS_HUBER）でのみ使用できます。

戻り値

損失関数勾配値のベクトルまたは行列。勾配は、特定の点における損失関数のdx（xは予測値））に関する偏導関数です。

注意事項

勾配は、モデルの訓練時に逆伝播中に重み行列の重みを調整するためにニューラル　ネットワークで使用されます。

ニューラルネットワークは、損失関数が使用される訓練サンプルの誤差を最小限に抑えるアルゴリズムを見つけることを目的としています。

© 2000–2025, MetaQuotes Ltd.

問題に応じて異なる損失関数が使用されます。たとえば、平均二乗誤差(MSE)は回帰問題に使用され、バイナリ交差エントロピー(BCE) は二項分類に使用されます。

損失関数勾配の計算例

```
   matrixf y_true={{ 1, 2, 3, 4 },
                   { 5, 6, 7, 8 },
                   { 9,10,11,12 }};
   matrixf y_pred={{ 1, 2, 3, 4 },
                   {11,10, 9, 8 },
                   { 5, 6, 7,12 }};
   matrixf loss_gradient =y_pred.LossGradient(y_true,LOSS_MAE);
   matrixf loss_gradienth=y_pred.LossGradient(y_true,LOSS_MAE,AXIS_HORZ);
   matrixf loss_gradientv=y_pred.LossGradient(y_true,LOSS_MAE,AXIS_VERT);
   Print("loss gradients\n",loss_gradient);
   Print("loss gradients on horizontal axis\n",loss_gradienth);
   Print("loss gradients on vertical axis\n",loss_gradientv);

/* 結果
   loss gradients
   [[0,0,0,0]
    [0.083333336,0.083333336,0.083333336,0]
    [-0.083333336,-0.083333336,-0.083333336,0]]
   loss gradients on horizontal axis
   [[0,0,0,0]
    [0.33333334,0.33333334,0.33333334,0]
    [-0.33333334,-0.33333334,-0.33333334,0]]
   loss gradients on vertical axis
   [[0,0,0,0]
    [0.25,0.25,0.25,0]
    [-0.25,-0.25,-0.25,0]]
*/
```

© 2000–2025, MetaQuotes Ltd.

RegressionMetric

回帰指標を計算して、実際のデータと比較して予測データの品質を評価します

```
double vector::RegressionMetric(
   const vector&           vector_true,   // 真の値のベクトル
   ENUM_REGRESSION_METRIC  metric         // 指標型
   );

double matrix::RegressionMetric(
   const matrix&           matrix_true,   // 真の値の行列
   ENUM_REGRESSION_METRIC  metric         // 指標型
);

vector matrix::RegressionMetric(
   const matrix&           matrix_true,   // 真の値の行列
   ENUM_REGRESSION_METRIC  metric,        // 指標型
   int                     axis           // 軸
   );
```

パラメータ

vector_true/matrix_true

[in] 真の値のベクトルまたは行列

metric

[in] ENUM_REGRESSION_METRIC列挙体からの指標型

axis

[in] 軸。0 — 水平軸、1 — 垂直軸

戻り値

真のデータと比較して予測データの品質を評価する計算された指標

注意事項

- REGRESSION_MAE — 予測値と対応する真の値との間の絶対差を表す平均絶対誤差
- REGRESSION_MSE — 予測値と対応する真の値の間の二乗差を表す平均二乗誤差
- REGRESSION_RMSE — MSEの平方根
- REGRESSION_R2 - 1 — MSE(regression) / MSE(mean)
- REGRESSION_MAPE — 百分率としてのMAE
- REGRESSION_MSPE — 百分率としてのMSE
- REGRESSION_RMSLE — 対数スケールで計算されたRMSE

例：

```
   vector y_true = {3, -0.5, 2, 7};
   vector y_pred = {2.5, 0.0, 2, 8};
```

© 2000–2025, MetaQuotes Ltd.

```
//---
   double mse=y_pred.RegressionMetric(y_true,REGRESSION_MSE);
   Print("mse=",mse);
//---
   double mae=y_pred.RegressionMetric(y_true,REGRESSION_MAE);
   Print("mae=",mae);
//---
   double r2=y_pred.RegressionMetric(y_true,REGRESSION_R2);
   Print("r2=",r2);

  /* 結果
   mae=0.375
   mse=0.5
   r2=0.9486081370449679
   */
```

© 2000–2025, MetaQuotes Ltd.

ConfusionMatrix

混同行列を計算します。このメソッドは、予測値のベクトルに適用されます。

```
matrix vector::ConfusionMatrix(
   const vector&        vect_true       // 真の値のベクトル
   );


matrix vector::ConfusionMatrix(
   const vector&        vect_true,      // 真の値のベクトル
   uint                 label           // ラベル値
   );
```

パラメータ

vect_true

[in] 真の値のベクトル

label

[in] 混同行列を計算するためのラベル値

戻り値

混同行列。ラベル値が指定されていない場合は、各ラベルが他のラベルと個別に照合される、マルチクラス混同行列が返されます。ラベル値が指定されている場合は、2 x 2行列が返されます。この行列では、指定されたラベルは正とみなされますが、他のすべてのラベルは負と見なされます(ovr、1 対残り)。

注意事項

混同行列Cは、Cijがグループiに含まれることがわかっており、グループjに含まれると予測される観測値の数に等しくなるような行列です。したがって、二項分類では、真陰性(TN)の数はC00、偽陰性(FN)はC10、真陽性(TP)はC11、偽陽性(FP)はC01になります。

つまり、行列は次のようにグラフで表すことができます。

TN	FP
FN	TP

真の値のベクトルと予測値のベクトルのサイズは同じである必要があります。

例：

```
   vector y_true={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,8,4,2,7,6,8,4,2,3,6};
   vector y_pred={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,2,9,4,9,5,9,2,7,7,0};
   matrix confusion=y_pred.ConfusionMatrix(y_true);
   Print(confusion);
```

© 2000-2025, MetaQuotes Ltd.

```
   confusion=y_pred.ConfusionMatrix(y_true,0);
   Print(confusion);
   confusion=y_pred.ConfusionMatrix(y_true,1);
   Print(confusion);
   confusion=y_pred.ConfusionMatrix(y_true,2);
   Print(confusion);


/*
  [[3,0,0,0,0,0,0,0,0,0]
   [0,3,0,0,0,0,0,0,0,0]
   [0,0,1,0,1,0,0,1,0,0]
   [0,0,0,1,0,0,0,1,0,0]
   [0,0,1,0,3,0,0,0,0,1]
   [0,0,0,0,0,2,0,0,0,0]
   [1,0,0,0,0,1,1,0,0,0]
   [0,0,0,0,0,0,0,2,0,1]
   [0,0,1,0,0,0,0,0,0,1]
   [0,0,0,0,0,0,0,0,0,4]]
  [[26,1]
   [0,3]]
  [[27,0]
   [0,3]]
  [[25,2]
   [2,1]]
*/
```

© 2000–2025, MetaQuotes Ltd.

ConfusionMatrixMultilabel

各ラベルの混同行列を計算します。このメソッドは、予測値のベクトルに適用されます。

```
uint vector::ConfusionMatrixMultiLabel(
   const vector&        vect_true,     // 真の値のベクトル
   matrix&              confusions[]   // 計算された混同行列の配列
   );
```

パラメータ

vect_true

[in] 真の値のベクトル

confusions

[out] 各ラベルの計算された混同行列を含む2 x 2行列の配列

戻り値

計算された混同行列の配列のサイズ。失敗した場合は0を返します。

注意事項

結果の配列は動的または静的になります。配列が静的である場合、そのサイズはクラスの数以上でなければなりません。

真の値のベクトルと予測値のベクトルのサイズは同じである必要があります。

例：

```
   vector y_true={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,8,4,2,7,6,8,4,2,3,6};
   vector y_pred={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,2,9,4,9,5,9,2,7,7,0};
   matrix label_confusions[12];

   uint   res=y_pred.ConfusionMatrixMultiLabel(y_true,label_confusions);
   Print("res=",res,"  size=",label_confusions.Size());
   for(uint i=0; i<res; i++)
      Print(label_confusions[i]);


/*
  res=10  size=12
  [[26,1]
   [0,3]]
  [[27,0]
   [0,3]]
  [[25,2]
   [2,1]]
```

© 2000–2025, MetaQuotes Ltd.

```
   [[28,0]
    [1,1]]
   [[24,1]
    [2,3]]
   [[27,1]
    [0,2]]
   [[27,0]
    [2,1]]
   [[25,2]
    [1,2]]
   [[28,0]
    [2,0]]
   [[23,3]
    [0,4]]
*/
```

© 2000–2025, MetaQuotes Ltd.

ClassificationMetric

分類指標を計算して、実際のデータと比較して予測データの品質を評価します。このメソッドは、予測値のベクトルに適用されます。

```
vector vector::ClassificationMetric(
   const vector&                vect_true,      // 真の値のベクトル
   ENUM_CLASSIFICATION_METRIC metric          // 指標タイプ
   );


vector vector::ClassificationMetric(
   const vector&                vect_true,      // 真の値のベクトル
   ENUM_CLASSIFICATION_METRIC metric          // 指標タイプ
   ENUM_AVERAGE_MODE          mode            // 平均化モード
   );
```

パラメータ

vect_true

[in] 真の値のベクトル

metric

[in] ENUM_CLASSIFICATION_METRIC列挙体からの指標型CLASSIFICATION_TOP_K_ACCURACY、CLASSIFICATION_AVERAGE_PRECISION、CLASSIFICATION_ROC_AUC（ClassificationScoreメソッドで使用）以外の値が適用されます。

mode

[in] ENUM_AVERAGE_MODE列挙の平均化モード。CLASSIFICATION_F1、CLASSIFICATION_JACCARD、CLASSIFICATION_PRECISION、CLASSIFICATION_RECALL指標に使用されます。

戻り値

計算された指標を含むベクトル。AVERAGE_NONE平均化モードの場合、ベクトルには平均化を行わない各クラスの指標値が含まれます。（たとえば、二項分類の場合、それぞれ「false」と「true」の2つの指標）。

平均化モードに関する注意事項

AVERAGE_BINARY: 二項分類にのみ意味があります。

AVERAGE_MICRO: 真陽性、偽陰性、偽陽性の合計をカウントすることにより、指標をグローバルに計算します。

AVERAGE_MACRO: 各ラベルの指標を計算し、加重されていない平均を見つけます。これにはラベルの不均衡は考慮されていません。

AVERAGE_WEIGHTED: 各ラベルの指標を計算し、サポートによって重み付けされた平均（各ラベルの真のインスタンスの数）を見つけます。これにより、ラベルの不均衡を考慮して「マクロ」が変更されます。精度と再現率の間にないF値が発生する可能性があります。

© 2000-2025, MetaQuotes Ltd.

注意事項

二項分類の場合、最初の列に負のラベルの確率が含まれ、2番目の列に正のラベルの確率が含まれるn x 2行列だけでなく、正の確率を含む1つの列で構成される行列も入力できます。これは、二項分類モデルが陽性ラベルに対して2つの確率または1つの確率を返すことができるためです。

例：

```
   vector y_true={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,8,4,2,7,6,8,4,2,3,6};
   vector y_pred={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,2,9,4,9,5,9,2,7,7,0};

   vector accuracy=y_pred.ClassificationMetric(y_true,CLASSIFICATION_ACCURACY);
   Print("accuracy=",accuracy);
   vector
balanced=y_pred.ClassificationMetric(y_true,CLASSIFICATION_BALANCED_ACCURACY);
   Print("balanced=",balanced);
   Print("");

   vector
f1_micro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_F1,AVERAGE_MICRO);
   Print("f1_micro=",f1_micro);
   vector
f1_macro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_F1,AVERAGE_MACRO);
   Print("f1_macro=",f1_macro);
   vector
f1_weighted=y_pred.ClassificationMetric(y_true,CLASSIFICATION_F1,AVERAGE_WEIGHTED);
   Print("f1_weighted=",f1_weighted);
   vector
f1_none=y_pred.ClassificationMetric(y_true,CLASSIFICATION_F1,AVERAGE_NONE);
   Print("f1_none=",f1_none);
   Print("");

   vector
jaccard_micro
=y_pred.ClassificationMetric(y_true,CLASSIFICATION_JACCARD,AVERAGE_MICRO);
   Print("jaccard_micro=",jaccard_micro);
   vector
jaccard_macro
=y_pred.ClassificationMetric(y_true,CLASSIFICATION_JACCARD,AVERAGE_MACRO);
   Print("jaccard_macro=",jaccard_macro);
   vector
jaccard_weighted
=y_pred.ClassificationMetric(y_true,CLASSIFICATION_JACCARD,AVERAGE_WEIGHTED);
   Print("jaccard_weighted=",jaccard_weighted);
   vector
jaccard_none
=y_pred.ClassificationMetric(y_true,CLASSIFICATION_JACCARD,AVERAGE_NONE);
   Print("jaccard_none=",jaccard_none);
   Print("");
```

© 2000-2025, MetaQuotes Ltd.

```
   vector
precision_micro
=y_pred.ClassificationMetric(y_true,CLASSIFICATION_PRECISION,AVERAGE_MICRO);
   Print("precision_micro=",precision_micro);
   vector
precision_macro
=y_pred.ClassificationMetric(y_true,CLASSIFICATION_PRECISION,AVERAGE_MACRO);
   Print("precision_macro=",precision_macro);
   vector
precision_weighted
=y_pred.ClassificationMetric(y_true,CLASSIFICATION_PRECISION,AVERAGE_WEIGHTED);
   Print("precision_weighted=",precision_weighted);
   vector
precision_none
=y_pred.ClassificationMetric(y_true,CLASSIFICATION_PRECISION,AVERAGE_NONE);
   Print("precision_none=",precision_none);
   Print("");


   vector
recall_micro
=y_pred.ClassificationMetric(y_true,CLASSIFICATION_RECALL,AVERAGE_MICRO);
   Print("recall_micro=",recall_micro);
   vector
recall_macro
=y_pred.ClassificationMetric(y_true,CLASSIFICATION_RECALL,AVERAGE_MACRO);
   Print("recall_macro=",recall_macro);
   vector
recall_weighted
=y_pred.ClassificationMetric(y_true,CLASSIFICATION_RECALL,AVERAGE_WEIGHTED);
   Print("recall_weighted=",recall_weighted);
   vector
recall_none=y_pred.ClassificationMetric(y_true,CLASSIFICATION_RECALL,AVERAGE_NONE);
   Print("recall_none=",recall_none);
   Print("");

//--- binary classification
   vector y_pred_bin={0,1,0,1,1,0,0,0,1};
   vector y_true_bin={1,0,0,0,1,0,1,1,1};


   vector
f1_bin=y_pred_bin.ClassificationMetric(y_true_bin,CLASSIFICATION_F1,AVERAGE_BINARY);
   Print("f1_bin=",f1_bin);
   vector
jaccard_bin
=y_pred_bin.ClassificationMetric(y_true_bin,CLASSIFICATION_JACCARD,AVERAGE_BINARY);
   Print("jaccard_bin=",jaccard_bin);
   vector
precision_bin
=y_pred_bin
.ClassificationMetric(y_true_bin,CLASSIFICATION_PRECISION,AVERAGE_BINARY);
   Print("precision_bin=",precision_bin);
```

© 2000–2025, MetaQuotes Ltd.

```
   vector
recall_bin
=y_pred_bin.ClassificationMetric(y_true_bin,CLASSIFICATION_RECALL,AVERAGE_BINARY);
   Print("recall_bin=",recall_bin);



/*
  accuracy=[0.6666666666666666]
  balanced=[0.6433333333333333]

  f1_micro=[0.6666666666666666]
  f1_macro=[0.6122510822510823]
  f1_weighted=[0.632049062049062]
  f1_none=[
0
.8571428571428571
,
1
,
0
.3333333333333333
,
0
.6666666666666666
,0.6666666666666665,0.8,0.5,0.5714285714285715,0,0.7272727272727273]

  jaccard_micro=[0.5]
  jaccard_macro=[0.4921428571428572]
  jaccard_weighted=[0.5056349206349205]
  jaccard_none=[
0.75,1,0.2,0.5,0.5,0.6666666666666666,0.3333333333333333,0.4,0,0.5714285714285714]

  precision_micro=[0.6666666666666666]
  precision_macro=[0.6571428571428571]
  precision_weighted=[0.6706349206349207]
  precision_none=[
0.75,1,0.3333333333333333,1,0.75,0.6666666666666666,1,0.5,0,0.5714285714285714]

  recall_micro=[0.6666666666666666]
  recall_macro=[0.6433333333333333]
  recall_weighted=[0.6666666666666666]
  recall_none=[
1,1,0.3333333333333333,0.5,0.6,1,0.3333333333333333,0.6666666666666666,0,1]

  f1_bin=[0.4444444444444445]
  jaccard_bin=[0.2857142857142857]
  precision_bin=[0.5]
  recall_bin=[0.4]
*/
```

© 2000–2025, MetaQuotes Ltd.

ClassificationScore

分類指標を計算して、実際のデータと比較して予測データの品質を評価します。

「機械学習」セクションにある他のメソッドとは異なり、このメソッドは予測値のベクトルではなく真の値のベクトルに適用されます。

```
vector vector::ClassificationScore(
   const matrix&               pred_scores,    // 各クラスの確率分布を含む行列
   ENUM_CLASSIFICATION_METRIC  metric          // 指標タイプ
   ENUM_AVERAGE_MODE           mode            // 平均化モード
   );


vector vector::ClassificationScore(
   const matrix&               pred_scores,    // 各クラスの確率分布を含む行列
   ENUM_CLASSIFICATION_METRIC  metric          // 指標タイプ
   int                         param           // 追加のパラメータ
   );
```

パラメータ

pred_scores

[in] 各クラスの確率を含む一連の水平ベクトルを含む行列。行列の行数は、真の値のベクトルのサイズに対応する必要があります。

metric

[in] ENUM_CLASSIFICATION_METRIC列挙体からの指標型CLASSIFICATION_TOP_K_ACCURACY、CLASSIFICATION_AVERAGE_PRECISION、CLASSIFICATION_ROC_AUCの値が使用されます。

mode

[in] ENUM_AVERAGE_MODE列挙の平均化モード。CLASSIFICATION_AVERAGE_PRECISIONおよびCLASSIFICATION_ROC_AUC指標に使用されます。

param

[in] CLASSIFICATION_TOP_K_ACCURACY指標の場合、平均化モードの代わりに整数のK値を指定する必要があります。

戻り値

計算された指標を含むベクトル。AVERAGE_NONE平均化モードの場合、ベクトルには平均化を行わない各クラスの指標値が含まれます。（たとえば、二項分類の場合、それぞれ「false」と「true」の2つの指標）。

平均化モードに関する注意事項

AVERAGE_BINARY: 二項分類にのみ意味があります。

AVERAGE_MICRO: 真陽性、偽陰性、偽陽性の合計をカウントすることにより、指標をグローバルに計算します。

© 2000–2025, MetaQuotes Ltd.

AVERAGE_MACRO: 各ラベルの指標を計算し、加重されていない平均を見つけます。これにはラベルの不均衡は考慮されていません。

AVERAGE_WEIGHTED: 各ラベルの指標を計算し、サポートによって重み付けされた平均（各ラベルの真のインスタンスの数）を見つけます。

注意事項

二項分類の場合、最初の列に負のラベルの確率が含まれ、2番目の列に正のラベルの確率が含まれるn x 2行列だけでなく、正の確率を含む1つの列で構成される行列も入力できます。これは、二項分類モデルが陽性ラベルに対して2つの確率または1つの確率を返すことができるためです。

例：

```
   vector y_true={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,8,4,2,7,6,8,4,2,3,6};
   //vector y_pred={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,2,9,4,9,5,9,2,7,7,0};


//--- label scores           0          1          2          3          4          5
     6          7          8          9     true pred
   matrix y_scores={{0.000109, 0.000186, 0.000449, 0.000052, 0.000002, 0.000022,
0.000005, 0.998059, 0.000010, 0.001104},  // 7     7
                    {0.000091, 0.081956, 0.916816, 0.001106, 0.000006, 0.000002,
0.000001, 0.000000, 0.000021, 0.000000},  // 2     2
                    {0.000108, 0.972863, 0.003600, 0.000021, 0.010479, 0.000015,
0.000131, 0.010385, 0.002339, 0.000060},  // 1     1
                    {0.925425, 0.000080, 0.002913, 0.000057, 0.000274, 0.000638,
0.063529, 0.000316, 0.000095, 0.006673},  // 0     0
                    {0.000060, 0.000126, 0.000006, 0.000000, 0.993513, 0.000000,
0.000003, 0.000222, 0.000001, 0.006069},  // 4     4
                    {0.000016, 0.982124, 0.000045, 0.000002, 0.008445, 0.000001,
0.000005, 0.009230, 0.000120, 0.000013},  // 1     1
                    {0.000000, 0.000040, 0.000001, 0.000000, 0.989395, 0.000167,
0.000004, 0.000070, 0.000177, 0.010146},  // 4     4
                    {0.000795, 0.002938, 0.023447, 0.007418, 0.021838, 0.002476,
0.000260, 0.047551, 0.000082, 0.893194},  // 9     9
                    {0.000091, 0.000226, 0.000038, 0.000007, 0.000048, 0.854910,
0.068644, 0.000080, 0.001097, 0.074860},  // 5     5
                    {0.000000, 0.000000, 0.000000, 0.000000, 0.003004, 0.000000,
0.000000, 0.000035, 0.000000, 0.996960},  // 9     9
                    {0.998856, 0.000009, 0.000976, 0.000002, 0.000000, 0.000013,
0.000131, 0.000006, 0.000000, 0.000007},  // 0     0
                    {0.000178, 0.000446, 0.000326, 0.000033, 0.000193, 0.000071,
0.998403, 0.000015, 0.000328, 0.000007},  // 6     6
                    {0.000005, 0.000016, 0.000153, 0.000045, 0.004110, 0.000012,
0.000015, 0.000031, 0.000076, 0.995537},  // 9     9
                    {0.994188, 0.000003, 0.002584, 0.000005, 0.000005, 0.000100,
0.000739, 0.001473, 0.000038, 0.000864},  // 0     0
                    {0.000173, 0.990569, 0.000792, 0.000040, 0.001798, 0.000035,
0.000114, 0.004750, 0.001716, 0.000013},  // 1     1
                    {0.000000, 0.000537, 0.000008, 0.005080, 0.000046, 0.992910,
0.000012, 0.000671, 0.000390, 0.000347},  // 5     5
                    {0.000127, 0.000003, 0.000003, 0.000000, 0.001583, 0.000000,
0.000002, 0.000555, 0.000016, 0.997712},  // 9     9
```

© 2000–2025, MetaQuotes Ltd.

```
                     {0.000001, 0.000012, 0.000072, 0.000020, 0.000000, 0.000000,
0.000000, 0.999868, 0.000000, 0.000026},  // 7    7
                     {0.000020, 0.000105, 0.001139, 0.901343, 0.002132, 0.083873,
0.000124, 0.000097, 0.010981, 0.000186},  // 3    3
                     {0.000002, 0.000048, 0.000019, 0.000000, 0.999347, 0.000002,
0.000040, 0.000051, 0.000000, 0.000489},  // 4    4
                     {0.000059, 0.001344, 0.612502, 0.002749, 0.000229, 0.000678,
0.000038, 0.001844, 0.379727, 0.000831},  // 8    2
                     {0.000586, 0.000740, 0.001625, 0.000007, 0.269341, 0.000076,
0.016417, 0.000199, 0.000107, 0.710902},  // 4    9
                     {0.009547, 0.018055, 0.283795, 0.071079, 0.426074, 0.082335,
0.036379, 0.021188, 0.003924, 0.047623},  // 2    4
                     {0.002506, 0.002545, 0.001148, 0.005659, 0.020416, 0.000112,
0.006092, 0.272536, 0.003148, 0.685839},  // 7    9
                     {0.001263, 0.001769, 0.000293, 0.000011, 0.000302, 0.881768,
0.112019, 0.000125, 0.002327, 0.000123},  // 6    5
                     {0.002904, 0.002909, 0.013421, 0.001461, 0.007519, 0.001251,
0.000555, 0.106219, 0.107125, 0.756637},  // 8    9
                     {0.000055, 0.001080, 0.893158, 0.000000, 0.104492, 0.000159,
0.001042, 0.000013, 0.000000, 0.000000},  // 4    2
                     {0.000344, 0.002693, 0.071184, 0.000262, 0.000001, 0.000003,
0.000032, 0.924362, 0.000714, 0.000404},  // 2    7
                     {0.001404, 0.009375, 0.002638, 0.229189, 0.000064, 0.000896,
0.007516, 0.743557, 0.004462, 0.000897},  // 3    7
                     {0.491140, 0.000125, 0.000024, 0.000302, 0.000038, 0.034947,
0.473161, 0.000170, 0.000028, 0.000066}}; // 6    0

   vector
top_k=y_true.ClassificationScore(y_scores,CLASSIFICATION_TOP_K_ACCURACY,1);
   Print("top 1 accuracy score = ",top_k);
   top_k=y_true.ClassificationScore(y_scores,CLASSIFICATION_TOP_K_ACCURACY,2);
   Print("top 2 accuracy score = ",top_k);
   vector y_true2={0, 1, 2, 2};
   matrix y_score2={{0.5, 0.2, 0.2},  // 0 is in top 2
                    {0.3, 0.4, 0.2},  // 1 is in top 2
                    {0.2, 0.4, 0.3},  // 2 is in top 2
                    {0.7, 0.2, 0.1}}; // 2 isn't in top 2
   top_k=y_true2.ClassificationScore(y_score2,CLASSIFICATION_TOP_K_ACCURACY,2);
   Print("top k = ",top_k);
   Print("");

   vector
ap_micro
=y_true
.ClassificationScore(y_scores,CLASSIFICATION_AVERAGE_PRECISION,AVERAGE_MICRO);
   Print("average precision score micro = ",ap_micro);
   vector
ap_macro
=y_true
.ClassificationScore(y_scores,CLASSIFICATION_AVERAGE_PRECISION,AVERAGE_MACRO);
   Print("average precision score macro = ",ap_macro);
```

© 2000–2025, MetaQuotes Ltd.

```
   vector
ap_weighted
=y_true
.ClassificationScore(y_scores,CLASSIFICATION_AVERAGE_PRECISION,AVERAGE_WEIGHTED);
   Print("average precision score weighted = ",ap_weighted);
   vector
ap_none
=y_true.ClassificationScore(y_scores,CLASSIFICATION_AVERAGE_PRECISION,AVERAGE_NONE);
   Print("average precision score none = ",ap_none);
   Print("");


   vector
area_micro
=y_true.ClassificationScore(y_scores,CLASSIFICATION_ROC_AUC,AVERAGE_MICRO);
   Print("roc auc score micro = ",area_micro);
   vector
area_macro
=y_true.ClassificationScore(y_scores,CLASSIFICATION_ROC_AUC,AVERAGE_MACRO);
   Print("roc auc score macro = ",area_macro);
   vector
area_weighted
=y_true.ClassificationScore(y_scores,CLASSIFICATION_ROC_AUC,AVERAGE_WEIGHTED);
   Print("roc auc score weighted = ",area_weighted);
   vector
area_none=y_true.ClassificationScore(y_scores,CLASSIFICATION_ROC_AUC,AVERAGE_NONE);
   Print("roc auc score none = ",area_none);
   Print("");

//--- binary classification
   vector y_pred_bin={0,1,0,1,1,0,0,0,1};
   vector y_true_bin={1,0,0,0,1,0,1,1,1};
   vector y_score_true={0.3,0.7,0.1,0.6,0.9,0.0,0.4,0.2,0.8};
   matrix y_score1_bin(y_score_true.Size(),1);
   y_score1_bin.Col(y_score_true,0);
   matrix y_scores_bin={{0.7, 0.3},
                        {0.3, 0.7},
                        {0.9, 0.1},
                        {0.4, 0.6},
                        {0.1, 0.9},
                        {1.0, 0.0},
                        {0.6, 0.4},
                        {0.8, 0.2},
                        {0.2, 0.8}};


   vector
ap
=y_true_bin
.ClassificationScore(y_scores_bin,CLASSIFICATION_AVERAGE_PRECISION,AVERAGE_BINARY);
   Print("average precision score binary = ",ap);
```

© 2000–2025, MetaQuotes Ltd.

```
   vector
ap2
=y_true_bin
.ClassificationScore(y_score1_bin,CLASSIFICATION_AVERAGE_PRECISION,AVERAGE_BINARY);
   Print("average precision score binary = ",ap2);
   vector
ap3
=y_true_bin
.ClassificationScore(y_scores_bin,CLASSIFICATION_AVERAGE_PRECISION,AVERAGE_NONE);
   Print("average precision score none = ",ap3);
   Print("");

   vector
area
=y_true_bin.ClassificationScore(y_scores_bin,CLASSIFICATION_ROC_AUC,AVERAGE_BINARY);
   Print("roc auc score binary = ",area);
   vector
area2
=y_true_bin.ClassificationScore(y_score1_bin,CLASSIFICATION_ROC_AUC,AVERAGE_BINARY);
   Print("roc auc score binary = ",area2);
   vector
area3
=y_true_bin.ClassificationScore(y_scores_bin,CLASSIFICATION_ROC_AUC,AVERAGE_NONE);
   Print("roc auc score none = ",area3);


/*
  top 1 accuracy score = [0.6666666666666666]
  top 2 accuracy score = [1]
  top k = [0.75]

  average precision score micro = [0.8513333333333333]
  average precision score macro = [0.9326666666666666]
  average precision score weighted = [0.9333333333333333]
  average precision score none =
[1,1,0.7,1,0.9266666666666666,0.8333333333333333,1,0.8666666666666667,1,1]

  roc auc score micro = [0.9839506172839506]
  roc auc score macro = [0.9892068783068803]
  roc auc score weighted = [0.9887354497354497]
  roc auc score none =
[1,1,0.9506172839506173,1,0.984,0.9821428571428571,1,0.9753086419753086,1,1]

  average precision score binary = [0.7961904761904761]
  average precision score binary = [0.7961904761904761]
  average precision score none = [0.7678571428571428,0.7961904761904761]

  roc auc score binary = [0.7]
  roc auc score binary = [0.7]
  roc auc score none = [0.7,0.7]
*/
```

© 2000–2025, MetaQuotes Ltd.

PrecisionRecall

値を計算して適合率-再現率曲線を構築します。ClassificationScoreと同様、このメソッドは真の値のベクトルに適用されます。

```
bool vector::PrecisionRecall(
   const matrix&                 pred_scores,    // 各クラスの確率分布を含む行列
   const ENUM_ENUM_AVERAGE_MODE  mode            // 平均化モード
   matrix&                       precision,      // 各閾値に対して計算された精度値
   matrix&                       recall,         // 各閾値に対して計算された再現率
   matrix&                       thresholds,     // 降順に並び替えられた閾値
   );
```

パラメータ

pred_scores

[in] 各クラスの確率を含む一連の水平ベクトルを含む行列。行列の行数は、真の値のベクトルのサイズに対応する必要があります。

mode

[in] ENUM_AVERAGE_MODE列挙の平均化モード。AVERAGE_NONE、AVERAGE_BINARY、AVERAGE_MICROのみが使用されます。

precision

[out] 計算された精度曲線値を含む行列。平均化が適用されない場合(AVERAGE_NONE)、行列の行数はモデル　クラスの数に対応します。列の数は、真の値のベクトルのサイズ(または確率分布行列pred_scoreの行数)に対応します。マイクロ平均化の場合、行列の行数は、重複を除いた閾値の合計数に対応します。

recall

[out] 計算された再現率曲線値を含む行列。

threshold

[out] 確率行列を並び替えて得られた閾値行列。

注意事項

ClassificationScoreメソッドの注意事項を参照してください。

例

mnist.onnxモデルから統計を収集する例(精度99%)。

```
//--- 分類メトリクスのデータ
   vectorf y_true(images);
   vectorf y_pred(images);
   matrixf y_scores(images,10);
//--- 入力-出力
   matrixf image(28,28);
```

© 2000-2025, MetaQuotes Ltd.

```
   vectorf result(10);

//--- テスト
   for(int test=0; test<images; test++)
     {
      image=test_data[test].image;
      if(!OnnxRun(model,ONNX_DEFAULT,image,result))
        {
         Print("OnnxRun error ",GetLastError());
         break;
        }
      result.Activation(result,AF_SOFTMAX);
     //--- データ収集
      y_true[test]=(float)test_data[test].label;
      y_pred[test]=(float)result.ArgMax();
      y_scores.Row(result,test);
     }    }
```

精度計算

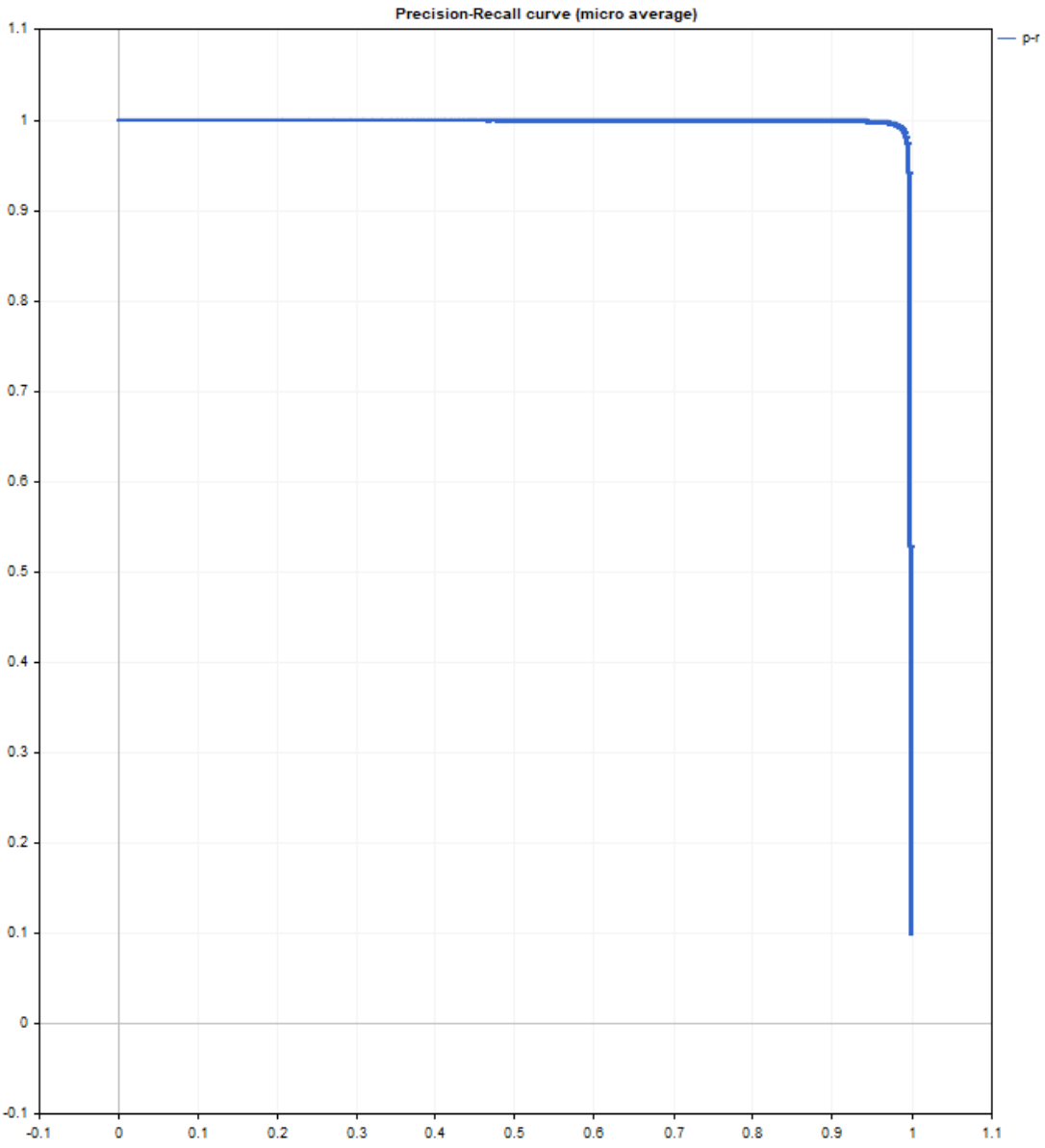
```
   vectorf accuracy=y_pred.ClassificationMetric(y_true,CLASSIFICATION_ACCURACY);
   PrintFormat("accuracy=%f",accuracy[0]);

accuracy=0.989000
```

適合率と再現率のグラフをプロットする例。適合率の値がY軸にプロットされ、再現率の値がX軸にプロットされます。また、精度と再現率のグラフは個別にプロットされ、閾値がX軸にプロットされます。

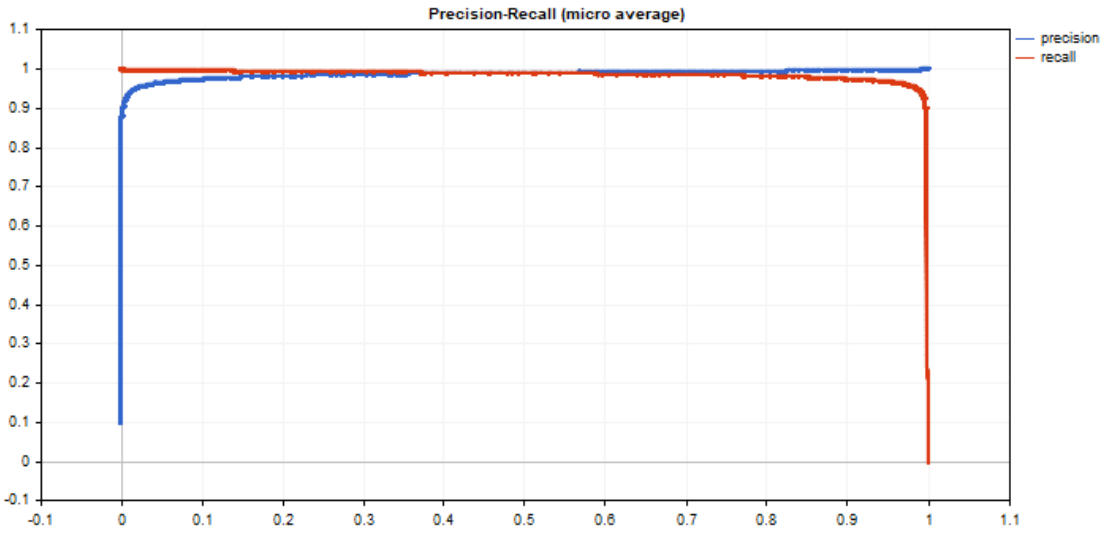
```
   if(y_true.PrecisionRecall(y_scores,AVERAGE_MICRO,mat_precision,mat_recall,mat_thr
es))
     {
      double precision[],recall[],thres[];
      ArrayResize(precision,mat_thres.Cols());
      ArrayResize(recall,mat_thres.Cols());
      ArrayResize(thres,mat_thres.Cols());

      for(uint i=0; i<thres.Size(); i++)
        {
         precision[i]=mat_precision[0][i];
         recall[i]=mat_recall[0][i];
         thres[i]=mat_thres[0][i];
        }
      thres[0]=thres[1]+0.001;

      PlotCurve("Precision-Recall curve (micro average)","p-r","",recall,precision);
      Plot2Curves("Precision-Recall (micro
average)","precision","recall",thres,precision,recall);
     }
```

© 2000-2025, MetaQuotes Ltd.

結果の曲線：



© 2000–2025, MetaQuotes Ltd.



© 2000–2025, MetaQuotes Ltd.

ReceiverOperatingCharacteristic

値を計算して受信者操作特性(ROC)曲線を構築します。ClassificationScoreと同様、このメソッドは真の値のベクトルに適用されます。

```
bool vector::ReceiverOperatingCharacteristic(
   const matrix&                 pred_scores,    // 各クラスの確率分布を含む行列
   const ENUM_ENUM_AVERAGE_MODE  mode            // 平均化モード
   matrix&                       fpr,            // 各閾値に対して計算された誤検知率の値
   matrix&                       tpr,            // 各閾値に対して計算された真陽性率
   matrix&                       thresholds,     // 降順に並び替えられた閾値
   );
```

パラメータ

pred_scores

[in] 各クラスの確率を含む一連の水平ベクトルを含む行列。行列の行数は、真の値のベクトルのサイズに対応する必要があります。

mode

[in] ENUM_AVERAGE_MODE列挙の平均化モード。AVERAGE_NONE、AVERAGE_BINARY、AVERAGE_MICROのみが使用されます。

fpr

[out] 偽陽性率曲線の計算値を含む行列。平均化が適用されない場合(AVERAGE_NONE)、行列の行数はモデル　クラスの数に対応します。列の数は、真の値のベクトルのサイズ(または確率分布行列pred_scoreの行数)に対応します。マイクロ平均化の場合、行列の行数は、重複を除いた閾値の合計数に対応します。

tpr

[out] 真陽性率曲線の計算値を含む行列。

threshold

[out] 確率行列を並び替えて得られた閾値行列。

注意事項

ClassificationScoreメソッドの注意事項を参照してください。

例

ROCグラフのプロットの例。tpr値がy軸にプロットされ、fpr値がx軸にプロットされます。また、fprグラフとtprグラフは個別にプロットされ、しきい値がX軸にプロットされます。

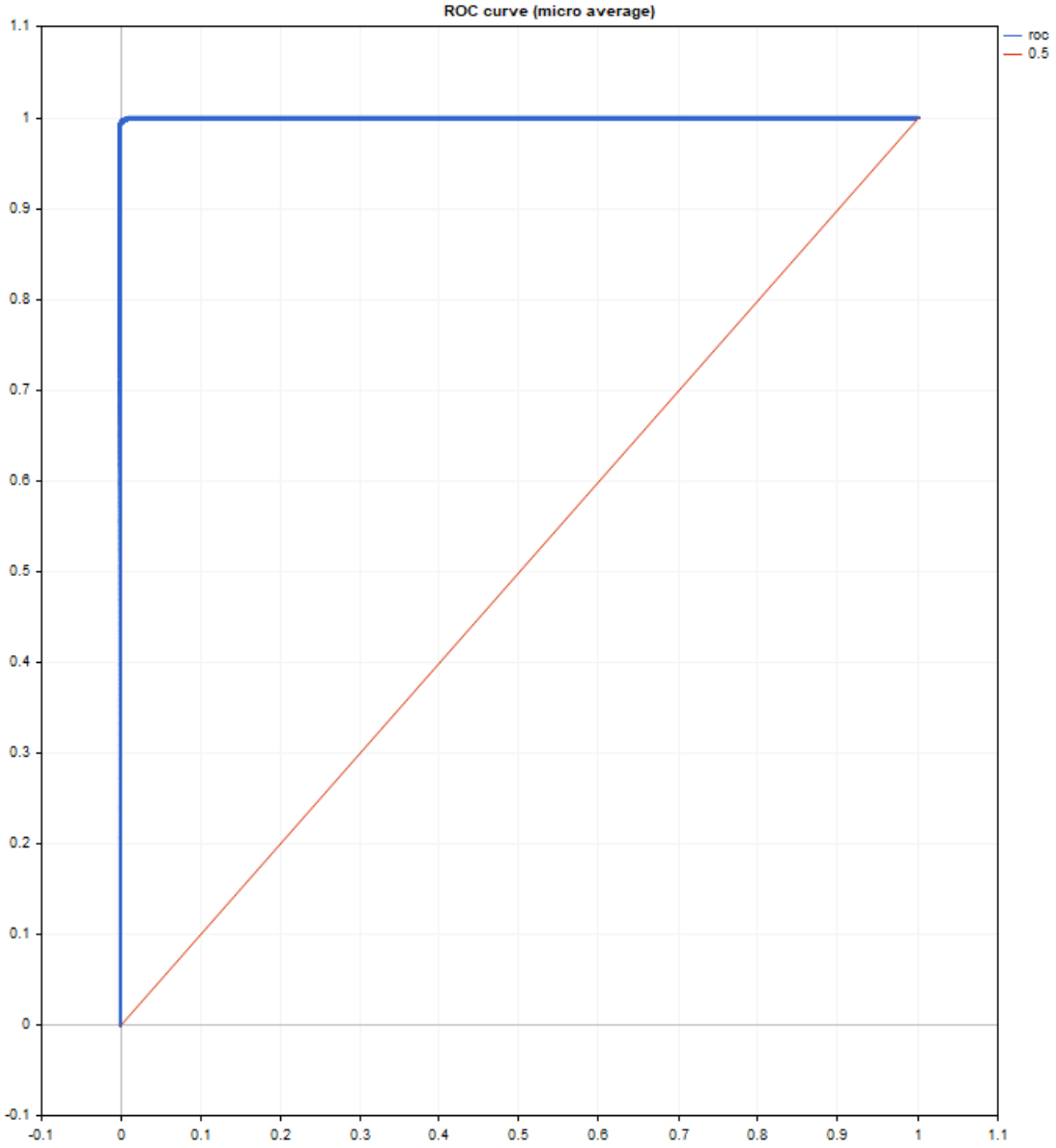
```
   matrixf mat_thres;
   matrixf mat_fpr;
   matrixf mat_tpr;

   if(y_true.ReceiverOperatingCharacteristic(y_scores,AVERAGE_MICRO,mat_fpr,mat_tpr,
mat_thres))
```

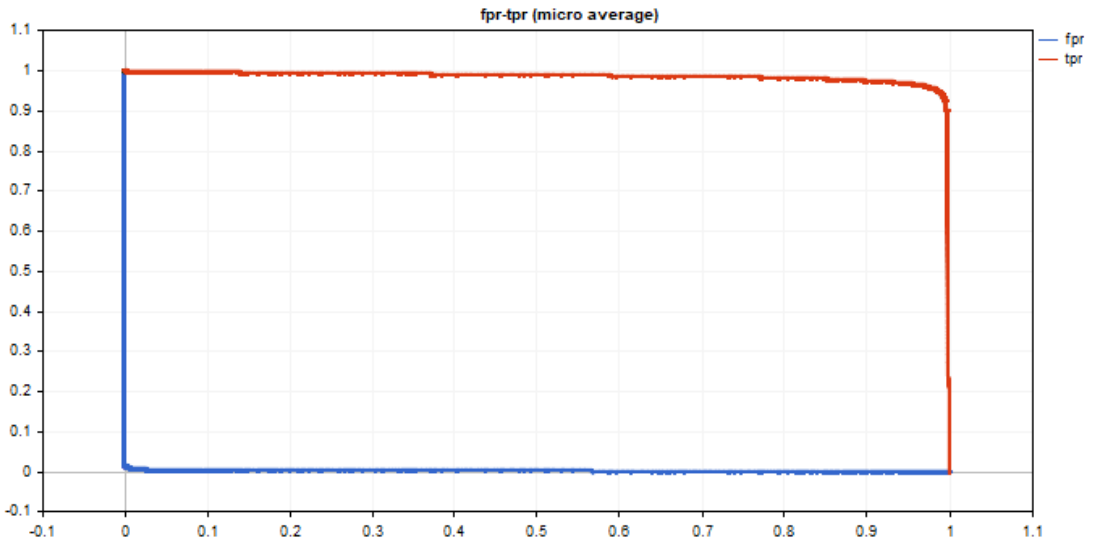
© 2000-2025, MetaQuotes Ltd.

```
  {
   double fpr[],tpr[],thres[];
   ArrayResize(fpr,mat_thres.Cols());
   ArrayResize(tpr,mat_thres.Cols());
   ArrayResize(thres,mat_thres.Cols());

   for(uint i=0; i<fpr.Size(); i++)
     {
      fpr[i]=mat_fpr[0][i];
      tpr[i]=mat_tpr[0][i];
      thres[i]=mat_thres[0][i];
     }
   thres[0]=thres[1]+0.001;

   PlotCurve("ROC curve (micro average)","roc","0.5",fpr,tpr);
   Plot2Curves("fpr-tpr (micro average)","fpr","tpr",thres,fpr,tpr);
  }
```

結果の曲線：

© 2000–2025, MetaQuotes Ltd.





© 2000–2025, MetaQuotes Ltd.

グラフ出力コードはシンプルで、<Graphics/Graphic.mqh>標準ライブラリに基づいています。

例では、mnist.onnxモデルのデータを使用します。コードはPrecisionRecallメソッドの説明に記載されています。

ROC AUCは理想に近いです。

```
roc auc score micro = [0.99991]
```

© 2000-2025, MetaQuotes Ltd.

OpenBLAS Methods

OpenBLAS is a high-performance open-source linear algebra library that implements BLAS (Basic Linear Algebra Subprograms) and some LAPACK functions. OpenBLAS is designed to improve computational performance, particularly in matrix and vector operations, which are often used in scientific and engineering tasks such as machine learning, numerical methods, and simulations.

Key features of OpenBLAS:

- Multithreading support: OpenBLAS can efficiently use multiple processor cores for parallel computations, significantly accelerating operations on multiprocessor systems.
- Optimization for processor architectures: OpenBLAS includes optimized builds for various processors such as Intel, AMD, ARM and others. The library automatically detects processor characteristics and selects the most suitable function implementations.
- Extensive BLAS operation support: OpenBLAS implements core BLAS functions, including vector operations (e.g., vector addition and dot product), matrix operations (multiplication), and vector-matrix operations.
- LAPACK compatibility: The library supports LAPACK (Linear Algebra PACKage) functions for more complex linear algebra operations, such as solving systems of linear equations, calculating matrix eigenvalues, and others.
- High performance: Compared to other BLAS libraries, OpenBLAS often shows better results due to hand-optimization for specific processor architectures.

Applications

OpenBLAS is widely used in applications involving numerical computations:

- Training neural networks and other machine learning tasks.
- Scientific computing (e.g. modeling of physical processes).
- Processing and analyzing large amounts of data.

The library is integrated into many popular scientific software packages such as NumPy, SciPy, and TensorFlow, which rely on high-performance linear algebra operations.

OpenBLAS is an excellent choice for those seeking an open-source solution for high-performance computing, particularly when working with large matrices and vectors.

Function	Action
SingularValueDecompositionDC	Singular Value Decomposition, "divide-and-conquer" algorithm. This algorithm is considered the fastest among other SVD algorithms (lapack function GESDD).
SingularValueDecompositionQR	Singular Value Decomposition, QR algorithm. This algorithm is considered a classical SVD algorithm (lapack function GESVD).

© 2000–2025, MetaQuotes Ltd.

Function	Action
SingularValueDecompositionQRPivot	Singular Value Decomposition, QR with pivoting algorithm (lapack function GESVDQ).
SingularValueDecompositionBisect	Singular Value Decomposition, bisection algorithm (lapack function GESVDX).
SingularValueDecompositionJacobiHigh	Singular Value Decomposition, Jacobi high level algorithm (lapack function GEJSV).
SingularValueDecompositionJacobiLow	Singular Value Decomposition, Jacobi low level algorithm (lapack function GESVJ). The method computes small singular values and their singular vectors with much greater accuracy than other SVD routines in certain cases.
SingularValueDecompositionBidiagDC	Singular Value Decomposition, divide-and-conquer algorithm for bidiagonal matrices (lapack function BDSVDX).
SingularValueDecompositionBidiagBisect	Singular Value Decomposition, bisection algorithm for bidiagonal matrices (lapack function BDSVDX).
EigenSolver	Compute eigenvalues and eigenvectors of a regular square matrix using the classical algorithm (lapack function GEEV).
EigenSolver2	Compute generalized eigenvalues and eigenvectors for a pair of ordinary square matrices (lapack function GGEV).
EigenSolverX	Compute eigenvalues and eigenvectors of a regular square matrix in Expert mode, i.e. with the ability to influence the computation algorithm and the ability to obtain accompanying computation data (lapack function GEEVX).
EigenSolverShur	Compute eigenvalues, upper triangular matrix in Schur form, and matrix of Schur vectors (lapack function GEES). See also Schur decomposition.
EigenSymmetricDC	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the divide-and-conquer algorithm (lapack functions SYEVD, HEEVD).
EigenSymmetricQR	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex

© 2000–2025, MetaQuotes Ltd.

Function	Action
	conjugate) matrix using the QR algorithm (lapack functions SYEV, HEEV).
EigenSymmetricRobust	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the Multiple Relatively Robust Representations, MRRR algorithm (lapack functions SYEVR, HEEVR).
EigenSymmetricBisect	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the bisection algorithm (lapack functions SYEVX, HEEVX).
SingularSpectrumAnalysisSpectrum	A method function for calculating the relative contributions of spectral components based on their eigenvalues.
SingularSpectrumAnalysisForecast	A method function for calculating reconstructed and predicted data using spectral components of the input time series.
SingularSpectrumAnalysisReconstructComponents	A method function for calculating reconstructed components of the input time series and their contributions.
SingularSpectrumAnalysisReconstructSeries	A method function for calculating the reconstructed time series using the first component_count components.

© 2000–2025, MetaQuotes Ltd.

Singular value decomposition

This section features functions for decomposing a matrix into three components: orthogonal matrices and a diagonal matrix of singular values. SVD is applied to solve various linear algebra problems such as data dimensionality reduction, image compression, solving systems of equations, and data analysis and optimization. The main functions allow you to compute singular values and vectors, reconstruct matrices, and approximate matrices with reduced rank accuracy.

Function	Action
SingularValueDecompositionDC	Singular Value Decomposition, "divide-and-conquer" algorithm. This algorithm is considered the fastest among other SVD algorithms (lapack function GESDD).
SingularValueDecompositionQR	Singular Value Decomposition, QR algorithm. This algorithm is considered a classical SVD algorithm (lapack function GESVD).
SingularValueDecompositionQRPivot	Singular Value Decomposition, QR with pivoting algorithm (lapack function GESVDQ).
SingularValueDecompositionBisect	Singular Value Decomposition, bisection algorithm (lapack function GESVDX).
SingularValueDecompositionJacobiHigh	Singular Value Decomposition, Jacobi high level algorithm (lapack function GEJSV).
SingularValueDecompositionJacobiLow	Singular Value Decomposition, Jacobi low level algorithm (lapack function GESVJ). The method computes small singular values and their singular vectors with much greater accuracy than other SVD routines in certain cases.
SingularValueDecompositionBidiagDC	Singular Value Decomposition, divide-and-conquer algorithm for bidiagonal matrices (lapack function BDSVDX).
SingularValueDecompositionBidiagBisect	Singular Value Decomposition, bisection algorithm for bidiagonal matrices (lapack function BDSVDX).

© 2000–2025, MetaQuotes Ltd.

SingularValueDecompositionDC

Singular Value Decomposition, "divide-and-conquer" algorithm. This algorithm is considered the fastest among other SVD algorithms (lapack function GESDD).

Computing for type matrix<double>

```
bool  matrix::SingularValueDecompositionDC(
   ENUM_SVD_Z       jobz,      // how to computed
   vector           S,         // vector of computed singular values
   matrix           U,         // matrix of computed left vectors U
   matrix           VT         // transposed matrix of right vectors VT
   );
```

Computing for type matrix<float>

```
bool  matrix::SingularValueDecompositionDC(
   ENUM_SVD_Z       jobz,      // how to computed
   vectorf          S,         // vector of computed singular values
   matrixf          U,         // matrix of computed left vectors U
   matrixf          VT         // transposed matrix of right vectors VT
   );
```

Computing for type matrix<complex>

```
bool  matrix::SingularValueDecompositionDC(
   ENUM_SVD_Z       jobz,      // how to computed
   vector           S,         // vector of computed singular values
   matrixc          U,         // matrix of computed left vectors U
   matrixc          VT         // transposed matrix of right vectors VT
   );
```

Parameters

jobz

[in] ENUM_SVD_Z enumeration value which determines the method for computing left and singular eigenvectors.

S

[out] Vector of singular values.

U

[out] Matrix of left singular vectors.

VT

[out] Matrix of right singular vectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

© 2000–2025, MetaQuotes Ltd.

Computation depends on the value of the jobz parameter.

When jobv is set to SVDZ_N, the left and right vectors are not computed. Only singular values are computed.

When jobv is set to SVDZ_A, the full matrices of the U and VT vectors are computed.

When the value is SVDZ_S, truncated matrices of vectors U and VT are computed.

ENUM_SVD_Z

An enumeration defining the way to compute left and right singular vectors.

ID	Description
SVDZ_N	Columns U or rows VT are not computed
SVDZ_A	All M columns of U or all N columns of VT are returned in arrays U and VT
SVDZ_S	The first min(M,N) columns of U or the first min(M,N) columns of VT are returned in arrays U and VT

See also

SingularValueDecompositionQR, SingularValueDecompositionQRPivot

© 2000–2025, MetaQuotes Ltd.

SingularValueDecompositionQR

Singular Value Decomposition, QR algorithm. Considered a classical SVD algorithm (lapack function GESVD).

Computing for type matrix<double>

```
bool  matrix::SingularValueDecompositionQR(
   ENUM_SVD_Z      jobu,     // how to compute left vectors
   ENUM_SVD_Z      jobv,     // how to compute right vectors
   vector          S,        // vector of computed singular values
   matrix          U,        // matrix of computed left vectors U
   matrix          VT        // transposed matrix of right vectors VT
   );
```

Computing for type matrix<float>

```
bool  matrix::SingularValueDecompositionQR(
   ENUM_SVD_Z      jobu,     // how to compute left vectors
   ENUM_SVD_Z      jobv,     // how to compute right vectors
   vectorf         S,        // vector of computed singular values
   matrixf         U,        // matrix of computed left vectors U
   matrixf         VT        // transposed matrix of right vectors VT
   );
```

Computing for type matrix<complex>

```
bool  matrix::SingularValueDecompositionQR(
   ENUM_SVD_Z      jobu,     // how to compute left vectors
   ENUM_SVD_Z      jobv,     // how to compute right vectors
   vector          S,        // vector of computed singular values
   matrixc         U,        // matrix of computed left vectors U
   matrixc         VT        // transposed matrix of right vectors VT
   );
```

Parameters

jobu

[in] ENUM_SVD_Z enumeration value defining how the left singular vectors should be computed.

jobv

[in] ENUM_SVD_Z enumeration value defining how the right singular vectors should be computed.

S

[out] Vector of singular values.

U

[out] Matrix of left singular vectors.

VT

[out] Matrix of right singular vectors.

© 2000–2025, MetaQuotes Ltd.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the value of the jobu and jobv parameters.

When set to SVDZ_N, the left (jobu) and right (jobv) vectors are not computed. Singular values are always computed.

When set to SVDZ_A, the full matrices of vectors U (jobu) or VT (jobv) are computed.

With the value SVDZ_S, truncated matrices of vectors U (jobu) or VT (jobv) are computed.

ENUM_SVD_Z

An enumeration defining the way to compute left and right singular vectors.

ID	Description
SVDZ_N	Columns U or rows VT are not computed
SVDZ_A	All M columns of U or all N columns of VT are returned in arrays U and VT
SVDZ_S	The first min(M,N) columns of U or the first min(M,N) columns of VT are returned in arrays U and VT

See also

SingularValueDecompositionDC, SingularValueDecompositionQRPivot

© 2000–2025, MetaQuotes Ltd.

SingularValueDecompositionQRPivot

Singular Value Decomposition, QR with pivoting algorithm (lapack function GESVDQ).

Computing for type matrix<double>

```
bool  matrix::SingularValueDecompositionQRPivot(
   ENUM_SVDQRP_A   joba,     // computation accuracy level
   ENUM_SVDQRP_P   jobp,     // use row reversal to compute
   ENUM_SVDQRP_R   jobr,     // use triangular matrix R to compute
   ENUM_SVDQRP_U   jobu,     // how to compute left vectors
   ENUM_SVDQRP_V   jobv,     // how to compute right vectors
   vector          S,        // vector of computed singular values
   matrix          U,        // matrix of computed left vectors U
   matrix          VT        // transposed matrix of right vectors VT
   );
```

Computing for type matrix<float>

```
bool  matrix::SingularValueDecompositionQRPivot(
   ENUM_SVDQRP_A   joba,     // computation accuracy level
   ENUM_SVDQRP_P   jobp,     // use row reversal to compute
   ENUM_SVDQRP_R   jobr,     // use triangular matrix R to compute
   ENUM_SVDQRP_U   jobu,     // how to compute left vectors
   ENUM_SVDQRP_V   jobv,     // how to compute right vectors
   vectorf         S,        // vector of computed singular values
   matrixf         U,        // matrix of computed left vectors U
   matrixf         VT        // transposed matrix of right vectors VT
   );
```

Computing for type matrix<complex>

```
bool  matrix::SingularValueDecompositionQRPivot(
   ENUM_SVDQRP_A   joba,     // computation accuracy level
   ENUM_SVDQRP_P   jobp,     // use row reversal to compute
   ENUM_SVDQRP_R   jobr,     // use triangular matrix R to compute
   ENUM_SVDQRP_U   jobu,     // how to compute left vectors
   ENUM_SVDQRP_V   jobv,     // how to compute right vectors
   vector          S,        // vector of computed singular values
   matrixc         U,        // matrix of computed left vectors U
   matrixc         VT        // transposed matrix of right vectors VT
   );
```

Parameters

joba

[in] ENUM_SVDQRP_A enumeration value defining the accuracy level of the SVD computation.

jobp

© 2000–2025, MetaQuotes Ltd.

[in] ENUM_SVDQRP_P enumeration value defining the use of row reversal during the computation process.

jobr

[in] ENUM_SVDQRP_R enumeration value defining whether to transpose the triangular matrix R obtained as a result of the initial QR factorization.

jobu

[in] ENUM_SVDQRP_U enumeration value that determines how the left singular vectors should be computed.

jobv

[in] ENUM_SVDQRP_V enumeration value defining how the right singular vectors should be computed.

S

[out] Vector of singular values.

U

[out] Matrix of left singular vectors.

VT

[out] Matrix of right singular vectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

The number of matrix rows must not be less than the number of columns.

If both left and right singular vectors are computed, jobv must be set to SVDQRPV_R when jobu=SVDQRPU_R.

ENUM_SVDQRP_A

An enumeration that specifies the level of accuracy of the SVD computation.

ID	Description
SVDQRP_A	The requested accuracy corresponds to the inverse error limited by epsilon. This is an aggressive level of truncation.
SVDQRPA_M	Similar to SVDQRP_A, but the truncation is softer. This is the average level of truncation.
SVDQRPA_H	High accuracy is required.

ENUM_SVDQRP_P

An enumeration that specifies whether to use row reversal during calculation.

© 2000–2025, MetaQuotes Ltd.

ID	Description
SVDQRPP_P	The rows of A are ordered in descending order. Recommended for numerical reliability.
SVDQRPP_N	No row reversal.

ENUM_SVDQRP_R

An enumeration that specifies whether to transpose the triangular matrix R obtained as a result of the initial QR factorization.

ID	Description
SVDQRPR_T	After the initial rotated QR factorization, GESVD is applied to the transpose R**T of the calculated triangular factor R.
SVDQRPR_N	The triangular factor R is given as input to GESVD.

ENUM_SVDQRP_U

An enumeration defining how left singular vectors should be computed.

ID	Description
SVDQRPU_A	All M left singular vectors are computed.
SVDQRPU_S	The min(M,N) left singular vectors are computed.
SVDQRPU_R	The numerical rank NUMRANK is determined and only the NUMRANK of the left singular vectors is computed.
SVDQRPU_F	N left singular vectors are returned.
SVDQRPU_N	Left singular vectors are not computed.

ENUM_SVDQRP_V

An enumeration defining how right singular vectors should be computed.

ID	Description
SVDQRPV_A	All N right singular vectors are computed.
SVDQRPV_R	The numerical rank NUMRANK is defined and only the NUMRANK of right singular vectors are computed.
SVDQRPV_N	Right singular vectors are not computed.

See also

SingularValueDecompositionDC, SingularValueDecompositionQR

© 2000–2025, MetaQuotes Ltd.

SingularValueDecompositionBisect

Singular Value Decomposition, bisection algorithm (lapack function GESVDX).

Computing for type matrix<double>

```
bool  matrix::SingularValueDecompositionBisect(
   ENUM_SVD_VECTORS     jobv,      // computation accuracy level
   ENUM_BLAS_RANGE      range,     // subset of computable singular values
   double               lower,     // lower limit of the subset
   double               upper,     // upper limit of the subset
   vector               S,         // vector of computed singular values
   matrix               U,         // matrix of computed left vectors U
   matrix               VT         // transposed matrix of right vectors VT
   );
```

Computing for type matrix<float>

```
bool  matrix::SingularValueDecompositionBisect(
   ENUM_SVD_VECTORS     jobv,      // computation accuracy level
   ENUM_BLAS_RANGE      range,     // subset of computable singular values
   double               lower,     // lower limit of the subset
   double               upper,     // upper limit of the subset
   vectorf              S,         // vector of computed singular values
   matrixf              U,         // matrix of computed left vectors U
   matrixf              VT         // ransposed matrix of right vectors VT
   );
```

Computing for type matrix<complex>

```
bool  matrix::SingularValueDecompositionBisect(
   ENUM_SVD_VECTORS     jobv,      // computation accuracy level
   ENUM_BLAS_RANGE      range,     // subset of computable singular values
   double               lower,     // lower limit of the subset
   double               upper,     // upper limit of the subset
   vector               S,         // vector of computed singular values
   matrixc              U,         // matrix of computed left vectors U
   matrixc              VT         // transposed matrix of right vectors VT
   );
```

Parameters

jobv

[in] ENUM_SVD_VECTORS enumeration value which determines the method for computing left and singular eigenvectors.

range

[in] ENUM_BLAS_RANGE enumeration value that defines a subset of computable singular values and vectors.

lower

© 2000–2025, MetaQuotes Ltd.

[in] The lower limit of singular values subset; specified depending on the value of the range parameter.

upper

[in] The upper limit of singular values subset; specified depending on the value of the range parameter.

S

[out] Vector of singular values.

U

[out] Matrix of left singular vectors.

VT

[out] Matrix of right singular vectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the values of the *jobuv* and *range* parameters.

When BLASRANGE_A is set, all singular values are computed, and the lower and upper parameters are ignored.

With the BLASRANGE_V value, only those singular values (and their vectors) that fall within the range of real values specified by the 'lower' and 'upper' parameters are computed.

With the BLASRANGE_I value, only those singular values (and their vectors) that fall within the range of integer indices specified by the 'lower' and 'upper' parameters are computed. For example, with lower=0 and upper=2, only the first three singular values are computed.

ENUM_SVD_VECTORS

An enumeration defining the way to compute left and right singular vectors.

ID	Description
SVDVECTORS_N	Only singular values are computed, without vectors.
SVDVECTORS_U	Left singular vectors are computed.
SVDVECTORS_V	Right singular vectors are computed.
SVDVECTORS_UV	Left and right singular vectors are computed.

ENUM_BLAS_RANGE

An enumeration defining a subset of computable singular values and vectors.

© 2000–2025, MetaQuotes Ltd.

ID	Description
BLASRANGE_A	All singular or eigenvalues will be found.
BLASRANGE_V	All singular or eigenvalues in the half-open interval (VL,VU] will be found.
BLASRANGE_I	Singular or eigenvalues from IL to IU will be found

See also

SingularValueDecompositionDC, SingularValueDecompositionQR, SingularValueDecompositionQRPivot

© 2000–2025, MetaQuotes Ltd.

SingularValueDecompositionJacobiHigh

Singular Value Decomposition, Jacobi high level algorithm (lapack function GEJSV).

Computing for type matrix<double>

```
bool  matrix::SingularValueDecompositionJacobiHigh(
   ENUM_SVDJH_A   joba,            // computation accuracy level
   ENUM_SVDJH_U   jobu,            // how to compute left vectors
   ENUM_SVDJH_V   jobv,            // how to compute right vectors
   ENUM_SVDJH_R   jobr,            // define range of computable singular values
   ENUM_SVDJH_T   jobt,            // define whether to transpose when computing a
square matrix
   ENUM_SVDJH_P   jobp,            // the possibility of structured perturbations to
remove denormalized values
   vector          S,               // vector of computed singular values
   matrix         U,               // matrix of computed left vectors U
   matrix         V,               // matrix of computed left vectors V
   vector         work_results     // additional computation results
   );
```

Computing for type matrix<float>

```
bool  matrix::SingularValueDecompositionJacobiHigh(
   ENUM_SVDJH_A   joba,            // computation accuracy level
   ENUM_SVDJH_U   jobu,            // how to compute left vectors
   ENUM_SVDJH_V   jobv,            // how to compute right vectors
   ENUM_SVDJH_R   jobr,            // define range of computable singular values
   ENUM_SVDJH_T   jobt,            // define whether to transpose when computing a
square matrix
   ENUM_SVDJH_P   jobp,            // the possibility of structured perturbations to
remove denormalized values
   vectorf        S,               // vector of computed singular values
   matrixf        U,               // U matrix of computed left vectors
   matrixf        V,               // V matrix of computed left vectors
   vector         work_results     // additional computation results
   );
```

Computing for type matrix<complex>

```
bool  matrix::SingularValueDecompositionJacobiHigh(
   ENUM_SVDJH_A   joba,            // computation accuracy level
   ENUM_SVDJH_U   jobu,            // how to compute left vectors
   ENUM_SVDJH_V   jobv,            // how to compute right vectors
   ENUM_SVDJH_R   jobr,            // define range of computable singular values
   ENUM_SVDJH_T   jobt,            // define whether to transpose when computing a
square matrix
   ENUM_SVDJH_P   jobp,            // the possibility of structured perturbations to
remove denormalized values
   vectorc        S,               // vector of computed singular values
   matrixc        U,               // U matrix of computed left vectors
```

© 2000–2025, MetaQuotes Ltd.

```
   matrixc        V,              // V matrix of computed left vectors
   vector         work_results    // additional computation results
   );
```

Parameters

joba

[in] ENUM_SVDJH_A enumeration value determining the accuracy level of the SVD computation.

jobu

[in] ENUM_SVDJH_U enumeration value that determines how the left singular vectors should be computed.

jobv

[in] ENUM_SVDJH_V enumeration value that determines how the right singular vectors should be computed.

jobr

[in] ENUM_SVDJH_R enumeration value defining the range of computable values

jobt

[in] ENUM_SVDJH_T enumeration value defining whether to transpose the matrix of it is square. If the matrix is non-square, this parameter is ignored.

jobp

[in] ENUM_SVDJH_P enumeration value defining the possibility of structured perturbations to remove denormalized values.

S

[out] Vector of singular values.

For work(1)/work(2) = one: the singular values of A. During the computation S contains Euclidean column norms of the iterated matrices in the array a.

For work(1)≠work(2): the singular values of A are (work(1)/work(2)) * S(1:n). This factored form is used if sigma_max(A) overflows or if small singular values have been saved from underflow by scaling the input matrix A.

jobr = 'R', some of the singular values may be returned as exact zeros obtained by 'setting to zero' because they are below the numerical rank threshold or are denormalized numbers.

U

[out] Matrix of left singular vectors.

V

[out] Matrix of right singular vectors.

work_results

[out] Vector consisting of 7 statistics obtained as a result of the computation.

work(1) = scale = work(2)/work(1) is the scaling factor such that scale*sva(1:n) are the computed singular values of A. See the description of S.

work(2) = see the description of work(1).

© 2000–2025, MetaQuotes Ltd.

work(3) = sconda is an estimate for the condition number of column equilibrated A. If joba = 'E' or 'G', sconda is an estimate of sqrt(||(R**t * R)**(-1)||_1). It is computed using ?pocon. It holds n**(-1/4) * sconda ≤ ||R**(-1)||_2 ≤ n**(1/4) * sconda, where R is the triangular factor from the QRF of A. However, if R is truncated and the numerical rank is determined to be strictly smaller than n, sconda is returned as -1, indicating that the smallest singular values might be lost.

If full SVD is needed, the following two condition numbers are useful for the analysis of the algorithm. They are provied for a user who is familiar with the details of the method.

work(4) = an estimate of the scaled condition number of the triangular factor in the first QR factorization.

work(5) = an estimate of the scaled condition number of the triangular factor in the second QR factorization.

The following two parameters are computed if jobt = 'T'. They are provided for a user who is familiar with the details of the method.

work(6) = the entropy of A**t*A :: this is the Shannon entropy of diag(A**t*A) / Trace(A**t*A) taken as point in the probability simplex.

work(7) = the entropy of A*A**t.

Return Value

Return true if successful, otherwise false in case of an error.

Note

The number of matrix rows must not be less than the number of columns.

ENUM_SVDJH_A

An enumeration that specifies the level of accuracy of the SVD computation.

ID	Description
SVDJHA_C	Computation as with 'C' with an additional estimate of the condition number. It provides a realistic error bound.
SVDJHA_E	Computation as with SVDJHA_C' with an additional estimate of the condition number. It provides a realistic error bound.
SVDJHA_F	Higher accuracy than the SVDJHA_C option.
SVDJHA_G	Computation as with SVDJHA_F with an additional estimate of the condition number.
SVDJHA_A	Small singular values are the noise and the matrix is treated as numerically rank deficient.
SVDJHA_R	Similar as in SVDJHA_A, but more accuracy.

ENUM_SVDJH_U

An enumeration defining how left singular vectors should be computed.

© 2000–2025, MetaQuotes Ltd.

ID	Description
SVDJHU_U	N columns of U are returned in the array U.
SVDJHU_F	Full set of M left singular vectors is returned in the array U.
SVDJHU_N	U is not computed.

ENUM_SVDJH_V

An enumeration defining how right singular vectors should be computed.

ID	Description
SVDJHV_V	N columns of V are returned in the array V
SVDJHV_J	N columns of V are returned in the array V, but they are computed as the product of Jacobi rotations
SVDJHV_N	V is not computed

ENUM_SVDJH_R

An enumeration that defines the range of values to be computed.

ID	Description
SVDJHR_N	Do not kill small columns of c*A.
SVDJHR_R	RESTRICTED range for sigma(c*A). This option is recommended.

ENUM_SVDJH_T

An enumeration that specifies whether a matrix should be transposed if it is square.

ID	Description
SVDJHT_T	Transpose if entropy test indicates possibly faster convergence of Jacobi process.
SVDJHT_N	Do not use transposition. Do not speculate.

ENUM_SVDJH_P

An enumeration that specifies the possibility of structured perturbations to remove denormalized values.

ID	Description
SVDJHP_P	Introduce perturbation.
SVDJHP_N	Do not perturb.

© 2000–2025, MetaQuotes Ltd.

See also

SingularValueDecompositionDC, SingularValueDecompositionQR

© 2000–2025, MetaQuotes Ltd.

SingularValueDecompositionJacobiLow

Singular Value Decomposition, Jacobi low level algorithm (lapack function GESVJ). The method computes small singular values and their singular vectors with much greater accuracy than other SVD routines in certain cases.

Computing for type matrix<double>

```
bool  matrix::SingularValueDecompositionJacobiLow(
   ENUM_SVDJH_U   jobu,            // how to compute left vectors
   ENUM_SVDJH_V   jobv,            // how to compute right vectors
   double         ctol,            // threshold for convergence if jobu='C'
   ulong          mv,              // number of first rows of matrix V if jobv='A'
   vector         S,               // vector of computed singular values
   matrix         U,               // matrix of computed left vectors U
   matrix         V,               // matrix of computed left vectors V
   vector         work_results     // additional computation results
   );
```

Computing for type matrix<float>

```
bool  matrix::SingularValueDecompositionJacobiLow(
   ENUM_SVDJH_U   jobu,            // how to compute left vectors
   ENUM_SVDJH_V   jobv,            // how to compute right vectors
   double         ctol,            // threshold for convergence if jobu='C'
   ulong          mv,              // number of first rows of matrix V if jobv='A'
   vectorf        S,               // vector of computed singular values
   matrixf        U,               // U matrix of computed left vectors
   matrixf        V,               // V matrix of computed left vectors
   vector         work_results     // additional computation results
   );
```

Computing for type matrix<complex>

```
bool  matrix::SingularValueDecompositionJacobiLow(
   ENUM_SVDJH_U   jobu,            // how to compute left vectors
   ENUM_SVDJH_V   jobv,            // how to compute right vectors
   double         ctol,            // threshold for convergence if jobu='C'
   ulong          mv,              // number of first rows of matrix V if jobv='A'
   vector         S,               // vector of computed singular values
   matrixc        U,               // U matrix of computed left vectors
   matrixc        V,               // V matrix of computed left vectors
   vector         work_results     // additional computation results
   );
```

Parameters

jobu

[in] ENUM_SVDJL_U enumeration value that determines how the left singular vectors should be computed.

© 2000–2025, MetaQuotes Ltd.

jobv

[in] ENUM_SVDJL_V enumeration value defining how the right singular vectors should be computed.

ctol

[in] Convergence threshold if jobu=SVDJLU_C. For other values of 'jobu' the parameter is ignored.

mv

[in] Number of rows of matrix V to be computed if jobv=SVDJLV_A. For other values of 'jobv' the parameter is ignored.

S

[out] Vector of singular values.

Depending on the value scale = work(1), where scale is the scaling factor:

if scale = 1, S(1:n) contains the computed singular values of a. During the computation, sva contains the Euclidean column norms of the iterated matrices in the array a.

if scale ≠ 1, the singular values of a are scale*S(1:n), and this factored representation is due to the fact that some of the singular values of a might underflow or overflow.

U

[out] Matrix of left singular vectors.

V

[out] Matrix of right singular vectors (non-transposed).

work_results

[out] Vector consisting of 7 statistics obtained as a result of the computation.

work(1) = scale is the scaling factor such that scale*S(1:n) are the computed singular values of A. See the description of S).

work(2) is the number of the computed nonzero singular value.

work(3) is the number of the computed singular values that are larger than the underflow threshold.

work(4) is the number of sweeps of Jacobi rotations needed for numerical convergence.

work(5) = max_{i.NE.j} |COS(A(:,i),A(:,j))| in the last sweep. This is useful information in cases when ?gesvj did not converge, as it can be used to estimate whether the output is still useful and for post festum analysis.

work(6) is the largest absolute value over all sines of the Jacobi rotation angles in the last sweep. It can be useful in a post festum analysis.

Return Value

Return true if successful, otherwise false in case of an error.

Note

The number of matrix rows must not be less than the number of columns.

© 2000–2025, MetaQuotes Ltd.

ENUM_SVDJL_U

An enumeration defining how left singular vectors should be computed.

ID	Description
SVDJLU_U	The left singular vectors corresponding to the nonzero singular values are computed and returned in the leading columns of A
SVDJLU_C	Analogous to SVDJLU_U, except that user can control the level of numerical orthogonality of the computed left singular vectors.
SVDJLU_N	The matrix U is not computed.

ENUM_SVDJL_V

An enumeration defining how right singular vectors should be computed.

ID	Description
SVDJLV_V	The matrix V is computed
SVDJLV_A	The Jacobi rotations are applied to the MV-by-N matrix V.
SVDJLV_N	The matrix V is not computed.

See also

SingularValueDecompositionDC, SingularValueDecompositionQR

© 2000–2025, MetaQuotes Ltd.

SingularValueDecompositionBidiagDC

Singular Value Decomposition, divide-and-conquer algorithm for bidiagonal matrices (lapack function BDSVDX).

Computing for type matrix<double>

```
bool  matrix::SingularValueDecompositionBidiagDC(
   ENUM_SVDBIDIAG_Z    jobz,       // how to compute left vectors
   ENUM_BLAS_RANGE     range,      // subset of computed singular values
   double              lower,      // lower limit of the subset
   double              upper,      // upper limit of the subset
   vector              S,          // vector of computed singular values
   matrix              U,          // U matrix of computed left vectors
   matrix              VT          // VT transposed matrix of right vectors
   );
```

Computing for type matrix<float>

```
bool  matrix::SingularValueDecompositionBidiagDC(
   ENUM_SVDBIDIAG_Z    jobz,       // how to compute left vectors
   ENUM_BLAS_RANGE     range,      // subset of computed singular values
   double              lower,      // lower limit of the subset
   double              upper,      // upper limit of the subset
   vectorf             S,          // vector of computed singular values
   matrixf             U,          // U matrix of computed left vectors
   matrixf             VT          // VT transposed matrix of right vectors
   );
```

Parameters

jobz

[in] ENUM_SVDBIDIAG_Z enumeration value that determines how the left singular vectors should be computed.

range

[in] ENUM_BLAS_RANGE enumeration value that defines a subset of computable singular values and vectors.

lower

[in] The lower limit of singular values subset; specified depending on the value of the range parameter.

upper

[in] The upper limit of singular values subset; specified depending on the value of the range parameter.

S

[out] Vector of singular values.

U

[out] Matrix of left singular vectors.

© 2000–2025, MetaQuotes Ltd.

V

[out] Transposed matrix of right singular vectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the values of the jobz and range parameters.

When BLASRANGE_A is set, all singular values are computed, and the lower and upper parameters are ignored.

With the BLASRANGE_V value, only those singular values (and their vectors) that fall within the range of real values specified by the 'lower' and 'upper' parameters are computed.

With the BLASRANGE_I value, only those singular values (and their vectors) that fall within the range of integer indices specified by the 'lower' and 'upper' parameters are computed. For example, with lower=0 and upper=2, only the first three singular values are computed.

A bidiagonal matrix is a square matrix with non-zero main diagonal and one of the sub-diagonals.

Upper bidiagonal matrix

```
[[x, x, 0, 0, 0],
 [0, x, x, 0, 0],
 [0, 0, x, x, 0],
 [0, 0, 0, x, x],
 [0, 0, 0, 0, x]]
```

Lower bidiagonal matrix

```
[[x, 0, 0, 0, 0],
 [x, x, 0, 0, 0],
 [0, x, x, 0, 0],
 [0, 0, x, x, 0],
 [0, 0, 0, x, x]]
```

ENUM_SVDBIDIAG_Z

An enumeration defining how left singular vectors should be computed.

ID	Description
SVDJOBZ_V	Compute singular values and singular vectors.
SVDJOBZ_N	Compute singular values only.

ENUM_BLAS_RANGE

© 2000–2025, MetaQuotes Ltd.

An enumeration defining how right singular vectors should be computed.

ID	Description
BLASRANGE_A	All singular or eigenvalues will be found.
BLASRANGE_V	All singular or eigenvalues in the half-open interval (VL,VU] will be found.
BLASRANGE_I	The IL-th through IU-th singular or eigenvalues will be found.

See also

SingularValueDecompositionDC, SingularValueDecompositionQR

© 2000–2025, MetaQuotes Ltd.

SingularValueDecompositionBidiagBisect

Singular Value Decomposition, bisection algorithm for bidiagonal matrices (lapack function BDSVDX).

Computing for type matrix<double>

```
bool  matrix::SingularValueDecompositionBidiagBisect(
   ENUM_SVDBIDIAG_Z    jobz,       // how to compute left vectors
   ENUM_BLAS_RANGE     range,      // subset of computed singular values
   double              lower,      // lower limit of the subset
   double              upper,      // upper limit of the subset
   vector              S,          // vector of computed singular values
   matrix              U,          // U matrix of computed left vectors
   matrix              VT          // VT transposed matrix of right vectors
   );
```

Computing for type matrix<float>

```
bool  matrix::SingularValueDecompositionBidiagBisect(
   ENUM_SVDBIDIAG_Z    jobz,       // how to compute left vectors
   ENUM_BLAS_RANGE     range,      // subset of computed singular values
   double              lower,      // lower limit of the subset
   double              upper,      // upper limit of the subset
   vectorf             S,          // vector of computed singular values
   matrixf             U,          // U matrix of computed left vectors
   matrixf             VT          // VT transposed matrix of right vectors
   );
```

Parameters

jobz

[in] ENUM_SVDBIDIAG_Z enumeration value that determines how the left singular vectors should be computed.

range

[in] ENUM_BLAS_RANGE enumeration value that defines a subset of computable singular values and vectors.

lower

[in] The lower limit of singular values subset; specified depending on the value of the range parameter.

upper

[in] The upper limit of singular values subset; specified depending on the value of the range parameter.

S

[out] Vector of singular values.

U

[out] Matrix of left singular vectors.

© 2000–2025, MetaQuotes Ltd.

V

[out] Transposed matrix of right singular vectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

When BLASRANGE_A is set, all singular values are computed, and the lower and upper parameters are ignored.

With the BLASRANGE_V value, only those singular values (and their vectors) that fall within the range of real values specified by the 'lower' and 'upper' parameters are computed.

With the BLASRANGE_I value, only those singular values (and their vectors) that fall within the range of integer indices specified by the 'lower' and 'upper' parameters are computed. For example, with lower=0 and upper=2, only the first three singular values are computed.

A bidiagonal matrix is a square matrix with non-zero main diagonal and one of the sub-diagonals.

Upper bidiagonal matrix

```
[[x, x, 0, 0, 0],
 [0, x, x, 0, 0],
 [0, 0, x, x, 0],
 [0, 0, 0, x, x],
 [0, 0, 0, 0, x]]
```

Lower bidiagonal matrix

```
[[x, 0, 0, 0, 0],
 [x, x, 0, 0, 0],
 [0, x, x, 0, 0],
 [0, 0, x, x, 0],
 [0, 0, 0, x, x]]
```

ENUM_SVDBIDIAG_Z

An enumeration defining how left singular vectors should be computed.

ID	Description
SVDJOBZ_V	Compute singular values and singular vectors.
SVDJOBZ_N	Compute singular values only.

ENUM_BLAS_RANGE

An enumeration defining how right singular vectors should be computed.

© 2000–2025, MetaQuotes Ltd.

ID	Description
BLASRANGE_A	All singular or eigenvalues will be found.
BLASRANGE_V	All singular or eigenvalues in the half-open interval (VL,VU] will be found.
BLASRANGE_I	The IL-th through IU-th singular or eigenvalues will be found.

See also

SingularValueDecompositionDC, SingularValueDecompositionQR

© 2000–2025, MetaQuotes Ltd.

Eigen Values

The section features functions for computing eigenvalues and eigenvectors. It describes methods for solving standard linear algebra problems using the LAPACK library algorithms. These functions are efficient for matrix analysis, diagonalization, system stabilization, and other tasks.

- **EigenSolver**: The function is designed to compute the eigenvalues and eigenvectors of an arbitrary square matrix using the classical algorithm represented by the GEEV lapack function. This method is applied to a wide range of matrices, allowing the decomposition of matrices into their eigenvalues and eigenvectors.
- **EigenSymmetricDC**: The function for computing eigenvalues and eigenvectors of symmetric or Hermitian matrices using the divide-and-conquer algorithm. The lapack functions SYEVD and HEEVD enable the efficient handling of symmetric or Hermitian matrices, providing faster and more accurate processing of such matrices.

Function	Action
EigenSolver	Compute eigenvalues and eigenvectors of a regular square matrix using the classical algorithm (lapack function GEEV).
EigenSolverX	Compute eigenvalues and eigenvectors of a regular square matrix in Expert mode, i.e. with the ability to influence the computation algorithm and the ability to obtain accompanying computation data (lapack function GEEVX).
EigenSolverShur	Compute eigenvalues, upper triangular matrix in Schur form, and matrix of Schur vectors (lapack function GEES). See also Schur decomposition.
EigenSolver2	Compute generalized eigenvalues and eigenvectors for a pair of ordinary square matrices (lapack function GGEV).
EigenSolver2X	Compute generalized eigenvalues and eigenvectors for a pair of regular square matrices in Expert mode, i.e. with the ability to influence the computation algorithm and the ability to obtain accompanying computation data (lapack function GGEVX). Both matrices must be the same size.
EigenSolver2Shur	Compute a pair of ordinary square matrices of generalized eigenvalues, generalized eigenvectors, generalized Schur forms, as well as left and right Schur vectors (lapack function GGES).
EigenSolver2Blocked	Compute generalized eigenvalues and eigenvectors for a pair of regular square matrices using a block algorithm (lapack function GGEV3). Both matrices must be the same size. The method parameters are exactly the same as EigenSolver2.

© 2000–2025, MetaQuotes Ltd.

Function	Action
EigenSolver2ShurBlocked	Compute a pair of regular square matrices of generalized eigenvalues, generalized eigenvectors, generalized Schur forms, as well as left and right Schur vectors (lapack function GGES3).
EigenSymmetricDC	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the divide-and-conquer algorithm (lapack functions SYEVD, HEEVD).
EigenSymmetricQR	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the QR algorithm (lapack functions SYEV, HEEV).
EigenSymmetricRobust	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the Multiple Relatively Robust Representations, MRRR algorithm (lapack functions SYEVR, HEEVR).
EigenSymmetricBisect	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the bisection algorithm (lapack functions SYEVX, HEEVX).

© 2000–2025, MetaQuotes Ltd.

General Matrices

Functions for calculating eigenvalues and eigenvectors of a square matrix using classical algorithms. It provides various methods for working with both real and complex matrices, allowing you to solve linear algebra problems with a choice of methods for calculating eigenvectors.

Function	Action
EigenSolver	Compute eigenvalues and eigenvectors of a regular square matrix using the classical algorithm (lapack function GEEV).
EigenSolverX	Compute eigenvalues and eigenvectors of a regular square matrix in Expert mode, i.e. with the ability to influence the computation algorithm and the ability to obtain accompanying computation data (lapack function GEEVX).
EigenSolverShur	Compute eigenvalues, upper triangular matrix in Schur form, and matrix of Schur vectors (lapack function GEES). See also Schur decomposition.
EigenSolver2	Compute generalized eigenvalues and eigenvectors for a pair of ordinary square matrices (lapack function GGEV).
EigenSolver2X	Compute generalized eigenvalues and eigenvectors for a pair of regular square matrices in Expert mode, i.e. with the ability to influence the computation algorithm and the ability to obtain accompanying computation data (lapack function GGEVX). Both matrices must be the same size.
EigenSolver2Shur	Compute a pair of ordinary square matrices of generalized eigenvalues, generalized eigenvectors, generalized Schur forms, as well as left and right Schur vectors (lapack function GGES).
EigenSolver2Blocked	Compute generalized eigenvalues and eigenvectors for a pair of regular square matrices using a block algorithm (lapack function GGEV3). Both matrices must be the same size. The method parameters are exactly the same as EigenSolver2.
EigenSolver2ShurBlocked	Compute a pair of regular square matrices of generalized eigenvalues, generalized eigenvectors, generalized Schur forms, as well as left and right Schur vectors (lapack function GGES3).

© 2000–2025, MetaQuotes Ltd.

EigenSolver

Compute eigenvalues and eigenvectors of a regular square matrix using the classical algorithm (lapack function GEEV).

Computing for type matrix<double>

```
bool  matrix::EigenSolver(
   ENUM_EIG_VECTORS      jobv,                 // compute left and right
eigenvectors
   vectorc&              eigen_values,         // vector of computed eigenvalues
   matrix&               left_eigenvectors,    // matrix of computed left vectors
   matrix&               right_eigenvectors    // matrix of computed right
vectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSolver(
   ENUM_EIG_VECTORS      jobv,                 // compute left and right
eigenvectors
   vectorcf&             eigen_values,         // vector of computed eigenvalues
   matrixf&              left_eigenvectors,    // matrix of computed left vectors
   matrixf&              right_eigenvectors    // matrix of computed right
vectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSolver(
   ENUM_EIG_VECTORS      jobv,                 // compute left and right vectors
   vectorc&              eigen_values,         // vector of computed eigenvalues
   matrixc&              left_eigenvectors,    // matrix of computed left vectors
   matrixc&              right_eigenvectors    // matrix of computed right
vectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSolver(
   ENUM_EIG_VECTORS      jobv,                 // compute left and right vectors
   vectorcf&             eigen_values,         // vector of computed eigenvalues
   matrixcf&             left_eigenvectors,    // matrix of computed left vectors
   matrixcf&             right_eigenvectors    // matrix of computed right
vectors
   );
```

Parameters

jobv

[in] ENUM_EIG_VECTORS enumeration value which determines the method for computing left and right eigenvectors.

© 2000–2025, MetaQuotes Ltd.

EV

[out] Vector of eigenvalues.

left_eigenvectors

[out] Matrix of left eigenvectors.

right_eigenvectors

[out] Matrix of right eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the value of the jobv parameter.

If EIGVECTORS_N is set, the left and right vectors are not computed. Only eigenvalues are computed.

With EIGVECTORS_L, only left eigenvectors are computed, right eigenvectors are not computed.

When EIGVECTORS_R is set, only the right eigenvectors are computed, the left vectors are not computed.

With EIGVECTORS_LR, the left and right eigenvectors are computed, Eigenvalues are always computed.

Real (non-complex) matrices can have a complex solution. Therefore, the vector of eigenvalues must be complex. In case of a complex solution, the error code is set to 4019 (ERR_MATH_OVERFLOW). Otherwise, only the real parts of the complex values of the eigenvalue vector should be used.

ENUM_EIG_VECTORS

An enumeration that specifies whether to calculate eigenvectors.

ID	Description
EIGVECTORS_N	Only eigenvalues are calculated, without vectors.
EIGVECTORS_L	Only left eigenvectors are computed.
EIGVECTORS_R	Only right eigenvectors are computed.
EIGVECTORS_LR	Left and right eigenvectors are computed, eigenvalues are always computed.

© 2000–2025, MetaQuotes Ltd.

EigenSolverX

Compute eigenvalues and eigenvectors of a regular square matrix in Expert mode, i.e. with the ability to influence the computation algorithm and the ability to obtain accompanying computation data (lapack function GEEVX).

Computing for type matrix<double>

```
bool  matrix::EigenSolverX(
   ENUM_EIG_BALANCE      balance,                // input matrix balancing method
   ENUM_EIG_VECTORS      jobv,                   // determines computation of right
and left eigenvectors
   ENUM_EIG_SENSE        sense,                  // determines computation of
reciprocal condition numbers
   vectorc&              eigen_values,           // vector of computed eigenvalues
   matrix&               left_eigenvectors,      // matrix of computed left vectors
   matrix&               right_eigenvectors      // matrix of computed right
vectors
   matrix&               shur_matrix,            // balanced matrix in Schur form
   long&                 ilo,                    // subscript of balanced matrix
   long&                 ihi,                    // superscript of balanced matrix
   vector&               scale,                  // details of permutations and
scaling when balancing the input matrix
   double&               ab_norm,                // 1-norm of balanced matrix
   vector&               rconde,                 // vector of reciprocal condition
numbers for each eigenvalue
   vector&               rcondv                  // vector of reciprocal condition
numbers for each eigenvector
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSolverX(
   ENUM_EIG_BALANCE      balance,                // input matrix balancing method
   ENUM_EIG_VECTORS      jobv,                   // determines computation of right
and left eigenvectors
   ENUM_EIG_SENSE        sense,                  // determines computation of
reciprocal condition numbers
   vectorcf&             eigen_values,           // vector of computed eigenvalues
   matrixf&              left_eigenvectors,      // matrix of computed left vectors
   matrixf&              right_eigenvectors      // matrix of computed right
vectors
   matrixf&              shur_matrix,            // balanced matrix in Schur form
   long&                 ilo,                    // subscript of balanced matrix
   long&                 ihi,                    // superscript of balanced matrix
   vectorf&              scale,                  // details of permutations and
scaling when balancing the input matrix
   float&                ab_norm,                // 1-norm of balanced matrix
   vectorf&              rconde,                 // vector of reciprocal condition
numbers for each eigenvalue
```

© 2000–2025, MetaQuotes Ltd.

```
   vectorf&              rcondv                  // vector of reciprocal condition
numbers for each eigenvector
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSolverX(
   ENUM_EIG_BALANCE      balance,                // input matrix balancing method
   ENUM_EIG_VECTORS      jobv,                   // determines computation of right
and left eigenvectors
   ENUM_EIG_SENSE        sense,                  // determines computation of
reciprocal condition numbers
   vectorc&              eigen_values,           // vector of computed eigenvalues
   matrixc&              left_eigenvectors,      // matrix of computed left vectors
   matrixc&              right_eigenvectors      // matrix of computed right
vectors
   matrixc&              shur_matrix,            // balanced matrix in Schur form
   long&                 ilo,                    // subscript of balanced matrix
   long&                 ihi,                    // superscript of balanced matrix
   vector&               scale,                  // details of permutations and
scaling when balancing the input matrix
   double&               ab_norm,                // 1-norm of balanced matrix
   vector&               rconde,                 // vector of reciprocal condition
numbers for each eigenvalue
   vector&               rcondv                  // vector of reciprocal condition
numbers for each eigenvector
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSolverX(
   ENUM_EIG_BALANCE      balance,                // input matrix balancing method
   ENUM_EIG_VECTORS      jobv,                   // determines computation of right
and left eigenvectors
   ENUM_EIG_SENSE        sense,                  // determines computation of
reciprocal condition numbers
   vectorcf&             eigen_values,           // vector of computed eigenvalues
   matrixcf&             left_eigenvectors,      // matrix of computed left vectors
   matrixcf&             right_eigenvectors      // matrix of computed right
vectors
   matrixcf&             shur_matrix,            // balanced matrix in Schur form
   long&                 ilo,                    // subscript of balanced matrix
   long&                 ihi,                    // superscript of balanced matrix
   vectorf&              scale,                  // details of permutations and
scaling when balancing the input matrix
   float&                ab_norm,                // 1-norm of balanced matrix
   vectorf&              rconde,                 // vector of reciprocal condition
numbers for each eigenvalue
   vectorf&              rcondv                  // vector of reciprocal condition
numbers for each eigenvector
   );
```

© 2000–2025, MetaQuotes Ltd.

Parameters

balance

[in] Value from the ENUM_EIG_BALANCE enumeration which determines the need and method for balancing the input matrix; it is used to improve the conditioning of the eigenvalues and eigenvectors.

jobv

[in] ENUM_EIG_VECTORS enumeration value which determines the method for computing left and right eigenvectors.

sense

[in] Value from the ENUM_EIG_SENSE enumeration determining the need to compute reciprocal condition numbers.

eigen_values

[out] Vector of eigenvalues.

left_eigenvectors

[out] Matrix of left eigenvectors.

right_eigenvectors

[out] Matrix of right eigenvectors.

shur_matrix

[out] Balanced matrix in Schur form; the matrix is not filled if neither left nor right eigenvectors are computed.

ilo

[out] Subscript of the balanced matrix; the matrix is not filled if no balancing is applied.

ihi

[out] Superscript of the balanced matrix; the matrix is not filled if no balancing is applied.

scale

[out] Vector of details of permutations and scaling when balancing the input matrix.

Details of the permutations and scaling factors applied when balancing A.

If P(j) is the index of the row and column interchanged with row and column j, and D(j) is the scaling factor applied to row and column j, then

scale(j) = P(j), for j = 1,...,ilo-1

= D(j), for j = ilo,...,ihi

= P(j) for j = ihi+1,..., n.

The order in which the interchanges are made is n to ihi+1, then 1 to ilo-1.

ab_norm

[out] 1-norm of the balanced matrix (the maximum of the sum of absolute values of elements in any of the matrix columns).

rconde

© 2000–2025, MetaQuotes Ltd.

[out] Vector of reciprocal condition numbers for each eigenvalue; it is computed if the 'sense' parameter is set to 'E' or 'B'.

rcondv

[out] Vector of reciprocal condition numbers for each eigenvector; it is computed if the 'sense' parameter is set to 'V' or 'B.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Input matrix balancing depends on the value of the '*balance*' parameter.

ENUM_EIG_BALANCE

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGBALANCE_N	Do not diagonally scale or permute
EIGBALANCE_P	Perform permutations to make the matrix more nearly upper triangular. Do not diagonally scale
EIGBALANCE_S	Diagonally scale the matrix. Do not permute
EIGBALANCE_B	Both diagonally scale and permute

ENUM_EIG_VECTORS

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVECTORS_N	Only eigenvalues are computed, without vectors.
EIGVECTORS_L	Only left eigenvectors are computed.
EIGVECTORS_R	Only right eigenvectors are computed.
EIGVECTORS_LR	Left and right eigenvectors are computed, eigenvalues are always computed.

ENUM_EIG_SENSE

An enumeration defining the need to compute eigenvectors.

© 2000–2025, MetaQuotes Ltd.

ID	Description
EIGSENSE_N	None of reciprocal condition numbers are computed
EIGSENSE_E	Computed for eigenvalues only
EIGSENSE_V	Computed for right eigenvectors only
EIGSENSE_B	Computed for eigenvalues and right eigenvectors

© 2000–2025, MetaQuotes Ltd.

EigenSolverShur

Compute eigenvalues, upper triangular matrix in Schur form, and matrix of Schur vectors (lapack function GEES). See also Schur decomposition.

Computing for type matrix<double>

```
bool  matrix::EigenSolverShur(
   ENUM_EIG_VECTORS      jobvs,                  // method to compute Shur vectors
   vectorc&              eigen_values,           // vector of computed eigenvalues
   matrix&               shur_matrix,            // matrix in Schur form
   matrix&               shur_vectors            // matrix of Schur vectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSolverShur(
   ENUM_EIG_VECTORS      jobvs,                  // method to compute Shur vectors
   vectorcf&             eigen_values,           // vector of computed eigenvalues
   matrixf&              shur_matrix,            // matrix in Schur form
   matrixf&              shur_vectors            // matrix of Schur vectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSolverShur(
   ENUM_EIG_VECTORS      jobvs,                  // method to compute Shur vectors
   vectorc&              eigen_values,           // vector of computed eigenvalues
   matrixc&              shur_matrix,            // matrix in Schur form
   matrixc&              shur_vectors            // matrix of Schur vectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSolverShur(
   ENUM_EIG_VECTORS      jobvs,                  // method to compute Shur vectors
   vectorcf&             eigen_values,           // vector of computed eigenvalues
   matrixcf&             shur_matrix,            // matrix in Schur form
   matrixcf&             shur_vectors            // matrix of Schur vectors
   );
```

Parameters

jobvs

[in] Value from the ENUM_EIG_SHUR enumeration, which defines the method for computing Shur vectors.

eigen_values

[out] Vector of eigenvalues.

shur_matrix

[out] Upper triangular Schur matrix (Schur form for the input matrix).

© 2000–2025, MetaQuotes Ltd.

shur_vectors

[out] Matrix of Schur vectors; it is not computed if the *jobvs* parameter is set to N.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the *jobvs* parameter values.

Real (non-complex) matrices can have a complex solution. Therefore, the input vector of eigenvalues must be complex. In case of a complex solution, the error code is set to 4019 (ERR_MATH_OVERFLOW). Otherwise, only the real parts of the complex values of the eigenvalue vector should be used.

EigenSolverShur

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGSHUR_N	Schur vectors are not computed
EIGSHUR_V	Schur vectors are computed

© 2000–2025, MetaQuotes Ltd.

EigenSolver2

Compute generalized eigenvalues and eigenvectors for a pair of ordinary square matrices (lapack function GGEV). Both matrices must be the same size.

Computing for type matrix<double>

```
bool  matrix::EigenSolver2(
   matrix&               B,                    // second matrix in the pair
   ENUM_EIG_VECTORS      jobv,                 // method to compute right and
left vectors
   vectorc&              alpha,                // vector of computed eigenvalues
   vector&               beta,                 // vector of eigenvalue divisors
   matrix&               left_eigenvectors,    // matrix of computed left vectors
   matrix&               right_eigenvectors    // matrix of computed right
vectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSolver2(
   matrix&               B,                    // second matrix in the pair
   ENUM_EIG_VECTORS      jobv,                 // method to compute right and
left vectors
   vectorcf&             alpha,                // vector of computed eigenvalues
   vectorf&              beta,                 // vector of eigenvalue divisors
   matrixf&              left_eigenvectors,    // matrix of computed left vectors
   matrixf&              right_eigenvectors    // matrix of computed right
vectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSolver2(
   matrixc&              B,                    // second matrix in the pair
   ENUM_EIG_VECTORS      jobv,                 // method to compute right and
left vectors
   vectorc&              alpha,                // vector of computed eigenvalues
   vectorc&              beta,                 // vector of eigenvalue divisors
   matrixc&              left_eigenvectors,    // matrix of computed left vectors
   matrixc&              right_eigenvectors    // matrix of computed right
vectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSolver2(
   matrixcf&             B,                    // second matrix in the pair
   ENUM_EIG_VECTORS      jobv,                 // method to compute right and
left vectors
   vectorcf&             alpha,                // vector of computed eigenvalues
   vectorcf&             beta,                 // vector of eigenvalue divisors
```

© 2000–2025, MetaQuotes Ltd.

```
   matrixcf&             left_eigenvectors,       // matrix of computed left vectors
   matrixcf&             right_eigenvectors       // matrix of computed right
vectors
   );
```

Parameters

B

[in] The second matrix in the pair.

jobv

[in] ENUM_EIG_VECTORS enumeration value which determines the method for computing left and right eigenvectors.

alpha

[out] Vector of eigenvalues.

beta

[out] Vector of eigen value divisors.

left_eigenvectors

[out] Matrix of left eigenvectors.

right_eigenvectors

[out] Matrix of right eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the value of the jobv parameter.

ENUM_EIG_VECTORS

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVECTORS_N	Only eigenvalues are computed, without vectors.
EIGVECTORS_L	Only left eigenvectors are computed.
EIGVECTORS_R	Only right eigenvectors are computed.
EIGVECTORS_LR	Left and right eigenvectors are computed, eigenvalues are always computed.

A generalized eigenvalue for a pair of matrices (A,B) is a scalar lambda or a ratio alpha/beta = lambda, such that A - lambda*B is singular. It is usually represented as the pair (alpha,beta), as there is a reasonable interpretation for beta=0, and even for both being zero.

The right eigenvector v(j) corresponding to the eigenvalue lambda(j) of (A,B) satisfies

© 2000–2025, MetaQuotes Ltd.

A * v(j) = lambda(j) * B * v(j).

The left eigenvector u(j) corresponding to the eigenvalue lambda(j) of (A,B) satisfies

u(j)**H * A = lambda(j) * u(j)**H * B .

where u(j)**H is the conjugate-transpose of u(j).

Real (non-complex) matrices can have a complex solution. Therefore, the input vector of eigenvalues must be complex. In case of a complex solution, the error code is set to 4019 (ERR_MATH_OVERFLOW). Otherwise, only the real parts of the complex values of the eigenvalue vector should be used.

© 2000–2025, MetaQuotes Ltd.

EigenSolver2X

Compute generalized eigenvalues and eigenvectors for a pair of regular square matrices in Expert mode, i.e. with the ability to influence the computation algorithm and the ability to obtain accompanying computation data (lapack function GGEVX). Both matrices must be the same size.

Optionally, it also computes a balancing transformation to improve the conditioning of the eigenvalues and eigenvectors (ILO, IHI, LSCALE, RSCALE, ABNRM, and BBNRM), reciprocal condition numbers for the eigenvalues (RCONDE), and reciprocal condition numbers for the right eigenvectors (RCONDV).

Computing for type matrix<double>

```
bool  matrix::EigenSolver2X(
   matrix&               B,                    // second matrix in the pair
   ENUM_EIG_BALANCE      balance,              // input matrix balancing method
   ENUM_EIG_VECTORS      jobv,                 // determines computation of right
and left eigenvectors
   ENUM_EIG_SENSE        sense,                // determines computation of
reciprocal condition numbers
   vectorc&              alpha,                // vector of computed eigenvalues
   vector&               beta,                 // vector of eigenvalue divisors
   matrix&               left_eigenvectors,    // matrix of computed left vectors
   matrix&               right_eigenvectors    // matrix of computed right
vectors
   matrix&               shur_matrix1,         // the first part of the real
Schur form of the "balanced" versions of the input A and B
   matrix&               shur_matrix2,         // the second part of the real
Schur form of the "balanced" versions of the input A and B
   long&                 ilo,                  // subscript of balanced matrix
   long&                 ihi,                  // superscript of balanced matrix
   vector&               lscale,               // details of the permutations and
scaling factors applied to the left side of A and B
   vector&               rscale,               // details of the permutations and
scaling factors applied to the right side of A and B
   double&               ab_norm,              // one-norm of balanced input
matrix
   double&               bb_norm,              // one-norm of balanced second
matrix B
   vector&               rconde,               // vector of reciprocal condition
numbers for each eigenvalue
   vector&               rcondv                // vector of reciprocal condition
numbers for each eigenvector
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSolver2X(
   matrixc&              B,                    // second matrix in the pair
   ENUM_EIG_BALANCE      balance,              // input matrix balancing method
   ENUM_EIG_VECTORS      jobv,                 // determines computation of right
and left eigenvectors
```

© 2000–2025, MetaQuotes Ltd.

```
   ENUM_EIG_SENSE        sense,                  // determines computation of
reciprocal condition numbers
   vectorc&              alpha,                  // vector of computed eigenvalues
   vectorc&              beta,                   // vector of eigenvalue divisors
   matrixc&              left_eigenvectors,      // matrix of computed left vectors
   matrixc&              right_eigenvectors      // matrix of computed right
vectors
   matrixc&              shur_matrix1,           // the first part of the real
Schur form of the "balanced" versions of the input A and B
   matrixc&              shur_matrix2,           // the second part of the real
Schur form of the "balanced" versions of the input A and B
   long&                 ilo,                    // subscript of balanced matrix
   long&                 ihi,                    // superscript of balanced matrix
   vector&               lscale,                 // details of the permutations and
scaling factors applied to the left side of A and B
   vector&               rscale,                 // details of the permutations and
scaling factors applied to the right side of A and B
   double&               ab_norm,                // one-norm of balanced input
matrix
   double&               bb_norm,                // one-norm of balanced second
matrix B
   vector&               rconde,                 // vector of reciprocal condition
numbers for each eigenvalue
   vector&               rcondv                  // vector of reciprocal condition
numbers for each eigenvector
   );
```

Parameters

B

[in] The second matrix in the pair.

balance

[in] Value from the ENUM_EIG_BALANCE enumeration which determines the need and method for balancing the input matrix; it is used to improve the conditioning of the eigenvalues and eigenvectors.

jobv

[in] Value from the ENUM_EIG_VECTORS enumeration which determines the method for computing left and right eigenvectors.

sense

[in] Value from the ENUM_EIG_SENSE enumeration which determines the need to compute reciprocal condition numbers.

eigen_values

[out] Vector of eigenvalues.

left_eigenvectors

[out] Matrix of left eigenvectors.

right_eigenvectors

[out] Matrix of right eigenvectors.

© 2000–2025, MetaQuotes Ltd.

shur_matrix1, shur_matrix2

[out] 2 parts of balanced matrix in Schur form; the matrix is not filled if neither left nor right eigenvectors are computed.

ilo

[out] Subscript of the balanced matrix; the matrix is not filled if no balancing is applied.

ihi

[out] Superscript of the balanced matrix; the matrix is not filled if no balancing is applied.

lscale

[out] Vector contains details of the permutations and scaling factors applied to the left side of A and B.

If PL(j) is the index of the row interchanged with row j, and DL(j) is the scaling factor applied to row j, then

lscale(j) = PL(j), for j = 1,..., ilo-1

= DL(j), for j = ilo,...,ihi

= PL(j) for j = ihi+1,..., n.

The order in which the interchanges are made is n to ihi+1, then 1 to ilo-1.

rscale

[out] Vector contains details of the permutations and scaling factors applied to the right side of A and B.

If PR(j) is the index of the column interchanged with column j, and DR(j) is the scaling factor applied to column j, then

rscale(j) = PR(j), for j = 1,..., ilo-1

= DR(j), for j = ilo,...,ihi

= PR(j) for j = ihi+1,..., n.

The order in which the interchanges are made is n to ihi+1, then 1 to ilo-1.

ab_norm

[out] One-norm of the balanced input matrix (the maximum of the sum of absolute values of elements in any of the matrix columns).

bb_norm

[out] One-norm of balanced second matrix B.

rconde

[out] Vector of reciprocal condition numbers for each eigenvalue; it is computed if the 'sense' parameter is set to 'E' or 'B'.

rcondv

[out] Vector of reciprocal condition numbers for each eigenvector; it is computed if the 'sense' parameter is set to 'V' or 'B'.

Return Value

© 2000–2025, MetaQuotes Ltd.

The function returns 'true' on success or 'false' if an error occurs.

Note

Input matrices balancing depends on the value of the *balance* parameter.

ENUM_EIG_BALANCE

An enumeration that specifies whether the matrices should be balanced.

ID	Description
EIGBALANCE_N	Do not diagonally scale or permute.
EIGBALANCE_P	Perform permutations to make the matrix more nearly upper triangular. Do not diagonally scale.
EIGBALANCE_S	Diagonally scale the matrix. Do not permute.
EIGBALANCE_B	Both diagonally scale and permute.

ENUM_EIG_VECTORS

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVECTORS_N	Only eigenvalues are computed, without vectors.
EIGVECTORS_L	Only left eigenvectors are computed.
EIGVECTORS_R	Only right eigenvectors are computed.
EIGVECTORS_LR	Left and right eigenvectors are computed, eigenvalues are always computed.

ENUM_EIG_SENSE

An enumeration determining the need to compute reciprocal condition numbers.

ID	Description
EIGSENSE_N	None of reciprocal condition numbers are computed.
EIGSENSE_E	Computed for eigenvalues only.
EIGSENSE_V	Computed for right eigenvectors only.
EIGSENSE_B	Computed for eigenvalues and right eigenvectors.

© 2000–2025, MetaQuotes Ltd.

EigenSolver2Shur

Compute a pair of ordinary square matrices of generalized eigenvalues, generalized eigenvectors, generalized Schur forms, as well as left and right Schur vectors (lapack function GGES).

Computes the generalized eigenvalues, the generalized real/complex Schur form (S,T), optionally, the left and/or right matrices of Schur vectors (vsl and vsr) for a pair of n-by-n real/complex nonsymmetric matrices (A,B). This gives the generalized Schur factorization:

(A,B) = (vsl*S *vsrH, vsl*T*vsrH)

Optionally, it also orders the eigenvalues so that a selected cluster of eigenvalues appears in the leading diagonal blocks of the upper quasi-triangular matrix S and the upper triangular matrix T. The leading columns of vsl and vsr then form an orthonormal/unitary basis for the corresponding left and right eigenspaces (deflating subspaces).

Computing for type matrix<double>

```
bool  matrix::EigenSolver2Shur(
   matrix&                B,         // second matrix in the pair
   ENUM_EIG_VECTORS       jobvs,     // method to compute left and right vectors
   vector&                alpha,     // vector of computed eigenvalues
   vector&                beta,      // vector of eigenvalue divisors
   matrix&                shur_s,    // matrix S in Schur form
   matrix&                shur_t,    // matrix T in Schur form
   matrix&                vsl,       // matrix of left Schur vectors vsl
   matrix&                vsr        // matrix of right Schur vectors vsr
   );
```

Computing for type matrix<complex>

```
bool  matrix::EigenSolver2Shur(
   matrixc&               B,         // second matrix in the pair
   ENUM_EIG_VECTORS       jobvs,     // method to compute left and right vectors
   vectorc&               alpha,     // vector of computed eigenvalues
   vectorc&               beta,      // vector of eigenvalue divisors
   matrixc&               shur_s,    // matrix S in Schur form
   matrixc&               shur_t,    // matrix T in Schur form
   matrixc&               vsl,       // matrix of left Schur vectors vsl
   matrixc&               vsr        // matrix of right Schur vectors vsr
   );
```

Parameters

B

[out] The second matrix in the pair.

jobvs

[in] ENUM_EIG_VECTORS enumeration value which determines the method for computing left and right eigenvectors.

alpha

© 2000–2025, MetaQuotes Ltd.

[out] Vector of eigenvalues.

beta

[out] Vector of eigenvalue divisors.

shur_s

[out] Matrix S, block upper triangular Schur matrix (Schur form for the input matrix).

shur_t

[out] Matrix T, block upper triangular Schur matrix (Schur form for the second matrix in the pair).

vsl

[out] Matrix of left Schur vectors.

vsr

[out] Matrix of right Schur vectors.

Return Value

The function returns 'true' on success or 'false' if an error occurs.

Note

Computation depends on the *jobvs* parameter values.

The second matrix in the pair must be the same size as the first (input) one.

Real (non-complex) matrices can have a complex solution. Therefore, the vector of eigenvalues must be complex. In case of a complex solution, the error code is set to 4019 (ERR_MATH_OVERFLOW). Otherwise, only the real parts of the complex values of the eigenvalue vector should be used.

EigenSolverShur

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVECTORS_N	Only eigenvalues are computed, without vectors.
EIGVECTORS_L	Only left eigenvectors are computed.
EIGVECTORS_R	Only right eigenvectors are computed.
EIGVECTORS_LR	Left and right eigenvectors are computed, eigenvalues are always computed.

© 2000–2025, MetaQuotes Ltd.

EigenSolver2Blocked

Compute generalized eigenvalues and eigenvectors for a pair of regular square matrices using a block algorithm (lapack function GGEV3). Both matrices must be the same size. The method parameters are exactly the same as EigenSolver2.

Computing for type matrix<double>

```
bool  matrix::EigenSolver2Blocked(
   matrix&               B,                    // second matrix in the pair
   ENUM_EIG_VECTORS      jobv,                 // method to compute right and
left vectors
   vectorc&              alpha,                // vector of computed eigenvalues
   vector&               beta,                 // vector of eigenvalue divisors
   matrix&               left_eigenvectors,    // matrix of computed left vectors
   matrix&               right_eigenvectors    // matrix of computed right
vectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSolver2Blocked(
   matrix&               B,                    // second matrix in the pair
   ENUM_EIG_VECTORS      jobv,                 // method to compute right and
left vectors
   vectorcf&             alpha,                // vector of computed eigenvalues
   vectorf&              beta,                 // vector of eigenvalue divisors
   matrixf&              left_eigenvectors,    // matrix of computed left vectors
   matrixf&              right_eigenvectors    // matrix of computed right
vectors
   );
```

Computing for type matrix<complex>

```
bool  matrix::EigenSolver2Blocked(
   matrixc&              B,                    // second matrix in the pair
   ENUM_EIG_VECTORS      jobv,                 // method to compute right and
left vectors
   vectorc&              alpha,                // vector of computed eigenvalues
   vectorc&              beta,                 // vector of eigenvalue divisors
   matrixc&              left_eigenvectors,    // matrix of computed left vectors
   matrixc&              right_eigenvectors    // matrix of computed right
vectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSolver2(
   matrixcf&             B,                    // second matrix in the pair
   ENUM_EIG_VECTORS      jobv,                 // method to compute right and
left vectors
   vectorcf&             alpha,                // vector of computed eigenvalues
```

© 2000–2025, MetaQuotes Ltd.

```
   vectorcf&            beta,                 // vector of eigenvalue divisors
   matrixcf&            left_eigenvectors,    // matrix of computed left vectors
   matrixcf&            right_eigenvectors    // matrix of computed right
vectors
   );
```

Parameters

B

[out] The second matrix in the pair.

jobv

[in] ENUM_EIG_VECTORS enumeration value which determines the method for computing left and right eigenvectors.

alpha

[out] Vector of eigenvalues.

beta

[out] Vector of eigenvalue divisors.

left_eigenvectors

[out] Matrix of left eigenvectors.

righeft_eigenvectors

[out] Matrix of right eigenvectors.

Return Value

The function returns 'true' on success or 'false' if an error occurs.

Note

Computation depends on the value of the jobv parameter.

A generalized eigenvalue for a pair of matrices (A,B) is a scalar lambda or a ratio alpha/beta = lambda, such that A - lambda*B is singular. It is usually represented as the pair (alpha,beta), as there is a reasonable interpretation for beta=0, and even for both being zero.

The right eigenvector v(j) corresponding to the eigenvalue lambda(j) of (A,B) satisfies:

A * v(j) = lambda(j) * B * v(j).

The left eigenvector u(j) corresponding to the eigenvalue lambda(j) of (A,B) satisfies:

u(j)**H * A = lambda(j) * u(j)**H * B .

where u(j)**H is the conjugate-transpose of u(j).

Real (non-complex) matrices can have a complex solution. Therefore, the input vector of eigenvalues must be complex. In case of a complex solution, the error code is set to 4019 (ERR_MATH_OVERFLOW). Otherwise, only the real parts of the complex values of the eigenvalue vector should be used.

© 2000–2025, MetaQuotes Ltd.

ENUM_EIG_VECTORS

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVECTORS_N	Only eigenvalues are computed, without vectors.
EIGVECTORS_L	Only left eigenvectors are computed.
EIGVECTORS_R	Only right eigenvectors are computed.
EIGVECTORS_LR	Left and right eigenvectors are computed, eigenvalues are always computed.

© 2000–2025, MetaQuotes Ltd.

EigenSolver2ShurBlocked

Compute a pair of regular square matrices of generalized eigenvalues, generalized eigenvectors, generalized Schur forms, as well as left and right Schur vectors (lapack function GGES3).

Computes the generalized eigenvalues, the generalized real/complex Schur form (S,T), optionally, the left and/or right matrices of Schur vectors (VSL and VSR) for a pair of n-by-n real/complex nonsymmetric matrices (A,B). This gives the generalized Schur factorization:

(A,B) = (vsl*S *vsrH, vsl*T*vsrH)

Optionally, it also orders the eigenvalues so that a selected cluster of eigenvalues appears in the leading diagonal blocks of the upper quasi-triangular matrix S and the upper triangular matrix T. The leading columns of vsl and vsr then form an orthonormal/unitary basis for the corresponding left and right eigenspaces (deflating subspaces).

Computing for type matrix<double>

```
bool  matrix::EigenSolver2ShurBlocked(
   matrix&                B,         // second matrix in the pair
   ENUM_EIG_VECTORS       jobvs,     // method to compute left and right vectors
   vectorc&               alpha,     // vector of computed eigenvalues
   vector&                beta,      // vector of eigenvalue divisors
   matrix&                shur_s,    // matrix S in Schur form
   matrix&                shur_t,    // matrix T in Schur form
   matrix&                vsl,       // matrix of left Schur vectors VSL
   matrix&                vsr        // matrix of right Schur vectors VSR
   );
```

Computing for type matrix<float>

```
bool  matrix::EigenSolver2ShurBlocked(
   matrixf&               B,         // second matrix in the pair
   ENUM_EIG_VECTORS       jobvs,     // method to compute left and right vectors
   vectorcf&              alpha,     // vector of computed eigenvalues
   vectorf&               beta,      // vector of eigenvalue divisors
   matrixf&               shur_s,    // matrix S in Schur form
   matrixf&               shur_t,    // matrix T in Schur form
   matrixf&               vsl,       // matrix of left Schur vectors VSL
   matrixf&               vsr        // matrix of right Schur vectors VSR
   );
```

Computing for type matrix<complex>

```
bool  matrix::EigenSolver2ShurBlocked(
   matrixc&               B,         // second matrix in the pair
   ENUM_EIG_VECTORS       jobvs,     // method to compute left and right vectors
   vectorc&               alpha,     // vector of computed eigenvalues
   vectorc&               beta,      // vector of eigenvalue divisors
   matrixc&               shur_s,    // matrix S in Schur form
   matrixc&               shur_t,    // matrix T in Schur form
```

© 2000–2025, MetaQuotes Ltd.

```
   matrixc&                vsl,      // matrix of left Schur vectors VSL
   matrixc&                vsr       // matrix of right Schur vectors VSR
   );
```

Computing for type matrix<complexf>

```
bool  matrix::EigenSolver2ShurBlocked(
   matrixcf&               B,        // second matrix in the pair
   ENUM_EIG_VECTORS        jobvs,    // method to compute left and right vectors
   vectorcf&               alpha,    // vector of computed eigenvalues
   vectorcf&               beta,     // vector of eigenvalue divisors
   matrixcf&               shur_s,   // matrix S in Schur form
   matrixcf&               shur_t,   // matrix T in Schur form
   matrixcf&               vsl,      // matrix of left Schur vectors VSL
   matrixcf&               vsr       // matrix of right Schur vectors VSR
   );
```

Parameters

B

[in] The second matrix in the pair.

jobvs

[in] ENUM_EIG_VECTORS enumeration value which determines the method for computing left and right eigenvectors.

alpha

[out] Vector of eigenvalues.

beta

[out] Vector of eigenvalue divisors.

shur_s

[out] Matrix S, block upper triangular Schur matrix (Schur form for the input matrix).

shur_t

[out] Matrix T, block upper triangular Schur matrix (Schur form for the second matrix in the pair).

vsl

[out] Matrix of left Schur vectors VSL.

vsr

[out] Matrix of right Schur vectors VSR.

Return Value

The function returns 'true' on success or 'false' if an error occurs.

Note

Computation depends on the *jobvs* parameter values.

© 2000–2025, MetaQuotes Ltd.

The second matrix in the pair must be the same size as the first (input) one.

Real (non-complex) matrices can have a complex solution. Therefore, the vector of eigenvalues must be complex. In case of a complex solution, the error code is set to 4019 (ERR_MATH_OVERFLOW). Otherwise, only the real parts of the complex values of the eigenvalue vector should be used.

EigenSolverShur

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVECTORS_N	Only eigenvalues are computed, without vectors.
EIGVECTORS_L	Only left eigenvectors are computed.
EIGVECTORS_R	Only right eigenvectors are computed.
EIGVECTORS_LR	Left and right eigenvectors are computed, eigenvalues are always computed.

© 2000–2025, MetaQuotes Ltd.

Symmetric Matrices

Functions for computing eigenvalues and eigenvectors of symmetric or Hermitian matrices using the divide and conquer algorithm, making the process efficient and fast. These methods can be applied to matrices of different data types, including real and complex numbers.

Function	Action
EigenSymmetricDC	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the divide-and-conquer algorithm (lapack functions SYEVD, HEEVD).
EigenSymmetricQR	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the QR algorithm (lapack functions SYEV, HEEV).
EigenSymmetricRobust	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the Multiple Relatively Robust Representations, MRRR algorithm (lapack functions SYEVR, HEEVR).
EigenSymmetricBisect	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the bisection algorithm (lapack functions SYEVX, HEEVX).

© 2000–2025, MetaQuotes Ltd.

EigenSymmetricDC

Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugated) matrix using the divide-and-conquer algorithm (lapack functions SYEVD, HEEVD).

Computing for type matrix<double>

```
bool  matrix::EigenSymmetricDC(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vector&               eigen_values,      // vector of computed eigenvalues
   matrix&               eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSymmetricDC(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vectorf&              eigen_values,      // vector of computed eigenvalues
   matrixf&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSymmetricDC(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vector&               eigen_values,      // vector of computed eigenvalues
   matrixc&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSymmetricDC(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vectorf&              eigen_values,      // vector of computed eigenvalues
   matrixcf&             eigen_vectors      // matrix of computed eigenvectors
   );
```

Parameters

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

eigen_values

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

© 2000–2025, MetaQuotes Ltd.

Note

Computation depends on the value of the jobv parameter.

When jobv = EIGVALUES_V, eigenvectors and eigenvalues are calculated.

If EIGVALUES_N is set, eigenvectors are not calculated. Only eigenvalues are computed.

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

ENUM_EIG_VALUES

An enumeration that specifies whether to calculate eigenvectors.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated.
EIGVALUES_N	Only eigenvalues are calculated, without vectors.

© 2000–2025, MetaQuotes Ltd.

EigenSymmetricQR

Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugated) matrix using the QR algorithm (lapack functions SYEV, HEEV).

Computing for type matrix<double>

```
bool  matrix::EigenSymmetricQR(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vector&               eigen_values,      // vector of computed eigenvalues
   matrix&               eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSymmetricQR(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vectorf&              eigen_values,      // vector of computed eigenvalues
   matrixf&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSymmetricQR(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vector&               eigen_values,      // vector of computed eigenvalues
   matrixc&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSymmetricQR(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vectorf&              eigen_values,      // vector of computed eigenvalues
   matrixcf&             eigen_vectors      // matrix of computed eigenvectors
   );
```

Parameters

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

eigen_values

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

© 2000–2025, MetaQuotes Ltd.

Note

Computation depends on the value of the jobv parameter.

When jobv = EIGVALUES_V, eigenvectors and eigenvalues are calculated.

If EIGVALUES_N is set, eigenvectors are not calculated. Only eigenvalues are computed.

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

ENUM_EIG_VALUES

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated.
EIGVALUES_N	Only eigenvalues are computed, without vectors.

© 2000–2025, MetaQuotes Ltd.

EigenSymmetricRobust

Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugated) matrix using the Multiple Relatively Robust Representations, MRRR algorithm (lapack functions SYEVR, HEEVR).

Computing for type matrix<double>

```
bool  matrix::EigenSymmetricRobust(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   double                lower,             // lower bound of the subset
   double                upper,             // Upper bound of the subset
   double                abstol,            // absolute error tolerance
   vector&               eigen_values,      // vector of computed eigenvectors
   matrix&               eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSymmetricRobust(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   float                 lower,             // lower bound of the subset
   float                 upper,             // upper bound of the subset
   float                 abstol,            // absolute error tolerance
   vectorf&              eigen_values,      // vector of computed eigenvectors
   matrixf&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSymmetricRobust(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   double                lower,             // lower bound of the subset
   double                upper,             // Upper bound of the subset
   double                abstol,            // absolute error tolerance
   vector&               eigen_values,      // vector of computed eigenvectors
   matrixc&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSymmetricRobust(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   float                 lower,             // lower bound of the subset
   float                 upper,             // Upper bound of the subset
   float                 abstol,            // absolute error tolerance
```

© 2000–2025, MetaQuotes Ltd.

```
   vectorf&              eigen_values,       // vector of computed eigenvectors
   matrixcf&             eigen_vectors       // matrix of computed eigenvectors
   );
```

Parameters

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

range

[in] ENUM_BLAS_RANGE enumeration value that defines a subset of computable eigenvalues and vectors.

lower

[in] The lower bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

upper

[in] The upper bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

abstol

[in] Absolute error tolerance.

The absolute error tolerance to which each eigenvalue/eigenvector is required.

If jobv = 'V', the eigenvalues and eigenvectors output have residual norms bounded by abstol, and the dot products between different eigenvectors are bounded by abstol.

If abstol < n *eps*|T|, then n *eps*|T| is used instead, where eps is the machine precision, and |T| is the 1-norm of the matrix T. The eigenvalues are computed to an accuracy of eps*|T| irrespective of abstol.

If high relative precision is important, 'abstol' should be set to a safe minimum value X such that 1.0/X does not overflow.

eigen_values

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the values of the *jobv* and *range* parameters.

When BLASRANGE_A is set, all eigenvalues are computed, and the lower and upper parameters are ignored.

With the BLASRANGE_V value, only those eigenvalues (and their vectors) are computed, which fall within the range of real values specified by the 'lower' and 'upper' parameters.

© 2000–2025, MetaQuotes Ltd.

With the BLASRANGE_I value, only those eigenvalues (and their vectors) are computed, which fall within the range of integer indices specified by the 'lower' and 'upper' parameters. For example, with lower=0 and upper=2, only the first three eigenvalues are computed.

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

ENUM_EIG_VALUES

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated.
EIGVALUES_N	Only eigenvalues are computed, without vectors.

ENUM_BLAS_RANGE

An enumeration defining how right singular vectors should be computed.

ID	Description
BLASRANGE_A	All singular or eigenvalues will be found.
BLASRANGE_V	All singular or eigenvalues in the half-open interval (VL,VU] will be found.
BLASRANGE_I	The IL-th through IU-th singular or eigenvalues will be found.

© 2000–2025, MetaQuotes Ltd.

EigenSymmetricBisect

Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugated) matrix using the bisection algorithm (lapack functions SYEVX, HEEVX).

Computing for type matrix<double>

```
bool  matrix::EigenSymmetricBisect(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   double                lower,             // lower bound of the subset
   double                upper,             // Upper bound of the subset
   double                abstol,            // absolute error tolerance
   vector&               eigen_values,      // vector of computed eigenvectors
   matrix&               eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSymmetricBisect(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   float                 lower,             // lower bound of the subset
   float                 upper,             // upper bound of the subset
   float                 abstol,            // absolute error tolerance
   vectorf&              eigen_values,      // vector of computed eigenvectors
   matrixf&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSymmetricBisect(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   double                lower,             // lower bound of the subset
   double                upper,             // Upper bound of the subset
   double                abstol,            // absolute error tolerance
   vector&               eigen_values,      // vector of computed eigenvectors
   matrixc&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSymmetricBisect(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   float                 lower,             // lower bound of the subset
   float                 upper,             // Upper bound of the subset
   float                 abstol,            // absolute error tolerance
   vectorf&              eigen_values,      // vector of computed eigenvectors
```

© 2000–2025, MetaQuotes Ltd.

```
   matrixcf&             eigen_vectors       // matrix of computed eigenvectors
   );
```

Parameters

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

range

[in] ENUM_BLAS_RANGE enumeration value that defines a subset of computable eigenvalues and vectors.

lower

[in] The lower bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

upper

[in] The upper bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

abstol

[in] Absolute error tolerance.

The absolute error tolerance to which each eigenvalue/eigenvector is required.

If jobv = 'V', the eigenvalues and eigenvectors output have residual norms bounded by abstol, and the dot products between different eigenvectors are bounded by abstol.

If abstol < n *eps*|T|, then n *eps*|T| is used instead, where eps is the machine precision, and |T| is the 1-norm of the matrix T. The eigenvalues are computed to an accuracy of eps*|T| irrespective of abstol.

If high relative precision is important, 'abstol' should be set to a safe minimum value X such that 1.0/X does not overflow.

eigen_values

[out] Vector of eigenvalues.

V

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the values of the *jobv* and *range* parameters.

When BLASRANGE_A is set, all eigenvalues are computed, and the lower and upper parameters are ignored.

With the BLASRANGE_V value, only those eigenvalues (and their vectors) are computed, which fall within the range of real values specified by the 'lower' and 'upper' parameters.

© 2000–2025, MetaQuotes Ltd.

With the BLASRANGE_I value, only those eigenvalues (and their vectors) are computed, which fall within the range of integer indices specified by the 'lower' and 'upper' parameters. For example, with lower=0 and upper=2, only the first three eigenvalues are computed.

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

ENUM_EIG_VALUES

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated.
EIGVALUES_N	Only eigenvalues are computed, without vectors.

ENUM_BLAS_RANGE

An enumeration defining how right singular vectors should be computed.

ID	Description
BLASRANGE_A	All singular or eigenvalues will be found.
BLASRANGE_V	All singular or eigenvalues in the half-open interval (VL,VU] will be found.
BLASRANGE_I	The IL-th through IU-th singular or eigenvalues will be found.

© 2000–2025, MetaQuotes Ltd.

Singular Spectrum Analysis

The section contains functions for decomposing a matrix into three components: orthogonal matrices and a diagonal matrix of singular values. SVD is applied to solve various linear algebra problems such as data dimensionality reduction, image compression, solving systems of equations, as well as data analysis and optimization. The main functions allow calculating singular values and vectors, reconstruct matrices, as well as approximate them with reduced rank accuracy.

Function	Action
SingularSpectrumAnalysisSpectrum	A method function for calculating the relative contributions of spectral components based on their eigenvalues.
SingularSpectrumAnalysisForecast	A method function for calculating reconstructed and predicted data using spectral components of the input time series.
SingularSpectrumAnalysisReconstructComponents	A method function for calculating reconstructed components of the input time series and their contributions.
SingularSpectrumAnalysisReconstructSeries	A method function for calculating the reconstructed time series using the first component_count components.

© 2000–2025, MetaQuotes Ltd.

SingularSpectrumAnalysisSpectrum

A method function for calculating the relative contributions of spectral components based on their eigenvalues.

Calculations for vector<double> type

```
bool  vector::SingularSpectrumAnalysisSpectrum(
   ulong   window_length,      // window size for constructing the trajectory matrix
   vector& spectrum            // vector of component contributions to the input
series (SSA spectrum)
   );
```

Calculations for vector<complex> type

```
bool  vector::SingularSpectrumAnalysisSpectrum(
   ulong   window_length,      // window size for constructing the trajectory matrix
   vectorc& spectrum           // vector of component contributions to the input
series (SSA spectrum)
   );
```

Parameters

window_length

[in] Window size for constructing the trajectory matrix, the number of components the input time series should be decomposed into.

spectrum

[out] Vector of component contributions to the input series - eigenvalues of the covariance matrix of the input time series.

Return Value

The function returns 'true' on success or 'false' if an error occurs.

Note

The *window_length* parameter value should be less than the size of the input time series. To construct a full-fledged trajectory matrix, the optimal size is considered to be approximately equal to half the size of the input time series.

© 2000–2025, MetaQuotes Ltd.

SingularSpectrumAnalysisForecast

A method function for calculating reconstructed and predicted data using spectral components of the input time series.

Calculations for vector<double> type

```
bool  vector::SingularSpectrumAnalysisForecast(
   ulong    window_length,         // window size for constructing the trajectory
matrix
   ulong    component_count,       // number of components used for forecasting
   ulong    forecast_horizon,      // number of points to forecast
   vector&  forecast               // vector consisting of reconstructed and
predicted values
   );
```

Calculations for vector<complex> type

```
bool  vector::SingularSpectrumAnalysisForecast(
   ulong    window_length,         // window size for constructing the trajectory
matrix
   ulong    component_count,       // number of components used for forecasting
   ulong    forecast_horizon,      // number of points to forecast
   vectorc& forecast               // vector consisting of reconstructed and
predicted values
   );
```

Parameters

window_length

[in] Window size for constructing the trajectory matrix, the number of components the input time series should be decomposed into.

component_count

[in] Number of components used for forecasting.

forecast_horizon

[in] Number of points to forecast.

forecast

[out] Combining points reconstructed by *component_count* plus *forecast_horizon* points predicted using the first *component_count* components. Thus, the *forecast* vector has the size of (T+forecast_horizon), where T is the input series length.

Return Value

The function returns 'true' on success or 'false' if an error occurs.

Note

The *window_length* parameter value should be less than the size of the input time series. To construct a full-fledged trajectory matrix, the optimal size is considered to be approximately equal to half the size of the input time series.

© 2000–2025, MetaQuotes Ltd.

SingularSpectrumAnalysisReconstructComponents

A method function for calculating reconstructed components of the input time series and their contributions.

Calculations for vector<double> and matrix<double> types

```
bool  vector::SingularSpectrumAnalysisReconstructComponents(
   ulong    window_length,      // window size for constructing the trajectory matrix
   matrix&  components,         // matrix of reconstructed components
   vector&  contributions       // vector of component contributions to the input series
   );
```

Calculations for vector<complex> and matrix<complex> types

```
bool  vector::SingularSpectrumAnalysisReconstructComponents(
   ulong    window_length,      // window size for constructing the trajectory matrix
   matrixc& components,         // matrix of reconstructed components
   vectorc& contributions       // vector of component contributions to the input series
   );
```

Parameters

window_length

[in] Window size for constructing the trajectory matrix, the number of components the input time series should be decomposed into.

components

[out] A matrix of reconstructed components, where each column describes a separate component.

contributions

[out] Vector of component contributions to the input series (eigenvalues of the covariance matrix of the input time series).

Return Value

The function returns 'true' on success or 'false' if an error occurs.

Note

The *window_length* parameter value should be less than the size of the input time series. To construct a full-fledged trajectory matrix, the optimal size is considered to be approximately equal to half the size of the input time series.

© 2000–2025, MetaQuotes Ltd.

SingularSpectrumAnalysisReconstructSeries

A method function for calculating the reconstructed time series using the first *component_count* components.

Calculations for vector<double> type

```
bool  vector::SingularSpectrumAnalysisReconstructSeries(
   ulong    window_length,       // window size for constructing the trajectory matrix
   ulong    component_count,     // number of components used for reconstruction
   vector&  reconstructed        // reconstructed time series
   );
```

Calculations for vector<complex> type

```
bool  vector::SingularSpectrumAnalysisReconstructSeries(
   ulong    window_length,       // window size for constructing the trajectory matrix
   ulong    component_count,     // number of components used for reconstruction
   vectorc& reconstructed        // reconstructed time series
   );
```

Parameters

window_length

[in] Window size for constructing the trajectory matrix, the number of components the input time series should be decomposed into.

component_count

[out] Number of components used for time series reconstruction.

reconstructed

[out] Vector containing the reconstructed output series.

Return Value

The function returns 'true' on success or 'false' if an error occurs.

Note

The *window_length* parameter value should be less than the size of the input time series. To construct a full-fledged trajectory matrix, the optimal size is considered to be approximately equal to half the size of the input time series.

© 2000–2025, MetaQuotes Ltd.

変換関数

これは、データフォーマットの変換を行う関数群です。

価格提示の必要な精度を提供する NormalizeDouble() 関数は特に留意されるべきです。トレーディング操作では、取引サーバによって必要とされた精度を少しでも超えた場合、非正規化価格は使用出来ません。

関数	アクション
CharToString	シンボルコードを1文字の文字列に変換します。
DoubleToString	指定された精度で数値をテキスト行に変換します。
EnumToString	任意のタイプの列挙値を文字列に変換します。
NormalizeDouble	浮動小数点数を指定された精度に丸めます。
StringToDouble	数のシンボル表現を含む文字列をdouble型の数値に変換します。
StringToInteger	数のシンボル表現を含む文字列をlong型の数値に変換します。
StringToTime	時間や日付を「yyyy.mm.dd [hh:mi]」フォーマットで含む文字列をdatetime型に変換します。
TimeToString	1970年1月1日からの経過秒を含む値を「yyyy.mm.dd hh:mi 」形式の文字列に変換します。
IntegerToString	int型を事前に決めた長さの文字列に変換します。
ShortToString	Unicodeシンボルコードを一シンボルの文字列に変換します。
ShortArrayToString	配列の一部を文字列に複製します。
StringToShortArray	文字列をシンボルごとにushort型の配列の一部に複製します。
CharArrayToString	ANSIシンボルコードを一シンボルの文字列に変換します。
StringToCharArray	Unicodeからansi,に変換された文字列をシンボルごとにuchar型の配列の一部に複製します。
CharArrayToStruct	uchar型の配列をPOD structureにコピーします
StructToCharArray	POD構造体をuchar型配列にコピーします
ColorToARGB	ARGB表現を受信するためのcolor型をuint型に変換します。
ColorToString	color値を「R,G,B」の文字列に変換します。
StringToColor	「R,G,B」の文字列またはカラー名を含む文字列をcolor型の値に変換します。
StringFormat	数を事前に決めたフォーマットの文字列に変換します。

参照

コードページの利用

© 2000-2025, MetaQuotes Ltd.

CharToString

シンボルコードを1文字の文字列に変換します。

```
string  CharToString(
   uchar  char_code      // シンボルの数値コード
   );
```

パラメータ

char_code

[in] ANSI シンボルのコード

戻り値

ANSI シンボルの文字列

例：

```
string ExtStrArray[224];

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- シンボルコード32（スペース）から255（UCHAR_MAX）までのループで、
//--- 現在のコードページに従って文字列に変換されたシンボルコードを配列に入力する
   for(int i=32; i<=UCHAR_MAX; i++)
      ExtStrArray[i-32]=CharToString((uchar)i);

//--- 操作ログにヘッダーとシンボル テーブルを出力する
   Print("Table of symbols:");
   ArrayPrint(ExtStrArray,_Digits," | ");
   /*
  結果:
   Table of symbols:
   [  0] " " | "!" | """ | "#" | "$" | "%" | "&" | "'" | "(" | ")" | "*" | "+" | ","
| "-" | "." | "/" | "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8"
   [ 25] "9" | ":" | ";" | "<" | "=" | ">" | "?" | "@" | "A" | "B" | "C" | "D" | "E"
| "F" | "G" | "H" | "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q"
   [ 50] "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z" | "[" | "\" | "]" | "^"
| "_" | "`" | "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" | "j"
   [ 75] "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" | "w"
| "x" | "y" | "z" | "{" | "|" | "}" | "~" | "□" | "Ђ" | "Ѓ" | "," | "ѓ"
   [100] "„" | "…" | "†" | "‡" | "€" | "‰" | "Љ" | "‹" | "Њ" | "Ќ" | "Ћ" | "Џ" | "ђ"
| "‘" | "’" | "“" | "”" | "•" | "–" | "—" | "" | "™" | "љ" | "›" | "њ"
   [125] "ќ" | "ћ" | "џ" | " " | "ў" | "Ў" | "Ј" | "¤" | "Ґ" | "¦" | "§" | "Ё" | "©"
| "Є" | "«" | "¬" | "" | "®" | "Ї" | "°" | "±" | "І" | "і" | "ґ" | "µ"
   [150] "¶" | "·" | "ё" | "№" | "є" | "»" | "ј" | "Ѕ" | "ѕ" | "ї" | "А" | "Б" | "В"
| "Г" | "Д" | "Е" | "Ж" | "З" | "И" | "Й" | "К" | "Л" | "М" | "Н" | "О"
```

© 2000-2025, MetaQuotes Ltd.

```
   [175] "П" | "Р" | "С" | "Т" | "У" | "Ф" | "Х" | "Ц" | "Ч" | "Ш" | "Щ" | "Ъ" | "Ы"
| "Ь" | "Э" | "Ю" | "Я" | "а" | "б" | "в" | "г" | "д" | "е" | "ж" | "з"
   [200] "и" | "й" | "к" | "л" | "м" | "н" | "о" | "п" | "р" | "с" | "т" | "у" | "ф"
| "х" | "ц" | "ч" | "ш" | "щ" | "ъ" | "ы" | "ь" | "э" | "ю" | "я"
   */
  }
```

参照

StringToCharArray、ShortToString、StringGetCharacter

© 2000–2025, MetaQuotes Ltd.

CharArrayToString

uchar 型の配列の一部を複製して戻り値の文字列に変換します。

```
string  CharArrayToString(
   uchar  array[],               // 配列
   int    start=0,               // 配列の開始位置
   int    count=-1               // シンボル数
   uint    codepage=CP_ACP       // コードページ
   );
```

パラメータ

array[]

[in] uchar 型の配列。

start=0

[in] コピー開始位置。デフォルトでは 0。

count=-1

[in] 複製する要素数。結果の文字列の長さを定義します。初期値は、配列の最後または終了する0まで複製することを意味する -1 です。

codepage=CP_ACP

[in] コードページの値。よく使用されるコードページでは、内蔵された定数がいくつもあります。

戻り値

文字列

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- シンボルコードで配列を作成する
   uchar array[]= { 84, 104, 105, 115,  32, 105, 115,  32,  97,  32,
                   116, 101, 115, 116,  32, 111, 102,  32, 116, 104,
                   101,  32,  67, 104,  97, 114,  65, 114, 114,  97,
                   121,  84, 111,  83, 116, 114, 105, 110, 103,  40,
                    41,  32, 102, 117, 110,  99, 116, 105, 111, 110
                  };
//--- 文字列に変換されたコードの配列を操作ログに出力する
   Print(CharArrayToString(array));

//--- シンボルコードと終端ゼロで配列を作成する
   uchar array_z[]= { 84, 104, 105, 115,  32, 105, 115,  32,  97,  32,
                     116, 101, 115, 116,   0,  32, 111, 102,  32, 116,
```

© 2000-2025, MetaQuotes Ltd.

```
                      104, 101,  32,  67, 104,  97, 114,  65, 114, 114,
                       97, 121,  84, 111,  83, 116, 114, 105, 110, 103,
                       40,  41,  32, 102, 117, 110,  99, 116, 105, 111, 110
                     };
//--- 操作ログに終端ゼロを持つコードの 2 番目の配列を出力する
   PrintFormat("%s ...", CharArrayToString(array_z));

   /*
  結果:
   This is a test of the CharArrayToString() function
   This is a test ...
   */
  }
```

参照

StringToCharArray、ShortArrayToString、コードページの利用

© 2000-2025, MetaQuotes Ltd.

CharArrayToStruct

uchar型の配列をPOD structureにコピーします。

```
bool  CharArrayToStruct(
   void&          struct_object,     // 構造体
   const uchar&   char_array[],      // 配列
   uint           start_pos=0        // 配列での開始位置
   );
```

パラメータ

struct_object

[in] 任意の型のPOD構造体への参照（単純なデータ型のみを含む）

char_array[]

[in] uchar型の配列。

start_pos=0

[in] 配列内のデータコピーが始まる位置

戻り値

成功の場合はtrue、それ以外の場合はfalse

例：

```
#define DATA_TOTAL 4

MqlRates ExtDataRates[DATA_TOTAL];
uchar    ExtCharArray[];

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 最後の4つのバーのデータをuchar配列にコピーする
   ResetLastError();
   for(int i=0; i<DATA_TOTAL; i++)
     {
      if(!CopyRatesToCharArray(i, ExtCharArray))
         return;
     }

//--- uchar配列からのデータ量に応じて、ループ内のすべての利用可能なデータをMql Rates構造体配列にコピーする
   for(int i=0; i<DATA_TOTAL; i++)
     {
      ResetLastError();
      if(!CharArrayToStruct(ExtDataRates[i], ExtCharArray, sizeof(MqlRates)*i))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print("CharArrayToStruct() failed. Error code: ", GetLastError());
       return;
      }
    //--- uchar 配列からMql Rates構造体へのデータのコピーが成功すると、
    //--- 構造体にコピーされたデータを操作ログに出力する
    MqlRatesPrint(ExtDataRates[i]);
   }

 /*
結果:
 MqlRates data:
  Time = 2024.02.21 09:00;
  Open = 1.08116;
  High = 1.08141;
  Low = 1.08062;
  Close = 1.08124;
  Tick volume = 5344;
  Spread = 1;
  Real volume = 0

 MqlRates data:
  Time = 2024.02.21 08:00;
  Open = 1.08149;
  High = 1.08153;
  Low = 1.08073;
  Close = 1.08114;
  Tick volume = 3607;
  Spread = 2;
  Real volume = 0

 MqlRates data:
  Time = 2024.02.21 07:00;
  Open = 1.08143;
  High = 1.08165;
  Low = 1.08122;
  Close = 1.08150;
  Tick volume = 2083;
  Spread = 0;
  Real volume = 0

 MqlRates data:
  Time = 2024.02.21 06:00;
  Open = 1.08178;
  High = 1.08190;
  Low = 1.08132;
  Close = 1.08142;
  Tick volume = 1733;
  Spread = 0;
```

© 2000-2025, MetaQuotes Ltd.

```
    Real volume = 0
   */
  }
//+------------------------------------------------------------------+
//| MqlRates構造体を介して指定されたインデックスによって                |
//| バーデータをuchar配列にコピーする                                  |
//+------------------------------------------------------------------+
bool CopyRatesToCharArray(const int index, uchar &data_array[])
  {
   MqlRates rates[1];
   uint     start=sizeof(MqlRates);

   ResetLastError();
   if(CopyRates(Symbol(), PERIOD_CURRENT, index, 1, rates)!=1)
     {
      Print("CopyRates() failed. Error code: ", GetLastError());
      return(false);
     }
   if(!StructToCharArray(rates[0], data_array, start*index))
     {
      Print("StructToCharArray() failed. Error code: ", GetLastError());
      return(false);
     }

   return(true);
  }
//+------------------------------------------------------------------+
//| MqlRates構造体のフィールドを操作ログに出力する                     |
//+------------------------------------------------------------------+
void MqlRatesPrint(MqlRates &rates)
  {
//--- 出力文字列を作成する
   string text=StringFormat(" Time = %s;\n"
                            " Open = %.*f;\n"
                            " High = %.*f;\n"
                            " Low = %.*f;\n"
                            " Close = %.*f;\n"
                            " Tick volume = %I64u;\n"
                            " Spread = %d;\n"
                            " Real volume = %I64u",
                            TimeToString(rates.time),
                            _Digits, rates.open,
                            _Digits, rates.high,
                            _Digits, rates.low,
                            _Digits, rates.close,
                            rates.tick_volume,
                            rates.spread,
                            rates.real_volume);
//--- ヘッダーと出力文字列を操作ログに出力する
```

© 2000-2025, MetaQuotes Ltd.

```
   Print("MqlRates data:\n", text, "\n");
  }
```

参照

StringToCharArray、ShortArrayToString、StructToCharArray、コードページの使用、FileReadStruct、ユニオン(union)、MathSwap

© 2000-2025, MetaQuotes Ltd.

StructToCharArray

POD構造体をuchar型配列にコピーします。

```
bool  StructToCharArray(
   const void&  struct_object,      // 構造体
   uchar&       char_array[],       // 配列
   uint         start_pos=0         // 配列の開始位置
   );
```

パラメータ

struct_object

[in] 任意の型のPOD構造体への参照(単純なデータ型のみを含む)

char_array[]

[in] uchar型の配列。

start_pos=0

[in] コピーされたデータを追加する配列内の位置。

戻り値

成功の場合はtrue、それ以外の場合はfalse

注意事項

コピーすると、十分なスペースがない場合、動的配列が自動的に拡張されます(ArrayResize)。配列を必要な値まで拡張できない場合、関数はエラーを返します。

例：

```
uchar    ExtCharArray[];
MqlRates ExtRates[1];

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 現在の1つの足のデータをMql Rates構造体にコピーする
   if(CopyRates(Symbol(), PERIOD_CURRENT, 0, 1, ExtRates)!=1)
     {
      Print("CopyRates() failed. Error code: ", GetLastError());
      return;
     }

//--- Mql Rates構造体で受信したデータを操作ログに出力する
   Print("Data in the structure immediately after receiving it:");
   MqlRatesPrint(ExtRates[0]);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 構造体をuchar型配列にコピーする
   ResetLastError();
   if(!StructToCharArray(ExtRates[0], ExtCharArray))
     {
      Print("StructToCharArray() failed. Error code: ", GetLastError());
      return;
     }

//--- 構造体をクリアする
   ZeroMemory(ExtRates[0]);
/--- クリア後に構造体のデータを出力する
   Print("\nData in the structure after ZeroMemory():");
   MqlRatesPrint(ExtRates[0]);

//--- 次にuchar配列からMql Rates構造体にデータをコピーする
   if(!CharArrayToStruct(ExtRates[0], ExtCharArray))
     {
      Print("CharArrayToStruct() failed. Error code: ", GetLastError());
      return;
     }
//--- uchar 配列からデータをコピーした後、Mql Rates構造体のデータを出力する
   Print("\nData in the MqlRates structure after copying it from a uchar array:");
   MqlRatesPrint(ExtRates[0]);

   /*
  結果:

   Data in the structure immediately after receiving it:
   MqlRates data:
    Time = 2024.02.21 07:00;
    Open = 1.08143;
    High = 1.08158;
    Low = 1.08122;
    Close = 1.08137;
    Tick volume = 1341;
    Spread = 4;
    Real volume = 0

   Data in the structure after ZeroMemory():
   MqlRates data:
    Time = 0;
    Open = 0.00000;
    High = 0.00000;
    Low = 0.00000;
    Close = 0.00000;
    Tick volume = 0;
    Spread = 0;
    Real volume = 0
```

© 2000-2025, MetaQuotes Ltd.

```
   Data in the MqlRates structure after copying it from a uchar array:
   MqlRates data:
    Time = 2024.02.21 07:00;
    Open = 1.08143;
    High = 1.08158;
    Low = 1.08122;
    Close = 1.08137;
    Tick volume = 1341;
    Spread = 4;
    Real volume = 0
   */
  }
//+------------------------------------------------------------------+
//| MqlRates構造体のフィールドを操作ログに出力する                   |
//+------------------------------------------------------------------+
void MqlRatesPrint(MqlRates &rates)
  {
//--- 出力文字列を作成する
   string text=StringFormat(" Time = %s;\n"
                            " Open = %.*f;\n"
                            " High = %.*f;\n"
                            " Low = %.*f;\n"
                            " Close = %.*f;\n"
                            " Tick volume = %I64u;\n"
                            " Spread = %d;\n"
                            " Real volume = %I64u",
                            TimeToString(rates.time),
                            _Digits, rates.open,
                            _Digits, rates.high,
                            _Digits, rates.low,
                            _Digits, rates.close,
                            rates.tick_volume,
                            rates.spread,
                            rates.real_volume);
//--- ヘッダーと出力文字列を操作ログに出力する
   Print("MqlRates data:\n", text);
  }
```

参照

StringToCharArray、ShortArrayToString、CharArrayToStruct、Use of a Codepage、FileWriteStruct、Unions (union)、MathSwap

© 2000–2025, MetaQuotes Ltd.

ColorToARGB

この関数は color 型を uint 型に変換し色の ARGB 表現を取得します。ARGB カラー形式は グラフィックリソースやテキスト表示に使用され、CCanvas 標準ライブラリクラスにも使用されます。

```
uint  ColorToARGB(
   color  clr,          // color フォーマットに変換された色
   uchar  alpha=255     // 色の透明度を管理するアルファチャンネル
   );
```

パラメータ

clr

[in] color 型変数のカラー値。

alpha

[in] ARGB 形式で色を受け取るためのアルファチャンネルの値。値は 0（前景ピクセル色は基礎ピクセル色の表示に影響しない）から 255（基本ピクセル色が完全に前景ピクセル色で置き換えられる）までです。百分率での色の透明度は (1-alpha/255)*100% のように計算されます。つまり、アルファチャンネルの小さい値は、より透明な色につながります。

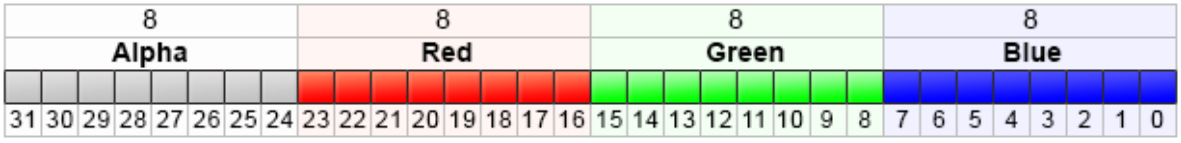
戻り値

アルファ、赤、緑、青（アルファチャンネル、赤、緑、青）値が 4 つの uint 型のバイト直列で設定されている ARGB 形式で表現された色

注意事項

RGBは、コンピュータグラフィックスの画面上のピクセルの色の記述の基本に一般的に使用される形式です。基本色の名称は、赤、緑、青の色成分を設定するために使用されています。各コンポーネントは 0～255（16 進形式の 0x00 から 0XFF ）の範囲の彩度を指定する 1 バイトで記述されています。白は全ての色を含むので 0xFFFFFF として記載されます。つまり、3 成分のそれぞれが最大値の 0xFF を持ちます。

ただし、一部のタスクでは、透明性のある色で覆われている場合の画像の外観を記述するために、色の透明度を指定する必要があります。アルファチャンネルの概念は、このような場合のために導入されています。アルファチャンネルは、RGB 形式の付加的なコンポーネントとして実装されています。ARGB 形式の構造体は下記の通りです。



ARGB値は、一般的に、それぞれ、アルファ、赤、緑、青のチャンネルの値を表す数字の各ペアに進形式を使用して表現されます。例えば 80FFFF00 は 50.2 パーセント不透明な黄色です。0x80 は初めに 0xFF の 50.2% である 50.2% アルファ値を設定します。そして、1 番目の FF ペアは赤成分の最大値を定義します。次の FF ペアは緑成分の最大値を定義します。最後の 00 ペアは、青色成分の最小値（青の不在）を表します。緑と赤の組み合わせは、黄色です。アルファチャンネルを使用しない場合、エントリは RRGGBB の6桁に低減することが出来ます。これがアルファチャンネル値を uint 整数型で先頭ビットに格納する理由です。

© 2000-2025, MetaQuotes Ltd.

16 進数は状況に応じて「 0x 」または「 # 」プレフィックスを使用して書かれます（例えば 80FFFF00、0x80FFFF00 または #80FFFF00 ）。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 透明度を設定する
   uchar alpha=0x55;  // 0x55 は 55/255=21.6 % 透明度を意味する
   //--- clrBlue 色のために、ARGBへの変換を導き出す
   PrintFormat("0x%.8X - clrBlue",clrBlue);
   PrintFormat("0x%.8X - clrBlue ARGB with alpha=0x55 (transparency 21.6%
%)",ColorToARGB(clrBlue,alpha));
   //--- clrGreen 色のために、ARGB への変換を導き出す
   PrintFormat("0x%.8X - clrGreen",clrGreen);
   PrintFormat("0x%.8X - clrGreen ARGB with alpha=0x55 (transparency 21.6%
%)",ColorToARGB(clrGreen,alpha));
   //--- clrRed 色のために、ARGB への変換を導き出す
   PrintFormat("0x%.8X - clrRed",clrRed);
   PrintFormat("0x%.8X - clrRed ARGB with alpha=0x55 (transparency 21.6%
%)",ColorToARGB(clrRed,alpha));
  }
```

参照

リソース、ResourceCreate()、TextOut()、color 型、char、short、int 及び long 型

© 2000-2025, MetaQuotes Ltd.

ColorToString

color 値を「R, G, B」の文字列に変換します。

```
string  ColorToString(
   color   color_value,      // color 値
   bool    color_name        // 色の名称を表示するかどうか
   );
```

パラメータ

color_value

[in] color 型変数のカラー値。

color_name

[in] 色の名称が事前に定義された color 定数 と同じ場合は色の名称を返します。

戻り値

「R, G, B」としての色の文字列表現。R、G 及び B は文字列に変換された0 から 255 の 10 進数定数です。color_name=true パラメータが設定されている場合、カラー値から色の名称への変換が試みられます。

例:

```
   string clr=ColorToString(C'0,255,0'); // 緑
   Print(clr);

   clr=ColorToString(C'0,255,0',true);   // カラー定数を取得
   Print(clr);
```

参照

StringToColor, ColorToARGB

© 2000–2025, MetaQuotes Ltd.

DoubleToString

数値をテキスト文字列に変換します。

```
string  DoubleToString(
   double  value,      // 数
   int     digits=8    // 小数点以下の桁数
   );
```

パラメータ

value

[in] 浮動小数点を持つ値

digits

[in] 精度フォーマット。*digits* 値が 0 から 16 の間の場合、指定された小数点以下の桁数で数値の文字列表現が取得されます。*digits* 値が -1 から -16 の間の場合、指定された小数点以下の桁数で数値の科学表記での文字列が表現取得されます。それ以外の場合では全て、文字列値には小数点以下8桁が含まれます。

戻り値

指定された精度の数のシンボル表現を含む文字列

例:

```
   Print("DoubleToString(120.0 + M_PI) : ",DoubleToString(120.0+M_PI));
   Print("DoubleToString(120.0 + M_PI,16) : ",DoubleToString(120.0+M_PI,16));
   Print("DoubleToString(120.0 + M_PI,-16) : ",DoubleToString(120.0+M_PI,-16));
   Print("DoubleToString(120.0 + M_PI,-1) : ",DoubleToString(120.0+M_PI,-1));
   Print("DoubleToString(120.0 + M_PI,-20) : ",DoubleToString(120.0+M_PI,-20));
```

参照

NormalizeDouble、StringToDouble

© 2000-2025, MetaQuotes Ltd.

EnumToString

任意型の列挙値を文字列に変換します。

```
string  EnumToString(
   any_enum  value      // 任意型の列挙値
   );
```

パラメータ

value

[in] 任意型の列挙値

戻り値

列挙のテキスト表現を含む文字列。エラーメッセージを取得するには GetLastError() 関数が呼ばれます。

注意事項

この関数は _LastError 変数に下記のエラー値を設定することがあります。

- ERR_INTERNAL_ERROR – 実行環境のエラー
- ERR_NOT_ENOUGH_MEMORY – メモリが操作完了に不充分
- ERR_INVALID_PARAMETER – 列挙値の名称が使用不可能

例:

```
enum interval  // 名前付き定数の列挙
  {
   month=1,      // 1 ヶ月の間隔
   two_months,   // 2 ヶ月
   quarter,      // 3 ヶ月(四半期)
   halfyear=6,   // 半年
   year=12,      // 1 年( 12ヶ月)
  };
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 時間間隔を 1 ヶ月に設定する
   interval period=month;
   Print(EnumToString(period)+"="+IntegerToString(period));

//--- 時間間隔を四半期(3 ヶ月)に設定する
   period=quarter;
   Print(EnumToString(period)+"="+IntegerToString(period));

//--- 時間間隔を 1 年(12ヶ月)に設定する
   period=year;
```

© 2000–2025, MetaQuotes Ltd.

```
   Print(EnumToString(period)+"="+IntegerToString(period));

//--- 注文の種類の表示をチェックする
   ENUM_ORDER_TYPE type=ORDER_TYPE_BUY;
   Print(EnumToString(type)+"="+IntegerToString(type));

//--- 不正な値の表示をチェックする
   type=WRONG_VALUE;
   Print(EnumToString(type)+"="+IntegerToString(type));

// 結果:
// month=1
// quarter=3
// year=12
// ORDER_TYPE_BUY=0
// ENUM_ORDER_TYPE::-1=-1
  }
```

参照

列挙、入力変数

© 2000-2025, MetaQuotes Ltd.

IntegerToString

この関数は、整数型の値を指定された長さの文字列に変換し、取得された文字列を返します。

```
string  IntegerToString(
   long     number,              // 数
   int      str_len=0,           // 結果の文字列の長さ
   ushort   fill_symbol=' '      // フィラー
   );
```

パラメータ

number

[in] 変換される数

str_len=0

[in] 文字列の長さ。結果の文字列が指定より長い場合、文字列は短くされません。結果の文字列が指定より短い場合には、フィラーシンボルが左側に追加されます。

fill_symbol=' '

[in] フィラーシンボル。デフォルトではスペース。

戻り値

文字列

例：

```
#define DATA_TOTAL 1001

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- DATA_TOTALによるループ内でインデックス100の倍数を持つ行を表示する
//--- 文字列は4文字の形式でインデックス値を表示する
//--- 文字列の長さが4文字未満の場合、文字列内のループインデックスの値の前に
//--- 先行ゼロが付けられる
   for(int i=0; i<DATA_TOTAL; i++)
     {
      if(i%100==0)
         Print("Converted index value: ",IntegerToString(i,4,'0'));
     }
   /*
  結果:
   Converted index value: 0000
   Converted index value: 0100
   Converted index value: 0200
   Converted index value: 0300
```

© 2000-2025, MetaQuotes Ltd.

```
   Converted index value: 0400
   Converted index value: 0500
   Converted index value: 0600
   Converted index value: 0700
   Converted index value: 0800
   Converted index value: 0900
   Converted index value: 1000
   */
  }
```

参照

StringToInteger

© 2000–2025, MetaQuotes Ltd.

ShortToString

Unicode シンボルコードを 1 シンボルの文字列に変換して返します。

```
string  ShortToString(
   ushort  symbol_code      // シンボル
   );
```

パラメータ

symbol_code

[in] シンボルコード。シンボルコードの代わりに、シンボルまたは Unicode の表にある 2 バイトの16 進コードの文字列を含むリテラル文字列を使用することが出来ます。

戻り値

文字列

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Unicode番号U+23E9の文字から始まる18文字をループで出力する
   for(int i=0; i<18; i++)
     {
      ushort code=0x23E9+(ushort)i;
      PrintFormat("Unicode number U+%hX: %s",code,ShortToString(code));
     }
   /*
  結果:
   Unicode number U+23E9: ⏩
   Unicode number U+23EA: ⏪
   Unicode number U+23EB: ⏫
   Unicode number U+23EC: ⏬
   Unicode number U+23ED: ⏭
   Unicode number U+23EE: ⏮
   Unicode number U+23EF: ⏯
   Unicode number U+23F0: ⏰
   Unicode number U+23F1: ⏱
   Unicode number U+23F2: ⏲
   Unicode number U+23F3: ⏳
   Unicode number U+23F4: □
   Unicode number U+23F5: □
   Unicode number U+23F6: □
   Unicode number U+23F7: □
   Unicode number U+23F8: ⏸
```

© 2000-2025, MetaQuotes Ltd.

```
   Unicode number U+23F9: ▣
   Unicode number U+23FA: ⦿
   */
  }
```

参照

StringToCharArray、CharToString、StringGetCharacter

© 2000–2025, MetaQuotes Ltd.

ShortArrayToString

配列の一部を戻り値の文字列に複製します。

```
string  ShortArrayToString(
   ushort  array[],      // 配列
   int     start=0,      // 配列でのコピー開始位置
   int     count=-1      // シンボル
   );
```

パラメータ

array[]

[in] ushort 型の配列（wchar_t type 型のアナログ）

start=0

[in] コピー開始位置。デフォルトは 0。

count=-1

[in] 複製する配列要素数。結果の文字列の長さを定義します。初期値は、配列の最後または終了する 0まで複製することを意味する -1 です。

戻り値

文字列

例：

```
#define ROW_SIZE 16

ushort ExtShortArray[]
={0x2190
,0x2191
,0x2192
,0x2193
,0x2194
,0x2195,0x2196,0x2197,0x2198,0x2199,0x219A,0x219B,0x219C,0x219D,0x219E,0x219F,
                          0x21A0
,0x21A1
,0x21A2
,0x21A3
,0x21A4
,0x21A5,0x21A6,0x21A7,0x21A8,0x21A9,0x21AA,0x21AB,0x21AC,0x21AD,0x21AE,0x21AF,
                          0x21B0
,0x21B1
,0x21B2
,0x21B3
,0x21B4
,0x21B5,0x21B6,0x21B7,0x21B8,0x21B9,0x21BA,0x21BB,0x21BC,0x21BD,0x21BE,0x21BF,
```

© 2000–2025, MetaQuotes Ltd.

```
                           0x21C0
,0x21C1
,0x21C2
,0x21C3
,0x21C4
,0x21C5,0x21C6,0x21C7,0x21C8,0x21C9,0x21CA,0x21CB,0x21CC,0x21CD,0x21CE,0x21CF,
                           0x21D0
,0x21D1
,0x21D2
,0x21D3
,0x21D4
,0x21D5,0x21D6,0x21D7,0x21D8,0x21D9,0x21DA,0x21DB,0x21DC,0x21DD,0x21DE,0x21DF,
                           0x21E0
,0x21E1
,0x21E2
,0x21E3
,0x21E4
,0x21E5,0x21E6,0x21E7,0x21E8,0x21E9,0x21EA,0x21EB,0x21EC,0x21ED,0x21EE,0x21EF,
                           0x21F0
,0x21F1
,0x21F2
,0x21F3
,0x21F4
,0x21F5,0x21F6,0x21F7,0x21F8,0x21F9,0x21FA,0x21FB,0x21FC,0x21FD,0x21FE,0x21FF};

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 7x16のシンボルテーブル内の文字列の数を計算する
   int total=(int)ExtShortArray.Size()/ROW_SIZE;
//--- テーブルの行ごとのループで、Unicode文字で各行に16個の7セットの矢印を行ごとに仕訳帳に表示する
   for(int i=0; i<total; i++)
     {
      int row=i*ROW_SIZE;
      PrintFormat("Arrow set %u: %
s",i+1,ShortArrayToString(ExtShortArray,row,ROW_SIZE));
     }
   /*
  結果:
   Arrow set 1: ←↑→↓↔↕□□□□□□□□□□□□
   Arrow set 2: □□□□□□□□↨□□□□□□□□
   Arrow set 3: □□□□□□□□□□□□□□□□
   Arrow set 4: □□□□□□□□□□□□□□□□
   Arrow set 5: □□⇒□⇔□□□□□□□□□□□□
   Arrow set 6: □□□□□□□□□□□□□□□□
   Arrow set 7: □□□□□□□□□□□□□□□□
   */
  }
```

参照

StringToShortArray、CharArrayToString、コードページの利用

© 2000-2025, MetaQuotes Ltd.

TimeToString

1970年1月1日からの経過秒を含む値を「yyyy.mm.dd hh:mi」フォーマットの文字列に変換します。

```
string  TimeToString(
   datetime  value,                              // 数
   int       mode=TIME_DATE|TIME_MINUTES         // 出力フォーマット
   );
```

パラメータ

value

[in] 01.01.1970 から経った秒数

mode=TIME_DATE|TIME_MINUTES

[[in] 任意のデータ入力モード。フラグまたはフラグの組み合わせです。
TIME_DATE の結果は "yyyy.mm.dd"、
TIME_MINUTES の結果は "hh:mi"、
TIME_SECONDS の結果は "hh:mi:ss" です。

戻り値

文字列

例：

```
datetime ExtBarTimeOpen;

//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- タイマーを1秒に設定する
   EventSetTimer(1);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//|  カスタム指標を初期化解除する関数                                |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   Comment("");
  }
//+------------------------------------------------------------------+
//|  カスタム指標の反復関数                                          |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
```

© 2000-2025, MetaQuotes Ltd.

```
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- 現在のバーが開いた時間を取得する
   ExtBarTimeOpen=time[rates_total-1];
//--- 次の呼び出しのためにprev_calculatedの値を返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Timer関数                                                        |
//+------------------------------------------------------------------+
void OnTimer()
  {
//--- 前のバーが開いた時間を取得する
   static datetime bar_open_time=ExtBarTimeOpen;
//--- バーが開いてから経過した秒数をカウントする
   static int seconds=int(TimeCurrent()-ExtBarTimeOpen);
//--- 前回の開始時間が現在の開始時間と異なる場合、これは新しいバーである
//--- 新しい開始時刻を前の時刻と同じように書き込み、秒をゼロに設定する
   if(bar_open_time!=ExtBarTimeOpen)
     {
      bar_open_time=ExtBarTimeOpen;
      seconds=0;
     }
//--- バーが開いてから経過した秒数を増やして調整する
   seconds++;
   if(seconds>PeriodSeconds(PERIOD_CURRENT))
      seconds=0;
//--- yyyy.mm.dd hh:miとしてのバーの開いた時刻
   string bar_time_open=TimeToString(ExtBarTimeOpen);
//--- yyyy.mm.dd hh:miとしての現在時刻
   string time_current=TimeToString(TimeCurrent(),TIME_DATE|TIME_MINUTES|
TIME_SECONDS);
//--- 新しいバーが開くまでの残り秒数
   int    sec_left=PeriodSeconds(PERIOD_CURRENT)-seconds;
/--- 現在のバーが閉じるまでの残り時間(hh:mm:ss)
   string time_left=TimeToString(sec_left,TIME_MINUTES|TIME_SECONDS);
//--- 出力文字列を作成する
   string txt=StringFormat("Opening time of the current bar: %s\n"+
                           "Time Current: %s\n"+
                           "Seconds have passed since the bar opened: %d\n"+
                           "Approximately seconds left before bar closes: %d\n"+
```

© 2000-2025, MetaQuotes Ltd.

```
                              "Time remaining until bar closes: %
s",bar_time_open,time_current,seconds,sec_left,time_left);
//--- バーの開いた時刻と現在時刻、
//--- 現在のバーが開いてから閉じるまでの経過秒数、
//--- コメント内の現在のバーが閉じるまでの残り時間を表示する
   Comment(txt);
   /*
   result on M1:
   Opening time of the current bar: 2024.02.22 18:06
   Time Current: 2024.02.22 18:06:24
   Seconds have passed since the bar opened: 25
   Approximately seconds left before bar closes: 35
   Time remaining until bar closes: 00:00:35

   result on M5:
   Opening time of the current bar: 2024.02.22 18:05
   Time Current: 2024.02.22 18:07:28
   Seconds have passed since the bar opened: 149
   Approximately seconds left before bar closes: 151
   Time remaining until bar closes: 00:02:31

   result on H1:
   Opening time of the current bar: 2024.02.22 18:00
   Time Current: 2024.02.22 18:08:13
   Seconds have passed since the bar opened: 494
   Approximately seconds left before bar closes: 3106
   Time remaining until bar closes: 00:51:46

   result on D1:
   Opening time of the current bar: 2024.02.22 00:00
   Time Current: 2024.02.22 18:11:01
   Seconds have passed since the bar opened: 65462
   Approximately seconds left before bar closes: 20938
   Time remaining until bar closes: 05:48:58
   */
  }
```

参照

StringToTime、TimeToStruct

© 2000-2025, MetaQuotes Ltd.

NormalizeDouble

浮動小数点数を指定された精度に丸めます。

```
double  NormalizeDouble(
   double   value,      // 正規化数
   int      digits      // 小数点以下の桁数
   );
```

パラメータ

value

[in] 浮動小数点を持つ値

digits

[in] 精度フォーマット。0~8 の小数点以下の桁数。

戻り値

事前に定義された精度を持つdouble 型の値

注意事項

計算された決済逆指値及び決済指値、また未決注文の価格は Digits() で取得される値によって正規化する必要があります。

正規化数は小数点以下の桁数を多く含むことがあるので、Print() 関数を使用しての操作ログへの出力時にはご注意ください。例えば

```
   double a=76.671;              // 小数点以下3桁の正規化数
   Print("Print(76.671)=",a);    // そのまま出力
   Print("DoubleToString(a,8)=",DoubleToString(a,8)); // 事前定義された精度で出力
```

端末には下記が表示されます。

```
   DoubleToString(a,8)=76.67100000

   Print(76.671)=76.67100000000001
```

例:

```
   double pi=M_PI;
   Print("pi = ",DoubleToString(pi,16));

   double pi_3=NormalizeDouble(M_PI,3);
   Print("NormalizeDouble(pi,3) = ",DoubleToString(pi_3,16))
   ;
   double pi_8=NormalizeDouble(M_PI,8);
   Print("NormalizeDouble(pi,8) = ",DoubleToString(pi_8,16));

   double pi_0=NormalizeDouble(M_PI,0);
   Print("NormalizeDouble(pi,0) = ",DoubleToString(pi_0,16));
```

© 2000-2025, MetaQuotes Ltd.

```
/*
   結果:
    pi= 3.1415926535897931
    NormalizeDouble(pi,3)= 3.1419999999999999
    NormalizeDouble(pi,8)= 3.1415926499999998
    NormalizeDouble(pi,0)= 3.0000000000000000
*/
```

参照

DoubleToString、Real 型（double、float）、型の削減

© 2000–2025, MetaQuotes Ltd.

StringToCharArray

Unicode から ANSI に変換された文字列をシンボルごとに uchar 型の配列の指定された一部に複製します。複製された要素の数が返されます。

```
int  StringToCharArray(
   string  text_string,          // ソースの文字列
   uchar&  array[],              // 配列
   int     start=0,              // 配列でのコピー開始位置
   int     count=-1              // シンボル数
   uint    codepage=CP_ACP       // コードページ
   );
```

パラメータ

text_string

[in] 複製する文字列

array[]

[out] uchar 型の配列

start=0

[in] コピー開始位置。デフォルトでは 0。

count=-1

[in] 複製する配列要素数。結果の文字列の長さを定義します。初期値は、配列の最後または終了する 0まで複製することを意味する -1 です。文字列を終了する 0 は受け取り側の配列に複製され、動的配列のサイズは文字列のサイズに応じて増加出来ます。動的配列のサイズはが文字列の長さを超える場合は、配列のサイズは縮小されません。

codepage=CP_ACP

[in] コードページの値。よく使用される コードページ では定数が提供されています。

戻り値

複製された要素の数

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 変換する文字列
   string text = "This is a test of the StringToCharArray() function";

//--- 設定されたコードページに従って入力文字列をuchar配列に変換する
   uchar char_array[];
   int   copied=StringToCharArray(text, char_array);
```

© 2000-2025, MetaQuotes Ltd.

```
   PrintFormat("String length: %u\nNumber of characters copied (with terminal 0): %
d\nArray of characters for the string '%s':",
               StringLen(text), copied, text);
//--- 結果の配列を操作ログに出力する
   ArrayPrint(char_array, 0, " | ");
   /*
  結果:
   String length: 50
   Number of characters copied (with terminal 0): 51
   Array of characters for the string 'This is a test of the StringToCharArray()
function':
   [ 0]  84 | 104 | 105 | 115 |  32 | 105 | 115 |  32 |  97 |  32 | 116 | 101 | 115
| 116 |  32 | 111 | 102
   [17]  32 | 116 | 104 | 101 |  32 |  83 | 116 | 114 | 105 | 110 | 103 |  84 | 111
|  67 | 104 |  97 | 114
   [34]  65 | 114 | 114 |  97 | 121 |  40 |  41 |  32 | 102 | 117 | 110 |  99 | 116
| 105 | 111 | 110 |   0
   */
  }
```

参照

CharArrayToString、StringToShortArray、コードページの利用

© 2000–2025, MetaQuotes Ltd.

StringToColor

「R, G, B」の文字列またはカラー名を含む文字列を color 型の値に変換します。

```
color  StringToColor(
   string  color_string      // 色の文字列表現
   );
```

パラメータ

color_string

[in] 「R, G, B」型の色、または、事前定義されたウェブカラーの名称の文字列表現

戻り値

色の値

例:

```
   color str_color=StringToColor("0,127,0");
   Print(str_color);
   Print((string)str_color);
//--- 色を少し変更する
   str_color=StringToColor("0,128,0");
   Print(str_color);
   Print((string)str_color);
```

参照

ColorToString, ColorToARGB

© 2000–2025, MetaQuotes Ltd.

StringToDouble

この関数は数のシンボル表現を含む文字列を double 型の数値に変換します。

```
double  StringToDouble(
   string  value      // 文字列
   );
```

パラメータ

value

[in] 数のシンボル表現を含む文字列

戻り値

double 型の値

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 変換する文字列
   string str = "12345.54321";
//--- 入力文字列を実数としてdoubl e型の変数に変換する
   double converted=StringToDouble(str);
//--- 結果の数値を小数点以下8桁まで正確に操作ログに表示する
   PrintFormat("The string '%s' is converted to the real number %.8f", str,
converted);
   /*
  結果:
   The string '12345.54321' is converted to the real number 12345.54321000
   */
  }
```

参照

NormalizeDouble、Real 型（double、float）、型の削減

© 2000-2025, MetaQuotes Ltd.

StringToInteger

この関数は数のシンボル表現を含む文字列をlong(整数)型の数値に変換します。

```
long  StringToInteger(
   string  value      // 文字列
   );
```

パラメータ

value

[in] 数を含む文字列

戻り値

long 型の値

例:

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 変換する文字列
   string str = "12345.54321";
//--- 入力文字列を数字として ong型の変数に変換する
   long converted=StringToInteger(str);
//--- 取得した数値を小数部分を切り捨てて操作ログに表示する
   PrintFormat("The string '%s' is converted to the integer number %I64d", str,
converted);
   /*
  結果:
   The string '12345.54321' is converted to the integer number 12345
   */
  }
```

S

参照

IntegerToString、Real 型(double、float)、型の削減

© 2000-2025, MetaQuotes Ltd.

StringToShortArray

この関数は文字列をシンボルごとに ushort 型の配列の指定された部分にコピーし、複製された要素の数を返します。

```
int  StringToShortArray(
   string  text_string,      // ソース文字列
   ushort& array[],          // 配列
   int     start=0,          // 配列でのコピー開始位置
   int     count=-1          // シンボル数
   );
```

パラメータ

text_string

[in] 複製する文字列

array[]

[out] ushort 型の配列（wchar_t type 型のアナログ）

start=0

[in] コピー開始位置。デフォルトでは 0。

count=-1

[in] 複製する配列要素数。結果の文字列の長さを定義します。初期値は、配列の最後または終了する 0まで複製することを意味する -1です。最後の 0 も受け取り側の配列に複製され、動的配列のサイズは文字列のサイズに応じて増加出来ます。動的配列のサイズはが文字列の長さを超える場合は、配列のサイズは縮小されません。

戻り値

複製された要素の数

例：

```
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                       |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 変換する文字を含む文字列
   string text = "Chars: ♡♫☎✂✢☏ ✿✉☆ ⌓♕ ♠®✧ ❄♣    ";
   ushort short_array[];
//--- 入力文字列内の「: 」文字の位置を検索し、次の文字から始まる部分文字列を抽出する
   int    pos=StringFind(text, ":");
   string str=(pos<0 ? text : StringSubstr(text, pos+1));
//--- 結果の文字列の左右からスペース、キャリッジ移動、タブ文字を削除する
   StringTrimLeft(str);
   StringTrimRight(str);
//--- 結果の文字列をushort配列にコピーし、結果の配列を操作ログに出力する
```

© 2000-2025, MetaQuotes Ltd.

```
  int copied=StringToShortArray(str, short_array);
  PrintFormat("String of characters length: %u\n"
              "Number of characters copied (with terminal 0): %d\n"
              "Array of chars for the string '%s':",
              StringLen(str), copied, str);
  ArrayPrint(short_array, 0, " | ");
  /*
 結果:
  String of characters length: 16
  Number of characters copied (with terminal 0): 17
  Array of chars for the string '♡♫☎✈✢☏❂✉☆⌓♛♠®✧❄♣':
  10084 |  9835 |  9742 |  9992 | 10019 |  9743 |  9728 |  9993 |  9734 |  9729 |
9813 |  9824 |   174 | 10023 | 10054 |  9827 |     0
  */
 }
```

参照

ShortArrayToString、StringToCharArray、コードページの利用

© 2000-2025, MetaQuotes Ltd.

StringToTime

"yyyy.mm.dd [hh:mi]"形式の時刻および/または日付を含む文字列を日時型の数値に変換します。

```
datetime  StringToTime(
   const string  time_string      // 日付の文字列
   );
```

パラメータ

time_string

[in] 次のいずれかで指定された形式での文字列:

- "yyyy.mm.dd [hh:mi]"
- "yyyy.mm.dd [hh:mi:ss]"
- "yyyymmdd [hh:mi:ss]"
- "yyyymmdd [hhmiss]"
- "yyyy/mm/dd [hh:mi:ss]"
- "yyyy-mm-dd [hh:mi:ss]"

戻り値

1970年1月1日からの経過秒数の合計を含むdatetime型の値

注意事項

StringToTime()を呼び出す前の*time_string*による追加処理を避けるために、日付と時刻の間にある一連のスペースと集計文字は単一スペースと見なされます。

例:

```
//--- 入力パラメータ
input group    "The date can be entered in any of the formats:"
input group    "yyyy.mm.dd [hh:mi], yyyy.mm.dd [hh:mi:ss]"
input group    "yyyymmdd [hh:mi:ss], yyyymmdd [hhmiss]"
input group    "yyyy/mm/dd [hh:mi:ss], yyyy-mm-dd [hh:mi:ss]"
input string   InpDateStr;    // Please enter the date here as a string

//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力に文字列として入力された時刻を日時値に変換する
   datetime time=StringToTime(InpDateStr);
//--- 入力された文字列と取得した時刻を操作ログに表示する
   PrintFormat("Date entered as a string in the form '%s' is converted to datetime
in the form '%s'",
```

© 2000-2025, MetaQuotes Ltd.

```
                InpDateStr, TimeToString(time, TIME_DATE|TIME_MINUTES|TIME_SECONDS));
//--- 受信した日時上に垂直線を作成し、チャートをこの位置に移動する
   if(CreateVLine(time))
      ChartNavigateToTime(time);
   /*
  結果:
   Date entered as a string in the form '' is converted to datetime in the form
'1970.01.01 00:00:00'
   Date entered as a string in the form '2024' is converted to datetime in the form
'2024.02.24 20:24:00'
   Date entered as a string in the form '202400' is converted to datetime in the
form '2024.02.24 20:24:00'
   Date entered as a string in the form '20240000' is converted to datetime in the
form '2024.02.24 00:00:00'
   Date entered as a string in the form '2024022410' is converted to datetime in the
 form '2030.09.06 00:00:00'
   Date entered as a string in the form '20240224 10' is converted to datetime in
the form '2024.02.24 10:00:00'
   Date entered as a string in the form '20240224 01' is converted to datetime in
the form '2024.02.24 01:00:00'
   Date entered as a string in the form '20240224 0030' is converted to datetime in
the form '2024.02.24 23:00:00'
   Date entered as a string in the form '20240224 0100' is converted to datetime in
the form '2024.02.24 01:00:00'
   */
  }
//+------------------------------------------------------------------+
//|  垂直線オブジェクトを作成する
//+------------------------------------------------------------------+
bool CreateVLine(const datetime line_time)
  {
   ResetLastError();

   string name=MQLInfoString(MQL_PROGRAM_NAME)+"_VLINE";
   if(!ObjectCreate(0, name, OBJ_VLINE, 0, line_time, 0))
     {
      Print("ObjectCreate() failed. Error code: ", GetLastError());
      return(false);
     }
   ObjectSetInteger(0, name, OBJPROP_STYLE, STYLE_DOT);
   ObjectSetInteger(0, name, OBJPROP_SELECTABLE, true);

   return(true);
  }
//+------------------------------------------------------------------+
//|  チャートを指定されたバーの開始時間にシフトする
//+------------------------------------------------------------------+
bool ChartNavigateToTime(const datetime time)
  {
   ChartSetInteger(0, CHART_AUTOSCROLL, false);
```

© 2000–2025, MetaQuotes Ltd.

```
   ResetLastError();

   int bar=iBarShift(_Symbol, PERIOD_CURRENT, time);
   if(bar<0)
     {
      PrintFormat("%s: iBarShift() failed. Error code: %d", __FUNCTION__,
GetLastError());
      return(false);
     }

   long first=0;
   if(!ChartGetInteger(0, CHART_FIRST_VISIBLE_BAR, 0, first))
     {
      PrintFormat("%s: ChartGetInteger() failed. Error code: %d", __FUNCTION__,
GetLastError());
      return(false);
     }

   return(ChartNavigate(0, CHART_CURRENT_POS, (int)first-bar));
  }
```

参照

TimeToString、TimeToStruct

© 2000–2025, MetaQuotes Ltd.

StringFormat

この関数は、取得されたパラメータをフォーマットして文字列として返します。

```
string  StringFormat(
   string  format,      // フォーマット記述した文字列
   ...     ...          // パラメータ
   );
```

パラメータ

フォーマット でありません。

[in] フォーマット方法を含む文字列。フォーマットの規則は PrintFormat 関数の場合と同じです。

. . .

[in] コンマで区切られたパラメータ

戻り値

文字列

例:

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//-- 文字列変数
   string output_string;
   string temp_string;
   string format_string;
//--- ヘッダを準備
   temp_string=StringFormat("Contract specification for %s:\n",_Symbol);
   StringAdd(output_string,temp_string);
//--- int 値の出力
   int digits=(int)SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);
   temp_string=StringFormat("   SYMBOL_DIGITS = %d (number of digits after the
decimal point)\n",
                            digits);
   StringAdd(output_string,temp_string);
//--- double 値の出力(小数点以下の桁数は変動)
   double point_value=SymbolInfoDouble(_Symbol,SYMBOL_POINT);
   format_string=StringFormat("   SYMBOL_POINT = %%.%df (point value)\n",
                              digits);
   temp_string=StringFormat(format_string,point_value);
   StringAdd(output_string,temp_string);
//--- int value output
   int spread=(int)SymbolInfoInteger(_Symbol,SYMBOL_SPREAD);
   temp_string=StringFormat("   SYMBOL_SPREAD = %d (current spread in points)\n",
                            spread);
   StringAdd(output_string,temp_string);
//--- int 値の出力
   int min_stop=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_STOPS_LEVEL);
   temp_string=StringFormat("   SYMBOL_TRADE_STOPS_LEVEL = %d (minimal indention in
points for Stop orders)\n",
                            min_stop);
   StringAdd(output_string,temp_string);
//--- double (分数部なし)
   double contract_size=SymbolInfoDouble(_Symbol,SYMBOL_TRADE_CONTRACT_SIZE);
   temp_string=StringFormat("   SYMBOL_TRADE_CONTRACT_SIZE = %.f (contract size)\n",
                            contract_size);
   StringAdd(output_string,temp_string);
//--- デフォルト精度での double 値の出力
   double tick_size=SymbolInfoDouble(_Symbol,SYMBOL_TRADE_TICK_SIZE);
   temp_string=StringFormat("   SYMBOL_TRADE_TICK_SIZE = %f (minimal price change)
\n",
                            tick_size);
   StringAdd(output_string,temp_string);
//--- スワップ計算モードの決定
   int swap_mode=(int)SymbolInfoInteger(_Symbol,SYMBOL_SWAP_MODE);
   string str_swap_mode;
   switch(swap_mode)
     {
      case SYMBOL_SWAP_MODE_DISABLED: str_swap_mode="SYMBOL_SWAP_MODE_DISABLED (no
swaps)"; break;
      case SYMBOL_SWAP_MODE_POINTS: str_swap_mode="SYMBOL_SWAP_MODE_POINTS (in
points)"; break;
      case SYMBOL_SWAP_MODE_CURRENCY_SYMBOL:
str_swap_mode="SYMBOL_SWAP_MODE_CURRENCY_SYMBOL (in money, in base currency of the
symbol)"; break;
      case SYMBOL_SWAP_MODE_CURRENCY_MARGIN:
str_swap_mode="SYMBOL_SWAP_MODE_CURRENCY_MARGIN (in money, in margin currency of the
symbol)"; break;
      case SYMBOL_SWAP_MODE_CURRENCY_DEPOSIT:
str_swap_mode="SYMBOL_SWAP_MODE_CURRENCY_DEPOSIT (in money, in client deposit
currency)"; break;
```

© 2000-2025, MetaQuotes Ltd.

```
      case SYMBOL_SWAP_MODE_INTEREST_CURRENT:
str_swap_mode="SYMBOL_SWAP_MODE_INTEREST_CURRENT (as the specified annual interest
from the instrument price at calculation of swap)"; break;
      case SYMBOL_SWAP_MODE_INTEREST_OPEN:
str_swap_mode="SYMBOL_SWAP_MODE_INTEREST_OPEN (as the specified annual interest from
the position open price)"; break;
      case SYMBOL_SWAP_MODE_REOPEN_CURRENT:
str_swap_mode="SYMBOL_SWAP_MODE_REOPEN_CURRENT (by reopening positions by the close
price +/- specified number of points)"; break;
      case SYMBOL_SWAP_MODE_REOPEN_BID: str_swap_mode="SYMBOL_SWAP_MODE_REOPEN_BID
(by reopening positions by the current Bid price +/- specified number of points)";
break;
     }
//--- 文字列値の出力
   temp_string=StringFormat("   SYMBOL_SWAP_MODE = %s\n",
                            str_swap_mode);
   StringAdd(output_string,temp_string);
//--- デフォルト精度でのdouble 値の出力
   double swap_long=SymbolInfoDouble(_Symbol,SYMBOL_SWAP_LONG);
   temp_string=StringFormat("   SYMBOL_SWAP_LONG = %f (long swap value)\n",
                            swap_long);
   StringAdd(output_string,temp_string);
//--- デフォルト精度でのdouble 値の出力
   double swap_short=SymbolInfoDouble(_Symbol,SYMBOL_SWAP_SHORT);
   temp_string=StringFormat("   SYMBOL_SWAP_SHORT = %f (short swap value)\n",
                            swap_short);
   StringAdd(output_string,temp_string);
//--- 取引モードの決定
   int trade_mode=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_MODE);
   string str_trade_mode;
   switch(trade_mode)
     {
      case SYMBOL_TRADE_MODE_DISABLED: str_trade_mode="SYMBOL_TRADE_MODE_DISABLED
(trade is disabled for the symbol)"; break;
      case SYMBOL_TRADE_MODE_LONGONLY: str_trade_mode="SYMBOL_TRADE_MODE_LONGONLY
(only long positions are allowed)"; break;
      case SYMBOL_TRADE_MODE_SHORTONLY: str_trade_mode="SYMBOL_TRADE_MODE_SHORTONLY
(only short positions are allowed)"; break;
      case SYMBOL_TRADE_MODE_CLOSEONLY: str_trade_mode="SYMBOL_TRADE_MODE_CLOSEONLY
(only position close operations are allowed)"; break;
      case SYMBOL_TRADE_MODE_FULL: str_trade_mode="SYMBOL_TRADE_MODE_FULL (no trade
restrictions)"; break;
     }
//--- 文字列値の出力
   temp_string=StringFormat("   SYMBOL_TRADE_MODE = %s\n",
                            str_trade_mode);
   StringAdd(output_string,temp_string);
//--- コンパクトな形式での double 値の出力
   double volume_min=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
   temp_string=StringFormat("   SYMBOL_VOLUME_MIN = %g (minimal volume for a deal)
\n",volume_min);
   StringAdd(output_string,temp_string);
//--- コンパクトな形式での double 値の出力
   double volume_step=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_STEP);
   temp_string=StringFormat("   SYMBOL_VOLUME_STEP = %g (minimal volume change step)
\n",volume_step);
   StringAdd(output_string,temp_string);
//--- コンパクトな形式での double 値の出力
   double volume_max=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MAX);
   temp_string=StringFormat("   SYMBOL_VOLUME_MAX = %g (maximal volume for a deal)
\n",volume_max);
```

© 2000–2025, MetaQuotes Ltd.

```
   StringAdd(output_string,temp_string);
//--- 契約金計算モードの決定
   int calc_mode=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_CALC_MODE);
   string str_calc_mode;
   switch(calc_mode)
     {
      case SYMBOL_CALC_MODE_FOREX:str_calc_mode="SYMBOL_CALC_MODE_FOREX
(Forex)";break;
      case SYMBOL_CALC_MODE_FUTURES:str_calc_mode="SYMBOL_CALC_MODE_FUTURES
(futures)";break;
      case SYMBOL_CALC_MODE_CFD:str_calc_mode="SYMBOL_CALC_MODE_CFD (CFD)";break;
      case SYMBOL_CALC_MODE_CFDINDEX:str_calc_mode="SYMBOL_CALC_MODE_CFDINDEX (CFD
for indices)";break;
      case SYMBOL_CALC_MODE_CFDLEVERAGE:str_calc_mode="SYMBOL_CALC_MODE_CFDLEVERAGE
(CFD at leverage trading)";break;
      case SYMBOL_CALC_MODE_EXCH_STOCKS:str_calc_mode="SYMBOL_CALC_MODE_EXCH_STOCKS
(trading securities on a stock exchange)";break;
      case
SYMBOL_CALC_MODE_EXCH_FUTURES:str_calc_mode="SYMBOL_CALC_MODE_EXCH_FUTURES (trading
futures contracts on a stock exchange)";break;
      case
SYMBOL_CALC_MODE_EXCH_FUTURES_FORTS:str_calc_mode="SYMBOL_CALC_MODE_EXCH_FUTURES_FOR
TS (trading futures contracts on FORTS)";break;
     }
//--- 文字列値の出力
   temp_string=StringFormat("   SYMBOL_TRADE_CALC_MODE = %s\n",
                            str_calc_mode);
   StringAdd(output_string,temp_string);
//--- 小数点以下の桁数が 2 桁の double 値の出力
   double margin_initial=SymbolInfoDouble(_Symbol,SYMBOL_MARGIN_INITIAL);
   temp_string=StringFormat("   SYMBOL_MARGIN_INITIAL = %.2f (initial margin)\n",
                            margin_initial);
   StringAdd(output_string,temp_string);
//--- 小数点以下の桁数が 2 桁の double 値の出力
   double margin_maintenance=SymbolInfoDouble(_Symbol,SYMBOL_MARGIN_MAINTENANCE);
   temp_string=StringFormat("   SYMBOL_MARGIN_MAINTENANCE = %.2f (maintenance
margin)\n",
                            margin_maintenance);
   StringAdd(output_string,temp_string);
//--- int 値の出力
   int freeze_level=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_FREEZE_LEVEL);
   temp_string=StringFormat("   SYMBOL_TRADE_FREEZE_LEVEL = %d (order freeze level
in points)",
                            freeze_level);
   StringAdd(output_string,temp_string);
   Print(output_string);
   Comment(output_string);
/* execution result
   Contract specification for EURUSD:
     SYMBOL_DIGITS = 5 (number of digits after the decimal point)
     SYMBOL_POINT = 0.00001 (point value)
     SYMBOL_SPREAD = 10 (current spread in points)
     SYMBOL_TRADE_STOPS_LEVEL = 18 (minimal indention in points for Stop orders)
     SYMBOL_TRADE_CONTRACT_SIZE = 100000 (contract size)
     SYMBOL_TRADE_TICK_SIZE = 0.000010 (minimal price change)
     SYMBOL_SWAP_MODE = SYMBOL_SWAP_MODE_POINTS (in points)
     SYMBOL_SWAP_LONG = -0.700000 (buy order swap value)
     SYMBOL_SWAP_SHORT = -1.000000 (sell order swap value)
     SYMBOL_TRADE_MODE = SYMBOL_TRADE_MODE_FULL (no trade restrictions)
     SYMBOL_VOLUME_MIN = 0.01 (minimal volume for a deal)
     SYMBOL_VOLUME_STEP = 0.01 (minimal volume change step)
```

© 2000–2025, MetaQuotes Ltd.

```
      SYMBOL_VOLUME_MAX = 500 (maximal volume for a deal)
      SYMBOL_TRADE_CALC_MODE = SYMBOL_CALC_MODE_FOREX (Forex)
      SYMBOL_MARGIN_INITIAL = 0.00 (initial margin)
      SYMBOL_MARGIN_MAINTENANCE = 0.00 (maintenance margin)
      SYMBOL_TRADE_FREEZE_LEVEL = 0 (order freeze level in points)
*/
  }
```

参照

PrintFormat、DoubleToString、ColorToString、TimeToString

© 2000–2025, MetaQuotes Ltd.

数学関数

数学関数及び三角関数のセット

数学関数はもともと、スカラー値に対して関連する演算をおこなうために設計されたものです。このビルド以降、ほとんどの関数が行列やベクトルに適用できるようになります。これには、MathAbs、MathArccos、MathArcsin、MathArctan、MathCeil、MathCos、MathExp、MathFloor、MathLog、MathLog10、MathMod、MathPow、MathRound、MathSin、MathSqrt、MathTan、MathExpm1、MathLog1p、MathArccosh、MathArcsinh、MathArctanh、MathCosh、MathSinh、MathTanhなどが含まれています。このような操作は、行列やベクトルを要素ごとに扱うことを意味します。例：

```
//---
  matrix a= {{1, 4}, {9, 16}};
  Print("matrix a=\n",a);
  a=MathSqrt(a);
  Print("MatrSqrt(a)=\n",a);
  /*
   matrix a=
   [[1,4]
    [9,16]]
   MatrSqrt(a)=
   [[1,2]
    [3,4]]
  */
```

MathModおよびMathPowの場合、2番目の要素には、適切なサイズのスカラーまたは行列/ベクトルを指定することができます。

関数	アクション
MathAbs	指定された数値の絶対値（係数）を返します。
MathArccos	ラジアンで x のアークコサインを返します。
MathArcsin	ラジアンで x のアークサインを返します。
MathArctan	ラジアンで x のアークタンジェントを返します。
MathClassify	浮動小数点数の型を戻します
MathCeil	切り上げされた整数値を返します。
MathCos	数のコサインを返します。
MathExp	数の指数を返します。
MathFloor	切り下げされた整数値を返します。
MathLog	自然対数を返します。
MathLog10	数の底が10の対数を返します。
MathMax	2 つの数値の最大値を返します。
MathMin	2 つの数値の最小値を返します。

© 2000–2025, MetaQuotes Ltd.

関数	アクション
MathMod	2つの数の分割後の実際の余りを返します。
MathPow	基数を指定された乗まで引き上げます。
MathRand	0～32,767の範囲内で擬似乱数値を返します。
MathRound	値を最も近い整数に丸めます。
MathSin	数のサインを返します。
MathSqrt	平方根を返します。
MathSrand	擬似ランダム整数の系列を生成するための開始点を設定します。
MathTan	数のタンジェントを返します。
MathIsValidNumber	実数の有効性をチェックします。
MathExpm1	MathExp(x)-1の式の値を返します。
MathLog1p	MathLog(1+x)の式の値を返します。
MathArccosh	双曲線逆余弦（アークコサイン）を返します。
MathArcsinh	双曲線逆正弦（アークサイン）を返します。
MathArctanh	双曲線逆正接（アークタンジェント）を返します。
MathCosh	双曲線余弦（コサイン）を返します。
MathSinh	双曲線正弦（サイン）を返します。
MathTanh	双曲線正接（タンジェント）を返します。
MathSwap	ushort/uint/ushort型の値のバイト順を変更します。

© 2000-2025, MetaQuotes Ltd.

MathAbs

指定された数値の絶対値（係数）を返します。

```
double  MathAbs(
   double  value      // 数値
   );
```

パラメータ

value

[in] 数値

戻り値

0以上のdouble型の値

注意事項

MathAbs() 関数の代わりにfabs()を使用できます。

例：

```
//--- 入力パラメータ
input int      InpValue = -10;   // 任意の値を入力

//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力された数値の絶対値を取得する
   uint abs_value=MathAbs(InpValue);
//--- 操作ログに値を出力する
   PrintFormat("The entered value %d received the value %u after using the MathAbs()
function",InpValue,abs_value);
  }
```

© 2000-2025, MetaQuotes Ltd.

MathArccos

xの逆余弦（0～π）をラジアン単位で返します。

```
double  MathArccos(
   double  val     // -1<val<1
   );
```

パラメータ

val

[in] 逆余弦が計算される、-1以上1以下の値

戻り値

ラジアン単位での数値の逆余弦（valが-1未満か1より大きい場合はNaN（不定値））

注意事項

MathArccos()関数の代わりに、acos()を使用できます。

例：

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/1000);
   delta[0]=-1;
//--- deltaステップで-1～2 piの101個の値を取得する
   vector X=delta.CumSum();
//--- Xベクトルの各値の逆余弦値を計算する
   vector Y=MathArccos(X);

//--- 計算された値をベクトルから配列に転送する
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- 計算されたベクトル値のグラフを描画する
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- EscapeキーまたはPgDnキーを押してグラフを削除し（スクリーンショットを撮り）、終了するまで待機する
```

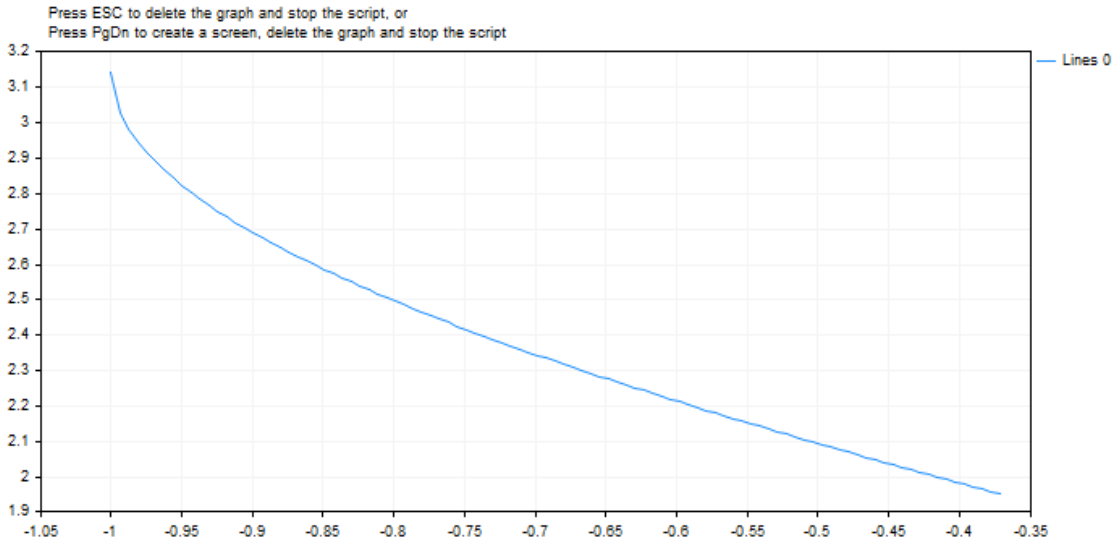
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- クリーンアップ
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//|  ESCが押されたら「true」を返す                                   |
//|  PgDnが押されたら、グラフのスクリーンショットを撮り、「true」を返す |
//|  その他の場合は「false」を返す                                   |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESCが押されたら「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDnが押されてグラフのスクリーンショットが正常に取得されたら、「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 &&
MakeAndSaveScreenshot(MQLInfoString(MQL_PROGRAM_NAME)+"_Screenshot"))
      return(true);
//--- 「false」を返す
   return(false);
  }
//+------------------------------------------------------------------+
//|  グラフオブジェクトを作成して曲線を描く                          |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the
script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//|  スクリーンショットを撮り、画像をファイルに保存する              |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
```

© 2000–2025, MetaQuotes Ltd.

```
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*",
FSD_WRITE_FILE, file_names, file_name+".png");
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d",
__FUNCTION__, GetLastError());
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

結果:



参照

実数型（double、float）、統計、科学的チャート、クライアント端末プロパティ

© 2000–2025, MetaQuotes Ltd.

MathArcsin

xの逆正弦(-π/2 ～ π/2 ラジアン)を返します。

```
double  MathArcsin(
   double  val      // -1<value<1
   );
```

パラメータ

val

[in] 逆正弦が計算される、-1以上1以下の値

戻り値

ラジアン単位でのvalの逆正弦(-π/2 ～ π/2ラジアン)。valが-1未満か1より大きい場合、この関数はNaN(不定値)を返します。

注意事項

MathArcsin()関数の代わりにasin()を使用できます。

例：

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/1000);
   delta[0]=-1;
//--- deltaステップで-1～2 piの101個の値を取得する
   vector X=delta.CumSum();
//--- Xベクトルの各値の逆正弦値を計算する
   vector Y=MathArcsin(X);

//--- 計算された値をベクトルから配列に転送する
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- 計算されたベクトル値のグラフを描画する
   CurvePlot(x_array,y_array,clrDodgerBlue);
```

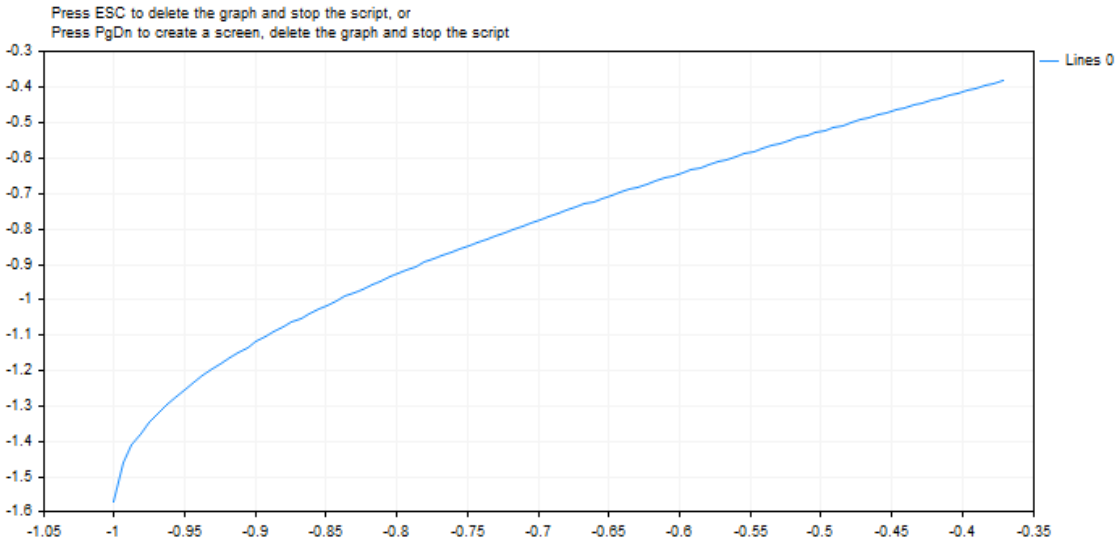
© 2000-2025, MetaQuotes Ltd.

```
//--- EscapeキーまたはPgDnキーを押してグラフを削除し（スクリーンショットを撮り）、終了するまで待機する
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- クリーンアップ
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//|  ESCが押されたら「true」を返す                                   |
//|  PgDnが押されたら、グラフのスクリーンショットを撮り、「true」を返す |
//|  その他の場合は「false」を返す                                   |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESCが押されたら「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDnが押されてグラフのスクリーンショットが正常に取得されたら、「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 &&
MakeAndSaveScreenshot(MQLInfoString(MQL_PROGRAM_NAME)+"_Screenshot"))
      return(true);
//--- 「false」を返す
   return(false);
  }
//+------------------------------------------------------------------+
//|  グラフオブジェクトを作成して曲線を描く                          |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the
script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//|  スクリーンショットを撮り、画像をファイルに保存する              |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*",
FSD_WRITE_FILE, file_names, file_name+".png");
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d",
__FUNCTION__, GetLastError());
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

結果:



参照

実数型（double、float）、統計、科学的チャート、クライアント端末プロパティ

© 2000–2025, MetaQuotes Ltd.

MathArctan

xの逆正接を返します。xが0に等しい場合は、0を返します。

```
double  MathArctan(
   double  value      // 逆正接
   );
```

パラメータ

value

[in] 逆正接を表現する数

戻り値

MathArctan は -π/2 ～ π/2 ラジアンの値を返します。

注意事項

MathArctan()関数の代わりにatan()を使用できます。

例：

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/100);
   delta[0]=0;
//--- deltaステップで-1～2 piの101個の値を取得する
   vector X=delta.CumSum();
//--- Xベクトルの各値の逆正接値を計算する
   vector Y=MathArctan(X);

//--- 計算された値をベクトルから配列に転送する
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- 計算されたベクトル値のグラフを描画する
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- EscapeキーまたはPgDnキーを押してグラフを削除し（スクリーンショットを撮り）、終了するまで待機する
```

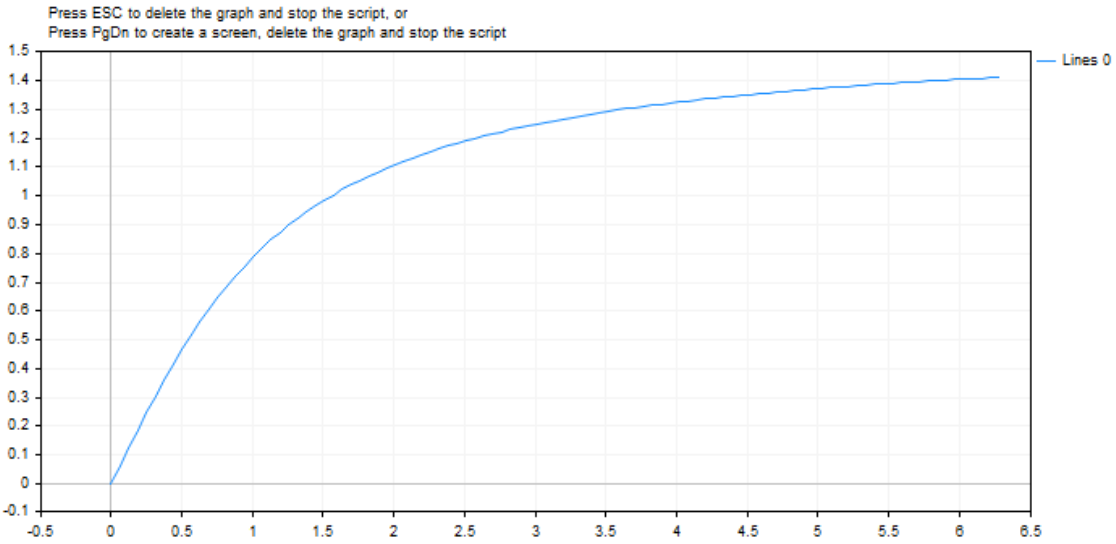
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- クリーンアップ
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//|  ESCが押されたら「true」を返す                                   |
//|  PgDnが押されたら、グラフのスクリーンショットを撮り、「true」を返す |
//|  その他の場合は「false」を返す                                   |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESCが押されたら「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDnが押されてグラフのスクリーンショットが正常に取得されたら、「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 &&
MakeAndSaveScreenshot(MQLInfoString(MQL_PROGRAM_NAME)+"_Screenshot"))
      return(true);
//--- 「false」を返す
   return(false);
  }
//+------------------------------------------------------------------+
//|  グラフオブジェクトを作成して曲線を描く                          |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the
script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//|  スクリーンショットを撮り、画像をファイルに保存する              |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
```

© 2000–2025, MetaQuotes Ltd.

```
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*",
FSD_WRITE_FILE, file_names, file_name+".png");
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d",
__FUNCTION__, GetLastError());
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

結果:



参照

実数型（double、float）、統計、科学的チャート、クライアント端末プロパティ

© 2000–2025, MetaQuotes Ltd.

MathArctan2

正接が指定された2つの数値の商である角度（ラジアン単位）を返します。

```
double MathArctan2(
 double y      // 点のy座標
 double x      // 点のx座標
  );
```

パラメータ

y

[in] Y座標値

x

[in] X座標値

戻り値

MathArctan2は、MathTan(θ)=y/xになるような、-πからπラジアンの範囲内の角度θを返します。

以下の点に注意してください。

- 象限1の(x, y)の場合、0< θ< π/2
- 象限2の(x, y)の場合、π/2< θ ≤ π
- 象限3の(x, y)の場合、-π < θ < -π/2
- 象限4の(x, y)の場合、-π/2 < θ < 0

象限の境界上の点の場合、戻り値は次のようになります。

- yが0でxが負でない場合、θ= 0。
- yが0でxが負の場合、θ= π。
- yが正でxが0の場合、θ= π/2。
- yが負でxが0の場合、θ= -π/2。
- y が0でxが0の場合、θ= 0。

注意事項

MathArctan2()関数の代わりにatan2()を使用できます。

例：

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
```

© 2000-2025, MetaQuotes Ltd.

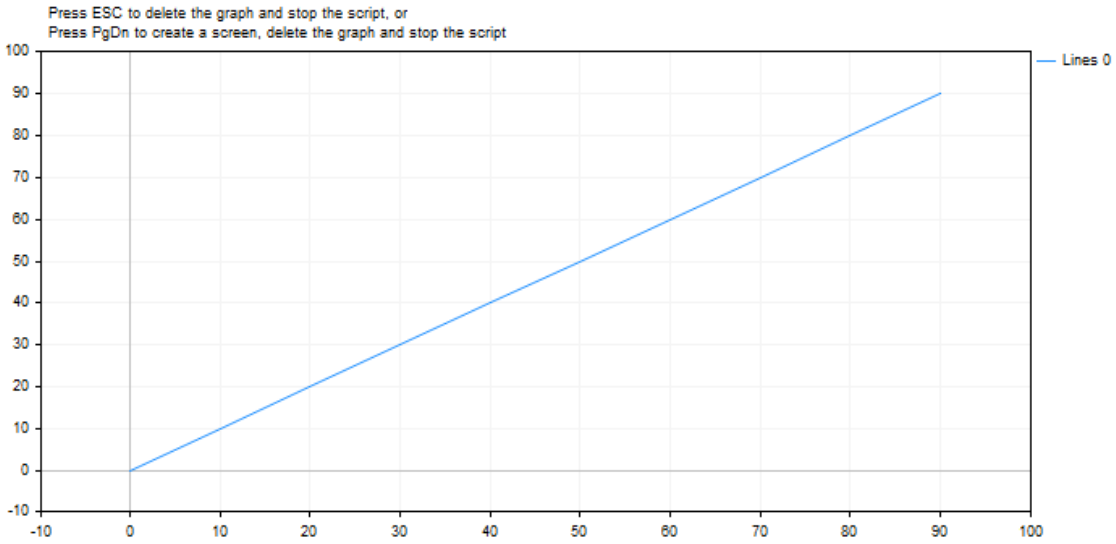
```
  {
   vector delta=vector::Full(10,10);
   delta[0]=0;
//--- deltaステップで-1～2 piの101個の値を取得する
   vector X=delta.CumSum();
//--- Xベクトルの各値の逆正接値を計算する
   vector Y=delta.CumSum();

   Print("vector delta = \n",delta);
   Print("vector X = \n",X);
   Print("vector Y = \n",Y);

//--- 計算された値をベクトルから配列に転送する
   double x_array[];;
   double y_array[];;
   X.Swap(x_array);
   Y.Swap(y_array);

   double array[10];
   for(int i=0; i<10; i++)
     {
      array[i]=MathArctan2(y_array[i],x_array[i]);
     }

//--- 計算されたベクトル値のグラフを描画する
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- EscapeキーまたはPgDnキーを押してグラフを削除し（スクリーンショットを撮り）、終了するまで待機する
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- クリーンアップ
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//|  ESCが押されたら「true」を返す                                   |
//|  PgDnが押されたら、グラフのスクリーンショットを撮り、「true」を返す |
//|  その他の場合は「false」を返す                                   |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESCが押されたら「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDnが押されてグラフのスクリーンショットが正常に取得されたら、「true」を返す
```

© 2000-2025, MetaQuotes Ltd.

```
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 &&
MakeAndSaveScreenshot(MQLInfoString(MQL_PROGRAM_NAME)+"_Screenshot"))
      return(true);
//--- 「false」を返す
   return(false);
  }
//+------------------------------------------------------------------+
//|  グラフオブジェクトを作成して曲線を描く                                  |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the
script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//|  スクリーンショットを撮り、画像をファイルに保存する                          |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*",
FSD_WRITE_FILE, file_names, file_name+".png");
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d",
__FUNCTION__, GetLastError());
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

結果：

© 2000–2025, MetaQuotes Ltd.



参照

実数型（double、float）、統計、科学的チャート、クライアント端末プロパティ

© 2000–2025, MetaQuotes Ltd.

MathClassify

浮動小数点数の型を判別して、ENUM_FP_CLASS列挙からの値として結果を返します

```
ENUM_FP_CLASS  MathClassify(
   double  value      // real number
   );
```

パラメータ

value

[in] 確認対象の浮動小数点数

戻り値

ENUM_FP_CLASS列挙体からの値

ENUM_FP_CLASS

ID	説明
FP_SUBNORMAL	表現可能な最小の正規数 DBL_MIN(2.2250738585072014e-308)よりもゼロに近い非正規数
FP_NORMAL	2.2250738585072014e-308～ 1.7976931348623158e+308の範囲にある正規数
FP_ZERO	正または負のゼロ
FP_INFINITE	適切な型で表すことができない数、正または負の無限大
FP_NAN	数字でない

例:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- test NaN
  double nan=double("nan");
  PrintFormat("Test NaN: %G is %s, MathIsValidNumber(NaN)=%s",
              nan,
              EnumToString(MathClassify(nan)),
              (string)MathIsValidNumber(nan));
//--- test infinity
  double inf=double("inf");
  PrintFormat("Test Inf: %G is %s, MathIsValidNumber(inf)=%s",
              inf,
              EnumToString(MathClassify(inf)),
```

© 2000-2025, MetaQuotes Ltd.

```
               (string)MathIsValidNumber(inf));
//--- test normal value
  double normal=1.2345e6;
  PrintFormat("Test Normal: %G is %s, MathIsValidNumber(normal)=%s",
               normal,
               EnumToString(MathClassify(normal)),
               (string)MathIsValidNumber(normal));
//--- test subnormal value
  double sub_normal=DBL_MIN/2.0;
  PrintFormat("Test Subnormal: %G is %s, MathIsValidNumber(sub_normal)=%s",
               sub_normal,
               EnumToString(MathClassify(sub_normal)),
               (string)MathIsValidNumber(sub_normal));
//--- test zero value
  double zero=0.0/(-1);
  PrintFormat("Test Zero: %G is %s, MathIsValidNumber(zero)=%s",
               zero,
               EnumToString(MathClassify(zero)),
               (string)MathIsValidNumber(zero));
 }
 /*
 Result:
   Test NaN: NAN is FP_NAN, MathIsValidNumber(NaN)=false
   Test Inf: INF is FP_INFINITE, MathIsValidNumber(inf)=false
   Test Normal: 1.2345E+06 is FP_NORMAL, MathIsValidNumber(normal)=true
   Test Subnormal: 1.11254E-308 is FP_SUBNORMAL, MathIsValidNumber(sub_normal)=true
   Test Zero: -0 is FP_ZERO, MathIsValidNumber(zero)=true
*/
//+------------------------------------------------------------------+
```

参照

浮動小数点数型(ダブル、フロート), MathIsValidNumber

© 2000-2025, MetaQuotes Ltd.

MathCeil

上から最も近い整数値を返します。

```
double  MathCeil(
   double  val      // 数値
   );
```

パラメータ

val

[in] 数値

戻り値

val以上の最小の整数を表す数値

注意事項

MathCeil()関数の代わりにceil()を使用できます。

例：

```
#define VALUES_TOTAL 31

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 変換用の変数を宣言する
   double value=0;                     // MathCeil 変換用の実数
   int    ceil_value=0;                // ここで結果を取得する
//--- 実数の小数増分の数によるループ
   for(int i=0; i<VALUES_TOTAL; i++)
     {
      //--- 数値を増やす
      //--- 上から最も近い整数値を取得する
      //--- 操作ログに制御値を表示する
      value+=0.1;
      ceil_value=(int)MathCeil(NormalizeDouble(value,1));
      PrintFormat("value: %.1f, ceil value: %d",value,ceil_value);
      /*
     結果:
      value: 0.1, ceil value: 1
      value: 0.2, ceil value: 1
      value: 0.3, ceil value: 1
      value: 0.4, ceil value: 1
      value: 0.5, ceil value: 1
      value: 0.6, ceil value: 1
```

© 2000-2025, MetaQuotes Ltd.

```
      value: 0.7, ceil value: 1
      value: 0.8, ceil value: 1
      value: 0.9, ceil value: 1
      value: 1.0, ceil value: 1
      value: 1.1, ceil value: 2
      value: 1.2, ceil value: 2
      value: 1.3, ceil value: 2
      value: 1.4, ceil value: 2
      value: 1.5, ceil value: 2
      value: 1.6, ceil value: 2
      value: 1.7, ceil value: 2
      value: 1.8, ceil value: 2
      value: 1.9, ceil value: 2
      value: 2.0, ceil value: 2
      value: 2.1, ceil value: 3
      value: 2.2, ceil value: 3
      value: 2.3, ceil value: 3
      value: 2.4, ceil value: 3
      value: 2.5, ceil value: 3
      value: 2.6, ceil value: 3
      value: 2.7, ceil value: 3
      value: 2.8, ceil value: 3
      value: 2.9, ceil value: 3
      value: 3.0, ceil value: 3
      value: 3.1, ceil value: 4
      */
    }
}
```

© 2000–2025, MetaQuotes Ltd.

MathCos

角度の余弦を返します。

```
double  MathCos(
   double  value      // 数値
   );
```

パラメータ

value

[in] ラジアンで表した角度

戻り値

−1～1のdouble型の値

注意事項

MathCos()の代わりにcos()を使用できます。

例：

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/100);
   delta[0]=0;
//--- deltaステップで−1～2 piの101個の値を取得する
   vector X=delta.CumSum();
//--- Xベクトルの各値の余弦値を計算する
   vector Y=MathCos(X);

//--- 計算された値をベクトルから配列に転送する
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- 計算されたベクトル値のグラフを描画する
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- EscapeキーまたはPgDnキーを押してグラフを削除し（スクリーンショットを撮り）、終了するまで待機する
```

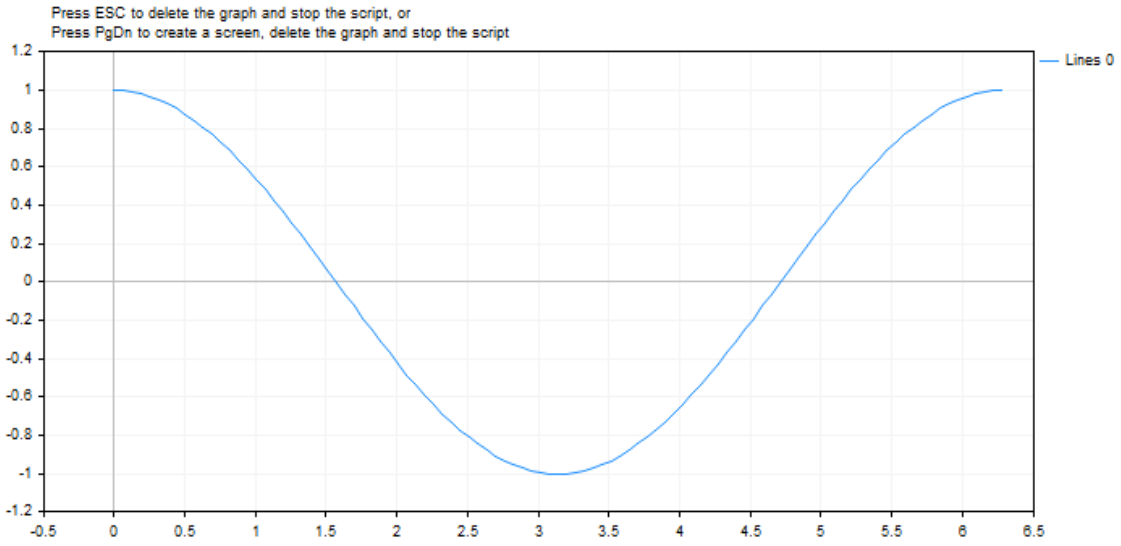
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- クリーンアップ
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//|  ESCが押されたら「true」を返す                                   |
//|  PgDnが押されたら、グラフのスクリーンショットを撮り、「true」を返す |
//|  その他の場合は「false」を返す                                   |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESCが押されたら「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDnが押されてグラフのスクリーンショットが正常に取得されたら、「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 &&
MakeAndSaveScreenshot(MQLInfoString(MQL_PROGRAM_NAME)+"_Screenshot"))
      return(true);
//--- 「false」を返す
   return(false);
  }
//+------------------------------------------------------------------+
//|  グラフオブジェクトを作成して曲線を描く                          |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the
script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//|  スクリーンショットを撮り、画像をファイルに保存する              |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
```

© 2000–2025, MetaQuotes Ltd.

```
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*",
FSD_WRITE_FILE, file_names, file_name+".png");
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d",
__FUNCTION__, GetLastError());
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

結果:



参照

実数型(double、float)、統計、科学的チャート、クライアント端末プロパティ

© 2000-2025, MetaQuotes Ltd.

MathExp

eのd乗を返します。

```
double  MathExp(
   double  value      // eの塁乗指数
   );
```

パラメータ

value

[in] 塁乗指数を指定する数値

戻り値

double型の数値（オーバーフローではINF（無限大）、アンダーフローでは0）

注意事項

MathExp()の代わりにexp()を使用できます。

例：

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- step 1で0～8の9個の値を取得する
   vector X(9,VectorArange);
   Print("vector X = \n",X);
//--- 各Xベクトル値の「e」(オイラー数)を計算する
   X=MathExp(X);
   Print("MathExp(X) = \n",X);

//--- 計算された値をベクトルから配列に転送する
   double y_array[];
   X.Swap(y_array);

//--- 計算されたベクトル値のグラフを描画する
   CurvePlot(y_array,clrDodgerBlue);

//--- Escapeキーまたは PgDn キーを押してグラフを削除し（スクリーンショットを撮り）、終了するまで待機する
   while(!IsStopped())
```

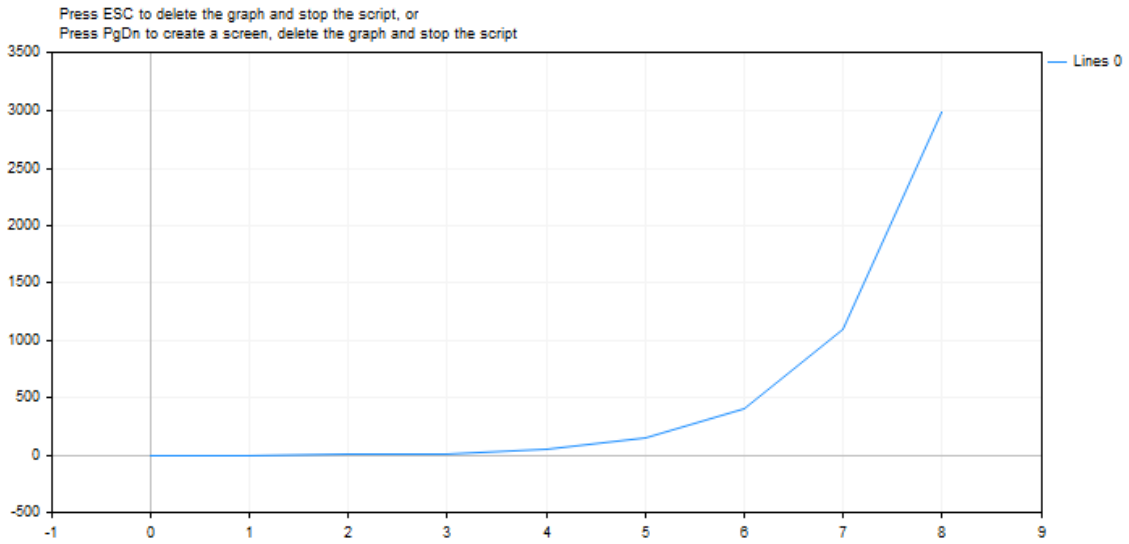
© 2000-2025, MetaQuotes Ltd.

```
      {
       if(StopKeyPressed())
          break;
       Sleep(16);
      }

//--- クリーンアップ
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//|  ベクトルに「value」を「step」単位で入力する                          |
//+------------------------------------------------------------------+
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
//+------------------------------------------------------------------+
//|  ESCが押されたら「true」を返す                                     |
//|  PgDnが押されたら、グラフのスクリーンショットを撮り、「true」を返す      |
//|  その他の場合は「false」を返す                                     |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESCが押されたら「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDnが押されてグラフのスクリーンショットが正常に取得されたら、「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 &&
MakeAndSaveScreenshot(MQLInfoString(MQL_PROGRAM_NAME)+"_Screenshot"))
      return(true);
//--- 「false」を返す
   return(false);
  }
//+------------------------------------------------------------------+
//|  グラフオブジェクトを作成して曲線を描く                              |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the
script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
```

© 2000-2025, MetaQuotes Ltd.

```
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//|  スクリーンショットを撮り、画像をファイルに保存する                |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*",
FSD_WRITE_FILE, file_names, file_name+".png");
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d",
__FUNCTION__, GetLastError());
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

結果:



参照

実数型（double、float）、統計、科学的チャート、クライアント端末プロパティ

© 2000–2025, MetaQuotes Ltd.

MathFloor

下から最も近い整数値を返します。

```
double  MathFloor(
   double  val     // 数値
   );
```

パラメータ

val

[in] 数値

戻り値

val以下の最大の整数を表す数値

注意事項

MathFloor()の代わりにfloor()を使用できます。

例：

```
#define VALUES_TOTAL 31

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 変換用の変数を宣言する
   double value=0;                     // MathFloor変換用の実数
   int    floor_value=0;               // ここで結果を取得する
//--- 実数の小数増分の数によるループ
   for(int i=0; i<VALUES_TOTAL; i++)
     {
      //--- 数値を増やす
      //--- 下から最も近い整数値を取得する
      //--- 操作ログに制御値を表示する
      value+=0.1;
      floor_value=(int)MathFloor(NormalizeDouble(value,1));
      PrintFormat("value: %.1f, floor value: %d",value,floor_value);
      /*
     結果:
      value: 0.1, floor value: 0
      value: 0.2, floor value: 0
      value: 0.3, floor value: 0
      value: 0.4, floor value: 0
      value: 0.5, floor value: 0
      value: 0.6, floor value: 0
```

© 2000-2025, MetaQuotes Ltd.

```
      value: 0.7, floor value: 0
      value: 0.8, floor value: 0
      value: 0.9, floor value: 0
      value: 1.0, floor value: 1
      value: 1.1, floor value: 1
      value: 1.2, floor value: 1
      value: 1.3, floor value: 1
      value: 1.4, floor value: 1
      value: 1.5, floor value: 1
      value: 1.6, floor value: 1
      value: 1.7, floor value: 1
      value: 1.8, floor value: 1
      value: 1.9, floor value: 1
      value: 2.0, floor value: 2
      value: 2.1, floor value: 2
      value: 2.2, floor value: 2
      value: 2.3, floor value: 2
      value: 2.4, floor value: 2
      value: 2.5, floor value: 2
      value: 2.6, floor value: 2
      value: 2.7, floor value: 2
      value: 2.8, floor value: 2
      value: 2.9, floor value: 2
      value: 3.0, floor value: 3
      value: 3.1, floor value: 3
      */
     }
```

© 2000–2025, MetaQuotes Ltd.

MathLog

自然対数を返します。

```
double  MathLog(
   double  val      // 対数を取る値
   );
```

パラメータ

val

[in] 対数を取る値

戻り値

成功の場合valの自然対数。valが負の場合はNaN(不定値)。valが0に等しい場合はINF(無限大)。

注意事項

MathLog()の代わりにlog()を使用できます。

例:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- step 1で0～8の9個の値を取得する
   vector X(9,VectorArange);
   Print("vector X = \n",X);
//--- Xベクトルの各値の対数を計算する
   X=MathLog(X);
   Print("MathLog(X) = \n",X);

//--- 計算された値をベクトルから配列に転送する
   double y_array[];
   X.Swap(y_array);

//--- 計算されたベクトル値のグラフを描画する
   CurvePlot(y_array,clrDodgerBlue);

//--- Escapeキーまたは PgDnキーを押してグラフを削除し(スクリーンショットを撮り)、終了するまで待機する
   while(!IsStopped())
```

© 2000-2025, MetaQuotes Ltd.

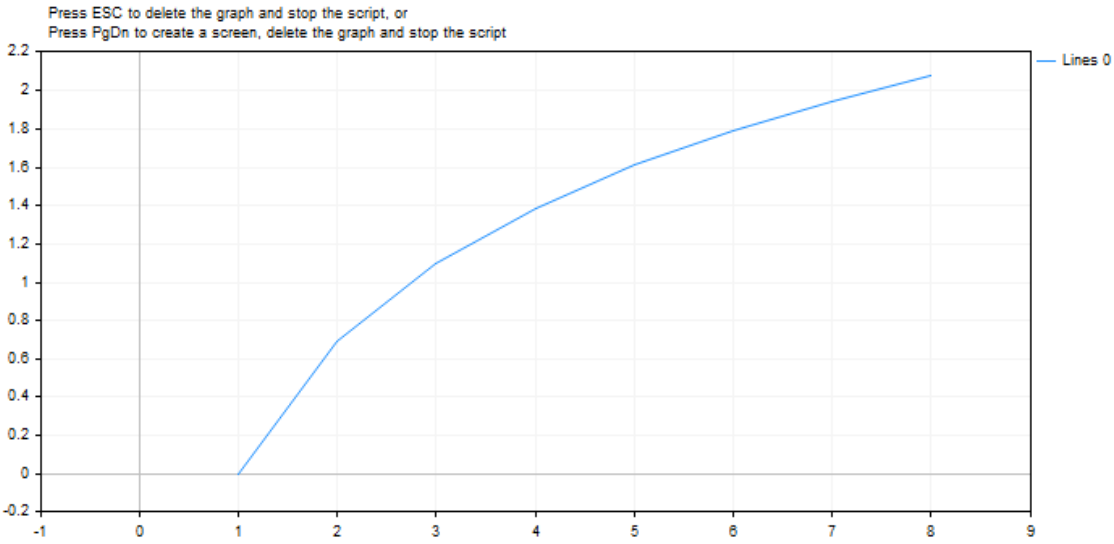
```
      {
       if(StopKeyPressed())
          break;
       Sleep(16);
      }

//--- クリーンアップ
   ExtGraph.Destroy();
   /*
  結果:
   vector X =
   [0,1,2,3,4,5,6,7,8]
   MathLog(X) =
   [-
inf
,
0
,
0
.6931471805599453
,
1
.09861228866811
,
1
.386294361119891
,1.6094379124341,1.791759469228055,1.945910149055313,2.079441541679836]
   */
  }
//+------------------------------------------------------------------+
//| ベクトルに「value」を「step」単位で入力する                                |
//+------------------------------------------------------------------+
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
//+------------------------------------------------------------------+
//| ESCが押されたら「true」を返す                                         |
//| PgDnが押されたら、グラフのスクリーンショットを撮り、「true」を返す                  |
//| その他の場合は「false」を返す                                        |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESCが押されたら「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDnが押されてグラフのスクリーンショットが正常に取得されたら、「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 &&
MakeAndSaveScreenshot(MQLInfoString(MQL_PROGRAM_NAME)+"_Screenshot"))
```

© 2000-2025, MetaQuotes Ltd.

```
      return(true);
//--- 「false」を返す
   return(false);
  }
//+------------------------------------------------------------------+
//|  グラフオブジェクトを作成して曲線を描く                                  |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the
script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//|  スクリーンショットを撮り、画像をファイルに保存する                          |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*",
FSD_WRITE_FILE, file_names, file_name+".png");
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d",
__FUNCTION__, GetLastError());
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

結果：

© 2000–2025, MetaQuotes Ltd.



参照

実数型（double、float）、統計、科学的チャート、クライアント端末プロパティ

© 2000–2025, MetaQuotes Ltd.

MathLog

数の底が10の対数を返します。

```
double  MathLog10(
   double  val      // 対数を取る数値
   );
```

パラメータ

val

[in] 対数を取る数値

戻り値

成功の場合は対数。valが負の場合はNaN(不定値)。valが0に等しい場合はINF(無限大)。

注意事項

MathLog10()の代わりにlog10()も使用できます。

例:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- step 1で0～8の9個の値を取得する
   vector X(9,VectorArange);
   Print("vector X = \n",X);
//--- Xベクトルの各値の底が10の対数を計算する
   X=MathLog10(X);
   Print("MathLog10(X) = \n",X);

//--- 計算された値をベクトルから配列に転送する
   double y_array[];
   X.Swap(y_array);

//--- 計算されたベクトル値のグラフを描画する
   CurvePlot(y_array,clrDodgerBlue);

//--- Escapeキーまたは PgDn キーを押してグラフを削除し(スクリーンショットを撮り)、終了するまで待機する
   while(!IsStopped())
```

© 2000-2025, MetaQuotes Ltd.

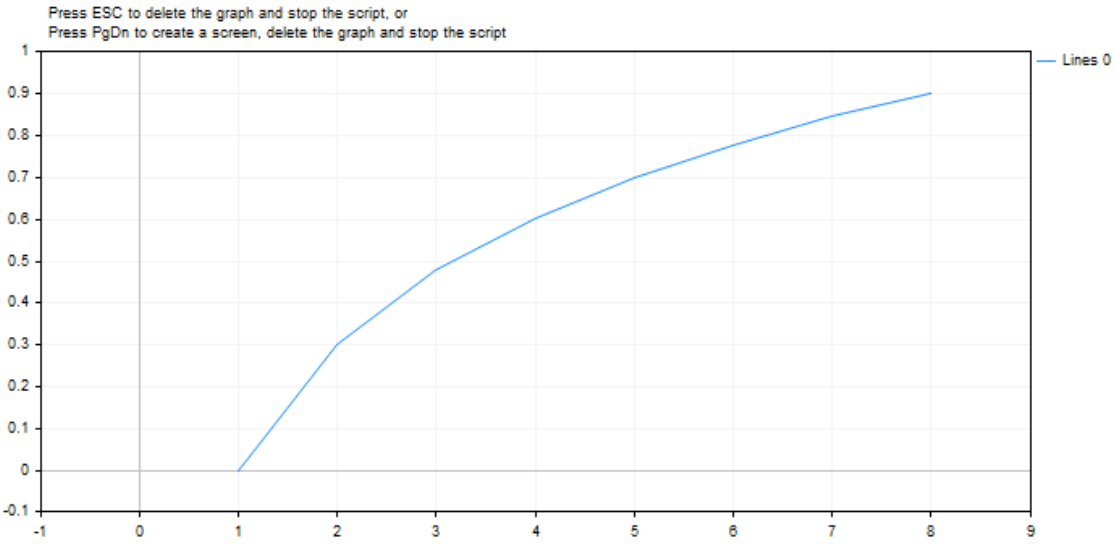
```
      {
       if(StopKeyPressed())
          break;
       Sleep(16);
      }

//--- クリーンアップ
   ExtGraph.Destroy();
   /*
  結果:
   vector X =
   [0,1,2,3,4,5,6,7,8]
   MathLog10(X) =
   [-
inf
,
0
,
0
.3010299956639812
,
0
.4771212547196624
,
0
.6020599913279624
,0.6989700043360189,0.7781512503836436,0.8450980400142568,0.9030899869919435]
   */
  }
//+------------------------------------------------------------------+
//|  ベクトルに「value」を「step」単位で入力する                          |
//+------------------------------------------------------------------+
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
//+------------------------------------------------------------------+
//|  ESCが押されたら「true」を返す                                     |
//|  PgDnが押されたら、グラフのスクリーンショットを撮り、「true」を返す      |
//|  その他の場合は「false」を返す                                     |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESCが押されたら「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDnが押されてグラフのスクリーンショットが正常に取得されたら、「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 &&
MakeAndSaveScreenshot(MQLInfoString(MQL_PROGRAM_NAME)+"_Screenshot"))
```

© 2000-2025, MetaQuotes Ltd.

```
      return(true);
//--- 「false」を返す
   return(false);
  }
//+------------------------------------------------------------------+
//|  グラフオブジェクトを作成して曲線を描く                                  |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the
script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//|  スクリーンショットを撮り、画像をファイルに保存する                          |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*",
FSD_WRITE_FILE, file_names, file_name+".png");
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d",
__FUNCTION__, GetLastError());
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

結果:

© 2000-2025, MetaQuotes Ltd.



参照

実数型（double、float）、統計、科学的チャート、クライアント端末プロパティ

© 2000–2025, MetaQuotes Ltd.

MathMax

2つの値の最大値を返します。

```
double  MathMax(
   double  value1,      // 1番目の値
   double  value2       // 2番目の値
   );
```

パラメータ

value1

[in] 1番目の数値

value2

[in] 2番目の数値

戻り値

2つの値のうちの最大値

注意事項

MathMax()の代わりにfmax()を使用できます。fmax()、fmin()、MathMax()、MathMin()はdouble型への型キャストなしで整数型での操作が出来ます。

異なる型のパラメータが関数に渡された場合、ビットサイズの小さいパラメータが自動的にビットサイズの大きいパラメータの型にキャストされます。戻り値の型はビットサイズの大きいパラメータの型と同じです。

同じ型のデータが渡された場合、キャストは行われません。

例：

```
//--- 入力パラメータ
input int                  InpPeriod = 10;            // 平均移動線の計算期間
input ENUM_MA_METHOD       InpMethod = MODE_SMA;      // 平均移動線の計算法
input ENUM_APPLIED_PRICE   InpPrice  = PRICE_CLOSE;   // 平均移動線の計算価格

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 移動平均期間が1未満の値に設定されている場合は、デフォルト値(10)が使用される
   int period=(InpPeriod<1 ? 10 : InpPeriod);
//--- 移動平均指標ハンドルを作成する
   int handle=iMA(Symbol(),Period(),period,0,InpMethod,InpPrice);
   if(handle==INVALID_HANDLE)
     {
      Print("Failed to create the Moving Average indicator handle. Error
",GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }
//--- 現在のBid値を取得する
   double bid=0;
   ResetLastError();
   if(!SymbolInfoDouble(Symbol(),SYMBOL_BID,bid))
     {
      Print("Failed to get Bid price. Error ",GetLastError());
      return;
     }
//--- 現在の足の移動平均値を取得する
   double array[1];
   int    copied=CopyBuffer(handle,0,0,1,array);
   if(copied!=1)
     {
      Print("Failed to get Moving Average data. Error ",GetLastError());
      return;
     }
//--- 2つの価格（Bid価格と移動平均値）のうち最も高い価格を取得し、結果のデータを操作ログに表示する
   double max_price=MathMax(bid,array[0]);
   PrintFormat("Bid: %.*f, Moving Average: %.*f, highest price of the two: %.*f",_Digits,bid,_Digits,array[0],_Digits,max_price);
   PrintFormat("Bid price %s moving average",(bid>array[0] ? "higher" : bid<array[0] ? "lower" : "equal to"));
  }
```

© 2000-2025, MetaQuotes Ltd.

MathMin

2つの数値の最小値を返します。

```
double  MathMin(
   double  value1,      // 1番目の値
   double  value2       // 2番目の値
   );
```

パラメータ

value1

[in] 1番目の数値

value2

[in] 2番目の数値

戻り値

2つの値のうちの最小値

注意事項

MathMin()の代わりにfmin()を使用できます。fmax()、fmin()、MathMax()、MathMin()はdouble型への型キャストなしで整数型での操作が出来ます。

異なる型のパラメータが関数に渡された場合、ビットサイズの小さいパラメータが自動的にビットサイズの大きいパラメータの型にキャストされます。戻り値の型はビットサイズの大きいパラメータの型と同じです。

同じ型のデータが渡された場合、キャストは行われません。

例：

```
//--- 入力パラメータ
input uint  InpBars  =  100000;  // 望みのバー数

//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- サーバー上で利用可能なバーの数を取得する
   uint bars_total = Bars(Symbol(),Period());
   if(bars_total==0)
     {
      PrintFormat("Data for timeseries %s %s not yet generated in the terminal.
Please try again later.",Symbol(),StringSubstr(EnumToString(Period()),7));
      return;
     }
//--- サーバー上で利用可能な値と設定で要求された数の2つの値からバーの最小数を取得する
   int bars = (int)MathMin(bars_total,InpBars);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- サーバーで利用可能なバーよりも多くのバーが要求された場合は、操作ログで報告する
   if(bars_total<InpBars)
      PrintFormat("Number of bars on the server (%u) is less than requested (%u)",bars_total,InpBars);
//--- 操作ログで作業可能なバーの数を表示する
   Print("Bars available for work: ",bars);
  }
```

© 2000–2025, MetaQuotes Ltd.

MathMod

2つの数値の分割後の実際の余りを返します。

```
double  MathMod(
   double  value,       // 被除数値
   double  value2       // 除数値
   );
```

パラメータ

value

[in] 被除数値

value2

[in] 除数値

戻り値

MathMod関数は「val = i * y + f」でのfを「val/y」の余りとして計算します。ここでiは整数、fとvalは同じ符号を有し、f の絶対値はyの絶対値未満です。

注意事項

MathMod()の代わりにfmod()を使用できます。

例：

```
#property script_show_inputs

//--- 入力パラメータ
input double   InpDividentValue  =  10;   // 被除数値
input double   InpDivisorValue   =  3;    // 除数値

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力された数値の除算の実剰余を取得する
   double res=MathMod(InpDividentValue,InpDivisorValue);
//--- 結果を操作ログに出力する
   PrintFormat("Real remainder when dividing %.2f by %.2f =
%.2f",InpDividentValue,InpDivisorValue,res);
  }
```

© 2000-2025, MetaQuotes Ltd.

MathPow

基数を指定された乗まで引き上げます。

```
double  MathPow(
   double  base,          // 底
   double  exponent       // 指数値
   );
```

パラメータ

base

[in] 底

exponent

[in] 指数値

戻り値

基数を指定された乗まで引き上げた値

注意事項

MathPow()の代わりにpow()を使用できます。

例：

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#property script_show_inputs

#include <Graphics\Graphic.mqh>

//--- 入力パラメータ
input double   InpExponentValue  =  2;    // 指数値

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- step 1で0～10の11の値を取得する
   vector X(11,VectorArange);
   Print("vector X = \n",X);
//--- X ベクトルの各値を InpExponentValue 乗して計算する
   X=MathPow(X,InpExponentValue);
   Print("MathPow(X,",(string)InpExponentValue,") = \n",X);
```

© 2000-2025, MetaQuotes Ltd.

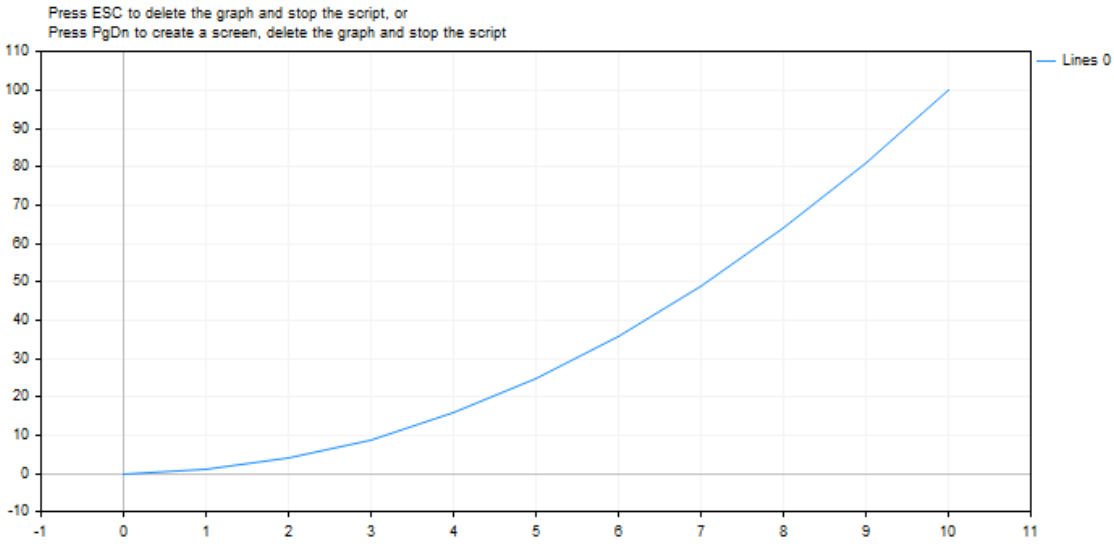
```
//--- 計算された値をベクトルから配列に転送する
   double y_array[];
   X.Swap(y_array);

//--- 計算されたベクトル値のグラフを描画する
   CurvePlot(y_array,clrDodgerBlue);

//--- Escapeキーまたは PgDnキーを押してグラフを削除し(スクリーンショットを撮り)、終了するまで待機する
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- クリーンアップ
   ExtGraph.Destroy();
   /*
  結果:
   vector X =
   [0,1,2,3,4,5,6,7,8,9,10]
   MathPow(X,2.0) =
   [0,1,4,9,16,25,36,49,64,81,100]
   */
  }
//+------------------------------------------------------------------+
//|  ベクトルに「value」を「step」単位で入力する                          |
//+------------------------------------------------------------------+
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
//+------------------------------------------------------------------+
//|  ESCが押されたら「true」を返す                                     |
//|  PgDnが押されたら、グラフのスクリーンショットを撮り、「true」を返す      |
//|  その他の場合は「false」を返す                                     |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESCが押されたら「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDnが押されてグラフのスクリーンショットが正常に取得されたら、「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 &&
MakeAndSaveScreenshot(MQLInfoString(MQL_PROGRAM_NAME)+"_Screenshot"))
      return(true);
//--- 「false」を返す
```

© 2000-2025, MetaQuotes Ltd.

```
    return(false);
   }
//+------------------------------------------------------------------+
//|  グラフオブジェクトを作成して曲線を描く                          |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
   {
    ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
    ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
    ExtGraph.IndentUp(30);
    ExtGraph.CurvePlotAll();
    string text1="Press ESC to delete the graph and stop the script, or";
    string text2="Press PgDn to create a screen, delete the graph and stop the
script";
    ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
    ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
    ExtGraph.Update();
   }
//+------------------------------------------------------------------+
//|  スクリーンショットを撮り、画像をファイルに保存する              |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
   {
    string file_names[];
    ResetLastError();
    int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*",
FSD_WRITE_FILE, file_names, file_name+".png");
    if(selected<1)
      {
       if(selected<0)
          PrintFormat("%s: FileSelectDialog() function returned error %d",
__FUNCTION__, GetLastError());
       return false;
      }

    bool res=false;
    if(ChartSetInteger(0,CHART_SHOW,false))
       res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
    ChartSetInteger(0,CHART_SHOW,true);
    return(res);
   }
```

結果:

© 2000-2025, MetaQuotes Ltd.



参照

実数型（double、float）、統計、科学的チャート、クライアント端末プロパティ

© 2000–2025, MetaQuotes Ltd.

MathRand

0～32,767の範囲内で擬似乱数値を返します。

```
int  MathRand();
```

戻り値

0～32,767 の整数値

注意事項

最初に関数を呼び出す前にMathSrandを呼び出して擬似乱数発生器を初期状態に設定する必要があります。

注意事項

MathRand()の代わりにrand()を使用できます。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 新しい初期状態を設定して、起動のたびに一連の擬似ランダム整数を生成する
   MathSrand(GetTickCount());
//--- ループ内で、生成された 10 個の擬似乱数を操作ログに表示する
   for(int i=0; i<10; i++)
     {
      int rand_value=MathRand();
      PrintFormat("Pseudorandom integer №%d: %u",i+1,rand_value);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

MathRound

指定された数値の最も近い整数に丸めた値を返します。

```
double  MathRound(
   double  value      // 丸められる値
   );
```

パラメータ

value

[in] 丸める前の数値

戻り値

最も近い整数に丸めた値

注意事項

MathRound()の代わりにround()を使用できます。

例：

```
#define VALUES_TOTAL 31

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 変換用の変数を宣言する
   double value=0;                     // MathRound変換用の実数
   int    round_value=0;               // ここで結果を取得する
//--- 実数の小数増分の数によるループ
   for(int i=0; i<VALUES_TOTAL; i++)
     {
      //--- 実数値を増やす
      //--- 最も近い整数に四捨五入した数値を取得する
      //--- 操作ログに制御値を表示する
      value+=0.1;
      round_value=(int)MathRound(NormalizeDouble(value,1));
      PrintFormat("value: %.1f, round value: %d",value,round_value);
      /*
     結果:
      value: 0.1, round value: 0
      value: 0.2, round value: 0
      value: 0.3, round value: 0
      value: 0.4, round value: 0
      value: 0.5, round value: 1
      value: 0.6, round value: 1
```

© 2000-2025, MetaQuotes Ltd.

```
      value: 0.7, round value: 1
      value: 0.8, round value: 1
      value: 0.9, round value: 1
      value: 1.0, round value: 1
      value: 1.1, round value: 1
      value: 1.2, round value: 1
      value: 1.3, round value: 1
      value: 1.4, round value: 1
      value: 1.5, round value: 2
      value: 1.6, round value: 2
      value: 1.7, round value: 2
      value: 1.8, round value: 2
      value: 1.9, round value: 2
      value: 2.0, round value: 2
      value: 2.1, round value: 2
      value: 2.2, round value: 2
      value: 2.3, round value: 2
      value: 2.4, round value: 2
      value: 2.5, round value: 3
      value: 2.6, round value: 3
      value: 2.7, round value: 3
      value: 2.8, round value: 3
      value: 2.9, round value: 3
      value: 3.0, round value: 3
      value: 3.1, round value: 3
      */
   }
}
```

© 2000–2025, MetaQuotes Ltd.

MathSin

指定された角度の正弦を返します。

```
double  MathSin(
   double  value      // ラジアンでの引数
   );
```

パラメータ

value

[in] ラジアンで表した角度

戻り値

ラジアン単位で測定された角度の正弦（-1 ～ 1）

注意事項

MathSin() の代わりに sin() を使用できます。

例：

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/100);
   delta[0]=0;
//--- deltaステップで-1～2 piの101個の値を取得する
   vector X=delta.CumSum();
//--- Xベクトルの各値の逆正弦値を計算する
   vector Y=MathSin(X);

//--- 計算された値をベクトルから配列に転送する
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- 計算されたベクトル値のグラフを描画する
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- EscapeキーまたはPgDnキーを押してグラフを削除し（スクリーンショットを撮り）、終了するまで待機する
```

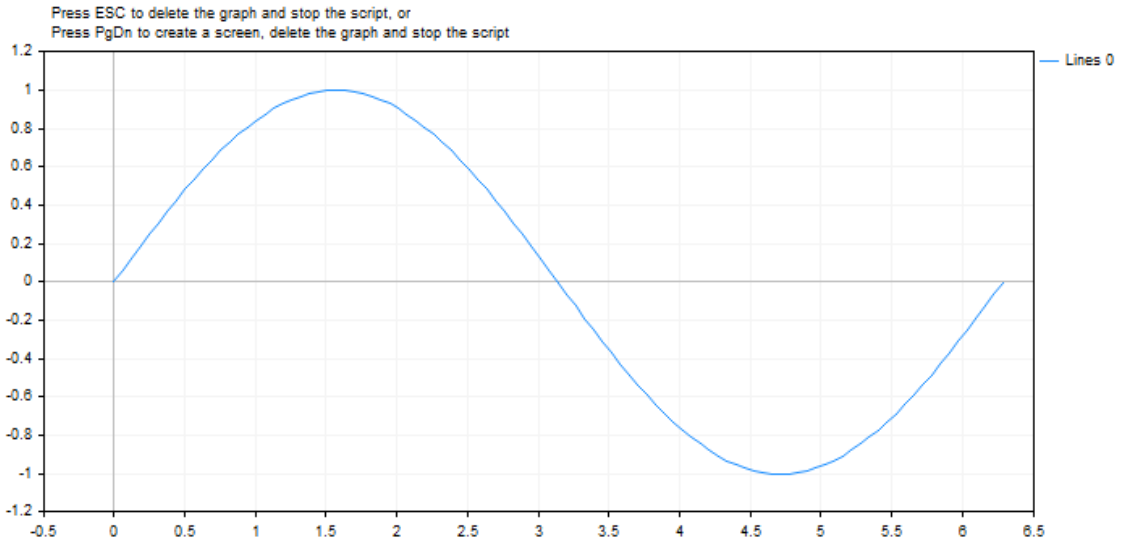
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- クリーンアップ
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//|  ESCが押されたら「true」を返す                                   |
//|  PgDnが押されたら、グラフのスクリーンショットを撮り、「true」を返す |
//|  その他の場合は「false」を返す                                   |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESCが押されたら「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDnが押されてグラフのスクリーンショットが正常に取得されたら、「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 &&
MakeAndSaveScreenshot(MQLInfoString(MQL_PROGRAM_NAME)+"_Screenshot"))
      return(true);
//--- 「false」を返す
   return(false);
  }
//+------------------------------------------------------------------+
//|  グラフオブジェクトを作成して曲線を描く                          |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the
script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//|  スクリーンショットを撮り、画像をファイルに保存する              |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*",
FSD_WRITE_FILE, file_names, file_name+".png");
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d",
__FUNCTION__, GetLastError());
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

結果:



参照

実数型（double、float）、統計、科学的チャート、クライアント端末プロパティ

© 2000–2025, MetaQuotes Ltd.

MathSqrt

数の平方根を返します。

```
double  MathSqrt(
   double  value      // 正の数値
   );
```

パラメータ

value

[in] 正の数値

戻り値

valueの平方根。valueが負の場合はNaN(不定値)。

注意事項

MathSqrt()の代わりにsqrt()を使用できます。

例:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- step 1で0～10の11の値を取得する
   vector X(11,VectorArange);
   Print("vector X = \n",X);
//--- Xベクトルの各値の平方根を計算する
   X=MathSqrt(X);
   Print("MathSqrt(X) = \n",X);

//--- 計算された値をベクトルから配列に転送する
   double y_array[];
   X.Swap(y_array);

//--- 計算されたベクトル値のグラフを描画する
   CurvePlot(y_array,clrDodgerBlue);

//--- Escapeキーまたは PgDnキーを押してグラフを削除し(スクリーンショットを撮り)、終了するまで待機する
   while(!IsStopped())
```

© 2000-2025, MetaQuotes Ltd.

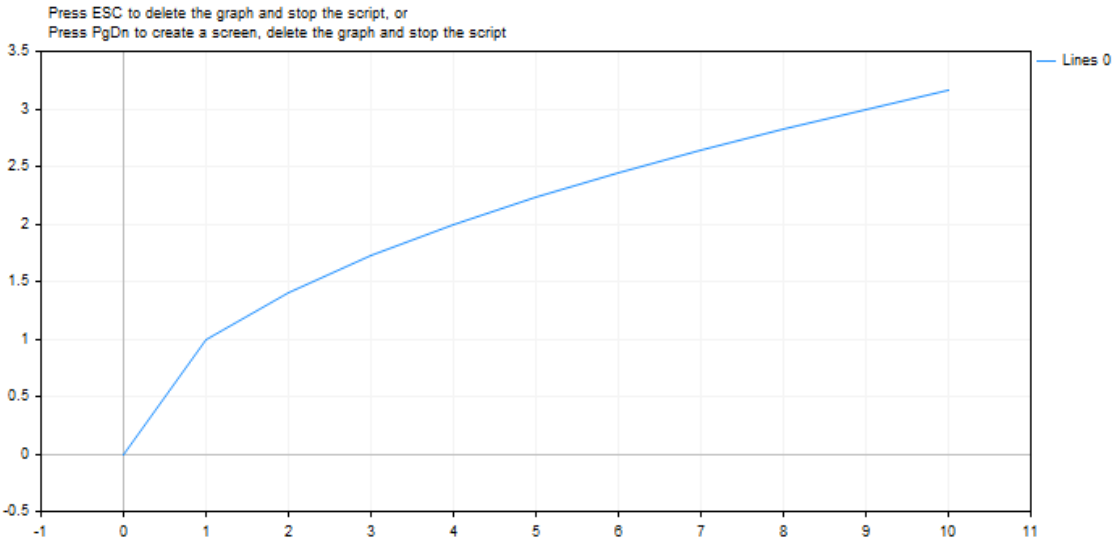
```
      {
       if(StopKeyPressed())
          break;
       Sleep(16);
      }

//--- クリーンアップ
   ExtGraph.Destroy();
   /*
  結果:
   vector X =
   [0,1,2,3,4,5,6,7,8,9,10]
   MathSqrt(X) =
   [
0
,
1
,
1
.414213562373095
,
1
.732050807568877
,
2
,
2
.23606797749979
,2.449489742783178,2.645751311064591,2.82842712474619,3,3.16227766016838]
   */
  }
//+------------------------------------------------------------------+
//| ベクトルに「value」を「step」単位で入力する                                |
//+------------------------------------------------------------------+
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
//+------------------------------------------------------------------+
//| ESCが押されたら「true」を返す                                          |
//| PgDnが押されたら、グラフのスクリーンショットを撮り、「true」を返す                  |
//| その他の場合は「false」を返す                                         |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESCが押されたら「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDnが押されてグラフのスクリーンショットが正常に取得されたら、「true」を返す
```

© 2000-2025, MetaQuotes Ltd.

```
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 &&
MakeAndSaveScreenshot(MQLInfoString(MQL_PROGRAM_NAME)+"_Screenshot"))
      return(true);
//--- 「false」を返す
   return(false);
  }
//+------------------------------------------------------------------+
//|  グラフオブジェクトを作成して曲線を描く                                  |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the
script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//|  スクリーンショットを撮り、画像をファイルに保存する                              |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*",
FSD_WRITE_FILE, file_names, file_name+".png");
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d",
__FUNCTION__, GetLastError());
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

結果：

© 2000–2025, MetaQuotes Ltd.



参照

実数型（double、float）、統計、科学的チャート、クライアント端末プロパティ

© 2000–2025, MetaQuotes Ltd.

MathSrand

擬似ランダム整数の系列を生成するための開始点を設定します。

```
void  MathSrand(
   int  seed      // 数の初期化
   );
```

パラメータ

seed

[in] 乱数列の初値

戻り値

なし

注意事項

MathRand() 関数は、擬似乱数列を生成するために使用されます。特定の初期数を使用してMathSrand() を呼び出すことで、常に同じ擬似乱数列の生成が可能です。

GetTickCount()は、オペレーティングシステムの開始の瞬間から増加し内蔵ミリ秒のカウンタがオーバーフローするまで49日以内に繰り返されないので、非経常な列の受信を確保するにはMathSrand(GetTickCount()) が使用されるべきです。TimeCurrent() 関数は最終ティックの時刻を戻し、週末などには長時間変わらないことがあるので、MathSrand(TimeCurrent()) の使用は不適切です。

指標及びエキスパートアドバイザーのためのMathSrand() を使用した乱数発生器の初期化は、OnTick() 及びOnCalculate() 内での発生器の複数にわたった再起動を避けるためにOnInit() ハンドラ内でよりよく行われます。

MathSrand() 関数の代わりにsrand() が使用可能です。

例:

```
#property description "The indicator shows the central limit theorem, which states:"
#property description "The sum of a sufficiently large number of weakly dependent random variables, "
#property description "having approximately equal magnitude (none of the summands dominates,"
#property description "or makes a determining contribution to the sum), has a distribution close to normal."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- グラフィック構成のプロパティ
#property indicator_label1  "Label"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrRoyalBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  5
//--- 入力変数
input int      sample_number=10;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 分布を描画するための指標バッファ
double          LabelBuffer[];
//--- ティックカウンタ
double          ticks_counter;
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- 配列と指標バッファを関連付ける
   SetIndexBuffer(0,LabelBuffer,INDICATOR_DATA);
//--- 指標バッファを現在から過去に向ける
   ArraySetAsSeries(LabelBuffer,true);
//--- 擬似乱数生成器を初期化する
   MathSrand(GetTickCount());
//--- ティックカウンタを初期化する
   ticks_counter=0;
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- ゼロカウンタで指標バッファをリセットする
   if(ticks_counter==0) ArrayInitialize(LabelBuffer,0);
//--- カウンタを増加
   ticks_counter++;
//--- 定期的に配信を復活させるために、ティックカウンターをリセットする必要がある
   if(ticks_counter>100)
     {
      Print("We've reset the indicator values, let's start filling the cells once
again");
      ticks_counter=0;
     }
//--- 0～7の3つの数の和としてランダムな値のサンプルを取得する
   for(int i=0;i<sample_number;i++)
     {
      //--- 乱数が他の 3 つの数の和であるセルのインデックスの計算
      int rand_index=0;
      //--- 0～7 の 3 つの乱数を取得
```

© 2000-2025, MetaQuotes Ltd.

```
      for(int k=0;k<3;k++)
        {
         //--- 7 による除算で余りは 0～6 の値を返す
         rand_index+=MathRand()%7;
        }
      //--- セル番号 rand_index の値を 1 で増やす
      LabelBuffer[rand_index]++;
     }
//--- OnCalculate() ハンドルを終了する
   return(rates_total);
  }
```

© 2000-2025, MetaQuotes Ltd.

MathTan

この関数は数値の正接を返します。

```
double  MathTan(
   double  rad      // ラジアン単位の引数
   );
```

パラメータ

rad

[in] ラジアンで表した角度

戻り値

rad の正接。radが263以上または-263以下の場合、結果内の重大な損失が起こり、不特定の数が戻されます。

注意事項

MathTan()の代わりにtan()を使用できます。

例：

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/100);
   delta[0]=-263;
//--- deltaステップで-263～2 piの101個の値を取得する
   vector X=delta.CumSum();
//--- X ベクトルの各値の正接値を計算する
   vector Y=MathTan(X);

//--- 計算された値をベクトルから配列に転送する
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- 計算されたベクトル値のグラフを描画する
   CurvePlot(x_array,y_array,clrDodgerBlue);
```

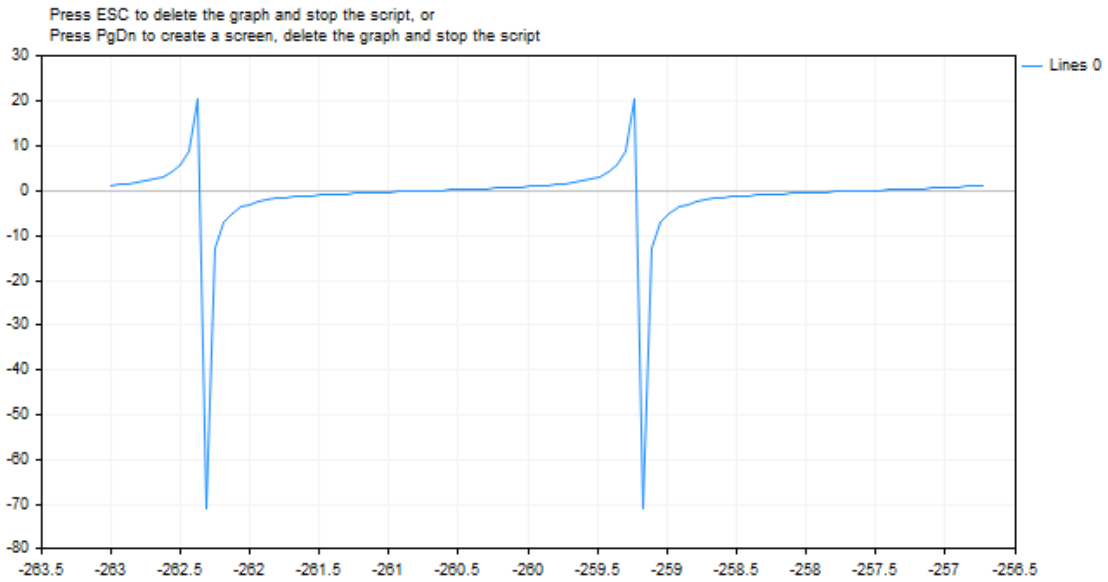
© 2000-2025, MetaQuotes Ltd.

```
//--- EscapeキーまたはPgDnキーを押してグラフを削除し(スクリーンショットを撮り)、終了するまで待機する
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- クリーンアップ
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//|  ESCが押されたら「true」を返す                                   |
//|  PgDnが押されたら、グラフのスクリーンショットを撮り、「true」を返す |
//|  その他の場合は「false」を返す                                   |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESCが押されたら「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDnが押されてグラフのスクリーンショットが正常に取得されたら、「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 &&
MakeAndSaveScreenshot(MQLInfoString(MQL_PROGRAM_NAME)+"_Screenshot"))
      return(true);
//--- 「false」を返す
   return(false);
  }
//+------------------------------------------------------------------+
//|  グラフオブジェクトを作成して曲線を描く                          |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the
script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//|  スクリーンショットを撮り、画像をファイルに保存する              |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*",
FSD_WRITE_FILE, file_names, file_name+".png");
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d",
__FUNCTION__, GetLastError());
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

結果:



参照

実数型(double、float)、統計、科学的チャート、クライアント端末プロパティ

© 2000-2025, MetaQuotes Ltd.

MathIsValidNumber

実数の有効性をチェックします。

```
bool  MathIsValidNumber(
   double  number      // チェックする数
   );
```

パラメータ

number

[in] チェックする数

戻り値

チェックされた値が有効な実数である場合は true、チェックされた値が正または負の無限大、または「非数」(NaN)の場合は false

例:

```
   double abnormal=MathArcsin(2.0);
   if(!MathIsValidNumber(abnormal)) Print("Attention! MathArcsin(2.0) = ",abnormal);
```

参照

浮動小数点数型(ダブル、フロート)

© 2000-2025, MetaQuotes Ltd.

MathExpm1

式 MathExp(x)-1 の値を返します。

```
double  MathExpm1(
   double  value      // eの累乗指数
   );
```

パラメータ

value

[in] 累乗指数を指定する数値

戻り値

double型の値（オーバーフローではINF（無限大）、アンダーフローでは0）

注意事項

xの値が0に近い場合、MathExpm1(x)関数はMathExp(x)-1関数よりもはるかに正確な値を生成します。

MathExpm1()関数の代わりにexpm1()を使用できます。

例：

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- step 1で0～8の9個の値を取得する
   vector X(9,VectorArange);
   Print("vector X = \n",X);
//--- 各Xベクトル値の「e」(オイラー数)を計算する
   X=MathExpm1(X);
   Print("MathExpm1(X) = \n",X);

//--- 計算された値をベクトルから配列に転送する
   double y_array[];
   X.Swap(y_array);

//--- 計算されたベクトル値のグラフを描画する
   CurvePlot(y_array,clrDodgerBlue);
```

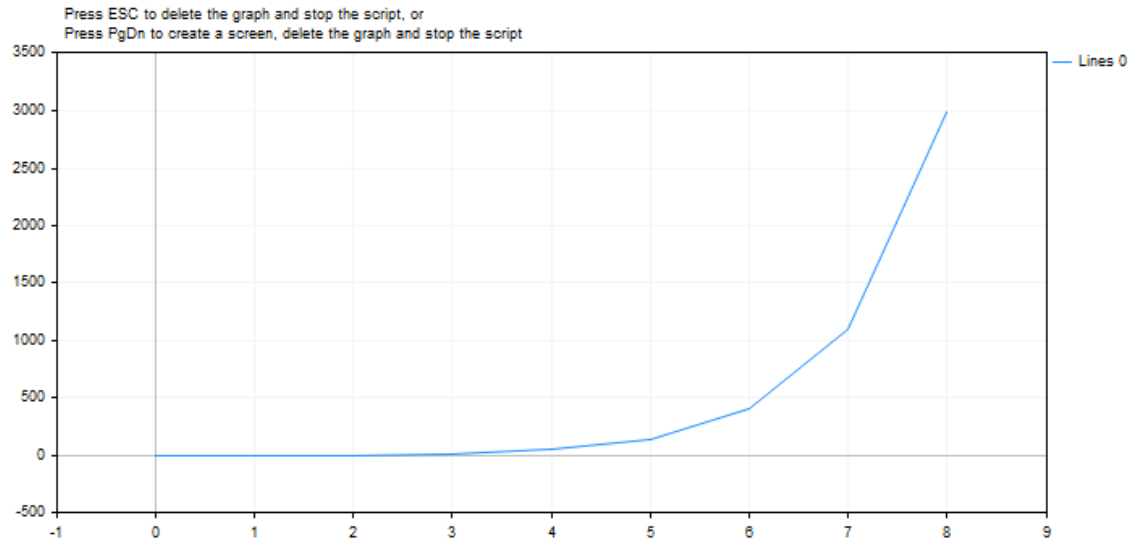
© 2000-2025, MetaQuotes Ltd.

```
//--- EscapeキーまたはPgDnキーを押してグラフを削除し(スクリーンショットを撮り)、終了するまで待機する
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- クリーンアップ
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| ベクトルに「value」を「step」単位で入力する                                |
//+------------------------------------------------------------------+
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
//+------------------------------------------------------------------+
//| ESCが押されたら「true」を返す |
//| ESCが押されたら「true」を返す                                          |
//| PgDnが押されたら、グラフのスクリーンショットを撮り、「true」を返す                  |
//| その他の場合は「false」を返す                                         |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESCが押されたら「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDnが押されてグラフのスクリーンショットが正常に取得されたら、「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 &&
MakeAndSaveScreenshot(MQLInfoString(MQL_PROGRAM_NAME)+"_Screenshot"))
      return(true);
//--- 「false」を返す
   return(false);
  }
//+------------------------------------------------------------------+
//| グラフオブジェクトを作成して曲線を描く                                    |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
```

© 2000-2025, MetaQuotes Ltd.

```
   string text2="Press PgDn to create a screen, delete the graph and stop the
script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//|  スクリーンショットを撮り、画像をファイルに保存する                                  |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*",
FSD_WRITE_FILE, file_names, file_name+".png");
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d",
__FUNCTION__, GetLastError());
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

結果:



© 2000–2025, MetaQuotes Ltd.

参照

実数型（double、float）、統計、科学的チャート、クライアント端末プロパティ

© 2000–2025, MetaQuotes Ltd.

MathLog1p

式MathLog(1+x)の値を返します。

```
double  MathLog1p(
   double  value      // 対数を取る値
   );
```

パラメータ

value

[in] 対数を取る値

戻り値

成功した場合は、valueの自然対数(value + 1)。valueが-1より小さい場合はNaN(未定義の値)を返します。valueが-1に等しい場合はINF(無限大)を返します。

注意事項

xの値が0に近い場合、MathLog1p(x)関数はMathLog(1+x)関数よりもはるかに正確な値を生成します。

MathLog1p()関数の代わりに、log1p()関数を使用できます。

例：

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- step 1で0～8の9個の値を取得する
   vector X(9,VectorArange);
   Print("vector X = \n",X);
//--- 各ベクトル値について、式の対数を計算する(1 + Xベクトル値)
   X=MathLog1p(X);
   Print("MathLog1p(X) = \n",X);

//--- 計算された値をベクトルから配列に転送する
   double y_array[];
   X.Swap(y_array);

//--- 計算されたベクトル値のグラフを描画する
   CurvePlot(y_array,clrDodgerBlue);
```

© 2000-2025, MetaQuotes Ltd.

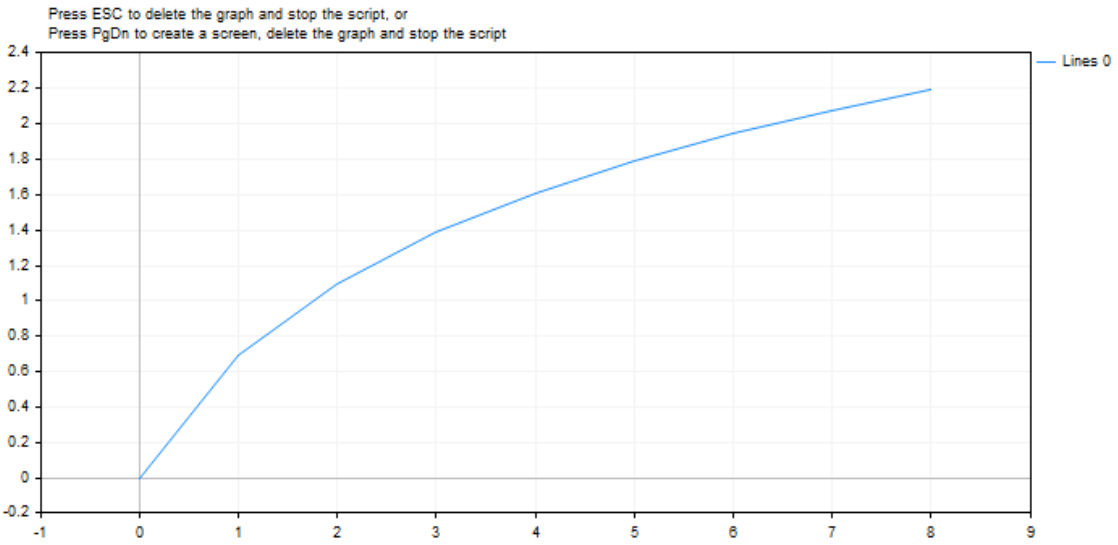
```
//--- EscapeキーまたはPgDnキーを押してグラフを削除し(スクリーンショットを撮り)、終了するまで待機する
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- クリーンアップ
   ExtGraph.Destroy();
   /*
  結果:
   vector X =
   [0,1,2,3,4,5,6,7,8]
   MathLog1p(X) =
   [
0
,
0
.6931471805599453
,
1
.09861228866811
,
1
.386294361119891
,
1
.6094379124341
,1.791759469228055,1.945910149055313,2.079441541679836,2.19722457733622]
   */
  }
//+------------------------------------------------------------------+
//| ベクトルに「value」を「step」単位で入力する                                |
//+------------------------------------------------------------------+
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
//+------------------------------------------------------------------+
//| ESCが押されたら「true」を返す                                          |
//| PgDnが押されたら、グラフのスクリーンショットを撮り、「true」を返す                  |
//| その他の場合は「false」を返す                                         |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESCが押されたら「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
```

© 2000-2025, MetaQuotes Ltd.

```
      return(true);
//--- PgDnが押されてグラフのスクリーンショットが正常に取得されたら、「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 &&
MakeAndSaveScreenshot(MQLInfoString(MQL_PROGRAM_NAME)+"_Screenshot"))
      return(true);
//--- 「false」を返す
   return(false);
  }
//+------------------------------------------------------------------+
//|  グラフオブジェクトを作成して曲線を描く                                  |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the
script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//|  スクリーンショットを撮り、画像をファイルに保存する                          |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*",
FSD_WRITE_FILE, file_names, file_name+".png");
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d",
__FUNCTION__, GetLastError());
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

© 2000-2025, MetaQuotes Ltd.

結果:



参照

実数型（double、float）、統計、科学的チャート、クライアント端末プロパティ

© 2000–2025, MetaQuotes Ltd.

MathArccosh

双曲線逆余弦を返します。

```
double  MathArccosh(
   double  value     // 1 <= value < ∞
   );
```

パラメータ

value

[in] 双曲線逆余弦を計算する値

戻り値

数値の双曲線逆余弦（valueが+1未満の場合、NaN（未定義の値））

注意事項

MathArccosh()関数の代わりにacosh()を使用できます。

例：

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/10);
   delta[0]=1;
//--- deltaステップで-1～2 piの101個の値を取得する
   vector X=delta.CumSum();
//--- Xベクトルの各値の双曲線逆余弦値を計算する
   vector Y=MathArccosh(X);

//--- 計算された値をベクトルから配列に転送する
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- 計算されたベクトル値のグラフを描画する
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- EscapeキーまたはPgDnキーを押してグラフを削除し（スクリーンショットを撮り）、終了するまで待機する
```

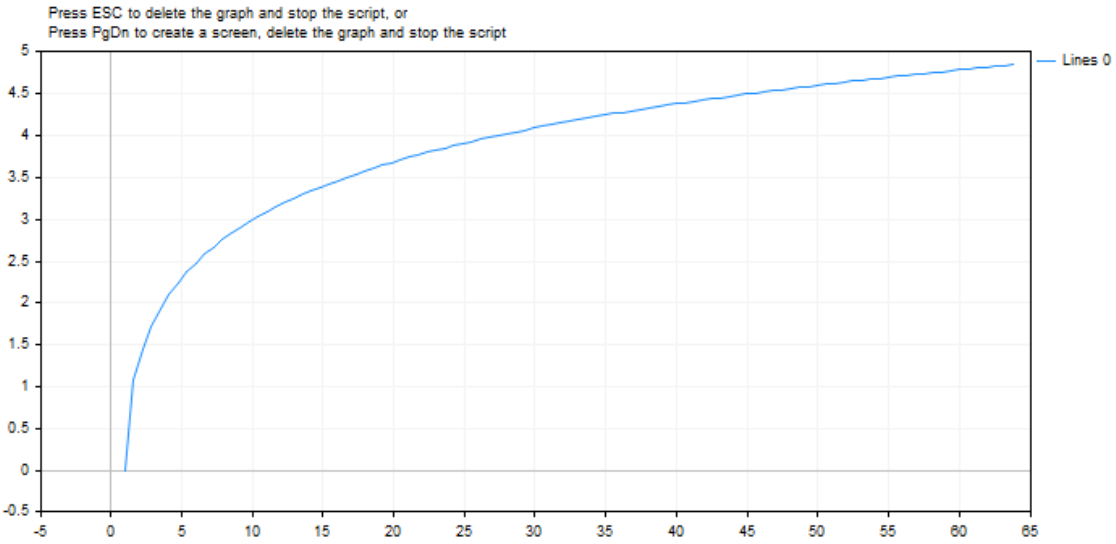
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- クリーンアップ
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//|  ESCが押されたら「true」を返す                                   |
//|  PgDnが押されたら、グラフのスクリーンショットを撮り、「true」を返す |
//|  その他の場合は「false」を返す                                   |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESCが押されたら「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDnが押されてグラフのスクリーンショットが正常に取得されたら、「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 &&
MakeAndSaveScreenshot(MQLInfoString(MQL_PROGRAM_NAME)+"_Screenshot"))
      return(true);
//--- 「false」を返す
   return(false);
  }
//+------------------------------------------------------------------+
//|  グラフオブジェクトを作成して曲線を描く                          |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the
script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//|  スクリーンショットを撮り、画像をファイルに保存する              |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
```

© 2000–2025, MetaQuotes Ltd.

```
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*",
FSD_WRITE_FILE, file_names, file_name+".png");
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d",
__FUNCTION__, GetLastError());
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

結果:



参照

実数型（double、float）、統計、科学的チャート、クライアント端末プロパティ

© 2000–2025, MetaQuotes Ltd.

MathArcsinh

双曲線逆正弦を返します。

```
double  MathArcsinh(
   double  value      // -∞ < value < +∞
   );
```

パラメータ

val

[in] 双曲線逆正弦を計算する値

戻り値

<t0>数の双曲線逆正弦

注意事項

MathArcsinh()関数の代わりにasinh()を使用できます。

例：

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/10);
   delta[0]=-31;
//--- deltaステップで-31～2 piの101個の値を取得する
   vector X=delta.CumSum();
//--- Xベクトルの各値の双曲線逆正弦値を計算する
   vector Y=MathArcsinh(X);

//--- 計算された値をベクトルから配列に転送する
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- 計算されたベクトル値のグラフを描画する
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- EscapeキーまたはPgDnキーを押してグラフを削除し（スクリーンショットを撮り）、終了するまで待機する
```

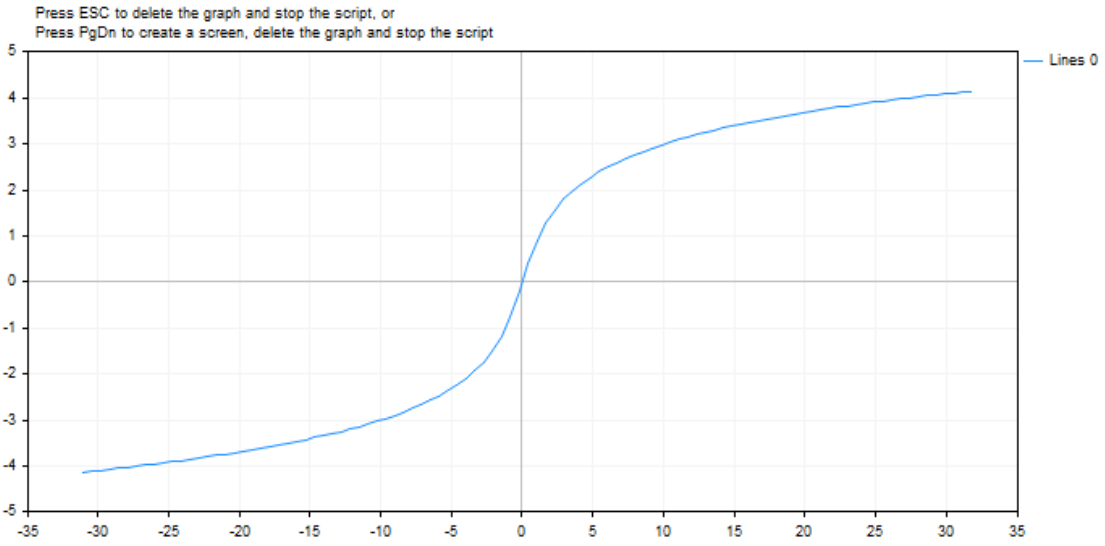
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- クリーンアップ
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//|  ESCが押されたら「true」を返す                                   |
//|  PgDnが押されたら、グラフのスクリーンショットを撮り、「true」を返す |
//|  その他の場合は「false」を返す                                   |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESCが押されたら「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDnが押されてグラフのスクリーンショットが正常に取得されたら、「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 &&
MakeAndSaveScreenshot(MQLInfoString(MQL_PROGRAM_NAME)+"_Screenshot"))
      return(true);
//--- 「false」を返す
   return(false);
  }
//+------------------------------------------------------------------+
//|  グラフオブジェクトを作成して曲線を描く                          |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the
script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//|  スクリーンショットを撮り、画像をファイルに保存する              |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
```

© 2000–2025, MetaQuotes Ltd.

```
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*",
FSD_WRITE_FILE, file_names, file_name+".png");
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d",
__FUNCTION__, GetLastError());
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

結果:



参照

実数型(double、float)、統計、科学的チャート、クライアント端末プロパティ

© 2000-2025, MetaQuotes Ltd.

MathArctanh

双曲線逆正接を返します。

```
double  MathArctanh(
   double  value      // -1 < value < 1
   );
```

パラメータ

value

[in] 正接を表す、-1 < 値 < 1 の範囲内の数値

戻り値

数値の逆正弦

注意事項

MathArctanh()関数の代わりにatanh()が使用可能です。

例：

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(34,2*M_PI/100);
   delta[0]=-1;
//--- deltaステップで -1 から 2 pi までの 34 個の値を取得する
   vector X=delta.CumSum();
//--- X ベクトルの各値の双曲線逆正弦値を計算する
   vector Y=MathArctanh(X);

//--- 計算された値をベクトルから配列に転送する
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- 計算されたベクトル値のグラフを描画する
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- EscapeキーまたはPgDnキーを押してグラフを削除し（スクリーンショットを撮り）、終了するまで待機する
```

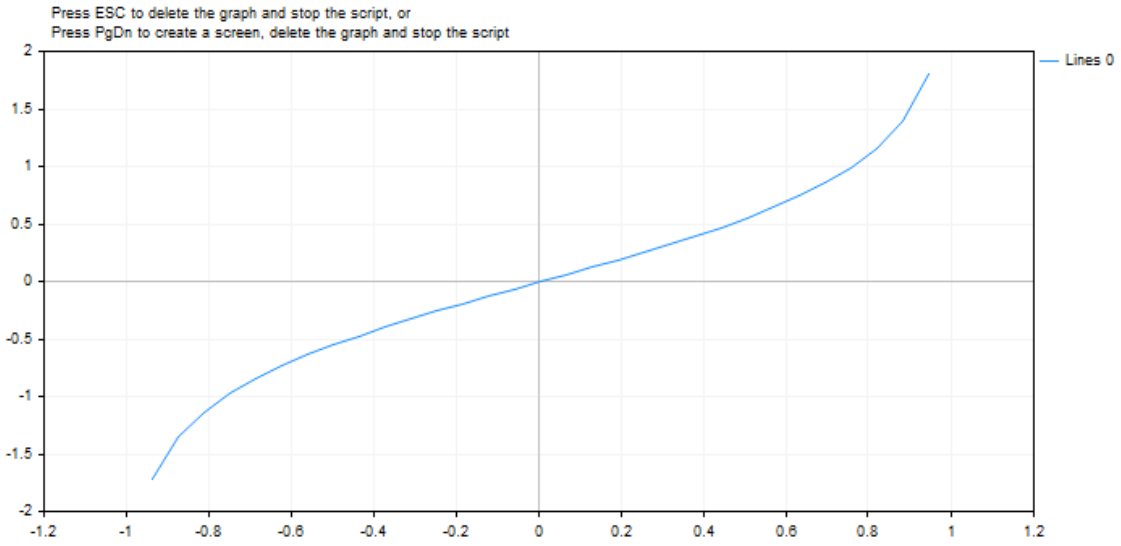
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- クリーンアップ
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//|  ESCが押されたら「true」を返す                                   |
//|  PgDnが押されたら、グラフのスクリーンショットを撮り、「true」を返す |
//|  その他の場合は「false」を返す                                   |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESCが押されたら「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDnが押されてグラフのスクリーンショットが正常に取得されたら、「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 &&
MakeAndSaveScreenshot(MQLInfoString(MQL_PROGRAM_NAME)+"_Screenshot"))
      return(true);
//--- 「false」を返す
   return(false);
  }
//+------------------------------------------------------------------+
//|  グラフオブジェクトを作成して曲線を描く                          |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the
script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//|  スクリーンショットを撮り、画像をファイルに保存する              |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
```

© 2000–2025, MetaQuotes Ltd.

```
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*",
FSD_WRITE_FILE, file_names, file_name+".png");
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d",
__FUNCTION__, GetLastError());
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

結果:



参照

統計、科学チャート、クライアント端末プロパティ

© 2000–2025, MetaQuotes Ltd.

MathCosh

数値の双曲線余弦を返します。

```
double  MathCosh(
   double  value      // 数値
   );
```

パラメータ

value

[in] 値

戻り値

数値の双曲線余弦（+1 から正の無限大までの範囲内の値）

注意事項

MathCosh()関数の代わりにcosh()を使用できます。

例：

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/100);
   delta[0]=0;
//--- deltaステップで-1～2 piの101個の値を取得する
   vector X=delta.CumSum();
//--- Xベクトルの各値の双曲線余弦値を計算する
   vector Y=MathCosh(X);

//--- 計算された値をベクトルから配列に転送する
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- 計算されたベクトル値のグラフを描画する
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- EscapeキーまたはPgDnキーを押してグラフを削除し（スクリーンショットを撮り）、終了するまで待機する
```

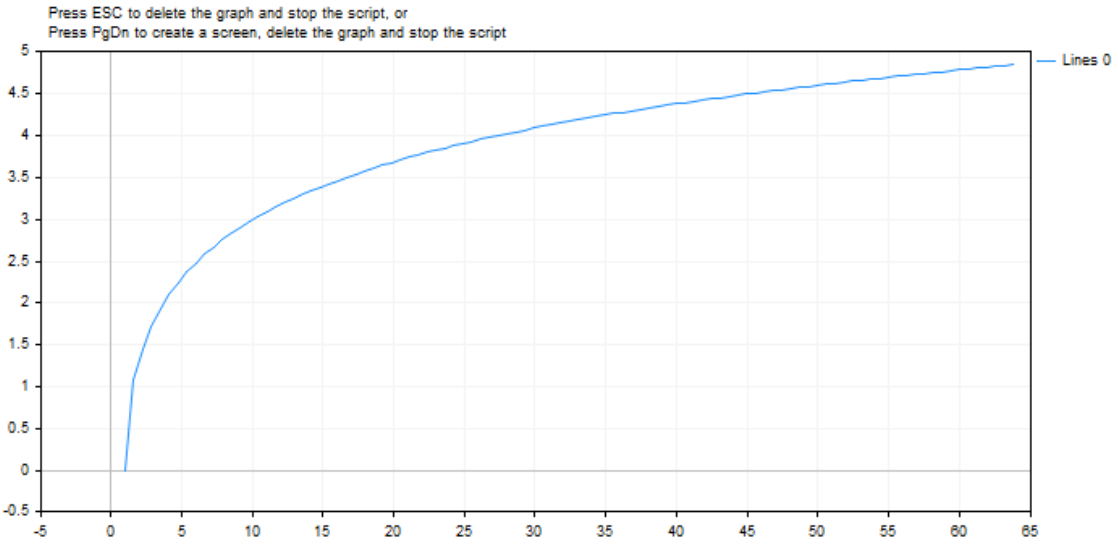
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- クリーンアップ
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//|  ESCが押されたら「true」を返す                                   |
//|  PgDnが押されたら、グラフのスクリーンショットを撮り、「true」を返す |
//|  その他の場合は「false」を返す                                   |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESCが押されたら「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDnが押されてグラフのスクリーンショットが正常に取得されたら、「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 &&
MakeAndSaveScreenshot(MQLInfoString(MQL_PROGRAM_NAME)+"_Screenshot"))
      return(true);
//--- 「false」を返す
   return(false);
  }
//+------------------------------------------------------------------+
//|  グラフオブジェクトを作成して曲線を描く                          |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the
script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//|  スクリーンショットを撮り、画像をファイルに保存する              |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
```

© 2000–2025, MetaQuotes Ltd.

```
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*",
FSD_WRITE_FILE, file_names, file_name+".png");
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d",
__FUNCTION__, GetLastError());
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

結果:



参照

統計、科学チャート、クライアント端末プロパティ

© 2000–2025, MetaQuotes Ltd.

MathSinh

数値の双曲線正弦を返します。

```
double  MathSinh(
   double  value      // 数値
   );
```

パラメータ

value

[in] 値

戻り値

数値の双曲線正弦

注意事項

MathSinh()関数の代わりにsinh()を使用できます。

例：

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/100);
   delta[0]=0;
//--- deltaステップで-1～2 piの101個の値を取得する
   vector X=delta.CumSum();
//--- Xベクトルの各値の双曲線正弦値を計算する
   vector Y=MathSinh(X);

//--- 計算された値をベクトルから配列に転送する
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- 計算されたベクトル値のグラフを描画する
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- Escapeキーまたは PgDn キーを押してグラフを削除し（スクリーンショットを撮り）、終了するまで待機する
```

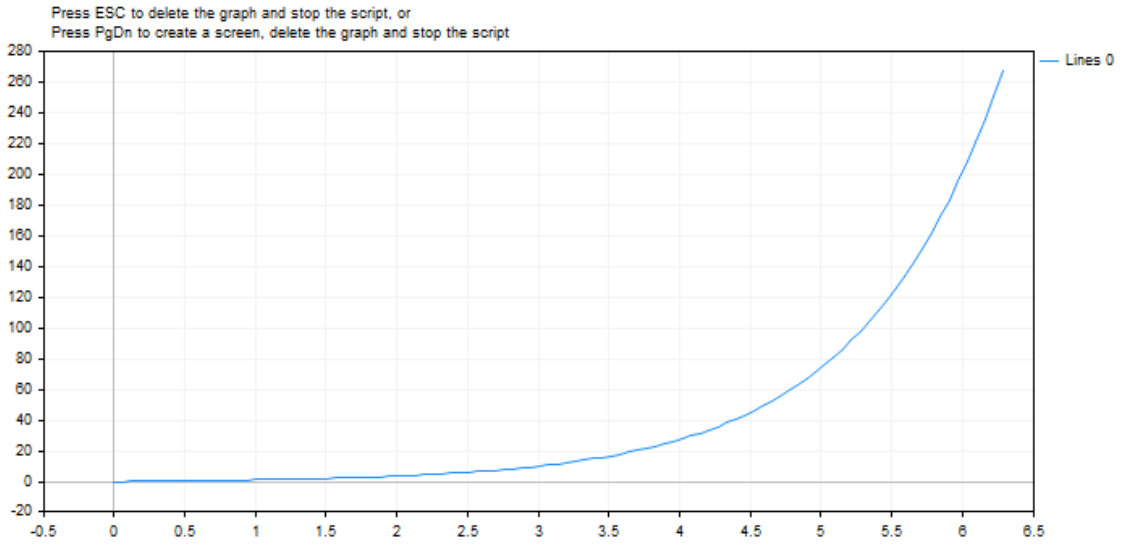
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- クリーンアップ
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//|  ESCが押されたら「true」を返す                                   |
//|  PgDnが押されたら、グラフのスクリーンショットを撮り、「true」を返す |
//|  その他の場合は「false」を返す                                   |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESCが押されたら「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDnが押されてグラフのスクリーンショットが正常に取得されたら、「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 &&
MakeAndSaveScreenshot(MQLInfoString(MQL_PROGRAM_NAME)+"_Screenshot"))
      return(true);
//--- 「false」を返す
   return(false);
  }
//+------------------------------------------------------------------+
//|  グラフオブジェクトを作成して曲線を描く                          |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the
script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//|  スクリーンショットを撮り、画像をファイルに保存する              |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*",
FSD_WRITE_FILE, file_names, file_name+".png");
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d",
__FUNCTION__, GetLastError());
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

結果:



参照

統計、科学チャート、クライアント端末プロパティ

© 2000–2025, MetaQuotes Ltd.

MathTanh

数値の双曲線正弦を返します。

```
double  MathTanh(
   double  value      // 数値
   );
```

パラメータ

value

[in] 値

戻り値

数値の双曲線正弦（+1～+1の範囲内の値）

注意事項

MathTanh()関数の代わりに tanh()を使用できます。

例：

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/100);
   delta[0]=0;
//--- deltaステップで-1～2 piの101個の値を取得する
   vector X=delta.CumSum();
//--- calculate the hyperbolic tangent value for each value of the X vector
   vector Y=MathTanh(X);

//--- 計算された値をベクトルから配列に転送する
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- 計算されたベクトル値のグラフを描画する
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- EscapeキーまたはPgDnキーを押してグラフを削除し（スクリーンショットを撮り）、終了するまで待機する
```

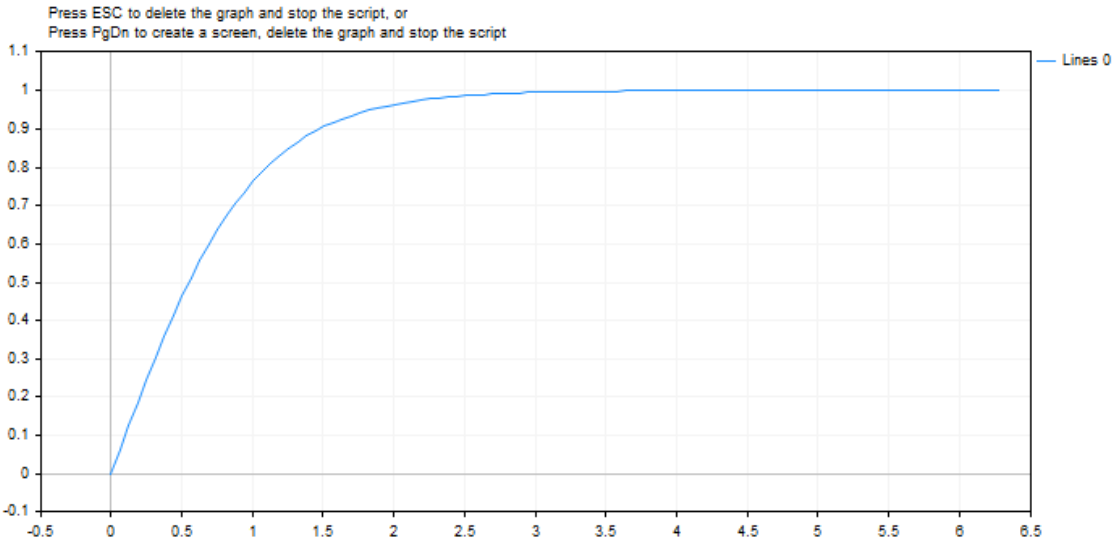
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- クリーンアップ
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//|  ESCが押されたら「true」を返す                                   |
//|  PgDnが押されたら、グラフのスクリーンショットを撮り、「true」を返す |
//|  その他の場合は「false」を返す                                   |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESCが押されたら「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDnが押されてグラフのスクリーンショットが正常に取得されたら、「true」を返す
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 &&
MakeAndSaveScreenshot(MQLInfoString(MQL_PROGRAM_NAME)+"_Screenshot"))
      return(true);
//--- 「false」を返す
   return(false);
  }
//+------------------------------------------------------------------+
//|  グラフオブジェクトを作成して曲線を描く                          |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the
script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//|  スクリーンショットを撮り、画像をファイルに保存する              |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
```

© 2000–2025, MetaQuotes Ltd.

```
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*",
FSD_WRITE_FILE, file_names, file_name+".png");
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d",
__FUNCTION__, GetLastError());
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

結果:



参照

実数型（double、float）、統計、科学的チャート、クライアント端末プロパティ

© 2000–2025, MetaQuotes Ltd.

MathSwap

ushort型 値 のバイト 順を変 更します。

```
ushort  MathSwap(
   ushort  value      // 値
   );
```

パラメータ

value

[in] バイト 順を変 更する値

戻り値

バイト 順 が反 転されたushort値

MathSwap

uint型 値 のバイト 順を変 更します。

```
uint  MathSwap(
   uint   value      // 値
   );
```

パラメータ

value

[in] バイト 順を変 更する値

戻り値

バイト 順 が反 転されたuint値

MathSwap

ulong型 値 のバイト 順を変 更します。

```
ulong  MathSwap(
   ulong  value      // 値
   );
```

パラメータ

value

[in] バイト 順を変 更する値

戻り値

バイト 順 が反 転されたulong値

例：

© 2000–2025, MetaQuotes Ltd.

```
#property script_show_inputs

input ulong  InpLongValue  =  1;     // ここに任意のulong値を入力
input uint   InpIntValue   =  2;     // ここに任意のuint値を入力
input ushort InpShortValue =  3;     // ここに任意のushort値を入力

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {

//--- 入力され、MathSwap()によって変換された値を10進数および2進数表現で操作ログに出力する
   Print(ValueDescription(InpLongValue));
   Print(ValueDescription(InpIntValue));
   Print(ValueDescription(InpShortValue));
   /*
  結果:
   ulong value: 1
   ulong value: 72057594037927936 using MathSwap()
   binary ulong value:
0000000000000000000000000000000000000000000000000000000000000001
   binary ulong value:
0000000100000000000000000000000000000000000000000000000000000000 using MathSwap()

   uint value: 2
   uint value: 33554432 using MathSwap()
   binary uint value: 00000000000000000000000000000010
   binary uint value: 00000010000000000000000000000000 using MathSwap()

   ushort value: 3
   ushort value: 768 using MathSwap()
   binary ushort value: 0000000000000011
   binary ushort value: 0000001100000000 using MathSwap()
   */
  }
//+------------------------------------------------------------------+
//|  変数値を説明するテキストを返す                                  |
//+------------------------------------------------------------------+
template <typename T>
string ValueDescription(T x)
  {
   int    num_bits    = sizeof(T)*8;
   string type_name   = typename(T);
   string bin_x       = NumberToBinaryString(x);
   string bin_swap_x = NumberToBinaryString(MathSwap(x));
   return(StringFormat("%s value: %lld\n%s value: %lld using MathSwap()\nbinary %s
value: %0*s\nbinary %s value: %0*s using MathSwap()\n",
```

© 2000-2025, MetaQuotes Ltd.

```
                        type_name, x, type_name, MathSwap(x), type_name, num_bits,
bin_x, type_name, num_bits, bin_swap_x));
  }
//+------------------------------------------------------------------+
//|  数値の2進数表現を文字列として返す                                |
//+------------------------------------------------------------------+
template <typename T>
string NumberToBinaryString(T x)
  {
   string res  = "";
   int    i    = -1;
   uchar  size = sizeof(T)*8-1;
   ulong  mask = (ulong)1<<size;
   while(!((x<<++i)& mask));
   for(; i <=size; i++)
      res += !((x<<i)& mask) ? "0" : "1";
   return res;
  }
```

参照

ネットワーク関数、SocketRead、SocketSend、SocketTlsRead、SocketTlsReadAvailable、SocketTlsSend

© 2000-2025, MetaQuotes Ltd.

文字列関数

これは、string 型のデータを操作するための関数群です。

関数	アクション
StringAdd	文字列の終わりに別の文字列を追加します。
StringBufferLen	文字列に割り当てられたバッファのサイズを返します。
StringCompare	2 つの文字列を比較し、1 番目の文字列が2 番目より大きい場合は 1 、両方が等しい場合は 0、1 番目が2 番目より小さい場合は -1 を返します。
StringConcatenate	渡されたパラメータの文字列を形成します。
StringFill	指定された文字列に選択されたシンボルを書き入れます。
StringFind	文字列内で部分文字列を検索します。
StringGetCharacter	指定された文字列の位置での数の値を返します。
StringInit	指定された記号で文字列を初期化し、指定された文字列の長さを提供します。
StringLen	文字列のシンボルの数を返します。
StringSetLength	文字列に指定された長さ(文字数)を設定します
StringReplace	文字列で見つかった全ての部分文字列を指定されたシンボルの配列で置換します。
StringReserve	指定されたサイズのバッファをメモリ内の文字列用に予約します
StringSetCharacter	指定された位置にシンボルの変更値を持つ文字列のコピーを返します。
StringSplit	指定された文字列から指定されたセパレータで部分文字列を取得し、取得された部分文字列の数を返します。
StringSubstr	指定された位置から開始してテキスト文字列から部分文字列を抽出します。
StringToLower	選択された文字列の全てのシンボルを小文字に変換します。
StringToUpper	選択された文字列の全てのシンボルを大文字に変換します。
StringTrimLeft	文字列の左部分の改行文字、スペースやタブを切り取ります。
StringTrimRight	文字列の右部分の改行文字、スペースやタブを切り取ります。

© 2000-2025, MetaQuotes Ltd.

StringAdd

この関数は文字列の終わりに別の文字列を追加します。

```
bool  StringAdd(
   string&  string_var,          // 文字列を追加する文字列
   string   add_substring        // 追加される文字列
   );
```

パラメータ

string_var

[in][out] 文字列を追加する文字列

add_substring

[in] 元の文字列に追加される文字列

戻り値

成功の場合は true、それ以外の場合は false。エラーコードを受け取るには GetLastError() 関数が呼ばれるべきです。

例:

```
void OnStart()
  {
   long length=1000000;
   string a="a",b="b",c;
//--- 1 番目の方法
   uint start=GetTickCount(),stop;
   long i;
   for(i=0;i<length;i++)
     {
      c=a+b;
     }
   stop=GetTickCount();
   Print("time for 'c = a + b' = ",(stop-start)," milliseconds, i = ",i);

//--- 2 番目の方法
   start=GetTickCount();
   for(i=0;i<length;i++)
     {
      StringAdd(a,b);
     }
   stop=GetTickCount();
   Print("time for 'StringAdd(a,b)' = ",(stop-start)," milliseconds, i = ",i);

//--- 3 番目の方法
   start=GetTickCount();
   a="a"; // 変数 a を再初期化する
   for(i=0;i<length;i++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       StringConcatenate(c,a,b);
      }
   stop=GetTickCount();
   Print("time for 'StringConcatenate(c,a,b)' = ",(stop-start)," milliseconds, i =
",i);
  }
```

参照

StringConcatenate, StringSplit, StringSubstr

© 2000–2025, MetaQuotes Ltd.

StringBufferLen

この関数は文字列に割り当てられたバッファのサイズを返します。

```
int  StringBufferLen(
   string  string_var      // 文字列
  )
```

パラメータ

string_var

[in] 文字列

戻り値

0 は、文字列が定数でバッファサイズが変更出来ないことを意味します。-1 は、文字列がクライアント端末に属し、バッファ内容の修正が不確定な結果を有することが出来ることを意味します。

例:

```
void OnStart()
  {
   long length=1000;
   string a="a",b="b";
//---
   long i;
   Print("before: StringBufferLen(a) = ",StringBufferLen(a),
         "  StringLen(a) = ",StringLen(a));
   for(i=0;i<length;i++)
     {
      StringAdd(a,b);
     }
   Print("after: StringBufferLen(a) = ",StringBufferLen(a),
         "  StringLen(a) = ",StringLen(a));
  }
```

参照

StringAdd、StringInit、StringLen、StringFill

© 2000-2025, MetaQuotes Ltd.

StringCompare

この関数は 2 つの文字列を比較し、比較結果を整数値で返します。

```
int  StringCompare(
   const string&  string1,                  // 比較される1 番目の文字列
   const string&  string2,                  // 比較される2 番目の文字列
   bool           case_sensitive=true       // 比較時の大文字と小文字の区別モードの選択
   );
```

パラメータ

string1

[in] 1 番目の文字列

string2

[in] 2 番目の文字列

case_sensitive=true

[in] 大文字と小文字の区別モードの選択。true の場合 "A" > "a"。false の場合 "A" = "a"。初期値は true です。

戻り値

- string1 < string2 の場合は -1
- string1 = string2 の場合は 0
- string1 > string2 の場合は 1

注意事項

文字列はシンボル毎に比較され、シンボルは、現在のコードページに基づいてアルファベット順に比較されます。

例:

```
void OnStart()
  {
//--- apple と home を比較する
   string s1="Apple";
   string s2="home";

//--- 大文字と小文字を区別して比較する
   int result1=StringCompare(s1,s2);
   if(result1>0) PrintFormat("Case sensitive comparison: %s > %s",s1,s2);
   else
     {
      if(result1<0)PrintFormat("Case sensitive comparison: %s < %s",s1,s2);
      else PrintFormat("Case sensitive comparison: %s = %s",s1,s2);
     }

//--- 大文字と小文字を区別しないで比較する
   int result2=StringCompare(s1,s2,false);
   if(result2>0) PrintFormat("Case insensitive comparison: %s > %s",s1,s2);
   else
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       if(result2<0)PrintFormat("Case insensitive comparison: %s < %s",s1,s2);
       else PrintFormat("Case insensitive comparison: %s = %s",s1,s2);
      }
/* 結果:
      Case-sensitive comparison: Apple < home
      Case insensitive comparison: Apple < home
*/
   }
```

参照

String 型、CharToString()、ShortToString()、StringToCharArray()、StringToShortArray()、StringGetCharacter()、コードページの利用

© 2000–2025, MetaQuotes Ltd.

StringConcatenate

この関数は渡されたパラメータの文字列を形成しそのサイズを返します。パラメータは任意の型です。パラメータの数は 2 以上で 64を超えることは出来ません。

```
int  StringConcatenate(
   string&  string_var,    // 形成する文字列
   void argument1          // 任意の基本データ型を持つ1 番目のパラメータ
   void argument2          // 任意の基本データ型を持つ2 番目のパラメータe
   ...                     // 任意の基本データ型を持つ次のパラメータ
   );
```

パラメータ

string_var

[out] 連結の結果として形成される文字列

argumentN

[in] コンマで区切られた任意の値。(任意の基本データ型で 2 から 63)

戻り値

連結の結果として形成された文字列の長さパラメータは Print() 及び Comment() と同じ規則に従って文字列に変換されます。

例：

```
void OnStart()
  {
//--- 連結に関与する変数を宣言および定義する
   string text="";
   string text1="This script shows how the StringConcatenate() function works.\n";
   string text2="This is the second line, at the end of which there is a line break control code.\n";
   string text3="This is line number ";
   int    num3=3;
   string text31=", the number of which is entered into the function as a separate parameter.";
   string textN="\n";
   string text4="This is line number 4, preceded by a separate parameter with a line break code.";
   int    length=StringConcatenate(text, text1, text2, text3, num3, text31, textN, text4, "\nLine 5 includes a real number: ", 0.12345);
   Print(text, "\nLength of the resulting string = ", length);

  /*
  結果
   This script shows how the StringConcatenate() function works.
   This is the second line, at the end of which there is a line break control code.
   This is line number 3, the number of which is entered into the function as a separate parameter.
   This is line number 4, preceded by a separate parameter with a line break code.
```

© 2000-2025, MetaQuotes Ltd.

```
   Line 5 includes a real number: 0.12345
   Length of the resulting string = 358
  */
  }
```

参照

StringAdd、StringSplit、StringSubstr

© 2000–2025, MetaQuotes Ltd.

StringFill

指定された文字列に選択されたシンボルを書き入れます。

```
bool  StringFill(
   string&   string_var,        // 書き込みされる文字列
   ushort    character          // 書き込みに使用されるシンボル
   );
```

パラメータ

string_var

[in][out] 選択されたシンボルを書き入れられる文字列

character

[in] 文字列の書き込みに使用されるシンボル

戻り値

成功の場合は true、それ以外の場合は false。エラーコードを取得するには、GetLastError()が呼ばれます。

注意事項

「in place」での文字列の記入は、新しい文字列の作成またはコピーの移行操作なしで、シンボルが文字列に直接挿入されていることを意味します。これは操作時間を節約します。

例:

```
void OnStart()
  {
   string str;
   StringInit(str,20,'_');
   Print("str = ",str);
   StringFill(str,0);
   Print("str = ",str,": StringBufferLen(str) = ", StringBufferLen(str));
  }
// 結果
//   str = ____________________
//   str =  : StringBufferLen(str) = 20
//
```

参照

StringBufferLen、StringLen、StringInit

© 2000-2025, MetaQuotes Ltd.

StringFind

文字列内で部分文字列を検索します。

```
int  StringFind(
   string  string_value,         // 検索が行われる文字列
   string  match_substring,      // 検索される文字列
   int     start_pos=0           // 検索開始位置
   );
```

パラメータ

string_value

[in] 検索が行われる文字列

match_substring

[in] 検索される部分文字列

start_pos=0

[in] 検索が開始される文字列内の位置

戻り値

検索された部分文字列が始まる文字列の位置。部分文字列が見つからない場合は -1 。

例：

```
#define   RESERVE    100

//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- シンボルの基本通貨と利益通貨を取得する
   string symbol_currency_base  =SymbolInfoString(Symbol(), SYMBOL_CURRENCY_BASE);
   string symbol_currency_profit=SymbolInfoString(Symbol(), SYMBOL_CURRENCY_PROFIT);
   PrintFormat("Symbol Currency Base: %s\nSymbol Currency Profit: %s",
symbol_currency_base, symbol_currency_profit);

//--- サーバ上で利用可能なすべてのシンボルをループ内で処理する
   int total=SymbolsTotal(false), pos=-1;
   for(int i=0; i<total; i++)
     {
      //--- 次のシンボルの名前を取得する
      string name=SymbolName(i, false);

      //--- シンボル名の中で基本通貨の名前を含む部分文字列を探し、
      //--- 部分文字列が見つかった場合は、シンボル名、通貨リスト内のそのインデックス、およびログ内の検索された通
貨の名前を表示する
      pos = StringFind(name, symbol_currency_base);
      if(pos >= 0)
```

© 2000-2025, MetaQuotes Ltd.

```
         PrintFormat("The '%s' symbol at index %u in the list contains the '%s'
currency. Substring position in the symbol name: %d", name, i, symbol_currency_base,
pos);

      //--- シンボル名の中で相場通貨の名前を持つ部分文字列を検索し、
      //--- 部分文字列が見つかった場合は、シンボル名、通貨リスト内のそのインデックス、およびログ内の検索された通
貨の名前を表示する
      pos = StringFind(name, symbol_currency_profit);
      if(pos >= 0)
         PrintFormat("The '%s' symbol at index %u in the list contains the '%s'
currency. Substring position in the symbol name: %d", name, i,
symbol_currency_profit, pos);
     }

   /*
  結果
   StringFind (EURUSD,D1)   Symbol Currency Base: EUR
   StringFind (EURUSD,D1)   Symbol Currency Profit: USD
   The 'EURUSD' symbol at index 0 in the list contains the 'EUR' currency. Substring
 position in the symbol name: 0
   The 'EURUSD' symbol at index 0 in the list contains the 'USD' currency. Substring
 position in the symbol name: 3
   The 'GBPUSD' symbol at index 1 in the list contains the 'USD' currency. Substring
 position in the symbol name: 3
   The 'USDCHF' symbol at index 2 in the list contains the 'USD' currency. Substring
 position in the symbol name: 0
   The 'USDJPY' symbol at index 3 in the list contains the 'USD' currency. Substring
 position in the symbol name: 0
   The 'USDCNH' symbol at index 4 in the list contains the 'USD' currency. Substring
 position in the symbol name: 0
   The 'USDRUB' symbol at index 5 in the list contains the 'USD' currency. Substring
 position in the symbol name: 0
   The 'AUDUSD' symbol at index 6 in the list contains the 'USD' currency. Substring
 position in the symbol name: 3
   The 'NZDUSD' symbol at index 7 in the list contains the 'USD' currency. Substring
 position in the symbol name: 3
   The 'USDCAD' symbol at index 8 in the list contains the 'USD' currency. Substring
 position in the symbol name: 0
   The 'USDSEK' symbol at index 9 in the list contains the 'USD' currency. Substring
 position in the symbol name: 0
   The 'USDHKD' symbol at index 10 in the list contains the 'USD' currency.
Substring position in the symbol name: 0
   The 'USDSGD' symbol at index 11 in the list contains the 'USD' currency.
Substring position in the symbol name: 0
   The 'USDNOK' symbol at index 12 in the list contains the 'USD' currency.
Substring position in the symbol name: 0
   The 'USDDKK' symbol at index 13 in the list contains the 'USD' currency.
Substring position in the symbol name: 0
   The 'USDTRY' symbol at index 14 in the list contains the 'USD' currency.
Substring position in the symbol name: 0
   The 'USDZAR' symbol at index 15 in the list contains the 'USD' currency.
Substring position in the symbol name: 0
```

© 2000–2025, MetaQuotes Ltd.

```
   The 'USDCZK' symbol at index 16 in the list contains the 'USD' currency.
Substring position in the symbol name: 0
   The 'USDHUF' symbol at index 17 in the list contains the 'USD' currency.
Substring position in the symbol name: 0
   The 'USDPLN' symbol at index 18 in the list contains the 'USD' currency.
Substring position in the symbol name: 0
   The 'USDRUR' symbol at index 19 in the list contains the 'USD' currency.
Substring position in the symbol name: 0
   The 'EURAUD' symbol at index 27 in the list contains the 'EUR' currency.
Substring position in the symbol name: 0
   The 'EURCAD' symbol at index 28 in the list contains the 'EUR' currency.
Substring position in the symbol name: 0
   The 'EURCHF' symbol at index 29 in the list contains the 'EUR' currency.
Substring position in the symbol name: 0
   The 'EURCZK' symbol at index 30 in the list contains the 'EUR' currency.
Substring position in the symbol name: 0
   The 'EURDKK' symbol at index 31 in the list contains the 'EUR' currency.
Substring position in the symbol name: 0
   The 'EURGBP' symbol at index 32 in the list contains the 'EUR' currency.
Substring position in the symbol name: 0
   The 'EURHKD' symbol at index 33 in the list contains the 'EUR' currency.
Substring position in the symbol name: 0
   The 'EURHUF' symbol at index 34 in the list contains the 'EUR' currency.
Substring position in the symbol name: 0
   The 'EURJPY' symbol at index 35 in the list contains the 'EUR' currency.
Substring position in the symbol name: 0
   The 'EURNOK' symbol at index 36 in the list contains the 'EUR' currency.
Substring position in the symbol name: 0
   The 'EURNZD' symbol at index 37 in the list contains the 'EUR' currency.
Substring position in the symbol name: 0
   The 'EURPLN' symbol at index 38 in the list contains the 'EUR' currency.
Substring position in the symbol name: 0
   The 'EURRUR' symbol at index 39 in the list contains the 'EUR' currency.
Substring position in the symbol name: 0
   The 'EURRUB' symbol at index 40 in the list contains the 'EUR' currency.
Substring position in the symbol name: 0
   The 'EURSEK' symbol at index 41 in the list contains the 'EUR' currency.
Substring position in the symbol name: 0
   The 'EURTRY' symbol at index 42 in the list contains the 'EUR' currency.
Substring position in the symbol name: 0
   The 'EURZAR' symbol at index 43 in the list contains the 'EUR' currency.
Substring position in the symbol name: 0
   The 'XAUUSD' symbol at index 47 in the list contains the 'USD' currency.
Substring position in the symbol name: 3
   The 'XAUEUR' symbol at index 48 in the list contains the 'EUR' currency.
Substring position in the symbol name: 3
   The 'XAGUSD' symbol at index 50 in the list contains the 'USD' currency.
Substring position in the symbol name: 3
   The 'XAGEUR' symbol at index 51 in the list contains the 'EUR' currency.
Substring position in the symbol name: 3
   The 'USDCRE' symbol at index 53 in the list contains the 'USD' currency.
Substring position in the symbol name: 0
   The 'XPDUSD' symbol at index 65 in the list contains the 'USD' currency.
Substring position in the symbol name: 3
```

© 2000–2025, MetaQuotes Ltd.

```
   The 'XPTUSD' symbol at index 66 in the list contains the 'USD' currency.
Substring position in the symbol name: 3
   The 'USDGEL' symbol at index 67 in the list contains the 'USD' currency.
Substring position in the symbol name: 0
   The 'USDMXN' symbol at index 68 in the list contains the 'USD' currency.
Substring position in the symbol name: 0
   The 'EURMXN' symbol at index 69 in the list contains the 'EUR' currency.
Substring position in the symbol name: 0
   The 'USDCOP' symbol at index 75 in the list contains the 'USD' currency.
Substring position in the symbol name: 0
   The 'USDARS' symbol at index 76 in the list contains the 'USD' currency.
Substring position in the symbol name: 0
   The 'USDCLP' symbol at index 77 in the list contains the 'USD' currency.
Substring position in the symbol name: 0
   The 'EURSGD' symbol at index 89 in the list contains the 'EUR' currency.
Substring position in the symbol name: 0
   The 'USDILS' symbol at index 95 in the list contains the 'USD' currency.
Substring position in the symbol name: 0
   The 'USDTHB' symbol at index 122 in the list contains the 'USD' currency.
Substring position in the symbol name: 0
   The 'USDRMB' symbol at index 123 in the list contains the 'USD' currency.
Substring position in the symbol name: 0
   The 'EURILS' symbol at index 126 in the list contains the 'EUR' currency.
Substring position in the symbol name: 0
   The 'EURCNH' symbol at index 137 in the list contains the 'EUR' currency.
Substring position in the symbol name: 0
   The 'USDBRL' symbol at index 139 in the list contains the 'USD' currency.
Substring position in the symbol name: 0
   */
  }
```

参照

StringSubstr, StringGetCharacter, StringLen, StringLen

© 2000–2025, MetaQuotes Ltd.

StringGetCharacter

文字列の指定された位置にあるシンボルの値を返します。

```
ushort  StringGetCharacter(
   string  string_value,      // 文字列
   int     pos                // 文字列内でのシンボルの位置
   );
```

パラメータ

string_value

[in] 文字列

pos

[in] 文字列内でのシンボルの位置（ 0～StringLen(text) -1 ）

戻り値

シンボルコード。エラーの場合は 0 。エラーコード を取得するには、GetLastError()が呼ばれます。

例：

```
void OnStart()
  {
//--- チャート上のすべてのコメントを削除する
   Comment("");
//--- 文字列を宣言し、そこからシンボルコードの値を取得して文字列の長さを記憶する
   string message = "The script demonstrates the operation of the
StringGetCharacter() function";
   int    length  = StringLen(message);
//--- デモ文字列から取得したシンボルを追加する文字列変数を宣言する
   string text    = "";
//--- デモ文字列の長さによるループ
   for(int i=0; i<length; i++)
     {
      //--- 1/10 秒待つ
      Sleep(100);
      //--- デモ文字列のループ インデックスにある文字列からシンボルを取得する
      ushort char_code=StringGetCharacter(message, i);
      //--- 表示された文字列に記号を追加し、結果の文字列をチャートのコメントとして表示する
      text+=ShortToString(char_code);
      Comment(text);
     }
//--- 2秒待って、チャートからコメントを削除する
   Sleep(2000);
   Comment("");

   /*
  結果：デモ文字列が 1 文字ずつ画面に表示される
   The script demonstrates the operation of the StringGetCharacter() function
```

© 2000-2025, MetaQuotes Ltd.

```
   */
  }
```

参照

StringSetCharacter、StringBufferLen、StringLen、<li6 >StringFill、StringInit、StringToCharArray、StringToShortArray

© 2000–2025, MetaQuotes Ltd.

StringInit

指定された記号で文字列を初期化し、指定された文字列の長さを提供します。

```
bool  StringInit(
   string&    string_var,        // 初期化される文字列
   int        new_len=0,         // 初期化後の必要な文字列の長さ
   ushort     character=0        // 初期化に使用されるシンボル
   );
```

パラメータ

string_var

[in][out] 初期化及び初期化解除される文字列

new_len=0

[in] 初期化後の文字列の長さlength=0 の場合、文字列は初期化解除されます。つまり、文字列バッファはクリアされバッファアドレスがゼロに設定されます。

character=0

[in] 文字列に書き込むシンボル

戻り値

成功の場合は true、それ以外の場合は false。エラーコード を取得するには、GetLastError()が呼ばれます。

注意事項

character=0 及び 新しい長さを示すnew_len > 0 の場合、指定された長さの文字列のバッファが配布され、ゼロで満たされます。バッファ全体に文字列ターミネータが記入されているため、文字列の長さはゼロに等しくなります。

例:

```
void OnStart()
  {
//---
   string str;
   StringInit(str,200,0);
   Print("str = ",str,": StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
  }
/*  結果
str = : StringBufferLen(str) = 200   StringLen(str) = 0
*/
```

参照

StringBufferLen、StringLen

© 2000-2025, MetaQuotes Ltd.

StringLen

文字列のシンボルの数を返します。

```
int  StringLen(
   string  string_value      // 文字列
   );
```

パラメータ

string_value

[in] 長さを計算する文字列

戻り値

文字列での最後のゼロを除いたシンボル数

例：

```
void OnStart()
  {
//--- テスト文字列を定義する
   string text="123456789012345";
//--- 文字列内のシンボルの数を取得する
   int str_len=StringLen(text);
//--- 文字列とその中のシンボルの数をログに表示する
   PrintFormat("The StringLen() function returned the value of %d chars in string: '%s'", str_len, text);

  /*
  結果
   The StringLen() function returned the value of 15 chars in string: '123456789012345'
  */
  }
```

参照

StringBufferLen、StringTrimLeft、StringTrimRight、<li6 >StringToCharArray、StringToShortArray

© 2000–2025, MetaQuotes Ltd.

StringSetLength

文字列に指定された長さ(文字数)を設定します。

```
bool  StringSetLength(
   string&     string_var,       // 文字列
   uint        new_length        // 新しい文字列の長さ
   );
```

パラメータ

string_var

[in][out] 新しい文字数を設定する文字列

new_capacity

[in] 文字数での必要な文字列の長さ。new_lengthが現在のサイズより小さい場合、余分な文字は破棄されます。

戻り値

実行成功の場合はtrue、それ以外の場合はfalseエラーコードを受け取るには、GetLastError()関数が呼び出されるべきです。

注意事項

StringSetLength()関数は、文字列に割り当てられたバッファのサイズを変更しません。

例：

```
void OnStart()
  {
//--- 文字列を定義する
   string text="123456789012345";

//--- 文字列とその長さをログに表示する
   PrintFormat("Before StringSetLength() the string '%s' has a size of %d characters", text, StringLen(text));

//--- 文字列サイズを10文字に減らす
   StringSetLength(text, 10);

//--- StringSetLength()操作によって変更された文字列とその新しい長さをログに表示する
   PrintFormat("After StringSetLength() the string is now '%s', and has a size of %d characters", text, StringLen(text));

  /*
  結果
   Before StringSetLength() the string '123456789012345' has a size of 15 characters
   After StringSetLength() the string is now '1234567890', and has a size of 10 characters
  */
  }
```

© 2000-2025, MetaQuotes Ltd.

参照

StringLen、StringBufferLen、StringReserve、<li6 >StringInit、StringSetCharacter

© 2000–2025, MetaQuotes Ltd.

StringReplace

文字列で見つかった全ての部分文字列を指定されたシンボルの配列で置換します。

```
int  StringReplace(
   string&          str,              // 部分文字列が書き込まれる文字列
   const string     find,             // 検索された部分文字列
   const string     replacement       // 見つかった位置に挿入される部分文字列
   );
```

パラメータ

str

[in][out] 中で部分文字列が置き換えられる文字列

find

[in] 置き換えられる部分文字列

replacement

[in] 見つかった部分文字列の代わりに挿入される文字列

戻り値

成功の場合は置き換えの回数、それ以外の場合は -1。エラーコードの取得には GetLastError() 関数を使用します。

注意事項

関数が正常に実行されたが、(置き換えられる部分文字列が見つからなかったため) 置換が行われていない場合は、0 が返されます。

不正な *str* または *find* パラメータ(空または初期化されていない文字列。 StringInit() をご参照ください) によってエラーが発生することがあります。また、置き換えに充分なメモリがない場合にもエラーが発生します。

例:

```
  string text="The quick brown fox jumped over the lazy dog.";
  int replaced=StringReplace(text,"quick","slow");
  replaced+=StringReplace(text,"brown","black");
  replaced+=StringReplace(text,"fox","bear");
  Print("Replaced: ", replaced,". Result=",text);

//  結果
//  Replaced: 3. Result=The slow black bear jumped over the lazy dog.
//
```

参照

StringSetCharacter()、StringSubstr()

© 2000-2025, MetaQuotes Ltd.

StringReserve

指定されたサイズのバッファをメモリ内の文字列用に予約します。

```
bool  StringReserve(
   string&     string_var,        // 文字列
   uint        new_capacity       // 文字列を格納するバッファのサイズ
   );
```

パラメータ

string_var

[in][out] バッファサイズが合わせてサイズを変更する文字列

new_capacity

[in] 文字列に必要なバッファサイズ。new_capacityのサイズが文字列の長さより小さい場合、現在のバッファのサイズは変わりません。

戻り値

実行成功の場合はtrue、それ以外の場合はfalseエラーコードを受け取るには、GetLastError()関数が呼び出されるべきです。

注意事項

一般的に、文字列サイズは文字列を格納するためのバッファのサイズと等しくありません。文字列を作成するとき、適切なバッファは通常マージンを持って割り当てられます。StringReserve()関数を使用すると、バッファサイズを管理し、将来の操作に最適なサイズを指定できます。

StringInit()と異なり、StringReserve()関数は文字列の内容を変更せず、文字で埋めません。

例:

```
void OnStart()
  {
   string s;
//--- StringReserveを使わずに動作速度を確認する
   ulong t0=GetMicrosecondCount();
   for(int i=0; i< 1024; i++)
      s+=" "+(string)i;
   ulong msc_no_reserve=GetMicrosecondCount()-t0;
   s=NULL;
//--- StringReserveを使用して同じことをする
   StringReserve(s,1024 * 3);
   t0=GetMicrosecondCount();
   for(int i=0; i< 1024; i++)
      s+=" "+(string)i;
   ulong msc_reserve=GetMicrosecondCount()-t0;
//--- 時刻を確認する
   Print("Test with StringReserve passed for "+(string)msc_reserve+" msc");
   Print("Test without StringReserve passed for "+(string)msc_no_reserve+" msc");

/* 結果:
```

© 2000-2025, MetaQuotes Ltd.

```
      Test with StringReserve passed for 50 msc
      Test without StringReserve passed for 121 msc
*/
  }
```

参照

StringBufferLen、StringSetLength、StringInit、StringSetCharacter

© 2000–2025, MetaQuotes Ltd.

StringSetCharacter

指定された位置での文字が変更された文字列のコピーを返します。

```
bool  StringSetCharacter(
   string&   string_var,        // 文字列
   int       pos,               // 位置
   ushort    character          // 文字
   );
```

パラメータ

string_var

[in][out] 文字列

pos

[in] 文字列内での文字の位置0からStringLen(text)

character

[in] Unicodeシンボルコード

戻り値

成功の場合はtrue、それ以外の場合はfalse。エラーコードを受け取るにはGetLastError()関数が呼ばれるべきです。

注意事項

posがStringLen文字列の長さより少なくシンボルコードの値が0の場合、文字列は短く切られます(が文字列に割り当てられたバッファサイズは変わりません。)。文字列の長さはposに等しくなります。

*pos*が文字列の長さと等しい場合、指定されたシンボルが文字列の終わりに追加され、長さが1つ長くなります。

例:

```
void OnStart()
  {
   string str="0123456789";
   Print("before: str = ",str,",StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
//--- 真ん中にゼロ値を追加する
   StringSetCharacter(str,6,0);
   Print("after: str = ",str,",StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
//--- 終わりにシンボルを追加する
   int size=StringLen(str);
   StringSetCharacter(str,size,'+');
   Print("addition: str = ",str,",StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
  }
/* 結果
   before: str = 0123456789 ,StringBufferLen(str) = 0    StringLen(str) = 10
```

© 2000-2025, MetaQuotes Ltd.

```
    after: str = 012345 ,StringBufferLen(str) = 16   StringLen(str) = 6
   addition: str = 012345+ ,StringBufferLen(str) = 16   StringLen(str) = 7
*/
```

参照

StringBufferLen, StringLen, StringFill, StringInit, CharToString, ShortToString, CharArrayToString, ShortArrayToString

© 2000–2025, MetaQuotes Ltd.

StringSplit

指定された文字列から指定されたセパレータで部分文字列を取得し、取得された部分文字列の数を返します。.

```
int  StringSplit(
   const string   string_value,        // 検索の行われる文字列
   const ushort   separator,           // 区切り文字
   string         & result[]           // 見つかった部分文字列を受け取るための参照で渡された配列
   );
```

パラメータ

string_value

[in] 部分文字列が取得される文字列。文字列は変更されません。

pos

[in] 区切り文字のコード。コードの取得には StringGetCharacter() 関数が使用されます。

result[]

[out] 取得された部分文字列が含まれている文字列配列

戻り値

result[] 配列での部分文字列の数セパレータが渡された文字列内に見つからない場合、元の文字列のみが配列内に配置されます。

string_value が空または NULL の場合は 0エラーの場合は -1エラーコードの取得には GetLastError() 関数が呼ばれます。

例:

```
string to_split="_life_is_good_"; // 部分文字列に分ける文字列
   string sep="_";                // 区切り文字
   ushort u_sep;                  // 区切り文字のコード
   string result[];               // 文字列を受け取る配列
   //--- 区切り文字のコードを取得する
   u_sep=StringGetCharacter(sep,0);
   //--- 文字列を部分文字列に分ける
   int k=StringSplit(to_split,u_sep,result);
   //--- コメントを表示する
   PrintFormat("Strings obtained: %d. Used separator '%s' with the code %
d",k,sep,u_sep);
   //--- 取得された文字列を全て出力する
   if(k>0)
     {
      for(int i=0;i<k;i++)
        {
         PrintFormat("result[%d]=\"%s\"",i,result[i]);
        }
     }
```

参照

© 2000-2025, MetaQuotes Ltd.

StringReplace()、StringSubstr()、StringConcatenate()

© 2000–2025, MetaQuotes Ltd.

StringSubstr

指定された位置から開始してテキスト文字列から部分文字列を抽出します。

```
string  StringSubstr(
   string  string_value,      // 文字列
   int     start_pos,         // 開始位置
   int     length=-1          // 抽出された文字列の長さ
   );
```

パラメータ

string_value

[in] 部分文字列を抽出する文字列

start_pos

[in] 部分文字列の初期の位置（0～StringLen(text) -1）

length=-1

[in] 抽出された文字列の長さ。このパラメータ値が設定されていないか -1 に等しい場合、部分文字列は示された位置から文字列の最後までから抽出されます。

戻り値

可能な場合は抽出された部分文字列のコピー、その他の場合は空の文字列

例：

```
void OnStart()
  {
//--- 現在のシンボルの名前を取得する
   string name = Symbol();

//--- 基本通貨とクウォートされたシンボル通貨を取得する
   string base   = StringSubstr(name, 0, 3);
   string quoted = StringSubstr(name, 3, 3);

//--- 取得したシンボル通貨をログに表示する
   PrintFormat("Symbol: %s. Currency base: %s, currency quoted: %s", name, base,
quoted);

  /*
  結果
   Symbol: EURUSD. Currency base: EUR, currency quoted: USD
  */
  }
```

参照

StringSplit、StringFind、StringGetCharacter

© 2000-2025, MetaQuotes Ltd.

StringToLower

選択された文字列の全てのシンボルを小文字に変換します。

```
bool  StringToLower(
   string&  string_var      // 処理される文字列
   );
```

パラメータ

string_var

[in][out] 文字列

戻り値

成功の場合は true、それ以外の場合は false。エラーコードを取得するには、GetLastError()が呼ばれます。

例：

```
void OnStart()
  {
//--- ソース文字列を大文字で定義する
   string text=" - THIS STRING, WRITTEN IN UPPERCASE, MUST BE WRITTEN IN LOWERCASE";
//--- ソース文字列をログで表示する
   Print("Source line:\n", text);
//--- すべての文字列文字を小文字に変換し、結果をログに表示する
   if(StringToLower(text))
      Print("The original string after using the StringToLower() function:\n",
text);

  /*
  結果
   Source line:
    - THIS STRING, WRITTEN IN UPPERCASE, MUST BE WRITTEN IN LOWERCASE
   The original string after using the StringToLower() function:
    - this string, written in uppercase, must be written in lowercase
  */
  }
```

参照

StringToUpper、StringTrimLeft、StringTrimRight

© 2000-2025, MetaQuotes Ltd.

StringToUpper

選択された文字列の全てのシンボルを大文字に変換します。

```
bool  StringToUpper(
   string&  string_var      // 処理される文字列
   );
```

パラメータ

string_var

[in][out] 文字列

戻り値

成功の場合は true、それ以外の場合は false。エラーコードを取得するには、GetLastError()が呼ばれます。

例：

```
void OnStart()
  {
//--- ソース文字列を小文字で定義する
   string text=" - this string, written in lowercase, must be written in uppercase";
//--- ソース文字列をログで表示する
   Print("Source line:\n", text);
//--- すべての文字列文字を大文字に変換し、結果をログに表示する
   if(StringToUpper(text))
      Print("The original string after using the StringToUpper() function:\n",
text);

  /*
  結果
   Source line:
    - this string, written in lowercase, must be written in uppercase
   The original string after using the StringToUpper() function:
    - THIS STRING, WRITTEN IN LOWERCASE, MUST BE WRITTEN IN UPPERCASE
  */
  }
```

参照

StringToLower、StringTrimLeft、StringTrimRight

© 2000-2025, MetaQuotes Ltd.

StringTrimLeft

この関数は、文字列の最初の意味のあるシンボルの左部分にある改行文字、スペースやタブを切り取ります。文字列は直接（コピーなどされずに）変更されます。

```
int  StringTrimLeft(
   string&  string_var      // 切られる文字列
   );
```

パラメータ

string_var

[in][out] 左から切られる文字列

戻り値

切り取られたシンボルの数

例：

```
void OnStart()
  {
//--- 左側に6つのスペースを含むソース文字列を定義する
   string text="      All spaces on the left will be removed from this string";
//--- ソース文字列をログで表示する
   PrintFormat("Source line:\n'%s'", text);
//--- 左側のスペースをすべて削除し、削除された文字数と結果の文字列をログに表示する
   int num=StringTrimLeft(text);
   PrintFormat("The StringTrimLeft() function removed %d chars from the left side. Now the line looks like this:\n'%s'", num, text);

  /*
  結果
   Source line:
   '      All spaces on the left will be removed from this string'
   The StringTrimLeft() function removed 6 chars from the left side. Now the line looks like this:
   'All spaces on the left will be removed from this string'
  */
  }
```

参照

StringTrimRight、StringToLower、StringToUpper

© 2000–2025, MetaQuotes Ltd.

StringTrimRight

この関数は、文字列の最後の意味あるシンボルの右部分にある改行文字、スペースやタブを切り取ります。文字列は直接（コピーなどされずに）変更されます。

```
int  StringTrimRight(
   string&  string_var      // 切られる文字列
   );
```

パラメータ

string_var

[in][out] 右から切られる文字列

戻り値

切り取られたシンボルの数

例：

```
void OnStart()
  {
//--- 右側に6つのスペースを含むソース文字列を定義する
   string text="All spaces on the right will be removed from this string      ";
//--- ソース文字列をログで表示する
   PrintFormat("Source line:\n'%s'", text);
//--- 右側のスペースをすべて削除し、削除された文字数と結果の文字列をログに表示する
   int num=StringTrimRight(text);
   PrintFormat("The StringTrimRight() function removed %d chars from the right side. Now the line looks like this:\n'%s'", num, text);

  /*
  結果
   Source line:
   'All spaces on the right will be removed from this string      '
   The StringTrimRight() function removed 6 chars from the right side. Now the line looks like this:
   'All spaces on the right will be removed from this string'
  */
  }
```

参照

StringTrimLeft、StringToLower、StringToUpper

© 2000–2025, MetaQuotes Ltd.

日付と時刻

これは datetime（1970年1月1日の0時間からの経過秒数を表す整数）のデータを扱う関数群です。

高解像度のカウンターとタイマーを配置するには、ミリ秒単位で値を生成する GetTickCount() 関数を使用します。

関数	アクション
TimeCurrent	最後の既知のサーバ時間（相場受信の最終時刻）を datetime フォーマットで返します。
TimeTradeServer	取引サーバの現在の計算時間を返します。
TimeLocal	ローカルコンピューター時刻を datetime フォーマットで返します。
TimeGMT	クライアント端末が実行しているコンピュータのローカル時間をGMTを夏時間を考慮した datetime フォーマットで返します。
TimeDaylightSavings	夏時間のスイッチの符号を返します。
TimeGMTOffset	DST スイッチを考慮して、GMT時間とローカルコンピュータの現在時刻との差を秒で返します。
TimeToStruct	datetime 値を MqlDateTime 構造体型の変数に変換します。
StructToTime	MqlDateTime 構造体型の変数を datetime 値に変換します。

© 2000–2025, MetaQuotes Ltd.

TimeCurrent

最後の既知のサーバ時間、つまり、「気配値表示」ウィンドウで選択したシンボルのうちの 1 つの相場受信の最終時刻を返します。OnTick() ハンドラでは、この関数は取り扱かったティックを受信した時間を返します。他の場合（例えば ハンドラ OnInit()、OnDeinit()、OnTimer() などでの呼び出し）、これはウィンドウのタイトルともなっている、「気配値表示」ウィンドウでのシンボルの相場の最終受信の時刻です。この時刻値は、取引サーバ上で形成され、お使いのコンピュータの時刻の設定に依存しません。この関数には 2 つのバージョンがあります。

パラメータなしでの呼び出し

```
datetime  TimeCurrent();
```

MqlDateTime 型のパラメータとの呼び出し

```
datetime  TimeCurrent(
   MqlDateTime&  dt_struct      // 構造体型の変数
   );
```

パラメータ

dt_struct

[out] MqlDateTime 構造体型の変数

戻り値

datetime 型の値

注意事項

MqlDateTime 構造型の変数がパラメータとして渡された場合は、それに応じて満たされます。

高解像度のカウンターとタイマーを配置するには、ミリ秒単位で値を生成する GetTickCount() 関数を使用します。

ストラテジーテスターのテスト中には TimeCurrent() は過去のデータに応じてシミュレートされます。

例：

```
void OnStart()
  {
//--- 日付/時刻データを格納するMql DateTime変数を宣言し、気配値ウィンドウから最後の相場の時刻を取得する
   MqlDateTime tm={};
   datetime    time1=TimeCurrent();     // 呼び出しの最初の形式：気配値表示ウィンドウ内のいずれかのシンボルの最後の相場の時刻
   datetime    time2=TimeCurrent(tm);   // 呼び出しの2番目の形式：Mql DateTime構造体を埋めた、気配値ウィンドウ内のいずれかのシンボルの最後の相場の時刻

//--- 日付/時刻を受信し、ログ内の対応するデータを構造体に入力した結果を表示する
   PrintFormat("Tick time: %s\n- Year: %u\n- Month: %02u\n- Day: %02u\n- Hour: %02u\n- Min: %02u\n- Sec: %02u\n- Day of Year: %03u\n- Day of Week: %u (%s)",
               (string)time1, tm.year, tm.mon, tm.day, tm.hour, tm.min, tm.sec, tm.day_of_year, tm.day_of_week, EnumToString((ENUM_DAY_OF_WEEK)tm.day_of_week));
  /*
  結果：
```

© 2000–2025, MetaQuotes Ltd.

```
   Tick time: 2024.04.18 15:40:06
   - Year: 2024
   - Month: 04
   - Day: 18
   - Hour: 15
   - Min: 40
   - Sec: 06
   - Day of Year: 108
   - Day of Week: 4 (THURSDAY)
  */
  }
```

© 2000–2025, MetaQuotes Ltd.

TimeTradeServer

取引サーバの計算された現在時刻を返します。TimeCurrent() と異なり、時間値の計算はクライアント端末で実行されお使いのコンピュータの時刻設定に依存します。この関数には 2 つのバージョンがあります。

パラメータなしでの呼び出し

```
datetime  TimeTradeServer();
```

MqlDateTime 型のパラメータとの呼び出し

```
datetime  TimeTradeServer(
   MqlDateTime&  dt_struct      // 構造体型の変数
   );
```

パラメータ

dt_struct

[out] MqlDateTime 構造体型の変数

戻り値

datetime 型の値

注意事項

MqlDateTime 構造型の変数がパラメータとして渡された場合は、それに応じて満たされます。

高解像度のカウンターとタイマーを配置するには、ミリ秒単位で値を生成する GetTickCount() 関数を使用します。

ストラテジーテスターのテスト中には TimeTradeServer() は常にシミュレーとされたサーバタイムの TimeCurrent() に等しいです。

例：

```
void OnStart()
  {
//--- 日付/時刻データを格納するMql DateTime変数を宣言し、最後の相場の時刻と取引サーバーの推定現在時刻を取得する
   MqlDateTime tm={};
   datetime    time_current=TimeCurrent();                  // 呼び出しの最初の形式：気配値表示ウィンドウ内のいずれかのシンボルの最後の相場の時刻
   datetime    time_server =TimeTradeServer(tm);            // second form of call: estimated current time of the trade server with filling in the MqlDateTime structure
   int         difference  =int(time_current-time_server);  // difference between Time Current and Time Trade Server

//--- ログ内の埋め込まれたMql DateTime構造体のデータを使用して、最後の相場の時刻と取引サーバーの推定現在時刻を表示する
   PrintFormat("Time Current: %s\nTime Trade Server: %s\n- Year: %u\n- Month: %02u\n- Day: %02u\n"+
               "- Hour: %02u\n- Min: %02u\n- Sec: %02u\n- Day of Year: %03u\n- Day of Week: %u (%s)\nDifference between Time Current and Time Trade Server: %+d",
```

© 2000–2025, MetaQuotes Ltd.

```
                (string)time_current, (string)time_server, tm.year, tm.mon, tm.day,
tm.hour, tm.min, tm.sec, tm.day_of_year, tm.day_of_week,
                EnumToString((ENUM_DAY_OF_WEEK)tm.day_of_week), difference);
  /*
  結果:
   Time Current: 2024.04.18 16:10:14
   Time Trade Server: 2024.04.18 16:10:15
   - Year: 2024
   - Month: 04
   - Day: 18
   - Hour: 16
   - Min: 10
   - Sec: 15
   - Day of Year: 108
   - Day of Week: 4 (THURSDAY)
   Difference between Time Current and Time Trade Server: -1
  */
  }
```

© 2000–2025, MetaQuotes Ltd.

TimeLocal

クライアント端末が実行しているコンピュータのローカル時間を返します。この関数には 2 つのバージョンがあります。

パラメータなしでの呼び出し

```
datetime  TimeLocal();
```

MqlDateTime 型のパラメータとの呼び出し

```
datetime  TimeLocal(
   MqlDateTime&  dt_struct      // 構造体型の変数
   );
```

パラメータ

dt_struct

[out] MqlDateTime 構造体型の変数

戻り値

datetime 型の値

注意事項

MqlDateTime 構造型の変数がパラメータとして渡された場合は、それに応じて満たされます。

高解像度のカウンターとタイマーを配置するには、ミリ秒単位で値を生成する GetTickCount() 関数を使用します。

ストラテジーテスターのテスト中には TimeLocal() は常にシミュレーとされたサーバタイムの TimeCurrent() に等しいです。

例：

```
void OnStart()
  {
//--- PCの現地時間データを入れるMqlDateTime変数を宣言する
   MqlDateTime tm={};
   datetime    time1=TimeLocal();   // 最初の呼び出し形式：PCの現地時間
   datetime    time2=TimeLocal(tm); // 2番目の呼び出し形式：MqlDateTime構造体を埋めてPCの現地時間から計算されたGMT

//--- PCの現地時間を受信し、ログ内の対応するデータを構造体に入力した結果を表示する
   PrintFormat("Local time: %s\n- Year: %u\n- Month: %02u\n- Day: %02u\n- Hour: %02u\n- Min: %02u\n- Sec: %02u\n- Day of Year: %03u\n- Day of Week: %u (%s)",
               (string)time1, tm.year, tm.mon, tm.day, tm.hour, tm.min, tm.sec, tm.day_of_year, tm.day_of_week, EnumToString((ENUM_DAY_OF_WEEK)tm.day_of_week));
  /*
  結果:
   Local time: 2024.04.18 19:44:09
   - Year: 2024
   - Month: 04
```

© 2000-2025, MetaQuotes Ltd.

```
   - Day: 18
   - Hour: 19
   - Min: 44
   - Sec: 09
   - Day of Year: 108
   - Day of Week: 4 (THURSDAY)
  */
  }
```

© 2000–2025, MetaQuotes Ltd.

TimeGMT

クライアント端末が実行しているコンピュータのローカル時間のDST スイッチを考慮して、GMTを返します。この関数には 2 つのバージョンがあります。

パラメータなしでの呼び出し

```
datetime  TimeGMT();
```

MqlDateTime 型のパラメータとの呼び出し

```
datetime  TimeGMT(
   MqlDateTime&  dt_struct      // 構造体型の変数
   );
```

パラメータ

dt_struct

[out] MqlDateTime 構造体型の変数

戻り値

datetime 型の値

注意事項

MqlDateTime 構造型の変数がパラメータとして渡された場合は、それに応じて満たされます。

高解像度のカウンターとタイマーを配置するには、ミリ秒単位で値を生成する GetTickCount() 関数を使用します。

ストラテジーテスターのテスト中には TimeGMT() は常にシミュレーターとされたサーバタイムの TimeTradeServer() に等しいです。

例：

```
void OnStart()
  {
//--- 日付/時刻データを格納するMql DateTime変数を宣言し、PCの現地時間とGMTを取得する
   MqlDateTime tm={};
   datetime    time1=TimeLocal();              // 最初の呼び出し形式：PCの現地時間
   datetime    time2=TimeGMT(tm);              // 2番目の呼び出し形式：Mql DateTime構造体を埋めてPCの現地時間から計算されたGMT
   int         shift=int(time1-time2)/3600;    // GMTを基準とした現地時間のオフセット

//--- ログ内の埋め込まれたMql DateTime構造体のデータを使用して現地時間とGMTを表示する
   PrintFormat("Time Local: %s\nTime GMT: %s\n- Year: %u\n- Month: %02u\n- Day: %02u\n"+
               "- Hour: %02u\n- Min: %02u\n- Sec: %02u\n- Day of Year: %03u\n- Day of Week: %u (%s)\nLocal time offset relative to GMT: %+d",
               (string)time1, (string)time2, tm.year, tm.mon, tm.day, tm.hour, tm.min, tm.sec, tm.day_of_year, tm.day_of_week,
               EnumToString((ENUM_DAY_OF_WEEK)tm.day_of_week), shift);
  /*
```

© 2000–2025, MetaQuotes Ltd.

```
結果:
 Time Local: 2024.04.18 19:37:23
 Time GMT: 2024.04.18 12:37:23
 - Year: 2024
 - Month: 04
 - Day: 18
 - Hour: 12
 - Min: 37
 - Sec: 23
 - Day of Year: 108
 - Day of Week: 4 (THURSDAY)
 Local time offset relative to GMT: +7
*/
}
```

© 2000–2025, MetaQuotes Ltd.

TimeDaylightSavings

夏時間への切り替えが行われている場合、夏時間への補正を秒単位で返します。お使いのコンピュータの時刻設定に依存します。

```
int  TimeDaylightSavings();
```

戻り値

冬（標準）時間への切り替えが行われた場合は0を返します。

例：

```
void OnStart()
  {
//--- 夏時間の調整を秒単位で取得する
   int sec_dl=TimeDaylightSavings();

//--- 受け取った値を説明するテキストを作成する
   string text=(sec_dl==0 ? "Standard \"winter\" time is used" :
                StringFormat("Daylight saving time has been switched over. The
correction is %d seconds", sec_dl));

//--- 夏時間調整の説明をログに秒単位で表示する
   Print(text);
   /*
   Result for "winter" time:
   Standard "winter" time is used

   Result for "summer" time:
   Daylight saving time has been switched over. The correction is -3600 seconds
   */
  }
```

© 2000–2025, MetaQuotes Ltd.

TimeGMTOffset

DST スイッチを考慮してGMT時刻とローカルコンピュータの時刻との差を秒で返します。お使いのコンピュータの時刻設定に依存します。

```
int  TimeGMTOffset();
```

戻り値

GMT時刻とローカルコンピュータの時刻TimeLocal()の間の現在の差を秒で表したint型の数値。
TimeGMTOffset() = TimeGMT() - TimeLocal()

例：

```
void OnStart()
  {
//--- 現地時間、GMT、GMTとローカルコンピュータ時刻との差を秒単位で取得する
   datetime time_local=TimeLocal();
   datetime time_gmt  =TimeGMT();
   int      offset    =TimeGMTOffset();

//--- 取得した時間とオフセット値をログに表示する
   PrintFormat("Local Time: %s, GMT Time: %s, Seconds Offset: %+d", (string)
time_local, (string)time_gmt, offset);
  /*
  結果:
   Local Time: 2024.04.18 19:35:52, GMT Time: 2024.04.18 12:35:52, Seconds Offset: -
25200
  */
  }
```

© 2000–2025, MetaQuotes Ltd.

TimeToStruct

datetime 型の値（1970年1月1日から経過した秒数）を MqlDateTime 構造体の変数に変換します。

```
bool  TimeToStruct(
   datetime      dt,            // 日付と時刻
   MqlDateTime&  dt_struct      // 値を採用するための構造体
   );
```

パラメータ

dt

[in] 変換する日付値

dt_struct

[out] MqlDateTime 構造体型の変数

戻り値

成功の場合は true、それ以外の場合は false。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

例：

```
void OnStart()
  {
//--- サーバーの既知の最後の時刻を取得し、日付/時刻構造を宣言し、構造体フィールドに値を入力する
   datetime     time=TimeCurrent();
   MqlDateTime tm  ={};
   if(!TimeToStruct(time,tm))
      Print("TimeToStruct() failed. Error ", GetLastError());

//--- 取得したサーバー時刻と、TimeToStruct()を使用してMqlDateTime構造体に入力した結果をログに表示する
   PrintFormat("Server time: %s\n- Year: %u\n- Month: %02u\n- Day: %02u\n- Hour: %02u\n- Min: %02u\n- Sec: %02u\n- Day of Year: %03u\n- Day of Week: %u (%s)",
               (string)time, tm.year, tm.mon, tm.day, tm.hour, tm.min, tm.sec, tm.day_of_year, tm.day_of_week, EnumToString((ENUM_DAY_OF_WEEK)tm.day_of_week));
  /*
  結果:
   Server time: 2024.04.18 15:47:27
   - Year: 2024
   - Month: 04
   - Day: 18
   - Hour: 15
   - Min: 47
   - Sec: 27
   - Day of Year: 108
   - Day of Week: 4 (THURSDAY)
  */
  }
```

© 2000-2025, MetaQuotes Ltd.

StructToTime

MqlDateTime 構造体型の変数を datetime 型に変換した値を返します。

```
datetime  StructToTime(
   MqlDateTime&  dt_struct      // 日付と時刻の構造体
   );
```

パラメータ

dt_struct

[in] MqlDateTime構造体型の変数

戻り値

1970年1月1日から経過した秒数を含むtdatetime型の値

例：

```
#property script_show_inputs

input int   InpYear  =  0;    // Year
input int   InpMonth =  0;    // Month
input int   InpDay   =  0;    // Day
input int   InpHour  =  0;    // Hour
input int   InpMin   =  0;    // Minutes
input int   InpSec   =  0;    // Seconds

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 入力された値を調整し、変数に書き込む
   int year =  (InpYear<1970 ?1970 : InpYear); // 入力された年が1970年以前の場合は、1970 年が使用される
   int month=  (InpMonth<1 ?  1  :  InpMonth > 12  ?  12 :  InpMonth);
   int day  =  (InpDay  <1 ?  1  :  InpDay   > 31  ?  31 :  InpDay);
   int hour =  (InpHour <0 ?  0  :  InpHour  > 23  ?  23 :  InpHour);
   int min  =  (InpMin  <0 ?  0  :  InpMin   > 59  ?  59 :  InpMin);
   int sec  =  (InpSec  <0 ?  0  :  InpSec   > 59  ?  59 :  InpSec);

//--- 入力された値をログに表示する
   PrintFormat("Entered date and time: %04u.%02u.%02u %02u:%02u:%02u", InpYear, InpMonth, InpDay, InpHour, InpMin, InpSec);

//--- 調整済みの入力値をログに表示する
   PrintFormat("Corrected date and time: %04u.%02u.%02u %02u:%02u:%02u", year, month, day, hour, min, sec);

//--- 入力値を対応する構造体フィールドに書き込む
   MqlDateTime time_struct={};
```

© 2000–2025, MetaQuotes Ltd.

```
   time_struct.year= year;
   time_struct.mon = month;
   time_struct.day = day;
   time_struct.hour= hour;
   time_struct.min = min;
   time_struct.sec = sec;

//--- 構造体の日付と時刻をdateti me型の変数に変換し、
   datetime time = StructToTime(time_struct);

//--- MqlDateTi me構造体型変数から日時型値への変換結果を表示する
   Print("Converted date and time: ",TimeToString(time,TIME_DATE|TIME_MINUTES|
TIME_SECONDS));
  /*
  ゼロのデフォルト値が入力された場合の結果は次のようになります。
   Entered date and time: 0000.00.00 00:00:00
   Corrected date and time: 1970.01.01 00:00:00
   Converted date and time: 1970.01.01 00:00:00

  今年の2月の間違った日を入力した場合の結果は次のようになります。
   Entered date and time: 2024.02.31 00:00:00
   Corrected date and time: 2024.02.31 00:00:00
   Converted date and time: 2024.03.02 00:00:00
  */
  }
```

© 2000-2025, MetaQuotes Ltd.

口座情報

現在の口座のパラメータを返す関数です。

関数	アクション
AccountInfoDouble	対応する口座のプロパティの double 型の値を返します。
AccountInfoInteger	対応する口座のプロパティの integer 型（bool、int または long）の値を返します。
AccountInfoString	対応する口座のプロパティの string 型の値を返します。

© 2000–2025, MetaQuotes Ltd.

AccountInfoDouble

対応する口座のプロパティの値を返します。

```
double  AccountInfoDouble(
   ENUM_ACCOUNT_INFO_DOUBLE  property_id      // プロパティ識別子
   );
```

パラメータ

property_id

[in] プロパティの識別子。可能な値は ENUM_ACCOUNT_INFO_DOUBLE の値のいずれかになります。

戻り値

double 型の値

例:

```
void OnStart()
  {
//--- AccountInfoDouble() 関数で使用可能な情報を全て表示する
   printf("ACCOUNT_BALANCE =  %G",AccountInfoDouble(ACCOUNT_BALANCE));
   printf("ACCOUNT_CREDIT =  %G",AccountInfoDouble(ACCOUNT_CREDIT));
   printf("ACCOUNT_PROFIT =  %G",AccountInfoDouble(ACCOUNT_PROFIT));
   printf("ACCOUNT_EQUITY =  %G",AccountInfoDouble(ACCOUNT_EQUITY));
   printf("ACCOUNT_MARGIN =  %G",AccountInfoDouble(ACCOUNT_MARGIN));
   printf("ACCOUNT_MARGIN_FREE =  %G",AccountInfoDouble(ACCOUNT_MARGIN_FREE));
   printf("ACCOUNT_MARGIN_LEVEL =  %G",AccountInfoDouble(ACCOUNT_MARGIN_LEVEL));
   printf("ACCOUNT_MARGIN_SO_CALL = %G",AccountInfoDouble(ACCOUNT_MARGIN_SO_CALL));
   printf("ACCOUNT_MARGIN_SO_SO = %G",AccountInfoDouble(ACCOUNT_MARGIN_SO_SO));
  }
```

参照

SymbolInfoDouble、SymbolInfoString、SymbolInfoInteger、PrintFormat

© 2000-2025, MetaQuotes Ltd.

AccountInfoInteger

対応する口座のプロパティの値を返します。

```
long  AccountInfoInteger(
   ENUM_ACCOUNT_INFO_INTEGER  property_id      // プロパティ識別子
   );
```

パラメータ

property_id

[in] プロパティの識別子。可能な値は ENUM_ACCOUNT_INFO_INTEGER の値のいずれかになります。

戻り値

long型の値

注意事項

このプロパティは bool、int または long 型に限られています。

例:

```
void OnStart()
  {
//--- AccountInfoInteger() 関数で使用可能な情報を全て表示する
   printf("ACCOUNT_LOGIN =  %d",AccountInfoInteger(ACCOUNT_LOGIN));
   printf("ACCOUNT_LEVERAGE =  %d",AccountInfoInteger(ACCOUNT_LEVERAGE));
   bool thisAccountTradeAllowed=AccountInfoInteger(ACCOUNT_TRADE_ALLOWED);
   bool EATradeAllowed=AccountInfoInteger(ACCOUNT_TRADE_EXPERT);
   ENUM_ACCOUNT_TRADE_MODE tradeMode=(ENUM_ACCOUNT_TRADE_MODE)
AccountInfoInteger(ACCOUNT_TRADE_MODE);
   ENUM_ACCOUNT_STOPOUT_MODE stopOutMode=(ENUM_ACCOUNT_STOPOUT_MODE)
AccountInfoInteger(ACCOUNT_MARGIN_SO_MODE);

//--- 取引操作を実行する可能性について通知する
   if(thisAccountTradeAllowed)
      Print("Trade for this account is permitted");
   else
      Print("Trade for this account is prohibited!");

//--- この口座でエキスパートアドバイザーを使用しての取引が出来るかどうかをみる
   if(EATradeAllowed)
      Print("Trade by Expert Advisors is permitted for this account");
   else
      Print("Trade by Expert Advisors is prohibited for this account!");

//--- 口座の種類を検出する
   switch(tradeMode)
     {
      case(ACCOUNT_TRADE_MODE_DEMO):
         Print("This is a demo account");
```

© 2000-2025, MetaQuotes Ltd.

```
         break;
      case(ACCOUNT_TRADE_MODE_CONTEST):
         Print("This is a competition account");
         break;
      default:Print("This is a real account!");
     }

//--- StopOut レベル設定モードを検出する
   switch(stopOutMode)
     {
      case(ACCOUNT_STOPOUT_MODE_PERCENT):
         Print("The StopOut level is specified percentage");
         break;
      default:Print("The StopOut level is specified in monetary terms");
     }
  }
```

参照

口座情報

© 2000-2025, MetaQuotes Ltd.

AccountInfoString

対応する口座のプロパティの値を返します。

```
string  AccountInfoString(
   ENUM_ACCOUNT_INFO_STRING  property_id      // プロパティ識別子
   );
```

パラメータ

property_id

[in] プロパティの識別子。可能な値は ENUM_ACCOUNT_INFO_STRING の値のいずれかになります。

戻り値

string 型の値

例:

```
void OnStart()
  {
//--- AccountInfoString() 関数で使用可能な情報を全て表示する
   Print("The name of the broker = ",AccountInfoString(ACCOUNT_COMPANY));
   Print("Deposit currency = ",AccountInfoString(ACCOUNT_CURRENCY));
   Print("Client name = ",AccountInfoString(ACCOUNT_NAME));
   Print("The name of the trade server = ",AccountInfoString(ACCOUNT_SERVER));
  }
```

参照

口座情報

© 2000-2025, MetaQuotes Ltd.

状態のチェック

クライアント端末の現在の状態を返す関数です。

関数	アクション
GetLastError	最後に発生したエラーを返します。
IsStopped	MQL5 プログラムがその処理を停止するコマンドを受けた場合は true を返します。
UninitializeReason	初期化解除理由のコードを返します。
TerminalInfoInteger	MQL5 プログラム環境の対応するプロパティの整数値を返します。
TerminalInfoDouble	MQL5 プログラム環境の対応するプロパティの double 値を返します。
TerminalInfoString	MQL5 プログラム環境の対応するプロパティの文字列値を返します。
MQLInfoInteger	実行中のMQL5 プログラムの対応するプロパティの整数値を返します。
MQLInfoString	実行中のMQL5 プログラムの対応するプロパティの文字列値を返します。
Symbol	現在のチャートの銘柄名を返します。
Period	現在のチャートの時間軸を返します。
Digits	チャートシンボルの価格の値の精度を決定する小数点以下の桁数を返します。
Point	現在の通貨/シンボルのポイントサイズを返します。

© 2000–2025, MetaQuotes Ltd.

GetLastError

_LastError システム変数の内容を返します。

```
int  GetLastError();
```

戻り値

MQL5 プログラム実行中に起こった最後のエラー

注意事項

_LastError は関数呼び出し後にリセットされません。この変数をリセットするには ResetLastError() を呼び出す必要があります。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   MqlRates rates[1]={};   // display the current bar data here

//--- 意図的に不適切なパラメータで関数を呼び出す
   int res=CopyRates(NULL, PERIOD_CURRENT, 0, 2, rates);
   if(res!=2)
      PrintFormat("CopyRates() returned %d. LastError %d", res, GetLastError());

//--- 現在のバーデータをMql Rates構造体にコピーする前に、最後のエラーコードをリセットする
   ResetLastError();
//--- 関数が正しく動作しない場合、エラーコードは0と異なる
   CopyRates(NULL, PERIOD_CURRENT, 0, 1, rates);
   Print("CopyRates() error ", GetLastError());

//--- 取得した値の配列を出力する
   ArrayPrint(rates);
  }
```

参照

取引サーバのリターンコード

© 2000-2025, MetaQuotes Ltd.

IsStopped

MQL5 プログラムの強制シャットダウンをチェックします。

```
bool  IsStopped();
```

戻り値

_StopFlagシステム変数が 0 以外の場合は true 。MQL5 プログラムが操作を完了するように命令されている場合、0 以外の値が _StopFlag に書き入れられます。この場合、すぐにプログラムを終了する必要があります。そうしないとプログラムは 3 秒後に外部から強制的に終了されます。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 停止チェック付きの無限ループ
   while(!IsStopped())
     {
      //--- グラフにローカルPCの時間を表示する
      Comment("Time Local: ", TimeToString(TimeLocal(), TIME_DATE|TIME_MINUTES|
TIME_SECONDS));
      Sleep(16);
     }
   Print("The StopFlag is set. The program will be stopped.");

//--- クリーンアップ
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

UninitializeReason

初期化解除理由のコードを返します。。

```
int  UninitializeReason();
```

戻り値

OnDeinit()が呼び出される前に作成される _UninitReason の値。値は初期化解除に至った理由に依存します。

例：

```
//+------------------------------------------------------------------+
//| エキスパート初期化関数                                           |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 初期化解除理由コードをreason変数に取得する
   int reason=UninitializeReason();
//--- 初期化解除の理由を含むメッセージ文字列を作成し、メッセージを操作ログに送信する
   string message=StringFormat("%s: Uninitialize reason code: %d (%s)",__FUNCTION__,
reason, UninitializeReasonDescription(reason));
   Print(message);
//--- 成功
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| エキスパート初期化解除関数                                       |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- reason仮変数から初期化解除理由コードを含むメッセージ文字列を作成し、メッセージを操作ログに送信する
   string message=StringFormat("%s: Uninitialize reason code: %d (%s)",__FUNCTION__,
reason, UninitializeReasonDescription(reason));
   Print(message);
  }
//+------------------------------------------------------------------+
//| エキスパートティック関数                                         |
//+------------------------------------------------------------------+
void OnTick()
  {
  }
//+------------------------------------------------------------------+
//| 初期化解除の理由コードの説明を戻す                               |
//+------------------------------------------------------------------+
string UninitializeReasonDescription(const int reason)
  {
   switch(reason)
     {
      //--- EAはExpertRemove()関数の呼び出しを停止した
```

© 2000-2025, MetaQuotes Ltd.

```
      case REASON_PROGRAM :
        return("Expert Advisor terminated its operation by calling the
ExpertRemove() function");
      //--- チャートからプログラムを削除した
      case REASON_REMOVE :
        return("Program has been deleted from the chart");
      //--- プログラムが再コンパイルされた
      case REASON_RECOMPILE :
        return("Program has been recompiled");
      //--- シンボルまたはチャート期間が変更された
      case REASON_CHARTCHANGE :
        return("Symbol or chart period has been changed");
      //--- チャートが閉じられた
      case REASON_CHARTCLOSE :
        return("Chart has been closed");
      //--- ユーザーによって入力が変更された
      case REASON_PARAMETERS :
        return("Input parameters have been changed by a user");
      //--- 別のアカウントが有効化されたか、アカウント設定の変更により取引サーバーへの再接続が発生した
      case REASON_ACCOUNT :
        return("Another account has been activated or reconnection to the trade
server has occurred due to changes in the account settings");
      //--- 別の定形チャートが適用された
      case REASON_TEMPLATE :
        return("A new template has been applied");
      //--- OnInit()ハンドラがゼロ以外の値を返した
      case REASON_INITFAILED :
        return("This value means that OnInit() handler has returned a nonzero
value");
      //--- ターミナルが閉じられた
      case REASON_CLOSE :
        return("Terminal has been closed");
     }

//--- 初期化解除の理由は不明
   return("Unknown reason");
  }
```

© 2000-2025, MetaQuotes Ltd.

TerminalInfoInteger

MQL5 プログラム環境の対応するプロパティの値を返します。

```
int  TerminalInfoInteger(
   int  property_id      // プロパティ識別子
   );
```

パラメータ

property_id

[in] プロパティの識別子。ENUM_TERMINAL_INFO_INTEGER 列挙のいずれかです。

戻り値

int 型の値

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 実行中のターミナルのビルド番号と「64-bit terminal」プロパティを取得する
   int  build = TerminalInfoInteger(TERMINAL_BUILD);
   bool x64   = TerminalInfoInteger(TERMINAL_X64);

//--- 取得したターミナルデータを操作ログに出力する
   PrintFormat("MetaTrader 5 %s build %d", (x64 ? "x64" : "x32"), build);
   /*
  結果:
   MetaTrader 5 x64 build 4330
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

TerminalInfoDouble

MQL5 プログラム環境の対応するプロパティの値を返します。

```
double  TerminalInfoDouble(
   int  property_id      // プロパティ識別子
   );
```

パラメータ

property_id

[in] プロパティの識別子。ENUM_TERMINAL_INFO_DOUBLE 列挙のいずれかです。

戻り値

double 型の値

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 実行中のターミナルのビルド番号と「64-bit terminal」プロパティを取得する
   int  build = TerminalInfoInteger(TERMINAL_BUILD);
   bool x64   = TerminalInfoInteger(TERMINAL_X64);

//--- 取得したターミナルデータを操作ログに出力する
   PrintFormat("MetaTrader 5 %s build %d", (x64 ? "x64" : "x32"), build);
   /*
  結果:
   MetaTrader 5 x64 build 4330
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

TerminalInfoString

MQL5 プログラム環境の対応するプロパティの値を返します。プロパティは文字列型でなければなりません。

```
string  TerminalInfoString(
   int  property_id      // プロパティ識別子
   );
```

パラメータ

property_id

[in] プロパティの識別子。ENUM_TERMINAL_INFO_STRING 列挙のいずれかです。

戻り値

string 型の値

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- OSとターミナルデータを取得する
   string os_ver  = TerminalInfoString(TERMINAL_OS_VERSION);       // user's OS
   string name    = TerminalInfoString(TERMINAL_NAME);             // terminal name
   string path    = TerminalInfoString(TERMINAL_PATH);             // folder the terminal is launched from
   string data    = TerminalInfoString(TERMINAL_DATA_PATH);        // folder for storing the terminal data
   string common  = TerminalInfoString(TERMINAL_COMMONDATA_PATH); // common folder for all client terminals installed on the computer

//--- 取得したデータを操作ログに送信する
   PrintFormat("OS: %s\nTerminal: %s\n- Path: %s\n- Data path: %s\n- Common Data path: %s", os_ver, name, path, data, common);
   /*
   OS: Windows 10 build 19045
   Terminal: MetaTrader 5
   - Path: E:\MetaQuotes\MetaTrader 5
   - Data path: E:\MetaQuotes\MetaTrader 5
   - Common Data path: C:\Users\admin\AppData\Roaming\MetaQuotes\Terminal\Common
   */
  }
```

© 2000–2025, MetaQuotes Ltd.

MQLInfoInteger

実行中のMQL5プログラムの対応するプロパティの値を返します。

```
int  MQLInfoInteger(
   int  property_id      // プロパティ識別子
   );
```

パラメータ

property_id

[in] プロパティの識別子。ENUM_MQL_INFO_INTEGER 列挙のいずれかです。

戻り値

int 型の値

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- MQLプログラムの使用可能なメモリと消費されたメモリの値を取得する
   int limit = MQLInfoInteger(MQL_MEMORY_LIMIT);    // MQL5プログラムの最大可能動的メモリ量(MB単位)
   int used  = MQLInfoInteger(MQL_MEMORY_USED);     // MQL5プログラムによって使用されるメモリ量(MB単位)

//--- 取得した値を操作ログに送信する
   PrintFormat("Maximum possible amount of dynamic memory for MQL5 program: %d Mb\n"+
               "Memory used by MQL5 program: %d Mb", limit, used);
   /*
  結果
   Maximum possible amount of dynamic memory for MQL5 program: 8388608 Mb
   Memory used by MQL5 program: 2 Mb
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

MQLInfoString

実行中のMQL5プログラムの対応するプロパティの値を返します。

```
string  MQLInfoString(
   int  property_id      // プロパティ識別子
   );
```

パラメータ

property_id

[in] プロパティの識別子。ENUM_MQL_INFO_STRING 列挙のいずれかです。

戻り値

string 型の値

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 起動したプログラムデータを取得する
   string name = MQLInfoString(MQL_PROGRAM_NAME);  // name of the launched MQL5 program
   string path = MQLInfoString(MQL_PROGRAM_PATH);  // running program path

//--- 取得したデータを操作ログに送信する
   PrintFormat("Name of the running MQL program: '%s'\nPath of the running MQL program: %s", name, path);
   /*
  結果:
   Name of the running MQL program: 'MQLInfoString'
   Path of the running MQL program: E:\MetaQuotes\MetaTrader 5\MQL5\Scripts\MQLInfoString.ex5
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

Symbol

現在のチャートの銘柄名を返します。

```
string  Symbol();
```

戻り値

現在のチャートの銘柄名を含む _Symbol システム関数の値

注意事項

エキスパートアドバイザー、指標、スクリプトとは異なり、サービスは特定のチャートに縛られません。よって、Symbol()はサービスに空の文字列("")を返します。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 現在のチャート銘柄名を取得する
   string name = Symbol();

//--- 取得したデータを操作ログに送信する
   PrintFormat("Current chart symbol name: '%s'", name);
   /*
  結果
   Current chart symbol name: 'EURUSD'
   */
  }
```

© 2000–2025, MetaQuotes Ltd.

Period

現在のチャートの時間軸を返します。

```
ENUM_TIMEFRAMES  Period();
```

戻り値

現在のチャートの時間軸を含む _Period 変数の内容 ENUM_TIMEFRAMES 列挙のいずれかです。

参照

PeriodSeconds、チャート時間軸、日付と時刻、オブジェクトの可視性

注意事項

エキスパートアドバイザー、指標、スクリプトとは異なり、サービスは特定のチャートに縛られません。よって、Period()はサービスに0を返します。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 現在のチャートの時間枠の値とその説明を取得する
   ENUM_TIMEFRAMES period    = Period();
   string          timeframe = StringSubstr(EnumToString(period), 7);

//--- 取得したデータを操作ログに送信する
   PrintFormat("Current chart timeframe: %s\nTimeframe value: %s (%d)",
               timeframe, EnumToString(period), period);
   /*
  結果:
   Current chart timeframe: H4
   Timeframe value: PERIOD_H4 (16388)
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

Digits

現在のチャートシンボルの価格の値の精度を決定する小数点以下の桁数を返します。

```
int  Digits();
```

戻り値

現在のチャートシンボルの価格の値の精度を決定する小数点以下の桁数を保存する_Digits 変数の値

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 現在のチャートシンボルの小数点以下の桁数を取得する
   int digits = Digits();

//--- 取得したデータを操作ログに送信する
   Print("Number of decimal digits for the current chart symbol: ", digits);
   /*
  結果:
   Number of decimal digits for the current chart symbol: 5
   */
  }
```

© 2000–2025, MetaQuotes Ltd.

Point

現在の通貨/シンボルのポイントサイズを返します。

```
double  Point();
```

戻り値

現在の通貨/シンボルのポイントサイズを含む _Point 変数の内容

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 相場通貨の現在のシンボルポイントサイズを取得する
   double point = Point();

//--- 取得したデータを操作ログに送信する
   Print("Point size of the current symbol in the quote currency: ",
DoubleToString(point, _Digits));
   /*
  結果:
   Point size of the current symbol in the quote currency: 0.00001
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

イベント処理

MQL5言語ではいくつかの定義済みイベントが処理されます。これらのイベントを処理するための関数はMQL5　プログラムで定義されている必要があります。関数名、戻り値の型、パラメータの組成物（もしあれば）と型は、イベントハンドラ関数の記述に厳格に適合しなければなりません。

クライアント端末イベントハンドラーは、戻り値およびパラメーターの型を使用して、イベントを処理する関数を識別します。関数に、以下の記述に対応しないいくつかのパラメータまたは戻り値の型がある場合、そのような関数をイベントの処理に使用することはできません。

関数	説明
OnStart	この関数は、Startイベントがスクリプト内で設定されたアクションを実行するために発生したときに呼び出されます
OnInit	この関数は、Initイベントが発生して起動したMQL5プログラムが初期化されるときに、指標やEAで呼び出されます。
OnDeinit	この関数は、Deinitイベントが発生して起動したMQL5プログラムが初期化解除されるときに、指標やEAで呼び出されます。
OnTick	この関数は、NewTickイベントが発生して新しい気配値が処理されるときに、EAで呼び出されます。
OnCalculate	この関数は、Calculateイベントが発生して価格の変化が処理されるときに、指標で呼び出されます。
OnTimer	この関数は、一定の時間間隔でターミナルによって生成されるTimerイベント中に、指標やEAで呼び出されます。
OnTrade	この関数は、取引サーバ上の取引操作の終了時に生成されるTradeイベント中に、EAで呼び出されます。
OnTradeTransaction	この関数は、TradeTransactionイベントが発生して取引リクエスト実行結果が処理されるときに、EAで呼び出されます。
OnBookEvent	この関数は、BookEventイベントが発生して板情報の変化が処理されるときに、EAで呼び出されます。
OnChartEvent	この関数は、ChartEventイベントが発生してユーザまたはMQL5プログラムによるチャートの変化が処理されるときに、指標やEAで呼び出されます。
OnTester	この関数は、Testerイベントが発生してEAをテストした後に履歴データに対して必要なアクションが実行されるときに、EAで呼び出されます。
OnTesterInit	この関数は、 TesterInitイベントが発生してストラテジーテスターでの最適化に先立って必要なアクションが実行されるときに、EAで呼び出されます。
OnTesterDeinit	この関数は、ストラテジーテスターでの最適化の後にTesterDeinit イベントが発生したときに、EAで呼び出されます。
OnTesterPass	この関数は、ストラテジーテスターでのEAの最適化の際に TesterPassイベントが発生して新しい時間枠の到着を処理するときに、EAで呼び出されます。

© 2000–2025, MetaQuotes Ltd.

クライアント端末は、着信イベントを対応するオープンチャートに送信します。イベントはチャート（チャートイベント）またはMQL5プログラム（カスタムイベント）によっても生成出来ます。チャート上のグラフィカルオブジェクトの作成または削除のイベントの生成はCHART_EVENT_OBJECT_CREATE及びCHART_EVENT_OBJECT_DELETEチャートプロパティによって有効または無効に出来ます。各MQL5アプリケーションとチャートは、新たに到着したイベントが配置される独自のイベントキューを持っています。

プログラムは、それが動作しているチャートからのイベントのみを受け取ります。全てのイベントは、受信された順に次々に処理されます。キューにすでにNewTickイベントがある、またはこのイベントが処理段階にある場合、新しいNewTickイベントはMQL5アプリケーションキューには追加されません。似たようにChartEventがすでにMQL5プログラムキューに存在する場合、またはこのイベントが現在処理中の場合、同じ型のイベントはキューには追加されません。Timerイベント処理も同じように扱われ、Timerイベントがキューに存在するか処理中の場合、新しいTmerイベントはキューに追加されません。

イベントキューは限られてはいますが充分な大きさを持っています。よく書かれたプログラムでのキューのオーバーフローは起こりにくいです。オーバーフローの場合、新しいイベントはキューに追加されずに廃棄されます。

無限ループを使用してのイベント処理は推奨されません。単一のStartイベントを処理するスクリプトは例外です。

ライブラリはイベントを処理しません。

© 2000–2025, MetaQuotes Ltd.

OnStart

この関数はStartイベントの発生時にスクリプトとサービスで呼び出されます。この関数は、プログラムに実装されているアクションの1回限りの実行を目的としています。関数には2種類あります。

結果を返すバージョン

```
int  OnStart(void);
```

戻り値

操作ログタブで表示されるint型の値。

スクリプトの実行が完成すると「script script_name removed (result code N)」というエントリがターミナルの操作ログに作成されます。ここで、NはOnStart()関数によって返される値です。

サービスの実行が完了すると、エントリ「service service_name stopped (result code N)」がターミナル操作ログに作成されます。ここで、NはOnStart()関数によって返される値です。

実行結果を返すOnStart()の呼び出しは、スクリプトまたはサービスの実行を許可するだけでなく、プログラムの実行結果を分析するためのエラーコードまたはその他の有用なデータも返すため、使用をお勧めします。

結果を返さないバージョンは古いコードとの相互性の為のみに残されています。使用は推奨されません。

```
void  OnStart(void);
```

注意事項

OnStart()はスクリプトおよびサービス内のイベントを処理するための唯一の関数です。これらのプログラムに他のイベントは送信されません。その結果、Startイベントはエキスパートおよびカスタム指標には渡されません。

スクリプトの例:

```
//--- 色を使うためのマクロ
#define XRGB(r,g,b)     (0xFF000000|(uchar(r)<<16)|(uchar(g)<<8)|uchar(b))
#define GETRGB(clr)     ((clr)&0xFFFFFF)
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 下降ローソク足の色を設定する
   Comment("Set a downward candle color");
   ChartSetInteger(0,CHART_COLOR_CANDLE_BEAR,GetRandomColor());
   ChartRedraw(); // 新しいティックを待たずにチャートをすぐに更新する
   Sleep(1000);   // すべての変更を表示するために1秒間一時停止する
//--- 上昇ローソク足の色を設定する
   Comment("Set an upward candle color");
   ChartSetInteger(0,CHART_COLOR_CANDLE_BULL,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- 背景色を設定する
   Comment("Set the background color");
```

© 2000-2025, MetaQuotes Ltd.

```
   ChartSetInteger(0,CHART_COLOR_BACKGROUND,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- 買値線の色を設定する
   Comment("Set color of Ask line");
   ChartSetInteger(0,CHART_COLOR_ASK,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- 売値線の色を設定する
   Comment("Set color of Bid line");
   ChartSetInteger(0,CHART_COLOR_BID,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- 下向きバーと下向きローソク足フレームの色を設定する
   Comment("Set color of a downward bar and a downward candle frame");
   ChartSetInteger(0,CHART_COLOR_CHART_DOWN,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- チャートの線と同時ローソク足の色を設定する
   Comment("Set color of a chart line and Doji candlesticks");
   ChartSetInteger(0,CHART_COLOR_CHART_LINE,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- 上向きバーと上向きローソク足フレームの色を設定する
   Comment("Set color of an upward bar and an upward candle frame");
   ChartSetInteger(0,CHART_COLOR_CHART_UP,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- 軸、スケール、OHLCラインの色を設定する
   Comment("Set color of axes, scale and OHLC line");
   ChartSetInteger(0,CHART_COLOR_FOREGROUND,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- グリッドの色を設定する
   Comment("Set a grid color");
   ChartSetInteger(0,CHART_COLOR_GRID,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- 直近値の色を設定する
   Comment("Set Last price color");
   ChartSetInteger(0,CHART_COLOR_LAST,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- ストップロスおよびテイクプロフィット注文値の色を設定する
   Comment("Set color of Stop Loss and Take Profit order levels");
   ChartSetInteger(0,CHART_COLOR_STOP_LEVEL,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- ボリュームとマーケットエントリーレベルの色を設定する
```

© 2000-2025, MetaQuotes Ltd.

```
   Comment("Set color of volumes and market entry levels");
   ChartSetInteger(0,CHART_COLOR_VOLUME,GetRandomColor());
   ChartRedraw();
  }
//+------------------------------------------------------------------+
//|  ランダムに生成された色を返す                                    |
//+------------------------------------------------------------------+
color GetRandomColor()
  {
   color clr=(color)GETRGB(XRGB(rand()%255,rand()%255,rand()%255));
   return clr;
  }
```

参照

Event handling functions、Program running、Client terminal events

© 2000–2025, MetaQuotes Ltd.

OnInit

この関数は、Initイベントが発生するときに指標やEAで呼び出され、実行中のMQL5プログラムを初期化するのに使用されます。この関数の型は 2 つあります。

結果を返すバージョン

```
int  OnInit(void);
```

戻り値

int型の値、ゼロは正常な初期化

INIT_FAILEDを返すと、EAはチャートから強制的にアンロードされます。

INIT_FAILEDを返すと、指標はチャートからアンロードされません。チャート上に残っている指標は動作しません。イベントハンドラは指標内で呼び出されません。

実行結果を返すOnInit()を呼び出すと、プログラムを初期化できるだけでなく、プログラムが異常に終了した場合にはエラーコードが返されるため、これを使用することをお勧めします。

結果を返さないバージョンは古いコードとの相互性の為のみに残されています。使用は推奨されません。

```
void  OnInit(void);
```

注意事項

Initイベントは、EAまたは指標を読み込んだ直後に生成されます。このイベントはスクリプトでは生成されません。OnInit()関数は、MQL5プログラムを初期化するために使用されます。OnInit()の戻り値がint型の場合、ゼロ以外の戻り値は初期化が失敗したことを示し、Deinitイベントを REASON_INITFAILED初期化解除の理由コードで作成します。

void型のOnInit()関数は常に初期化が成功したことを示すので、使用は推奨されません。

EAの入力を最適化するために、リターンコードとしてENUM_INIT_RETCODE列挙体からの値を使用することがおすすめです。これらの値は、最も適切なテストエージェントの選択などの最適化プロセス管理を確立するためのものです。テストを始める前に、EAの初期化中にTerminalInfoInteger()関数を使用して、エージェントの設定とリソース(コア数、空きメモリ量など)のデータをリクエストすることができます。　取得したデータに基づいて、テストエージェントの使用を許可するか、EAの最適化を禁止することができます。

ID	説明
INIT_SUCCEEDED	初期化が成功し、EAのテストを続けることが出来ます。 このコードはゼロ値と同じ意味を持ちます。EAはテスター内で正常に初期化されています。
INIT_FAILED	初期化が失敗しました。致命的なエラーが起こったのでテストを続ける理由がありません。たとえば、EAの操作に必要な指標を作成することは不可能です。 この戻り値はゼロ以外の値と同じ意味です。　テスター内でのEAの初期化は失敗しました。
INIT_PARAMETERS_INCORRECT	プログラマーの間違った入力パラメータを示すように設計されています。一般最適化テーブルでは、この戻りコードを含む結果文字列

© 2000–2025, MetaQuotes Ltd.

ID	説明
	が赤で強調表示されます。 このようなEA入力セットのテストは実行されません。エージェントは新しいタスクを受け取る用意ができています。 この値を受け取ると、ストラテジーテスターはこのタスクを他のエージェントに再試行のために渡しません。
INIT_AGENT_NOT_SUITABLE	初期化時にプログラム実行エラーはありませんでしたが、エージェントが何らかの理由でテストに適していません。例えば、RAM不足、OpenCLサポートがないなどです。 このコードが返された後は、エージェントは この最適化が終了するまでタスクを受信しません。

OnInit()を使用してテスターでINIT_FAILED/INIT_PARAMETERS_INCORRECTを返すと、EAの最適化の際に考慮されるべきことがあります。

- OnInit()がINIT_PARAMETERS_INCORRECTを返すのに使用したパラメータはテストには不向きとみなされ、遺伝的最適化では次世代の取得には使用されません。最適なEAパラメータを検索する際に、「廃棄された」パラメータセットが多すぎると誤った結果につながる可能性があります。検索アルゴリズムでは、最適化基準関数は滑らかで、多数の入力パラメータにギャップがないと仮定しています。
- OnInit()がINIT_FAILEDを返した場合、これはテストを起動できないことを意味するため、EAはエージェントのメモリからアンロードされます。EAは新しいパラメータのセットで次のパスを実行するために再び読み込まれます。次の最適化パスを起動すると、TesterStop()を呼び出すよりもはるかに時間がかかります。

EAのOnInit()関数の例

```
//--- 入力パラメータ
input int      ma_period=20; // 移動平均期間

//--- EAで使用される指標のハンドル
int indicator_handle;
//+------------------------------------------------------------------+
//| エキスパート初期化関数                                           |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- check ma_period validity
   if(ma_period<=0)
     {
      PrintFormat("Invalid ma_period input value: %d",ma_period);
      return (INIT_PARAMETERS_INCORRECT);
     }
//--- 最適化中
   if(MQLInfoInteger(MQL_OPTIMIZATION))
     {
      //--- エージェントが利用できるRAMを確認する
      int available_memory_mb=TerminalInfoInteger(TERMINAL_MEMORY_TOTAL);
      if(available_memory_mb<2000)
        {
         PrintFormat("Insufficient memory for the test agent: %d MB",
```

© 2000-2025, MetaQuotes Ltd.

```
                        available_memory_mb);
          return (INIT_AGENT_NOT_SUITABLE);
         }
      }
//--- 指標を確認する
   indicator_handle=iCustom(_Symbol,_Period,"My_Indicator",ma_period);
   if(indicator_handle==INVALID_HANDLE)
     {
      PrintFormat("Failed to generate My_Indicator handle. Error code %d",
                  GetLastError());
      return (INIT_FAILED);
     }
//--- EA初期化完了
   return(INIT_SUCCEEDED);
  }
```

参照

OnDeinit, Event handling functions、Program running、Client terminal events、Initialization of variables、Creating and deleting objects

© 2000–2025, MetaQuotes Ltd.

OnDeinit

この関数はDeinitイベントが発生するときに指標やEAで呼び出され、実行中のMQL5プログラムの初期化を解除するのに使用されます。

```
void  OnDeinit(
   const int  reason          // 初期化解除の理由コード
   );
```

パラメータ

reason

[in] 初期化解除の理由コード

戻り値

なし

注意事項

以下の場合、Deinitイベントが*EA*と指標に生成されます。

- MQL5 プログラムが接続されているシンボルまたはチャートの期間の変更に伴う再初期化の前
- 入力の変更に伴う再初期化の前
- MQL5プログラムをアンロードする前

*reason*パラメータには下記の値を含むことができます。

定数	値	説明
REASON_PROGRAM	0	ExpertRemove()関数の呼び出しでEAが作動しなくなった
REASON_REMOVE	1	プログラムがチャートから削除された
REASON_RECOMPILE	2	プログラムが再コンパイルされた
REASON_CHARTCHANGE	3	シンボルまたはチャート期間が変更された
REASON_CHARTCLOSE	4	チャートが閉じられた
REASON_PARAMETERS	5	ユーザが入力を変更した
REASON_ACCOUNT	6	別のアカウントが有効化されるか、アカウントの設定変更によって取引サーバへの再接続が発生した
REASON_TEMPLATE	7	他のチャートテンプレートが適用された
REASON_INITFAILED	8	OnInit()ハンドラーがゼロ以外の値を返した
REASON_CLOSE	9	端末が閉じられた

EAの初期化解除の理由コードはUninitializeReason()関数または事前定義された _UninitReason変数から受信されます。

EAのOnInit()およびOnDeinit()関数の例

```
input int fake_parameter=3;       // 無用のパラメータ
```

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  エキスパート初期化関数                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- プログラムがコンパイルされるビルド数を取得する
   Print(__FUNCTION__," Build #",__MQLBUILD__);
//--- リセットされた理由コードはOnInit()でも取得できる
   Print(__FUNCTION__," Deinitialization reason code can be received during the EA reset");
//--- 初期化解除の理由コードを取得する1番目の方法
   Print(__FUNCTION__," _UninitReason = ",getUninitReasonText(_UninitReason));
//--- 初期化解除の理由コードを取得する2番目の方法
   Print(__FUNCTION__," UninitializeReason() = ",getUninitReasonText(UninitializeReason()));
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//|  エキスパート初期化解除に使用される関数                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- 初期化解除の理由コードを取得する1番目の方法
   Print(__FUNCTION__," Deinitialization reason code = ",reason);
//--- 初期化解除の理由コードを取得する2番目の方法
   Print(__FUNCTION__," _UninitReason = ",getUninitReasonText(_UninitReason));
//--- 初期化解除の理由コードを取得する3番目の方法
   Print(__FUNCTION__," UninitializeReason() = ",getUninitReasonText(UninitializeReason()));
  }
//+------------------------------------------------------------------+
//|  初期化理由コードのテキスト記述を返す                            |
//+------------------------------------------------------------------+
string getUninitReasonText(int reasonCode)
  {
   string text="";
//---
   switch(reasonCode)
     {
      case REASON_ACCOUNT:
         text="Account was changed";break;
      case REASON_CHARTCHANGE:
         text="Symbol or timeframe was changed";break;
      case REASON_CHARTCLOSE:
         text="Chart was closed";break;
      case REASON_PARAMETERS:
         text="Input-parameter was changed";break;
      case REASON_RECOMPILE:
```

© 2000–2025, MetaQuotes Ltd.

```
         text="Program "+__FILE__+" was recompiled";break;
      case REASON_REMOVE:
         text="Program "+__FILE__+" was removed from chart";break;
      case REASON_TEMPLATE:
         text="New template was applied to chart";break;
      default:text="Another reason";
     }
//---
   return text;
  }
```

参照

OnInit, Event handling functions、Program running、Client terminal events、Uninitialization reason codes、Visibility scope and lifetime of variables、Creating and deleting objects

© 2000–2025, MetaQuotes Ltd.

OnTick

この関数は、NewTickイベントが発生して新しい相場が処理されるときに、EAで呼び出されます。

```
void  OnTick(void);
```

戻り値

なし

注意事項

NewTickイベントは、EAが取り付けられているチャートのシンボルに対して新しいティックを受け取ったときに、EAに対してのみ生成されます。NewTickイベントが生成されていないため、カスタム指標またはスクリプトでOnTick()関数を定義することはできません。

Tickイベントは EAに対してのみ生成されますが、Timer、BookEvent、およびChartEventイベントはNewTickに加えてEAに対しても生成されるため、EAにOnTick()関数を実装する必要はありません。

全てのイベントは、受信された順に次々に処理されます。キューにすでに NewTickイベントがある、またはこのイベントが処理段階にある場合、新しいNewTickイベントはMQL5アプリケーションキューには追加されません。

NewTickイベントは、自動取引が有効になっているかどうかに関係なく生成されます(自動取引ボタン)。自動取引の無効化とは、EAからの取引リクエスト送信を禁止することのみを意味します。EAの動作は停止されません。

AutoTradingボタンを押して自動取引を無効にしても、OnTick()関数の現在の実行が中断されることはありません。

OnTick()関数内の取引ロジック全体を特長とするEAの例

```
//+------------------------------------------------------------------+
//|                                                   TradeByATR.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Sample EA trading in the \"explosive\" candle direction"
#property description "\"Explosive\" candle has the body size exceeding k*ATR"
#property description "The \"revers\" parameter reverses the signal direction"

input double lots=0.1;         // ロット単位のボリューム
input double kATR=3;           // ATRのシグナルローソク足の長さ
input int    ATRperiod=20;     // ATR指標期間
input int    holdbars=8;       // ポジションのバー数
input int    slippage=10;      // 許容されるスリッページ
input bool   revers=false;     // シグナルを反転するかどうか
input ulong  EXPERT_MAGIC=0;   // EAのマジックナンバー
//--- ATR指標ハンドルを格納
int atr_handle;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- ここで最後のATR値とローソク足のの実体を保存する
double last_atr,last_body;
datetime lastbar_timeopen;
double trade_lot;
//+------------------------------------------------------------------+
//| エキスパート初期化関数                                           |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- グローバル変数を初期化する
   last_atr=0;
   last_body=0;
//--- 正しいボリュームを設定する
   double min_lot=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
   trade_lot=lots>min_lot?lots:min_lot;
//--- ATR指標ハンドルを作成する
   atr_handle=iATR(_Symbol,_Period,ATRperiod);
   if(atr_handle==INVALID_HANDLE)
     {
      PrintFormat("%s: failed to create iATR, error code %
d",__FUNCTION__,GetLastError());
      return(INIT_FAILED);
     }
//--- EA初期化が成功した
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| エキスパート初期化解除に使用される関数                           |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- EA操作終了コードを通知する
   Print(__FILE__,": Deinitialization reason code = ",reason);
  }
//+------------------------------------------------------------------+
//| エキスパートティック関数                                         |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- 取引シグナル
   static int signal=0; // +1は買いシグナル、 -1は売りシグナル
//--- 'holdbars' バーより前に開かれた以前のポジションを確認して決済する
   ClosePositionsByBars(holdbars,slippage,EXPERT_MAGIC);
//--- 新しいバーを確認する
   if(isNewBar())
     {
      //--- シグナルの存在を確認する
      signal=CheckSignal();
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- ネットポジションが開かれている場合は、信号をスキップして終了する
   if(signal!=0 && PositionsTotal()>0 && (ENUM_ACCOUNT_MARGIN_MODE)
AccountInfoInteger(ACCOUNT_MARGIN_MODE)==ACCOUNT_MARGIN_MODE_RETAIL_NETTING)
     {
      signal=0;
      return; // NewTick イベントハンドラを終了し、新しいバーが現れる前に市場に入らない
     }
//--- ヘッジ口座の場合、各ポジションは個別に保持され、決済される
   if(signal!=0)
     {
      //--- 買いシグナル
      if(signal>0)
        {
         PrintFormat("%s: Buy signal!Revers=%s",__FUNCTION__,string(revers));
         if(Buy(trade_lot,slippage,EXPERT_MAGIC))
            signal=0;
        }
      //--- 売りシグナル
      if(signal<0)
        {
         PrintFormat("%s: Sell signal!Revers=%s",__FUNCTION__,string(revers));
         if(Sell(trade_lot,slippage,EXPERT_MAGIC))
            signal=0;
        }
     }
//--- OnTick 関数の終わり
  }
//+------------------------------------------------------------------+
//| 新しい取引シグナルを確認する                                     |
//+------------------------------------------------------------------+
int CheckSignal()
  {
//--- 0はシグナルが不在なことを意味する
   int res=0;
//--- 終わりから2番目の完全なバーにATR値を取得する(バーのインデックスは2)
   double atr_value[1];
   if(CopyBuffer(atr_handle,0,2,1,atr_value)!=-1)
     {
      last_atr=atr_value[0];
      //--- 最後に閉じたバーのデータをMql Rates型配列に取得する
      MqlRates bar[1];
      if(CopyRates(_Symbol,_Period,1,1,bar)!=-1)
        {
         //--- 最後の完全なバーでバー本体のサイズを計算する
         last_body=bar[0].close-bar[0].open;
         //--- 最後のバー(インデックス1)の本体が以前のATR値(インデックス2のバー上)を超える場合、取引シグナル
が受信される
         if(MathAbs(last_body)>kATR*last_atr)
            res=last_body>0?1:-1; // 上昇ローソク足では正の値
```

© 2000-2025, MetaQuotes Ltd.

```
         }
       else
          PrintFormat("%s: Failed to receive the last
bar!Error",__FUNCTION__,GetLastError());
      }
   else
      PrintFormat("%s: Failed to receive ATR indicator
value!Error",__FUNCTION__,GetLastError());
//--- 反転取引モードが有効な場合
   res=revers?-res:res;  // 必要に応じてシグナルを反転させる(1の代わりに-1を返し、逆も同様)
//--- 取引シグナル値を返す
   return (res);
  }
//+------------------------------------------------------------------+
//|    新しいバーが現れると'true'                                    |
//+------------------------------------------------------------------+
bool isNewBar(const bool print_log=true)
  {
   static datetime bartime=0; // 現在のバーの開いた時刻を格納する
//--- ゼロバーの開いた時間を取得する
   datetime currbar_time=iTime(_Symbol,_Period,0);
//--- 新しいバーが到着すると開いた時刻が変わる
   if(bartime!=currbar_time)
     {
      bartime=currbar_time;
      lastbar_timeopen=bartime;
      //--- ログに新しいバーが開いている時間のデータを表示する
      if(print_log && !(MQLInfoInteger(MQL_OPTIMIZATION)||
MQLInfoInteger(MQL_TESTER)))
        {
         //--- 新しいバーを開いた時間のメッセージを表示する
         PrintFormat("%s: new bar on %s %s opened at %s",__FUNCTION__,_Symbol,
                     StringSubstr(EnumToString(_Period),7),
                     TimeToString(TimeCurrent(),TIME_SECONDS));
         //--- 最後のティックのデータを取得する
         MqlTick last_tick;
         if(!SymbolInfoTick(Symbol(),last_tick))
            Print("SymbolInfoTick() failed, error = ",GetLastError());
         //--- 最後のティックタイムをミリ秒まで表示する
         PrintFormat("Last tick was at %s.%03d",
                     TimeToString(last_tick.time,TIME_SECONDS),last_tick.time_msc%
1000);
        }
      //--- 新しいバーがある
      return (true);
     }
//--- 新しいバーがない
   return (false);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  成行価格で指定された量で買う                                    |
//+------------------------------------------------------------------+
bool Buy(double volume,ulong deviation=10,ulong  magicnumber=0)
  {
//--- 成行価格で買う
   return (MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber));
  }
//+------------------------------------------------------------------+
//|  成行価格で指定された量で売る                                    |
//+------------------------------------------------------------------+
bool Sell(double volume,ulong deviation=10,ulong  magicnumber=0)
  {
//--- 成行価格で売る
   return (MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber));
  }
//+------------------------------------------------------------------+
//|  バー数によるポジション保留時間ごとにポジションを決済する       |
//+------------------------------------------------------------------+
void ClosePositionsByBars(int holdtimebars,ulong deviation=10,ulong  magicnumber=0)
  {
   int total=PositionsTotal(); // ポジション数
//--- ポジションをすべて見る
   for(int i=total-1; i>=0; i--)
     {
      //--- ポジションパラメータ
      ulong  position_ticket=PositionGetTicket(i);
    // ポジションチケット
      string
position_symbol=PositionGetString(POSITION_SYMBOL);                    // シンボル
      ulong
 magic=PositionGetInteger(POSITION_MAGIC);                              // ポジショ
ンマジックナンバー
      datetime position_open=(datetime)
PositionGetInteger(POSITION_TIME);               // ポジションの開かれた時刻
      int bars=iBarShift(_Symbol,PERIOD_CURRENT,position_open)+1;
    // ポジションがどれだけ前(バー数)に開かれたか

      //--- マジックナンバーとシンボルが一致していてポジションがすでに長くある場合
      if(bars>holdtimebars && magic==magicnumber && position_symbol==_Symbol)
        {
         int    digits=(int)
SymbolInfoInteger(position_symbol,SYMBOL_DIGITS);          // 小数点以下桁数
         double
volume=PositionGetDouble(POSITION_VOLUME);                             // ポジションボ
リューム
         ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)
PositionGetInteger(POSITION_TYPE); // ポジションの種類
         string str_type=StringSubstr(EnumToString(type),14);
         StringToLower(str_type); // 正しいメッセージ書式設定のために小文字にする
         PrintFormat("Close position #%I64u %s %s %.2f",
```

© 2000-2025, MetaQuotes Ltd.

```
                     position_ticket,position_symbol,str_type,volume);
         //--- 注文タイプを設定し、取引リクエストを送信する
         if(type==POSITION_TYPE_BUY)
            MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber,position_ticket
);
         else
            MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber,position_ticket)
;
        }
     }
  }
//+------------------------------------------------------------------+
//|  取引リクエストを準備して送信する                                |
//+------------------------------------------------------------------+
bool MarketOrder(ENUM_ORDER_TYPE type,double volume,ulong slip,ulong
magicnumber,ulong pos_ticket=0)
  {
//--- 構造体の宣言と初期化
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(type==ORDER_TYPE_BUY)
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
//--- リクエストパラメータ
   request.action   =TRADE_ACTION_DEAL;                // 取引操作の種類
   request.position =pos_ticket;                       // 決済する場合のポジションチケッ
ト
   request.symbol   =Symbol();                         // シンボル
   request.volume   =volume;                           // ボリューム
   request.type     =type;                             // 注文の種類
   request.price    =price;                            // 取引価格
   request.deviation=slip;                             // 価格からの許容偏差
   request.magic    =magicnumber;                      // 注文のマジックナンバー
//--- リクエストを送信する
   if(!OrderSend(request,result))
     {
      //--- 失敗したらデータを表示する
      PrintFormat("OrderSend %s %s %.2f at %.5f error %d",
                  request.symbol
,EnumToString(type),volume,request.price,GetLastError());
      return (false);
     }
//--- 操作の成功を知らせる
   PrintFormat("retcode=%u  deal=%I64u  order=%
I64u",result.retcode,result.deal,result.order);
   return (true);
  }
```

参照

© 2000-2025, MetaQuotes Ltd.

Event handling functions、Program running、Client terminal events、OnTimer、OnBookEvent、OnChartEvent

© 2000–2025, MetaQuotes Ltd.

OnCalculate

この関数は、Calculateイベントが発生して価格の変化が処理されるときに、指標で呼び出されます。この関数の型は 2 つあります。1つの指標内では1つだけ使用できます。

データ配列に基づく計算

```
int  OnCalculate(
   const int         rates_total,       // price[]配列サイズ
   const int         prev_calculated,   // 以前の呼び出しで処理されたバーの数
   const int         begin,             // 意味のあるデータが始まるprice []配列のインデックス番号
   const double&     price[]            // 計算のための値の配列
   );
```

現在の時間枠に基づいた計算

```
int  OnCalculate(
   const int         rates_total,       // 入力時間枠のサイズ
   const int         prev_calculated,   // 以前の呼び出しで処理されたバーの数
   const datetime&   time[],            // 時間配列
   const double&     open[],            // 始値配列
   const double&     high[],            // 高値配列
   const double&     low[],             // 安値配列
   const double&     close[],           // 終値配列
   const long&       tick_volume[],     // ティックボリューム配列
   const long&       volume[],          // 真のボリュームの配列
   const int&        spread[]           // スプレッドの配列
   );
```

パラメータ

rates_total

[in] 計算のために指標に使用できるprice[]配列または入力系列のサイズ第2の関数タイプでは、パラメータ値は、それが開始されたチャート上の棒の数に対応します。

prev_calculated

[in] 直前の呼び出しでOnCalculate()関数によって返された値が含まれます。この関数が前回起動されて以来変更されていないバーをスキップするように設計されています。

begin

[in] price[]配列で有意義なデータが始まるインデックス。正しい値を持たない失われたデータや初期のデータをスキップすることができます。

price[]

[in] 計算のための値の配列。価格時系列または指標バッファのうちの一つがprice[]配列として渡されます。計算に渡されるデータの種類は_AppliedTo事前定義変数を使用して定義できます。

time[]

[in] バーの開いた時間の配列

open[]

[in] バーの始値の配列

© 2000-2025, MetaQuotes Ltd.

high[]
[in] バーの高値の配列

low[]
[in] バーの安値の配列

close[]
[in] バーの終値の配列

tick_volume[]
[in] ティックボリューム値の配列

volume[]
[in] 取引量値の配列

spread[]
[in] バーのスプレッド値の配列

戻り値

次に関数が呼び出されるときに*prev_calculated*パラメータとして渡されるint型の値

注意事項

OnCalculate()関数がゼロに等しい場合、クライアント端末のデータウィンドウには指標値が表示されません。

OnCalculate()関数の最後の呼び出し以降に価格データが変更された場合（より長期間の履歴が読み込まれているか、履歴のギャップが埋められている場合）、 *prev_calculated*入力パラメータは、端末自身によってゼロに設定されます。

time[]、*open[]*、*high[]*、*low[]*、*close[]*、*tick_volume[]*、*volume[]*、*spread[]*配列で索引付けの方向を定義するには、ArrayGetAsSeries()関数を呼び出します。デフォルトに依存しないためには、作業する配列でArraySetAsSeries()関数を呼び出します。

最初の関数タイプを使用する場合、指標を起動するときに必要な時系列または指標がパラメータタブのprice[]配列としてユーザによって選択されます。そのためには "Apply to"（適用）フィールドのドロップダウンリストに必要な項目を指定する必要があります。

カスタム指標値を他のMQL5プログラムから取得するにはiCustom()関数が使用されます。これは、後の操作のために指標ハンドルを返します。また、必要な*price []*配列または他の指標ハンドルを指定することも出来ます。このパラメータは、カスタム指標の入力変数のリストに最後に送信されるべきです。

OnCalculate()の戻り値と*prev_calculated*　の2番目の入力パラメータとの関係はここで指摘される必要があります。関数が呼び出されたとき、*prev_calculated*パラメータは以前のOnCalculate()呼び出しで返された値を持ちます。これにより、この関数が前回実行されてから変更されていないバーの繰り返した計算を避けてカスタム指標を計算するためのリソース節約アルゴリズムを実装することができます。

指標の例

```
//+------------------------------------------------------------------+
//|                                          OnCalculate_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
```

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Sample Momentum indicator calculation"

//---- 指標の設定
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
#property indicator_type1   DRAW_LINE
#property indicator_color1  Blue
//---- 入力
input int MomentumPeriod=14; // 計算期間
//---- 指標バッファ
double    MomentumBuffer[];
//--- 計算期間を格納するグローバル変数
int       IntPeriod;
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- 入力パラメータを確認する
   if(MomentumPeriod<0)
     {
      IntPeriod=14;
      Print("Period parameter has an incorrect value. The following value is to be
used for calculations ",IntPeriod);
     }
   else
      IntPeriod=MomentumPeriod;
//---- バッファ
   SetIndexBuffer(0,MomentumBuffer,INDICATOR_DATA);
//---- データウィンドウとサブウィンドウで表示される指標名
   IndicatorSetString(INDICATOR_SHORTNAME,"Momentum"+"("+string(IntPeriod)+")");
//--- 描画が始まるバーのインデックスを設定する
   PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,IntPeriod-1);
//--- 描画されない空の値を0.0に設定する
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0.0);
//--- 表示される指標の精度
   IndicatorSetInteger(INDICATOR_DIGITS,2);
  }
//+------------------------------------------------------------------+
//|  モメンタム指標計算                                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,      // price[]配列サイズ
                const int prev_calculated,  // 以前に処理されたバーの数
                const int begin,            // 有意義なデータの始まり
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &price[])      // 処理される値の配列
  {
//--- 初期の計算位置
   int StartCalcPosition=(IntPeriod-1)+begin;
//---- 計算データが不十分の場合
   if(rates_total<StartCalcPosition)
      return(0);  // ゼロ値で終了 - 指標は未計算
//--- 正しい描画を開始する
   if(begin>0)
      PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,StartCalcPosition+(IntPeriod-1));
//--- 計算を開始し、開始位置を定義する
   int pos=prev_calculated-1;
   if(pos<StartCalcPosition)
      pos=begin+IntPeriod;
//--- 計算のメインループ
   for(int i=pos;i<rates_total && !IsStopped();i++)
      MomentumBuffer[i]=price[i]*100/price[i-IntPeriod];
//--- OnCalculateの実行が完了したので、後に続く呼び出しのために新しいprev_calculated値を返す
   return(rates_total);
  }
```

参照

ArrayGetAsSeries、ArraySetAsSeries、iCustom、Event handling functions、Program running、Client terminal events、Access to timeseries and indicators

© 2000-2025, MetaQuotes Ltd.

OnTimer

この関数は、一定の時間間隔で端末によって生成されるTimerイベント中に、EAで呼び出されます。

```
void  OnTimer(void);
```

戻り値

なし

注意事項

Timerイベントは、EventSetTimer()関数を使用してタイマーを起動したEAのクライアント端末によって定期的に生成されます。. 通常この関数はOnInit()関数で呼び出されます。EAの動作が止まった場合、タイマーはEventKillTimer()を使用して破壊されるべきです。これは通常OnDeinit()関数で呼び出されます。

それぞれのエキスパートアドバイザーと指標は独自のタイマーを操作しそのタイマーのみからイベントを受信します。MQL5 プログラムが停止した場合、作成されて EventKillTimer() 関数で無効にされていないタイマーは強制的に破壊されます。

タイマーイベントを1秒間に1回よりも頻繁に受信する必要がある場合は、高解像度タイマーを作成するためにEventSetMillisecondTimer()を使用します。

一般的に、タイマー期間が減少する際にはタイマーイベントのハンドラがより頻繁に呼び出されるようにテスト時間が増加します。リアルタイムモードで操作している場合、タイマーイベントは、ハードウェアの制約のために10～16ミリ秒に1回以下で生成されます。

1 つのプログラムでは 1 つのタイマーのみが起動出来ます。各MQL5アプリケーションとチャートは、新たに到着したイベントが配置される独自のイベントキューを持っています。キューにすでにタイマーイベントがある、またはこのイベントが処理段階にある場合、新しいタイマーイベントは MQL5 アプリケーションキューには追加されません。

OnTimer()ハンドラーを持つEAの例

```
//+------------------------------------------------------------------+
//|                                                OnTimer_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Example of using the timer for calculating the trading server time"
#property description "It is recommended to run the EA at the end of a trading week before the weekend"
//+------------------------------------------------------------------+
//| エキスパート初期化関数                                           |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 1秒の周期でタイマーを作成する
   EventSetTimer(1);
```

© 2000–2025, MetaQuotes Ltd.

```
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| エキスパート初期化解除に使用される関数                                |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- 作業終了後にタイマーを破壊する
   EventKillTimer();

  }
//+------------------------------------------------------------------+
//| エキスパートティック関数                                           |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| Timer 関数                                                       |
//+------------------------------------------------------------------+
void OnTimer()
  {
//--- OnTimer()の最初の呼び出しの時刻
   static datetime start_time=TimeCurrent();
//--- 最初のOnTimer() 呼び出し中の取引サーバ時刻
   static datetime start_tradeserver_time=0;
//--- 計算された取引サーバ時刻
   static datetime calculated_server_time=0;
//--- ローカルPC時刻
   datetime local_time=TimeLocal();
//--- 現在の推測取引サーバ時刻
   datetime trade_server_time=TimeTradeServer();
//--- 何らかの理由でサーバの時刻が不明な場合は、先に終了する
   if(trade_server_time==0)
      return;
//--- 取引サーバ初期値がまだ設定されていない場合
   if(start_tradeserver_time==0)
     {
      start_tradeserver_time=trade_server_time;
      //--- 取引サーバの計算された現在時刻を設定する
      Print(trade_server_time);
      calculated_server_time=trade_server_time;
     }
   else
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- OnTimer()の初めの呼び出しの時刻を増やす
      if(start_tradeserver_time!=0)
         calculated_server_time=calculated_server_time+1;;
     }
//---
   string com=StringFormat("                  Start time: %
s\r\n",TimeToString(start_time,TIME_MINUTES|TIME_SECONDS));
   com=com+StringFormat("                   Local time: %
s\r\n",TimeToString(local_time,TIME_MINUTES|TIME_SECONDS));
   com=com+StringFormat("TimeTradeServer time: %
s\r\n",TimeToString(trade_server_time,TIME_MINUTES|TIME_SECONDS));
   com=com+StringFormat(" EstimatedServer time: %
s\r\n",TimeToString(calculated_server_time,TIME_MINUTES|TIME_SECONDS));
//--- チャート上のすべてのカウンタの値を表示する
   Comment(com);
  }
```

参照

EventSetTimer、EventSetMillisecondTimer、EventKillTimer、GetTickCount、GetMicrosecondCount、Client terminal events

© 2000–2025, MetaQuotes Ltd.

OnTrade

この関数はTradeイベントが発生するときにEAで呼び出され、注文、ポジション、取引リストの変更を処理するためのものです。

```
void  OnTrade(void);
```

戻り値

なし

注意事項

OnTrade()はエキスパートアドバイザーでのみ呼び出されます。同じ名前と型の関数をそこに追加しても、指標やスクリプトには使用されません

指値注文の実行、ポジションの開閉、ストップの設定、指値注文の発動などの取引操作によって、注文および取引の履歴、および/またはポジションおよび現在の注文のリストが適切に変更されます。

注文を処理する際、取引サーバは、着信されるTrade イベントに関するメッセージを端末に送信します。履歴から注文と取引に関する関連データを検索するには、まずHistorySelect()を使用して取引履歴リクエストを実行する必要があります。

取引イベントは、次の場合にサーバによって生成されます。

- アクティブ注文の変更
- ポジションの変更
- 約定の変更
- 取引履歴の変更

それぞれの Tradeイベントは、1つまたは複数の取引要求の結果として表示されることがあります。取引リクエストは、 OrderSend()またはOrderSendAsync()を使用してサーバに送信されます。それぞれのリクエストは、いくつかの取引イベントにつながる可能性があります。イベントの処理はいくつかの段階で実行されることがあり、各操作によって注文、ポジション、取引履歴の状態が変更される可能性があるため、「1つのリクエスト – 1つの取引イベント」という声明に頼ることはできません。

OnTrade()ハンドラーはOnTradeTransaction()が呼び出された後で呼び出されます。一般に、OnTrade()とOnTradeTransaction()の呼び出しの数には正確な相関関係はありません。1つのOnTrade()コールは、1つまたは複数のOnTradeTransactionコールに対応します。

OnTrade()を持つEAの例

```
//+------------------------------------------------------------------+
//|                                                OnTrade_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

input    int days=7;             // 取引履歴の日数
//--- グローバルスコープで取引履歴の上限を設定する
```

© 2000–2025, MetaQuotes Ltd.

```
datetime      start;              // キャッシュ内の取引履歴の開始日
datetime      end;                // キャッシュ内の取引履歴の終了日
//--- グローバルカウンタ
int           orders;             // アクティブ注文数
int           positions;          // ポジション数
int           deals;              // 取引履歴キャッシュ内の約定数
int           history_orders;     // 取引履歴キャッシュ内の注文数
bool          started=false;      // カウンタ有効性のフラグ

//+------------------------------------------------------------------+
//| エキスパート初期化関数                                           |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   end=TimeCurrent();
   start=end-days*PeriodSeconds(PERIOD_D1);
   PrintFormat("Limits of the history to be loaded: start - %s, end - %s",
               TimeToString(start),TimeToString(end));
   InitCounters();
//---
   return(0);
  }
//+------------------------------------------------------------------+
//| ポジション、注文、取引カウンタの初期化                           |
//+------------------------------------------------------------------+
void InitCounters()
  {
   ResetLastError();
//--- 履歴を読み込む
   bool selected=HistorySelect(start,end);
   if(!selected)
     {
      PrintFormat("%s. Failed to load history from %s to %s to cache. Error code: %
d",
                  __FUNCTION__
,TimeToString(start),TimeToString(end),GetLastError());
      return;
     }
//--- 現在値を取得する
   orders=OrdersTotal();
   positions=PositionsTotal();
   deals=HistoryDealsTotal();
   history_orders=HistoryOrdersTotal();
   started=true;
   Print("Counters of orders, positions and deals successfully initialized");
  }
//+------------------------------------------------------------------+
//| エキスパートティック関数                                         |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnTick()
  {
   if(started) SimpleTradeProcessor();
   else InitCounters();
  }
//+------------------------------------------------------------------+
//| Tradeイベントが到着すると呼び出される                                |
//+------------------------------------------------------------------+
void OnTrade()
  {
   if(started) SimpleTradeProcessor();
   else InitCounters();
  }
//+------------------------------------------------------------------+
//| 取引と履歴の変更を処理する例                                         |
//+------------------------------------------------------------------+
void SimpleTradeProcessor()
  {
   end=TimeCurrent();
   ResetLastError();
//--- 指定された間隔でプログラムキャッシュに取引履歴をダウンロードする
   bool selected=HistorySelect(start,end);
   if(!selected)
     {
      PrintFormat("%s. Failed to load history from %s to %s to cache. Error code: %
d",
                  __FUNCTION__
,TimeToString(start),TimeToString(end),GetLastError());
      return;
     }
//--- 現在値を取得する
   int curr_orders=OrdersTotal();
   int curr_positions=PositionsTotal();
   int curr_deals=HistoryDealsTotal();
   int curr_history_orders=HistoryOrdersTotal();
//--- アクティブ注文の数が変更されているかどうかを確認する
   if(curr_orders!=orders)
     {
      //--- アクティブ注文の数が変更されている
      PrintFormat("Number of orders has been changed. Previous value is %d, current
value is %d",
                  orders,curr_orders);
      //--- 値を更新する
      orders=curr_orders;
     }
//--- ポジション数の変更
   if(curr_positions!=positions)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- ポジション数が変更されている
      PrintFormat("Number of positions has been changed. Previous value is %d,
current value is %d",
                  positions,curr_positions);
      //--- 値を更新する
      positions=curr_positions;
     }
//--- 取引履歴キャッシュ内の約定数の変更
   if(curr_deals!=deals)
     {
      //--- 取引履歴キャッシュ内の約定数が変更されている
      PrintFormat("Number of deals has been changed. Previous value is %d, current
value is %d",
                  deals,curr_deals);
      //--- 値を更新する
      deals=curr_deals;
     }
//--- 取引履歴キャッシュ内の履歴注文数の変更
   if(curr_history_orders!=history_orders)
     {
      //--- 取引履歴キャッシュ内の履歴注文数が変更されている
      PrintFormat("Number of orders in history has been changed. Previous value is %
d, current value is %d",
                  history_orders,curr_history_orders);
     //--- 値を更新する
     history_orders=curr_history_orders;
     }
//--- キャッシュ内でリクエストされる取引履歴の上限を変更する必要があるかどうかを確認する
   CheckStartDateInTradeHistory();
  }
//+------------------------------------------------------------------+
//|   取引履歴リクエストの開始日の変更                                |
//+------------------------------------------------------------------+
void CheckStartDateInTradeHistory()
  {
//--- 今すぐ作業を始めると仮定した場合の最初の間隔、
   datetime curr_start=TimeCurrent()-days*PeriodSeconds(PERIOD_D1);
//--- 取引履歴の開始制限が意図された日から1日以上経過して
//--- いないことを確認する
   if(curr_start-start>PeriodSeconds(PERIOD_D1))
     {
      //--- キャッシュに読み込まれる履歴の開始日を修正する
      start=curr_start;
      PrintFormat("New start limit of the trade history to be loaded: start => %s",
                  TimeToString(start));
      //---更新された期間の取引履歴を再度読み込む
      HistorySelect(start,end);
      //--- さらなる比較のために約定および注文カウンタを修正する
      history_orders=HistoryOrdersTotal();
```

© 2000-2025, MetaQuotes Ltd.

```
      deals=HistoryDealsTotal();
     }
  }
//+------------------------------------------------------------------+
/* 出力例:
  Limits of the history to be loaded: start - 2018.07.16 18:11, end - 2018.07.23 18:11
  The counters of orders, positions and deals are successfully initialized
  Number of orders has been changed. Previous value 0, current value 1
  Number of orders has been changed. Previous value 1, current value 0
  Number of positions has been changed. Previous value 0, current value 1
  Number of deals has been changed. Previous value 0, current value 1
  Number of orders in the history has been changed. Previous value 0, current value 1
*/
```

参照

OrderSend、OrderSendAsync、OnTradeTransaction、Client terminal events

© 2000–2025, MetaQuotes Ltd.

OnTradeTransaction

この関数はTradeTransactionイベントが発生するときにEAで呼び出され、取引リクエストの実行結果を処理するためのものです。

```
void  OnTradeTransaction()
   const MqlTradeTransaction&    trans,     // 取引トランザクション構造体
   const MqlTradeRequest&        request,   // リクエスト構造体
   const MqlTradeResult&         result     // 返答構造体
   );
```

パラメータ

trans

[in] MqlTradeTransaction取引口座で行われた取引を記述する型変数

request

[in] MqlTradeRequestトランザクションにつながった取引リクエストを記述する型変数で、TRADE_TRANSACTION_REQUEST型トランザクション値のみを含む

result

[in] MqlTradeResultトランザクションにつながった取引リクエストの実行結果を含む型変数で、TRADE_TRANSACTION_REQUEST型トランザクション値のみを含む

戻り値

なし

注意事項

OnTradeTransaction()は、下記の場合に取引サーバが端末に送信したTradeTransactionイベントを処理するために呼び出されます。

- OrderSend()/OrderSendAsync()関数を使ったMQL5プログラムからの取引リクエストの送信とそれに続く実行
- GUIを介した手動での取引リクエストの送信とそれに続く実行
- サーバでの指値注文やストップ注文のアクティベーション
- 取引サーバ側での操作の実行

トランザクション型のデータは*trans*変数の*type*フィールドに含まれています。トランザクションの型はENUM_TRADE_TRANSACTION_TYPE列挙体に記述されています。

- TRADE_TRANSACTION_ORDER_ADD – 新規アクティブ注文の追加
- TRADE_TRANSACTION_ORDER_UPDATE – 既存注文の変更
- TRADE_TRANSACTION_ORDER_DELETE – アクティブ注文リストからの注文の削除
- TRADE_TRANSACTION_DEAL_ADD – 履歴への約定の追加
- TRADE_TRANSACTION_DEAL_UPDATE – 履歴での約定の変更
- TRADE_TRANSACTION_DEAL_DELETE – 履歴での約定の削除
- TRADE_TRANSACTION_HISTORY_ADD – 実行またはキャンセルの結果としての履歴への注文の追加
- TRADE_TRANSACTION_HISTORY_UPDATE – 注文履歴内の注文の変更
- TRADE_TRANSACTION_HISTORY_DELETE – 注文履歴内の注文の削除
- TRADE_TRANSACTION_POSITION – 取引の実行に関連していないポジション変更
- TRADE_TRANSACTION_REQUEST – 取引リクエストがサーバで処理され結果が受け取られたことの通知

© 2000–2025, MetaQuotes Ltd.

TRADE_TRANSACTION_REQUEST型のトランザクションを処理するには、OnTradeTransaction()関数の第2及び第3パラメータ(*request*、*result*)を分析し追加情報を受け取る必要があります。

買い取引リクエストを送信すると、取引口座では下記の一連の取引トランザクションが発生します。1)処理リクエストの受領、2)口座に対した適切な注文書の作成、3)注文実行、4) 実行された注文をアクティブ注文のリストから削除、5)同注文を注文履歴に追加、6)後続のトランザクションを履歴に追加、7)新しいポジションを作成これらの段階はすべて取引トランザクションです。このようなトランザクションの端末への到着がTradeTransactionイベントです。端末におけるこれらのトランザクションの到着の優先順位は保証されません。よって取引アルゴリズムを開発する際にトランザクション到着の順番を仮定すべきではありません。

EAのOnTradeTransaction()ハンドラで取引トランザクションを処理する場合、端末は新たに到着した取引トランザクションの処理を継続します。したがって、取引口座の状態は、OnTradeTransaction()操作の過程で変更される可能性があります。例えば、MQL5 プログラムが新しい注文の追加を処理している間に、注文が実行されて注文のリストから削除され、履歴に移動されることもあります。プログラムは、これらのイベントすべてを通知されます。

トランザクションキューは1,024要素を備えます。OnTradeTransaction()ハンドラーが新しいトランザクションを処理するのに長くかかりすぎると、代わりにキュー内の新しいトランザクションが処理されることがあります。

OnTrade()ハンドラーは適切なOnTradeTransaction()が呼び出された後で呼び出されます。一般に、OnTrade()とOnTradeTransaction()の呼び出しの数には正確な相関関係はありません。1つのOnTrade()コールは、1つまたは複数のOnTradeTransactionコールに対応します。

それぞれの<t1> </t1><li2>Trade</li2><t3>イベントは、1つまたは複数の取引要求の結果として表示されることがあります。取引リクエストは、OrderSend()またはOrderSendAsync()を使用してサーバに送信されます。それぞれのリクエストは、いくつかの取引イベントにつながる可能性があります。イベントの処理はいくつかの段階で実行されることがあり、各操作によって注文、ポジション、取引履歴の状態が変更される可能性があるため、「1つのリクエスト -1つの取引イベント」という声明に頼ることはできません。

OnTradeTransaction()ハンドラーを持つEAの例

```
//+------------------------------------------------------------------+
//|                                    OnTradeTransaction_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Sample listener of TradeTransaction events"
//+------------------------------------------------------------------+
//| エキスパート初期化関数                                           |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   PrintFormat("LAST PING=%.f ms",
               TerminalInfoInteger(TERMINAL_PING_LAST)/1000.);
//---
   return(INIT_SUCCEEDED);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| エキスパートティック関数                                         |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| TradeTransaction 関数                                            |
//+------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction &trans,
                        const MqlTradeRequest &request,
                        const MqlTradeResult &result)
  {
//---
   static int counter=0;   // OnTradeTransaction() 呼び出しのカウンタ
   static uint lasttime=0; // OnTradeTransaction() の最後の呼び出し時刻
//---
   uint time=GetTickCount();
//--- 最後のトランザクションが 1 秒以上前に実行された場合
   if(time-lasttime>1000)
     {
      counter=0; // これは新しい取引であり、カウンタをリセットできる
      if(IS_DEBUG_MODE)
         Print(" New trade operation");
     }
   lasttime=time;
   counter++;
   Print(counter,". ",__FUNCTION__);
//--- 取引リクエストの実行結果
   ulong             lastOrderID   =trans.order;
   ENUM_ORDER_TYPE   lastOrderType =trans.order_type;
   ENUM_ORDER_STATE  lastOrderState=trans.order_state;
//--- トランザクションが実行される取引シンボルの名称
   string trans_symbol=trans.symbol;
//--- トランザクションの種類
   ENUM_TRADE_TRANSACTION_TYPE  trans_type=trans.type;
   switch(trans.type)
     {
      case  TRADE_TRANSACTION_POSITION:    // ポジション修正
        {
         ulong pos_ID=trans.position;
         PrintFormat("MqlTradeTransaction: Position  #%I64u %s modified: SL=%.5f TP=
%.5f",
                     pos_ID,trans_symbol,trans.price_sl,trans.price_tp);
        }
      break;
```

© 2000-2025, MetaQuotes Ltd.

```
      case TRADE_TRANSACTION_REQUEST:      // 取引リクエストの送信
         PrintFormat("MqlTradeTransaction: TRADE_TRANSACTION_REQUEST");
         break;
      case TRADE_TRANSACTION_DEAL_ADD:     // 取引の追加
        {
         ulong           lastDealID   =trans.deal;
         ENUM_DEAL_TYPE lastDealType =trans.deal_type;
         double         lastDealVolume=trans.volume;
         //--- 外部システムの取引ＩＤ - 取引所で割り当てられたチケット
         string Exchange_ticket="";
         if(HistoryDealSelect(lastDealID))
            Exchange_ticket=HistoryDealGetString(lastDealID,DEAL_EXTERNAL_ID);
         if(Exchange_ticket!="")
            Exchange_ticket=StringFormat("(Exchange deal=%s)",Exchange_ticket);

         PrintFormat("MqlTradeTransaction: %s deal #%I64u %s %s %.2f lot   %
s",EnumToString(trans_type),
                     lastDealID,EnumToString(lastDealType),trans_symbol,lastDealVolu
me,Exchange_ticket);
         }
      break;
      case TRADE_TRANSACTION_HISTORY_ADD: // 履歴への注文の追加
        {
         //--- 外部システムの注文ＩＤ - 取引所で割り当てられたチケット
         string Exchange_ticket="";
         if(lastOrderState==ORDER_STATE_FILLED)
           {
            if(HistoryOrderSelect(lastOrderID))
               Exchange_ticket=HistoryOrderGetString(lastOrderID,ORDER_EXTERNAL_ID);
            if(Exchange_ticket!="")
               Exchange_ticket=StringFormat("(Exchange ticket=%s)",Exchange_ticket);
           }
         PrintFormat("MqlTradeTransaction: %s order #%I64u %s %s %s   %
s",EnumToString(trans_type),
                     lastOrderID,EnumToString(lastOrderType),trans_symbol,EnumToStri
ng(lastOrderState),Exchange_ticket);
         }
      break;
      default: // 他のトランザクション
        {
         //--- 外部システムの注文ＩＤ - 取引所で割り当てられたチケット
         string Exchange_ticket="";
         if(lastOrderState==ORDER_STATE_PLACED)
           {
            if(OrderSelect(lastOrderID))
               Exchange_ticket=OrderGetString(ORDER_EXTERNAL_ID);
            if(Exchange_ticket!="")
               Exchange_ticket=StringFormat("Exchange ticket=%s",Exchange_ticket);
           }
```

© 2000-2025, MetaQuotes Ltd.

```
         PrintFormat("MqlTradeTransaction: %s order #%I64u %s %s   %
s",EnumToString(trans_type),
                     lastOrderID,EnumToString(lastOrderType),EnumToString(lastOrderS
tate),Exchange_ticket);
        }
      break;
     }
//--- 注文チケット
   ulong orderID_result=result.order;
   string retcode_result=GetRetcodeID(result.retcode);
   if(orderID_result!=0)
      PrintFormat("MqlTradeResult: order #%d retcode=%s
",orderID_result,retcode_result);
//---
  }
//+------------------------------------------------------------------+
//|  数値をテキスト文字列に変換する                                  |
//+------------------------------------------------------------------+
string GetRetcodeID(int retcode)
  {
   switch(retcode)
     {
      case 10004: return("TRADE_RETCODE_REQUOTE");             break;
      case 10006: return("TRADE_RETCODE_REJECT");              break;
      case 10007: return("TRADE_RETCODE_CANCEL");              break;
      case 10008: return("TRADE_RETCODE_PLACED");              break;
      case 10009: return("TRADE_RETCODE_DONE");                break;
      case 10010: return("TRADE_RETCODE_DONE_PARTIAL");        break;
      case 10011: return("TRADE_RETCODE_ERROR");               break;
      case 10012: return("TRADE_RETCODE_TIMEOUT");             break;
      case 10013: return("TRADE_RETCODE_INVALID");             break;
      case 10014: return("TRADE_RETCODE_INVALID_VOLUME");      break;
      case 10015: return("TRADE_RETCODE_INVALID_PRICE");       break;
      case 10016: return("TRADE_RETCODE_INVALID_STOPS");       break;
      case 10017: return("TRADE_RETCODE_TRADE_DISABLED");      break;
      case 10018: return("TRADE_RETCODE_MARKET_CLOSED");       break;
      case 10019: return("TRADE_RETCODE_NO_MONEY");            break;
      case 10020: return("TRADE_RETCODE_PRICE_CHANGED");       break;
      case 10021: return("TRADE_RETCODE_PRICE_OFF");           break;
      case 10022: return("TRADE_RETCODE_INVALID_EXPIRATION");  break;
      case 10023: return("TRADE_RETCODE_ORDER_CHANGED");       break;
      case 10024: return("TRADE_RETCODE_TOO_MANY_REQUESTS");   break;
      case 10025: return("TRADE_RETCODE_NO_CHANGES");          break;
      case 10026: return("TRADE_RETCODE_SERVER_DISABLES_AT");  break;
      case 10027: return("TRADE_RETCODE_CLIENT_DISABLES_AT");  break;
      case 10028: return("TRADE_RETCODE_LOCKED");              break;
      case 10029: return("TRADE_RETCODE_FROZEN");              break;
      case 10030: return("TRADE_RETCODE_INVALID_FILL");        break;
      case 10031: return("TRADE_RETCODE_CONNECTION");          break;
```

© 2000–2025, MetaQuotes Ltd.

```
      case 10032: return("TRADE_RETCODE_ONLY_REAL");           break;
      case 10033: return("TRADE_RETCODE_LIMIT_ORDERS");        break;
      case 10034: return("TRADE_RETCODE_LIMIT_VOLUME");        break;
      case 10035: return("TRADE_RETCODE_INVALID_ORDER");       break;
      case 10036: return("TRADE_RETCODE_POSITION_CLOSED");     break;
      default:
         return("TRADE_RETCODE_UNKNOWN="+IntegerToString(retcode));
         break;
     }
//---
  }
```

参照

OrderSend、OrderSendAsync、OnTradeTransaction、Trade request structure、Trade transaction structure、Trade transaction types、Trade operation types、Client terminal events

© 2000–2025, MetaQuotes Ltd.

OnBookEvent

この関数は、BookEventイベントが発生するときに指標やEAで呼び出され、板情報の変化を処理するために設計されています。

```
void  OnBookEvent(
   const string&  symbol          // 銘柄
   );
```

パラメータ

symbol

[in] BookEventが到着した銘柄の名称

戻り値

なし

注意事項

任意のシンボルのBookEventイベントを受け取るには、MarketBookAdd() 関数を使用してこのシンボルのBookEventイベント受信を事前に設定する必要があります。指定されたシンボルのBookEventへの購読を停止するにはMarketBookRelease()関数を呼ぶ必要があります。

BookEventはすべてのチャートにブロードキャストされます。これは、1 つのアプリケーションがMarketBookAddを使用してBookEventイベントの受信に購読した場合、同じチャートにある他のすべてのOnBookEvent() ハンドラを持っている指標とEAがこのイベントを受け取ることを意味します。よって、*symbol*パラメータとしてOnBookEvent()ハンドラに渡された銘柄名を分析する必要があります。

同じチャートで実行されているすべてのアプリケーションについて銘柄でソートされた個別のBookEventカウンターが提供されます。これは、各チャートが異なるシンボルに対して複数のサブスクリプションを持つことができ、各シンボルに対してカウンタが提供されることを意味します。BookEventを購読および購読解除すると、指定したシンボルの購読カウンタが1つのチャート内でのみ変更されます。言い換えれば、同じシンボルであるにもかかわらず購読カウンタの値が異なるBookEventには、2つの隣接するチャートが存在する可能性があります。

購読カウンタの初期値はゼロです。MarketBookAdd()が呼び出されるごとに、チャート上の指定されたシンボルの購読カウンタが1つ増えます（チャートシンボルと MarketBookAdd()内のシンボルは一致する必要はありません）。MarketBookRelease()を呼び出すと、チャート内の特定のシンボルに対する購読カウンタが1つ減少します。任意のシンボルのBookEventイベントは、カウンタが0になるまでチャート内でブロードキャストされます。したがって、MarketBookAdd()の呼び出しを含むMQL5プログラムが全て、動作後にMarketBookRelease()を使用して正しくイベントの購読を解除することが重要です。これを達成するためには、MQL5プログラムの全体のライフタイムでのMarketBookAdd()とMarketBookRelease()の呼び出し数は偶数であるべきです。プログラム内でフラグやカスタム購読カウンタを使用することで、BookEventイベントを安全に扱うことができ、購読を無効にして、同じイベントのサードパーティプログラムでイベントを取得することを防ぐことができます。

BookEventイベントはスキップされることはなく、前のBookEventの処理がまだ終わっていなくても、常にキューに入れられます。

例

```
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|                                                OnBookEvent_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com/en/articles/2635"
#property version   "1.00"
#property description "Example of measuring the market depth refresh rate using OnBookEvent()"
#property description "The code is taken from the article https://www.mql5.com/en/articles/2635"
//--- 入力パラメータ
input ulong ExtCollectTime    =30;  // 秒単位でのテスト時間
input ulong ExtSkipFirstTicks=10;  // 初めに抜かされるティックの数
//--- BookEvent イベント購読のフラグ
bool book_subscribed=false;
//--- 板情報からリクエストを受け取る配列
MqlBookInfo  book[];
//+------------------------------------------------------------------+
//| エキスパート初期化関数                                           |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 初めを表示する
   Comment(StringFormat("Waiting for the first %I64u ticks to arrive",ExtSkipFirstTicks));
   PrintFormat("Waiting for the first %I64u ticks to arrive",ExtSkipFirstTicks);
//--- 板情報ブロードキャストを有効にする
   if(MarketBookAdd(_Symbol))
     {
      book_subscribed=true;
      PrintFormat("%s: MarketBookAdd(%s) function returned true",__FUNCTION__,_Symbol);
     }
   else
      PrintFormat("%s: MarketBookAdd(%s) function returned false!GetLastError()=%d",__FUNCTION__,_Symbol,GetLastError());
//--- 初期化が正常に完了
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| エキスパートの初期化を解除する                                   |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- 初期化解除の理由コードを表示する
   Print(__FUNCTION__,": Deinitialization reason code = ",reason);
//--- 板情報イベント受信の購読を取り消す
   if(book_subscribed)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       if(!MarketBookRelease(_Symbol))
          PrintFormat("%s: MarketBookRelease(%s) returned false!GetLastError()=%
d",_Symbol,GetLastError());
       else
          book_subscribed=false;
      }
//---
  }
//+------------------------------------------------------------------+
//| BookEvent関数                                                    |
//+------------------------------------------------------------------+
void OnBookEvent(const string &symbol)
  {
   static ulong starttime=0;              // テスト開始時刻
   static ulong tickcounter=0;            // 板情報更新カウンタ
//--- 板情報イベントを扱うのは、自分たちで購読している場合のみである
   if(!book_subscribed)
      return;
//--- 特定シンボルのみの更新を数える
   if(symbol!=_Symbol)
      return;
//--- 最初のティックをスキップしてキューをクリアする
   tickcounter++;
   if(tickcounter<ExtSkipFirstTicks)
      return;
//--- 開始時刻を記憶する
   if(tickcounter==ExtSkipFirstTicks)
      starttime=GetMicrosecondCount();
//--- 板情報データをリクエストする
   MarketBookGet(symbol,book);
//--- 終了時
   ulong endtime=GetMicrosecondCount()-starttime;
   ulong ticks  =1+tickcounter-ExtSkipFirstTicks;
// テストの開始からどれくらいの時間が経過したか
   if(endtime>ExtCollectTime*1000*1000)
     {
      PrintFormat("%I64u ticks for %.1f seconds: %.1f ticks/sec
",ticks,endtime/1000.0/1000.0,ticks*1000.0*1000.0/endtime);
      ExpertRemove();
      return;
     }
//--- カウンタをコメント欄に表示する
   if(endtime>0)
      Comment(StringFormat("%I64u ticks for %.1f seconds: %.1f ticks/sec
",ticks,endtime/1000.0/1000.0,ticks*1000.0*1000.0/endtime));
  }
```

© 2000-2025, MetaQuotes Ltd.

参照

MarketBookAdd、MarketBookRelease、MarketBookGet、OnTrade、OnTradeTransaction、OnTick、イベント処理関数、プログラムの実行、クライエント端末イベント

© 2000–2025, MetaQuotes Ltd.

OnChartEvent

この関数はChartEventイベントが発生するときに指標やEAで呼び出されます。この関数は、ユーザまたはMQL5プログラムによって行われたチャートの変更を処理するためのものです。

```
void  OnChartEvent()
   const int        id,        // イベントID
   const long&      lparam,    // long型イベントパラメータ
   const double&    dparam,    // double型イベントパラメータ
   const string&    sparam     // string型イベントパラメータ
   );
```

パラメータ

id

[in] ENUM_CHART_EVENT列挙体からのイベントID

lparam

[in] long型イベントパラメータ

dparam

[in] double型イベントパラメータ

sparam

[in] string型イベントパラメータ

戻り値

なし

注意事項

事前定義されたOnChartEvent()関数を使用して処理出来るイベントには11種類あります。CHARTEVENT_CUSTOMからCHARTEVENT_CUSTOM_LASTまでの65535のIDはカスタムイベントのために提供されています。カスタムイベントの作成にはEventChartCustom()関数が使用されます。

ENUM_CHART_EVENT列挙体からの短いイベントの記述

- CHARTEVENT_KEYDOWN – チャートウィンドウがフォーカスされている時のキー押下
- CHARTEVENT_MOUSE_MOVE – マウスの移動、マウスボタンクリック（チャートのCHART_EVENT_MOUSE_MOVE=trueの場合）
- CHARTEVENT_OBJECT_CREATE – グラフィックオブジェクトの作成（チャートのCHART_EVENT_OBJECT_CREATE=trueの場合）
- CHARTEVENT_OBJECT_CHANGE – プロパティダイアログを介してのオブジェクトプロパティ変更
- CHARTEVENT_OBJECT_DELETE – グラフィックオブジェクト削除（チャートのCHART_EVENT_OBJECT_DELETE=trueの場合）
- CHARTEVENT_CLICK – チャートのクリック
- CHARTEVENT_OBJECT_CLICK – チャートに属するグラフィックオブジェクトでのマウスクリック
- CHARTEVENT_OBJECT_DRAG – マウスを使用してのグラフィカルオブジェクトのドラッグ
- CHARTEVENT_OBJECT_ENDEDIT – グラフィックオブジェクトの入力ボックスのテキスト編集（OBJ_EDIT）の完成
- CHARTEVENT_CHART_CHANGE – チャート変更

© 2000–2025, MetaQuotes Ltd.

- CHARTEVENT_CUSTOM+n — カスタムイベントID（nは0～65535）CHARTEVENT_CUSTOM_LAST — カスタムイベントの最大の使用可能なID（CHARTEVENT_CUSTOM +65535）

すべてのMQL5プログラムは、アプリケーションのメインスレッド以外のスレッドで動作します。メインのアプリケーションスレッドは、すべてのWindowsシステムメッセージを処理し、この処理の結果、独自のアプリケーション用のWindowsメッセージを生成します。たとえば、チャート上でマウスを動かすと（WM_MOUSE_MOVEイベント）、後のアプリケーションウインドウでのレンダリングのためにいくつかのシステムメッセージが生成され、内部メッセージがチャート上で起動されたエキスパートや指標に送信されます。EAまたは指標が既にマウス移動イベントを受信している間に、メインアプリケーションスレッドがWM_PAINTシステムメッセージをまだ処理していない（したがって、まだ変更されたチャートをレンダリングしていない）状況が発生する可能性があります。この場合、チャートプロパティCHART_FIRST_VISIBLE_BARはチャートがレンダリングされた後にのみ変更されます。

各イベントタイプに対して、OnChartEvent()関数の入力には、そのイベントを処理するために必要な特定の値があります。この表では、パラメータを介して渡されるイベントと値が一覧できます。

イベント	'id'パラメータ値	'lparam'パラメータ値	'dparam'パラメータ値	'sparam'パラメータ値
キーストロークイベント	CHARTEVENT_KEYDOWN	押下されたボタンのコード	キーが長押しされた状態で生成されたキー押下の数	キーボードのキーの状態を表すビットマスクの文字列値
マウスイベント（チャートのCHART_EVENT_MOUSE_MOVE=trueの場合）	CHARTEVENT_MOUSE_MOVE	X座標	Y座標	マウスキーの状態を表すビットマスクの文字列値
マウスホイールイベント（チャートのCHART_EVENT_MOUSE_WHEEL=trueの場合）	CHARTEVENT_MOUSE_WHEEL	キーとマウスボタンの状態のフラグ、カーソルのX座標とY座標。例をご覧ください。	マウスホイールスクロールのデルタ値	—
グラフィックオブジェクト作成（チャートのCHART_EVENT_OBJECT_CREATE=trueの場合）	CHARTEVENT_OBJECT_CREATE	—	—	作成したグラフィックオブジェクトの名称
プロパティダイアログを介してのオブジェクトプロパティ変更	CHARTEVENT_OBJECT_CHANGE	—	—	変更したグラフィックオブジェクトの名称
グラフィックオブジェクトの削除（チャートのCHART_EVENT_OBJECT_DELETE=trueの場合）	CHARTEVENT_OBJECT_DELETE	—	—	削除したグラフィックオブジェクトの名称

© 2000-2025, MetaQuotes Ltd.

イベント	'id'パラメータ値	'lparam'パラメータ値	'dparam'パラメータ値	'sparam'パラメータ値
チャート上でのマウスクリック	CHARTEVENT_CLICK	X座標	Y座標	—
グラフィックオブジェクトでのマウスクリック	CHARTEVENT_OBJECT_CLICK	X座標	Y座標	イベントが発生したグラフィック・オブジェクトの名称
マウスでのグラフィックオブジェクトの移動	CHARTEVENT_OBJECT_DRAG	—	—	移動したグラフィックオブジェクトの名称
"Input field"（入力フィールド）グラフィックオブジェクトのテキスト編集の完成	CHARTEVENT_OBJECT_ENDEDIT	—	—	テキスト編集が完成した「入力フィールド」グラフィックオブジェクトの名称
プロパティダイアログウィンドウでのチャートのサイズ変更やチャートのプロパティ変更	CHARTEVENT_CHART_CHANGE	—	—	—
N番号を使用したカスタムイベント	CHARTEVENT_CUSTOM+N	EventChartCustom()関数で定義された値	EventChartCustom()関数で定義された値	EventChartCustom()関数で定義された値

チャートイベントリスナーの例

```
//+------------------------------------------------------------------+
//|                                          OnChartEvent_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Sample chart event listener and custom events generator"
//--- サービスキーID
#define KEY_NUMPAD_5        12
#define KEY_LEFT            37
#define KEY_UP              38
#define KEY_RIGHT           39
#define KEY_DOWN            40
#define KEY_NUMLOCK_DOWN    98
#define KEY_NUMLOCK_LEFT   100
#define KEY_NUMLOCK_5      101
#define KEY_NUMLOCK_RIGHT  102
#define KEY_NUMLOCK_UP     104
```

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  エキスパート初期化関数                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- CHARTEVENT_CUSTOM定数値を表示する
   Print("CHARTEVENT_CUSTOM=",CHARTEVENT_CUSTOM);
//---
   Print("Launched the EA ",MQLInfoString(MQL5_PROGRAM_NAME));
//--- チャートオブジェクト作成イベントの受信フラグを設定する
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_CREATE,true);
//--- チャートオブジェクト削除イベントの受信フラグを設定する
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_DELETE,true);
//--- マウスホイールスクロールメッセージの有効化
   ChartSetInteger(0,CHART_EVENT_MOUSE_WHEEL,1);
//--- チャートプロパティの強制更新により、イベント処理の準備が整う
   ChartRedraw();
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//|  エキスパートティック関数                                        |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- カスタムイベント生成のためのティックカウンタ
   static int tick_counter=0;
//--- 累積されたティックをこの値で除算する
   int simple_number=113;
//---
   tick_counter++;
//--- ティックカウンタがsimple_numberの倍数である場合、カスタムイベントを送信する
   if(tick_counter%simple_number==0)
     {
      //--- カスタムイベントIDを0～65535で形成する
      ushort custom_event_id=ushort(tick_counter%65535);
      //---  カスタムイベントを送信し、パラメータを記入する
      EventChartCustom
(ChartID(),custom_event_id,tick_counter,SymbolInfoDouble
(Symbol(),SYMBOL_BID),__FUNCTION__);
      //--- 結果例を分析するためにログに追加する
      Print(__FUNCTION__,": Sent a custom event ID=",custom_event_id);
     }
//---
  }
//+------------------------------------------------------------------+
//|  ChartEvent 関数                                                 |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
```

© 2000-2025, MetaQuotes Ltd.

```
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- キーの押下
   if(id==CHARTEVENT_KEYDOWN)
     {
      switch((int)lparam)
        {
         case KEY_NUMLOCK_LEFT:  Print("Pressed KEY_NUMLOCK_LEFT");   break;
         case KEY_LEFT:          Print("Pressed KEY_LEFT");           break;
         case KEY_NUMLOCK_UP:    Print("Pressed KEY_NUMLOCK_UP");     break;
         case KEY_UP:            Print("Pressed KEY_UP");             break;
         case KEY_NUMLOCK_RIGHT: Print("Pressed KEY_NUMLOCK_RIGHT");  break;
         case KEY_RIGHT:         Print("Pressed KEY_RIGHT");          break;
         case KEY_NUMLOCK_DOWN:  Print("Pressed KEY_NUMLOCK_DOWN");   break;
         case KEY_DOWN:          Print("Pressed KEY_DOWN");           break;
         case KEY_NUMPAD_5:      Print("Pressed KEY_NUMPAD_5");       break;
         case KEY_NUMLOCK_5:     Print("Pressed KEY_NUMLOCK_5");      break;
         default:                Print("Pressed unlisted key");
        }
     }
//--- チャートの左クリック
   if(id==CHARTEVENT_CLICK)
      Print("Mouse click coordinates on a chart: x = ",lparam,"  y = ",dparam);
//--- グラフィックオブジェクトのクリック
   if(id==CHARTEVENT_OBJECT_CLICK)
      Print("Clicking a mouse button on an object named '"+sparam+"'");
//--- オブジェクト削除
   if(id==CHARTEVENT_OBJECT_DELETE)
      Print("Removed object named ",sparam);
//--- オブジェクト作成
   if(id==CHARTEVENT_OBJECT_CREATE)
      Print("Created object named ",sparam);
//--- オブジェクト変更
   if(id==CHARTEVENT_OBJECT_CHANGE)
      Print("Changed object named ",sparam);
//--- オブジェクト移動またはアンカーポイント座標変更
   if(id==CHARTEVENT_OBJECT_DRAG)
      Print("Changing anchor points of object named ",sparam);
//--- Edi t グラフィックオブジェクトのテキスト編集
   if(id==CHARTEVENT_OBJECT_ENDEDIT)
      Print("Changed text in Edit object ",sparam,"  id=",id);
//--- マウス移動イベント
   if(id==CHARTEVENT_MOUSE_MOVE)
      Comment("POINT: ",(int)lparam,",",(int)dparam,"\n",MouseState((uint)sparam));
   if(id==CHARTEVENT_MOUSE_WHEEL)
     {
      //--- このイベントのマウスボタンとホイールの状態を考察する
```

© 2000-2025, MetaQuotes Ltd.

```
      int flg_keys = (int)(lparam>>32);          // Ctrl キー、Shift キー、マウスボタンの状態のフラグ
      int x_cursor = (int)(short)lparam;         // マウスホイールイベントが発生したX座標
      int y_cursor = (int)(short)(lparam>>16);   // マウスホイールイベントが発生したY座標
      int delta    = (int)dparam;                // マウススクロールの合計値(+120または -120が越されるとトリガされる)
      //--- フラグの処理
      string str_keys="";
      if((flg_keys&0x0001)!=0)
         str_keys+="LMOUSE ";
      if((flg_keys&0x0002)!=0)
         str_keys+="RMOUSE ";
      if((flg_keys&0x0004)!=0)
         str_keys+="SHIFT ";
      if((flg_keys&0x0008)!=0)
         str_keys+="CTRL ";
      if((flg_keys&0x0010)!=0)
         str_keys+="MMOUSE ";
      if((flg_keys&0x0020)!=0)
         str_keys+="X1MOUSE ";
      if((flg_keys&0x0040)!=0)
         str_keys+="X2MOUSE ";

      if(str_keys!="")
         str_keys=", keys='"+StringSubstr(str_keys,0,StringLen(str_keys)-1)+"'";
      PrintFormat("%s: X=%d, Y=%d, delta=%d%s",EnumToString(CHARTEVENT_MOUSE_WHEEL),x_cursor,y_cursor,delta,str_keys);
     }
//--- プロパティダイアログウィンドウでのチャートのサイズおよびプロパティ変更イベント
   if(id==CHARTEVENT_CHART_CHANGE)
      Print("Changing the chart size or properties");
//--- カスタムイベント
   if(id>CHARTEVENT_CUSTOM)
      PrintFormat("Custom event ID=%d, lparam=%d, dparam=%G, sparam=%s",id,lparam,dparam,sparam);
  }
//+------------------------------------------------------------------+
//| マウスの状態                                                     |
//+------------------------------------------------------------------+
string MouseState(uint state)
  {
   string res;
   res+="\nML: "    +(((state& 1)== 1)?"DN":"UP");   // 左ボタン
   res+="\nMR: "    +(((state& 2)== 2)?"DN":"UP");   // 右ボタン
   res+="\nMM: "    +(((state&16)==16)?"DN":"UP");   // 中央ボタン
   res+="\nMX: "    +(((state&32)==32)?"DN":"UP");   // 追加ボタン1
   res+="\nMY: "    +(((state&64)==64)?"DN":"UP");   // 追加ボタン 2
   res+="\nSHIFT: "+(((state& 4)== 4)?"DN":"UP");   // シフト キー
   res+="\nCTRL: " +(((state& 8)== 8)?"DN":"UP");   // control キー
```

© 2000-2025, MetaQuotes Ltd.

```
   return(res);
  }
```

参照

EventChartCustom、Types of chart events、Event handling functions、Program running、Client terminal events

© 2000–2025, MetaQuotes Ltd.

OnTester

この関数は、Testerイベントが発生してテスト後に適切なアクションを取るために、EAで呼び出されます。

```
double  OnTester(void);
```

戻り値

テスト結果を評価するためのカスタム条件最適化の値

注意事項

OnTester()関数は、EAのテスト時にのみ使用でき、主に入力パラメータの遺伝的最適化のためのカスタム最大基準として使用される値の計算のために意図されています。

遺伝的最適化では一世代内の結果は降順にソートされます。これは、最高値の結果が最適化基準の観点から最良とみなされることを意味します。このようなソーティングの最悪値は最後に配置され、その後破棄されます。したがって、それらは次世代の形成に参加しません。

したがって、OnTester()関数を使用すると、独自のテスト結果レポートが作成され保存されるだけでなく、最適化プロセスを制御してトレーディング戦略の最良のパラメータを見つけることができます。

以下はカスタム条件最適化を計算する例です。アイディアは、バランスグラフの線形回帰を計算することです。これはOptimizing a strategy using balance graph and comparing results with "Balance + max Sharpe Ratio" criterion（バランスグラフを使用した戦略の最適化と、結果の「バランス+最大シャープレシオ」基準との比較）稿で説明されています。

```
//+------------------------------------------------------------------+
//|                                                OnTester_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Sample EA with the OnTester() handler"
#property description "As a custom optimization criterion, "
#property description "the ratio of the balance graph linear regression"
#property description "divided by the deviation mean-square error is returned"
//-- 取引操作クラスをインクルードする
#include <Trade\Trade.mqh>
//--- EA入力パラメータ
input double Lots                = 0.1;      // ボリューム
input int    Slippage            = 10;       // 許容されるスリッページ
input int    MovingPeriod        = 80;       // 移動平均期間
input int    MovingShift         = 6;        // 移動平均シフト
//--- グローバル変数
int    IndicatorHandle=0;  // 指標ハンドル
bool   IsHedging=false;    // アカウントのフラグ
CTrade trade;              // 取引操作の実行
//---
#define EA_MAGIC 18052018
```

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  ポジションを開く条件を確認する                                  |
//+------------------------------------------------------------------+
void CheckForOpen(void)
  {
   MqlRates rt[2];
//--- 新しいバーの始めのみで取引する
   if(CopyRates(_Symbol,_Period,0,2,rt)!=2)
     {
      Print("CopyRates of ",_Symbol," failed, no history");
      return;
     }
//--- ティックボリューム
   if(rt[1].tick_volume>1)
      return;
//--- 移動平均値を受け取る
   double   ma[1];
   if(CopyBuffer(IndicatorHandle,0,1,1,ma)!=1)
     {
      Print("CopyBuffer from iMA failed, no data");
      return;
     }
//--- シグナルの存在を確認する
   ENUM_ORDER_TYPE signal=WRONG_VALUE;
//--- ローソク足は移動平均より高く開いたが、移動平均より低く閉じた
   if(rt[0].open>ma[0] && rt[0].close<ma[0])
      signal=ORDER_TYPE_BUY;     // 買いシグナル
   else // ローソク足は移動平均より低く開いたが、移動平均より高く閉じた
     {
      if(rt[0].open<ma[0] && rt[0].close>ma[0])
         signal=ORDER_TYPE_SELL;// 売りシグナル
     }
//--- 追加的確認
   if(signal!=WRONG_VALUE)
     {
      if(TerminalInfoInteger(TERMINAL_TRADE_ALLOWED) && Bars(_Symbol,_Period)>100)
        {
         double price=SymbolInfoDouble(_Symbol,signal==ORDER_TYPE_SELL ?
SYMBOL_BID:SYMBOL_ASK);
         trade.PositionOpen(_Symbol,signal,Lots,price,0,0);
        }
     }
//---
  }
//+------------------------------------------------------------------+
//|  ポジションを閉じる条件を確認する                                |
//+------------------------------------------------------------------+
void CheckForClose(void)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   MqlRates rt[2];
//--- 新しいバーの始めのみで取引する
   if(CopyRates(_Symbol,_Period,0,2,rt)!=2)
     {
      Print("CopyRates of ",_Symbol," failed, no history");
      return;
     }
   if(rt[1].tick_volume>1)
      return;
//--- 移動平均値を受け取る
   double   ma[1];
   if(CopyBuffer(IndicatorHandle,0,1,1,ma)!=1)
     {
      Print("CopyBuffer from iMA failed, no data");
      return;
     }
//--- ポジションはPositionSelect()を使用してすでに選択されている
   bool signal=false;
   long type=PositionGetInteger(POSITION_TYPE);
//--- ローソク足は移動平均より高く開いたが、移動平均より低く閉じた - ショートポジションを決済する
   if(type==(long)POSITION_TYPE_SELL && rt[0].open>ma[0] && rt[0].close<ma[0])
      signal=true;
//--- ローソク足は移動平均より低く開いたが、移動平均より高く閉じた - ロングポジションを決済する
   if(type==(long)POSITION_TYPE_BUY && rt[0].open<ma[0] && rt[0].close>ma[0])
      signal=true;
//--- 追加的確認
   if(signal)
     {
      if(TerminalInfoInteger(TERMINAL_TRADE_ALLOWED) && Bars(_Symbol,_Period)>100)
         trade.PositionClose(_Symbol,Slippage);
     }
//---
  }
//+------------------------------------------------------------------+
//|  口座タイプ(ネッティングまたはヘッジ)を考慮してポジションを選択する                  |
//+------------------------------------------------------------------+
bool SelectPosition()
  {
   bool res=false;
//--- ヘッジ口座のポジションを選択する
   if(IsHedging)
     {
      uint total=PositionsTotal();
      for(uint i=0; i<total; i++)
        {
         string position_symbol=PositionGetSymbol(i);
         if(_Symbol==position_symbol &&
EA_MAGIC==PositionGetInteger(POSITION_MAGIC))
           {
```

© 2000-2025, MetaQuotes Ltd.

```
            res=true;
            break;
           }
        }
     }
//--- ネッティング口座のポジションを選択する
   else
     {
      if(!PositionSelect(_Symbol))
         return(false);
      else
         return(PositionGetInteger(POSITION_MAGIC)==EA_MAGIC); //---check Magic
number
     }
//--- 実行結果
   return(res);
  }
//+------------------------------------------------------------------+
//| エキスパート初期化関数                                           |
//+------------------------------------------------------------------+
int OnInit(void)
  {
//--- 取引タイプ(ネッティングまたはヘッジ)を設定する
   IsHedging=((ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger(ACCOUNT_MARGIN_MODE)
==ACCOUNT_MARGIN_MODE_RETAIL_HEDGING);
//--- 正しいポジション制御のためにオブジェクトを初期化する
   trade.SetExpertMagicNumber(EA_MAGIC);
   trade.SetMarginMode();
   trade.SetTypeFillingBySymbol(Symbol());
   trade.SetDeviationInPoints(Slippage);
//--- 移動平均指標を作成する
   IndicatorHandle=iMA
(_Symbol,_Period,MovingPeriod,MovingShift,MODE_SMA,PRICE_CLOSE);
   if(IndicatorHandle==INVALID_HANDLE)
     {
      printf("Error creating iMA indicator");
      return(INIT_FAILED);
     }
//--- ok
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| エキスパートティック関数                                         |
//+------------------------------------------------------------------+
void OnTick(void)
  {
//--- ポジションが既に開かれている場合は、決済条件を確認する
   if(SelectPosition())
      CheckForClose();
```

© 2000-2025, MetaQuotes Ltd.

```
// ポジションを開く条件を確認する
   CheckForOpen();
//---
  }
//+------------------------------------------------------------------+
//| テスタ関数                                                       |
//+------------------------------------------------------------------+
double OnTester()
  {
//--- カスタム条件最適化の値（高いほど良い）
   double ret=0.0;
//--- 取引結果を配列に入れる
   double array[];
   double trades_volume;
   GetTradeResultsToArray(array,trades_volume);
   int trades=ArraySize(array);
//--- 10取引未満の場合、肯定的結果がないことをテストする
   if(trades<10)
      return (0);
//--- 取引あたりの平均結果
   double average_pl=0;
   for(int i=0;i<ArraySize(array);i++)
      average_pl+=array[i];
   average_pl/=trades;
//--- 単一テストモード用のメッセージを表示する
   if(MQLInfoInteger(MQL_TESTER) && !MQLInfoInteger(MQL_OPTIMIZATION))
      PrintFormat("%s: Trades=%d, Average profit=
%.2f",__FUNCTION__,trades,average_pl);
//--- 利益グラフの線形回帰を計算する
   double a,b,std_error;
   double chart[];
   if(!CalculateLinearRegression(array,chart,a,b))
      return (0);
//--- 回帰直線からグラフの偏差の誤差を計算する
   if(!CalculateStdError(chart,a,b,std_error))
      return (0);
//--- 傾向偏差の標準偏差を計算する
   ret=(std_error == 0.0) ? a*trades : a*trades/std_error;
//--- カスタム条件最適化値を返す
   return(ret);
  }
//+------------------------------------------------------------------+
//| 取引の利益/損失の配列を得る                                      |
//+------------------------------------------------------------------+
bool GetTradeResultsToArray(double &pl_results[],double &volume)
  {
//--- 完全な取引履歴をリクエストする
   if(!HistorySelect(0,TimeCurrent()))
      return (false);
```

© 2000–2025, MetaQuotes Ltd.

```
   uint total_deals=HistoryDealsTotal();
   volume=0;
//--- 証拠金を持つ配列の初期サイズを、履歴の取引数で設定する
   ArrayResize(pl_results,total_deals);
//--- 取引結果を修正する取引のカウンター - 利益または損失
   int counter=0;
   ulong ticket_history_deal=0;
//--- 全ての取引を見る
   for(uint i=0;i<total_deals;i++)
     {
      //--- 取引を選択する
      if((ticket_history_deal=HistoryDealGetTicket(i))>0)
        {
         ENUM_DEAL_ENTRY deal_entry  =(ENUM_DEAL_ENTRY)
HistoryDealGetInteger(ticket_history_deal,DEAL_ENTRY);
         long            deal_type
  =HistoryDealGetInteger(ticket_history_deal,DEAL_TYPE);
         double          deal_profit
=HistoryDealGetDouble(ticket_history_deal,DEAL_PROFIT);
         double          deal_volume
=HistoryDealGetDouble(ticket_history_deal,DEAL_VOLUME);
         //--- 興味があるのは取引操作のみである
         if((deal_type!=DEAL_TYPE_BUY) && (deal_type!=DEAL_TYPE_SELL))
            continue;
         //--- 損益を固定する取引のみ
         if(deal_entry!=DEAL_ENTRY_IN)
           {
            //--- 取引結果を配列に書き込み、取引のカウンターを増やす
            pl_results[counter]=deal_profit;
            volume+=deal_volume;
            counter++;
           }
        }
     }
//--- 配列の最終サイズを設定する
   ArrayResize(pl_results,counter);
   return (true);
  }
//+------------------------------------------------------------------+
//|  線形回帰を計算する y=a*x+b                                      |
//+------------------------------------------------------------------+
bool CalculateLinearRegression(double  &change[],double &chartline[],
                               double  &a_coef,double  &b_coef)
  {
//--- データが十分か確認する
   if(ArraySize(change)<3)
      return (false);
//--- 蓄積されたチャート配列を作成する
   int N=ArraySize(change);
```

© 2000-2025, MetaQuotes Ltd.

```
   ArrayResize(chartline,N);
   chartline[0]=change[0];
   for(int i=1;i<N;i++)
      chartline[i]=chartline[i-1]+change[i];
//--- 線形回帰を計算する
   double x=0,y=0,x2=0,xy=0;
   for(int i=0;i<N;i++)
     {
      x=x+i;
      y=y+chartline[i];
      xy=xy+i*chartline[i];
      x2=x2+i*i;
     }
   a_coef=(N*xy-x*y)/(N*x2-x*x);
   b_coef=(y-a_coef*x)/N;
//---
   return (true);
  }
//+------------------------------------------------------------------+
//|  指定されたaとbの平均二乗偏差誤差を計算する                      |
//+------------------------------------------------------------------+
bool  CalculateStdError(double  &data[],double  a_coef,double  b_coef,double
&std_err)
  {
//--- 誤差の平方和
   double error=0;
   int N=ArraySize(data);
   if(N<=2)
      return (false);
   for(int i=0;i<N;i++)
      error+=MathPow(a_coef*i+b_coef-data[i],2);
   std_err=MathSqrt(error/(N-2));
//---
   return (true);
  }
```

参照

Testing trading strategies、TesterHideIndicators、Working with optimization results、TesterStatistics、OnTesterInit、OnTesterDeinit、OnTesterPass、MQL_TESTER、MQL_OPTIMIZATION、FileOpen、FileWrite、FileLoad、FileSave

© 2000–2025, MetaQuotes Ltd.

OnTesterInit

この関数は、 TesterInitイベントが発生してストラテジーテスターでの最適化に先立って必要なアクションが実行されるときに、EAで呼び出されます。この関数の型は 2 つあります。

結果を返すバージョン

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

戻り値

int型の値。ゼロは、最適化が開始される前にチャート上で起動されたEAの初期化に成功したことを意味します。

実行結果を返すOnTesterInit() を呼び出すと、プログラムを初期化できるだけでなく、最適化が異常に終了した場合にはエラーコードが返されるため、これを使用することをお勧めします。INIT_SUCCEEDED (0)以外の戻り値はエラーを意味し、最適化は行われません。

結果を返さないバージョンは古いコードとの相互性の為のみに残されています。使用は推奨されません。

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

注意事項

TesterInitイベントは、ストラテジーテスターにおけるEAの最適化の前に生成されます。このイベントでは、OnTesterDeInit()またはOnTesterPass()イベントハンドラを持つEAが自動的に別の端末チャートにダウンロードされます。テスターで指定されたシンボルと期間があります。

このようなイベントはTesterInit、TesterDeinit、TesterPassイベントを受信しますが、Init、Deinit、NewTickイベントは受信されません。したがって、最適化中に各パスの結果を処理するために必要なロジックはすべて、OnTesterInit ()、OnTesterDeinit ()、OnTesterPass ()ハンドラで実装されるべきです。

戦略最適化中の各シングルパスの結果は、FrameAdd ()関数を使用してOnTester ()ハンドラからフレームを介して渡すことができます。

OnTesterInit()関数は、最適化を開始する前にエキスパートアドバイザーを起動して、最適化結果をさらに処理するために使用されます。これは常にOnTesterDeinit()ハンドラと共に使用されます。

OnTesterInit()の実行時間は制限されています。超過した場合、EAは強制的に停止され、最適化自体は取り消されます。テスター操作ログにメッセージが表示されます。

```
TesterOnTesterInit works too long. Tester cannot be initialized.
```

例はOnTickからとられています。 OnTesterInit()ハンドラは最適化パラメータを設定するために追加されました。:

```
//+------------------------------------------------------------------+
//|                                          OnTesterInit_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

© 2000–2025, MetaQuotes Ltd.

```
#property description "Sample EA with the OnTesterInit() handler,"
#property description "in which values and limitations of "
#property description "inputs during optimization are set"

input double lots=0. 1;        // ロット単位のボリューム
input double kATR=3;           // ATRのシグナルローソク足の長さ
input int    ATRperiod=20;     // ATR指標期間
input int    holdbars=8;       // ポジションのバー数
input int    slippage=10;      // 許容されるスリッページ
input bool   revers=false;     // シグナルを反転するかどうか
input ulong  EXPERT_MAGIC=0;   // EAのマジックナンバー
//--- ATR指標ハンドルを格納
int atr_handle;
//--- ここで最後のATR値とローソク足のの実体を保存する
double last_atr,last_body;
datetime lastbar_timeopen;
double trade_lot;
//--- 最適化開始時間を記憶する
datetime optimization_start;
//--- 最適化が終了した後にチャートに期間を表示する
string report;
//+------------------------------------------------------------------+
//| TesterInit 関数                                                  |
//+------------------------------------------------------------------+
void OnTesterInit()
  {
//--- 最適化のための入力値を設定する
   ParameterSetRange("lots",false,0.1,0,0,0);
   ParameterSetRange("kATR",true,3.0,1.0,0.3,7.0);
   ParameterSetRange("ATRperiod",true,10,15,1,30);
   ParameterSetRange("holdbars",true,5,3,1,15);
   ParameterSetRange("slippage",false,10,0,0,0);
   ParameterSetRange("revers",true,false,false,1,true);
   ParameterSetRange("EXPERT_MAGIC",false,123456,0,0,0);
   Print("Initial values and optimization parameter limitations are set");
//--- 最適化開始時間を記憶する
   optimization_start=TimeLocal();
   report=StringFormat("%s: optimization launched at %s",
                       __FUNCTION__,TimeToString(TimeLocal(),TIME_MINUTES|
TIME_SECONDS));
//--- メッセージをチャートと端末操作ログに表示する
   Print(report);
   Comment(report);
//---
  }
//+------------------------------------------------------------------+
//| TesterDeinit 関数                                                |
//+------------------------------------------------------------------+
void OnTesterDeinit()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- 最適化期間
   string log_message=StringFormat("%s: optimization took %d seconds",
                                   __FUNCTION__,TimeLocal()-optimization_start);
   PrintFormat(log_message);
   report=report+"\r\n"+log_message;
   Comment(report);
  }
//+------------------------------------------------------------------+
//| エキスパート初期化関数                                           |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- グローバル変数を初期化する
   last_atr=0;
   last_body=0;
//--- 正しいボリュームを設定する
   double min_lot=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
   trade_lot=lots>min_lot?lots:min_lot;
//--- ATR指標ハンドルを作成する
   atr_handle=iATR(_Symbol,_Period,ATRperiod);
   if(atr_handle==INVALID_HANDLE)
     {
      PrintFormat("%s: failed to create iATR, error code %
d",__FUNCTION__,GetLastError());
      return(INIT_FAILED);
     }
//--- EA初期化が成功した
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| エキスパートティック関数                                         |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- 取引シグナル
   static int signal=0; // +1は買いシグナル、 -1は売りシグナル
//--- 'holdbars' バーより前に開かれた以前のポジションを確認して決済する
   ClosePositionsByBars(holdbars,slippage,EXPERT_MAGIC);
//--- 新しいバーを確認する
   if(isNewBar())
     {
      //--- シグナルの存在を確認する
      signal=CheckSignal();
     }
//--- ネットポジションが開かれている場合は、信号をスキップして終了する
   if(signal!=0 && PositionsTotal()>0 && (ENUM_ACCOUNT_MARGIN_MODE)
AccountInfoInteger(ACCOUNT_MARGIN_MODE)==ACCOUNT_MARGIN_MODE_RETAIL_NETTING)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      signal=0;
      return; // NewTick イベントハンドラを終了し、新しいバーが現れる前に市場に入らない
     }
//--- ヘッジ口座の場合、各ポジションは個別に保持され、決済される
   if(signal!=0)
     {
      //--- 買いシグナル
      if(signal>0)
        {
         PrintFormat("%s: Buy signal!Revers=%s",__FUNCTION__,string(revers));
         if(Buy(trade_lot,slippage,EXPERT_MAGIC))
            signal=0;
        }
      //--- 売りシグナル
      if(signal<0)
        {
         PrintFormat("%s: Sell signal!Revers=%s",__FUNCTION__,string(revers));
         if(Sell(trade_lot,slippage,EXPERT_MAGIC))
            signal=0;
        }
     }
//--- OnTick 関数の終わり
  }
//+------------------------------------------------------------------+
//| 新しい取引シグナルを確認する                                     |
//+------------------------------------------------------------------+
int CheckSignal()
  {
//--- 0はシグナルが不在なことを意味する
   int res=0;
//--- 終わりから2番目の完全なバーにATR値を取得する(バーのインデックスは2)
   double atr_value[1];
   if(CopyBuffer(atr_handle,0,2,1,atr_value)!=-1)
     {
      last_atr=atr_value[0];
      //--- 最後に閉じたバーのデータをMql Rates型配列に取得する
      MqlRates bar[1];
      if(CopyRates(_Symbol,_Period,1,1,bar)!=-1)
        {
         //--- 最後の完全なバーでバー本体のサイズを計算する
         last_body=bar[0].close-bar[0].open;
         //--- 最後のバー(インデックス1)の本体が以前のATR値(インデックス2のバー上)を超える場合、取引シグナルが受信される
         if(MathAbs(last_body)>kATR*last_atr)
            res=last_body>0?1:-1; // 上昇ローソク足では正の値
        }
      else
         PrintFormat("%s: Failed to receive the last bar!Error",__FUNCTION__,GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   else
      PrintFormat("%s: Failed to receive ATR indicator
value!Error",__FUNCTION__,GetLastError());
//--- 反転取引モードが有効な場合
   res=revers?-res:res;  // 必要に応じてシグナルを反転させる(1の代わりに-1を返し、逆も同様)
//--- 取引シグナル値を返す
   return (res);
  }
//+------------------------------------------------------------------+
//|    新しいバーが現れると'true'                                    |
//+------------------------------------------------------------------+
bool isNewBar(const bool print_log=true)
  {
   static datetime bartime=0; // 現在のバーの開いた時刻を格納する
//--- ゼロバーの開いた時間を取得する
   datetime currbar_time=iTime(_Symbol,_Period,0);
//--- 新しいバーが到着すると開いた時刻が変わる
   if(bartime!=currbar_time)
     {
      bartime=currbar_time;
      lastbar_timeopen=bartime;
      //--- ログに新しいバーが開いている時間のデータを表示する
      if(print_log && !(MQLInfoInteger(MQL_OPTIMIZATION)||
MQLInfoInteger(MQL_TESTER)))
        {
         //--- 新しいバーを開いた時間のメッセージを表示する
         PrintFormat("%s: new bar on %s %s opened at %s",__FUNCTION__,_Symbol,
                     StringSubstr(EnumToString(_Period),7),
                     TimeToString(TimeCurrent(),TIME_SECONDS));
         //--- 最後のティックのデータを取得する
         MqlTick last_tick;
         if(!SymbolInfoTick(Symbol(),last_tick))
            Print("SymbolInfoTick() failed, error = ",GetLastError());
         //--- 最後のティックタイムをミリ秒まで表示する
         PrintFormat("Last tick was at %s.%03d",
                     TimeToString(last_tick.time,TIME_SECONDS),last_tick.time_msc%
1000);
        }
      //--- 新しいバーがある
      return (true);
     }
//--- 新しいバーがない
   return (false);
  }
//+------------------------------------------------------------------+
//|  成行価格で指定された量で買う                                    |
//+------------------------------------------------------------------+
bool Buy(double volume,ulong deviation=10,ulong  magicnumber=0)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- 成行価格で買う
   return (MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber));
  }
//+------------------------------------------------------------------+
//| 成行価格で指定された量で売る                                     |
//+------------------------------------------------------------------+
bool Sell(double volume,ulong deviation=10,ulong  magicnumber=0)
  {
//--- 成行価格で売る
   return (MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber));
  }
//+------------------------------------------------------------------+
//| バー数によるポジション保留時間ごとにポジションを決済する        |
//+------------------------------------------------------------------+
void ClosePositionsByBars(int holdtimebars,ulong deviation=10,ulong  magicnumber=0)
  {
   int total=PositionsTotal(); // ポジション数
//--- ポジションをすべて見る
   for(int i=total-1; i>=0; i--)
     {
      //--- ポジションパラメータ
      ulong  position_ticket=PositionGetTicket(i);
    // ポジションチケット
      string
position_symbol=PositionGetString(POSITION_SYMBOL);                          // シンボル
      ulong
 magic=PositionGetInteger(POSITION_MAGIC);                                    // ポジショ
ンマジックナンバー
      datetime position_open=(datetime)
PositionGetInteger(POSITION_TIME);                // ポジションの開かれた時刻
      int bars=iBarShift(_Symbol,PERIOD_CURRENT,position_open)+1;
    // ポジションがどれだけ前(バー数)に開かれたか

      //--- マジックナンバーとシンボルが一致していてポジションがすでに長くある場合
      if(bars>holdtimebars && magic==magicnumber && position_symbol==_Symbol)
        {
         int    digits=(int)
SymbolInfoInteger(position_symbol,SYMBOL_DIGITS);          // 小数点以下桁数
         double
volume=PositionGetDouble(POSITION_VOLUME);                                   // ポジションボ
リューム
         ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)
PositionGetInteger(POSITION_TYPE); // ポジションの種類
         string str_type=StringSubstr(EnumToString(type),14);
         StringToLower(str_type); // 正しいメッセージ書式設定のために小文字にする
         PrintFormat("Close position #%I64u %s %s %.2f",
                     position_ticket,position_symbol,str_type,volume);
         //--- 注文タイプを設定し、取引リクエストを送信する
         if(type==POSITION_TYPE_BUY)
```

© 2000-2025, MetaQuotes Ltd.

```
            MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber,position_ticket
);
         else
            MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber,position_ticket)
;
        }
     }
  }
//+------------------------------------------------------------------+
//|  取引リクエストを準備して送信する                                |
//+------------------------------------------------------------------+
bool MarketOrder(ENUM_ORDER_TYPE type,double volume,ulong slip,ulong
magicnumber,ulong pos_ticket=0)
  {
//--- 構造体の宣言と初期化
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(type==ORDER_TYPE_BUY)
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
//--- リクエストパラメータ
   request.action   =TRADE_ACTION_DEAL;                     // 取引操作の種類
   request.position =pos_ticket;                            // 決済する場合のポジションチケッ
ト
   request.symbol   =Symbol();                              // シンボル
   request.volume   =volume;                                // ボリューム
   request.type     =type;                                  // 注文の種類
   request.price    =price;                                 // 取引価格
   request.deviation=slip;                                  // 価格からの許容偏差
   request.magic    =magicnumber;                           // 注文のマジックナンバー
//--- リクエストを送信する
   if(!OrderSend(request,result))
     {
      //--- 失敗したらデータを表示する
      PrintFormat("OrderSend %s %s %.2f at %.5f error %d",
                  request.symbol
,EnumToString(type),volume,request.price,GetLastError());
      return (false);
     }
//--- 操作の成功を知らせる
   PrintFormat("retcode=%u  deal=%I64u  order=%
I64u",result.retcode,result.deal,result.order);
   return (true);
  }
```

参照

Testing trading strategies、Working with optimization results、OnTesterDeinit、OnTesterPass、ParameterGetRange、ParameterSetRange

© 2000-2025, MetaQuotes Ltd.

OnTesterDeinit

この関数は、EA最適化の後にTesterDeinit イベントが発生したときに、EAで呼び出されます。

```
void  OnTesterDeinit(void);
```

戻り値

なし

注意事項

TesterDeinitイベントは、ストラテジーテスターにおけるEAの最適化の終了後に生成されます。

OnTesterDeInit()またはOnTesterPass()　イベントハンドラを持つEAは、最適化開始時に別の端末チャートに自動的にダウンロードされます。テスターで指定されたシンボルと期間があります。関数はすべての最適化結果の最終処理用に設計されています。

FrameAdd()関数を使用してテストエージェントによって送信された最適化フレームはバンドルされ、配信に時間がかかることに注意してください。よって、すべてのフレームやTesterPassイベントが最適化が終わるまでに到着しOnTesterPass()で処理されるわけではありません。遅れているフレームをすべてOnTesterDeinit()で受け取る場合は、FrameNext()関数を使用してコードブロックを配置します。

参照

Testing trading strategies、Working with optimization results、TesterStatistics、OnTesterInit、OnTesterPass、ParameterGetRange、ParameterSetRange

© 2000–2025, MetaQuotes Ltd.

OnTesterPass

この関数は、EA最適化中に新しいデータフレームを処理するためにTesterPassTesterPassイベントが発生するときに、EAで呼び出されます。

```
void  OnTesterPass(void);
```

戻り値

なし

注意事項

TesterPassイベントは、ストラテジーテスターでのエキスパートアドバイザーの最適化中にフレームを受け取ると自動的に生成されます。

OnTesterDeInit()またはOnTesterPass()　イベントハンドラを持つEAは、最適化開始時に別の端末チャートに自動的にダウンロードされます。テスターで指定されたシンボルと期間があります。この関数は、最適化中にテストエージェントから受信したフレームを処理するためのものです。テスト結果を含むフレームは、FrameAdd()関数を使用してOnTester()ハンドラーから送信する必要があります。

FrameAdd()関数を使用してテストエージェントによって送信された最適化フレームはバンドルされ、配信に時間がかかることに注意してください。よって、すべてのフレームやTesterPassイベントが最適化が終わるまでに到着しOnTesterPass()で処理されるわけではありません。遅れているフレームをすべてOnTesterDeinit()で受け取る場合は、FrameNext()関数を使用してコードブロックを配置します。

OnTesterDeinit()の最適化を完了すると、FrameFirst()/FrameFilterとFrameNext()関数を使用して再び、受け取ったフレームをすべて並び替えることができます。

参照

Testing trading strategies、Working with optimization results、OnTesterInit、OnTesterDeinit、FrameFirst、FrameFilter、FrameNext、FrameInputs

© 2000–2025, MetaQuotes Ltd.

市場情報の取得

これらは、市場の状態に関する情報を受信するために意図された関数です。

関数	アクション
SymbolsTotal	(「気配値表示」で選択されたまたは全ての) 利用可能なシンボルの数を返します。
SymbolExist	指定された名称の銘柄が存在するかどうかを確認します
SymbolName	指定されたシンボルの名称を返します。
SymbolSelect	「気配値表示」ウィンドウでシンボルを選択したり、ウィンドウからシンボルを削除したりします。
SymbolIsSynchronized	端末で選択されたシンボルのデータがレードサーバ—上のデータと同期されているかどうかをチェックします。
SymbolInfoDouble	シンボルプロパティの double 値を返します。
SymbolInfoInteger	指定されたシンボルの整数型 (long、datetime、int または bool) プロパティの値を返します。
SymbolInfoString	対応するプロパティに指定されたシンボルの文字列型の値を返します。
SymbolInfoMarginRate	注文の種類と方向に応じて補償金率を返します。
SymbolInfoTick	指定されたシンボルの現在の価格をMqlTick 型の変数として返します。
SymbolInfoSessionQuote	指定された銘柄と曜日での指定された相場セッションの開始と終了の時間を受け取ることができます。
SymbolInfoSessionTrade	指定された銘柄と曜日での指定された相場セッションの開始と終了の時間を受け取ることができます。
MarketBookAdd	選択したシンボルの板情報のオープンを提供し、DOMの変更の通知を受信にサブスクライブします。
MarketBookRelease	指定されたシンボルの板情報の閉鎖を提供し、DOMの変更の通知受信のサブスクライブをキャンセルします。
MarketBookGet	指定されたシンボルの板情報を含む MqlBookInfo 構造体の配列を返します。

© 2000–2025, MetaQuotes Ltd.

SymbolsTotal

(「気配値表示」で選択されたまたは全ての) 利用可能なシンボルの数を返します。

```
int  SymbolsTotal(
   bool  selected      // True - 「気配値表示」のシンボルのみ
   );
```

パラメータ

selected

[in] リクエストモード (true または false)

戻り値

selected パラメータが true の場合、この関数は「気配値表示」内で選択されたシンボルの数を返します。値が false の場合、全てのシンボルの数を返します。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 利用可能なシンボルの数を取得する
   int total    = SymbolsTotal(false);       // all symbols on the server
   int selected = SymbolsTotal(true);        // all symbols added to the Market
Watch window

//--- 取得したデータを操作ログに送信する
   PrintFormat("Number of available symbols on the server: %d\n"+
               "Number of available symbols selected in MarketWatch: %d", total,
selected);
   /*
  結果:
   Number of available symbols on the server: 140
   Number of available symbols selected in MarketWatch: 11
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

SymbolExist

指定された名称の銘柄が存在するかどうかを確認します。

```
bool  SymbolExist(
   const string  name,     // 銘柄名
   bool&   is_custom       // カスタム銘柄プロパティ
   );
```

パラメータ

name

[in] 銘柄名

is_custom

[out] 正常に実行されると設定されるカスタム銘柄プロパティ。trueの場合、検知された銘柄はカスタム銘柄です。

戻り値

falseの場合、銘柄は標準銘柄とカスタム銘柄のリストで見つかりません。

例：

```
#define SYMBOL_NAME "GBPUSDn"
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- カスタムシンボルフラグを宣言し、SYMBOL_ NAMEで指定された名前を持つシンボルの存在を確認する
   bool custom = false;
   bool result = SymbolExist(SYMBOL_NAME, custom);

//--- デフォルトの「シンボルが見つかりません」メッセージテキストを宣言する
   string text = StringFormat("The symbol '%s' was not found among either the
standard or custom symbols.", SYMBOL_NAME);

//--- シンボルが見つかった場合、シンボルが見つかったリストに応じてメッセージテキストを作成する
   if(result)
     {
      //--- これが標準シンボルの場合
      if(!custom)
         text = StringFormat("The '%s' symbol is available on the server.",
SYMBOL_NAME);
      //--- これがカスタムシンボルの場合
      else
         text = StringFormat("The symbol '%s' was found in the list of custom
symbols.", SYMBOL_NAME);
     }

//--- チェック結果に関するメッセージを操作ログに送信する
```

© 2000-2025, MetaQuotes Ltd.

```
   Print(text);
   /*
   result for standard 'GBPUSD' symbol:
   The 'GBPUSD' symbol is available on the server.

   result for custom 'GBPUSDx' symbol:
   The symbol 'GBPUSDx' was found in the list of custom symbols.

   result for missing 'GBPUSDn' symbol:
   The symbol 'GBPUSDn' was not found among either the standard or custom symbols.
   */
  }
```

参照

SymbolsTotal、SymbolSelect、カスタム銘柄

© 2000–2025, MetaQuotes Ltd.

SymbolName

シンボルの名称を返します。

```
string  SymbolName(
   int    pos,           // リスト内での番号
   bool   selected       // 「気配値表示」内のシンボルでは true
   );
```

パラメータ

pos

[in] シンボルの番号

selected

[in] リクエストモード値が true の場合、シンボルは、「気配値表示」で選択されたシンボルリストから取得されます。値がfalseの場合、シンボルは通常のリストから取得されます。

戻り値

銘柄名を示す string 型の値

例：

```
#define SYMBOL_NAME "GBPHKD"

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- サーバーシンボル検索結果のフラグを設定する
   bool found = false;

//--- すべてのサーバーシンボルのリストでSYMBOL_NAMEシンボルを検索する
   int total = SymbolsTotal(false);
   for(int i=0; i<total; i++)
     {
      //--- ループインデックスでリスト内のシンボル名を取得する
      string name = SymbolName(i, false);

      //--- これが目的のシンボルである場合、その名前とリスト内の位置を操作ログに送信し、ループを終了する
      if(name == SYMBOL_NAME)
        {
         PrintFormat("The '%s' symbol was found in the list of server symbols at
index %d", name, i);
         found = true;
         break;
        }
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- サーバー上にシンボルが見つからない場合は、シャットダウンする前にこれを報告する
   if(!found)
      PrintFormat("The '%s' symbol was not found on the server.", SYMBOL_NAME);
  }
```

© 2000–2025, MetaQuotes Ltd.

SymbolSelect

「気配値表示」ウィンドウでシンボルを選択したり、ウィンドウからシンボルを削除したりします。

```
bool  SymbolSelect(
   string  name,        // 銘柄名
   bool    select       // 追加または削除
   );
```

パラメータ

name

[in] 銘柄名

select

[in] スイッチ。値がfalseの場合、シンボルは「気配値表示」から削除されるべきです。でなければ、シンボルはウィンドウで選択されるべきです。シンボルチャートが開いている、またはシンボルのための建玉がある場合は、シンボルは削除することが出来ません。

戻り値

失敗の場合 false

例：

```
#define SYMBOL_NAME "GBPHKD"

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- リストにシンボルが存在するかどうかを確認し、見つからない場合は報告して作業を完了する
   bool custom = false;
   if(!SymbolExist(SYMBOL_NAME, custom))
     {
      PrintFormat("'%s' symbol not found in the lists", SYMBOL_NAME);
      return;
     }

//--- 気配値ウィンドウにシンボルを追加する
   ResetLastError();
   if(!SymbolSelect(SYMBOL_NAME, true))
     {
      Print("SymbolSelect() failed. Error ", GetLastError());
      return;
     }
//--- シンボルがリストに正常に追加された場合は、気配値表示ウィンドウでそのインデックスを取得し、結果を操作ログに送信する
   int index = SymbolIndex(SYMBOL_NAME);
   PrintFormat("The '%s' symbol has been added to the MarketWatch list. Symbol index in the list: %d", SYMBOL_NAME, index);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 気配値表示ウィンドウからシンボルを削除する
   ResetLastError();
   if(!SymbolSelect(SYMBOL_NAME, false))
     {
      Print("SymbolSelect() failed. Error ", GetLastError());
      return;
     }
//--- シンボルがリストに正常に削除された場合は、気配値表示ウィンドウでのインデックスは-1で、削除結果を操作ログに送信する
   index = SymbolIndex(SYMBOL_NAME);
   PrintFormat("The '%s' symbol has been removed from the MarketWatch list. Symbol index in the list: %d", SYMBOL_NAME, index);

   /*
  結果:
   The 'GBPHKD' symbol has been added to the MarketWatch list. Symbol index in the list: 12
   The 'GBPHKD' symbol has been removed from the MarketWatch list. Symbol index in the list: -1
   */
  }
//+------------------------------------------------------------------+
//|  気配値表示シンボルリスト内のシンボルインデックスを返す          |
//+------------------------------------------------------------------+
int SymbolIndex(const string symbol)
  {
   int total = SymbolsTotal(true);
   for(int i=0; i<total; i++)
     {
      string name = SymbolName(i, true);
      if(name == symbol)
         return i;
     }
   return(WRONG_VALUE);
  }
```

© 2000–2025, MetaQuotes Ltd.

SymbolIsSynchronized

この関数は、端末で選択されたシンボルのデータがレードサーバ—上のデータと同期化されているかどうかをチェックします。

```
bool  SymbolIsSynchronized(
   string  name,        // 銘柄名
   );
```

パラメータ

name

[in] 銘柄名

戻り値

データが同期化されている場合は true、それ以外の場合は false

例：

```
#define SYMBOL_NAME "EURUSD"

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- SYMBOL_NAMEシンボルを使用して、ターミナル内のデータをサーバーデータと同期するためのフラグを取得する
   bool    sync = SymbolIsSynchronized(SYMBOL_NAME);

//--- 同期フラグに応じてメッセージを作成する
   string text = StringFormat("The data on the '%s' symbol in the terminal is
synchronized with the data on the trading server.", SYMBOL_NAME);
   if(!sync)
      text = StringFormat("The data for the '%s' symbol in the terminal is not
synchronized with the data on the trading server.", SYMBOL_NAME);

//--- 取得した結果を操作ログに送信する
   Print(text);

   /*
   同期データの結果：
   The data on the 'EURUSD' symbol in the terminal is synchronized with the data on
the trading server.

   非同期データの結果：
   The data for the 'GBPHKD' symbol in the terminal is not synchronized with the
data on the trading server.
   */
  }
```

参照

© 2000–2025, MetaQuotes Ltd.

SymbolInfoInteger、データアクセスの整 理

© 2000–2025, MetaQuotes Ltd.

SymbolInfoDouble

指定されたシンボルの指定されたプロパティを返します。この関数には 2 つのバージョンがあります。

1. すぐにプロパティ値を返します。

```
double  SymbolInfoDouble(
   string                    name,        // シンボル
   ENUM_SYMBOL_INFO_DOUBLE   prop_id      // プロパティの識別子
   );
```

2. 関数実行の成功に応じて true または false を返す成功した場合には、プロパティの値は、最後のパラメータで参照によって受け渡され、変数に配置されます。

```
bool  SymbolInfoDouble(
   string                    name,        // シンボル
   ENUM_SYMBOL_INFO_DOUBLE   prop_id,     // プロパティの識別子
   double&                   double_var   // プロパティ値を受け取る
   );
```

パラメータ

name

[in] 銘柄名

prop_id

[in] シンボルプロパティの識別子（値は ENUM_SYMBOL_INFO_DOUBLE 列挙のいずれか）

double_var

[out] リクエストされたプロパティの double 型の値を受け取る変数

戻り値

double 型の値実行が失敗した場合、エラー情報は GetLastError() 関数で取得出来ます。

- 5040 – 銘柄名指定での不正な文字列パラメータの使用
- 4301 – 未知のシンボル（銘柄）
- 4302 – シンボルが「気配値表示」で未選択（利用可能なシンボルのリストに不在）
- 4303 – 不正なシンボルプロパティ識別子

注意事項

最後のティック情報取得には SymbolInfoTick() の使用が推奨されます。端末が取引口座に接続されてから、相場が一つも出ていない可能性があります。このような場合には、リクエストされた値は不定となります。

ほとんどの場合は SymbolInfoTick() 関数の1 回の呼び出しで、ユーザの買値、売値、最終値、ボリューム、また最後のティックの到着時間の受信を可能にするので充分です。

SymbolInfoMarginRate()関数は、注文の種類と方向に応じて必要な証拠金額に関するデータを提供します。

例:

```
void OnTick()
```

© 2000–2025, MetaQuotes Ltd.

```
  {
//--- シンボルプロパティのスプレッドを取得
   bool spreadfloat=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD_FLOAT);
   string comm=StringFormat("Spread %s = %I64d points\r\n",
                            spreadfloat?"floating":"fixed",
                            SymbolInfoInteger(Symbol(),SYMBOL_SPREAD));
//--- スプレッドを自分で計算する
   double ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
   double bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   double spread=ask-bid;
   int spread_points=(int)MathRound(spread/SymbolInfoDouble(Symbol(),SYMBOL_POINT));
   comm=comm+"Calculated spread = "+(string)spread_points+" points";
   Comment(comm);
  }
```

© 2000–2025, MetaQuotes Ltd.

SymbolInfoInteger

指定されたシンボルの指定されたプロパティを返します。この関数には 2 つのバージョンがあります。

1. すぐにプロパティ値を返します。

```
long  SymbolInfoInteger(
   string                     name,       // シンボル
   ENUM_SYMBOL_INFO_INTEGER   prop_id     // プロパティの識別子

   );
```

2. 関数実行の成功に応じて true または false を返す成功した場合には、プロパティの値は、最後のパラメータで参照によって受け渡され、変数に配置されます。

```
bool  SymbolInfoInteger(
   string                     name,       // シンボル
   ENUM_SYMBOL_INFO_INTEGER   prop_id,    // プロパティの識別子
   long&                      long_var    // プロパティ値を受け取る
   );
```

パラメータ

name

[in] 銘柄名

prop_id

[in] シンボルプロパティの識別子（値は ENUM_SYMBOL_INFO_INTEGER 列挙のいずれか）

long_var

[out] リクエストされたプロパティの long 型の値を受け取る変数

戻り値

long 型の値実行が失敗した場合、エラー情報は GetLastError() 関数で取得出来ます。

- 5040 – 銘柄名指定での不正な文字列パラメータの使用
- 4301 – 未知のシンボル（銘柄）
- 4302 – シンボルが「気配値表示」で未選択（利用可能なシンボルのリストに不在）
- 4303 – 不正なシンボルプロパティ識別子

注意事項

最後のティック情報取得には SymbolInfoTick() の使用が推奨されます。端末が取引口座に接続されてから、相場が一つも出ていない可能性があります。このような場合には、リクエストされた値は不定となります。

ほとんどの場合は SymbolInfoTick() 関数の1 回の呼び出しで、ユーザの買値、売値、最終値、ボリューム、また最後のティックの到着時間の受信を可能にするので充分です。

例:

```
void OnTick()
  {
//--- シンボルプロパティのスプレッドを取得
```

© 2000–2025, MetaQuotes Ltd.

```
   bool spreadfloat=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD_FLOAT);
   string comm=StringFormat("Spread %s = %I64d points\r\n",
                            spreadfloat?"floating":"fixed",
                            SymbolInfoInteger(Symbol(),SYMBOL_SPREAD));
//--- スプレッドを自分で計算する
   double ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
   double bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   double spread=ask-bid;
   int spread_points=(int)MathRound(spread/SymbolInfoDouble(Symbol(),SYMBOL_POINT));
   comm=comm+"Calculated spread = "+(string)spread_points+" points";
   Comment(comm);
  }
```

© 2000–2025, MetaQuotes Ltd.

SymbolInfoString

指定されたシンボルの指定されたプロパティを返します。この関数には 2 つのバージョンがあります。

1. すぐにプロパティ値を返します。

```
string  SymbolInfoString(
   string                    name,         // シンボル
   ENUM_SYMBOL_INFO_STRING   prop_id       // プロパティ識別子
   );
```

2. 関数の成功に応じて、true または false を返します。成功した場合、プロパティの値は、参照によって渡された最後のパラメータに配置されます。

```
bool  SymbolInfoString(
   string                    name,         // シンボル
   ENUM_SYMBOL_INFO_STRING   prop_id,      // プロパティ識別子
   string&                   string_var    // プロパティ値を受け取る
   );
```

パラメータ

name

[in] 銘柄名

prop_id

[in] シンボルプロパティの識別子（値は ENUM_SYMBOL_INFO_STRING 列挙のいずれか）

string_var

[out] リクエストされたプロパティ値を受け取るstring 型の変数

戻り値

string 型の値実行が失敗した場合、エラー情報は GetLastError() 関数で取得出来ます。

- 5040 – 銘柄名指定での不正な文字列パラメータの使用
- 4301 – 未知のシンボル（銘柄）
- 4302 – シンボルが「気配値表示」で未選択（利用可能なシンボルのリストに不在）
- 4303 – 不正なシンボルプロパティ識別子

注意事項

最後のティック情報取得には SymbolInfoTick() の使用が推奨されます。端末が取引口座に接続されてから、相場が一つも出ていない可能性があります。このような場合には、リクエストされた値は不定となります。

ほとんどの場合は SymbolInfoTick() 関数の1 回の呼び出しで、ユーザの買値、売値、最終値、ボリューム、また最後のティックの到着時間の受信を可能にするので充分です。

例：

```
#define SYMBOL_NAME "USDJPY"

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
```

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- SYMBOL_NAMEシンボルの文字列データを取得する
   string currency_base   = SymbolInfoString(SYMBOL_NAME,
SYMBOL_CURRENCY_BASE);     // Base currency of the symbol
   string currency_profit = SymbolInfoString(SYMBOL_NAME,
SYMBOL_CURRENCY_PROFIT);   // Profit currency
   string currency_margin = SymbolInfoString(SYMBOL_NAME,
SYMBOL_CURRENCY_MARGIN);   // Margin currency
   string symbol_descript = SymbolInfoString(SYMBOL_NAME,
SYMBOL_DESCRIPTION);       // String description of the symbol

//--- 取得したデータを使用してメッセージテキストを作成する
   string text=StringFormat("Symbol %s:\n"+
                            "- Currency Base: %s\n"+
                            "- Currensy Profit: %s\n"+
                            "- Currency Margin: %s\n"+
                            "- Symbol Description: %s",
                            SYMBOL_NAME, currency_base,
                            currency_profit, currency_margin,
                            symbol_descript);

//--- 取得したデータを操作ログに送信する
   Print(text);
   /*
   Symbol USDJPY:
   - Currency Base: USD
   - Currensy Profit: JPY
   - Currency Margin: USD
   - Symbol Description: US Dollar vs Yen
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

SymbolInfoMarginRate

注文の種類と方向に応じて証拠金率を返します。

```
bool  SymbolInfoMarginRate(
   string              name,                    // 銘柄名
   ENUM_ORDER_TYPE     order_type,              // 注文の種類
   double&             initial_margin_rate,     // 初期の証拠金率
   double&             maintenance_margin_rate  // 維持証拠金率
   );
```

パラメータ

name

[in] 銘柄名

order_type

[in] 注文の種類

initial_margin_rate

[in] 初期の証拠金率を受け取る double 型の値。当初証拠金は、適切な方向での 1 ロットの証拠金です。当初証拠金によってレートを倍増し、指定された種類の注文をする際に、口座に保留される資金の額を受け取ります。

maintenance_margin_rate

[out] 維持証拠金率を受け取る double 型の変数。維持証拠金は、適切な方向に1ロットの建玉の維持に必要な最小限の資金です。維持証拠金によってレートを倍増し、指定された種類の注文を有効にする際に、口座に保留される資金の額を受け取ります。

戻り値

プロパティリクエストが成功の場合は true、それ以外の場合は false

例：

```
#define SYMBOL_NAME Symbol()

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 比率が設定される変数を宣言する
   double initial_margin_rate = 0;      // initial margin rate
   double maintenance_margin_rate = 0; // maintenance margin rate

//--- 買い市場注文のSYMBOL_NAMEシンボル比率を取得して操作ログに送信する
   SymbolInfoMarginRatePrint(SYMBOL_NAME, ORDER_TYPE_BUY, initial_margin_rate,
maintenance_margin_rate);

//--- 売り市場注文のSYMBOL_NAMEシンボル比率を取得して操作ログに送信する
```

© 2000-2025, MetaQuotes Ltd.

```
   SymbolInfoMarginRatePrint(SYMBOL_NAME, ORDER_TYPE_SELL, initial_margin_rate,
maintenance_margin_rate);

   /*
  結果:
   Initial margin rate for symbol AFKS for the Buy market order: 0.225600
   Maintenance margin rate for symbol AFKS for the Buy market order: 0.112800
   Initial margin rate for symbol AFKS for the Sell market order: 0.254400
   Maintenance margin rate for symbol AFKS for the Sell market order: 0.127200
   */
  }
//+------------------------------------------------------------------+
//|  証拠金比率を操作ログに送信する                                  |
//+------------------------------------------------------------------+
void SymbolInfoMarginRatePrint(const string symbol, const ENUM_ORDER_TYPE
order_type, double &initial_margin_rate, double &maintenance_margin_rate)
  {
//--- order_type成行注文のsymbol比率を取得する
   ResetLastError();
   if(!SymbolInfoMarginRate(symbol, order_type, initial_margin_rate,
maintenance_margin_rate))
     {
      Print("SymbolInfoMarginRate() failed. Error ", GetLastError());
      return;
     }
//--- 取得した比の値を出力する
   string type=(order_type==ORDER_TYPE_BUY ? "Buy" : order_type==ORDER_TYPE_SELL ?
"Sell" : "Unknown");
   PrintFormat("Initial margin rate for symbol %s for the %s market order: %f\n"+
               "Maintenance margin rate for symbol %s for the %s market order: %f",
               SYMBOL_NAME, type, initial_margin_rate,
               SYMBOL_NAME, type, maintenance_margin_rate);
  }
```

© 2000–2025, MetaQuotes Ltd.

SymbolInfoTick

指定されたシンボルの現在の価格を MqlTick 型の変数として返します。

```
bool  SymbolInfoTick(
   string    symbol,     // 銘柄名
   MqlTick&  tick        // 構造体への参照
   );
```

パラメータ

symbol

[in] 銘柄名

tick

[out] 現在価格と直近約定更新時刻が配置されるMqlTick 型の構造体へのリンク

戻り値

成功の場合は true、それ以外の場合は false

例：

```
#define SYMBOL_NAME "EURUSD"

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 次元1のMql Tick構造体型の配列を宣言する
   MqlTick tick[1]={};

//--- SYMBOL_NAMEシンボルの最新価格をMql Tick構造体に取得する
   if(!SymbolInfoTick(SYMBOL_NAME, tick[0]))
     {
      Print("SymbolInfoTick() failed. Error ", GetLastError());
      return;
     }

//--- 取得したデータを操作ログに送信する
   PrintFormat("Latest price data for the '%s' symbol:", SYMBOL_NAME);
   ArrayPrint(tick);
   /*
  結果:
   Latest price data for the 'EURUSD' symbol:
                    [time]    [bid]    [ask] [last] [volume]    [time_msc] [flags]
[volume_real]
   [0] 2024.05.17 23:58:54 1.08685 1.08695 0.0000        0 1715990334319       6
  0.00000
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

SymbolInfoSessionQuote

指定された銘柄と曜日での指定された相場セッションの開始と終了の時間を受け取ることができます。

```
bool  SymbolInfoSessionQuote(
   string            name,                // 銘柄名
   ENUM_DAY_OF_WEEK  day_of_week,         // 曜日
   uint              session_index,       // セッションインデックス
   datetime&         from,                // セッション開始時
   datetime&         to                   // セッション終了時
   );
```

パラメータ

name

[in] 銘柄名

ENUM_DAY_OF_WEEK

[in] 曜日。ENUM_DAY_OF_WEEK 列挙の値

uint

[in] 開始と終了時刻を受け取りたいセッションの序数。セッションの番号付けは 0 から始まります。

from

[out] 00 時 00 分から経過した秒数でのセッション開始時刻。戻り値の date は無効。

to

[out] 00 時 00 分から経過した秒数でのセッション終了時刻。戻り値の date は無効。

戻り値

指定されたセッションのデータ、銘柄と曜日が受信された場合は true、それ以外の場合は false

例：

```
#define SYMBOL_NAME    Symbol()
#define SESSION_INDEX 0

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- シンボルとSESSION_INDEXを含むヘッダーを出力し、
//--- 月曜日から金曜日までの曜日ごとのループで、操作ログの相場セッションの開始時刻と終了時刻を出力する
   PrintFormat("Symbol %s, Quote session %d:", SYMBOL_NAME, SESSION_INDEX);
   for(int i=MONDAY; i<SATURDAY; i++)
      SymbolInfoSessionQuotePrint(SYMBOL_NAME, (ENUM_DAY_OF_WEEK)i, SESSION_INDEX);
   /*
  結果:
   Symbol RU000A103661, Quote session 0:
   - Monday      06:45 - 00:00
```

© 2000-2025, MetaQuotes Ltd.

```
   - Tuesday    06:45 - 00:00
   - Wednesday  06:45 - 00:00
   - Thursday   06:45 - 00:00
   - Friday     06:45 - 00:00
   */
  }
//+------------------------------------------------------------------+
//| 指定されたシンボルと曜日の指定された相場セッションの                |
//| 開始時刻と終了時刻を操作ログに送信する                            |
//+------------------------------------------------------------------+
void SymbolInfoSessionQuotePrint(const string symbol, const ENUM_DAY_OF_WEEK
day_of_week, const uint session_index)
  {
//--- 相場セッションの開始と終了を記録する変数を宣言する
   datetime date_from;  // セッションの開始時刻
   datetime date_to;    // セッションnお終了時刻

//--- 銘柄と曜日別に相場セッションからデータを取得する
   if(!SymbolInfoSessionQuote(symbol, day_of_week, session_index, date_from,
date_to))
     {
      Print("SymbolInfoSessionQuote() failed. Error ", GetLastError());
      return;
     }

//--- 列挙定数から曜日名を作成する
   string week_day=EnumToString(day_of_week);
   if(week_day.Lower())
      week_day.SetChar(0, ushort(week_day.GetChar(0)-32));

//--- 指定された相場セッションのデータを操作ログに送信する
   PrintFormat("- %-10s %s - %s", week_day, TimeToString(date_from, TIME_MINUTES),
TimeToString(date_to, TIME_MINUTES));
  }
```

参照

シンボルプロパティ、TimeToStruct、データ構造体

© 2000-2025, MetaQuotes Ltd.

SymbolInfoSessionTrade

指定された銘柄と曜日での指定された相場セッションの開始と終了の時間を受け取ることができます。

```
bool  SymbolInfoSessionTrade(
   string            name,                // 銘柄名
   ENUM_DAY_OF_WEEK  day_of_week,         // 曜日
   uint              session_index,       // セッションインデックス
   datetime&         from,                // セッション開始時刻
   datetime&         to                   // セッション終了時刻
   );
```

パラメータ

name

[in] 銘柄名

ENUM_DAY_OF_WEEK

[in] 曜日。ENUM_DAY_OF_WEEK 列挙の値

uint

[in] 開始と終了時刻を受け取りたいセッションの序数。セッションの番号付けは 0 から始まります。

from

[out] 00 時 00 分から経過した秒数でのセッション開始時刻。戻り値の date は無効。

to

[out] 00 時 00 分から経過した秒数でのセッション終了時刻。戻り値の date は無効。

戻り値

指定されたセッションのデータ、銘柄と曜日が受信された場合は true、それ以外の場合は false

例：

```
#define SYMBOL_NAME    Symbol()
#define SESSION_INDEX 0

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- シンボルとSESSION_INDEXを含むヘッダーを出力し、
//--- 月曜日から金曜日までの曜日ごとのループで、取引セッションの開始時刻と終了時刻を操作ログに出力する
   PrintFormat("Symbol %s, Trade session %d:", SYMBOL_NAME, SESSION_INDEX);
   for(int i=MONDAY; i<SATURDAY; i++)
      SymbolInfoSessionTradePrint(SYMBOL_NAME, (ENUM_DAY_OF_WEEK)i, SESSION_INDEX);
   /*
  結果:
   Symbol GBPUSD, Trade session 0:
   - Monday      00:15 - 23:55
```

© 2000-2025, MetaQuotes Ltd.

```
   - Tuesday    00:15 - 23:55
   - Wednesday  00:15 - 23:55
   - Thursday   00:15 - 23:55
   - Friday     00:15 - 23:55
   */
  }
//+------------------------------------------------------------------+
//| 指定された取引セッションの開始時刻と終了時刻を送信する                  |
//| 開始時刻と終了時刻を操作ログに送信する                              |
//+------------------------------------------------------------------+
void SymbolInfoSessionTradePrint(const string symbol, const ENUM_DAY_OF_WEEK
day_of_week, const uint session_index)
  {
//--- 取引セッションの開始と終了を記録する変数を宣言する
   datetime date_from;  // セッションの開始時刻
   datetime date_to;    // セッションnお終了時刻

//--- 銘柄と曜日別に取引セッションからデータを取得する
   if(!SymbolInfoSessionTrade(symbol, day_of_week, session_index, date_from,
date_to))
     {
      Print("SymbolInfoSessionTrade() failed. Error ", GetLastError());
      return;
     }

//--- 列挙定数から曜日名を作成する
   string week_day=EnumToString(day_of_week);
   if(week_day.Lower())
      week_day.SetChar(0, ushort(week_day.GetChar(0)-32));

//--- 指定された取引セッションのデータを操作ログに送信する
   PrintFormat("- %-10s %s - %s", week_day, TimeToString(date_from, TIME_MINUTES),
TimeToString(date_to, TIME_MINUTES));
  }
```

参照

シンボルプロパティ、TimeToStruct、データ構造体

© 2000-2025, MetaQuotes Ltd.

MarketBookAdd

選択されたシンボルの板情報のオープンを提供し、DOMの変更の通知を受信にサブスクライブします。

```
bool  MarketBookAdd(
   string  symbol      // シンボル
   );
```

パラメータ

symbol

[in] エキスパートアドバイザーまたはスクリプトで板情報が使用されるシンボルの名称

戻り値

オープンが成功の場合は true、それ以外の場合は false

注意事項

通常、この関数は OnInit() 関数またはクラスコンストラクタで呼ばれるべきです。着信アラートを処理するには、エキスパートアドバイザープログラムに void OnBookEvent(string& symbol) 関数が含まれている必要があります。

例：

```
#define    SYMBOL_NAME    "GBPUSD"

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- SYMBOL_NAMEシンボルの板情報を開く
   if(!MarketBookAdd(SYMBOL_NAME))
     {
      PrintFormat("MarketBookAdd(%s) failed. Error ", SYMBOL_NAME, GetLastError());
      return;
     }

//--- 板情報を正常に開いたというメッセージを操作ログに送信する
   PrintFormat("The MarketBook for the '%s' symbol was successfully opened and a
subscription to change it was received", SYMBOL_NAME);

//--- 2秒待つ
   Sleep(2000);

//--- 完了したら、開いた板情報から登録解除する
   ResetLastError();
   if(MarketBookRelease(SYMBOL_NAME))
      PrintFormat("MarketBook for the '%s' symbol was successfully closed",
SYMBOL_NAME);
   else
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("Error %d occurred when closing MarketBook using the '%s' symbol",
GetLastError(), SYMBOL_NAME);


   /*
  結果:
   The MarketBook for the 'GBPUSD' symbol was successfully opened and a subscription
 to change it was received
   MarketBook for the 'GBPUSD' symbol was successfully closed
   */
  }
```

参照

板情報の構造体、構造体とクラス

© 2000–2025, MetaQuotes Ltd.

MarketBookRelease

指定されたシンボルの板情報の閉鎖を提供し、DOMの変更の通知受信のサブスクライブをキャンセルします。

```
bool  MarketBookRelease(
   string  symbol      // シンボル
   );
```

パラメータ

symbol

[in] 銘柄名

戻り値

閉鎖が成功した場合はtrue、それ以外の場合はfalse

注意事項

通常、この関数は、対応するMarketBookAdd()がOnInit()関数から呼ばれた場合 OnInit()関数で呼ばれなければなりません。もしくは、対応するMarketBookAdd()関数がクラスコンストラクタから呼ばれた場合は、クラスデストラクタから呼ばれなければなりません。

例：

```
#define SYMBOL_NAME "GBPUSD"

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- SYMBOL_NAMEシンボルの板情報を開く
   if(!MarketBookAdd(SYMBOL_NAME))
     {
      PrintFormat("MarketBookAdd(%s) failed. Error ", SYMBOL_NAME, GetLastError());
      return;
     }

//--- 板情報を正常に開いたというメッセージを操作ログに送信する
   PrintFormat("The MarketBook for the '%s' symbol was successfully opened and a
subscription to change it was received", SYMBOL_NAME);

//--- 2秒待つ
   Sleep(2000);

//--- 完了したら、開いた板情報から登録解除する
//--- 板情報からの購読解除に成功したこと、または操作ログにエラーが発生したことに関するメッセージを送信する
   ResetLastError();
   if(MarketBookRelease(SYMBOL_NAME))
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("MarketBook for the '%s' symbol was successfully closed",
SYMBOL_NAME);
   else
      PrintFormat("Error %d occurred when closing MarketBook using the '%s' symbol",
GetLastError(), SYMBOL_NAME);


   /*
  結果:
   The MarketBook for the 'GBPUSD' symbol was successfully opened and a subscription
 to change it was received
   MarketBook for the 'GBPUSD' symbol was successfully closed
   */
  }
```

参照

板情報の構造体、構造体とクラス

© 2000–2025, MetaQuotes Ltd.

MarketBookGet

指定されたシンボルの板情報を含むMqlBookInfo構造体の配列を返します。

```
bool  MarketBookGet(
   string         symbol,      // シンボル
   MqlBookInfo&   book[]       // 配列への参照
   );
```

パラメータ

symbol

[in] 銘柄名

book[]

[in] 板情報記録の配列への参照配列は、事前に記録の数に充分なように割り当てることが出来ます。動的配列が予め動作メモリに割り当てられていない場合、クライアント端末は、配列自体を配布します。

戻り値

成功の場合はtrue、それ以外の場合はfalse

注意事項

板情報は予めMarketBookAdd()関数で開かれなければなりません。

例:

```
   MqlBookInfo priceArray[];
   bool getBook=MarketBookGet(NULL,priceArray);
   if(getBook)
     {
      int size=ArraySize(priceArray);
      Print("MarketBookInfo for ",Symbol());
      for(int i=0;i<size;i++)
        {
         Print(i+":",priceArray[i].price
               +"    Volume = "+priceArray[i].volume,
               " type = ",priceArray[i].type);
        }
     }
   else
     {
      Print("Could not get contents of the symbol DOM ",Symbol());
     }
```

参照

板情報の構造体、構造体とクラス

© 2000-2025, MetaQuotes Ltd.

経済指標カレンダー関数

このセクションでは、メタトレーダープラットフォームで直接利用できる経済指標カレンダーの操作関数を説明します。経済指標カレンダーは、マクロ経済指標、その発表日および重要度の説明を含む既製の「百科事典」です。関連するマクロ経済指標の値は、発行直後にメタトレーダープラットフォームに送信され、国、通貨、重要度ごとに必要な指標を視覚的に追跡できるタグとしてチャートに表示されます。

経済指標カレンダーを使用するためのすべての関数は、取引サーバ時間(TimeTradeServer)を使用します。つまり、MqlCalendarValue構造体の時刻と、CalendarValueHistoryByEvent/CalendarValueHistory関数は、ユーザの現地時間ではなく、取引サーバの時間帯で設定されます。

経済指標カレンダー関数を使用すると、必要な国/通貨の観点から、カスタム重要度基準に従って、着信したイベントの自動分析を実行できます。

関数	アクション
CalendarCountryById	IDによって国の説明を取得する
CalendarEventById	IDによってイベントの説明を取得する
CalendarValueById	IDによってイベントの値を取得する
CalendarCountries	カレンダーで利用可能な国名の配列を取得する
CalendarEventByCountry	指定された国コードでカレンダーで利用可能なすべてのイベントの説明の配列を取得する
CalendarEventByCurrency	指定された通貨でカレンダーで利用可能なすべてのイベントの説明の配列を取得する
CalendarValueHistoryByEvent	イベントIDによって、指定された期間のすべてのイベントの値の配列を取得する
CalendarValueHistory	指定された期間のすべてのイベントの値の配列を取得する(国や通貨で並び替え可能)
CalendarValueLastByEvent	指定されたchange_idで、カレンダーデータベースのステータスからイベントIDの配列をIDで取得する
CalendarValueLast	指定されたchange_idで、カレンダデータベースのステータスからすべてのイベントの値の配列を取得する(国や通貨で並び替え可能)

© 2000-2025, MetaQuotes Ltd.

CalendarCountryById

IDによって国の説明を取得します。

```
bool  CalendarCountryById(
   const long           country_id,      // 国ID
   MqlCalendarCountry&  country          // 国の説明を受け取るための配列
   );
```

パラメータ

country_id

[in] Country ID (ISO 3166-1).

country

[out] 国の説明を受け取るためのMqlCalendarCountry型変数

戻り値

成功の場合はtrue、それ以外の場合はfalse。エラー情報を取得するためにはGetLastError()関数を呼び出します。可能なエラー:

- 4001 – ERR_INTERNAL_ERROR(一般的なランタイムエラー)
- 5402 – ERR_CALENDAR_NO_DATA(無効な国)
- 5401 – ERR_CALENDAR_TIMEOUT(リクエスト制限時間の超過)

例:

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 経済指標カレンダーから国のリストを取得する
   MqlCalendarCountry countries[];
   int count=CalendarCountries(countries);
//--- 結果を確認する
   if(count==0)
      PrintFormat("CalendarCountries() returned 0!Error %d",GetLastError());
//--- 2国以上が存在する場合
   if(count>=2)
     {
      MqlCalendarCountry country;
      //--- IDによって国の説明を取得する
      if(CalendarCountryById(countries[1].id, country))
        {
         //--- 国の説明を準備する
         string descr="id = "+IntegerToString(country.id)+"\n";
         descr+=("name = " + country.name+"\n");
         descr+=("code = " + country.code+"\n");
         descr+=("currency = " + country.currency+"\n");
         descr+=("currency_symbol = " + country.currency_symbol+"\n");
```

© 2000–2025, MetaQuotes Ltd.

```
         descr+=("url_name = " + country.url_name);
         //--- 国の説明を表示する
         Print(descr);
        }
      else
         Print("CalendarCountryById() failed. Error ",GetLastError());
     }
//---
  }
/*
  結果:
   id = 999
   name = European Union
   code = EU
   currency = EUR
   currency_symbol = €
   url_name = european-union
*/
```

参照

CalendarCountries、CalendarEventByCountry

© 2000–2025, MetaQuotes Ltd.

CalendarEventById

IDによってイベントの説明を取得します。

```
bool  CalendarEventById(
   ulong                event_id,      // イベントID
   MqlCalendarEvent&    event          // イベントの説明を受け取るための変数
   );
```

パラメータ

event_id

[in] イベントID

event

[out] イベントの説明を受け取るためのMqlCalendarEvent型の変数

戻り値

成功の場合はtrue、それ以外の場合はfalse。エラー情報を取得するためにはGetLastError()関数を呼び出します。可能なエラー:

- 4001 – ERR_INTERNAL_ERROR(一般的なランタイムエラー)
- 5402 – ERR_CALENDAR_NO_DATA(無効な国)
- 5401 – ERR_CALENDAR_TIMEOUT(リクエスト制限時間の超過)

例:

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- ドイツの国コード(ISO 3166-1 Alpha-2)
   string germany_code="DE";
//--- ドイツのイベントを取得する
   MqlCalendarEvent events[];
   int events_count=CalendarEventByCountry(germany_code,events);
//--- ドイツのイベントを操作ログに表示する
   if(events_count>0)
     {
      PrintFormat("Germany events: %d",events_count);
      ArrayPrint(events);
     }
   else
     {
      PrintFormat("Failed to receive events for the country code %s, error %d",
                  germany_code,GetLastError());
      //--- スクリプトが完了した
      return;
     }
//--- events[]配列の最後のイベントの説明を取得する
```

© 2000–2025, MetaQuotes Ltd.

```
   MqlCalendarEvent event;
   ulong event_id=events[events_count-1].id;
   if(CalendarEventById(event_id,event))
     {
      MqlCalendarCountry country;
      CalendarCountryById(event.country_id,country);
      PrintFormat("Event description with event_id=%d received",event_id);
      PrintFormat("Country: %s (country code = %d)",country.name,event.country_id);
      PrintFormat("Event name: %s",event.name);
      PrintFormat("Event code: %s",event.event_code);
      PrintFormat("Event importance: %
s",EnumToString((ENUM_CALENDAR_EVENT_IMPORTANCE)event.importance));
      PrintFormat("Event type: %s",EnumToString((ENUM_CALENDAR_EVENT_TYPE)
event.type));
      PrintFormat("Event sector: %s",EnumToString((ENUM_CALENDAR_EVENT_SECTOR)
event.sector));
      PrintFormat("Event frequency: %s",EnumToString((ENUM_CALENDAR_EVENT_FREQUENCY)
event.frequency));
      PrintFormat("Event release mode: %
s",EnumToString((ENUM_CALENDAR_EVENT_TIMEMODE)event.time_mode));
      PrintFormat("Event measurement unit: %
s",EnumToString((ENUM_CALENDAR_EVENT_UNIT)event.unit));
      PrintFormat("Number of decimal places: %d",event.digits);
      PrintFormat("Event multiplier: %
s",EnumToString((ENUM_CALENDAR_EVENT_MULTIPLIER)event.multiplier));
      PrintFormat("Source URL: %s",event.source_url);
     }
   else
      PrintFormat("Failed to get event description for event_d=%s, error %d",
                  event_id,GetLastError());
  }
/*
  結果:
  Germany events: 50
             [id] [type] [sector] [frequency] [time_mode] [country_id] [unit]
[importance] [multiplier] [digits]                                [source_url]
                  [event_code]                                        [name] [reserved]
   [ 0] 276010001      1        6           2           0          276      1
     1            0         1 "https://www.destatis.de/EN/Homepage.html"  "exports-
mm"                         "Exports m/m"                                    0
   [ 1] 276010002      1        6           2           0          276      1
     1            0         1 "https://www.destatis.de/EN/Homepage.html"  "imports-
mm"                         "Imports m/m"                                    0
   [ 2] 276010003      1        4           2           0          276      1
     1            0         1 "https://www.destatis.de/EN/Homepage.html"  "import-
price-index-mm"             "Import Price Index m/m"                       0
   [ 3] 276010004      1        4           2           0          276      1
     1            0         1 "https://www.destatis.de/EN/Homepage.html"  "import-
price-index-yy"             "Import Price Index y/y"                       0
   ....
   [47] 276500001      1        8           2           0          276      0
     2            0         1 "https://www.markiteconomics.com"           "markit-
manufacturing-pmi"          "Markit Manufacturing PMI"                     0
```

© 2000–2025, MetaQuotes Ltd.

```
   [48] 276500002      1        8           2           0          276       0
     2            0        1 "https://www.markiteconomics.com"           "markit-
services-pmi"               "Markit Services PMI"                      0
   [49] 276500003      1        8           2           0          276       0
     2            0        1 "https://www.markiteconomics.com"           "markit-
composite-pmi"              "Markit Composite PMI"                     0
  Event description with event_id=276500003 received
  Country: Germany (country code = 276)
  Event name: Markit Composite PMI
  Event code: markit-composite-pmi
   Event importance: CALENDAR_IMPORTANCE_MODERATE
   Event type: CALENDAR_TYPE_INDICATOR
   Event sector: CALENDAR_SECTOR_BUSINESS
   Event frequency: CALENDAR_FREQUENCY_MONTH
   Event release mode: CALENDAR_TIMEMODE_DATETIME
   Event measurement unit: CALENDAR_UNIT_NONE
  Number of decimal places: 1
   Value multiplier: CALENDAR_MULTIPLIER_NONE
   Source URL: https://www.markiteconomics.com
*/
```

参照

CalendarEventByCountry、CalendarEventByCurrency、CalendarValueById

© 2000–2025, MetaQuotes Ltd.

CalendarValueById

IDによってイベントの値を取得します。

```
bool  CalendarValueById(
   ulong                value_id,      // イベント値ID
   MqlCalendarValue&    value          // イベント値を受け取るための変数
   );
```

パラメータ

value_id

[in] イベント値ID

value

[out] イベントの説明を受け取るためのMqlCalendarValue型の変数。カレンダーイベント処理の例を参照してください。

戻り値

成功の場合はtrue、それ以外の場合はfalse。エラー情報を取得するためにはGetLastError()関数を呼び出します。可能なエラー:

- 4001 – ERR_INTERNAL_ERROR(一般的なランタイムエラー)
- 5402 – ERR_CALENDAR_NO_DATA(無効な国)
- 5401 – ERR_CALENDAR_TIMEOUT(リクエスト制限時間の超過)

注意事項

経済指標カレンダーを使用するためのすべての関数は、取引サーバ時間(TimeTradeServer)を使用します。つまり、MqlCalendarValue構造体の時刻と、CalendarValueHistoryByEvent/CalendarValueHistory関数は、ユーザの現地時間ではなく、取引サーバの時間帯で設定されます。

MqlCalendarValue構造体はactual_value、forecast_value、prev_value、revised_prev_valueフィールドの値を確認/設定するメソッドを備えています。値が指定されていない場合、フィールドにはLONG_MIN (-9223372036854775808)が格納されます。

これらのフィールドに格納されている値は100万倍されることに注意してください。CalendarValueById、CalendarValueHistoryByEvent、CalendarValueHistory、CalendarValueLastByEvent、CalendarValueLastのいずれかの関数を使用してMqlCalendarValueで値を受け取る場合、フィールド値がLONG_MINに等しいかどうかを確認する必要があります。フィールドに値が指定されている場合、値を取得するには、値を100万で除算する必要があります。値を取得するもう1つの方法は、MqlCalendarValue構造体の関数を使用して値を確認および取得することです。

例:

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 日本の国コード(ISO 3166-1 Alpha-2)
   string japan_code="JP";
//--- イベントを取得する期間の境界を設定する
```

© 2000–2025, MetaQuotes Ltd.

```
   datetime date_from=D'01.01.2018';  // 2018のすべてのイベントを取得する
   datetime date_to=0;                // 0はすべての既知イベント( 今後起こるものを含む)
//--- 日本のイベント値の配列を取得する
   MqlCalendarValue values[];
   int values_count=CalendarValueHistory(values,date_from,date_to,japan_code);
//--- 検知されたイベント値とともに移動する
   if(values_count>0)
     {
      PrintFormat("Number of values for Japan events: %d",values_count);
      //--- すべての「空の」値を削除する( actual_value==-9223372036854775808)
      for(int i=values_count-1;i>=0;i--)
        {
         if(values[i].actual_value==-9223372036854775808)
            ArrayRemove(values,i,1);
        }
      PrintFormat("Number of values after deleting empty ones: %
d",ArraySize(values));
     }
   else
     {
      PrintFormat("Failed to receive events for the country code %s, error %d",
                  japan_code,GetLastError());
      //--- スクリプトが完了した
      return;
     }
//--- values[]配列に10以上の値を残さない
   if(ArraySize(values)>10)
     {
      PrintFormat("Reduce the list of values to 10 and display them");
      ArrayRemove(values,0,ArraySize(values)-10);
     }
   ArrayPrint(values);

//--- 既知のvalue_idに基づいてイベント値の説明を取得する方法を表示する
   for(int i=0;i<ArraySize(values);i++)
     {
      MqlCalendarValue value;
      CalendarValueById(values[i].id,value);
      PrintFormat("%d: value_id=%d value=%d impact=%s",
                  i,values[i].id,value.actual_value,EnumToString
(ENUM_CALENDAR_EVENT_IMPACT(value.impact_type)));
     }
//---
  }
/*
  結果:
  Number of values for Japan events: 1734
  Number of values after deleting empty ones: 1017
  Reduce the list of values to 10 and display them
```

© 2000-2025, MetaQuotes Ltd.

```
        [id] [event_id]               [time]             [period] [revision]
[actual_value] [prev_value] [revised_prev_value] [forecast_value] [impact_type]
[reserved]
   [0] 56500  392030004 2019.03.28 23:30:00 2019.03.01 00:00:00          0
900000        600000 -9223372036854775808            500000              1          0
   [1] 56501  392030005 2019.03.28 23:30:00 2019.03.01 00:00:00          0
700000        700000 -9223372036854775808            700000              0          0
   [2] 56502  392030006 2019.03.28 23:30:00 2019.03.01 00:00:00          0
1100000       1100000 -9223372036854775808            900000              1          0
   [3] 56544  392030007 2019.03.28 23:30:00 2019.02.01 00:00:00          0
2300000       2500000 -9223372036854775808           2200000              2          0
   [4] 56556  392050002 2019.03.28 23:30:00 2019.02.01 00:00:00          0
1630000       1630000               1610000           1620000              1          0
   [5] 55887  392020003 2019.03.28 23:50:00 2019.02.01 00:00:00          0
400000        600000 -9223372036854775808           1300000              2          0
   [6] 55888  392020004 2019.03.28 23:50:00 2019.02.01 00:00:00          0       -
1800000      -3300000 -9223372036854775808          -2000000              1          0
   [7] 55889  392020002 2019.03.28 23:50:00 2019.02.01 00:00:00          0
200000      -2300000              -1800000            300000              2          0
   [8] 55948  392020006 2019.03.28 23:50:00 2019.02.01 00:00:00          1
1400000      -3400000 -9223372036854775808           -300000              1          0
   [9] 55949  392020007 2019.03.28 23:50:00 2019.02.01 00:00:00          1       -
1000000        300000 -9223372036854775808           -100000              2          0
  Display brief data on event values based on value_id
   0: value_id=56500 value=900000 impact=CALENDAR_IMPACT_POSITIVE
   1: value_id=56501 value=700000 impact=CALENDAR_IMPACT_NA
   2: value_id=56502 value=1100000 impact=CALENDAR_IMPACT_POSITIVE
   3: value_id=56544 value=2300000 impact=CALENDAR_IMPACT_NEGATIVE
   4: value_id=56556 value=1630000 impact=CALENDAR_IMPACT_POSITIVE
   5: value_id=55887 value=400000 impact=CALENDAR_IMPACT_NEGATIVE
   6: value_id=55888 value=-1800000 impact=CALENDAR_IMPACT_POSITIVE
   7: value_id=55889 value=200000 impact=CALENDAR_IMPACT_NEGATIVE
   8: value_id=55948 value=1400000 impact=CALENDAR_IMPACT_POSITIVE
   9: value_id=55949 value=-1000000 impact=CALENDAR_IMPACT_NEGATIVE
*/
```

参照

CalendarValueHistoryByEvent、CalendarValueHistory、CalendarValueLastByEvent、CalendarValueLast

© 2000–2025, MetaQuotes Ltd.

CalendarCountries

カレンダーで利用可能な国名の配列を取得します。

```
int  CalendarCountries(
   MqlCalendarCountry&  countries[]          // カレンダーの国の説明を受け取る配列
   );
```

パラメータ

countries[]

[out] カレンダーの国の説明を受け取るためのMqlCalendarCountry型の配列

戻り値

受け取られた説明の数。エラー情報を取得するためにはGetLastError()関数を呼び出します。可能なエラー:

- 4001 – ERR_INTERNAL_ERROR(一般的なランタイムエラー)
- 5401 – ERR_CALENDAR_TIMEOUT(リクエスト制限時間の超過)
- 5400 – ERR_CALENDAR_MORE_DATA(配列サイズはすべての国の説明を受け取るには不十分で、配列に収まるものだけが受信された)

例:

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 経済指標カレンダーから国のリストを取得する
   MqlCalendarCountry countries[];
   int count=CalendarCountries(countries);
//--- 操作ログに配列を表示する
   if(count>0)
      ArrayPrint(countries);
   else
      PrintFormat("CalendarCountries() returned 0!Error %d",GetLastError());
/*
  結果:
        [id]            [name] [code] [currency] [currency_symbol]        [url_name]
[reserved]
   [ 0]    0 "Worldwide"       "WW"   "ALL"      ""                  "worldwide"
        0
   [ 1]  999 "European Union" "EU"   "EUR"      "€"                 "european-union"
        0
   [ 2]  840 "United States"  "US"   "USD"      "$"                 "united-states"
        0
   [ 3]  124 "Canada"         "CA"   "CAD"      "$"                 "canada"
        0
   [ 4]   36 "Australia"      "AU"   "AUD"      "$"                 "australia"
        0
```

© 2000–2025, MetaQuotes Ltd.

```
    [ 5]  554 "New Zealand"    "NZ"   "NZD"      "$"              "new-zealand"
          0
    [ 6]  392 "Japan"          "JP"   "JPY"      "Ґ"              "japan"
          0
    [ 7]  156 "China"          "CN"   "CNY"      "Ґ"              "china"
          0
    [ 8]  826 "United Kingdom" "GB"   "GBP"      "J"              "united-kingdom"
          0
    [ 9]  756 "Switzerland"    "CH"   "CHF"      "₣"              "switzerland"
          0
    [10]  276 "Germany"        "DE"   "EUR"      "€"              "germany"
          0
    [11]  250 "France"         "FR"   "EUR"      "€"              "france"
          0
    [12]  380 "Italy"          "IT"   "EUR"      "€"              "italy"
          0
    [13]  724 "Spain"          "ES"   "EUR"      "€"              "spain"
          0
    [14]   76 "Brazil"         "BR"   "BRL"      "R$"             "brazil"
          0
    [15]  410 "South Korea"    "KR"   "KRW"      "₩"              "south-korea"
          0
*/
  }
```

参照

CalendarEventByCountry、CalendarCountryById

© 2000–2025, MetaQuotes Ltd.

CalendarEventByCountry

指定された国コードでカレンダーで利用可能なすべてのイベントの説明の配列を取得します。

```
int  CalendarEventByCountry(
   string               country_code,     // 国コード名(ISO 3166-1 alpha-2)
   MqlCalendarEvent&    events[]          // 説明配列を受け取るための変数
   );
```

パラメータ

country_code

[in] 国コード名(ISO 3166-1 alpha-2)

events[]

[out] 指定された国の説明を受け取るためのMqlCalendarEvent型配列

戻り値

受け取られた説明の数。エラー情報を取得するためにはGetLastError()関数を呼び出します。可能なエラー:

- 4001 – ERR_INTERNAL_ERROR(一般的なランタイムエラー)
- 4004 – ERR_NOT_ENOUGH_MEMORY(メモリが操作完了に不充分)
- 5401 – ERR_CALENDAR_TIMEOUT(リクエスト制限時間の超過)
- ArrayResize()操作の実行エラー

例:

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- EUの国コード(ISO 3166-1 Alpha-2)
   string EU_code="EU";
//--- EUイベントを取得する
   MqlCalendarEvent events[];
   int events_count=CalendarEventByCountry(EU_code,events);
//--- EUイベントを操作ログに表示する
   if(events_count>0)
     {
      PrintFormat("EU events: %d",events_count);
      ArrayPrint(events);
     }
//---
  }
/*
  結果:
  EU events: 56
             [id] [type]  [country_id] [unit] [importance] [multiplier] [digits]
[event_code]
```

© 2000–2025, MetaQuotes Ltd.

```
   [ 0] 999010001      0          999      0          2          0          0
"ECB Non-monetary Policy Meeting"
   [ 1] 999010002      0          999      0          2          0          0
"ECB Monetary Policy Meeting Account
   [ 2] 999010003      0          999      0          3          0          0
"ECB Monetary Policy Press Conferenc
   [ 3] 999010004      0          999      0          3          0          0
"ECB President Draghi Speech"
   [ 4] 999010005      0          999      0          2          0          0
"ECB Vice President Constancio Speec
   [ 5] 999010006      1          999      1          3          0          2
"ECB Deposit Facility Rate Decision"
   [ 6] 999010007      1          999      1          3          0          2
"ECB Interest Rate Decision"
   [ 7] 999010008      0          999      0          2          0          0
"ECB Economic Bulletin"
   [ 8] 999010009      1          999      2          2          3          3
"ECB Targeted LTRO"
   [ 9] 999010010      0          999      0          2          0          0
"ECB Executive Board Member Praet Sp
   [10] 999010011      0          999      0          2          0          0
"ECB Executive Board Member Mersch S
   ...

*/
```

参照

CalendarCountries、CalendarCountryById

© 2000–2025, MetaQuotes Ltd.

CalendarEventByCurrency

指定された通貨でカレンダーで利用可能なすべてのイベントの説明の配列を取得します。

```
int  CalendarEventByCurrency(
   const string          currency,      // 国の通貨のコード名
   MqlCalendarEvent&     events[]       // 説明配列を受け取るための変数
   );
```

パラメータ

currency

[in] 国の通貨のコード名

events[]

[out] 指定された通貨の説明を受け取るためのMqlCalendarEvent型配列

戻り値

受け取られた説明の数。エラー情報を取得するためにはGetLastError()関数を呼び出します。可能なエラー:

- 4001 – ERR_INTERNAL_ERROR(一般的なランタイムエラー)
- 4004 – ERR_NOT_ENOUGH_MEMORY(メモリが操作完了に不充分)
- 5401 – ERR_CALENDAR_TIMEOUT(リクエスト制限時間の超過)
- ArrayResize()操作の実行エラー

例:

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 経済指標カレンダーを受け取るための配列を宣言する
      MqlCalendarEvent events[];
//--- EU通貨イベントを取得する
      int count = CalendarEventByCurrency("EUR",events);
      Print("count = ", count);
//--- この例では10イベントで足りる
      if(count>10)
         ArrayResize(events,10);
//--- 操作ログにイベントを表示する
      ArrayPrint(events);
  }
/*
  結果:
             [id] [type] [country_id] [unit] [importance]
             [source_url]                                  [event_code]
                      [name]
```

© 2000–2025, MetaQuotes Ltd.

```
   [0] 999010001      0          999      0            2
"https://www.ecb.europa.eu/home/html/index.en.html" "ecb-non-monetary-policy-
meeting"            "ECB Non-monetary Policy Meeting"
   [1] 999010002      0          999      0            2
"https://www.ecb.europa.eu/home/html/index.en.html" "ecb-monetary-policy-meeting-
accounts"       "ECB Monetary Policy Meeting Accounts"
   [2] 999010003      0          999      0            3
"https://www.ecb.europa.eu/home/html/index.en.html" "ecb-monetary-policy-press-
conference"       "ECB Monetary Policy Press Conference"
   [3] 999010004      0          999      0            3
"https://www.ecb.europa.eu/home/html/index.en.html" "ecb-president-draghi-speech"
            "ECB President Draghi Speech"
   [4] 999010005      0          999      0            2
"https://www.ecb.europa.eu/home/html/index.en.html" "ecb-vice-president-vitor-
constancio-speech" "ECB Vice President Constancio Speech"
   [5] 999010006      1          999      1            3
"https://www.ecb.europa.eu/home/html/index.en.html" "ecb-deposit-rate-decision"
            "ECB Deposit Facility Rate Decision"
   [6] 999010007      1          999      1            3
"https://www.ecb.europa.eu/home/html/index.en.html" "ecb-interest-rate-decision"
            "ECB Interest Rate Decision"
   [7] 999010008      0          999      0            2
"https://www.ecb.europa.eu/home/html/index.en.html" "ecb-economic-bulletin"
            "ECB Economic Bulletin"
   [8] 999010009      1          999      2            2
"https://www.ecb.europa.eu/home/html/index.en.html" "targeted-ltro"
            "ECB Targeted LTRO"
   [9] 999010010      0          999      0            2
"https://www.ecb.europa.eu/home/html/index.en.html" "ecb-executive-board-member-
praet-speech"    "ECB Executive Board Member Praet Speech"
*/
```

参照

CalendarEventById、CalendarEventByCountry

© 2000–2025, MetaQuotes Ltd.

CalendarValueHistoryByEvent

イベントIDによって、指定された期間のすべてのイベントの値の配列を取得します。

```
bool  CalendarValueHistoryByEvent(
   ulong              event_id,          // イベントID
   MqlCalendarValue&  values[],          // 値の説明の配列
   datetime           datetime_from,     // 期間の左の境界
   datetime           datetime_to=0      // 期間の右の境界
   );
```

パラメータ

event_id

[in] イベントID

values[]

[out] イベント値を受け取るためのMqlCalendarValue型の配列。カレンダーイベント処理の例を参照してください。

datetime_from

[in] 指定されたIDによって選択された、期間内のイベントの開始日（*datetime_from* < *datetime_to*）

datetime_to=0

[in] 指定されたIDによって選択された期間内のイベントの終了日。*datetime_to* が設定されていない場合（または0の場合）、指定された*datetime_from*以降のカレンダーデータベースのすべてのイベント値が返されます（将来のイベントを含む）。

戻り値

成功の場合はtrue、それ以外の場合はfalse。エラー情報を取得するためにはGetLastError()関数を呼び出します。可能なエラー:

- 4001 – ERR_INTERNAL_ERROR（一般的なランタイムエラー）
- 4004 – ERR_NOT_ENOUGH_MEMORY（メモリが操作完了に不充分）
- 5401 – ERR_CALENDAR_TIMEOUT（リクエスト制限時間の超過）
- 5400 – ERR_CALENDAR_MORE_DATA（配列サイズはすべての値の説明を受け取るには不十分で、配列に収まるものだけが受信された）
- ArrayResize()操作の実行エラー

注意事項

経済指標カレンダーを使用するためのすべての関数は、取引サーバ時間（TimeTradeServer）を使用します。つまり、MqlCalendarValue構造体の時刻と、CalendarValueHistoryByEvent/CalendarValueHistory関数は、ユーザの現地時間ではなく、取引サーバの時間帯で設定されます。

MqlCalendarValue構造体はactual_value、forecast_value、prev_value、revised_prev_valueフィールドの値を確認/設定するメソッドを備えています。値が指定されていない場合、フィールドにはLONG_MIN (-9223372036854775808)が格納されます。

これらのフィールドに格納されている値は100万倍されることに注意してください。CalendarValueById、CalendarValueHistoryByEvent、CalendarValueHistory、CalendarValueLastByEvent、CalendarValueLastのいずれかの関数を使用してMqlCalendarValueで値を受け取る場合、フィールド値がLONG_MINに等しいかどうかを確認する必要があります。フィールドに値が指定されている場合、値を取得するには、値を100

© 2000–2025, MetaQuotes Ltd.

万で除算する必要があります。値を取得するもう1つの方法は、MqlCalendarValue構造体の関数を使用して値を確認および取得することです。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- EUの国コード(ISO 3166-1 Alpha-2)
   string EU_code="EU";
//--- EUイベントを取得する
   MqlCalendarEvent events[];
   int events_count=CalendarEventByCountry(EU_code,events);
//--- EUイベントを操作ログに表示する
   if(events_count>0)
     {
      PrintFormat("EU events: %d",events_count);
      //--- 分析には10イベントで十分であるため、イベントリストを減らす
      ArrayResize(events,10);
      ArrayPrint(events);
     }
//--- "ECB Interest Rate Decision"イベントがevent_id=999010007であることを確認する
   ulong event_id=events[6].id;         // イベントのIDはカレンダーで変わる可能性があるので、必ず確認する
   string event_name=events[6].name;    // カレンダーイベントの名前
   PrintFormat("Get values for event_name=%s event_id=%d",event_name,event_id);
//--- "ECB Interest Rate Decision"イベントのすべての値を取得する
   MqlCalendarValue values[];
//--- イベントを取得する期間の境界を設定する
   datetime date_from=0;            // 利用可能な履歴の初めからのすべてのイベントをとる
   datetime date_to=D'01.01.2016'; // 2016以降のイベントをとる
   if(CalendarValueHistoryByEvent(event_id,values,date_from,date_to))
     {
      PrintFormat("Received values for %s: %d",
                  event_name,ArraySize(values));
      //--- 分析には10イベントで十分であるため、値リストを減らす
      ArrayResize(values,10);
      ArrayPrint(values);
     }
   else
     {
      PrintFormat("Error!Failed to get values for event_id=%d",event_id);
      PrintFormat("Error code: %d",GetLastError());
     }
  }
//---
/*
```

© 2000-2025, MetaQuotes Ltd.

結果:

```
  EU events: 56
            [id] [type] [sector] [frequency] [time_mode] [country_id] [unit]
[importance] [multiplier] [digits]
[source_url]                                  [event_code]
         [name] [reserv
   [0] 999010001      0        5           0           0          999      0
    2            0        0 "https://www.ecb.europa.eu/home/html/index.en.html"
"ecb-non-monetary-policy-meeting"            "ECB Non-monetary Policy Meeting"

   [1] 999010002      0        5           0           0          999      0
    2            0        0 "https://www.ecb.europa.eu/home/html/index.en.html"
"ecb-monetary-policy-meeting-accounts"       "ECB Monetary Policy Meeting Accounts"

   [2] 999010003      0        5           0           0          999      0
    3            0        0 "https://www.ecb.europa.eu/home/html/index.en.html"
"ecb-monetary-policy-press-conference"       "ECB Monetary Policy Press Conference"

   [3] 999010004      0        5           0           0          999      0
    3            0        0 "https://www.ecb.europa.eu/home/html/index.en.html"
"ecb-president-draghi-speech"                "ECB President Draghi Speech"

   [4] 999010005      0        5           0           0          999      0
    2            0        0 "https://www.ecb.europa.eu/home/html/index.en.html"
"ecb-vice-president-vitor-constancio-speech" "ECB Vice President Constancio Speech"

   [5] 999010006      1        5           0           0          999      1
    3            0        2 "https://www.ecb.europa.eu/home/html/index.en.html"
"ecb-deposit-rate-decision"                  "ECB Deposit Facility Rate Decision"

   [6] 999010007      1        5           0           0          999      1
    3            0        2 "https://www.ecb.europa.eu/home/html/index.en.html"
"ecb-interest-rate-decision"                 "ECB Interest Rate Decision"

   [7] 999010008      0        5           0           0          999      0
    2            0        0 "https://www.ecb.europa.eu/home/html/index.en.html"
"ecb-economic-bulletin"                      "ECB Economic Bulletin"

   [8] 999010009      1        5           0           0          999      2
    2            3        3 "https://www.ecb.europa.eu/home/html/index.en.html"
"targeted-ltro"                              "ECB Targeted LTRO"

   [9] 999010010      0        5           0           0          999      0
    2            0        0 "https://www.ecb.europa.eu/home/html/index.en.html"
"ecb-executive-board-member-praet-speech"    "ECB Executive Board Member Praet
Speech"
  Get values for event_name=ECB Interest Rate Decision event_id=999010007
  Received ECB Interest Rate Decision event values: 102
       [id] [event_id]              [time]             [period] [revision]
[actual_value] [prev_value] [revised_prev_value]     [forecast_value] [impact_type]
[reserved]
   [0] 2776  999010007 2007.03.08 11:45:00 1970.01.01 00:00:00          0
3750000      4250000 -9223372036854775808 -9223372036854775808             0
  0
   [1] 2777  999010007 2007.05.10 11:45:00 1970.01.01 00:00:00          0
3750000      3750000 -9223372036854775808 -9223372036854775808             0
  0
```

© 2000–2025, MetaQuotes Ltd.

```
   [2] 2778  999010007 2007.06.06 11:45:00 1970.01.01 00:00:00          0
4000000       3750000 -9223372036854775808 -9223372036854775808             0
  0
   [3] 2779  999010007 2007.07.05 11:45:00 1970.01.01 00:00:00          0
4000000       4000000 -9223372036854775808 -9223372036854775808             0
  0
   [4] 2780  999010007 2007.08.02 11:45:00 1970.01.01 00:00:00          0
4000000       4000000 -9223372036854775808 -9223372036854775808             0
  0
   [5] 2781  999010007 2007.09.06 11:45:00 1970.01.01 00:00:00          0
4000000       4000000 -9223372036854775808 -9223372036854775808             0
  0
   [6] 2782  999010007 2007.10.04 11:45:00 1970.01.01 00:00:00          0
4000000       4000000 -9223372036854775808 -9223372036854775808             0
  0
   [7] 2783  999010007 2007.11.08 12:45:00 1970.01.01 00:00:00          0
4000000       4000000 -9223372036854775808 -9223372036854775808             0
  0
   [8] 2784  999010007 2007.12.06 12:45:00 1970.01.01 00:00:00          0
4000000       4000000 -9223372036854775808 -9223372036854775808             0
  0
   [9] 2785  999010007 2008.01.10 12:45:00 1970.01.01 00:00:00          0
4000000       4000000 -9223372036854775808 -9223372036854775808             0
  0
*/
```

参照

CalendarCountries、CalendarEventByCountry、CalendarValueHistory、CalendarEventById、CalendarValueById

© 2000–2025, MetaQuotes Ltd.

CalendarValueHistory

指定された期間のすべてのイベントの値の配列を取得します(国や通貨で並び替え可能)。

```
bool  CalendarValueHistory(
   MqlCalendarValue&  values[],              // 値の説明の配列
   datetime           datetime_from,         // 期間の左の境界
   datetime           datetime_to=0          // 期間の右の境界
   const string       country_code=NULL,     // 国コード名(ISO 3166-1 alpha-2)
   const string       currency=NULL          // 国の通貨コード名
   );
```

パラメータ

values[]

[out] イベント値を受け取るためのMqlCalendarValue型の配列。カレンダーイベント処理の例を参照してください。

datetime_from

[in] 指定されたIDによって選択された、期間内のイベントの開始日(*datetime_from* < *datetime_to*)

datetime_to=0

[in] 指定されたIDによって選択された期間内のイベントの終了日。*datetime_to* が設定されていない場合(または0の場合)、指定された*datetime_from*以降のカレンダーデータベースのすべてのイベント値が返されます(将来のイベントを含む)。

country_code=NULL

[in] 国コード名(ISO 3166-1 alpha-2)

currency=NULL

[in] 国の通貨のコード名

戻り値

成功の場合はtrue、それ以外の場合はfalse。エラー情報を取得するためにはGetLastError()関数を呼び出します。可能なエラー:

- 4001 – ERR_INTERNAL_ERROR(一般的なランタイムエラー)
- 4004 – ERR_NOT_ENOUGH_MEMORY(メモリが操作完了に不充分)
- 5401 – ERR_CALENDAR_TIMEOUT(リクエスト制限時間の超過)
- 5400 – ERR_CALENDAR_MORE_DATA(配列サイズはすべての値の説明を受け取るには不十分で、配列に収まるものだけが受信された)
- ArrayResize()操作の実行エラー

注意事項

経済指標カレンダーを使用するためのすべての関数は、取引サーバ時間(TimeTradeServer)を使用します。つまり、MqlCalendarValue構造体の時刻と、CalendarValueHistoryByEvent/CalendarValueHistory関数は、ユーザの現地時間ではなく、取引サーバの時間帯で設定されます。

固定された長さの*events[]*配列が関数に渡されて、結果全体を保存するのに十分なスペースがなかった場合、ERR_CALENDAR_MORE_DATA(5400)エラーが発生します。

© 2000–2025, MetaQuotes Ltd.

datetime_to が設定されていない場合（または0の場合）、指定された*datetime_from*以降のカレンダーデータベースのすべてのイベント値が返されます（将来のイベントを含む）。

*country_code*および*currency*による絞り込みについては、NULLおよび""は道同等で、絞り込みがないことを意味します。

*country_code*については、MqlCalendarCountry構造体の*code*フィールドが使用されるべきです（例："US"、"RU"、"EU"）。

*currency*については、MqlCalendarCountry構造体の*currency*フィールドが使用されるべきです（例：USD"、"RUB"、"EUR"）。

フィルタは結合によって適用されます。つまり、論理 'AND'は、両方の条件（国と通貨）が同時に満たされるイベントの値のみを選択するために使用されます。

MqlCalendarValue構造体はactual_value、forecast_value、prev_value、revised_prev_valueフィールドの値を確認/設定するメソッドを備えています。値が指定されていない場合、フィールドにはLONG_MIN (-9223372036854775808)が格納されます。

これらのフィールドに格納されている値は100万倍されることに注意してください。CalendarValueById、CalendarValueHistoryByEvent、CalendarValueHistory、CalendarValueLastByEvent、CalendarValueLastのいずれかの関数を使用してMqlCalendarValueで値を受け取る場合、フィールド値がLONG_MINに等しいかどうかを確認する必要があります。フィールドに値が指定されている場合、値を取得するには、値を100万で除算する必要があります。値を取得するもう1つの方法は、MqlCalendarValue構造体の関数を使用して値を確認および取得することです。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- EUの国コード（ISO 3166-1 Alpha-2）
   string EU_code="EU";
//--- EUイベント値を取得する
   MqlCalendarValue values[];
//--- イベントを取得する期間の境界を設定する
   datetime date_from=D'01.01.2018';  // 2018のすべてのイベントを取得する
   datetime date_to=0;                // 0はすべての既知イベント（今後起こるものを含む）
//--- 2018年以来のEUイベント履歴をリクエストする
   if(CalendarValueHistory(values,date_from,date_to,EU_code))
     {
      PrintFormat("Received event values for country_code=%s: %d",
                  EU_code,ArraySize(values));
      //--- 操作ログに出力する配列サイズを減らす
      ArrayResize(values,10);
//--- 操作ログにイベント値を表示する
      ArrayPrint(values);
     }
   else
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("Error!Failed to receive events for country_code=%s",EU_code);
      PrintFormat("Error code: %d",GetLastError());
     }
//---
  }
/*
  結果:
  Received event values for country_code=EU: 1384
        [id] [event_id]           [time]               [period] [revision]
[actual_value] [prev_value] [revised_prev_value] [forecast_value] [impact_type]
[reserved]
   [0] 54215  999500001 2018.01.02 09:00:00 2017.12.01 00:00:00          3
60600000     60600000 -9223372036854775808          60500000              1           0
   [1] 54221  999500002 2018.01.04 09:00:00 2017.12.01 00:00:00          3
56600000     56500000 -9223372036854775808          56000000              1           0
   [2] 54222  999500003 2018.01.04 09:00:00 2017.12.01 00:00:00          3
58100000     58000000 -9223372036854775808          58400000              2           0
   [3] 45123  999030005 2018.01.05 10:00:00 2017.11.01 00:00:00          0
600000       400000 -9223372036854775808           100000              1           0
   [4] 45124  999030006 2018.01.05 10:00:00 2017.11.01 00:00:00          0
2800000      2500000 -9223372036854775808          1500000              1           0
   [5] 45125  999030012 2018.01.05 10:00:00 2017.12.01 00:00:00          1
900000       900000 -9223372036854775808          1000000              2           0
   [6] 45126  999030013 2018.01.05 10:00:00 2017.12.01 00:00:00          1
1400000      1500000 -9223372036854775808          1500000              2           0
   [7] 54953  999520001 2018.01.05 20:30:00 2018.01.02 00:00:00          0
127900000     92100000 -9223372036854775808          76400000             0
0
   [8] 22230  999040003 2018.01.08 10:00:00 2017.12.01 00:00:00          0
9100000      8200000              8100000          7600000              1           0
   [9] 22231  999040004 2018.01.08 10:00:00 2017.12.01 00:00:00          0
18400000     16300000             16400000         16800000             1           0

*/
```

参照

CalendarCountries、CalendarEventByCountry、CalendarValueHistoryByEvent、CalendarEventById、CalendarValueById

© 2000–2025, MetaQuotes Ltd.

CalendarValueLastByEvent

指定されたchange_idで、カレンダーデータベースのステータスからイベントIDの配列をIDで取得します。

```
int  CalendarValueLastByEvent(
   ulong                event_id,      // イベントID
   ulong&               change_id,     // イベント値ID
   MqlCalendarValue&    values[]       // 値の説明の配列
   );
```

パラメータ

event_id

[in] イベントID

change_id

[in][out] 変化ID

values[]

[out] イベント値を受け取るためのMqlCalendarValue型の配列。カレンダーイベント処理の例を参照してください。

戻り値

受け取ったイベント値の数。エラー情報を取得するためにはGetLastError()関数を呼び出します。可能なエラー:

- 4001 – ERR_INTERNAL_ERROR(一般的なランタイムエラー)
- 4004 – ERR_NOT_ENOUGH_MEMORY(メモリが操作完了に不充分)
- 5401 – ERR_CALENDAR_TIMEOUT(リクエスト制限時間の超過)
- 5400 – ERR_CALENDAR_MORE_DATA(配列サイズはすべての値の説明を受け取るには不十分で、配列に収まるものだけが受信された)
- ArrayResize()操作の実行エラー

注意事項

経済指標カレンダーを使用するためのすべての関数は、取引サーバ時間(TimeTradeServer)を使用します。つまり、MqlCalendarValue構造体の時刻と、CalendarValueHistoryByEvent/CalendarValueHistory関数は、ユーザの現地時間ではなく、取引サーバの時間帯で設定されます。

固定された長さの*events[]*配列が関数に渡されて、結果全体を保存するのに十分なスペースがなかった場合、ERR_CALENDAR_MORE_DATA(5400)エラーが発生します。

change_id = 0が関数に渡された場合、関数は常に0を返しますが、現在のカレンダーデータベースが*change_id*に返されます。

この関数は、指定されたニュースの配列と、それ以降の関数の呼び出しでニュースの新しい値を受け取るために使用できる新しい*change_id*を返します。したがって、最後の既知の*change_id* を使用してこの関数を呼び出すことで、特定のニュースの値を更新することができます。

MqlCalendarValue構造体はactual_value、forecast_value、prev_value、revised_prev_valueフィールドの値を確認/設定するメソッドを備えています。値が指定されていない場合、フィールドにはLONG_MIN (-9223372036854775808)が格納されます。

© 2000–2025, MetaQuotes Ltd.

これらのフィールドに格納されている値は100万倍されることに注意してください。CalendarValueById、CalendarValueHistoryByEvent、CalendarValueHistory、CalendarValueLastByEvent、CalendarValueLastのいずれかの関数を使用してMqlCalendarValueで値を受け取る場合、フィールド値がLONG_MINに等しいかどうかを確認する必要があります。フィールドに値が指定されている場合、値を取得するには、値を100万で除算する必要があります。値を取得するもう1つの方法は、MqlCalendarValue構造体の関数を使用して値を確認および取得することです。

非農業部門就業者数の公開リスナーであるサンプルEA:

```
#property description "Example of using the CalendarValueLastByEvent function"
#property description " for tracking the release of the Nonfarm Payrolls report."
#property description "To achieve this, get the current change ID"
#property description " of the Calendar database. Then, use this ID to receive"
#property description " only new events via the timer survey"
//+------------------------------------------------------------------+
//| エキスパート初期化関数                                           |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- タイマーを作成する
   EventSetTimer(60);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| エキスパート初期化解除関数                                       |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- タイマーを破壊する
   EventKillTimer();
  }
//+------------------------------------------------------------------+
//| エキスパートティック関数                                         |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| Timer関数                                                        |
//+------------------------------------------------------------------+
void OnTimer()
  {
//--- カレンダーデータベース変更ID
   static ulong calendar_change_id=0;
//--- 1番目の実行属性
   static bool first=true;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- イベントID
   static ulong event_id=0;
//--- イベント名
   static string event_name=NULL;
//--- イベント値の配列
   MqlCalendarValue values[];
//--- 初期化を行う - 現在のcalendar_change_idを取得する
   if(first)
     {
      MqlCalendarEvent events[];
      //--- 米国の国コード(ISO 3166-1 Alpha-2)
      string USA_code="US";
      //--- 米国のイベントを取得する
      int events_count=CalendarEventByCountry(USA_code,events);
      //--- 必要なイベントの 'events' 配列内の位置
      int event_pos=-1;
      //--- 米国のイベントを操作ログに表示する
      if(events_count>0)
        {
         PrintFormat("%s: USA events: %d",__FUNCTION__,events_count);
         for(int i=0;i<events_count;i++)
           {
            string event_name_low=events[i].name;
            //--- イベント名を小文字に変換する
            if(!StringToLower(event_name_low))
              {
               PrintFormat("StringToLower() returned %d error",GetLastError());
               //--- 前もって関数を終了する
               return;
              }
            //---「非農業部門就業者数」イベントを探す
            if(StringFind(event_name_low,"nonfarm payrolls")!=-1)
              {
               //--- イベントが見つかったのでIDを記憶する
               event_id=events[i].id;
               //---「非農業部門就業者数」イベント名を書く
               event_name=events[i].name;
               //--- イベントの 'events[]' 配列内の位置を記憶する
               event_pos=i;
               //--- カレンダーには名前に「非農業部門就業者数」を含むイベントが複数ある
               PrintFormat("Event \"Nonfarm Payrolls\" found: event_id=%d
event_name=%s",event_id,event_name);
               //--- この例をよく理解するためには、'break' 演算子をコメントアウトしてすべてのイベントを表示できる
               break;
              }
           }
         //---「非農業部門就業者数」以降のイベントを削除してリストを減らす
         ArrayRemove(events,event_pos+1);
```

© 2000-2025, MetaQuotes Ltd.

```
         //--- より便利な分析のために「非農業部門就業者数」以前の9つのイベントをそのままにする
         ArrayRemove(events,0,event_pos-9);
         ArrayPrint(events);
        }
      else
        {
         PrintFormat("%s: CalendarEventByCountry(%s) returned 0 events, error code=%
d",
                     USA_code,__FUNCTION__,GetLastError());
         //--- 操作は失敗して終了したため、タイマーの次の呼び出し中に再試行する
         return;
        }

      //--- 指定されたイベントのカレンダーデータベース変更ID を取得する
      if(CalendarValueLastByEvent(event_id,calendar_change_id,values)>0)
        {
         //--- このコードブロックは最初の起動時には実行できないがとにかく追加する
         PrintFormat("%s: Received the Calendar database current ID: change_id=%d",
                     __FUNCTION__,calendar_change_id);
         //--- フラグを設定してタイマーの次のイベントの前に終了する
         first=false;
         return;
        }
      else
        {
         //--- データが受信されない（これは最初の起動では正常である）、エラーがないか確認する
         int error_code=GetLastError();
         if(error_code==0)
           {
            PrintFormat("%s: Received the Calendar database current ID: change_id=%
d",
                        __FUNCTION__,calendar_change_id);
         //--- フラグを設定してタイマーの次のイベントの前に終了する
            first=false;
            //--- calendar_change_idを取得した
            return;
           }
         else
           {
            //--- 本当のエラー
            PrintFormat("%s: Failed to get values for event_id=%
d",__FUNCTION__,event_id);
            PrintFormat("Error code: %d",error_code);
         //--- 操作は失敗して終了したため、タイマーの次の呼び出し中に再試行する
            return;
           }
        }
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- カレンダーの変更IDの最後の既知の値(change_id)がある
   ulong old_change_id=calendar_change_id;
//---「非農業部門就業者数」イベント値を確認する
   if(CalendarValueLastByEvent(event_id,calendar_change_id,values)>0)
     {
      PrintFormat("%s: Received new events for \"%s\": %d",
                  __FUNCTION__,event_name,ArraySize(values));
      //--- 'values' 配列のデータを操作ログで表示する
      ArrayPrint(values);
      //--- 前のカレンダーIDと新しいカレンダーIDの値を操作ログで表示する
      PrintFormat("%s: Previous change_id=%d, new change_id=%d",
                  __FUNCTION__,old_change_id,calendar_change_id);
/*
      非農業部門就業者数データの発表を処理するコードをここで書く
      */
     }
//---
  }
/*
  結果:
   OnTimer: USA events: 202
  Event "Nonfarm Payrolls" found: event_id=840030016  event_name=Nonfarm Payrolls
            [id] [type] [sector] [frequency] [time_mode] [country_id] [unit]
[importance] [multiplier] [digits]          [source_url]
 [event_code]                   [name] [reserved]
   [0] 840030007      1        4           2           0          840      1
    1            0        1 "https://www.bls.gov" "consumer-price-index-yy"
      "CPI y/y"                          0
   [1] 840030008      1        4           2           0          840      1
    1            0        1 "https://www.bls.gov" "consumer-price-index-ex-food-
energy-yy" "Core CPI y/y"                    0
   [2] 840030009      1        4           2           0          840      0
    1            0        3 "https://www.bls.gov" "consumer-price-index-nsa"
      "CPI n.s.a."                       0
   [3] 840030010      1        4           2           0          840      0
    1            0        3 "https://www.bls.gov" "consumer-price-index-ex-food-
energy"    "Core CPI"                      0
   [4] 840030011      1        4           2           0          840      1
    1            0        1 "https://www.bls.gov" "import-price-index-mm"
      "Import Price Index m/m"           0
   [5] 840030012      1        4           2           0          840      1
    1            0        1 "https://www.bls.gov" "import-price-index-yy"
      "Import Price Index y/y"           0
   [6] 840030013      1        4           2           0          840      1
    1            0        1 "https://www.bls.gov" "export-price-index-mm"
      "Export Price Index m/m"           0
   [7] 840030014      1        4           2           0          840      1
    1            0        1 "https://www.bls.gov" "export-price-index-yy"
      "Export Price Index y/y"           0
   [8] 840030015      1        3           2           0          840      1
    2            0        1 "https://www.bls.gov" "unemployment-rate"
      "Unemployment Rate"                0
```

© 2000-2025, MetaQuotes Ltd.

```
   [9] 840030016      1        3           2           0          840       4
    3            1        0 "https://www.bls.gov" "nonfarm-payrolls"
      "Nonfarm Payrolls"                0
   OnTimer: Received the Calendar database current ID: change_id=33986560


*/
```

参照

CalendarValueLast、CalendarValueHistory、CalendarValueHistoryByEvent、CalendarValueById

© 2000–2025, MetaQuotes Ltd.

CalendarValueLast

指定されたchange_idで、カレンダデータベースのステータスからすべてのイベントの値の配列を取得します（国や通貨で並び替え可能）。

```
int  CalendarValueLast(
   ulong&               change_id,              // イベント値ID
   MqlCalendarValue&    values[],               // 値の説明の配列
   const string         country_code=NULL,      // 国コード名(ISO 3166-1 alpha-2)
   const string         currency=NULL           // 国の通貨コード名
   );
```

パラメータ

change_id

[in][out] 変化ID

values[]

[out] イベント値を受け取るためのMqlCalendarValue型の配列。カレンダーイベント処理の例を参照してください。

country_code=NULL

[in] 国コード名(ISO 3166-1 alpha-2)

currency=NULL

[in] 国の通貨のコード名

戻り値

受け取ったイベント値の数。エラー情報を取得するためにはGetLastError()関数を呼び出します。可能なエラー:

- 4001 – ERR_INTERNAL_ERROR(一般的なランタイムエラー)
- 4004 – ERR_NOT_ENOUGH_MEMORY(メモリが操作完了に不充分)
- 5401 – ERR_CALENDAR_TIMEOUT(リクエスト制限時間の超過)
- 5400 – ERR_CALENDAR_MORE_DATA(配列サイズはすべての値の説明を受け取るには不十分で、配列に収まるものだけが受信された)
- ArrayResize()操作の実行エラー

注意事項

経済指標カレンダーを使用するためのすべての関数は、取引サーバ時間(TimeTradeServer)を使用します。つまり、MqlCalendarValue構造体の時刻と、CalendarValueHistoryByEvent/CalendarValueHistory関数は、ユーザの現地時間ではなく、取引サーバの時間帯で設定されます。

固定された長さの*events[]*配列が関数に渡されて、結果全体を保存するのに十分なスペースがなかった場合、ERR_CALENDAR_MORE_DATA(5400)エラーが発生します。

change_id = 0が関数に渡された場合、関数は常に0を返しますが、現在のカレンダーデータベースが*change_id*に返されます。

*country_code*および*currency*による絞り込みについては、NULLおよび""は道同等で、絞り込みがないことを意味します。

© 2000–2025, MetaQuotes Ltd.

*country_code*については、MqlCalendarCountry構造体の*code*フィールドが使用されるべきです(例: "US"、"RU"、"EU")。

*currency*については、MqlCalendarCountry構造体の*currency*フィールドが使用されるべきです(例: USD"、"RUB"、"EUR")。

フィルタは結合によって適用されます。つまり、論理 'AND'は、両方の条件(国と通貨)が同時に満たされるイベントの値のみを選択するために使用されます。

この関数は、指定されたニュースの配列と、それ以降の関数の呼び出しでニュースの新しい値を受け取るために使用できる新しい*change_id*を返します。したがって、最後の既知の*change_id*　を使用してこの関数を呼び出すことで、特定のニュースの値を更新することができます。

MqlCalendarValue構造体はactual_value、forecast_value、prev_value、revised_prev_valueフィールドの値を確認/設定するメソッドを備えています。値が指定されていない場合、フィールドにはLONG_MIN　(-9223372036854775808)が格納されます。

これらのフィールドに格納されている値は100万倍されることに注意してください。CalendarValueById、CalendarValueHistoryByEvent、CalendarValueHistory、CalendarValueLastByEvent、CalendarValueLastのいずれかの関数を使用してMqlCalendarValueで値を受け取る場合、フィールド値がLONG_MINに等しいかどうかを確認する必要があります。フィールドに値が指定されている場合、値を取得するには、値を100万で除算する必要があります。値を取得するもう1つの方法は、MqlCalendarValue構造体の関数を使用して値を確認および取得することです。

経済指標カレンダーイベントを聴取するサンプルEA:

```
#property description "Example of using the CalendarValueLast function"
#property description " to develop the economic calendar events listener."
#property description "To achieve this, get the current change ID"
#property description " of the Calendar database. Then, use this ID to receive"
#property description " only new events via the timer survey"
//+------------------------------------------------------------------+
//|  エキスパート初期化関数                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- タイマーを作成する
   EventSetTimer(60);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//|  エキスパート初期化解除関数                                      |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- タイマーを破壊する
   EventKillTimer();
  }
//+------------------------------------------------------------------+
//|  エキスパートティック関数                                        |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//|  Timer関数                                                       |
//+------------------------------------------------------------------+
void OnTimer()
  {
//--- カレンダーデータベース変更ID
   static ulong calendar_change_id=0;
//--- 1番目の実行属性
   static bool first=true;
//--- イベント値の配列
   MqlCalendarValue values[];
//--- 初期化を行う - 現在のcalendar_change_idを取得する
   if(first)
     {
      //--- カレンダーデータベース変更IDを取得する
      if(CalendarValueLast(calendar_change_id,values)>0)
        {
         //--- このコードブロックは最初の起動時には実行できないがとにかく追加する
         PrintFormat("%s: Received the Calendar database current ID: change_id=%d",
                     __FUNCTION__,calendar_change_id);
         //--- フラグを設定してタイマーの次のイベントの前に終了する
         first=false;
         return;
        }
      else
        {
         //--- データが受信されない(これは最初の起動では正常である)、エラーがないか確認する
         int error_code=GetLastError();
         if(error_code==0)
           {
            PrintFormat("%s: Received the Calendar database current ID: change_id=%
d",
                        __FUNCTION__,calendar_change_id);
         //--- フラグを設定してタイマーの次のイベントの前に終了する
            first=false;
            //--- calendar_change_idを取得した
            return;
           }
         else
           {
            //--- 本当のエラー
            PrintFormat("%s: Failed to get events in CalendarValueLast. Error code:
%d",
                        __FUNCTION__,error_code);
```

© 2000-2025, MetaQuotes Ltd.

```
            //--- 操作は失敗して終了し、タイマーの次の呼び出し中に再初期化される
            return;
           }
        }
     }

//--- カレンダーの変更I Dの最後の既知の値( change_i d) がある
   ulong old_change_id=calendar_change_id;
//--- 新しいカレンダーイベントがあるかどうか確認する
   if(CalendarValueLast(calendar_change_id,values)>0)
     {
      PrintFormat("%s: Received new Calendar events: %d",
                  __FUNCTION__,ArraySize(values));
      //--- ' val ues' 配列のデータを操作ログで表示する
      ArrayPrint(values);
      //--- 前のカレンダーI Dと新しいカレンダーI Dの値を操作ログで表示する
      PrintFormat("%s: Previous change_id=%d, new change_id=%d",
                  __FUNCTION__,old_change_id,calendar_change_id);
      //--- 新しいイベントを操作ログで表示する
      ArrayPrint(values);
      /*
      ここでイベントの発生を処理するためのコードを書く
      */
     }
//---
  }
/*
  リスナー操作の例:
  OnTimer: Received the Calendar database current ID: change_id=33281792
  OnTimer: Received new events for the Calendar: 1
        [id] [event_id]              [time]             [period] [revision]
[actual_value] [prev_value] [revised_prev_value] [forecast_value] [impact_type]
[reserved]
   [0] 91040   76020013 2019.03.20 15:30:00 1970.01.01 00:00:00          0       -
5077000     -1913000 -9223372036854775808         -4077000             2          0
  OnTimer: Previous change_id=33281792, new change_id=33282048
        [id] [event_id]              [time]             [period] [revision]
[actual_value] [prev_value] [revised_prev_value] [forecast_value] [impact_type]
[reserved]
   [0] 91040   76020013 2019.03.20 15:30:00 1970.01.01 00:00:00          0       -
5077000     -1913000 -9223372036854775808         -4077000             2          0
  OnTimer: Received new events for the Calendar: 1
        [id] [event_id]              [time]             [period] [revision]
[actual_value] [prev_value] [revised_prev_value] [forecast_value] [impact_type]
[reserved]
   [0] 91041   76020013 2019.03.27 15:30:00 1970.01.01 00:00:00          0 -
9223372036854775808     -5077000 -9223372036854775808         -7292000             0
         0
  OnTimer: Previous change_id=33282048, new change_id=33282560
```

© 2000-2025, MetaQuotes Ltd.

```
        [id] [event_id]               [time]             [period] [revision]
[actual_value] [prev_value] [revised_prev_value] [forecast_value] [impact_type]
[reserved]
   [0] 91041   76020013 2019.03.27 15:30:00 1970.01.01 00:00:00          0 -
9223372036854775808     -5077000 -9223372036854775808         -7292000             0
         0


*/
```

参照

CalendarValueLast、CalendarValueHistory、CalendarValueHistoryByEvent、CalendarValueById

© 2000–2025, MetaQuotes Ltd.

時系列と指標データへのアクセス

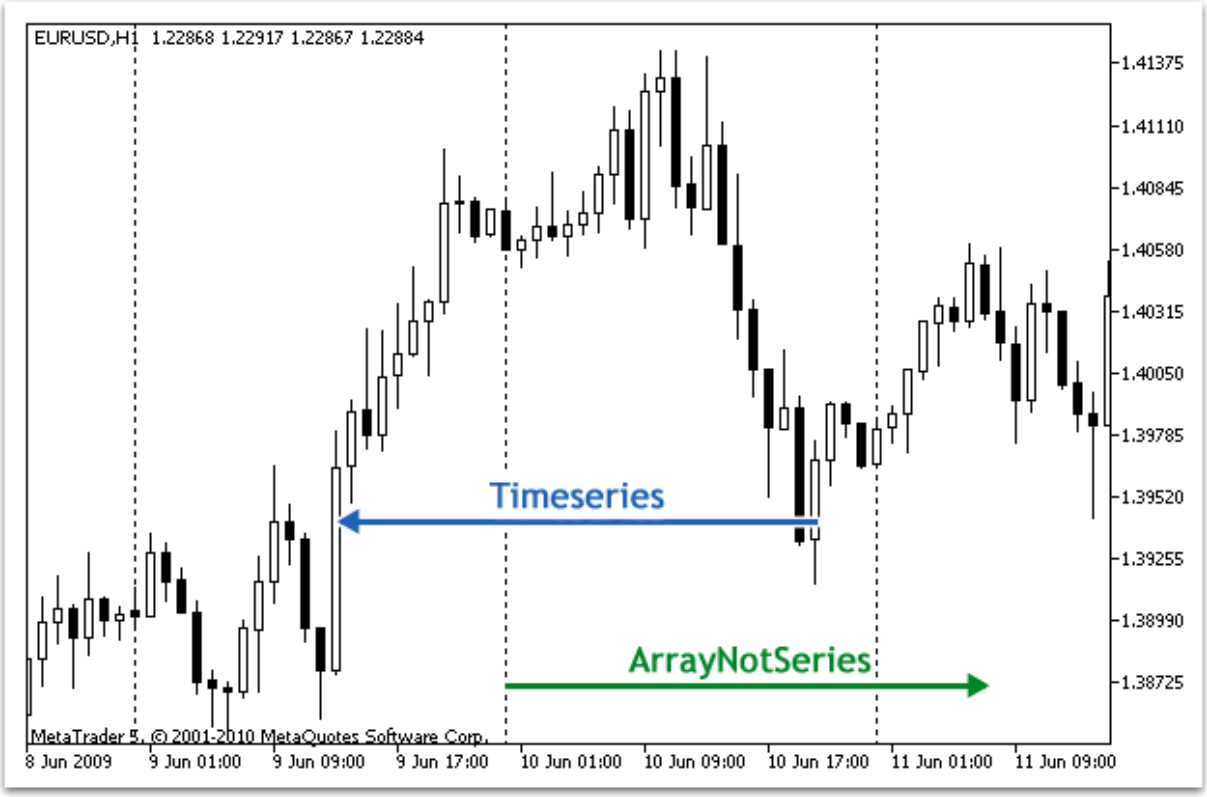
これらの関数は時系列と指標の操作に使用されます。時系列は、索引付けが後ろから前へ(最新のデータから古いデータへ)されている点で通常の配列とは異なります。コピー関数は値を受け取る配列の必要なサイズを割り当てるように設計されているので、時系列の値と指標データのコピーには動的配列の使用のみが推奨されます。

この規則には重要な例外があります。時系列と指標の値が、例えばエキスパートアドバイザーでのOnTick()や指標での OnCalculate() の呼び出しの度に頻繁に複製される場合には動的配列でのメモリ操作には時間がかかりテストと最適化に影響が出るので、静的配列が使用されるべきです。

時系列と指標値にアクセスする関数を使用する場合は、索引付けの方向が考慮されるべきです。これは配列、バッファ及び時系列における索引付けの方向 セクションで説明されています。

指標及び時系列データへのアクセスはリクエストされたデータが準備されているかどうかに関係なく実装されています(いわゆる非同期アクセス)。これは、カスタム指標の計算に非常に重要なので、データが存在しない場合は *Copy...()* 型の関数が即時にエラーを返します。しかし、エキスパートアドバイザーやスクリプトからのアクセス時には、複数のデータを受信するための試みが、指標値を算出するために必要な時系列をダウンロードするために必要な時間を提供する休止をはさんで行われます。

データアクセスの整理セクションは MetaTrader 5 クライアント端末での価格データの受信、格納及びリクエストの詳細を含みます。



配列内の価格データへのアクセスはデータの終わりから行われることは歴史的に認められています。物理的には新しいデータは常に配列の最後に書かれていますが、配列のインデックスは常にゼロに等しいです。時系列配列のインデックス 0 は、この時間軸内の未完成の時間間隔に対応するバー、すなわち、現在足のデータを表します。

時間軸とは、単一の価格バーが形成されている期間です。事前定義された標準時間軸は 21 あります。

© 2000–2025, MetaQuotes Ltd.

関数	アクション
SeriesInfoInteger	履歴データの状態情報を返します。
Bars	指定された銘柄と期間の履歴内のバーの数を返します。
BarsCalculated	指標バッファで計算されたデータの数、またはエラー(データがまだ計算されていません。)の場合 -1 を返します。
IndicatorCreate	MqlParam 型パラメータの配列によって作成された指定されたテクニカル指標へのハンドルを返します。
IndicatorParameters	指定されたハンドルに基づいて、指標の入力パラメータの数、値と型を返します。
IndicatorRelease	指標ハンドルを削除し、誰によっても使用されていない場合は指標の計算ブロックを解放します。
CopyBuffer	指定された指標の指定されたバッファデータを配列に取得します。
CopyRates	指定された銘柄と期間の Rates 構造体の履歴データを配列に取得します。
CopySeries	MqlRates構造体から、指定された銘柄期間と指定された量の同期済み時系列を取得します。
CopyTime	指定された銘柄と期間のオープン時間の履歴データを配列に取得します。
CopyOpen	指定された銘柄と期間のバー始値の履歴データを配列に取得します。
CopyHigh	指定された銘柄と期間のバー高値の履歴データを配列に取得します。
CopyLow	指定された銘柄と期間のバー安値の履歴データを配列に取得します。
CopyClose	配列に指定された銘柄と期間のバー終値の履歴データを取得します。
CopyTickVolume	指定された銘柄と期間のティックボリュームの履歴データを配列に取得します。
CopyRealVolume	指定された銘柄と期間の取引高の履歴データを配列に取得します。
CopySpread	指定された銘柄と期間のスプレッドの履歴データを配列に取得します。
CopyTicks	現在の作業セッションので端末に蓄積されたティックを配列に取得します。
CopyTicksRange	この関数は指定された期間内のティックをMqlTick形式で ticks_arrayで受け取ります。
iBars	該当する銘柄と期間の履歴内のバーの数を返します。
iBarShift	この関数は、指定された時刻に対応するバーのインデックスを返します。
iClose	対応するチャートのバー('shift'パラメータで示される)の終値を返します。
iHigh	対応するチャートのバー('shift'パラメータで示される)の高値を返します。
iHighest	対応するチャート上で見つかった最大値のインデックスを返します(現在のバーとの相対的なシフト)。
iLow	対応するチャートのバー('shift'パラメータで示される)の安値を返します。

© 2000-2025, MetaQuotes Ltd.

関数	アクション
iLowest	該当するチャート上で見つかった最小値のインデックスを返します（現在のバーとの相対的なシフト）。
iOpen	対応するチャートのバー（'shift'パラメータで示される）の始値を返します。
iTime	対応するチャートのバー（'shift'パラメータで示される）の時間を返します。
iTickVolume	対応するチャートのバー（'shift'パラメータで示される）のティックボリュームを返します。
iRealVolume	対応するチャートのバー（'shift'パラメータで示される）の実ボリュームを返します。
iVolume	対応するチャートのバー（'shift'パラメータで示される）のティックボリュームを返します。
iSpread	対応するチャートのバー（'shift'パラメータで示される）のスプレッドを返します。

ArraySetAsSeries()関数を使用して配列要素のアクセスを時系列風に設定することは可能ですが、配列要素は物理的には同じ向きで格納されていて、索引付けの方向だけが変わることを忘れてはいけません。この事実を実証するための例を実行してみましょう。

```
   datetime TimeAsSeries[];
//--- 時系列風に配列アクセスを設定する
   ArraySetAsSeries(TimeAsSeries,true);
   ResetLastError();
   int copied=CopyTime(NULL,0,0,10,TimeAsSeries);
   if(copied<=0)
     {
      Print("The copy operation of the open time values for last 10 bars has
failed");
      return;
     }
   Print("TimeCurrent =",TimeCurrent());
   Print("ArraySize(Time) =",ArraySize(TimeAsSeries));
   int size=ArraySize(TimeAsSeries);
   for(int i=0;i<size;i++)
     {
      Print("TimeAsSeries["+i+"] =",TimeAsSeries[i]);
     }

   datetime ArrayNotSeries[];
   ArraySetAsSeries(ArrayNotSeries,false);
   ResetLastError();
   copied=CopyTime(NULL,0,0,10,ArrayNotSeries);
   if(copied<=0)
     {
      Print("The copy operation of the open time values for last 10 bars has
failed");
      return;
```

© 2000–2025, MetaQuotes Ltd.

```
   }
 size=ArraySize(ArrayNotSeries);
 for(int i=size-1;i>=0;i--)
   {
    Print("ArrayNotSeries["+i+"] =",ArrayNotSeries[i]);
   }
```

結果としてこのような出力が得られます。

```
TimeCurrent = 2009.06.11 14:16:23
ArraySize(Time) = 10
TimeAsSeries[0] = 2009.06.11 14:00:00
TimeAsSeries[1] = 2009.06.11 13:00:00
TimeAsSeries[2] = 2009.06.11 12:00:00
TimeAsSeries[3] = 2009.06.11 11:00:00
TimeAsSeries[4] = 2009.06.11 10:00:00
TimeAsSeries[5] = 2009.06.11 09:00:00
TimeAsSeries[6] = 2009.06.11 08:00:00
TimeAsSeries[7] = 2009.06.11 07:00:00
TimeAsSeries[8] = 2009.06.11 06:00:00
TimeAsSeries[9] = 2009.06.11 05:00:00

ArrayNotSeries[9] = 2009.06.11 14:00:00
ArrayNotSeries[8] = 2009.06.11 13:00:00
ArrayNotSeries[7] = 2009.06.11 12:00:00
ArrayNotSeries[6] = 2009.06.11 11:00:00
ArrayNotSeries[5] = 2009.06.11 10:00:00
ArrayNotSeries[4] = 2009.06.11 09:00:00
ArrayNotSeries[3] = 2009.06.11 08:00:00
ArrayNotSeries[2] = 2009.06.11 07:00:00
ArrayNotSeries[1] = 2009.06.11 06:00:00
ArrayNotSeries[0] = 2009.06.11 05:00:00
```

出力からわかるように、TimeAsSeries 配列のインデックスが増加するにつれてインデックスの time 値が減少します。つまり、現在から過去に存在から移動するわけです。ArrayNotSeries 通常配列では結果が異なります。インデックスが大きくなるにつれて、過去から現在までの移動が行われます。

参照

ArrayIsDynamic、ArrayGetAsSeries、ArraySetAsSeries、ArrayIsSeries

© 2000–2025, MetaQuotes Ltd.

配列、バッファ及び時系列における索引付けの方向

全ての配列及び指標バッファの索引付けはデフォルトでは左から右です。最初の要素のインデックスは常にゼロに等しいです。従って、インデックス 0 の配列または指標バッファの最初の要素は一番左にあり、最後の要素は一番右にあるのがデフォルトです。

指標バッファは double 型の動的配列で、サイズは、常に指標が計算されているバーの数に等しいようにクライアント端末で管理されています。通常の double 型の動的配列は SetIndexBuffer() 関数を使用して指標バッファとして割り当てられます。指標バッファのサイズの設定は端末の実行システムによって行われるので ArrayResize() 関数を使用する必要はありません。

時系列は逆方向の索引付けを持つ配列です。時系列の最初の要素は一番右、最後の要素は一番左にあります。履歴価格データの格納に使用される時系列には時間情報が含まれているので、最も古いデータが一番左にあって最新のデータが一番右にあると言えます。

従って、時系列のインデックス0 の要素は、シンボルの最新の相場に関する情報を含んでいます。時系列が 1 日の時間軸のデータを含んでいる場合は、まだ未完了の今日のデータがゼロの位置に配置されていて、インデックス1の位置には昨日のデータが含まれています。

索引付けの方向の変更

ArraySetAsSeries() 関数は、動的配列の要素にアクセスする方法を変更することが出来ます。コンピュータメモリに格納されるデータの物理的な順序は変更されません。この関数は、単に配列要素のアドレス指定する方法を変更するので、ArrayCopy() を使用して 1 つの配列をもう 1 つに複製する際には、受け取り側の配列の内容は元の配列に索引付け方向に依存しません。

索引付けの方向は静的配列では変更することは出来ません。配列を関数にパラメータとして渡す場合も、関数内での索引付けの方向を変更する試みは、何の効果ももたらしません。

指標バッファは、通常の配列の場合と同様に、索引付けの方向も反対に(時系列のように)設定出来ます。指標バッファでのゼロ位置への参照は対応する指標バッファの最後の値への参照を意味し最新のバーの指標値に対応します。それでも、指標バーの物理的な位置は変更されません。

指標の価格データの受け取り

カスタム指標は、指示バッファ内の値の計算に必要な価格データが渡されるOnCalculate() 関数を含まなければなりません。渡された配列内のインデックス方向は ArrayGetAsSeries() 関数を使用して見つけることが出来ます。

関数に渡された配列は価格データで時系列であるのでArrayIsSeries() でのチェックは true を返します。しかし、いずれの場合、インデックス方向のチェックは ArrayGetAsSeries() 関数で行われるべきです。

初期値に依存しないために ArraySetAsSeries() を作業する配列に無条件に呼び出し、必要な方向を設定するべきです。

価格データと指標値の受け取り

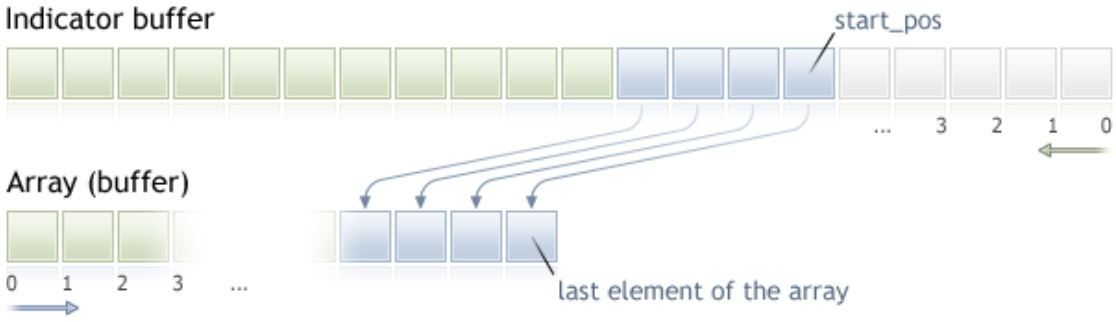
エキスパートアドバイザーの全ての配列、指標及びスクリプトの索引付けはデフォルトでは左から右です。MQL5 プログラムでは、必要に応じて、シンボル及び時間軸で計算された指標値、またその時系列値をリクエストすることが出来ます。

この目的には Copy...() 関数群を使用します。

© 2000-2025, MetaQuotes Ltd.

- CopyBuffer – 指標バッファの値をdouble 型の配列に複製する
- CopyRates – 価格履歴を MqlRates 構造体の配列に複製する
- CopyTime – Time 値をdatetime 型の配列に複製する
- CopyOpen – Open 値をdouble 型の配列に複製する
- CopyHigh – High 値をdouble 型の配列に複製する
- CopyLow – Low 値をdouble 型の配列に複製する
- CopyClose – Close 値をdouble 型の配列に複製する
- CopyTickVolume – ティックボリュームを long 型の配列に複製する
- CopyRealVolume – 資本金ボリュームを long 型の配列に複製する
- CopySpread – スプレッド履歴を int 型の配列に複製する

これらの関数は全て同様の方法で動作します。CopyBuffer() の例でデータ取得メカニズムを考えてみましょう。リクエストされたデータの索引付け方向が時系列と同じであることを暗示し、現在まだ完了していないバーはインデックス0（ゼロ）でデータを格納しています。これらのデータへのアクセスを取得するにはデータの必要量を受け取り側の配列バッファに複製する必要があります。



複製時には、データが受け取り側の配列に複製され始める元の配列内の開始位置を指定する必要があります。成功した場合には、指定された数の要素が元の配列（ここでは指標バッファ(Indicator buffer)）から受け取り側の配列に複製されます。受け取り側の配列に設定された値の索引付けにかかわらず、コピーは常に上図に示されたように実行されます。

価格データが繰り返しの数が多いループ内で処理されることが期待されている場合 IsStopped() 関数を使用して強制的なプログラム終了を確認することをお勧めします。

```
int copied=CopyBuffer(ma_handle,// 指標値
                      0,        // 指標バッファのインデックス
                      0,        // コピー開始位置
                      number,   // 複製する値の数
                      Buffer    // 値を受け取る配列
                      );
if(copied<0) return;
int k=0;
while(k<copied && !IsStopped())
  {
   //--- k インデックスの値を取得する
   double value=Buffer[k];
   // ...
```

© 2000–2025, MetaQuotes Ltd.

```
   // 値を操作する
    k++;
  }
```

例：

```
input int per=10; // 指数の期間
int ma_handle;    // 指標ハンドル
//+------------------------------------------------------------------+
//| エキスパート初期化に使用される関数                                |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   ma_handle=iMA(_Symbol,0,per,0,MODE_EMA,PRICE_CLOSE);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| エキスパートティック関数                                          |
//+------------------------------------------------------------------+
void OnTick()
  {
//---
   double ema[10];
   int copied=CopyBuffer(ma_handle,// 指標ハンドル
                         0,        // 指標バッファのインデックス
                         0,        // コピー開始位置
                         10,       // 複製する値の数
                         ema       // 値を受け取る配列
                         );
   if(copied<0) return;
// .... 更なるコード
  }
```

参照

データアクセスの整理

© 2000-2025, MetaQuotes Ltd.

データアクセスの整理

このセクションでは、価格データ(時系列)の取得、格納、及びリクエストに関する質問が考慮されます。

取引サーバからのデータ受信

価格データは、MetaTrader 5 端末で使用可能になる前に受信されて処理されなければなりません。データを受信するには MetaTrader 5 取引サーバへの接続が必要です。データは、端末のリクエストに応じてサーバから分足のパックブロックの形式で受信されます。

サーバのデータ参照メカニズムは、リクエストがチャートをナビゲートしているユーザによって送られたか MQL5 言語のプログラムで送られたかなどの方法には依存しません。

中間データの格納

サーバーから受信されたデータは自動的に解凍されてHCC 中間形式で格納されます。各シンボルのデータは *terminal_directory*\bases*server_name*\history*symbol_name* の別々のフォルダに書き入れられます。例えば MetaQuotes-Demo サーバから受信された EURUSD データは *terminal_directory*¥bases¥MetaQuotes-Demo¥history¥EURUSD¥ に格納されます。

データは .hcc 拡張子を持つファイルに書き込まれます。各ファイルは一年分の分足を格納します。例えば EURUSD フォルダの 2009.hcc ファイルは 2009 年の EURUSD 分足を含みます。これらのファイルは全ての時間軸の価格データを準備するために使用され、直接アクセスは意図されていません。

中間データからの必要な時間軸のデータの取得

中間 HCC ファイルは、リクエストされた時間軸の価格データをHC 形式で構築するためのデータ元として使用されます。HC 形式のデータは最大限に迅速なアクセスのために用意された時系列です。これらは MQL5 プログラムチャートのリクエストに応じて作成されます。データ量は「チャートでのバーの最大数」パラメータの値を超えてはいけません。データは更に使用されるために hc 拡張子を持つファイルに格納されます。

システムリソース節約のために、時間軸のデータは必要時のみに RAM に格納されて保存されます。データは長時間呼び出されない場合には RAM から解放されてファイルに保存されます。データは、他の時間軸かの準備ができたデータがあるかどうかにかかわらず各時間軸で用意されます。データ形成及びアクセスのルールは全ての時間軸で同じです。HCC に分足 (M1)データが格納されていても、同じボリュームの M1 軸のデータの HC としての可用性があるわけではありません。

サーバからの新規データの受信によって、全ての時間軸で使用された価格データの HC 形式での自動更新が呼び出されます。また、それらのデータを計算の入力データとして暗黙的に使用する指標全ての再計算につながります。

「チャートのバーの最大数」パラメータ

「チャートのバーの最大数」パラメータはチャート、指標、及び MQL5 プログラムで使用可能な HC 形式のバーの数を制限します。これは全ての利用可能な時間軸のために有効であり、コンピュータリソースを節約するために提供されています。

このパラメータに大きな値を設定する場合は、小さな時間軸で多くの履歴の価格データが利用可能であると、時系列と指標バッファ格納に使用されるメモリは数百メガバイトになってクライアント端末プログラムの RAMの制限(MS Windows の32 ビットアプリケーション).に達することがあるので注意が必要です。

© 2000-2025, MetaQuotes Ltd.

「チャートのバーの最大数」の変更はクライアント端末の再起動の際に有効になります。このパラメータの変更は、サーバでの追加データの自動や時系列の追加のバーの形成にはつながりません。チャートがデータのない領域にスクロールするかデータが MQL5 プログラムによってリクエストされた場合は、追加の価格データがサーバからリクエストされ、時系列は新しい制限をもって更新されます。

サーバから要求されたデータの量は「チャートのバーの最大数」パラメータを考慮した上のこの時間軸のバーの必要な数に相当します。このパラメータで設定された制限は厳密ではなく、時間軸で利用可能なバーの数が現在のパラメータ値を少し上回る場合があります。

データの可用性

HCC 形式でのデータの可用性または HC 形式での使用が準備中のデータの可用性は、これらのデータの絶対的な可用性を意味するものではありません。

MQL5 プログラムから価格データや指標値にアクセスする場合、それらの特定の瞬間、また特定の開始時間での可用性が保証されていないことを忘れてはなりません。これは、リソース節約の目的で、MQL5 プログラムに必要なデータの完全なコピーは MetaTrader 5 に格納されず、端末データベースへの直接アクセスのみが与えられているという事実と関係しています。

全ての時間軸の価格履歴が HCC 形式の共通のデータから構築され、サーバからのデータの更新は、全ての時間軸データの更新及び指標の再計算ににつながります。そのために、データは少し前に利用可能であったとしても利用不可能になることがあります。

端末データとサーバデータの同期化

MQL5 プログラムは任意のシンボルと時間軸のデータを呼び出すことが出来るので、必要な時系列のデータが端末にまだ形成されていないまたは必要な価格データが取引サーバと同期されていない可能性があります。この場合、待ち時間を予測するのは難しいです。

「無演算」ループを使用したアルゴリズムは最善の解決策ではありません。スクリプトは、イベント処理を持たないため代替アルゴリズムの選択肢がないので、唯一の例外です。カスタム指標では、このようなアルゴリズムや他の「無演算」ループは、全ての指標の計算と他のシンボルの価格データの処理の終了につながるため、推奨出来ません。

エキスパートアドバイザーと指標ではハンドルのイベントモデルが使用されるべきです。OnTick() または OnCalculate() イベントの取り扱い中に必要な時系列のデータの受信に失敗した場合、ハンドラの次の呼び出しの間にアクセスの可用性に依存してイベントハンドラを終了する必要があります。

履歴追加スクリプトの例

選択されたシンボルの履歴を取引サーバから受信するためのリクエストを実行するスクリプトの例を考えてみましょう。このスクリプトは、選択したシンボルのチャートで実行するためのものです。前述したように、価格データは取引サーバからパック化された 1分間のデータとして受信され、事前定義された時系列はそれから構築されるので、時間軸は重要ではありません。

データ受信に関する全てのアクションは別の関数 CheckLoadHistory(symbol, timeframe, start_date) として書きます。

```
int CheckLoadHistory(string symbol,ENUM_TIMEFRAMES period,datetime start_date)
  {
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckLoadHistory() 関数は、任意のプログラム(エキスパートアドバイザー、スクリプトまたは指標)から呼び出すことが出来る普遍的な機能として設計されており、銘柄名、期間、必要な価格履歴の始まりを示す開始日の3つの入力パラメータを必要とします。

不足している履歴をリクエストする前に、関数コードに必要なチェックを挿入します。まず第一に、銘柄名と期間値が正しいことの確認が必要です。

```
   if(symbol==NULL || symbol=="") symbol=Symbol();
   if(period==PERIOD_CURRENT)     period=Period();
```

取引サーバにリクエストを送信する時にシンボルの履歴が利用出来る用、シンボルが「気配値表示」ウィンドウで利用可能であることを確認します。シンボルが「気配値表示」にない場合 SymbolSelect() 関数で追加します。

```
   if(!SymbolInfoInteger(symbol,SYMBOL_SELECT))
     {f
      if(GetLastError()==ERR_MARKET_UNKNOWN_SYMBOL) return(-1);
      SymbolSelect(symbol,true);
     }
```

示された銘柄/期間のペアで利用可能な履歴の開始日を受信する必要があります。CheckLoadHistory() に渡された startdate 入力パラメータの値が利用可能な履歴の範囲内にあるなら、その後取引サーバリクエストは必要ありません。現時点での銘柄/期間のペアの最初の日を取得するには、SeriesInfoInteger() 関数と SERIES_FIRSTDATE 修飾子が使用されます。

```
   SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date);
   if(first_date>0 && first_date<=start_date) return(1);
```

次の重要なチェックは、関数が呼び出されるプログラムの種類のチェックです。指標と同じ期間を持つ時系列の更新リクエストを送信することは望ましくありません。指標と同じシンボル期間のデータのリクエストが望ましくないのは、履歴データの更新が指標が作動するのと同じスレッドで実行されるという事実があるからです。従ってデッドロックが発生する可能性が高いです。これをチェックするには MQL5InfoInteger() 関数を MQL5_PROGRAM_TYPE 修飾子と使用します。

```
   if(MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR && Period()==period &&
Symbol()==symbol)
      return(-4);
```

全てのチェックが成功した場合、もう1 回取引サーバにリクエストを送らないですむか見てみます。初めに HCC 形式でのデータが使用可能な初めの日を見つけます。この値を SeriesInfoInteger() 関数と SERIES_TERMINAL_FIRSTDATE 修飾子でリクエストし start_date パラメータ値と比べます。

```
   if(SeriesInfoInteger(symbol,PERIOD_M1,SERIES_TERMINAL_FIRSTDATE,first_date))
     {
      //--- 時系列を構築するために読み込まれたデータ
      if(first_date>0)
        {
         //--- 時系列の構築を強制する
         CopyTime(symbol,period,first_date+PeriodSeconds(period),1,times);
         //--- 日にちをチェックする
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            if(first_date>0 && first_date<=start_date) return(2);
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      }
```

全てのチェック後に実行スレッドが CheckLoadHistory() 関数の本体に残っている場合、取引サーバから欠落している価格データをリクエストする必要があります。初めに TerminalInfoInteger() 関数を使用して「チャートでのバーの最大数」値を返します。

```
   int max_bars=TerminalInfoInteger(TERMINAL_MAXBARS);
```

これは、余分なデータのリクエストを防止するのに必要です。次に、取引サーバでの(期間に関係ない)シンボル履歴の初日を SeriesInfoInteger() 関数とSERIES_SERVER_FIRSTDATE 修飾子を使用して見つけます。

```
    datetime first_server_date=0;
    while(!SeriesInfoInteger
(symbol,PERIOD_M1,SERIES_SERVER_FIRSTDATE,first_server_date) && !IsStopped())
       Sleep(5);
```

リクエストは非同期動作なので、この関数は first_server_date 変数が値を受け取るまでの遅延の5ミリ秒の小さい枠で呼び出されます。サイクルの実行がユーザによって終了された場合 IsStopped() は *true* を返します。取引サーバからの価格データリクエストを開始する日付けの正しい値を示してみましょう。

```
    if(first_server_date>start_date) start_date=first_server_date;
    if(first_date>0 && first_date<first_server_date)
       Print("Warning: first server date ",first_server_date," for ",
symbol," does not match to first series date ",first_date);
```

サーバの開始日である *first_server_date* が HCC 形式のシンボルの開始日である *first_date* より早い場合、対応するエントリーが操作ログに出力されます。

欠落した価格データを取引サーバからリクエストする準備ができました。ループ形式でリクエストを作成し本体に記入を開始します。

```
    while(!IsStopped())
      {
       //1. HHC として再構築された時系列と中間の履歴との間の同期を待ちます。
       //2. この時間軸での現在のバーの数を受け取ります。
       //     バーの数が Max_bars_in_chart より多い場合、終了します。
       //3. 再構築された時系列で first_date 開始日を取得して start_date と比較します。
       //     first_date が start_date, より早い場合、作業が終わったので終了します。
       //4. サーバから新しい期間の履歴をリクエストします。(「 bars」で番号図けられた最後の使用可能のバーから100足
      }
```

初めの 3 点は、すでに知られている手段によって実施されます。

```
    while(!IsStopped())
      {
       //--- 1. 時系列の最構築が完成するまで待つ
       while(!SeriesInfoInteger(symbol,period,SERIES_SYNCHRONIZED) && !IsStopped())
          Sleep(5);
       //--- 2. 存在するバーの数をリクエストする
       int bars=Bars(symbol,period);
       if(bars>0)
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          //--- チャートに描画出来る以上のバーがあるので終了する
          if(bars>=max_bars) return(-2);
          //--- 3. 時間軸での現在の開始日を返す
          if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
             // 開始日がリクエストされたものより早いので、作業完了
             if(first_date>0 && first_date<=start_date) return(0);
         }
       //4. サーバから新しい期間の履歴をリクエストします。(「bars」で番号図けられた最後の使用可能のバーから100足)。
      }
```

最後の4番目のポイントが残っています。履歴のリクエストです。サーバに直接参照することは出来ません。が、HCC形式の履歴が充分でない場合 コピー関数 は自動的にサーバリクエストの送信を開始します。*first_date* 変数で表される初日の時刻がリクエストの実行度を評価するためのシンプルで自然な基準なので CopyTime() 関数の使用が一番容易です。

時系列からデータを複製する関数を呼び出す際には、*start* パラメータ(価格データコピーの開始点からのバーの数)は端末で利用可能な履歴の範囲からとられなければなりません。例えば、100 足しかない時に、インデックス500から始まって300足を複製する試みは無意味です。このような要求は誤ったものとして理解されて処理されず、追加の履歴は取引サーバから読み込まれません。

これが *bars* インデックスから始めて 100 足を複製する理由です。これは、欠落している履歴の取引サーバからのスムーズな読み込みを提供します。サーバは余分の履歴を送るので、実際にはリクエストされた 100 より少し大きい数の足が読み込まれます。

```
   int copied=CopyTime(symbol,period,bars,100,times);
```

コピー操作の後では、複製された要素の数が分析されるべきです。試みが失敗した場合 *copied* の値は null に等しくfail_cnt カウンタの値は 1 で増えます。100 の試行失敗の後、関数の動作が停止されます。

```
int fail_cnt=0;
...
   int copied=CopyTime(symbol,period,bars,100,times);
   if(copied>0)
     {
      //--- データをチェックする
      if(times[0]<=start_date)   return(0);  // 複製された値のほうが小さい。準備完了
      if(bars+copied>=max_bars) return(-2); // バーの数がチャートに描画出来るより多い。準備完了
      fail_cnt=0;
     }
   else
     {
      //--- 100以上の失敗の試みが連続していない
      fail_cnt++;
      if(fail_cnt>=100) return(-5);
      Sleep(10);
     }
```

© 2000-2025, MetaQuotes Ltd.

従って、各瞬間における現在の状況の正しい処理が関数内で実装されているだけではなく、CheckLoadHistory() 関数の呼び出し後に処理出来る追加情報を取得するための終了コードも戻されます。例えばこのようにです。

```
   int res=CheckLoadHistory(InpLoadedSymbol,InpLoadedPeriod,InpStartDate);
   switch(res)
     {
      case -1 : Print("Unknown symbol ",InpLoadedSymbol);                    break;
      case -2 : Print("More requested bars than can be drawn in the chart"); break;
      case -3 : Print("Execution stopped by user");                          break;
      case -4 : Print("Indicator mustn't load its own data");               break;
      case -5 : Print("Loading failed");                                    break;
      case  0 : Print("All data loaded");                                   break;
      case  1 : Print("Already available data in timeseries are enough");   break;
      case  2 : Print("Timeseries is built from available terminal data");  break;
      default : Print("Execution result undefined");
     }
```

この関数の完全なコードは、リクエスト結果の取り扱いを持つ任意のデータへのアクセスの正しい構成を示すスクリプトの例に記載されています。

コード：

```
//+------------------------------------------------------------------+
//|                                                TestLoadHistory.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.02"
#property script_show_inputs
//--- 入力パラメータ
input string          InpLoadedSymbol="NZDUSD";    // 読み込まれるシンボル
input ENUM_TIMEFRAMES InpLoadedPeriod=PERIOD_H1;   // Period to be loaded
input datetime        InpStartDate=D'2006.01.01'; // 開始日
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
   Print("Start
load",InpLoadedSymbol+","+GetPeriodName(InpLoadedPeriod),"from",InpStartDate);
//---
   int res=CheckLoadHistory(InpLoadedSymbol,InpLoadedPeriod,InpStartDate);
   switch(res)
     {
      case -1 : Print("Unknown symbol ",InpLoadedSymbol);                  break;
      case -2 : Print("Requested bars more than max bars in chart"); break;
      case -3 : Print("Program was stopped");                          break;
```

© 2000-2025, MetaQuotes Ltd.

```
      case -4 : Print("Indicator shouldn't load its own data");      break;
      case -5 : Print("Load failed");                                break;
      case  0 : Print("Loaded OK");                                  break;
      case  1 : Print("Loaded previously");                         break;
      case  2 : Print("Loaded previously and built");               break;
      default : Print("Unknown result");
     }
//---
   datetime first_date;
   SeriesInfoInteger(InpLoadedSymbol,InpLoadedPeriod,SERIES_FIRSTDATE,first_date);
   int bars=Bars(InpLoadedSymbol,InpLoadedPeriod);
   Print("First date ",first_date," - ",bars," bars");
//---
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
int CheckLoadHistory(string symbol,ENUM_TIMEFRAMES period,datetime start_date)
  {
   datetime first_date=0;
   datetime times[100];
//--- 銘柄と期間をチェックする
   if(symbol==NULL || symbol=="") symbol=Symbol();
   if(period==PERIOD_CURRENT)     period=Period();
//--- シンボルが「気配値表示」で選ばれているかをチェックする
   if(!SymbolInfoInteger(symbol,SYMBOL_SELECT))
     {
      if(GetLastError()==ERR_MARKET_UNKNOWN_SYMBOL) return(-1);
      SymbolSelect(symbol,true);
     }
//--- データがあるかをチェックする
   SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date);
   if(first_date>0 && first_date<=start_date) return(1);
//--- 指標の場合自身のデータの読み込みはリクエストしない
   if(MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR && Period()==period &&
Symbol()==symbol)
      return(-4);
//--- 2 回目の試み
   if(SeriesInfoInteger(symbol,PERIOD_M1,SERIES_TERMINAL_FIRSTDATE,first_date))
     {
      //--- 時系列を構築するために読み込まれたデータ
      if(first_date>0)
        {
         //--- 時系列の構築を強制する
         CopyTime(symbol,period,first_date+PeriodSeconds(period),1,times);
         //--- 日にちをチェックする
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            if(first_date>0 && first_date<=start_date) return(2);
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- 端末オプションでのチャートのバーの最大数
   int max_bars=TerminalInfoInteger(TERMINAL_MAXBARS);
//--- シンボル履歴情報を読み込む
   datetime first_server_date=0;
   while(!SeriesInfoInteger
(symbol,PERIOD_M1,SERIES_SERVER_FIRSTDATE,first_server_date) && !IsStopped())
      Sleep(5);
//--- 読み込み開始日を治す
   if(first_server_date>start_date) start_date=first_server_date;
   if(first_date>0 && first_date<first_server_date)
      Print("Warning: first server date ",first_server_date," for ",symbol,
            " does not match to first series date ",first_date);
//--- ステップごとにデータを読み込む
   int fail_cnt=0;
   while(!IsStopped())
     {
      //--- 時系列の構築を待つ
      while(!SeriesInfoInteger(symbol,period,SERIES_SYNCHRONIZED) && !IsStopped())
         Sleep(5);
      //--- 構築されたバーをリクエストする
      int bars=Bars(symbol,period);
      if(bars>0)
        {
         if(bars>=max_bars) return(-2);
         //--- 初日をリクエストする
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            if(first_date>0 && first_date<=start_date) return(0);
        }
      //--- 次の部分のコピーはデータ読み込みを強制する
      int copied=CopyTime(symbol,period,bars,100,times);
      if(copied>0)
        {
         //--- データをチェックする
         if(times[0]<=start_date)  return(0);
         if(bars+copied>=max_bars) return(-2);
         fail_cnt=0;
        }
      else
        {
         //--- 100 以下の失敗の試み
         fail_cnt++;
         if(fail_cnt>=100) return(-5);
         Sleep(10);
        }
     }
//--- 停止した
   return(-3);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  期間の文字列値を返す                                            |
//+------------------------------------------------------------------+
string GetPeriodName(ENUM_TIMEFRAMES period)
  {
   if(period==PERIOD_CURRENT) period=Period();
//---
   switch(period)
     {
      case PERIOD_M1:  return("M1");
      case PERIOD_M2:  return("M2");
      case PERIOD_M3:  return("M3");
      case PERIOD_M4:  return("M4");
      case PERIOD_M5:  return("M5");
      case PERIOD_M6:  return("M6");
      case PERIOD_M10: return("M10");
      case PERIOD_M12: return("M12");
      case PERIOD_M15: return("M15");
      case PERIOD_M20: return("M20");
      case PERIOD_M30: return("M30");
      case PERIOD_H1:  return("H1");
      case PERIOD_H2:  return("H2");
      case PERIOD_H3:  return("H3");
      case PERIOD_H4:  return("H4");
      case PERIOD_H6:  return("H6");
      case PERIOD_H8:  return("H8");
      case PERIOD_H12: return("H12");
      case PERIOD_D1:  return("Daily");
      case PERIOD_W1:  return("Weekly");
      case PERIOD_MN1: return("Monthly");
     }
//---
   return("unknown period");
  }
```

© 2000–2025, MetaQuotes Ltd.

SeriesInfoInteger

履歴データの状態情報を返します。この関数の呼び出し方は 2 つあります。

直接プロパティ値を返します。

```
long  SeriesInfoInteger(
   string                      symbol_name,      // 銘柄名
   ENUM_TIMEFRAMES             timeframe,        // 期間
   ENUM_SERIES_INFO_INTEGER    prop_id,          // プロパティ識別子
   );
```

関数実行の成功に応じて true または false を返します。

```
bool  SeriesInfoInteger(
   string                      symbol_name,      // 銘柄名
   ENUM_TIMEFRAMES             timeframe,        // 期間
   ENUM_SERIES_INFO_INTEGER    prop_id,          // プロパティ識別子
   long&                       long_var          // 情報を受け取る変数
   );
```

パラメータ

symbol_name
[in] 銘柄名

timeframe
[in] 期間

prop_id
[in] リクエストされたプロパティの識別子。ENUM_SERIES_INFO_INTEGER 列挙の値

long_var
[out] リクエストされたプロパティの値が配置される変数

戻り値

1 番目の呼び方では long 型の値。

2 番目の呼び方では、指定されたプロパティが利用可能で値が long_var 変数に置かれてい場合には true それ以外の場合は false。エラーの詳細については GetLastError() を呼びます。

例:

```
void OnStart()
  {
//---
   Print("Total number of bars for the symbol-period at this moment = ",
         SeriesInfoInteger(Symbol(),Period(),SERIES_BARS_COUNT));

   Print("The first date for the symbol-period at this moment = ",
         (datetime)SeriesInfoInteger(Symbol(),Period(),SERIES_FIRSTDATE));
```

© 2000-2025, MetaQuotes Ltd.

```
   Print("The first date in the history for the symbol-period on the server = ",
         (datetime)SeriesInfoInteger(Symbol(),Period(),SERIES_SERVER_FIRSTDATE));

   Print("Symbol data are synchronized = ",
         (bool)SeriesInfoInteger(Symbol(),Period(),SERIES_SYNCHRONIZED));
  }
```

© 2000–2025, MetaQuotes Ltd.

Bars

指定された銘柄と期間の履歴内のバーの数を返します。この関数の呼び出し方は 2 つあります。

履歴内のバーを全てリクエスト

```
int  Bars(
   string           symbol_name,      // 銘柄名
   ENUM_TIMEFRAMES  timeframe         // 期間
   );
```

選択された時間間隔内の履歴内のバーをリクエスト

```
int  Bars(
   string           symbol_name,      // 銘柄名
   ENUM_TIMEFRAMES  timeframe,        // 期間
   datetime         start_time,       // 開始日付と時刻
   datetime         stop_time         // 終了日付と時刻
   );
```

パラメータ

symbol_name

[in] 銘柄名

timeframe

[in] 期間

start_time

[in] 最初の要素に対応するバーの時刻

stop_time

[in] 最後の要素に対応するバーの時刻

戻り値

start_time 及び stop_time パラメータが定義された場合、指定された時間間隔でのバーの数。その他の場合、全部のバーの数。

注意事項

Bars() 関数の呼び出しまでに指定されたパラメータで時系列のデータが端末に形成されていない場合、または時系列のデータが関数呼び出しまでに取引サーバと同期化されていない場合、この関数はゼロ値を返します。

指定した範囲の日付でバーの数をリクエストする際は、その範囲に入る開始時刻のバーのみが含まれます。例えば、現在の曜日が土曜日で、start_time=最後の__火曜日とstop_time=最後の__金曜日と指定してバーの週あたりの数をリクエストすると関数は0を返します。これは週の時間軸の開始時刻が常に日曜日になっていて、1つの週足バーもこの指定した範囲に入らない為です。

履歴の中の全てのバーの数をリクエストする例：

```
   int bars=Bars(_Symbol,_Period);
   if(bars>0)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print("Number of bars in the terminal history for the symbol-period at the
moment = ",bars);
      }
   else  // バーが存在しない
      {
       //--- シンボル上のデータがサーバ上のデータと同期されていない可能性がある
       bool synchronized=false;
       //--- ループカウンタ
       int attempts=0;
       // 同期を待つ試みを 5 回
       while(attempts<5)
         {
          if(SeriesInfoInteger(Symbol(),0,SERIES_SYNCHRONIZED))
            {
             //--- 同期完了。終了する。
             synchronized=true;
             break;
            }
          //--- カウンタを増加する
          attempts++;
          //--- 次の反復まで 10 ミリ秒待つ
          Sleep(10);
         }
       //--- 同期化の後ループを終了する
       if(synchronized)
         {
          Print("Number of bars in the terminal history for the symbol-period at the
moment = ",bars);
          Print("The first date in the terminal history for the symbol-period at the
moment = ",
                (datetime)SeriesInfoInteger(Symbol(),0,SERIES_FIRSTDATE));
          Print("The first date in the history for the symbol on the server = ",
                (datetime)SeriesInfoInteger(Symbol(),0,SERIES_SERVER_FIRSTDATE));
         }
       //--- データの同期が実現しなかった
       else
         {
          Print("Failed to get number of bars for ",_Symbol);
         }
      }
```

指定したインターバルでのバーの数をリクエストする例：

```
   int n;
   datetime date1 = D'2016.09.02 23:55'; // 金曜日
   datetime date2 = D'2016.09.05 00:00'; // 月曜日
   datetime date3 = D'2016.09.08 00:00'; // 木曜日
   //---
   n=Bars(_Symbol,PERIOD_H1,D'2016.09.02 02:05',D'2016.09.02 10:55');
```

© 2000-2025, MetaQuotes Ltd.

```
   Print("バーの数: ",n); // 『バーの数:8』を出力、計算されたものには2時間バーが含まれ、11時間バーは含まれません
   n=Bars(_Symbol,PERIOD_D1,date1,date2);
   Print("バーの数: ",n); // 『バーの数:1』を出力、範囲内に一つの日足バー(月曜日のもの)の開始時刻のみ入っている為
   n=Bars(_Symbol,PERIOD_W1,date2,date3);
   Print("バーの数: ",n); // 『バーの数:0』を出力、範囲内に一つの日足バーの開始時刻も入らなかった為
```

参照

イベント処理の関数

© 2000-2025, MetaQuotes Ltd.

BarsCalculated

指定された指標の計算されたデータ数を返します。

```
int  BarsCalculated(
   int       indicator_handle,     // 指標ハンドル
   );
```

パラメータ

indicator_handle

[in] 対応する指標関数によって戻された指標ハンドル

戻り値

指標バッファで計算されたデータの量。エラーの場合（データはまだ計算されていない）-1 。

注意事項

この関数は、作成後すぐに指標データを取得することが必要な場合に便利です（指標ハンドルが利用可能）。

例：

```
void OnStart()
  {
   double Ups[];
//--- 配列に時系列の索引付けを設定する
   ArraySetAsSeries(Ups,true);
//--- フラクタル指標のハンドルを作成する
   int FractalsHandle=iFractals(NULL,0);
//--- エラーコードをリセットする
   ResetLastError();
//--- 指標値をコピーを試みる
   int i,copied=CopyBuffer(FractalsHandle,0,0,1000,Ups);
   if(copied<=0)
     {
      Sleep(50);
      for(i=0;i<100;i++)
        {
         if(BarsCalculated(FractalsHandle)>0)
            break;
         Sleep(50);
        }
      copied=CopyBuffer(FractalsHandle,0,0,1000,Ups);
      if(copied<=0)
        {
         Print("Failed to copy upper fractals. Error = ",GetLastError(),
         "i = ",i,"    copied = ",copied);
         return;
        }
      else
```

© 2000-2025, MetaQuotes Ltd.

```
         Print("Upper fractals copied",
         "i = ",i,"    copied = ",copied);
      }
   else Print("Upper fractals copied. ArraySize = ",ArraySize(Ups));
  }
```

© 2000–2025, MetaQuotes Ltd.

IndicatorCreate

この関数は iMqlParam 型のパラメータの配列に基づいて作成された特殊のテクニカル指標のハンドルを返します。

```
int  IndicatorCreate(
   string           symbol,                    // 銘柄名
   ENUM_TIMEFRAMES  period,                    // 時間軸
   ENUM_INDICATOR   indicator_type,            // ENUM_INDICATOR 列挙からの指標の種類
   int              parameters_cnt=0,          // パラメータ数
   const MqlParam&  parameters_array[]=NULL,   // パラメータの配列
   );
```

パラメータ

symbol

[in] 指標計算が基づいているデータを持つシンボルの名称。NULL は現在のシンボルです。

period

[in] 時間軸。ENUM_TIMEFRAMES 列挙のいずれかの値。0 は現在の時間軸。

indicator_type

[in] 指標の種類。ENUM_INDICATOR 列挙のいずれかの値。

parameters_cnt

[in] parameters_array[] 配列に渡されるパラメータ数。配列の要素は特殊な MqlParam 構造体型です。デフォルトではゼロで、パラメータが受け渡されません。 1 つ以上のパラメータが指定された場合 *parameters_array* は必須です。パラメータ数は 64 を超すことは出来ません。

parameters_array[]=NULL

[in] MqlParam 型の配列。要素はテクニカル指標の入力パラメータの型と値を含みます。

戻り値

指定されたテクニカル指標のハンドル。失敗の場合 INVALID_HANDLE。

注意事項

IND_CUSTOM 型の指標ハンドルが作成された場合、*tparameters_array* 入力パラメータ配列の初めの要素の*type* フィールドは tENUM_DATATYPE 列挙の TYPE_STRING 値を持ち、*string_value* フィールドはカスタム指標の名称を含まなければいけません。カスタム指標は（EX5拡張子を持つファイルに）コンパイルし、クライアント端末の MQL5/Indicators ディレクトリまたはサブディレクトリ内に配置される必要があります。

テストが必要な指標は、パラメータが文字列定数で設定されていれば、iCustom() 関数の呼び出しから自動的に定義されています。他の全ての場合（IndicatorCreate() 関数の使用や指標名を設定するパラメータでの非定数文字列の使用）#property tester_indicator プロパティが必要です。

```
#property tester_indicator "indicator_name.ex5"
```

カスタム指標が最初の呼び出し形式を使用する場合、入力パラメータを受け渡す際に最後のパラメータで計算されるデータを示すことも出来ます。「適用価格」パラメータが明確に選択されていない場合、デフォルト計算は PRICE_CLOSE 値に基づきます。

© 2000-2025, MetaQuotes Ltd.

例：

```
void OnStart()
  {
   MqlParam params[];
   int      h_MA,h_MACD;
//--- iMA("EURUSD",PERIOD_M15,8,0,MODE_EMA,PRICE_CLOSE); iを作成する
   ArrayResize(params,4);
//--- ma_period を設定する
   params[0].type          =TYPE_INT;
   params[0].integer_value=8;
//--- ma_shift を設定する
   params[1].type          =TYPE_INT;
   params[1].integer_value=0;
//--- ma_method を設定する
   params[2].type          =TYPE_INT;
   params[2].integer_value=MODE_EMA;
//--- applied_price を設定する
   params[3].type          =TYPE_INT;
   params[3].integer_value=PRICE_CLOSE;
//--- MA を作成する
   h_MA=IndicatorCreate("EURUSD",PERIOD_M15,IND_MA,4,params);
//--- iMACD("EURUSD",PERIOD_M15,12,26,9,h_MA); を作成する
   ArrayResize(params,4);
//--- 高速 ma_period を設定する
   params[0].type          =TYPE_INT;
   params[0].integer_value=12;
//--- 低速 ma_period を設定する
   params[1].type          =TYPE_INT;
   params[1].integer_value=26;
//--- 差の平滑期間を設定する
   params[2].type          =TYPE_INT;
   params[2].integer_value=9;
//--- 指標ハンドルを applied_price として設定する
   params[3].type          =TYPE_INT;
   params[3].integer_value=h_MA;
//--- 移動平均に基づいて MACD を作成する
   h_MACD=IndicatorCreate("EURUSD",PERIOD_M15,IND_MACD,4,params);
//--- 指標を使用する
//--- . にて。にて。
//--- 指標を解放する(初めの h_MACD)
   IndicatorRelease(h_MACD);
   IndicatorRelease(h_MA);
  }
```

© 2000-2025, MetaQuotes Ltd.

IndicatorParameters

指定されたハンドルに基づいて、指標の入力パラメータの数、値と型を返します。

```
int  IndicatorParameters(
   int               indicator_handle,     // 指標ハンドル
   ENUM_INDICATOR&   indicator_type,       // 指標の種類を受け取る変数
   MqlParam&         parameters[]          // パラメータを受け取る配列
   );
```

パラメータ

indicator_handle

[in] 計算されたパラメータの数を知る必要のある指標のハンドル

indicator_type

[out] 指標の種類が書き入れられる ENUM_INDICATOR 型の変数

parameters[]

[out] 指標パラメータのリストが書き入れられる MqlParam 型の値を受け取る動的配列。配列サイズは IndicatorParameters() 関数で返されます。

戻り値

指定されたハンドルを持つ指標の入力パラメータの数。エラーの場合は -1。エラーの詳細については GetLastError() を呼びます。

例:

```
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {

//--- チャート上のウィンドウの数(少なくとも1つのメインウィンドウは常に存在する)
   int windows=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
//--- チャートウィンドウを全部見る
   for(int w=0;w<windows;w++)
     {
      //--- このウィンドウ/サブウィンドウ内の指標の数
      int total=ChartIndicatorsTotal(0,w);
      //--- ウィンドウ内の指標を全て取る
      for(int i=0;i<total;i++)
        {
         //--- 指標の短縮名を取得する
         string name=ChartIndicatorName(0,w,i);
         //--- 指標ハンドルを取得する
         int handle=ChartIndicatorGet(0,w,name);
         //--- ログに加える
         PrintFormat("Window=%d,  indicator #%d,  handle=%d",w,i,handle);
         //---
```

© 2000-2025, MetaQuotes Ltd.

```
         MqlParam parameters[];
         ENUM_INDICATOR indicator_type;
         int params=IndicatorParameters(handle,indicator_type,parameters);
         //--- メッセージヘッダー
         string par_info="Short name "+name+", type "
                         +EnumToString(ENUM_INDICATOR(indicator_type))+"\r\n";
         //---
         for(int p=0;p<params;p++)
           {
            par_info+=StringFormat("parameter %d: type=%s, long_value=%d,
double_value=%G,string_value=%s\r\n",
                                   p,
                                   EnumToString((ENUM_DATATYPE)parameters[p].type),
                                   parameters[p].integer_value,
                                   parameters[p].double_value,
                                   parameters[p].string_value
                                   );
           }
         Print(par_info);
        }
      //--- ウィンドウの全ての指標で完成
     }
//---
  }
```

参照

ChartIndicatorGet()

© 2000-2025, MetaQuotes Ltd.

IndicatorRelease

この関数は、指標ハンドルを削除し、誰によっても使用されていない場合は指標の計算ブロックを解放します。

```
bool  IndicatorRelease(
   int       indicator_handle     // 指標ハンドル
   );
```

戻り値

成功の場合は true、それ以外の場合は false

注意事項

この関数は不要になった指標ハンドルを削除しメモリを節約します。ハンドルがすぐに削除され演算ブロックは（もはや呼び出されていない場合）しばらくたって削除されます。

ストラテジーテスタでの作業ではIndicatorRelease() 関数は実行されません。

例：

```
//+------------------------------------------------------------------+
//|                                        Test_IndicatorRelease.mq5 |
//|                        Copyright 2010, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "2010, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- 入力パラメータ
input int                MA_Period=15;
input int                MA_shift=0;
input ENUM_MA_METHOD     MA_smooth=MODE_SMA;
input ENUM_APPLIED_PRICE price=PRICE_CLOSE;
//--- 指標ハンドルを格納する
int MA_handle=INVALID_HANDLE;
//+------------------------------------------------------------------+
//| エキスパート初期化に使用される関数                                |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 指標ハンドルを作成する
   MA_handle=iMA(Symbol(),0,MA_Period,MA_shift,MA_smooth,PRICE_CLOSE);
//--- グローバル変数を削除する
   if(GlobalVariableCheck("MA_value"))
      GlobalVariableDel("MA_value");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| エキスパートティック関数                                          |
```

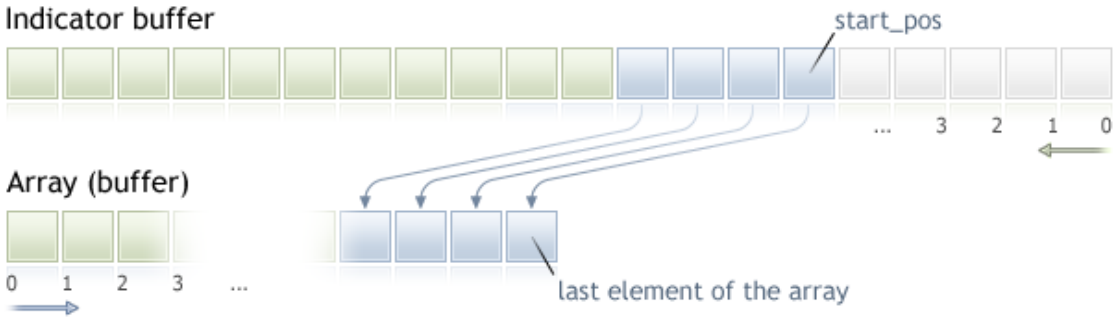
© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnTick()
  {
//--- グローバル変数の値が存在しない場合
   if(!GlobalVariableCheck("MA_value"))
     {
      //--- 最後の 2 つのバーの指標値を取得する
      if(MA_handle!=INVALID_HANDLE)
        {
         //--- 指標値の動的配列
         double values[];
         if(CopyBuffer(MA_handle,0,0,2,values)==2 && values[0]!=EMPTY_VALUE)
           {
            //--- 最後から2 番目のバーをグローバル変数の値に保存する
            if(GlobalVariableSet("MA_value",values[0]))
              {
               //--- 指標ハンドルを解放する
               if(!IndicatorRelease(MA_handle))
                  Print("IndicatorRelease() failed. Error ",GetLastError());
               else MA_handle=INVALID_HANDLE;
              }
            else
               Print("GlobalVariableSet failed. Error ",GetLastError());
           }
        }
     }
//---
  }
```

© 2000–2025, MetaQuotes Ltd.

CopyBuffer

指標の指定されたバッファデータを取得します。



複製されたデータの要素は、開始位置から現在から過去に向かって数えられます（インデックス buffer_num を持つ指標バッファ）。すなわち、0 の開始位置は現在足（の指標値）を意味します。

CopyBuffer() 関数は受け取り側の配列のサイズの複製されたデータへの割り当てを試みるので、複製するデータ量が不明の場合、受け取り側の buffer[] 配列としての動的配列の使用が推奨されます。指標バッファ（SetIndexBufer() 関数で指標値を格納するために事前に割り当てられる配列）が受け取り側の buffer[] 配列として使用された場合、部分的な複製が許可されています。例は標準端末パッケージの Awesome_Oscillator.MQL5 カスタム指標に見られます。

別の配列（非指標バッファ）に指示値の部分的なコピーを作成する必要がある場合は、所望の数が複製される中間配列を使用する必要があります。その後、この中間配列から受け取り側の配列の必要な場所に必要な数の値の要素ごとのコピーを行います。

複製する必要があるデータの量がわかっている場合は、過剰なメモリの割り当てを防止するために静的に割り当てられたバッファを受け取り側の配列として使用するべきです。

受け取り側の配列が as_series=true であるか as_series=false であるかは関係ありません。データは、一番古い要素が配列に割り当てられた物理メモリの先頭に配置されるように複製されます。この関数の呼び出し方は 3 つあります。

1 番目の位置と必要な要素数によっての呼び出し

```
int  CopyBuffer(
   int       indicator_handle,     // 指標ハンドル
   int       buffer_num,           // 指標バッファ番号
   int       start_pos,            // 開始位置
   int       count,                // 複製する量
   double    buffer[]              // 受け取り側の配列
   );
```

開始日と必要な要素数によっての呼び出し

```
int  CopyBuffer(
   int       indicator_handle,     // 指標ハンドル
   int       buffer_num,           // 指標バッファ番号
   datetime  start_time,           // 開始日と時刻
```

© 2000-2025, MetaQuotes Ltd.

```
   int       count,                  // 複製する量
   double    buffer[]                // 受け取り側の配列
   );
```

必要な時間間隔の開始日と終了日によっての呼び出し

```
int  CopyBuffer(
   int       indicator_handle,       // 指標ハンドル
   int       buffer_num,             // 指標バッファ番号
   datetime  start_time,             // 開始日と時刻
   datetime  stop_time,              // 終了日と時刻
   double    buffer[]                // 受け取り側の配列
   );
```

パラメータ

indicator_handle

[in] 対応する指標関数によって戻された指標ハンドル

buffer_num

[in] 指標バッファの番号

start_pos

[in] 1番目の複製される要素の位置

count

[in] 複製する日数

start_time

[in] 最初の要素に対応するバーの時刻

stop_time

[in] 最後の要素に対応するバーの時刻

buffer[]

[out] double 型の配列

戻り値

複製されたデータ数（エラーの場合は -1 ）

注意事項

指標からデータがリクエストされて、リクエストされた時系列がまだ構築されていないかサーバからダウンロードされる必要がある場合、この関数はすぐに -1 を返し、ダウンロード/構築のプロセスが開始されます。

エキスパートアドバイザーやスクリプトからデータがリクエストされて、端末が必要なデータをローカルに持っていない場合サーバからのダウンロードが開始されます。データがローカル履歴から構築することが出来るがまだされていない場合、必要な時系列の構築が開始します。この関数は、タイムアウトの有効期限までに準備されるデータの量を返します。

例：

```
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|                                                TestCopyBuffer3.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- MA をプロットする
#property indicator_label1  "MA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 入力パラメータ
input bool                AsSeries=true;
input int                 period=15;
input ENUM_MA_METHOD      smootMode=MODE_EMA;
input ENUM_APPLIED_PRICE  price=PRICE_CLOSE;
input int                 shift=0;
//--- 指標バッファ
double                    MABuffer[];
int                       ma_handle;
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 指標バッファマッピング
   SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
   Print("Parameter AsSeries = ",AsSeries);
   Print("Indicator buffer after SetIndexBuffer() is a timeseries = ",
         ArrayGetAsSeries(MABuffer));
//--- 指標短縮名を設定する
   IndicatorSetString(INDICATOR_SHORTNAME,"MA("+period+")"+AsSeries);
//--- AsSeries を設定する(入力パラメータによる)
   ArraySetAsSeries(MABuffer,AsSeries);
   Print("Indicator buffer after ArraySetAsSeries(MABuffer,true); is a timeseries =
",
         ArrayGetAsSeries(MABuffer));
//---
   ma_handle=iMA(Symbol(),0,period,shift,smootMode,price);
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- データが全て計算されたかをチェックする
   if(BarsCalculated(ma_handle)<rates_total) return(0);
//--- 全てのデータのコピーは出来ない
   int to_copy;
   if(prev_calculated>rates_total || prev_calculated<=0) to_copy=rates_total;
   else
     {
      to_copy=rates_total-prev_calculated;
      //--- 最終値は常に複製される
      to_copy++;
     }
//--- コピーを試みる
   if(CopyBuffer(ma_handle,0,0,to_copy,MABuffer)<=0) return(0);
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

上記の例は、指標バッファが同じ銘柄/期間の指標から作られた別の指標バッファの値で記入される方法を示しています。

履歴データリクエストの詳細な例はオブジェクト結合の方法で参照出来ます。そのセクションで使用可能なスクリプトは iFractals 指標の最後の 1,000 足の値を取得して最後の各 10 の上向きと下向きのフラクタルをチャートに表示する方法を示しています。データ欠落時には、同様の技術が通常次のスタイルを使用して描画されている指標全てで使用することが出来ます。

- DRAW_SECTION
- DRAW_ARROW
- DRAW_ZIGZAG
- DRAW_COLOR_SECTION
- DRAW_COLOR_ARROW
- DRAW_COLOR_ZIGZAG

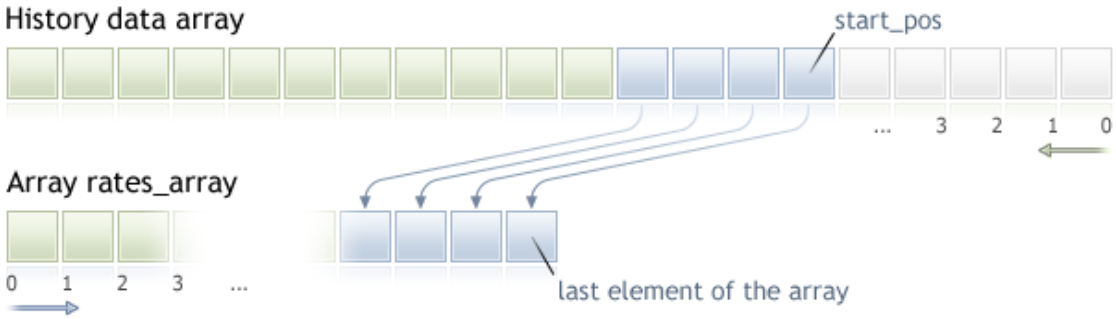
参照

カスタム指標のプロパティ、SetIndexBuffer

© 2000-2025, MetaQuotes Ltd.

CopyRates

指定された銘柄と期間の MqlRates 構造体の指定された量の履歴データを rates_array 配列に配置します。要素は現在から過去の順に並べられ、インデックス0が現在足です。



リクエストされたデータ数が受け取り側の配列の長さよりも小さい(または大きい)と、この関数はリクエストされたデータが完全に収まるようにメモリを再割り当てしようとするので、複製するデータ量が不明の場合、受け取り側の配列に動的配列を使用することが推奨されます。

複製する必要があるデータの量がわかっている場合は、過剰なメモリの割り当てを防止するために静的に割り当てられたバッファを受け取り側の配列として使用するべきです。

受け取り側の配列が as_series=true であるか as_series=false であるかは関係ありません。データは、一番古い要素が配列に割り当てられた物理メモリの先頭に配置されるように複製されます。この関数の呼び出し方は 3 つあります。

1 番目の位置と必要な要素数によっての呼び出し

```
int  CopyRates(
   string           symbol_name,       // 銘柄名
   ENUM_TIMEFRAMES  timeframe,         // 期間
   int              start_pos,         // 開始位置
   int              count,             // 複製するデータ数
   MqlRates         rates_array[]      // 受け取り側の配列
   );
```

開始日と必要な要素数によっての呼び出し

```
int  CopyRates(
   string           symbol_name,       // 銘柄名
   ENUM_TIMEFRAMES  timeframe,         // 期間
   datetime         start_time,        // 開始日と時刻
   int              count,             // 複製するデータ数
   MqlRates         rates_array[]      // 受け取り側の配列
   );
```

必要な時間間隔の開始日と終了日によっての呼び出し

```
int  CopyRates(
   string           symbol_name,       // 銘柄名
```

© 2000-2025, MetaQuotes Ltd.

```
   ENUM_TIMEFRAMES   timeframe,          // 期間
   datetime          start_time,         // 開始日と時刻
   datetime          stop_time,          // 終了日と時刻
   MqlRates          rates_array[]       // 受け取り側の配列
   );
```

パラメータ

symbol_name

[in] 銘柄名

timeframe

[in] 期間

start_time

[in] 複製される最後の要素のバーの時刻

start_pos

[in] 1番目の複製される要素の開始位置

count

[in] 複製する日数

stop_time

[in] 複製される最後の要素に対応するバーの時刻

rates_array[]

[out] MqlRates 型の配列

戻り値

複製された要素数（エラーの場合は -1 ）

注意事項

リクエストされたデータがサーバ上で利用可能なデータに含まれていない場合、この関数は -1 を返します。TERMINAL_MAXBARS（チャートでのバーの最大数）以外のデータがリクエストされた場合も、この関数は -1を返します。

指標からデータがリクエストされて、リクエストされた時系列がまだ構築されていないかサーバからダウンロードされる必要がある場合、この関数はすぐに -1 を返し、ダウンロード/構築のプロセスが開始されます。

エキスパートアドバイザーやスクリプトからデータがリクエストされて、端末が必要なデータをローカルに持っていない場合サーバからのダウンロードが開始されます。データがローカル履歴から構築することが出来るがまだされていない場合、必要な時系列の構築が開始します。この関数は、タイムアウトの有効期限までに準備されるデータの量を返します。が、履歴ダウンロードは続き、次に似たようなリクエストがされた際に、もっとデータを返します。

開始日及び必要な要素の数によってデータがリクエストされる場合、指定された日付とそれ以前のデータのみが戻されます。これは、値（指標バッファのボリューム、スプレッド、値、始値、高値、安値、終値、開始時刻、決済時刻）が返されたバーの開始時刻は常に指定されたもの以下だということです。

日付の指定された範囲のデータを要求する場合、この区間からのデータのみが返されます。間隔が設定され秒まで数えられます。これは、値（指標バッファのボリューム、スプレッド、値、始値、高値、安値、終値、開始時刻、決済時刻）が返されたバーの開始時刻は常に指定された区間内にあるということです。

© 2000-2025, MetaQuotes Ltd.

今日が土曜日で一週間の時間軸で *start_time=Last_Tuesday* 及び *stop_time=Last_Friday* を指定してデータを複製する試みに、この関数は 0 を返します。一週間の時間軸が開くのは常に日曜日ですが、一週間バーは指定された間隔にはまらないからです。

現在の未完成のバーに対応する値を返す必要がある場合 *start_pos*=0 及び *count*=1 を指定して1 番目の呼び出しの方法を用います。

例:

```
void OnStart()
  {
//---
   MqlRates rates[];
   ArraySetAsSeries(rates,true);
   int copied=CopyRates(Symbol(),0,0,100,rates);
   if(copied>0)
     {
      Print("Bars copied: "+copied);
      string format="open = %G, high = %G, low = %G, close = %G, volume = %d";
      string out;
      int size=fmin(copied,10);
      for(int i=0;i<size;i++)
        {
         out=i+":"+TimeToString(rates[i].time);
         out=out+" "+StringFormat(format,
                                  rates[i].open,
                                  rates[i].high,
                                  rates[i].low,
                                  rates[i].close,
                                  rates[i].tick_volume);
         Print(out);
        }
     }
   else Print("Failed to get history data for the symbol ",Symbol());
  }
```

履歴データリクエストの詳細な例はオブジェクト結合の方法で参照出来ます。そのセクションで使用可能なスクリプトは iFractals 指標の最後の 1,000 足の値を取得して最後の各 10 の上向きと下向きのフラクタルをチャートに表示する方法を示しています。データ欠落時には、同様の技術が通常次のスタイルを使用して描画されている指標全てで使用することが出来ます。

- DRAW_SECTION
- DRAW_ARROW
- DRAW_ZIGZAG
- DRAW_COLOR_SECTION
- DRAW_COLOR_ARROW
- DRAW_COLOR_ZIGZAG

参照

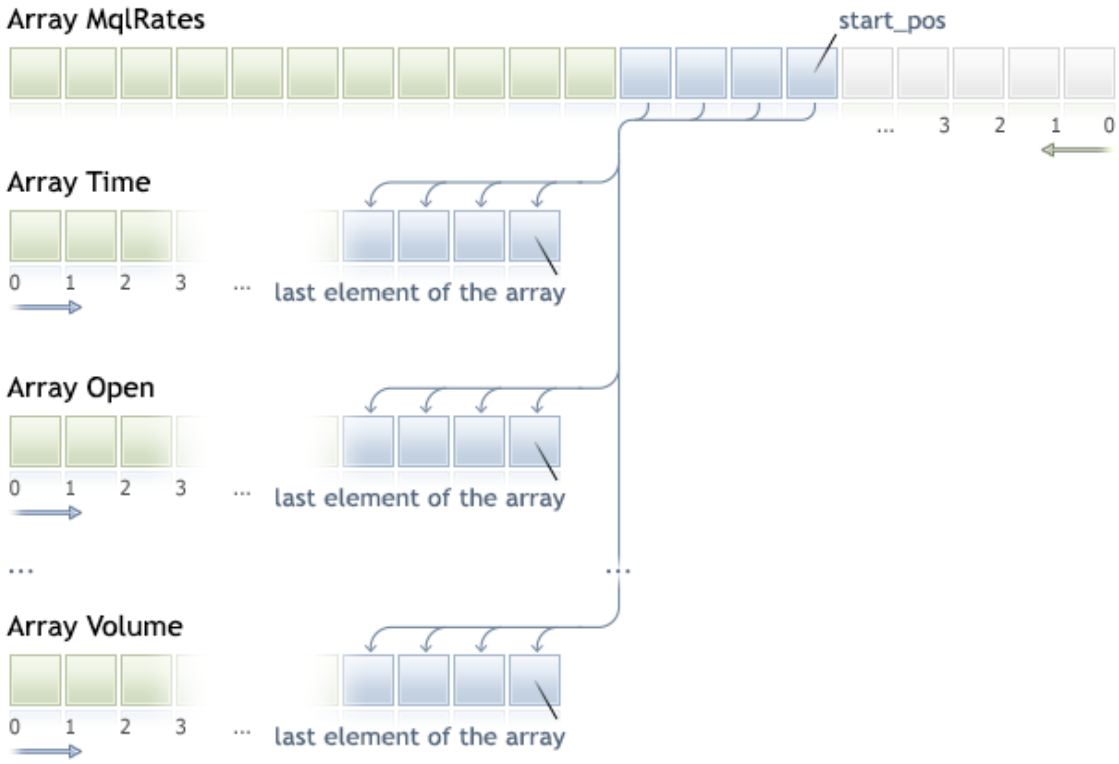
© 2000-2025, MetaQuotes Ltd.

構造体とクラス、TimeToString、StringFormat

© 2000–2025, MetaQuotes Ltd.

CopySeries

MqlRates構造体から、指定された銘柄期間と指定された量の同期済み時系列を取得します。データは、指定された配列のセットに受信されます。要素は現在から過去までカウントダウンされます。つまり、開始位置が0の場合は現在足を意味します。



複製するデータ量が不明な場合は、受信配列に動的配列を使用することをお勧めします。データ量が配列に格納できる量を超えると、要求されたすべてのデータに適合するように配列を再分配できるからです。

あらかじめ決められた量のデータを複製する必要がある場合は、不必要なメモリの再割り当てを避けるために、静的に割り当てられたバッファを使用することをお勧めします。

受信配列のプロパティ(as_series=true または as_series=false)は無視されます。複製中は、最も古い時系列要素が配列に割り当てられた物理メモリの先頭に複製されます。

```
int  CopySeries(
   string           symbol_name,      // 銘柄名
   ENUM_TIMEFRAMES  timeframe,        // 期間
   int              start_pos,        // 開始位置
   int              count,            // 複製する量
   ulong            rates_mask,       // 要求されたシリーズを指定するフラグの組み合わせ
   void&            array1[],         // 最初にコピーの時系列のデータを受け取る配列
   void&            array2[]          // 2番目のコピーの時系列のデータを受け取る配列
   ...
   );
```

パラメータ

© 2000-2025, MetaQuotes Ltd.

symbol_name

[in] シンボル

timeframe

[in] 期間

start_pos

[in] 最初に複製された要素のインデックス

count

[in] 複製された要素の数

rates_mask

[in] ENUM_COPY_RATES列挙体からのフラグの組み合わせ

array1, array2,...

[out] MqlRates構造体から時系列を受け取る適切な型の配列。関数に渡される配列の順序は、MqlRates構造内のフィールドの順序と一致する必要があります。

戻り値

複製された要素の数（エラーの場合は-1）

注意事項

要求されたデータがサーバ上で利用可能なデータに含まれていない場合、この関数は-1を返します。TERMINAL_MAXBARS（チャートでのバーの最大数）を越えたデータが要求された場合も、この関数は-1を返します。

指標からデータを要求するとき、要求された時系列がまだ構築されていない場合、またはサーバからダウンロードする必要がある場合、関数はすぐに-1を返します。ただし、これにより、データのダウンロード/構築自体が開始されます。

エキスパートアドバイザーまたはスクリプトからデータを要求するとき、端末が適切なデータをローカルに持っていない場合、またはデータがローカル履歴から構築できるがまだ準備ができていない場合、サーバからのダウンロードが開始されます。 この関数は、タイムアウトの期限が切れるまでに準備ができているデータの量を返しますが、履歴のダウンロードは続行され、関数は次の同様の要求でさらに多くのデータを返します。

CopySeriesとCopyRatesの違い

CopySeries関数を使用すると、1回の呼び出しで必要な時系列のみを別の指定された配列に取得できますが、時系列データはすべて同期されます。これは、特定のインデックスNで結果として得られる配列のすべての値が、指定された銘柄/時間枠のペアの同じバーに属することを意味します。したがって、プログラマーがバーの開始時間までに受信したすべての時系列の同期を確保する必要はありません。

時系列の完全なセットをMqlRates配列として返すCopyRatesとは異なり、CopySeries関数を使用すると、プログラマーは必要な時系列のみを個別の配列として取得できます。これは、フラグの組み合わせを指定して時系列のタイプを選択することで実行できます。関数に渡される配列の順序は、MqlRates構造内のフィールドの順序と一致する必要があります。

```
struct MqlRates
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   datetime time;          // 期間開始時間
   double   open;          // 始値
   double   high;          // 期間の高値
   double   low;           // 期間の安値
   double   close;         // 終値
   long     tick_volume;   // ティックボリューム
   int      spread;        // スプレッド
   long     real_volume;   // 取引高
  }
```

したがって、現在の銘柄/時間枠の最後の100バーのtime、close、real_volume時系列の値を取得する必要がある場合は、次の呼び出しを使用する必要があります。

```
datetime time[];
double   close[];
long     volume[];
CopySeries(NULL,0,0,100,COPY_RATES_TIME|COPY_RATES_CLOSE|
COPY_RATES_VOLUME_REAL,time,close,volume);
```

time、close、volume配列の順番に気を付けてください。これは、 MqlRates構造体のフィールドの値と同じでなければいけません。rates_maskの値の順序は重要ではありません。マスクは次のようになります。

```
COPY_RATES_VOLUME_REAL|COPY_RATES_TIME|COPY_RATES_CLOSE
```

例：

```
//--- 入力パラメータ
input datetime InpDateFrom=D'2022.01.01 00:00:00';
input datetime InpDateTo  =D'2023.01.01 00:00:00';
input uint     InpCount   =20;
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart(void)
  {
//--- MqlRates価格構造体から時系列を取得するための配列
   double   open[];
   double   close[];
   float    closef[];
   datetime time1[], time2[];
//--- double 配列に終値を要求する
   ResetLastError();
   int res1=CopySeries(NULL, PERIOD_CURRENT, 0, InpCount,
                       COPY_RATES_TIME|COPY_RATES_CLOSE, time1, close);
   PrintFormat("1. CopySeries  returns %d values. Error code=%d", res1,
GetLastError());
   ArrayPrint(close);

//--- 始値も要求する; 終値にはfloat配列を使用
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
   int res2=CopySeries(NULL, PERIOD_CURRENT, 0, InpCount,
                       COPY_RATES_TIME|COPY_RATES_CLOSE|COPY_RATES_OPEN, time2,
open, closef);
   PrintFormat("2. CopySeries  returns %d values. Error code=%d", res2,
GetLastError());
   ArrayPrint(closef);
//--- 受信したデータを比較する
   if((res1==res2) && (time1[0]==time2[0]))
     {
      Print("  | Time              |    Open      | Close double | Close float |");
      for(int i=0; i<10; i++)
        {
         PrintFormat("%d | %s |   %.5f    |   %.5f    |   %.5f   |",
                     i, TimeToString(time1[i]), open[i], close[i], closef[i]);
        }
     }
//--- 結果
      1. CopySeriesは20の値を戻します。Error code=0
      [ 0] 1.06722 1.06733 1.06653 1.06520 1.06573 1.06649 1.06694 1.06675 1.06684
1.06604
      [10] 1.06514 1.06557 1.06456 1.06481 1.06414 1.06394 1.06364 1.06386 1.06239
1.06247
      2. CopySeriesは20の値を戻します。Error code=0
      [ 0] 1.06722 1.06733 1.06653 1.06520 1.06573 1.06649 1.06694 1.06675 1.06684
1.06604
      [10] 1.06514 1.06557 1.06456 1.06481 1.06414 1.06394 1.06364 1.06386 1.06239
1.06247
        | Time              |    Open      | Close double | Close float |
      0 | 2023.03.01 17:00 |   1.06660    |   1.06722    |   1.06722   |
      1 | 2023.03.01 18:00 |   1.06722    |   1.06733    |   1.06733   |
      2 | 2023.03.01 19:00 |   1.06734    |   1.06653    |   1.06653   |
      3 | 2023.03.01 20:00 |   1.06654    |   1.06520    |   1.06520   |
      4 | 2023.03.01 21:00 |   1.06520    |   1.06573    |   1.06573   |
      5 | 2023.03.01 22:00 |   1.06572    |   1.06649    |   1.06649   |
      6 | 2023.03.01 23:00 |   1.06649    |   1.06694    |   1.06694   |
      7 | 2023.03.02 00:00 |   1.06683    |   1.06675    |   1.06675   |
      8 | 2023.03.02 01:00 |   1.06675    |   1.06684    |   1.06684   |
      9 | 2023.03.02 02:00 |   1.06687    |   1.06604    |   1.06604   |
//---
  }
```

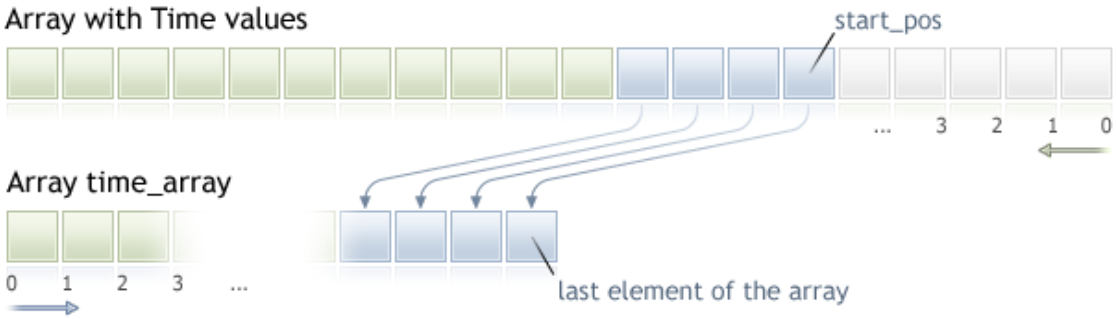
参照

構造体とクラス、CopyRates

© 2000–2025, MetaQuotes Ltd.

CopyTime

この関数は選択された銘柄/期間ペアのバーの開始時刻の指定された量の履歴データをtime_array に配置します。要素が現在から過去の順に並べられており、インデックス0が現在足であることにご注意下さい



リクエストされたデータ数が受け取り側の配列の長さよりも小さい(または大きい)と、この関数はリクエストされたデータが完全に収まるようにメモリを再割り当てしようとするので、複製するデータ量が不明の場合、受け取り側の配列に動的配列を使用することが推奨されます。

複製する必要があるデータの量がわかっている場合は、過剰なメモリの割り当てを防止するために静的に割り当てられたバッファを受け取り側の配列として使用するべきです。

受け取り側の配列がas_series=true であるかas_series=false であるかは関係ありません。データは、一番古い要素が配列に割り当てられた物理メモリの先頭に配置されるように複製されます。この関数の呼び出し方は 3 つあります。

1 番目の位置と必要な要素数によっての呼び出し

```
int  CopyTime(
   string           symbol_name,      // 銘柄名
   ENUM_TIMEFRAMES  timeframe,        // 期間
   int              start_pos,        // 開始位置
   int              count,            // 複製するデータ数
   datetime         time_array[]      // 開始時刻を複製する受け取り側の配列
   );
```

開始日と必要な要素数によっての呼び出し

```
int  CopyTime(
   string           symbol_name,      // 銘柄名
   ENUM_TIMEFRAMES  timeframe,        // 期間
   datetime         start_time,       // 開始日付と時刻
   int              count,            // 複製するデータ数
   datetime         time_array[]      // 開始時刻を複製する受け取り側の配列
   );
```

必要な時間間隔の開始日と終了日によっての呼び出し

```
int  CopyTime(
   string           symbol_name,      // 銘柄名
```

© 2000-2025, MetaQuotes Ltd.

```
   ENUM_TIMEFRAMES   timeframe,        // 期間
   datetime          start_time,       // 開始日付と時刻
   datetime          stop_time,        // 終了日と時刻
   datetime          time_array[]      // 開始時刻を複製する受け取り側の配列
   );
```

パラメータ

symbol_name
[in] 銘柄名

timeframe
[in] 期間

start_pos
[in] 1番目の複製される要素の開始位置

count
[in] 複製する日数

start_time
[in] 1番目の複製される要素の開始時刻

stop_time
[in] 複製される最後の要素に対応するバーの時刻

time_array[]
[out] datetime 型の配列

戻り値

複製されたデータ数（エラーの場合は -1 ）

注意事項

リクエストされたデータがサーバ上で利用可能なデータに含まれていない場合、この関数は -1 を返します。TERMINAL_MAXBARS（チャートでのバーの最大数）以外のデータがリクエストされた場合も、この関数は -1を返します。

指標からデータがリクエストされて、リクエストされた時系列がまだ構築されていないかサーバからダウンロードされる必要がある場合、この関数はすぐに -1 を返し、ダウンロード/構築のプロセスが開始されます。

エキスパートアドバイザーやスクリプトからデータがリクエストされて、端末が必要なデータをローカルに持っていない場合サーバからのダウンロードが開始されます。データがローカル履歴から構築することが出来るがまだされていない場合、必要な時系列の構築が開始します。この関数は、タイムアウトの有効期限までに準備されるデータの量を返します。が、履歴ダウンロードは続き、次に似たようなリクエストがされた際に、もっとデータを返します。

開始日及び必要な要素の数によってデータがリクエストされる場合、指定された日付とそれ以前のデータのみが戻されます。これは、値（指標バッファのボリューム、スプレッド、値、始値、高値、安値、終値、開始時刻、決済時刻）が返されたバーの開始時刻は常に指定されたもの以下だということです。

日付の指定された範囲のデータを要求する場合、この区間からのデータのみが返されます。間隔が設定され秒まで数えられます。これは、値（指標バッファのボリューム、スプレッド、値、始値、高値、安値、終値、開始時刻、決済時刻）が返されたバーの開始時刻は常に指定された区間内にあるということです。

© 2000-2025, MetaQuotes Ltd.

今日が土曜日で一週間の時間軸で *start_time=Last_Tuesday* 及び *stop_time=Last_Friday* を指定してデータを複製する試みに、この関数は 0 を返します。一週間の時間軸が開くのは常に日曜日ですが、一週間バーは指定された間隔にはまらないからです。

現在の未完成のバーに対応する値を返す必要がある場合 *start_pos*=0 及び *count*=1 を指定して1 番目の呼び出しの方法を用います。

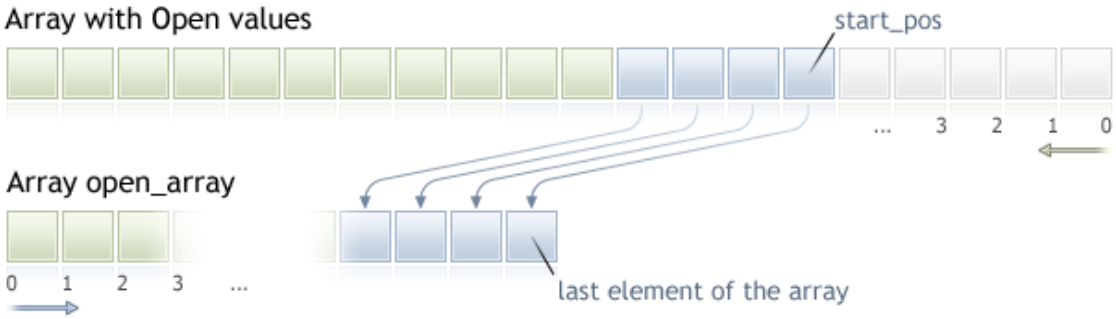
履歴データリクエストの詳細な例はオブジェクト結合の方法で参照出来ます。そのセクションで使用可能なスクリプトは iFractals 指標の最後の 1,000 足の値を取得して最後の各 10 の上向きと下向きのフラクタルをチャートに表示する方法を示しています。データ欠落時には、同様の技術が通常次のスタイルを使用して描画されている指標全てで使用することが出来ます。

- DRAW_SECTION
- DRAW_ARROW
- DRAW_ZIGZAG
- DRAW_COLOR_SECTION
- DRAW_COLOR_ARROW
- DRAW_COLOR_ZIGZAG

© 2000-2025, MetaQuotes Ltd.

CopyOpen

この関数は選択された銘柄/期間ペアのバーの始値の指定された量の履歴データをopen_array に配置します。要素が現在から過去の順に並べられており、インデックス0が現在足であることにご注意下さい。



リクエストされたデータ数が受け取り側の配列の長さよりも小さい(または大きい)と、この関数はリクエストされたデータが完全に収まるようにメモリを再割り当てしようとするので、複製するデータ量が不明の場合、受け取り側の配列に動的配列を使用することが推奨されます。

複製する必要があるデータの量がわかっている場合は、過剰なメモリの割り当てを防止するために静的に割り当てられたバッファを受け取り側の配列として使用するべきです。

受け取り側の配列がas_series=true であるかas_series=false であるかは関係ありません。データは、一番古い要素が配列に割り当てられた物理メモリの先頭に配置されるように複製されます。この関数の呼び出し方は3つあります。

1番目の位置と必要な要素数によっての呼び出し

```
int  CopyOpen(
   string           symbol_name,      // 銘柄名
   ENUM_TIMEFRAMES  timeframe,        // 期間
   int              start_pos,        // 開始位置
   int              count,            // 複製するデータ数
   double           open_array[]      // 始値の複製先の配列
   );
```

開始日と必要な要素数によっての呼び出し

```
int  CopyOpen(
   string           symbol_name,      // 銘柄名
   ENUM_TIMEFRAMES  timeframe,        // 期間
   datetime         start_time,       // 開始日付と時刻
   int              count,            // 複製するデータ数
   double           open_array[]      // バーの始値の複製先の配列
   );
```

必要な時間間隔の開始日と終了日によっての呼び出し

```
int  CopyOpen(
   string           symbol_name,      // 銘柄名
```

© 2000-2025, MetaQuotes Ltd.

```
   ENUM_TIMEFRAMES   timeframe,       // 期間
   datetime          start_time,      // 開始日付と時刻
   datetime          stop_time,       // 終了日と時刻
   double            open_array[]     // バーの始値の複製先の配列
   );
```

パラメータ

symbol_name
[in] 銘柄名

timeframe
[in] 期間

start_pos
[in] 1番目の複製される要素の開始位置

count
[in] 複製する日数

start_time
[in] 1番目の複製される要素の開始時刻

stop_time
[in] 最後に複製される要素の開始時刻

open_array[]
[out] double 型の配列

戻り値

配列の要素数（エラーの場合は -1）

注意事項

リクエストされたデータがサーバ上で利用可能なデータに含まれていない場合、この関数は -1 を返します。TERMINAL_MAXBARS（チャートでのバーの最大数）以外のデータがリクエストされた場合も、この関数は -1を返します。

指標からデータがリクエストされて、リクエストされた時系列がまだ構築されていないかサーバからダウンロードされる必要がある場合、この関数はすぐに -1 を返し、ダウンロード/構築のプロセスが開始されます。

エキスパートアドバイザーやスクリプトからデータがリクエストされて、端末が必要なデータをローカルに持っていない場合サーバからのダウンロードが開始されます。データがローカル履歴から構築することが出来るがまだされていない場合、必要な時系列の構築が開始します。この関数は、タイムアウトの有効期限までに準備されるデータの量を返します。が、履歴ダウンロードは続き、次に似たようなリクエストがされた際に、もっとデータを返します。

開始日及び必要な要素の数によってデータがリクエストされる場合、指定された日付とそれ以前のデータのみが戻されます。これは、値（指標バッファのボリューム、スプレッド、値、始値、高値、安値、終値、開始時刻、決済時刻）が返されたバーの開始時刻は常に指定されたもの以下だということです。

日付の指定された範囲のデータを要求する場合、この区間からのデータのみが返されます。間隔が設定され秒まで数えられます。これは、値（指標バッファのボリューム、スプレッド、値、始値、高値、安値、終値、開始時刻、決済時刻）が返されたバーの開始時刻は常に指定された区間内にあるということです。

© 2000-2025, MetaQuotes Ltd.

今日が土曜日で一週間の時間軸で *start_time=Last_Tuesday* 及び *stop_time=Last_Friday* を指定してデータを複製する試みに、この関数は 0 を返します。一週間の時間軸が開くのは常に日曜日ですが、一週間バーは指定された間隔にはまらないからです。

現在の未完成のバーに対応する値を返す必要がある場合 *start_pos*=0 及び *count*=1 を指定して1 番目の呼び出しの方法を用います。

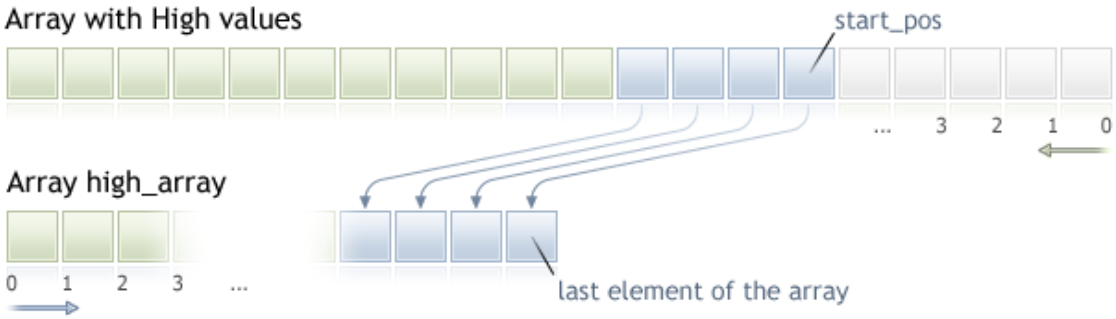
履歴データリクエストの詳細な例はオブジェクト結合の方法で参照出来ます。そのセクションで使用可能なスクリプトは iFractals 指標の最後の 1,000 足の値を取得して最後の各 10 の上向きと下向きのフラクタルをチャートに表示する方法を示しています。データ欠落時には、同様の技術が通常次のスタイルを使用して描画されている指標全てで使用することが出来ます。

- DRAW_SECTION
- DRAW_ARROW
- DRAW_ZIGZAG
- DRAW_COLOR_SECTION
- DRAW_COLOR_ARROW
- DRAW_COLOR_ZIGZAG

© 2000–2025, MetaQuotes Ltd.

CopyHigh

この関数は選択された銘柄/期間ペアのバーの終値の指定された量の履歴データをclose_array に配置します。要素が現在から過去の順に並べられており、インデックス0が現在足であることにご注意下さい。



リクエストされたデータ数が受け取り側の配列の長さよりも小さい(または大きい)と、この関数はリクエストされたデータが完全に収まるようにメモリを再割り当てしようとするので、複製するデータ量が不明の場合、受け取り側の配列に動的配列を使用することが推奨されます。

複製する必要があるデータの量がわかっている場合は、過剰なメモリの割り当てを防止するために静的に割り当てられたバッファを受け取り側の配列として使用するべきです。

受け取り側の配列がas_series=true であるかas_series=false であるかは関係ありません。データは、一番古い要素が配列に割り当てられた物理メモリの先頭に配置されるように複製されます。この関数の呼び出し方は3つあります。

1番目の位置と必要な要素数によっての呼び出し

```
int  CopyHigh(
   string           symbol_name,      // 銘柄名
   ENUM_TIMEFRAMES  timeframe,        // 期間
   int              start_pos,        // 開始位置
   int              count,            // 複製するデータ数
   double           high_array[]      // 受け取り側の配列
   );
```

開始日と必要な要素数によっての呼び出し

```
int  CopyHigh(
   string           symbol_name,      // 銘柄名
   ENUM_TIMEFRAMES  timeframe,        // 期間
   datetime         start_time,       // 開始日と時刻
   int              count,            // 複製するデータ数
   double           high_array[]      // 受け取り側の配列
   );
```

必要な時間間隔の開始日と終了日によっての呼び出し

```
int  CopyHigh(
   string           symbol_name,      // 銘柄名
```

© 2000-2025, MetaQuotes Ltd.

```
   ENUM_TIMEFRAMES   timeframe,        // 期間
   datetime          start_time,       // 開始日と時刻
   datetime          stop_time,        // 終了日と時刻
   double            high_array[]      // 受け取り側の配列
   );
```

パラメータ

symbol_name
[in] 銘柄名

timeframe
[in] 期間

start_pos
[in] 1番目の複製される要素の開始位置

count
[in] 複製する日数

start_time
[in] 1番目の複製される要素の開始時刻

stop_time
[in] 複製される最後の要素に対応するバーの時刻

high_array[]
[out] double 型の配列

戻り値

複製されたデータ数（エラーの場合は -1 ）

注意事項

リクエストされたデータがサーバ上で利用可能なデータに含まれていない場合、この関数は -1 を返します。TERMINAL_MAXBARS（チャートでのバーの最大数）以外のデータがリクエストされた場合も、この関数は -1を返します。

指標からデータがリクエストされて、リクエストされた時系列がまだ構築されていないかサーバからダウンロードされる必要がある場合、この関数はすぐに -1 を返し、ダウンロード/構築のプロセスが開始されます。

エキスパートアドバイザーやスクリプトからデータがリクエストされて、端末が必要なデータをローカルに持っていない場合サーバからのダウンロードが開始されます。データがローカル履歴から構築することが出来るがまだされていない場合、必要な時系列の構築が開始します。この関数は、タイムアウトの有効期限までに準備されるデータの量を返します。が、履歴ダウンロードは続き、次に似たようなリクエストがされた際に、もっとデータを返します。

開始日及び必要な要素の数によってデータがリクエストされる場合、指定された日付とそれ以前のデータのみが戻されます。これは、値（指標バッファのボリューム、スプレッド、値、始値、高値、安値、終値、開始時刻、決済時刻）が返されたバーの開始時刻は常に指定されたもの以下だということです。

日付の指定された範囲のデータを要求する場合、この区間からのデータのみが返されます。間隔が設定され秒まで数えられます。これは、値（指標バッファのボリューム、スプレッド、値、始値、高値、安値、終値、開始時刻、決済時刻）が返されたバーの開始時刻は常に指定された区間内にあるということです。

© 2000-2025, MetaQuotes Ltd.

今日が土曜日で一週間の時間軸で *start_time=Last_Tuesday* 及び *stop_time=Last_Friday* を指定してデータを複製する試みに、この関数は 0 を返します。一週間の時間軸が開くのは常に日曜日ですが、一週間バーは指定された間隔にはまらないからです。

現在の未完成のバーに対応する値を返す必要がある場合 *start_pos*=0 及び *count*=1 を指定して1 番目の呼び出しの方法を用います。

例:

```
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "An example for output of the High[i] and Low[i]"
#property description "for a random chosen bars"

double High[],Low[];
//+------------------------------------------------------------------+
//|  指定されたバーインデックスの安値を取得する                                  |
//+------------------------------------------------------------------+
double iLow(string symbol,ENUM_TIMEFRAMES timeframe,int index)
  {
   double low=0;
   ArraySetAsSeries(Low,true);
   int copied=CopyLow(symbol,timeframe,0,Bars(symbol,timeframe),Low);
   if(copied>0 && index<copied) low=Low[index];
   return(low);
  }
//+------------------------------------------------------------------+
//|  指定されたバーインデックスの高値を取得する                                  |
//+------------------------------------------------------------------+
double iHigh(string symbol,ENUM_TIMEFRAMES timeframe,int index)
  {
   double high=0;
   ArraySetAsSeries(High,true);
   int copied=CopyHigh(symbol,timeframe,0,Bars(symbol,timeframe),High);
   if(copied>0 && index<copied) high=High[index];
   return(high);
  }
//+------------------------------------------------------------------+
//|  エキスパートティック関数                                           |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- ティックごとにティックが到着した秒に等しいバーインデックスの
//--- 高値と安値を出力する
   datetime t=TimeCurrent();
   int sec=t%60;
   printf("High[%d] = %G  Low[%d] = %G",
          sec,iHigh(Symbol(),0,sec),
```

© 2000–2025, MetaQuotes Ltd.

```
            sec,iLow(Symbol(),0,sec));
  }
```

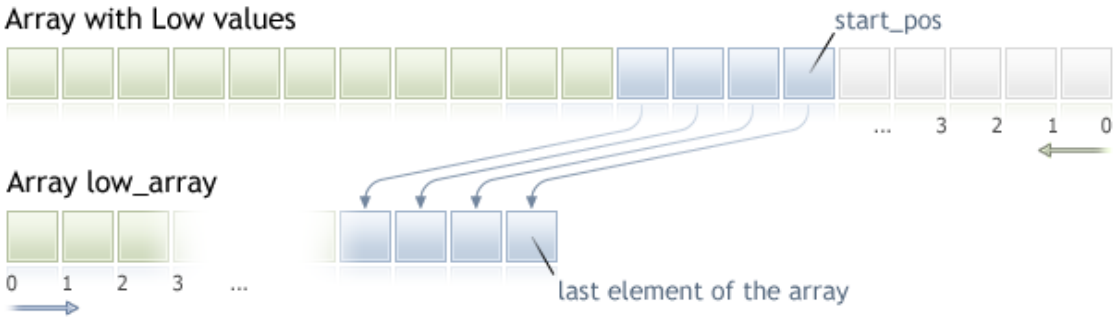
履歴データリクエストの詳細な例はオブジェクト結合の方法で参照出来ます。そのセクションで使用可能なスクリプトは iFractals 指標の最後の 1,000 足の値を取得して最後の各 10 の上向きと下向きのフラクタルをチャートに表示する方法を示しています。データ欠落時には、同様の技術が通常次のスタイルを使用して描画されている指標全てで使用することが出来ます。

- DRAW_SECTION
- DRAW_ARROW
- DRAW_ZIGZAG
- DRAW_COLOR_SECTION
- DRAW_COLOR_ARROW
- DRAW_COLOR_ZIGZAG

© 2000-2025, MetaQuotes Ltd.

CopyLow

この関数は選択された銘柄/期間ペアの足の安値の指定された量の履歴データをlow_array に配置します。要素が現在から過去の順に並べられており、インデックス0が現在足であることにご注意下さい。



リクエストされたデータ数が受け取り側の配列の長さよりも小さい(または大きい)と、この関数はリクエストされたデータが完全に収まるようにメモリを再割り当てしようとするので、複製するデータ量が不明の場合、受け取り側の配列に動的配列を使用することが推奨されます。

複製する必要があるデータの量がわかっている場合は、過剰なメモリの割り当てを防止するために静的に割り当てられたバッファを受け取り側の配列として使用するべきです。

受け取り側の配列が as_series=true であるか as_series=false であるかは関係ありません。データは、一番古い要素が配列に割り当てられた物理メモリの先頭に配置されるように複製されます。この関数の呼び出し方は 3 つあります。

1 番目の位置と必要な要素数によっての呼び出し

```
int  CopyLow(
   string            symbol_name,      // 銘柄名
   ENUM_TIMEFRAMES   timeframe,        // 期間
   int               start_pos,        // 開始位置
   int               count,            // 複製するデータ数
   double            low_array[]       // 受け取り側の配列
   );
```

開始日と必要な要素数によっての呼び出し

```
int  CopyLow(
   string            symbol_name,      // 銘柄名
   ENUM_TIMEFRAMES   timeframe,        // 期間
   datetime          start_time,       // 開始日付と時刻
   int               count,            // 複製するデータ数
   double            low_array[]       // 受け取り側の配列
   );
```

必要な時間間隔の開始日と終了日によっての呼び出し

```
int  CopyLow(
   string            symbol_name,      // 銘柄名
```

© 2000-2025, MetaQuotes Ltd.

```
   ENUM_TIMEFRAMES   timeframe,        // 期間
   datetime          start_time,       // 開始日付と時刻
   datetime          stop_time,        // 終了日と時刻
   double            low_array[]       // 受け取り側の配列
   );
```

パラメータ

symbol_name

[in] シンボル

timeframe

[in] 期間

start_pos

[in] 1番目の複製される要素の開始位置

count

[in] 複製する日数

start_time

[in] 複製される最初の要素に対応するバーの時刻

stop_time

[in] 複製される最後の要素に対応するバーの時刻

low_array[]

[out] double 型の配列

戻り値

複製されたデータ数（エラーの場合は -1 ）

注意事項

リクエストされたデータがサーバ上で利用可能なデータに含まれていない場合、この関数は -1 を返します。TERMINAL_MAXBARS（チャートでのバーの最大数）以外のデータがリクエストされた場合も、この関数は -1を返します。

指標からデータがリクエストされて、リクエストされた時系列がまだ構築されていないかサーバからダウンロードされる必要がある場合、この関数はすぐに -1 を返し、ダウンロード/構築のプロセスが開始されます。

エキスパートアドバイザーやスクリプトからデータがリクエストされて、端末が必要なデータをローカルに持っていない場合サーバからのダウンロードが開始されます。データがローカル履歴から構築することが出来るがまだされていない場合、必要な時系列の構築が開始します。この関数は、タイムアウトの有効期限までに準備されるデータの量を返します。が、履歴ダウンロードは続き、次に似たようなリクエストがされた際に、もっとデータを返します。

開始日及び必要な要素の数によってデータがリクエストされる場合、指定された日付とそれ以前のデータのみが戻されます。これは、値（指標バッファのボリューム、スプレッド、値、始値、高値、安値、終値、開始時刻、決済時刻）が返されたバーの開始時刻は常に指定されたもの以下だということです。

日付の指定された範囲のデータを要求する場合、この区間からのデータのみが返されます。間隔が設定され秒まで数えられます。これは、値（指標バッファのボリューム、スプレッド、値、始値、高値、安値、終値、開始時刻、決済時刻）が返されたバーの開始時刻は常に指定された区間内にあるということです。

© 2000-2025, MetaQuotes Ltd.

今日が土曜日で一週間の時間軸で *start_time=Last_Tuesday* 及び *stop_time=Last_Friday* を指定してデータを複製する試みに、この関数は 0 を返します。一週間の時間軸が開くのは常に日曜日ですが、一週間バーは指定された間隔にはまらないからです。

現在の未完成のバーに対応する値を返す必要がある場合 *start_pos*=0 及び *count*=1 を指定して1 番目の呼び出しの方法を用います。

履歴データリクエストの詳細な例はオブジェクト結合の方法で参照出来ます。そのセクションで使用可能なスクリプトは iFractals 指標の最後の 1,000 足の値を取得して最後の各 10 の上向きと下向きのフラクタルをチャートに表示する方法を示しています。データ欠落時には、同様の技術が通常次のスタイルを使用して描画されている指標全てで使用することが出来ます。

- DRAW_SECTION
- DRAW_ARROW
- DRAW_ZIGZAG
- DRAW_COLOR_SECTION
- DRAW_COLOR_ARROW
- DRAW_COLOR_ZIGZAG

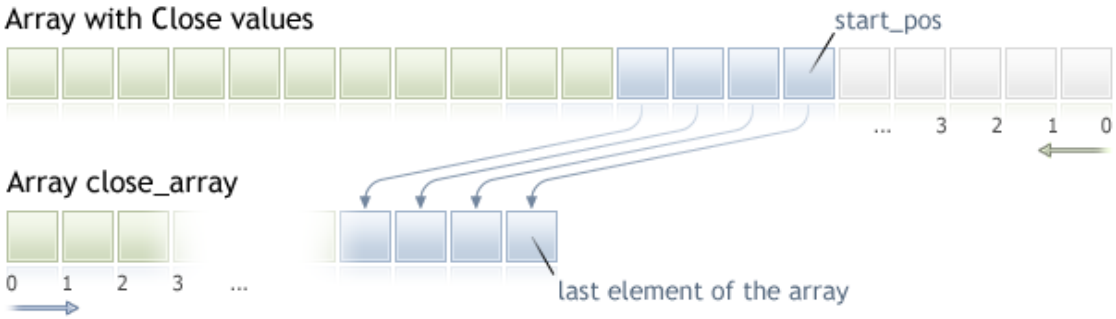
参照

CopyHigh

© 2000–2025, MetaQuotes Ltd.

CopyClose

この関数は選択されたシンボル期間のペアのバーの終値の指定された量の履歴データをclose_array　に配置します。要素が現在から過去の順に並べられており、インデックス0が現在足であることにご注意下さい。



リクエストされたデータ数が受け取り側の配列の長さよりも小さい（または大きい）と、この関数はリクエストされたデータが完全に収まるようにメモリを再割り当てしようとするので、複製するデータ量が不明の場合、受け取り側の配列に動的配列を使用することが推奨されます。

複製する必要があるデータの量がわかっている場合は、過剰なメモリの割り当てを防止するために静的に割り当てられたバッファを受け取り側の配列として使用するべきです。

受け取り側の配列がas_series=true であるかas_series=false であるかは関係ありません。データは、一番古い要素が配列に割り当てられた物理メモリの先頭に配置されるように複製されます。この関数の呼び出し方は 3 つあります。

1 番目の位置と必要な要素数によっての呼び出し

```
int  CopyClose(
   string           symbol_name,       // 銘柄名
   ENUM_TIMEFRAMES  timeframe,         // 期間
   int              start_pos,         // 開始位置
   int              count,             // 複製するデータ数
   double           close_array[]      // 受け取り側の配列
   );
```

開始日と必要な要素数によっての呼び出し

```
int  CopyClose(
   string           symbol_name,       // 銘柄名
   ENUM_TIMEFRAMES  timeframe,         // 期間
   datetime         start_time,        // 開始日と時刻
   int              count,             // 複製するデータ数
   double           close_array[]      // 受け取り側の配列
   );
```

必要な時間間隔の開始日と終了日によっての呼び出し

```
int  CopyClose(
   string           symbol_name,       // 銘柄名
```

© 2000–2025, MetaQuotes Ltd.

```
   ENUM_TIMEFRAMES   timeframe,          // 期間
   datetime          start_time,         // 開始日と時刻
   datetime          stop_time,          // 終了日と時刻
   double            close_array[]       // 受け取り側の配列
   );
```

パラメータ

symbol_name

[in] 銘柄名

timeframe

[in] 期間

start_pos

[in] 1番目の複製される要素の開始位置

count

[in] 複製する日数

start_time

[in] 1番目の複製される要素の開始時刻

stop_time

[in] 複製される最後の要素に対応するバーの時刻

close_array[]

[out] double 型の配列

戻り値

複製されたデータ数（エラーの場合は -1 ）

注意事項

リクエストされたデータがサーバ上で利用可能なデータに含まれていない場合、この関数は -1 を返します。TERMINAL_MAXBARS（チャートでのバーの最大数）以外のデータがリクエストされた場合も、この関数は -1を返します。

指標からデータがリクエストされて、リクエストされた時系列がまだ構築されていないかサーバからダウンロードされる必要がある場合、この関数はすぐに -1 を返し、ダウンロード/構築のプロセスが開始されます。

エキスパートアドバイザーやスクリプトからデータがリクエストされて、端末が必要なデータをローカルに持っていない場合サーバからのダウンロードが開始されます。データがローカル履歴から構築することが出来るがまだされていない場合、必要な時系列の構築が開始します。この関数は、タイムアウトの有効期限までに準備されるデータの量を返します。が、履歴ダウンロードは続き、次に似たようなリクエストがされた際に、もっとデータを返します。

開始日及び必要な要素の数によってデータがリクエストされる場合、指定された日付とそれ以前のデータのみが戻されます。これは、値（指標バッファのボリューム、スプレッド、値、始値、高値、安値、終値、開始時刻、決済時刻）が返されたバーの開始時刻は常に指定されたもの以下だということです。

日付の指定された範囲のデータを要求する場合、この区間からのデータのみが返されます。間隔が設定され秒まで数えられます。これは、値（指標バッファのボリューム、スプレッド、値、始値、高値、安値、終値、開始時刻、決済時刻）が返されたバーの開始時刻は常に指定された区間内にあるということです。

© 2000-2025, MetaQuotes Ltd.

今日が土曜日で一週間の時間軸で *start_time=Last_Tuesday* 及び *stop_time=Last_Friday* を指定してデータを複製する試みに、この関数は 0 を返します。一週間の時間軸が開くのは常に日曜日ですが、一週間バーは指定された間隔にはまらないからです。

現在の未完成のバーに対応する値を返す必要がある場合 *start_pos*=0 及び *count*=1 を指定して1 番目の呼び出しの方法を用います。

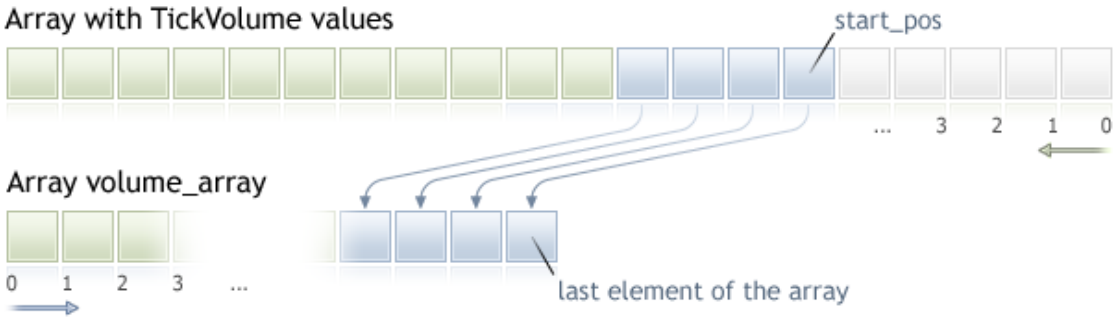
履歴データリクエストの詳細な例はオブジェクト結合の方法で参照出来ます。そのセクションで使用可能なスクリプトは iFractals 指標の最後の 1,000 足の値を取得して最後の各 10 の上向きと下向きのフラクタルをチャートに表示する方法を示しています。データ欠落時には、同様の技術が通常次のスタイルを使用して描画されている指標全てで使用することが出来ます。

- DRAW_SECTION
- DRAW_ARROW
- DRAW_ZIGZAG
- DRAW_COLOR_SECTION
- DRAW_COLOR_ARROW
- DRAW_COLOR_ZIGZAG

© 2000–2025, MetaQuotes Ltd.

CopyTickVolume

この関数は選択された銘柄/期間ペアのバーのティックボリュームの指定された量の履歴データをvolume_arrayy に配置します。要素が現在から過去の順に並べられており、インデックス0が現在足であることにご注意下さい。



リクエストされたデータ数が受け取り側の配列の長さよりも小さい(または大きい)と、この関数はリクエストされたデータが完全に収まるようにメモリを再割り当てしようとするので、複製するデータ量が不明の場合、受け取り側の配列に動的配列を使用することが推奨されます。

複製する必要があるデータの量がわかっている場合は、過剰なメモリの割り当てを防止するために静的に割り当てられたバッファを受け取り側の配列として使用するべきです。

受け取り側の配列がas_series=true であるかas_series=false であるかは関係ありません。データは、一番古い要素が配列に割り当てられた物理メモリの先頭に配置されるように複製されます。この関数の呼び出し方は3つあります。

1番目の位置と必要な要素数によっての呼び出し

```
int  CopyTickVolume(
   string           symbol_name,      // 銘柄名
   ENUM_TIMEFRAMES  timeframe,        // 期間
   int              start_pos,        // 開始位置
   int              count,            // 複製するデータ数
   long             volume_array[]    // ティックボリュームを受け取る配列
   );
```

開始日と必要な要素数によっての呼び出し

```
int  CopyTickVolume(
   string           symbol_name,      // 銘柄名
   ENUM_TIMEFRAMES  timeframe,        // 期間
   datetime         start_time,       // 開始日と時刻
   int              count,            // 複製するデータ数
   long             volume_array[]    // ティックボリュームを受け取る配列
   );
```

必要な時間間隔の開始日と終了日によっての呼び出し

```
int  CopyTickVolume(
```

© 2000-2025, MetaQuotes Ltd.

```
   string            symbol_name,      // 銘柄名
   ENUM_TIMEFRAMES   timeframe,        // 期間
   datetime          start_time,       // 開始日と時刻
   datetime          stop_time,        // 終了日と時刻
   long              volume_array[]    // ティックボリュームを受け取る配列
   );
```

パラメータ

symbol_name

[in] 銘柄名

timeframe

[in] 期間

start_pos

[in] 1番目の複製される要素の開始位置

count

[in] 複製する日数

start_time

[in] 1番目の複製される要素の開始時刻

stop_time

[in] 複製される最後の要素に対応するバーの時刻

volume_array[]

[out] long 型の配列

戻り値

複製されたデータ数（エラーの場合は -1 ）

注意事項

リクエストされたデータがサーバ上で利用可能なデータに含まれていない場合、この関数は -1 を返します。TERMINAL_MAXBARS（チャートでのバーの最大数）以外のデータがリクエストされた場合も、この関数は -1を返します。

指標からデータがリクエストされて、リクエストされた時系列がまだ構築されていないかサーバからダウンロードされる必要がある場合、この関数はすぐに -1 を返し、ダウンロード/構築のプロセスが開始されます。

エキスパートアドバイザーやスクリプトからデータがリクエストされて、端末が必要なデータをローカルに持っていない場合サーバからのダウンロードが開始されます。データがローカル履歴から構築することが出来るがまだされていない場合、必要な時系列の構築が開始します。この関数は、タイムアウトの有効期限までに準備されるデータの量を返します。が、履歴ダウンロードは続き、次に似たようなリクエストがされた際に、もっとデータを返します。

開始日及び必要な要素の数によってデータがリクエストされる場合、指定された日付とそれ以前のデータのみが戻されます。これは、値（指標バッファのボリューム、スプレッド、値、始値、高値、安値、終値、開始時刻、決済時刻）が返されたバーの開始時刻は常に指定されたもの以下だということです。

© 2000-2025, MetaQuotes Ltd.

日付の指定された範囲のデータを要求する場合、この区間からのデータのみが返されます。間隔が設定され秒まで数えられます。これは、値（指標バッファのボリューム、スプレッド、値、始値、高値、安値、終値、開始時刻、決済時刻）が返されたバーの開始時刻は常に指定された区間内にあるということです。

今日が土曜日で一週間の時間軸で *start_time=Last_Tuesday* 及び *stop_time=Last_Friday* を指定してデータを複製する試みに、この関数は 0 を返します。一週間の時間軸が開くのは常に日曜日ですが、一週間バーは指定された間隔にはまらないからです。

現在の未完成のバーに対応する値を返す必要がある場合 *start_pos*=0 及び *count*=1 を指定して1 番目の呼び出しの方法を用います。

例：

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- TickVolume をプロットする
#property indicator_label1  "TickVolume"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  C'143,188,139'
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 入力パラメータ
input int      bars=3000;
//--- 指標バッファ
double         TickVolumeBuffer[];
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- 指標バッファマッピング
   SetIndexBuffer(0,TickVolumeBuffer,INDICATOR_DATA);
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
  }
//+------------------------------------------------------------------+
//|  カスタム指標の反復関数                                          |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
```

© 2000–2025, MetaQuotes Ltd.

```
//---
   if(prev_calculated==0)
     {
      long timeseries[];
      ArraySetAsSeries(timeseries,true);
      int prices=CopyTickVolume(Symbol(),0,0,bars,timeseries);
      for(int i=0;i<rates_total-prices;i++) TickVolumeBuffer[i]=0.0;
      for(int i=0;i<prices;i++) TickVolumeBuffer[rates_total-1-i]=timeseries[prices-
1-i];
      Print("We have received the following number of TickVolume values: "+prices);
     }
   else
     {
      long timeseries[];
      int prices=CopyTickVolume(Symbol(),0,0,1,timeseries);
      TickVolumeBuffer[rates_total-1]=timeseries[0];
     }
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
```

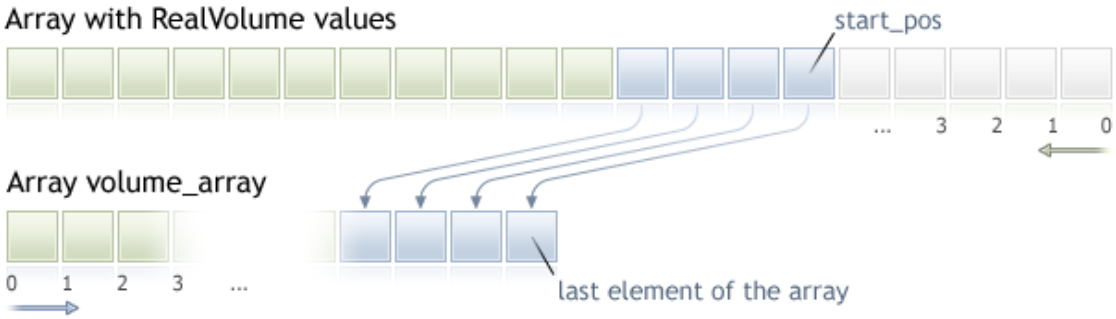
履歴データリクエストの詳細な例はオブジェクト結合の方法で参照出来ます。そのセクションで使用可能なスクリプトは iFractals 指標の最後の 1,000 足の値を取得して最後の各 10 の上向きと下向きのフラクタルをチャートに表示する方法を示しています。データ欠落時には、同様の技術が通常次のスタイルを使用して描画されている指標全てで使用することが出来ます。

- DRAW_SECTION
- DRAW_ARROW
- DRAW_ZIGZAG
- DRAW_COLOR_SECTION
- DRAW_COLOR_ARROW
- DRAW_COLOR_ZIGZAG

© 2000-2025, MetaQuotes Ltd.

CopyRealVolume

この関数は選択された銘柄/期間ペアのバーの取引高の指定された量の履歴データをvolume_arrayy に配置します。要素が現在から過去の順に並べられており、インデックス0が現在足であることにご注意下さい。



リクエストされたデータ数が受け取り側の配列の長さよりも小さい(または大きい)と、この関数はリクエストされたデータが完全に収まるようにメモリを再割り当てしようとするので、複製するデータ量が不明の場合、受け取り側の配列に動的配列を使用することが推奨されます。

複製する必要があるデータの量がわかっている場合は、過剰なメモリの割り当てを防止するために静的に割り当てられたバッファを受け取り側の配列として使用するべきです。

受け取り側の配列が as_series=true であるか as_series=false であるかは関係ありません。データは、一番古い要素が配列に割り当てられた物理メモリの先頭に配置されるように複製されます。この関数の呼び出し方は 3 つあります。

1 番目の位置と必要な要素数によっての呼び出し

```
int  CopyRealVolume(
   string           symbol_name,      // 銘柄名
   ENUM_TIMEFRAMES  timeframe,        // 期間
   int              start_pos,        // 開始位置
   int              count,            // 複製するデータ数
   long             volume_array[]    // ボリューム値を受け取る配列
   );
```

開始日と必要な要素数によっての呼び出し

```
int  CopyRealVolume(
   string           symbol_name,      // 銘柄名
   ENUM_TIMEFRAMES  timeframe,        // 期間
   datetime         start_time,       // 開始日と時刻
   int              count,            // 複製するデータ数
   long             volume_array[]    // ボリューム値を受け取る配列
   );
```

必要な時間間隔の開始日と終了日によっての呼び出し

```
int  CopyRealVolume(
```

© 2000-2025, MetaQuotes Ltd.

```
   string            symbol_name,      // 銘柄名
   ENUM_TIMEFRAMES   timeframe,        // 期間
   datetime          start_time,       // 開始日と時刻
   datetime          stop_time,        // 終了日と時刻
   long              volume_array[]    // ボリューム値を受け取る配列
   );
```

パラメータ

symbol_name

[in] 銘柄名

timeframe

[in] 期間

start_pos

[in] 1番目の複製される要素の開始位置

count

[in] 複製する日数

start_time

[in] 1番目の複製される要素の開始時刻

stop_time

[in] 複製される最後の要素に対応するバーの時刻

volume_array[]

[out] long 型の配列

戻り値

複製されたデータ数（エラーの場合は -1 ）

注意事項

リクエストされたデータがサーバ上で利用可能なデータに含まれていない場合、この関数は -1 を返します。TERMINAL_MAXBARS（チャートでのバーの最大数）以外のデータがリクエストされた場合も、この関数は -1を返します。

指標からデータがリクエストされて、リクエストされた時系列がまだ構築されていないかサーバからダウンロードされる必要がある場合、この関数はすぐに -1 を返し、ダウンロード/構築のプロセスが開始されます。

エキスパートアドバイザーやスクリプトからデータがリクエストされて、端末が必要なデータをローカルに持っていない場合サーバからのダウンロードが開始されます。データがローカル履歴から構築することが出来るがまだされていない場合、必要な時系列の構築が開始します。この関数は、タイムアウトの有効期限までに準備されるデータの量を返します。が、履歴ダウンロードは続き、次に似たようなリクエストがされた際に、もっとデータを返します。

開始日及び必要な要素の数によってデータがリクエストされる場合、指定された日付とそれ以前のデータのみが戻されます。これは、値（指標バッファのボリューム、スプレッド、値、始値、高値、安値、終値、開始時刻、決済時刻）が返されたバーの開始時刻は常に指定されたもの以下だということです。

© 2000-2025, MetaQuotes Ltd.

日付の指定された範囲のデータを要求する場合、この区間からのデータのみが返されます。間隔が設定され秒まで数えられます。これは、値（指標バッファのボリューム、スプレッド、値、始値、高値、安値、終値、開始時刻、決済時刻）が返されたバーの開始時刻は常に指定された区間内にあるということです。

今日が土曜日で一週間の時間軸で *start_time=Last_Tuesday* 及び *stop_time=Last_Friday* を指定してデータを複製する試みに、この関数は 0 を返します。一週間の時間軸が開くのは常に日曜日ですが、一週間バーは指定された間隔にはまらないからです。

現在の未完成のバーに対応する値を返す必要がある場合 *start_pos*=0 及び *count*=1 を指定して1 番目の呼び出しの方法を用います。

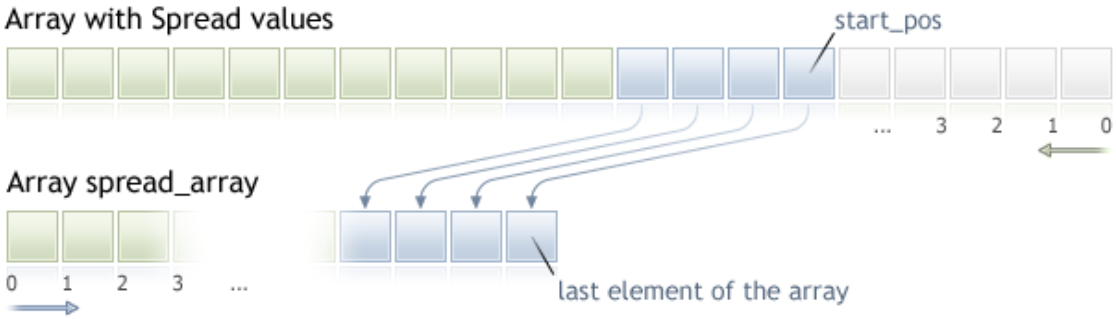
履歴データリクエストの詳細な例はオブジェクト結合の方法で参照出来ます。そのセクションで使用可能なスクリプトは iFractals 指標の最後の 1,000 足の値を取得して最後の各 10 の上向きと下向きのフラクタルをチャートに表示する方法を示しています。データ欠落時には、同様の技術が通常次のスタイルを使用して描画されている指標全てで使用することが出来ます。

- DRAW_SECTION
- DRAW_ARROW
- DRAW_ZIGZAG
- DRAW_COLOR_SECTION
- DRAW_COLOR_ARROW
- DRAW_COLOR_ZIGZAG

© 2000–2025, MetaQuotes Ltd.

CopySpread

この関数は選択された銘柄/期間ペアのバーのスプレッド値の指定された量の履歴データをspread_arrayに配置します。要素が現在から過去の順に並べられており、インデックス0が現在足であることにご注意下さい。



リクエストされたデータ数が受け取り側の配列の長さよりも小さい(または大きい)と、この関数はリクエストされたデータが完全に収まるようにメモリを再割り当てしようとするので、複製するデータ量が不明の場合、受け取り側の配列に動的配列を使用することが推奨されます。

複製する必要があるデータの量がわかっている場合は、過剰なメモリの割り当てを防止するために静的に割り当てられたバッファを受け取り側の配列として使用するべきです。

受け取り側の配列がas_series=trueであるかas_series=falseであるかは関係ありません。データは、一番古い要素が配列に割り当てられた物理メモリの先頭に配置されるように複製されます。この関数の呼び出し方は3つあります。

1番目の位置と必要な要素数によっての呼び出し

```
int  CopySpread(
   string            symbol_name,      // 銘柄名
   ENUM_TIMEFRAMES   timeframe,        // 期間
   int               start_pos,        // 開始位置
   int               count,            // 複製するデータ数
   int               spread_array[]    // スプレッド値を受け取る配列
   );
```

開始日と必要な要素数によっての呼び出し

```
int  CopySpread(
   string            symbol_name,      // 銘柄名
   ENUM_TIMEFRAMES   timeframe,        // 期間
   datetime          start_time,       // 開始日と時刻
   int               count,            // 複製するデータ数
   int               spread_array[]    // スプレッド値を受け取る配列
   );
```

必要な時間間隔の開始日と終了日によっての呼び出し

```
int  CopySpread(
```

© 2000-2025, MetaQuotes Ltd.

```
   string            symbol_name,      // 銘柄名
   ENUM_TIMEFRAMES   timeframe,        // 期間
   datetime          start_time,       // 開始日と時刻
   datetime          stop_time,        // 終了日と時刻
   int               spread_array[]    // スプレッド値を受け取る配列
   );
```

パラメータ

symbol_name

[in] 銘柄名

timeframe

[in] 期間

start_pos

[in] 1番目の複製される要素の開始位置

count

[in] 複製する日数

start_time

[in] 1番目の複製される要素の開始時刻

stop_time

[in] 複製される最後の要素に対応するバーの時刻

spread_array[]

[out] int型の配列

戻り値

複製されたデータ数（エラーの場合は -1 ）

注意事項

リクエストされたデータがサーバ上で利用可能なデータに含まれていない場合、この関数は -1 を返します。TERMINAL_MAXBARS（チャートでのバーの最大数）以外のデータがリクエストされた場合も、この関数は -1を返します。

指標からデータがリクエストされて、リクエストされた時系列がまだ構築されていないかサーバからダウンロードされる必要がある場合、この関数はすぐに -1 を返し、ダウンロード/構築のプロセスが開始されます。

エキスパートアドバイザーやスクリプトからデータがリクエストされて、端末が必要なデータをローカルに持っていない場合サーバからのダウンロードが開始されます。データがローカル履歴から構築することが出来るがまだされていない場合、必要な時系列の構築が開始します。この関数は、タイムアウトの有効期限までに準備されるデータの量を返します。が、履歴ダウンロードは続き、次に似たようなリクエストがされた際に、もっとデータを返します。

開始日及び必要な要素の数によってデータがリクエストされる場合、指定された日付とそれ以前のデータのみが戻されます。これは、値（指標バッファのボリューム、スプレッド、値、始値、高値、安値、終値、開始時刻、決済時刻）が返されたバーの開始時刻は常に指定されたもの以下だということです。

© 2000-2025, MetaQuotes Ltd.

日付の指定された範囲のデータを要求する場合、この区間からのデータのみが返されます。間隔が設定され秒まで数えられます。これは、値（指標バッファのボリューム、スプレッド、値、始値、高値、安値、終値、開始時刻、決済時刻）が返されたバーの開始時刻は常に指定された区間内にあるということです。

今日が土曜日で一週間の時間軸で *start_time=Last_Tuesday* 及び *stop_time=Last_Friday* を指定してデータを複製する試みに、この関数は 0 を返します。一週間の時間軸が開くのは常に日曜日ですが、一週間バーは指定された間隔にはまらないからです。

現在の未完成のバーに対応する値を返す必要がある場合 *start_pos*=0 及び *count*=1 を指定して1 番目の呼び出しの方法を用います。

例：

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- スプレッドをプロットする
#property indicator_label1  "Spread"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 入力パラメータ
input int      bars=3000;
//--- 指標バッファ
double         SpreadBuffer[];
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- 指標バッファマッピング
   SetIndexBuffer(0,SpreadBuffer,INDICATOR_DATA);
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
```

© 2000–2025, MetaQuotes Ltd.

```
//---
   if(prev_calculated==0)
     {
      int spread_int[];
      ArraySetAsSeries(spread_int,true);
      int spreads=CopySpread(Symbol(),0,0,bars,spread_int);
      Print("We have received the following number of Spread values: ",spreads);
      for (int i=0;i<spreads;i++)
      {
      SpreadBuffer[rates_total-1-i]=spread_int[i];
      if(i<=30) Print("spread["+i+"] = ",spread_int[i]);
      }
     }
   else
     {
      double Ask,Bid;
      Ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
      Bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
      Comment("Ask = ",Ask,"  Bid = ",Bid);
      SpreadBuffer[rates_total-1]=(Ask-Bid)/Point();
     }
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
```

履歴データリクエストの詳細な例はオブジェクト結合の方法で参照出来ます。そのセクションで使用可能なスクリプトは iFractals 指標の最後の 1,000 足の値を取得して最後の各 10 の上向きと下向きのフラクタルをチャートに表示する方法を示しています。データ欠落時には、同様の技術が通常次のスタイルを使用して描画されている指標全てで使用することが出来ます。

- DRAW_SECTION
- DRAW_ARROW
- DRAW_ZIGZAG
- DRAW_COLOR_SECTION
- DRAW_COLOR_ARROW
- DRAW_COLOR_ZIGZAG

© 2000-2025, MetaQuotes Ltd.

CopyTicks

この関数はティックをMqlTick形式でticks_arrayに受け取ります。この場合、ティックの索引付けは過去から現在に向かうため、0インデックスを持ったティックは配列で一番古いものとなります。ティックの分析には*flags*フィールドをチェックします。このフィールドはティックで何が変わったのかを正確に示します。

```
int  CopyTicks(
   string            symbol_name,           // 銘柄名
   MqlTick&          ticks_array[],         // ティックを受け取る配列
   uint              flags=COPY_TICKS_ALL,  // 受け取られたティックの種類を定義するフラグ
   ulong             from=0,                // ティックがリクエストされた日
   uint              count=0                // 受け取りたいティックの数
   );
```

パラメータ

symbol_name

[in] 銘柄。

ticks_array

[out] ティックを受け取るMqlTick型の配列。

フラグ

[in] 受け取るティックの種類を定義するフラグ。COPY_TICKS_INFO – 買値及び/または売値の変更を持つティック、COPY_TICKS_TRADE – ラストとボリュームの変更を持つティック、COPY_TICKS_ALL – すべてのティック。リクエストのタイプにかかわらず、以前のティックの値はMqlTick構造体の残りのフィールドに追加されます。

from

[in] ティックをリクエストしたい日。1970.01.01から経過したミリ秒。*from*=0の場合は直近の*count*ティックが戻されます。

count

[in] リクエストされたティックの数。「from」及び「count」パラメータが指定されない場合は全てのティック（2000個以下）がticks_array[]に書き込まれます。

戻り値

複製されたティック数（エラーの場合は -1）。

更なる注意事項

CopyTicks() 関数は受け取られた全てのティックのリクエストと分析を可能にします。CopyTicks()の初めの呼び出しによって、ハードディスクに保存されている銘柄のティックデータベースの同期が開始されます。リクエストされたティックのすべてがローカルデータベースに存在しない場合は、不足しているティックは自動的に取引サーバからダウンロードされます。CopyTicks() で指定された***from***の日付から現時点のティックがすべて同期されます。その後、この銘柄に到着するすべてのティックはデータベースに追加され、同期された状態が保たれます。

*from*及び*count*パラメータが指定されない場合は、全てのティック（2000個以下）が*ticks_array[]*に書き込まれます。*flags*パラメータは、リクエストされたティックの種類の設定を可能にします。

COPY_TICKS_INFO – 買値及び/または売値の変更を持つティックが戻されます。他のフィールドのデータもまた追加されます。例えば、買値のみが変わった場合 *売値とボリューム*フィールドは最後に知られている

© 2000–2025, MetaQuotes Ltd.

値で書き込まれます。何が変更されたかを正確に調べるには*flags*フィールドを分析します。このフィールドはTICK_FLAG_BID及び/または TICK_FLAG_ASK値を持ちます。ティックの買値と売値がゼロでフラグがこれらのデータの変更を示す場合は(*flags*=TICK_FLAG_BID| TICK_FLAG_ASK)、オーダーブック(板情報)が空です。つまり、買い注文も売り注文も存在しないということです。

COPY_TICKS_TRADE - ラスト価格とボリュームの変更を持つティックが戻されます。他のフィールドのデータも追加されます。すなわち買値と売値の最後に知られている値がそれぞれのフィールドで指定されます。何が変更されたかを正確に調べるには*flags*フィールドを分析します。このフィールドはTICK_FLAG_LAST及びTICK_FLAG_VOLUME値を持ちます。

COPY_TICKS_ALL - 変更を持つ全てのティックが戻されます。変更されていないフィールドには最後の既知の値が入力されます。

CopyTicks()のCOPY_TICKS_ALLフラグでの呼び出しはリクエストされた間隔からのすべてのティックを戻します。一方、他のモードでの呼び出しはティックの処理と選択にいくらかの時間を要するので、速度面の利点を提供しません。

すべてのティックは、リクエストされた(COPY_TICKS_INFOまたはCOPY_TICKS_TRADE)時点での完全な価格情報を含みます(*買値*、*売値*、*ラスト価格*、及び*ボリューム*)。この機能は、各ティックの時点での取引状態を簡単に分析するために設けられているので、長いティック履歴をリクエストして他のフィールドの値を検索する必要はありません。

インディケータではCopyTicks()関数は結果を戻します:インディケータから呼び出されると、CopyTick()は即時、銘柄のティックをすべて戻し、使えるデータが十分でない場合にはティックデータベースを同期します。1銘柄のすべてのインディケータは1つの共通のスレッドで動作するので、インディケータは同期の完了を待つことはできません。CopyTicks() は同期後に次に呼び出された時に、リクエストされたティックをすべて返します。インディケータではOnCalculate()関数は各ティックの到着後に呼び出されます。

エキスパートアドバイザー及びスクリプトでは、CopyTicks()は45秒間結果を待つことがあります。これに対して、インディケータではエキスパートアドバイザーとスクリプトはそれぞれのスレッドで実行されるので、同期が終了するまで45秒間待つことがあります。この期間中に必要なティックの量の同期が失敗した場合は、 CopyTicks() はタイムアウトして使用可能なティックを返して同期を続けます。エキスパートアドバイザーのOnTick()は1つごとのティックのハンドラではなく、市場の変更についてのみをエキスパートアドバイザーに通知します。変化は単一でないかもしれません。端末では複数のティックが同時に出現することがありますが、OnTick() は最新の市場状態をエキスパートアドバイザーに通知するために一度だけ呼び出されます。

データリターンの速度:端末の高速アクセスキャッシュには、各製品の直近の4096ティックが格納されます(板情報で実行中の銘柄の場合は65536ティック)。現在の取引セッションのためにリクエストされたティックがキャッシュに存在しない場合は、CopyTicks() は端末のメモリに保存されているティックを呼び出します。これらのリクエストは、実行のためのより多くの時間を必要とします。データがディスクから読み込まれるため、一番時間のかかるリクエストは、他の日からのティックのリクエストです。

例:

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property script_show_inputs
//--- Requesting 100 million ticks to be sure we receive the entire tick history
input int      getticks=100000000; // The number of required ticks
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//---
   int      attempts=0;      // Count of attempts
   bool     success=false;   // The flag of a successful copying of ticks
   MqlTick tick_array[];     // Tick receiving array
   MqlTick lasttick;         // To receive last tick data
   SymbolInfoTick(_Symbol,lasttick);
//--- Make 3 attempts to receive ticks
   while(attempts<3)
     {
      //--- Measuring start time before receiving the ticks
      uint start=GetTickCount();
//--- Requesting the tick history since 1970.01.01 00:00.001 (parameter from=1 ms)
      int received=CopyTicks(_Symbol,tick_array,COPY_TICKS_ALL,1,getticks);
      if(received!=-1)
        {
         //--- Showing information about the number of ticks and spent time
         PrintFormat("%s: received %d ticks in %d
ms",_Symbol,received,GetTickCount()-start);
         //--- If the tick history is synchronized, the error code is equal to zero
         if(GetLastError()==0)
           {
            success=true;
            break;
           }
         else
            PrintFormat("%s: Ticks are not synchronized yet, %d ticks received for %
d ms. Error=%d",
            _Symbol,received,GetTickCount()-start,_LastError);
        }
      //--- Counting attempts
      attempts++;
      //--- A one-second pause to wait for the end of synchronization of the tick
database
      Sleep(1000);
     }
//--- Receiving the requested ticks from the beginning of the tick history failed in
three attempts
   if(!success)
     {
      PrintFormat("Error! Failed to receive %d ticks of %s in three
attempts",getticks,_Symbol);
      return;
     }
   int ticks=ArraySize(tick_array);
//--- Showing the time of the first tick in the array
   datetime firstticktime=tick_array[ticks-1].time;
   PrintFormat("Last tick time = %s.%03I64u",
```

© 2000–2025, MetaQuotes Ltd.

```
               TimeToString(firstticktime,TIME_DATE|TIME_MINUTES|
TIME_SECONDS),tick_array[ticks-1].time_msc%1000);
//--- выведем время последнего тика в массиве
   datetime lastticktime=tick_array[0].time;
   PrintFormat("First tick time = %s.%03I64u",
               TimeToString(lastticktime,TIME_DATE|TIME_MINUTES|
TIME_SECONDS),tick_array[0].time_msc%1000);

//---
   MqlDateTime today;
   datetime current_time=TimeCurrent();
   TimeToStruct(current_time,today);
   PrintFormat("current_time=%s",TimeToString(current_time));
   today.hour=0;
   today.min=0;
   today.sec=0;
   datetime startday=StructToTime(today);
   datetime endday=startday+24*60*60;
   if((ticks=CopyTicksRange
(_Symbol,tick_array,COPY_TICKS_ALL,startday*1000,endday*1000))==-1)
     {
      PrintFormat("CopyTicksRange(%s,tick_array,COPY_TICKS_ALL,%s,%s) failed, error
%d",
                  _Symbol
,TimeToString(startday),TimeToString(endday),GetLastError());
      return;

     }
   ticks=MathMax(100,ticks);
//--- Showing the first 100 ticks of the last day
   int counter=0;
   for(int i=0;i<ticks;i++)
     {
      datetime time=tick_array[i].time;
      if((time>=startday) && (time<endday) && counter<100)
        {
         counter++;
         PrintFormat("%d. %s",counter,GetTickDescription(tick_array[i]));
        }
     }
//--- Showing the first 100 deals of the last day
   counter=0;
   for(int i=0;i<ticks;i++)
     {
      datetime time=tick_array[i].time;
      if((time>=startday) && (time<endday) && counter<100)
        {
         if(((tick_array[i].flags&TICK_FLAG_BUY)==TICK_FLAG_BUY) ||
((tick_array[i].flags&TICK_FLAG_SELL)==TICK_FLAG_SELL))
           {
```

© 2000–2025, MetaQuotes Ltd.

```
            counter++;
            PrintFormat("%d. %s",counter,GetTickDescription(tick_array[i]));
           }
        }
     }
  }
//+------------------------------------------------------------------+
//| Returns the string description of a tick                         |
//+------------------------------------------------------------------+
string GetTickDescription(MqlTick &tick)
  {
   string desc=StringFormat("%s.%03d ",
                            TimeToString(tick.time),tick.time_msc%1000);
//--- Checking flags
   bool buy_tick=((tick.flags&TICK_FLAG_BUY)==TICK_FLAG_BUY);
   bool sell_tick=((tick.flags&TICK_FLAG_SELL)==TICK_FLAG_SELL);
   bool ask_tick=((tick.flags&TICK_FLAG_ASK)==TICK_FLAG_ASK);
   bool bid_tick=((tick.flags&TICK_FLAG_BID)==TICK_FLAG_BID);
   bool last_tick=((tick.flags&TICK_FLAG_LAST)==TICK_FLAG_LAST);
   bool volume_tick=((tick.flags&TICK_FLAG_VOLUME)==TICK_FLAG_VOLUME);
//--- Checking trading flags in a tick first
   if(buy_tick || sell_tick)
     {
      //--- Forming an output for the trading tick
      desc=desc+(buy_tick?StringFormat("Buy Tick: Last=%G Volume=%d
",tick.last,tick.volume):"");
      desc=desc+(sell_tick?StringFormat("Sell Tick: Last=%G Volume=%d
",tick.last,tick.volume):"");
      desc=desc+(ask_tick?StringFormat("Ask=%G ",tick.ask):"");
      desc=desc+(bid_tick?StringFormat("Bid=%G ",tick.ask):"");
      desc=desc+"(Trade tick)";
     }
   else
     {
      //--- Form a different output for an info tick
      desc=desc+(ask_tick?StringFormat("Ask=%G ",tick.ask):"");
      desc=desc+(bid_tick?StringFormat("Bid=%G ",tick.ask):"");
      desc=desc+(last_tick?StringFormat("Last=%G ",tick.last):"");
      desc=desc+(volume_tick?StringFormat("Volume=%d ",tick.volume):"");
      desc=desc+"(Info tick)";
     }
//--- Returning tick description
   return desc;
  }
//+------------------------------------------------------------------+
/* Example of the output
Si-12.16: received 11048387 ticks in 4937 ms
Last tick time = 2016.09.26 18:32:59.775
First tick time = 2015.06.18 09:45:01.000
```

© 2000–2025, MetaQuotes Ltd.

```
1.  2016.09.26 09:45.249 Ask=65370 Bid=65370 (Info tick)
2.  2016.09.26 09:47.420 Ask=65370 Bid=65370 (Info tick)
3.  2016.09.26 09:50.893 Ask=65370 Bid=65370 (Info tick)
4.  2016.09.26 09:51.827 Ask=65370 Bid=65370 (Info tick)
5.  2016.09.26 09:53.810 Ask=65370 Bid=65370 (Info tick)
6.  2016.09.26 09:54.491 Ask=65370 Bid=65370 (Info tick)
7.  2016.09.26 09:55.913 Ask=65370 Bid=65370 (Info tick)
8.  2016.09.26 09:59.350 Ask=65370 Bid=65370 (Info tick)
9.  2016.09.26 09:59.678 Bid=65370 (Info tick)
10. 2016.09.26 10:00.000 Sell Tick: Last=65367 Volume=3 (Trade tick)
11. 2016.09.26 10:00.000 Sell Tick: Last=65335 Volume=45 (Trade tick)
12. 2016.09.26 10:00.000 Sell Tick: Last=65334 Volume=95 (Trade tick)
13. 2016.09.26 10:00.191 Sell Tick: Last=65319 Volume=1 (Trade tick)
14. 2016.09.26 10:00.191 Sell Tick: Last=65317 Volume=1 (Trade tick)
15. 2016.09.26 10:00.191 Sell Tick: Last=65316 Volume=1 (Trade tick)
16. 2016.09.26 10:00.191 Sell Tick: Last=65316 Volume=10 (Trade tick)
17. 2016.09.26 10:00.191 Sell Tick: Last=65315 Volume=5 (Trade tick)
18. 2016.09.26 10:00.191 Sell Tick: Last=65313 Volume=3 (Trade tick)
19. 2016.09.26 10:00.191 Sell Tick: Last=65307 Volume=25 (Trade tick)
20. 2016.09.26 10:00.191 Sell Tick: Last=65304 Volume=1 (Trade tick)
21. 2016.09.26 10:00.191 Sell Tick: Last=65301 Volume=1 (Trade tick)
22. 2016.09.26 10:00.191 Sell Tick: Last=65301 Volume=10 (Trade tick)
23. 2016.09.26 10:00.191 Sell Tick: Last=65300 Volume=5 (Trade tick)
24. 2016.09.26 10:00.191 Sell Tick: Last=65300 Volume=1 (Trade tick)
25. 2016.09.26 10:00.191 Sell Tick: Last=65300 Volume=6 (Trade tick)
26. 2016.09.26 10:00.191 Sell Tick: Last=65299 Volume=1 (Trade tick)
27. 2016.09.26 10:00.191 Bid=65370 (Info tick)
28. 2016.09.26 10:00.232 Ask=65297 (Info tick)
29. 2016.09.26 10:00.276 Sell Tick: Last=65291 Volume=31 (Trade tick)
30. 2016.09.26 10:00.276 Sell Tick: Last=65290 Volume=1 (Trade tick)
*/
```

参照

SymbolInfoTick、現在価格の構造体、OnTick()

© 2000-2025, MetaQuotes Ltd.

CopyTicksRange

この関数は指定された期間内のティックをMqlTick形式で ticks_arrayで受け取ります。インデックスは過去に始まって現在に終わるので、インデックス 0 を持つティックが配列で一番古いものです。ティックの分析のためには*flags*フィールドを確認します。これは何が変わったのかを正確に示します。

```
int  CopyTicksRange(
   const string      symbol_name,            // 銘柄名
   MqlTick&          ticks_array[],          // ティックを受け取る配列
   uint              flags=COPY_TICKS_ALL,   // 受け取るティックの種類を定義するフラグ
   ulong             from_msc=0,             // ティックがリクエストされた初めの日
   ulong             to_msc=0                // ティックがリクエストされた最後の日
   );
```

パラメータ

symbol_name

[in] シンボル

ticks_array

[out] ティックを受け取る MqlTick 型の静的もしくは動的配列。静的配列が要求された時間間隔のすべてのティックを保持できない場合、可能な最大量のティックが受信されます。この場合、この関数は ERR_HISTORY_SMALL_BUFFER (4407) エラーを生成します。

フラグ

[in] 要求されたティックのタイプを定義するフラグCOPY_TICKS_INFO – 買値と売値の変更でのティック、COPY_TICKS_TRADE –ラストと数量の変更でのティック、COPY_TICKS_ALL – 全てのティック。任意のタイプの要求に対して、前のティックの値がMqlTick構造体の残りのフィールドに追加されます。

from_msc

[in] ティックリクエストの初めの日。1970.01.01 から経過したミリ秒。*from_msc*パラメータが指定されていない場合、履歴の先頭からのティックが送信されます。time >= from_msc を持つティックが送信されます。

to_msc

[in] ティックリクエストの最後の日。1970.01.01 から経過したミリ秒。time <= to_msc を持つティックが送信されます。*to_msc*パラメータが指定されていない場合、履歴の最後までのティックが送信されます。

戻り値

複製されたティック数（エラーの場合は -1 ）GetLastError()は下記のエラーを返します。

- ERR_HISTORY_TIMEOUT – ティック同期の待機時間が終わり、関数はすべてのものを送信しました。
- ERR_HISTORY_SMALL_BUFFER – 静的バッファーが小さすぎます。配列が格納できる量だけが送信されました。
- ERR_NOT_ENOUGH_MEMORY – 指定された範囲から動的ティック配列に履歴を受け取るためのメモリが不十分です。ティック配列に十分なメモリを割り当てられませんでした。

注意事項

CopyTicksRange() 関数は、指定された範囲（例えば履歴の特定の日など）から厳密にティックを要求するために使用されます。同時に、CopyTicks()では開始日のみを指定することができます。例えば、月の初めから現在の瞬間までのすべてのティックを受け取ります。

© 2000–2025, MetaQuotes Ltd.

参照

SymbolInfoTick、現在価格の構造体、OnTick、CopyTicks

© 2000-2025, MetaQuotes Ltd.

iBars

該当する銘柄と期間の履歴内のバーの数を返します。

```
int  iBars(
   const string           symbol,          // 銘柄
   ENUM_TIMEFRAMES        timeframe        // 期間
   );
```

パラメータ

symbol

[in] 金融商品の名前です。NULL は現在のシンボルです。

timeframe

[in] 期間です。ENUM_TIMEFRAMES列挙体の値の1つです。0は現在のチャート期間を意味します。

戻り値

プラットフォーム設定の「チャートでのバーの最大数」パラメータで許容される値を超えない、該当する銘柄と期間の履歴内のバーの数を返します。

例:

```
   Print("Bar count on the 'EURUSD,H1' is ",iBars("EURUSD",PERIOD_H1));
```

参照

Bars

© 2000-2025, MetaQuotes Ltd.

iBarShift

バーを時刻によって検索します。この関数は、指定された時刻に対応するバーのインデックスを返します。

```
int  iBarShift(
   const string        symbol,          // 銘柄
   ENUM_TIMEFRAMES     timeframe,       // 期間
   datetime            time,            // 時間
   bool                exact=false      // モード
   );
```

パラメータ

symbol

[in] 金融商品の名前です。NULL は現在のシンボルです。

timeframe

[in] 期間です。ENUM_TIMEFRAMES列挙体の値の1つです。PERIOD_CURRENTは現在のチャート期間を意味します。

time

[in] 検索する時間の値

exact=false

[in] 指定された時間のバーが見つからない場合の戻り値です。. *exact=false*の場合、*iBarShift*は、開始時間が指定された時刻より早い(*time_open<time*)一番近くのバーのインデックスを返します。 そのようなバーが見つからない場合(指定された時間前の履歴が使用できない)、関数は-1を返します。*exact=true*の場合、iBarShiftは一番近いバーを検索せずにすぐに-1を返します。

戻り値

指定された時刻に対応するバーのインデックスを返します。指定された時間に対応するバーが見つからない場合(履歴にギャップがある場合)、関数は'*exact*'に応じて、-1または最も近いバーのインデックスを返します。

例:

```
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 日曜日
   datetime time=D'2002.04.25 12:00';
   string symbol="GBPUSD";
   ENUM_TIMEFRAMES tf=PERIOD_H1;
   bool exact=false;
//--- 指定された時間に対応するバーがない場合、iBarShiftは一番近いバーのインデックスを返す
   int bar_index=iBarShift(symbol,tf,time,exact);
//--- iBarShift()呼び出しの後でエラーコードを確認する
   int error=GetLastError();
   if(error!=0)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       PrintFormat("iBarShift(): GetLastError=%d - The requested date %s "+
                   "for %s %s is not found in the available history",
                   error,TimeToString(time),symbol,EnumToString(tf));
       return;
      }
//--- iBarShift()関数が正常に実行されて、exact=falseの結果を返す
   PrintFormat("1. %s %s %s(%s): bar index is %d (exact=%s)",
               symbol,EnumToString(tf),TimeToString(time),
               DayOfWeek(time),bar_index,string(exact));
   datetime bar_time=iTime(symbol,tf,bar_index);
   PrintFormat("Time of bar #%d is %s (%s)",
               bar_index,TimeToString(bar_time),DayOfWeek(bar_time));
//--- 指定された時間でバーのインデックスを要求し、バーがない場合は-1を返す
   exact=true;
   bar_index=iBarShift(symbol,tf,time,exact);
//--- iBarShift()関数が正常に実行されて、exact=trueの結果を返す
   PrintFormat("2. %s %s %s (%s):bar index is %d (exact=%s)",
               symbol,EnumToString(tf),TimeToString(time)
               ,DayOfWeek(time),bar_index,string(exact));
  }
//+------------------------------------------------------------------+
//|  曜日名を返す                                                    |
//+------------------------------------------------------------------+
string DayOfWeek(const datetime time)
  {
   MqlDateTime dt;
   string day="";
   TimeToStruct(time,dt);
   switch(dt.day_of_week)
     {
      case 0: day=EnumToString(SUNDAY);
      break;
      case 1: day=EnumToString(MONDAY);
      break;
      case 2: day=EnumToString(TUESDAY);
      break;
      case 3: day=EnumToString(WEDNESDAY);
      break;
      case 4: day=EnumToString(THURSDAY);
      break;
      case 5: day=EnumToString(FRIDAY);
      break;
      default:day=EnumToString(SATURDAY);
      break;
     }
//---
   return day;
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
/* 実行結果
   1. GBPUSD PERIOD_H1 2018.06.10 12:00(SUNDAY): bar index is 64 (exact=false)
   Time of bar #64 is 2018.06.08 23:00 (FRIDAY)
   2. GBPUSD PERIOD_H1 2018.06.10 12:00 (SUNDAY):bar index is -1 (exact=true)
*/
```

© 2000–2025, MetaQuotes Ltd.

iClose

対応するチャートのバー('shift'パラメータで示される)の終値を返します。

```
double  iClose(
   const string         symbol,          // 銘柄
   ENUM_TIMEFRAMES      timeframe,       // 期間
   int                  shift            // シフト
   );
```

パラメータ

symbol

[in] 金融商品の名前です。NULL は現在のシンボルです。

timeframe

[in] 期間です。ENUM_TIMEFRAMES列挙体の値の1つです。0は現在のチャート期間を意味します。

shift

[in] 時系列から受け取った値のインデックス(現在のバーに対する指定された数のバーによる後方シフト

戻り値

対応するチャートのバー('shift'パラメータで示される)の終値、またはエラー発生時には0を返します。エラーの詳細はGetLastError()関数を呼び出して取得できます。

注意事項

この関数は常に実際のデータを返します。この目的のために、それは各呼び出しの間に指定された銘柄/期間を時系列から要求します。これは、最初の関数呼び出し中に準備のできたデータが存在しない場合、結果を準備するのに時間がかかることを意味します。

この関数は以前の呼び出しの結果を格納しません。また、即時値を返すためのローカルキャッシュはありません。

例:

```
input int shift=0;
//+------------------------------------------------------------------+
//| "tick" 関数イベントハンドラ                                      |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);

   Comment(Symbol(),",",EnumToString(Period()),"\n",
```

© 2000-2025, MetaQuotes Ltd.

```
            "Time: "   ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
            "Open: "   ,DoubleToString(open,Digits()),"\n",
            "High: "   ,DoubleToString(high,Digits()),"\n",
            "Low: "    ,DoubleToString(low,Digits()),"\n",
            "Close: "  ,DoubleToString(close,Digits()),"\n",
            "Volume: ",IntegerToString(volume),"\n",
            "Bars: "   ,IntegerToString(bars),"\n"
            );
  }
```

参照

CopyClose、CopyRates

© 2000–2025, MetaQuotes Ltd.

iHigh

対応するチャートのバー('shift'パラメータで示される)の高値を返します。

```
double  iHigh(
   const string         symbol,          // 銘柄
   ENUM_TIMEFRAMES      timeframe,       // 期間
   int                  shift            // シフト
   );
```

パラメータ

symbol

[in] 金融商品の名前です。NULL は現在のシンボルです。

timeframe

[in] 期間です。ENUM_TIMEFRAMES列挙体の値の1つです。0は現在のチャート期間を意味します。

shift

[in] 時系列から受け取った値のインデックス(現在のバーに対する指定された数のバーによる後方シフト

戻り値

対応するチャートのバー('shift'パラメータで示される)の高値、またはエラー発生時には0を返します。エラーの詳細はGetLastError()関数を呼び出して取得できます。

注意事項

この関数は常に実際のデータを返します。この目的のために、それは各呼び出しの間に指定された銘柄/期間を時系列から要求します。これは、最初の関数呼び出し中に準備のできたデータが存在しない場合、結果を準備するのに時間がかかることを意味します。

この関数は以前の呼び出しの結果を格納しません。また、即時値を返すためのローカルキャッシュはありません。

例:

```
input int shift=0;
//+------------------------------------------------------------------+
//| "tick" 関数イベントハンドラ                                      |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);

   Comment(Symbol(),",",EnumToString(Period()),"\n",
```

© 2000-2025, MetaQuotes Ltd.

```
            "Time: "   ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
            "Open: "   ,DoubleToString(open,Digits()),"\n",
            "High: "   ,DoubleToString(high,Digits()),"\n",
            "Low: "    ,DoubleToString(low,Digits()),"\n",
            "Close: "  ,DoubleToString(close,Digits()),"\n",
            "Volume: ",IntegerToString(volume),"\n",
            "Bars: "   ,IntegerToString(bars),"\n"
            );
  }
```

参照

CopyHigh、CopyRates

© 2000–2025, MetaQuotes Ltd.

iHighest

対応するチャート上で見つかった最大値のインデックスを返します（現在のバーとの相対的なシフト）。

```
int  iHighest(
   const string        symbol,              // 銘柄
   ENUM_TIMEFRAMES     timeframe,           // 期間
   ENUM_SERIESMODE     type,                // 時系列識別子
   int                 count=WHOLE_ARRAY,   // 要素数
   int                 start=0              // インデックス
  );
```

パラメータ

symbol

[in] 検索される銘柄です。NULL は現在のシンボルです。

timeframe

[in] 期間です。ENUM_TIMEFRAMES列挙体の値の1つです。0は現在のチャート期間を意味します。

型

[in] 検索される時系列の識別子です。ENUM_SERIESMODE列挙体の任意の値に等しくなれます。

count=WHOLE_ARRAY

[in] 検索を実行する時系列内の要素の数（現在のバーからインデックスの増加方向に向かって）。

start=0

[in] 最大値の検索が開始される最初のバーのインデックス（現在のバーに対する相対的なシフト）。負の値は無視され、ゼロの値に置き換えられます。

戻り値

対応するチャート上で見つかった最大値のインデックス（現在のバーとの相対的なシフト）またはエラー発生時には-1を返します。エラーの詳細はGetLastError()関数を呼び出して取得できます。

例:

```
   double val;
//--- 連続する20本のバーの中で最も高い終値の計算
//--- 現在の期間のインデックス4からインデックス23まで
   int val_index=iHighest(NULL,0,MODE_CLOSE,20,4);
   if(val_index!=-1)
      val=High[val_index];
   else
      PrintFormat("iHighest() call error. Error code=%d",GetLastError());
```

© 2000–2025, MetaQuotes Ltd.

iLow

対応するチャートのバー('shift'パラメータで示される)の安値を返します。

```
double  iLow(
   const string         symbol,          // 銘柄
   ENUM_TIMEFRAMES      timeframe,       // 期間
   int                  shift            // シフト
   );
```

パラメータ

symbol

[in] 金融商品の名前です。NULL は現在のシンボルです。

timeframe

[in] 期間です。ENUM_TIMEFRAMES列挙体の値の1つです。0は現在のチャート期間を意味します。

shift

[in] 時系列から受け取った値のインデックス(現在のバーに対する指定された数のバーによる後方シフト

戻り値

対応するチャートのバー('shift'パラメータで示される)の安値、またはエラー発生時には0を返します。エラーの詳細はGetLastError()関数を呼び出して取得できます。

注意事項

この関数は常に実際のデータを返します。この目的のために、それは各呼び出しの間に指定された銘柄/期間を時系列から要求します。これは、最初の関数呼び出し中に準備のできたデータが存在しない場合、結果を準備するのに時間がかかることを意味します。

この関数は以前の呼び出しの結果を格納しません。また、即時値を返すためのローカルキャッシュはありません。

例:

```
input int shift=0;
//+------------------------------------------------------------------+
//| "tick" 関数イベントハンドラ                                      |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);

   Comment(Symbol(),",",EnumToString(Period()),"\n",
```

© 2000-2025, MetaQuotes Ltd.

```
            "Time: "   ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
            "Open: "   ,DoubleToString(open,Digits()),"\n",
            "High: "   ,DoubleToString(high,Digits()),"\n",
            "Low: "    ,DoubleToString(low,Digits()),"\n",
            "Close: "  ,DoubleToString(close,Digits()),"\n",
            "Volume: ",IntegerToString(volume),"\n",
            "Bars: "   ,IntegerToString(bars),"\n"
            );
  }
```

参照

CopyLow、CopyRates

© 2000–2025, MetaQuotes Ltd.

iLowest

該当するチャート上で見つかった最小値のインデックスを返します（現在のバーとの相対的なシフト）。

```
int  iLowest(
   const string        symbol,              // 銘柄
   ENUM_TIMEFRAMES     timeframe,           // 期間
   ENUM_SERIESMODE     type,                // 時系列識別子
   int                 count=WHOLE_ARRAY,   // 要素数
   int                 start=0              // インデックス
  );
```

パラメータ

symbol

[in] 検索される銘柄です。NULL は現在のシンボルです。

timeframe

[in] 期間です。ENUM_TIMEFRAMES列挙体の値の1つです。（0 は現在のチャート期間）

型

[in] 検索される時系列の識別子です。ENUM_SERIESMODE列挙体の任意の値に等しくなれます。

count=WHOLE_ARRAY

[in] 検索を実行する時系列内の要素の数（現在のバーからインデックスの増加方向に向かって）。

start=0

[in] 最小値の検索が開始される最初のバーのインデックス（現在のバーに対する相対的なシフト）。負の値は無視され、ゼロの値に置き換えられます。

戻り値

該当するチャート上で見つかった最小値のインデックス（現在のバーとの相対的なシフト）またはエラー発生時には-1を返します。エラーの詳細はGetLastError()関数を呼び出して取得できます。

例：

```
   double val;
//---連続する15のバーの中で実ボリュームの最小値を持つバーを検索する
//--- 現在の期間のインデックス10からインデックス24まで
   int val_index=iLowest(NULL,0,MODE_REAL_VOLUME,15,10);
   if(val_index!=-1)
      val=Low[val_index];
   else
      PrintFormat("iLowest() call error. Error code=%d",GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

iOpen

対応するチャートのバー('shift'パラメータで示される)の始値を返します。

```
double  iOpen(
   const string         symbol,          // 銘柄
   ENUM_TIMEFRAMES      timeframe,       // 期間
   int                  shift            // シフト
   );
```

パラメータ

symbol

[in] 金融商品の名前です。NULL は現在のシンボルです。

timeframe

[in] 期間です。ENUM_TIMEFRAMES列挙体の値の1つです。0は現在のチャート期間を意味します。

shift

[in] 時系列から受け取った値のインデックス(現在のバーに対する指定された数のバーによる後方シフト

戻り値

対応するチャートのバー('shift'パラメータで示される)の始値、またはエラー発生時には0を返します。エラーの詳細はGetLastError()関数を呼び出して取得できます。

注意事項

この関数は常に実際のデータを返します。この目的のために、それは各呼び出しの間に指定された銘柄/期間を時系列から要求します。これは、最初の関数呼び出し中に準備のできたデータが存在しない場合、結果を準備するのに時間がかかることを意味します。

この関数は以前の呼び出しの結果を格納しません。また、即時値を返すためのローカルキャッシュはありません。

例:

```
input int shift=0;
//+------------------------------------------------------------------+
//| "tick" 関数イベントハンドラ                                      |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);

   Comment(Symbol(),",",EnumToString(Period()),"\n",
```

© 2000-2025, MetaQuotes Ltd.

```
            "Time: "   ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
            "Open: "   ,DoubleToString(open,Digits()),"\n",
            "High: "   ,DoubleToString(high,Digits()),"\n",
            "Low: "    ,DoubleToString(low,Digits()),"\n",
            "Close: "  ,DoubleToString(close,Digits()),"\n",
            "Volume: ",IntegerToString(volume),"\n",
            "Bars: "   ,IntegerToString(bars),"\n"
            );
  }
```

参照

CopyOpen、CopyRates

© 2000–2025, MetaQuotes Ltd.

iTime

対応するチャートのバー('shift'パラメータで示される)の時間を返します。

```
datetime  iTime(
   const string         symbol,          // 銘柄
   ENUM_TIMEFRAMES      timeframe,       // 期間
   int                  shift            // シフト
   );
```

パラメータ

symbol

[in] 金融商品の名前です。NULL は現在のシンボルです。

timeframe

[in] 期間です。ENUM_TIMEFRAMES列挙体の値の1つです。0は現在のチャート期間を意味します。

shift

[in] 時系列から受け取った値のインデックス(現在のバーに対する指定された数のバーによる後方シフト

戻り値

対応するチャートのバー('shift'パラメータで示される)の開始時間、またはエラー発生時には0を返します。エラーの詳細はGetLastError()関数を呼び出して取得できます。

注意事項

この関数は常に実際のデータを返します。この目的のために、それは各呼び出しの間に指定された銘柄/期間を時系列から要求します。これは、最初の関数呼び出し中に準備のできたデータが存在しない場合、結果を準備するのに時間がかかることを意味します。

この関数は以前の呼び出しの結果を格納しません。また、即時値を返すためのローカルキャッシュはありません。

例:

```
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 日曜日
   datetime time=D'2018.06.10 12:00';
   string symbol="GBPUSD";
   ENUM_TIMEFRAMES tf=PERIOD_H1;
   bool exact=false;
//--- 指定された時間に対応するバーがない場合、iBarShiftは一番近いバーのインデックスを返す
   int bar_index=iBarShift(symbol,tf,time,exact);
   PrintFormat("1. %s %s %s(%s): bar index is %d (exact=%s)",
               symbol
,EnumToString(tf),TimeToString(time),DayOfWeek(time),bar_index,string(exact));
```

© 2000-2025, MetaQuotes Ltd.

```
   datetime bar_time=iTime(symbol,tf,bar_index);
   PrintFormat("Time of bar #%d is %s (%s)",
               bar_index,TimeToString(bar_time),DayOfWeek(bar_time));
//PrintFormat(iTime(symbol,tf,bar_index));
//--- 指定された時間でバーのインデックスを要求し、バーがない場合は-1を返す
   exact=true;
   bar_index=iBarShift(symbol,tf,time,exact);
   PrintFormat("2. %s %s %s (%s):bar index is %d (exact=%s)",
               symbol
,EnumToString(tf),TimeToString(time),DayOfWeek(time),bar_index,string(exact));
  }
//+------------------------------------------------------------------+
//|  曜日名を返す                                                    |
//+------------------------------------------------------------------+
string DayOfWeek(const datetime time)
  {
   MqlDateTime dt;
   string day="";
   TimeToStruct(time,dt);
   switch(dt.day_of_week)
     {
      case 0: day=EnumToString(SUNDAY);
      break;
      case 1: day=EnumToString(MONDAY);
      break;
      case 2: day=EnumToString(TUESDAY);
      break;
      case 3: day=EnumToString(WEDNESDAY);
      break;
      case 4: day=EnumToString(THURSDAY);
      break;
      case 5: day=EnumToString(FRIDAY);
      break;
      default:day=EnumToString(SATURDAY);
      break;
     }
//---
   return day;
  }
/* 結果:
   1. GBPUSD PERIOD_H1 2018.06.10 12:00(SUNDAY): bar index is 64 (exact=false)
   Time of bar #64 is 2018.06.08 23:00 (FRIDAY)
   2. GBPUSD PERIOD_H1 2018.06.10 12:00 (SUNDAY):bar index is -1 (exact=true)
*/
```

参照

CopyTime、CopyRates

© 2000-2025, MetaQuotes Ltd.

iTickVolume

対応するチャートのバー('shift'パラメータで示される) のティックボリュームを返します。

```
long  iTickVolume(
   const string         symbol,          // 銘柄
   ENUM_TIMEFRAMES      timeframe,       // 期間
   int                  shift            // シフト
   );
```

パラメータ

symbol

[in] 金融商品の名前です。NULL は現在のシンボルです。

timeframe

[in] 期間です。ENUM_TIMEFRAMES列挙体の値の1つです。0は現在のチャート期間を意味します。

shift

[in] 時系列から受け取った値のインデックス(現在のバーに対する指定された数のバーによる後方シフト

戻り値

対応するチャートのバー('shift'パラメータで示される) のティックボリューム、またはエラー発生時には0を返します。エラーの詳細はGetLastError()関数を呼び出して取得できます。

注意事項

この関数は常に実際のデータを返します。この目的のために、それは各呼び出しの間に指定された銘柄/期間を時系列から要求します。これは、最初の関数呼び出し中に準備のできたデータが存在しない場合、結果を準備するのに時間がかかることを意味します。

この関数は以前の呼び出しの結果を格納しません。また、即時値を返すためのローカルキャッシュはありません。

例:

```
input int shift=0;
//+------------------------------------------------------------------+
//| "tick" 関数イベントハンドラ                                      |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);

   Comment(Symbol(),",",EnumToString(Period()),"\n",
```

© 2000-2025, MetaQuotes Ltd.

```
            "Time: "   ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
            "Open: "   ,DoubleToString(open,Digits()),"\n",
            "High: "   ,DoubleToString(high,Digits()),"\n",
            "Low: "    ,DoubleToString(low,Digits()),"\n",
            "Close: "  ,DoubleToString(close,Digits()),"\n",
            "Volume: ",IntegerToString(volume),"\n",
            "Bars: "   ,IntegerToString(bars),"\n"
            );
  }
```

参照

CopyTickVolume、CopyRates

© 2000–2025, MetaQuotes Ltd.

iRealVolume

対応するチャートのバー('shift'パラメータで示される)の実ボリュームを返します。

```
long  iRealVolume(
   const string         symbol,          // 銘柄
   ENUM_TIMEFRAMES      timeframe,       // 期間
   int                  shift            // シフト
   );
```

パラメータ

symbol

[in] 金融商品の名前です。NULL は現在のシンボルです。

timeframe

[in] 期間です。ENUM_TIMEFRAMES列挙体の値の1つです。0は現在のチャート期間を意味します。

shift

[in] 時系列から受け取った値のインデックス(現在のバーに対する指定された数のバーによる後方シフト

戻り値

対応するチャートのバー('shift'パラメータで示される)の実ボリューム、またはエラー発生時には0を返します。エラーの詳細はGetLastError()関数を呼び出して取得できます。

注意事項

この関数は常に実際のデータを返します。この目的のために、それは各呼び出しの間に指定された銘柄/期間を時系列から要求します。これは、最初の関数呼び出し中に準備のできたデータが存在しない場合、結果を準備するのに時間がかかることを意味します。

この関数は以前の呼び出しの結果を格納しません。また、即時値を返すためのローカルキャッシュはありません。

例:

```
input int shift=0;
//+------------------------------------------------------------------+
//| "tick" 関数イベントハンドラ                                      |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);

   Comment(Symbol(),",",EnumToString(Period()),"\n",
```

© 2000-2025, MetaQuotes Ltd.

```
            "Time: "   ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
            "Open: "   ,DoubleToString(open,Digits()),"\n",
            "High: "   ,DoubleToString(high,Digits()),"\n",
            "Low: "    ,DoubleToString(low,Digits()),"\n",
            "Close: "  ,DoubleToString(close,Digits()),"\n",
            "Volume: ",IntegerToString(volume),"\n",
            "Bars: "   ,IntegerToString(bars),"\n"
            );
  }
```

参照

CopyRealVolume、CopyRates

© 2000–2025, MetaQuotes Ltd.

iVolume

対応するチャートのバー('shift'パラメータで示される) のティックボリュームを返します。

```
long  iVolume(
   const string         symbol,          // 銘柄
   ENUM_TIMEFRAMES      timeframe,       // 期間
   int                  shift            // シフト
   );
```

パラメータ

symbol

[in] 金融商品の名前です。NULL は現在のシンボルです。

timeframe

[in] 期間です。ENUM_TIMEFRAMES列挙体の値の1つです。0は現在のチャート期間を意味します。

shift

[in] 時系列から受け取った値のインデックス(現在のバーに対する指定された数のバーによる後方シフト

戻り値

対応するチャートのバー('shift'パラメータで示される) のティックボリューム、またはエラー発生時には0を返します。エラーの詳細はGetLastError()関数を呼び出して取得できます。

注意事項

この関数は常に実際のデータを返します。この目的のために、それは各呼び出しの間に指定された銘柄/期間を時系列から要求します。これは、最初の関数呼び出し中に準備のできたデータが存在しない場合、結果を準備するのに時間がかかることを意味します。

この関数は以前の呼び出しの結果を格納しません。また、即時値を返すためのローカルキャッシュはありません。

例:

```
input int shift=0;
//+------------------------------------------------------------------+
//| "tick" 関数イベントハンドラ                                      |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);

   Comment(Symbol(),",",EnumToString(Period()),"\n",
```

© 2000-2025, MetaQuotes Ltd.

```
            "Time: "   ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
            "Open: "   ,DoubleToString(open,Digits()),"\n",
            "High: "   ,DoubleToString(high,Digits()),"\n",
            "Low: "    ,DoubleToString(low,Digits()),"\n",
            "Close: "  ,DoubleToString(close,Digits()),"\n",
            "Volume: ",IntegerToString(volume),"\n",
            "Bars: "   ,IntegerToString(bars),"\n"
            );
  }
```

参照

CopyTickVolume、CopyRates

© 2000–2025, MetaQuotes Ltd.

iSpread

対応するチャートのバー('shift'パラメータで示される) のスプレッドを返します。

```
long  iSpread(
   const string         symbol,          // 銘柄
   ENUM_TIMEFRAMES      timeframe,       // 期間
   int                  shift            // シフト
   );
```

パラメータ

symbol

[in] 金融商品の名前です。NULL は現在のシンボルです。

timeframe

[in] 期間です。ENUM_TIMEFRAMES列挙体の値の1つです。0は現在のチャート期間を意味します。

shift

[in] 時系列から受け取った値のインデックス(現在のバーに対する指定された数のバーによる後方シフト

戻り値

対応するチャートのバー('shift'パラメータで示される) のスプレッド、またはエラー発生時には0を返します。エラーの詳細はGetLastError()関数を呼び出して取得できます。

注意事項

この関数は常に実際のデータを返します。この目的のために、それは各呼び出しの間に指定された銘柄/期間を時系列から要求します。これは、最初の関数呼び出し中に準備のできたデータが存在しない場合、結果を準備するのに時間がかかることを意味します。

この関数は以前の呼び出しの結果を格納しません。また、即時値を返すためのローカルキャッシュはありません。

例:

```
input int shift=0;
//+------------------------------------------------------------------+
//| "tick" 関数イベントハンドラ                                      |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);

   Comment(Symbol(),",",EnumToString(Period()),"\n",
```

© 2000-2025, MetaQuotes Ltd.

```
            "Time: "   ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
            "Open: "   ,DoubleToString(open,Digits()),"\n",
            "High: "   ,DoubleToString(high,Digits()),"\n",
            "Low: "    ,DoubleToString(low,Digits()),"\n",
            "Close: "  ,DoubleToString(close,Digits()),"\n",
            "Volume: ",IntegerToString(volume),"\n",
            "Bars: "   ,IntegerToString(bars),"\n"
            );
  }
```

参照

CopySpread、CopyRates

© 2000–2025, MetaQuotes Ltd.

カスタム銘柄

カスタム銘柄のプロパティを作成および編集するための関数。

端末を特定の取引サーバに接続すると、ブローカーに提供された金融銘柄時系列での作業が可能になります。利用可能な金融銘柄は、板情報ウィンドウに一覧表示されます。別の関数グループは、取引セッションや板情報といった銘柄のプロパティに関するデータを受信と更新を可能にします。

このセクションで説明する一連の関数は、カスタム銘柄の作成を可能にします。これには、ユーザは取引サーバの既存のシンボル、テキストファイルまたは外部データソースを適用することができます。

関数	アクション
CustomSymbolCreate	指定されたグループに指定された名前のカスタム銘柄を作成します。
CustomSymbolDelete	指定された名前のカスタム銘柄を削除します。
CustomSymbolSetInteger	カスタム銘柄の整数型プロパティ値を設定します。
CustomSymbolSetDouble	カスタム銘柄の実数型プロパティ値を設定します。
CustomSymbolSetString	カスタム銘柄の文字列型プロパティ値を設定します。
CustomSymbolSetMarginRate	カスタム銘柄の注文の種類と方向に応じて証拠金率を設定します。
CustomSymbolSetSessionQuote	指定された銘柄と曜日の指定された相場セッションの開始時刻と終了時刻を設定します。
CustomSymbolSetSessionTrade	指定された銘柄と曜日の指定された取引セッションの開始時刻と終了時刻を設定します。
CustomRatesDelete	指定した時間間隔内のカスタム銘柄の価格履歴からすべてのバーを削除します。
CustomRatesReplace	指定された時間間隔内のカスタム銘柄の価格履歴をMqlRates型の配列からのデータで完全に置き換えます。
CustomRatesUpdate	カスタム銘柄履歴に欠けているバーを追加し、既存のデータをMqlRates型配列のものに置き換えます。
CustomTicksAdd	カスタム銘柄の価格履歴にMqlTick型の配列からデータを追加します。カスタム銘柄は気配値ウィンドウで選択されなければなりません。
CustomTicksDelete	指定した時間間隔内のカスタム銘柄の価格履歴からすべてのティックを削除します。
CustomTicksReplace	指定された時間間隔内のカスタム銘柄の価格履歴をMqlTick型の配列からのデータで完全に置き換えます。
CustomBookAdd	銘柄の板情報の構造体を渡します

© 2000–2025, MetaQuotes Ltd.

CustomSymbolCreate

指定されたグループで指定された名前を持つ新しいカスタム銘柄を作成します。

```
bool  CustomSymbolCreate(
   const string      symbol_name,          // カスタム銘柄名
   const string      symbol_path="",       // 銘柄が作成されるグループ名
   const string      symbol_origin=NULL    // カスタム銘柄を作成するの基盤として使用される銘柄の名前
   );
```

パラメータ

symbol_name

[in] カスタム銘柄名。銘柄が作成されたグループまたはサブグループを含んではいけない。

symbol_path=""

[in] 銘柄を持つグループの名前。

symbol_origin=NULL

[in] 作成するカスタム銘柄のプロパティのコピー元となる銘柄の名前。カスタム銘柄を作成した後は、適切な関数を使用して任意のプロパティ値を必要なものに変更できます。<分節 146763>

戻り値

成功の場合はtrue、それ以外の場合はfalse。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

注意事項

すべてのカスタム銘柄は特別なCustomセクションで作成されます。グループ名が指定されていない場合（CustomSymbolCreate関数の*symbol_path* パラメータが空の文字列またはNULL）、カスタム銘柄はCustomセクションのルートに生成されます。ここでは、グループとサブグループをフォルダーとサブフォルダーとして表示できるファイルシステムとの類似性を引き出すことができます。

銘柄名とグループ名には、句読点、スペース、特殊文字を含まないラテン文字のみを含めることができます（"."、"_"、"&"、"#"のみを含めることができます）。"<"、">"、":"、" ""、"/"、"|"、"?"、"*"の使用は推奨されません。

カスタム銘柄名は、それが作成されたグループの名前に関係なく一意である必要があります。同名の銘柄が既に存在する場合、CustomSymbolCreate()関数は'false'を返し、それ以降のGetLastError()の呼び出しはエラー5300（ERR_NOT_CUSTOM_SYMBOL）または5304（ERR_CUSTOM_SYMBOL_EXIST）を返します。

銘柄名は31字を超えられません。さもないと、CustomSymbolCreate()は'false'を返し、エラー5302 - ERR_CUSTOM_SYMBOL_NAME_LONGがアクティブになります。

*symbol_path*パラメータを設定する方法は2つあります。

- カスタム銘柄名を含まないグループ名（例："CFD¥¥Metals"）。エラーを避けるにはこのオプションが最適です。
- または<グループ>名 + グループ区切り文字"¥¥"+<カスタム銘柄名>（例："CFD¥¥Metals¥¥Platinum"）。この場合、グループ名は正確にカスタム銘柄名で終わる必要があります。不一致の場合、カスタム銘柄は作成されますが、意図したグループ内には作成されません。例として、*symbol_path*="CFD\\Metals\\Platinum"で*symbol_name*="platinum" (register error)の場合、カスタム銘柄"platinum"は"Custom¥CFD¥Metals¥Platinum"グループに作成されます。

© 2000-2025, MetaQuotes Ltd.

SymbolInfoGetString("platinum", SYMBOL_PATH)関数は
値"Custom¥CFD¥Metals¥Platinum¥platinum"を返します。

SYMBOL_PATHプロパティは銘柄名で終わるパスを返すことにご注意ください。したがって、同じグループ内にカスタム銘柄を作成する場合は、変更を加えずにコピーすることはできません。この場合、上記の結果が得られないように、銘柄名を切り捨てる必要があります。

存在しない銘柄がsymbol_originパラメータとして設定されると、カスタム銘柄はsymbol_originパラメータが設定されていない場合と同じく、空で作成されます。この場合、エラー4301 – ERR_MARKET_UNKNOWN_SYMBOLがアクティブになります。

"Custom¥"、"¥"グループ区切り文字と銘柄名が最後に指定されている場合、symbol_pathパラメータは127文字を超えることはできません。

例：

```
//+------------------------------------------------------------------+
//|                                              CustomSymbolCreate.mq5 |
//|                                  Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"  // カスタマ銘柄名
#define   CUSTOM_SYMBOL_PATH     "Forex"        // 銘柄が作成されるグループの名前
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()       // カスタム銘柄の基となる銘柄の名前

//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- カスタム銘柄の作成に失敗した場合、操作ログにその旨を通知する
   if(!CustomSymbolCreate(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH,
CUSTOM_SYMBOL_ORIGIN))
     {
      Print("CustomSymbolCreate() failed. Error ", GetLastError());
      return;
     }

//--- 作成された銘柄の存在を確認し、それが作成されたグループを取得する
   bool   custom= false;
   bool   exist = SymbolExist(CUSTOM_SYMBOL_NAME, custom);
   string path  = SymbolInfoString(CUSTOM_SYMBOL_NAME, SYMBOL_PATH);

//--- 銘柄の作成結果とグループ名（銘柄のプロパティから指定および取得）を操作ログに出力する
   PrintFormat("Custom symbol '%s' created\n"+
```

© 2000–2025, MetaQuotes Ltd.

```
                "Symbol '%s' is exist: %s\n"+
                "Symbol '%s' is custom: %s\n"+
                "Path specified in the settings: '%s'\n"+
                "Path returned from the 'SYMBOL_PATH' property: '%s'",
                CUSTOM_SYMBOL_NAME,
                CUSTOM_SYMBOL_NAME, (string)exist,
                CUSTOM_SYMBOL_NAME, (string)custom,
                CUSTOM_SYMBOL_PATH,
                path);

//--- 2秒待ってから、作成された銘柄と操作ログ内の結果メッセージを削除する
   Sleep(2000);
   ResetLastError();
   bool deleted = CustomSymbolDelete(CUSTOM_SYMBOL_NAME);
   Print(deleted ? StringFormat("Custom symbol '%s' removed", CUSTOM_SYMBOL_NAME) :
StringFormat("CustomSymbolDelete() failed. Error ",GetLastError()));
   /*
   結果:
   Custom symbol 'EURUSD.C' created
   Symbol 'EURUSD.C' is exist: true
   Symbol 'EURUSD.C' is custom: true
   Path specified in the settings: 'Forex'
   Path returned from the 'SYMBOL_PATH' property: 'Custom\Forex\EURUSD.C'
   Custom symbol 'EURUSD.C' removed
   */
  }
```

参照

SymbolName、SymbolSelect、CustomSymbolDelete

© 2000-2025, MetaQuotes Ltd.

CustomSymbolDelete

指定された名前のカスタム銘柄を削除します。

```
bool  CustomSymbolDelete(
   const string      symbol_name          // カスタム銘柄名
   );
```

パラメータ

symbol

[in] カスタム銘柄名既存のシンボルの名前と一致してはいけません。

戻り値

成功の場合はtrue、それ以外の場合はfalse。エラー情報を取得するためにはGetLastError()関数を呼び出します。

注意事項

板情報に表示されているカスタム銘柄またはチャートが開かれているカスタム銘柄は削除できません。

例：

```
//+------------------------------------------------------------------+
//|                                              CustomSymbolDelete.mq5 |
//|                                  Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"  // カスタマ銘柄名
#define   CUSTOM_SYMBOL_PATH     "Forex"        // 銘柄が作成されるグループの名前
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()       // カスタム銘柄の基となる銘柄の名前

//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- カスタム銘柄の作成に失敗した場合、操作ログにその旨を通知する
   if(!CustomSymbolCreate(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH,
CUSTOM_SYMBOL_ORIGIN))
     {
      Print("CustomSymbolCreate() failed. Error ", GetLastError());
      return;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 作成された銘柄の存在を確認し、結果を操作ログに出力する
   bool custom= false;
   bool exist = SymbolExist(CUSTOM_SYMBOL_NAME, custom);
   PrintFormat("Custom symbol '%s' exists: %s", CUSTOM_SYMBOL_NAME, (string)exist);

//--- 2秒待ってから、作成された銘柄と操作ログ内の結果メッセージを削除する
   Sleep(2000);
   ResetLastError();
   bool deleted = CustomSymbolDelete(CUSTOM_SYMBOL_NAME);
   Print(deleted ? StringFormat("Custom symbol '%s' removed", CUSTOM_SYMBOL_NAME) :
StringFormat("CustomSymbolDelete() failed. Error ",GetLastError()));

//--- 作成された銘柄の存在を確認し、結果を操作ログに出力する
   exist = SymbolExist(CUSTOM_SYMBOL_NAME, custom);
   PrintFormat("Custom symbol '%s' exists: %s", CUSTOM_SYMBOL_NAME, (string)exist);
   /*
   結果:
   Custom symbol 'EURUSD.C' exists: true
   Custom symbol 'EURUSD.C' removed
   Custom symbol 'EURUSD.C' exists: false
   */
  }
```

参照

SymbolName、SymbolSelect、CustomSymbolCreate

© 2000-2025, MetaQuotes Ltd.

CustomSymbolSetInteger

カスタム銘柄の整数型プロパティ値を設定します。

```
bool  CustomSymbolSetInteger(
   const string               symbol_name,      // 銘柄名
   ENUM_SYMBOL_INFO_INTEGER   property_id,      // プロパティID
   long                       property_value    // プロパティ値
   );
```

パラメータ

symbol_name

[in] カスタム銘柄名

property_id

[in] 銘柄プロパティID値はENUM_SYMBOL_INFO_INTEGER列挙体のうちの一つです。

property_value

[in] プロパティ値を含むlong型の変数

戻り値

成功の場合はtrue、それ以外の場合はfalse。エラー情報を取得するためにはGetLastError()関数を呼び出します。

注意事項

これらのプロパティのいずれかが銘柄仕様で変更されている場合、カスタム銘柄の分足及びティックの履歴は完全に削除されます。

- SYMBOL_CHART_MODE – バー構築のための価格タイプ（BidまたはLast）
- SYMBOL_DIGITS – 価格表示に使われる小数点以下の桁数

カスタム銘柄履歴を削除した後、端末は更新されたプロパティを使用して新たに履歴作成を試みます。カスタム銘柄のプロパティが手動で変更された場合も同様です。

例：

```
//+------------------------------------------------------------------+
//|                                      CustomSymbolSetInteger.mq5 |
//|                                  Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"  // カスタマ銘柄名
#define   CUSTOM_SYMBOL_PATH     "Forex"        // 銘柄が作成されるグループの名前
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()       // カスタム銘柄の基となる銘柄の名前
```

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- カスタム銘柄を作成するときにエラーコードを取得する
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH,
CUSTOM_SYMBOL_ORIGIN);

//--- エラーコードが0（銘柄の作成に成功）でも5304（銘柄がすでに作成されている）でもない場合は、終了する
   if(create!=0 && create!=5304)
      return;

//--- カスタム銘柄の基となる銘柄のプロパティを取得して操作ログに出力する
//--- （取引モード、ストップ注文のインストールレベル、取引操作の凍結距離）
   ENUM_SYMBOL_TRADE_EXECUTION origin_exe_mode = (ENUM_SYMBOL_TRADE_EXECUTION)
SymbolInfoInteger(CUSTOM_SYMBOL_ORIGIN, SYMBOL_TRADE_EXEMODE);
   int origin_stops_level = (int)SymbolInfoInteger(CUSTOM_SYMBOL_ORIGIN,
SYMBOL_TRADE_STOPS_LEVEL);
   int origin_freeze_level= (int)SymbolInfoInteger(CUSTOM_SYMBOL_ORIGIN,
SYMBOL_TRADE_FREEZE_LEVEL);

   PrintFormat("The '%s' symbol from which the custom '%s' was created:\n"+
               "  Deal execution mode: %s\n  Stops Level: %d\n  Freeze Level: %d",
               CUSTOM_SYMBOL_ORIGIN, CUSTOM_SYMBOL_NAME,
               StringSubstr(EnumToString(origin_exe_mode), 23), origin_stops_level,
origin_freeze_level);

//--- カスタム銘柄プロパティのその他の値を設定する
   ResetLastError();
   bool res=true;
   res &=CustomSymbolSetInteger(CUSTOM_SYMBOL_NAME, SYMBOL_TRADE_EXEMODE,
SYMBOL_TRADE_EXECUTION_MARKET);
   res &=CustomSymbolSetInteger(CUSTOM_SYMBOL_NAME, SYMBOL_TRADE_STOPS_LEVEL, 10);
   res &=CustomSymbolSetInteger(CUSTOM_SYMBOL_NAME, SYMBOL_TRADE_FREEZE_LEVEL, 3);

//--- いずれかのプロパティの設定時にエラーが発生した場合は、操作ログに適切なメッセージを表示する
   if(!res)
      Print("CustomSymbolSetInteger() failed. Error ", GetLastError());

//--- 変更されたカスタム銘柄のプロパティを取得して操作ログに出力する
//--- （取引モード、ストップ注文のインストールレベル、取引操作の凍結距離）
   ENUM_SYMBOL_TRADE_EXECUTION custom_exe_mode = (ENUM_SYMBOL_TRADE_EXECUTION)
SymbolInfoInteger(CUSTOM_SYMBOL_NAME, SYMBOL_TRADE_EXEMODE);
   int custom_stops_level = (int)SymbolInfoInteger(CUSTOM_SYMBOL_NAME,
SYMBOL_TRADE_STOPS_LEVEL);
   int custom_freeze_level= (int)SymbolInfoInteger(CUSTOM_SYMBOL_NAME,
SYMBOL_TRADE_FREEZE_LEVEL);

   PrintFormat("Custom symbol '%s' based on '%s':\n"+
```

© 2000-2025, MetaQuotes Ltd.

```
                "  Deal execution mode: %s\n  Stops Level: %d\n  Freeze Level: %d",
                CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_ORIGIN,
                StringSubstr(EnumToString(custom_exe_mode), 23), custom_stops_level,
custom_freeze_level);

//--- チャートのコメントにスクリプト終了キーに関するヒントを表示する
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and
exit", CUSTOM_SYMBOL_NAME));

//--- EscキーまたはDel キーが押されるまで待機して無限ループを終了する
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- Del キーを押すと、作成したカスタム銘柄が削除される
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully",
CUSTOM_SYMBOL_NAME);
         break;
        }
     }
//--- 終了する前にチャートをクリアする
   Comment("");
   /*
  結果:
   The 'EURUSD' symbol from which the custom 'EURUSD.C' was created:
     Deal execution mode: INSTANT
     Stops Level: 0
     Freeze Level: 0
   Custom symbol 'EURUSD.C' based on 'EURUSD':
     Deal execution mode: MARKET
     Stops Level: 10
     Freeze Level: 3
   */
  }
//+------------------------------------------------------------------+
//| カスタム銘柄ルを作成し、エラーコードを返す                                |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const
string symbol_origin=NULL)
  {
//--- カスタム銘柄の基となる銘柄の名前をする
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- カスタム銘柄の作成に失敗し、エラー5304ではない場合は、操作ログに報告する
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       error=GetLastError();
       if(error!=5304)
          PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name,
symbol_path, origin, error);
      }
//--- 成功
   return(error);
  }
//+------------------------------------------------------------------+
//| カスタム銘柄を削除する                                           |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- 気配値表示ウィンドウから銘柄を非表示にする
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- カスタム銘柄の削除に失敗した場合は、操作ログにこれを報告してfalseを返す
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name,
GetLastError());
      return(false);
     }
//--- 成功
   return(true);
  }
```

参照

SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

CustomSymbolSetDouble

カスタム銘柄の実数型プロパティ値を設定します。

```
bool  CustomSymbolSetDouble(
   const string              symbol_name,      // 銘柄名
   ENUM_SYMBOL_INFO_DOUBLE   property_id,      // プロパティID
   double                    property_value    // プロパティ値
   );
```

パラメータ

symbol_name

[in] カスタム銘柄名

property_id

[in] 銘柄プロパティID値はENUM_SYMBOL_INFO_DOUBLE列挙体のうちの一つです。

property_value

[in] プロパティ値を含むdouble型の変数

戻り値

成功の場合はtrue、それ以外の場合はfalse。エラー情報を取得するためにはGetLastError()関数を呼び出します。

注意事項

これらのプロパティのいずれかが銘柄仕様で変更されている場合、カスタム銘柄の分足及びティックの履歴は完全に削除されます。

- SYMBOL_POINT – 1ポイント値
- SYMBOL_TRADE_TICK_SIZE – 許容される最低価格の変更を指定するティックの値
- SYMBOL_TRADE_TICK_VALUE – 収益性の高いポジションの1ティック価格変更値

カスタム銘柄履歴を削除した後、端末は更新されたプロパティを使用して新たに履歴作成を試みます。カスタム銘柄のプロパティが手動で変更された場合も同様です。

例：

```
//+------------------------------------------------------------------+
//|                                        CustomSymbolSetDouble.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd.   |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"  // カスタマ銘柄名
#define   CUSTOM_SYMBOL_PATH     "Forex"        // 銘柄が作成されるグループの名前
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()       // カスタム銘柄の基となる銘柄の名前
```

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- カスタム銘柄を作成するときにエラーコードを取得する
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH,
CUSTOM_SYMBOL_ORIGIN);

//--- エラーコードが0（銘柄の作成に成功）でも5304（銘柄がすでに作成されている）でもない場合は、終了する
   if(create!=0 && create!=5304)
      return;

//--- カスタム銘柄の基となる銘柄のプロパティを取得して操作ログに出力する
//--- （取引実行の最小ボリューム、最大ボリューム、最小ボリューム変更ステップ）
   double origin_vol_min = SymbolInfoDouble(CUSTOM_SYMBOL_ORIGIN,
SYMBOL_VOLUME_MIN);
   double origin_vol_max = SymbolInfoDouble(CUSTOM_SYMBOL_ORIGIN,
SYMBOL_VOLUME_MAX);
   double origin_vol_step= SymbolInfoDouble(CUSTOM_SYMBOL_ORIGIN,
SYMBOL_VOLUME_STEP);

   PrintFormat("The '%s' symbol from which the custom '%s' was created:\n"+
               "  Volume Min: %.2f\n  Volume Max: %.2f\n  Volume Step: %.2f",
               CUSTOM_SYMBOL_ORIGIN, CUSTOM_SYMBOL_NAME,
               origin_vol_min, origin_vol_max, origin_vol_step);

//--- カスタム銘柄プロパティのその他の値を設定する
   ResetLastError();
   bool res=true;
   res &=CustomSymbolSetDouble(CUSTOM_SYMBOL_NAME, SYMBOL_VOLUME_MIN, 0.1);
   res &=CustomSymbolSetDouble(CUSTOM_SYMBOL_NAME, SYMBOL_VOLUME_MAX, 1000);
   res &=CustomSymbolSetDouble(CUSTOM_SYMBOL_NAME, SYMBOL_VOLUME_STEP, 0.1);

//--- いずれかのプロパティの設定時にエラーが発生した場合は、操作ログに適切なメッセージを表示する
   if(!res)
      Print("CustomSymbolSetDouble() failed. Error ", GetLastError());

//--- 変更されたカスタム銘柄のプロパティを取得して操作ログに出力する
//--- （取引実行の最小ボリューム、最大ボリューム、最小ボリューム変更ステップ）
   double custom_vol_min = SymbolInfoDouble(CUSTOM_SYMBOL_NAME, SYMBOL_VOLUME_MIN);
   double custom_vol_max = SymbolInfoDouble(CUSTOM_SYMBOL_NAME, SYMBOL_VOLUME_MAX);
   double custom_vol_step= SymbolInfoDouble(CUSTOM_SYMBOL_NAME, SYMBOL_VOLUME_STEP);

   PrintFormat("Custom symbol '%s' based on '%s':\n"+
               "  Volume Min: %.2f\n  Volume Max: %.2f\n  Volume Step: %.2f",
               CUSTOM_SYMBOL_ORIGIN, CUSTOM_SYMBOL_NAME,
               custom_vol_min, custom_vol_max, custom_vol_step);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- チャートのコメントにスクリプト終了キーに関するヒントを表示する
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and
exit", CUSTOM_SYMBOL_NAME));

//--- EscキーまたはDel キーが押されるまで待機して無限ループを終了する
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- Del キーを押すと、作成したカスタム銘柄が削除される
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully",
CUSTOM_SYMBOL_NAME);
         break;
        }
     }
//--- 終了する前にチャートをクリアする
   Comment("");
   /*
   結果:
   The 'EURUSD' symbol from which the custom 'EURUSD.C' was created:
     Volume Min: 0.01
     Volume Max: 500.00
     Volume Step: 0.01
   Custom symbol 'EURUSD' based on 'EURUSD.C':
     Volume Min: 0.10
     Volume Max: 1000.00
     Volume Step: 0.10
   */
  }
//+------------------------------------------------------------------+
//| カスタム銘柄ルを作成し、エラーコードを返す                                    |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const
string symbol_origin=NULL)
  {
//--- カスタム銘柄の基となる銘柄の名前をする
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- カスタム銘柄の作成に失敗し、エラー5304ではない場合は、操作ログに報告する
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
```

© 2000-2025, MetaQuotes Ltd.

```
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name,
symbol_path, origin, error);
     }
//--- 成功
   return(error);
  }
//+------------------------------------------------------------------+
//| カスタム銘柄を削除する                                           |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- 気配値表示ウィンドウから銘柄を非表示にする
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- カスタム銘柄の削除に失敗した場合は、操作ログにこれを報告してfalseを返す
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name,
GetLastError());
      return(false);
     }
//--- 成功
   return(true);
  }
```

参照

SymbolInfoDouble

© 2000-2025, MetaQuotes Ltd.

CustomSymbolSetString

カスタム銘柄の文字列型プロパティ値を設定します。

```
bool  CustomSymbolSetString(
   const string               symbol_name,      // 銘柄名
   ENUM_SYMBOL_INFO_STRING    property_id,      // プロパティID
   string                     property_value    // プロパティ値
   );
```

パラメータ

symbol_name

[in] カスタム銘柄名

property_id

[in] 銘柄プロパティID値はENUM_SYMBOL_INFO_STRING列挙体のうちの一つです。

property_value

[in] プロパティ値を含む文字列型の変数

戻り値

成功の場合はtrue、それ以外の場合はfalse。エラー情報を取得するためにはGetLastError()関数を呼び出します。

注意事項

銘柄仕様でSYMBOL_FORMULAプロパティ(カスタム銘柄価格構成の式を設定)が変更された場合、カスタム銘柄の分足及びティック履歴は完全に削除されます。カスタム銘柄履歴を削除した後、端末は新しい式を使用して新たに履歴作成を試みます。カスタム銘柄の式が手動で変更された場合も同様です。

例：

```
//+------------------------------------------------------------------+
//|                                        CustomSymbolSetString.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd.   |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"  // カスタマ銘柄名
#define   CUSTOM_SYMBOL_PATH     "Forex"        // 銘柄が作成されるグループの名前
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()       // カスタム銘柄の基となる銘柄の名前

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- カスタム銘柄を作成するときにエラーコードを取得する
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH,
CUSTOM_SYMBOL_ORIGIN);

//--- エラーコードが0(銘柄の作成に成功)でも5304(銘柄がすでに作成されている)でもない場合は、終了する
   if(create!=0 && create!=5304)
      return;

//--- カスタム銘柄の基となる銘柄のプロパティを取得して操作ログに出力する
//--- (取引実行の最小ボリューム、最大ボリューム、最小ボリューム変更ステップ)
   string origin_basis    = SymbolInfoString(CUSTOM_SYMBOL_ORIGIN, SYMBOL_BASIS);
   string origin_category = SymbolInfoString(CUSTOM_SYMBOL_ORIGIN, SYMBOL_CATEGORY);
   string origin_formula  = SymbolInfoString(CUSTOM_SYMBOL_ORIGIN, SYMBOL_FORMULA);

   PrintFormat("The '%s' symbol from which the custom '%s' was created:\n"+
               "  Basis: %s\n  Category: %s\n  Formula: %s",
               CUSTOM_SYMBOL_ORIGIN, CUSTOM_SYMBOL_NAME,
               origin_basis, origin_category, origin_formula);

//--- カスタム銘柄プロパティのその他の値を設定する
   ResetLastError();
   bool res=true;
   res &=CustomSymbolSetString(CUSTOM_SYMBOL_NAME, SYMBOL_BASIS,
CUSTOM_SYMBOL_ORIGIN);
   res &=CustomSymbolSetString(CUSTOM_SYMBOL_NAME, SYMBOL_CATEGORY, "FX");
   res &=CustomSymbolSetString(CUSTOM_SYMBOL_NAME, SYMBOL_FORMULA, ("1.0 /
"+CUSTOM_SYMBOL_ORIGIN));

//--- いずれかのプロパティの設定時にエラーが発生した場合は、操作ログに適切なメッセージを表示する
   if(!res)
      Print("CustomSymbolSetString() failed. Error ", GetLastError());

//--- 変更されたカスタム銘柄のプロパティを取得して操作ログに出力する
//--- (取引実行の最小ボリューム、最大ボリューム、最小ボリューム変更ステップ)
   string custom_basis    = SymbolInfoString(CUSTOM_SYMBOL_NAME, SYMBOL_BASIS);
   string custom_category = SymbolInfoString(CUSTOM_SYMBOL_NAME, SYMBOL_CATEGORY);
   string custom_formula  = SymbolInfoString(CUSTOM_SYMBOL_NAME, SYMBOL_FORMULA);

   PrintFormat("Custom symbol '%s' based on '%s':\n"+
               "  Basis: %s\n  Category: %s\n  Formula: %s",
               CUSTOM_SYMBOL_ORIGIN, CUSTOM_SYMBOL_NAME,
               custom_basis, custom_category, custom_formula);

//--- チャートのコメントにスクリプト終了キーに関するヒントを表示する
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and
exit", CUSTOM_SYMBOL_NAME));

//--- EscキーまたはDel キーが押されるまで待機して無限ループを終了する
```

© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- Del キーを押すと、作成したカスタム銘柄が削除される
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully",
CUSTOM_SYMBOL_NAME);
         break;
        }
     }
//--- 終了する前にチャートをクリアする
   Comment("");
   /*
   結果:
   The 'EURUSD' symbol from which the custom 'EURUSD.C' was created:
     Basis:
     Category:
     Formula:
   Custom symbol 'EURUSD' based on 'EURUSD.C':
     Basis: EURUSD
     Category: FX
     Formula: 1.0 / EURUSD
   */
  }
//+------------------------------------------------------------------+
//| カスタム銘柄ルを作成し、エラーコードを返す                                    |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const
string symbol_origin=NULL)
  {
//--- カスタム銘柄の基となる銘柄の名前をする
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- カスタム銘柄の作成に失敗し、エラー5304ではない場合は、操作ログに報告する
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name,
symbol_path, origin, error);
     }
//--- 成功
   return(error);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| カスタム銘柄を削除する                                                   |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- 気配値表示ウィンドウから銘柄を非表示にする
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- カスタム銘柄の削除に失敗した場合は、操作ログにこれを報告してfalseを返す
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name,
GetLastError());
      return(false);
     }
//--- 成功
   return(true);
  }
```

参照

SymbolInfoString

© 2000-2025, MetaQuotes Ltd.

CustomSymbolSetMarginRate

カスタム銘柄の注文の種類と方向に応じて証拠金率を設定します。

```
bool  CustomSymbolSetMarginRate(
   const string       symbol_name,              // 銘柄名
   ENUM_ORDER_TYPE    order_type,               // 注文の種類
   double             initial_margin_rate,      // 当初証拠金率
   double             maintenance_margin_rate   // 維持証拠金率
   );
```

パラメータ

symbol_name

[in] カスタム銘柄名

order_type

[in] 注文の種類

initial_margin_rate

[in] 当初証拠金率を表すdouble型の変数。当初証拠金とは、適切な方向の1ロットの取引に対する保証金です。レートに当初証拠金を掛けて、指定された種類の注文を出す時に口座に予約される金額を受け取ります。

maintenance_margin_rate

[in] 維持証拠金率を表すdouble型の変数。維持証拠金は、適切な方向の1ロットのポジションを維持するのに必要な最低限の証拠金です。レートに維持証拠金を掛けて、指定された種類の注文を出す時に口座に予約される金額を受け取ります。

戻り値

成功の場合はtrue、それ以外の場合はfalse。エラー情報を取得するためにはGetLastError()関数を呼び出します。

例：

```
//+------------------------------------------------------------------+
//|                                    CustomSymbolSetMarginRate.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd.   |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"  // カスタマ銘柄名
#define   CUSTOM_SYMBOL_PATH     "Forex"        // 銘柄が作成されるグループの名前
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()       // カスタム銘柄の基となる銘柄の名前

#define   INITIAL_MARGIN_RATE       1.5         // 当初証拠金率
```

© 2000–2025, MetaQuotes Ltd.

```
#define   MAINTENANCE_MARGIN_RATE   1.5          // 維持証拠金率

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- カスタム銘柄を作成するときにエラーコードを取得する
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH,
CUSTOM_SYMBOL_ORIGIN);

//--- エラーコードが0（銘柄の作成に成功）でも5304（銘柄がすでに作成されている）でもない場合は、終了する
   if(create!=0 && create!=5304)
      return;

//--- カスタム銘柄の基となる銘柄のプロパティを取得して操作ログに出力する
//---（買い注文と売り注文の当初証拠金率と維持証拠金率）
   double initial_margin_rate_buy=0;
   double maintenance_margin_rate_buy=0;
   double initial_margin_rate_sell=0;
   double maintenance_margin_rate_sell=0;

   if(!GetSymbolMarginRate(CUSTOM_SYMBOL_ORIGIN, ORDER_TYPE_BUY,
initial_margin_rate_buy, maintenance_margin_rate_buy))
      return;
   if(!GetSymbolMarginRate(CUSTOM_SYMBOL_ORIGIN,
ORDER_TYPE_SELL,initial_margin_rate_sell,maintenance_margin_rate_sell))
      return;

   PrintFormat("The '%s' symbol from which the custom '%s' was created:\n"+
               "  Buy order initial margin rate: %f\n  Buy order maintenance margin
rate: %f\n"+
               "  Sell order initial margin rate: %f\n  Sell order maintenance
margin rate: %f",
               CUSTOM_SYMBOL_ORIGIN, CUSTOM_SYMBOL_NAME,
               initial_margin_rate_buy, maintenance_margin_rate_buy,
initial_margin_rate_sell, maintenance_margin_rate_sell);

//--- カスタム銘柄プロパティのその他の値を設定する
   ResetLastError();
   bool res=true;
   res &=CustomSymbolSetMarginRate(CUSTOM_SYMBOL_NAME, ORDER_TYPE_BUY,
INITIAL_MARGIN_RATE, MAINTENANCE_MARGIN_RATE);
   res &=CustomSymbolSetMarginRate(CUSTOM_SYMBOL_NAME,
ORDER_TYPE_SELL,INITIAL_MARGIN_RATE, MAINTENANCE_MARGIN_RATE);

//--- いずれかのプロパティの設定時にエラーが発生した場合は、操作ログに適切なメッセージを表示する
   if(!res)
      Print("CustomSymbolSetMarginRate() failed. Error ", GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 変更されたカスタム銘柄のプロパティを取得して操作ログに出力する
//--- ( 買い注文と売り注文の当初証拠金率と維持証拠金率)
   if(!GetSymbolMarginRate(CUSTOM_SYMBOL_NAME, ORDER_TYPE_BUY,
initial_margin_rate_buy, maintenance_margin_rate_buy))
      return;
   if(!GetSymbolMarginRate(CUSTOM_SYMBOL_NAME,
ORDER_TYPE_SELL,initial_margin_rate_sell,maintenance_margin_rate_sell))
      return;

   PrintFormat("Custom symbol '%s' based on '%s':\n"+
               "  Buy order initial margin rate: %f\n  Buy order maintenance margin
rate: %f\n"+
               "  Sell order initial margin rate: %f\n  Sell order maintenance
margin rate: %f",
               CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_ORIGIN,
               initial_margin_rate_buy, maintenance_margin_rate_buy,
initial_margin_rate_sell, maintenance_margin_rate_sell);

//--- チャートのコメントにスクリプト終了キーに関するヒントを表示する
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and
exit", CUSTOM_SYMBOL_NAME));

//--- EscキーまたはDel キーが押されるまで待機して無限ループを終了する
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- Del キーを押すと、作成したカスタム銘柄が削除される
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully",
CUSTOM_SYMBOL_NAME);
         break;
        }
     }
//--- 終了する前にチャートをクリアする
   Comment("");
   /*
   結果:
   The 'EURUSD' symbol from which the custom 'EURUSD.C' was created:
     Buy order initial margin rate: 1.000000
     Buy order maintenance margin rate: 0.000000
     Sell order initial margin rate: 1.000000
     Sell order maintenance margin rate: 0.000000
   Custom symbol 'EURUSD.C' based on 'EURUSD':
     Buy order initial margin rate: 1.500000
     Buy order maintenance margin rate: 1.500000
     Sell order initial margin rate: 1.500000
     Sell order maintenance margin rate: 1.500000
   */
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| カスタム銘柄ルを作成し、エラーコードを返す                                  |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const
string symbol_origin=NULL)
  {
//--- カスタム銘柄の基となる銘柄の名前をする
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- カスタム銘柄の作成に失敗し、エラー5304ではない場合は、操作ログに報告する
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name,
symbol_path, origin, error);
     }
//--- 成功
   return(error);
  }
//+------------------------------------------------------------------+
//| カスタム銘柄を削除する                                                |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- 気配値表示ウィンドウから銘柄を非表示にする
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- カスタム銘柄の削除に失敗した場合は、操作ログにこれを報告してfalseを返す
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name,
GetLastError());
      return(false);
     }
//--- 成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 証拠金率を返す                                                     |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool GetSymbolMarginRate(const string symbol, const ENUM_ORDER_TYPE order_type,
double &initial_margin_rate, double &maintenance_margin_rate)
  {
   ResetLastError();
   if(!SymbolInfoMarginRate(symbol, order_type, initial_margin_rate,
maintenance_margin_rate))
     {
      PrintFormat("%s: SymbolInfoMarginRate(%s, %s) failed. Error %d",__FUNCTION__,
symbol, EnumToString(order_type), GetLastError());
      return false;
     }
   return true;
  }
```

参照

SymbolInfoMarginRate

© 2000–2025, MetaQuotes Ltd.

CustomSymbolSetSessionQuote

指定された銘柄と曜日の指定された相場セッションの開始時刻と終了時刻を設定します。

```
bool  CustomSymbolSetSessionQuote(
   const string      symbol_name,          // 銘柄名
   ENUM_DAY_OF_WEEK  day_of_week,          // 曜日
   uint              session_index,        // セッションのインデックス
   datetime          from,                 // セッションの開始時間
   datetime          to                    // セッションの終了時間
   );
```

パラメータ

symbol_name

[in] カスタム銘柄名

ENUM_DAY_OF_WEEK

[in] ENUM_DAY_OF_WEEK列挙からの値である曜日

uint

[in] 開始時刻と終了時刻を設定するセッションのインデックス。セッションインデックスは0から始まります。

from

[in] セッション開始時間（00:00からの秒数）で、変数のデータ値は無視されます。

to

[in] セッション終了時間（00:00からの秒数）で、変数のデータ値は無視されます。

戻り値

成功の場合はtrue、それ以外の場合はfalse。エラー情報を取得するためにはGetLastError()関数を呼び出します。

注意事項

指定された*session_index*を持つセッションが既存する場合、この関数は単にセッションの開始と終了を編集します。

セッションの開始パラメータと終了パラメータにゼロが渡された場合（*from=0*と*to=0*）、適切な*session_index*を持つセッションが削除され、セッションインデックスが下向きにシフトされます。

セッションは順番にしか追加できません。言い換えれば*session_index=1*は、インデックスが0であるセッションが既存する場合にのみ追加できます。このルールが守られていないと、新しいセッションは作成されず、関数自体は「false」を返します。

例：

```
//+------------------------------------------------------------------+
//|                                  CustomSymbolSetSessionQuote.mq5 |
//|                                  Copyright 2024, MetaQuotes Ltd. |
```

© 2000–2025, MetaQuotes Ltd.

```
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"            // カスタム銘柄名
#define   CUSTOM_SYMBOL_PATH     "Forex"                  // 銘柄が作成されるグループの名前
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()           // カスタム銘柄の基となる銘柄の名前

#define   SESSION_0_FROM         D'1970.01.01 00:15:00'  // session 0 start time
#define   SESSION_0_TO           D'1970.01.01 11:59:00'  // session 0 end time
#define   SESSION_1_FROM         D'1970.01.01 12:15:00'  // session 1 start time
#define   SESSION_1_TO           D'1970.01.01 23:59:00'  // session 1 end time

//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- カスタム銘柄を作成するときにエラーコードを取得する
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH,
CUSTOM_SYMBOL_ORIGIN);

//--- エラーコードが0(銘柄の作成に成功)でも5304(銘柄がすでに作成されている)でもない場合は、終了する
   if(create!=0 && create!=5304)
      return;

//--- 基となる銘柄とセッションインデックスを含むヘッダーを出力し
//---月曜日から金曜日までの曜日ごとのループで、操作ログ内の各相場セッションの開始時刻と終了時刻を出力する
   for(int session=0; session<2; session++)
     {
      PrintFormat("Quote session %d of '%s' symbol from which the custom '%s' was
created", session, CUSTOM_SYMBOL_ORIGIN, CUSTOM_SYMBOL_NAME);
      for(int day_of_week=MONDAY; day_of_week<SATURDAY; day_of_week++)
         SymbolInfoSessionQuotePrint(CUSTOM_SYMBOL_ORIGIN, (ENUM_DAY_OF_WEEK)
day_of_week, session);
     }

//--- 2つのセッションによるループ
   bool res=true;
   for(int session=0; session<2; session++)
     {
      datetime from = SESSION_0_FROM;
      datetime to   = SESSION_0_TO;
      if(session>0)
        {
         from = SESSION_1_FROM;
         to   = SESSION_1_TO;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   //--- 各曜日のカスタム銘柄の相場セッション時間を設定する
   ResetLastError();
   for(int day_of_week=MONDAY; day_of_week<SATURDAY; day_of_week++)
      res &=CustomSymbolSetSessionQuote(CUSTOM_SYMBOL_NAME, (ENUM_DAY_OF_WEEK)
day_of_week, session, from, to);
  }

//--- いずれかのセッションの設定時にエラーが発生した場合は、銘柄に適切なメッセージを表示する
 if(!res)
   Print("CustomSymbolSetSessionQuote() failed. Error ", GetLastError());

//--- カスタム銘柄とセッションインデックスを含むヘッダーを出力し
//---月曜日から金曜日までの曜日ごとのループで、操作ログ内の各相場セッションの開始時刻と終了時刻を出力する
 for(int session=0; session<2; session++)
  {
   PrintFormat("Quote session %d of custom symbol '%s' based on '%s'", session,
CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_ORIGIN);
   for(int day_of_week=MONDAY; day_of_week<SATURDAY; day_of_week++)
      SymbolInfoSessionQuotePrint(CUSTOM_SYMBOL_NAME, (ENUM_DAY_OF_WEEK)
day_of_week, session);
  }

//--- チャートのコメントにスクリプト終了キーに関するヒントを表示する
 Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and
exit", CUSTOM_SYMBOL_NAME));
//--- EscキーまたはDel キーが押されるまで待機して無限ループを終了する
 while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
  {
   Sleep(16);
   //--- Del キーを押すと、作成したカスタム銘柄が削除される
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
     {
      if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
         PrintFormat("Custom symbol '%s' deleted successfully",
CUSTOM_SYMBOL_NAME);
      break;
     }
  }
//--- 終了する前にチャートをクリアする
 Comment("");
 /*
 結果:
 Quote session 0 of 'EURUSD' symbol from which the custom 'EURUSD.C' was created
 - Monday     00:15 - 23:55
 - Tuesday    00:15 - 23:55
 - Wednesday  00:15 - 23:55
 - Thursday   00:15 - 23:55
 - Friday     00:15 - 23:55
 Quote session 1 of 'EURUSD' symbol from which the custom 'EURUSD.C' was created
```

© 2000-2025, MetaQuotes Ltd.

```
   - Monday     Session not set
   - Tuesday    Session not set
   - Wednesday  Session not set
   - Thursday   Session not set
   - Friday     Session not set
   Quote session 0 of custom symbol 'EURUSD.C' based on 'EURUSD'
   - Monday     00:15 - 11:59
   - Tuesday    00:15 - 11:59
   - Wednesday  00:15 - 11:59
   - Thursday   00:15 - 11:59
   - Friday     00:15 - 11:59
   Quote session 1 of custom symbol 'EURUSD.C' based on 'EURUSD'
   - Monday     12:15 - 23:59
   - Tuesday    12:15 - 23:59
   - Wednesday  12:15 - 23:59
   - Thursday   12:15 - 23:59
   - Friday     12:15 - 23:59
   */
  }
//+------------------------------------------------------------------+
//| カスタム銘柄ルを作成し、エラーコードを返す                                  |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const
string symbol_origin=NULL)
  {
//--- カスタム銘柄の基となる銘柄の名前をする
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- カスタム銘柄の作成に失敗し、エラー5304ではない場合は、操作ログに報告する
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name,
symbol_path, origin, error);
     }
//--- 成功
   return(error);
  }
//+------------------------------------------------------------------+
//| カスタム銘柄を削除する                                                |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- 気配値表示ウィンドウから銘柄を非表示にする
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
       return(false);
      }

//--- カスタム銘柄の削除に失敗した場合は、操作ログにこれを報告してfalseを返す
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
      {
       PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name,
GetLastError());
       return(false);
      }
//--- 成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指定されたシンボルと曜日の指定された相場セッションの                 |
//| 開始時刻と終了時刻を操作ログに送信する                             |
//+------------------------------------------------------------------+
void SymbolInfoSessionQuotePrint(const string symbol, const ENUM_DAY_OF_WEEK
day_of_week, const uint session_index)
  {
//--- 相場セッションの開始と終了を記録する変数を宣言する
   datetime date_from;  // session start time
   datetime date_to;    // session end time

//--- 列挙定数から曜日名を作成する
   string week_day=EnumToString(day_of_week);
   if(week_day.Lower())
      week_day.SetChar(0, ushort(week_day.GetChar(0)-32));

//--- 銘柄と曜日別に相場セッションからデータを取得する
   if(!SymbolInfoSessionQuote(symbol, day_of_week, session_index, date_from,
date_to))
      {
       int err=GetLastError();
       string message=(err==4307 ? StringFormat("- %-10s Session not set",
week_day) :
                       StringFormat("SymbolInfoSessionQuote(%s, %s, session %d)
failed. Error %d", symbol, week_day, session_index, GetLastError()));
       Print(message);
       return;
      }

//--- 指定された相場セッションのデータを操作ログに送信する
   PrintFormat("- %-10s %s - %s", week_day, TimeToString(date_from, TIME_MINUTES),
TimeToString(date_to, TIME_MINUTES));
  }
```

© 2000-2025, MetaQuotes Ltd.

参照

SymbolInfoSessionQuote、Symbol info、TimeToStruct、Date structure

© 2000–2025, MetaQuotes Ltd.

CustomSymbolSetSessionTrade

指定された銘柄と曜日の指定された取引セッションの開始時刻と終了時刻を設定します。

```
bool  CustomSymbolSetSessionTrade(
   const string      symbol_name,          // 銘柄名
   ENUM_DAY_OF_WEEK  day_of_week,          // 曜日
   uint              session_index,        // セッションのインデックス
   datetime          from,                 // セッションの開始時間
   datetime          to                    // セッションの終了時間
   );
```

パラメータ

symbol_name

[in] カスタム銘柄名

ENUM_DAY_OF_WEEK

[in] ENUM_DAY_OF_WEEK列挙からの値である曜日

uint

[in] 開始時刻と終了時刻を設定するセッションのインデックス。セッションインデックスは0から始まります。

from

[in] セッション開始時間（00:00からの秒数）で、変数のデータ値は無視されます。

to

[in] セッション終了時間（00:00からの秒数）で、変数のデータ値は無視されます。

戻り値

成功の場合はtrue、それ以外の場合はfalse。エラー情報を取得するためにはGetLastError()関数を呼び出します。

注意事項

指定された*session_index*を持つセッションが既存する場合、この関数は単にセッションの開始と終了を編集します。

セッションの開始パラメータと終了パラメータにゼロが渡された場合（*from=0*と*to=0*）、適切な*session_index*を持つセッションが削除され、セッションインデックスが下向きにシフトされます。

セッションは順番にしか追加できません。言い換えれば*session_index=1*は、インデックスが0であるセッションが既存する場合にのみ追加できます。このルールが守られていないと、新しいセッションは作成されず、関数自体は「false」を返します。

例：

```
//+------------------------------------------------------------------+
//|                                  CustomSymbolSetSessionTrade.mq5 |
//|                                  Copyright 2024, MetaQuotes Ltd. |
```

© 2000–2025, MetaQuotes Ltd.

```
//|                                                 https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"           // カスタム銘柄名
#define   CUSTOM_SYMBOL_PATH     "Forex"                 // 銘柄が作成されるグループの名前
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()                // カスタム銘柄の基となる銘柄の名前

#define   SESSION_0_FROM         D'1970.01.01 00:15:00'  // session 0 start time
#define   SESSION_0_TO           D'1970.01.01 11:59:00'  // session 0 end time
#define   SESSION_1_FROM         D'1970.01.01 12:15:00'  // session 1 start time
#define   SESSION_1_TO           D'1970.01.01 23:59:00'  // session 1 end time

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- カスタム銘柄を作成するときにエラーコードを取得する
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH,
CUSTOM_SYMBOL_ORIGIN);

//--- エラーコードが0（銘柄の作成に成功）でも5304（銘柄がすでに作成されている）でもない場合は、終了する
   if(create!=0 && create!=5304)
      return;

//--- 基となる銘柄とセッションインデックスを含むヘッダーを出力し
//---月曜日から金曜日までの曜日ごとのループで、操作ログに各取引セッションの開始時刻と終了時刻を出力する
   for(int session=0; session<2; session++)
     {
      PrintFormat("Trade session %d of the '%s' symbol from which the custom '%s'
was created", session, CUSTOM_SYMBOL_ORIGIN, CUSTOM_SYMBOL_NAME);
      for(int day_of_week=MONDAY; day_of_week<SATURDAY; day_of_week++)
         SymbolInfoSessionQuotePrint(CUSTOM_SYMBOL_ORIGIN, (ENUM_DAY_OF_WEEK)
day_of_week, session);
     }

//--- 2つのセッションによるループ
   bool res=true;
   for(int session=0; session<2; session++)
     {
      datetime from = SESSION_0_FROM;
      datetime to   = SESSION_0_TO;
      if(session>0)
        {
         from = SESSION_1_FROM;
         to   = SESSION_1_TO;
```

© 2000-2025, MetaQuotes Ltd.

```
         }
      //--- 各曜日のカスタム銘柄の相場セッション時間を設定する
      ResetLastError();
      for(int day_of_week=MONDAY; day_of_week<SATURDAY; day_of_week++)
         res &=CustomSymbolSetSessionQuote(CUSTOM_SYMBOL_NAME, (ENUM_DAY_OF_WEEK)
day_of_week, session, from, to);
     }

//--- いずれかのセッションの設定時にエラーが発生した場合は、銘柄に適切なメッセージを表示する
   if(!res)
      Print("CustomSymbolSetSessionTrade() failed. Error ", GetLastError());

//--- カスタム銘柄とセッションインデックスを含むヘッダーを出力し
//---月曜日から金曜日までの曜日ごとのループで、操作ログに各取引セッションの開始時刻と終了時刻を出力する
   for(int session=0; session<2; session++)
     {
      PrintFormat("Trade session %d of a custom symbol '%s' based on '%s'", session,
CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_ORIGIN);
      for(int day_of_week=MONDAY; day_of_week<SATURDAY; day_of_week++)
         SymbolInfoSessionQuotePrint(CUSTOM_SYMBOL_NAME, (ENUM_DAY_OF_WEEK)
day_of_week, session);
     }

//--- チャートのコメントにスクリプト終了キーに関するヒントを表示する
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and
exit", CUSTOM_SYMBOL_NAME));
//--- EscキーまたはDel キーが押されるまで待機して無限ループを終了する
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- Del キーを押すと、作成したカスタム銘柄が削除される
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully",
CUSTOM_SYMBOL_NAME);
         break;
        }
     }
//--- 終了する前にチャートをクリアする
   Comment("");
   /*
   結果:
   Trade session 0 of the 'EURUSD' symbol from which the custom 'EURUSD.C' was
created
   - Monday     00:15 - 23:55
   - Tuesday    00:15 - 23:55
   - Wednesday  00:15 - 23:55
   - Thursday   00:15 - 23:55
   - Friday     00:15 - 23:55
```

© 2000-2025, MetaQuotes Ltd.

```
   Trade session 1 of the 'EURUSD' symbol from which the custom 'EURUSD.C' was
created
   - Monday     Session not set
   - Tuesday    Session not set
   - Wednesday  Session not set
   - Thursday   Session not set
   - Friday     Session not set
   Trade session 0 of a custom symbol 'EURUSD.C' based on 'EURUSD'
   - Monday     00:15 - 11:59
   - Tuesday    00:15 - 11:59
   - Wednesday  00:15 - 11:59
   - Thursday   00:15 - 11:59
   - Friday     00:15 - 11:59
   Trade session 1 of a custom symbol 'EURUSD.C' based on 'EURUSD'
   - Monday     12:15 - 23:59
   - Tuesday    12:15 - 23:59
   - Wednesday  12:15 - 23:59
   - Thursday   12:15 - 23:59
   - Friday     12:15 - 23:59
   */
  }
//+------------------------------------------------------------------+
//|  カスタム銘柄ルを作成し、エラーコードを返す                                  |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const
string symbol_origin=NULL)
  {
//--- カスタム銘柄の基となる銘柄の名前をする
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- カスタム銘柄の作成に失敗し、エラー5304ではない場合は、操作ログに報告する
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name,
symbol_path, origin, error);
     }
//--- 成功
   return(error);
  }
//+------------------------------------------------------------------+
//|  カスタム銘柄を削除する                                                |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- 気配値表示ウィンドウから銘柄を非表示にする
```

© 2000–2025, MetaQuotes Ltd.

```
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- カスタム銘柄の削除に失敗した場合は、操作ログにこれを報告してfalseを返す
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name,
GetLastError());
      return(false);
     }
//--- 成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指定されたシンボルと曜日の指定された相場セッションの                  |
//| 開始時刻と終了時刻を操作ログに送信する                              |
//+------------------------------------------------------------------+
void SymbolInfoSessionTradePrint(const string symbol, const ENUM_DAY_OF_WEEK
day_of_week, const uint session_index)
  {
//--- 相場セッションの開始と終了を記録する変数を宣言する
   datetime date_from;  // session start time
   datetime date_to;    // session end time

//--- 列挙定数から曜日名を作成する
   string week_day=EnumToString(day_of_week);
   if(week_day.Lower())
      week_day.SetChar(0, ushort(week_day.GetChar(0)-32));

//--- 銘柄と曜日別に相場セッションからデータを取得する
   if(!SymbolInfoSessionTrade(symbol, day_of_week, session_index, date_from,
date_to))
     {
      int err=GetLastError();
      string message=(err==4307 ? StringFormat("- %-10s Session not set",
week_day) :
                      StringFormat("SymbolInfoSessionTrade(%s, %s, session %d)
failed. Error %d", symbol, week_day, session_index, GetLastError()));
      Print(message);
      return;
     }

//--- 指定された取引セッションのデータを操作ログに送信する
   PrintFormat("- %-10s %s - %s", week_day, TimeToString(date_from, TIME_MINUTES),
TimeToString(date_to, TIME_MINUTES));
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

参照

SymbolInfoSessionTrade、Symbol info、TimeToStruct、Date structure

© 2000–2025, MetaQuotes Ltd.

CustomRatesDelete

指定した時間間隔内のカスタム銘柄の価格履歴からすべてのバーを削除します。

```
int  CustomRatesDelete(
   const string      symbol,        // 銘柄名
   datetime          from,          // 開始の日付
   datetime          to             // 終了の日付
   );
```

パラメータ

symbol

[in] カスタム銘柄名

from

[in] 削除する指定範囲内の価格履歴の最初のバーの時間

to

[in] 削除する指定範囲内の価格履歴の最後のバーの時間

戻り値

削除されたバーの数（エラーの場合は -1 ）

例：

```
//+------------------------------------------------------------------+
//|                                              CustomRatesDelete.mq5 |
//|                                 Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"      // カスタム銘柄名
#define   CUSTOM_SYMBOL_PATH     "Forex"            // 銘柄が作成されるグループの名前
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()           // カスタム銘柄の基となる銘柄の名前

#define   DATARATES_COUNT        4                  // 操作ログに送信されるバーの数

//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- カスタム銘柄を作成するときにエラーコードを取得する
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH,
CUSTOM_SYMBOL_ORIGIN);
```

© 2000–2025, MetaQuotes Ltd.

```
//--- エラーコードが0（銘柄の作成に成功）でも5304（銘柄がすでに作成されている）でもない場合は、終了する
   if(create!=0 && create!=5304)
      return;

//--- 標準銘柄バーの数を取得する
   int bars=Bars(CUSTOM_SYMBOL_ORIGIN, PERIOD_M1);

//--- 標準銘柄の分単位の時間枠のすべてのバーのデータをMql Rates配列に取得する
   MqlRates rates[]={};
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_ORIGIN, PERIOD_M1, 0, bars, rates)!=bars)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d",
CUSTOM_SYMBOL_ORIGIN, bars, GetLastError());
      return;
     }

//--- コピーしたデータをカスタム銘柄の分単位の履歴に設定する
   ResetLastError();
   if(CustomRatesUpdate(CUSTOM_SYMBOL_NAME, rates)<0)
     {
      PrintFormat("CustomRatesUpdate(%s) failed. Error %d", CUSTOM_SYMBOL_NAME,
GetLastError());
      return;
     }

//--- 履歴データを更新した後、カスタム銘柄バーの数を取得する
   bars=Bars(CUSTOM_SYMBOL_NAME, PERIOD_M1);

//--- カスタム銘柄の分単位の時間枠のすべてのバーのデータをMql Rates配列に取得する
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_NAME, PERIOD_M1, 0, bars, rates)!=bars)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d",
CUSTOM_SYMBOL_NAME, bars, GetLastError());
      return;
     }

//--- カスタム銘柄の分単位履歴の最後のDATARATES_COUNTバーを操作ログに出力する
   int digits=(int)SymbolInfoInteger(CUSTOM_SYMBOL_NAME, SYMBOL_DIGITS);
   PrintFormat("Last %d bars of the custom symbol's minute history:",
DATARATES_COUNT);
   ArrayPrint(rates, digits, NULL, bars-DATARATES_COUNT, DATARATES_COUNT);

//--- カスタム銘柄の分単位の履歴の最後から2つのデータバーを削除する
   datetime time_from= rates[bars-3].time;
   datetime time_to  = rates[bars-2].time;
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
   int deleted=CustomRatesDelete(CUSTOM_SYMBOL_NAME, time_from, time_to);
   if(deleted<0)
     {
      PrintFormat("CustomRatesDelete(%s) failed. Error %d", CUSTOM_SYMBOL_NAME,
GetLastError());
      return;
     }

//--- 履歴データの2つのバーを削除した後、カスタム銘柄バーの数を再度取得する
   bars=Bars(CUSTOM_SYMBOL_NAME, PERIOD_M1);

//--- カスタム銘柄の分単位の残りのバーのデータを再度取得する
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_NAME, PERIOD_M1, 0, bars, rates)!=bars)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d",
CUSTOM_SYMBOL_NAME, bars, GetLastError());
      return;
     }

//--- 更新されたカスタム銘柄の分単位履歴の最後のDATARATES_COUNTバーを操作ログに出力する
   PrintFormat("\nLast %d bars after applying CustomRatesDelete() with %d deleted
bars:", DATARATES_COUNT, deleted);
   ArrayPrint(rates, digits, NULL, bars-DATARATES_COUNT, DATARATES_COUNT);

//--- チャートのコメントにスクリプト終了キーに関するヒントを表示する
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and
exit", CUSTOM_SYMBOL_NAME));
//--- EscキーまたはDel キーが押されるまで待機して無限ループを終了する
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- Del キーを押すと、作成したカスタム銘柄とそのデータが削除される
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         //--- バーデータを削除する
         int deleted=CustomRatesDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history bars of the custom symbol '%s' were successfully
deleted", deleted, CUSTOM_SYMBOL_NAME);

         //--- ティックデータを削除する
         deleted=CustomTicksDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history ticks of the custom symbol '%s' were
successfully deleted", deleted, CUSTOM_SYMBOL_NAME);

         //--- 銘柄を削除する
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
```

© 2000-2025, MetaQuotes Ltd.

```
            PrintFormat("Custom symbol '%s' deleted successfully",
CUSTOM_SYMBOL_NAME);
         break;
        }
     }
//--- 終了する前にチャートをクリアする
   Comment("");
   /*
   結果:
   Last 4 bars of the custom symbol's minute history:
                    [time]  [open]  [high]   [low] [close] [tick_volume] [spread]
[real_volume]
   [0] 2024.06.18 20:53:00 1.07341 1.07347 1.07336 1.07343            38        0
         0
   [1] 2024.06.18 20:54:00 1.07344 1.07354 1.07344 1.07353            21        0
         0
   [2] 2024.06.18 20:55:00 1.07353 1.07362 1.07351 1.07356            32        0
         0
   [3] 2024.06.18 20:56:00 1.07356 1.07358 1.07352 1.07354            24        0
         0

   Last 4 bars after applying CustomRatesDelete() with 2 deleted bars:
                    [time]  [open]  [high]   [low] [close] [tick_volume] [spread]
[real_volume]
   [0] 2024.06.18 20:51:00 1.07357 1.07358 1.07347 1.07349            25        0
         0
   [1] 2024.06.18 20:52:00 1.07349 1.07350 1.07336 1.07341            31        0
         0
   [2] 2024.06.18 20:53:00 1.07341 1.07347 1.07336 1.07343            38        0
         0
   [3] 2024.06.18 20:56:00 1.07356 1.07358 1.07352 1.07354            24        0
         0
   */
  }
//+------------------------------------------------------------------+
//| カスタム銘柄ルを作成し、エラーコードを返す                                |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const
string symbol_origin=NULL)
  {
//--- カスタム銘柄の基となる銘柄の名前をする
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- カスタム銘柄の作成に失敗し、エラー5304ではない場合は、操作ログに報告する
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
```

© 2000-2025, MetaQuotes Ltd.

```
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name,
symbol_path, origin, error);
     }
//--- 成功
   return(error);
  }
//+------------------------------------------------------------------+
//| カスタム銘柄を削除する                                           |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- 気配値表示ウィンドウから銘柄を非表示にする
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- カスタム銘柄の削除に失敗した場合は、操作ログにこれを報告してfalseを返す
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name,
GetLastError());
      return(false);
     }
//--- 成功
   return(true);
  }
```

参照

CustomRatesReplace、CustomRatesUpdate、CopyRates

© 2000-2025, MetaQuotes Ltd.

CustomRatesReplace

指定された時間間隔内のカスタム銘柄の価格履歴をMqlRates型の配列からのデータで完全に置き換えます。

```
int  CustomRatesReplace(
   const string      symbol,             // 銘柄名
   datetime          from,               // 開始の日付
   datetime          to,                 // 終了の日付
   const MqlRates&   rates[],            // カスタム銘柄に適用されるデータの配列
   uint              count=WHOLE_ARRAY   // 使用されるrates[]配列要素の数
   );
```

パラメータ

symbol

[in] カスタム銘柄名

from

[in] 更新する指定範囲内の価格履歴の最初のバーの時間

to

[in] 更新する指定範囲内の価格履歴の最後のバーの時間

rates[]

[in] MqlRates型のM1履歴データの配列

count=WHOLE_ARRAY

[in] 置換に使用される*rates[]*配列要素の数。WHOLE_ARRAYはすべての*rates[]*配列要素が置き換えに使用されることを意味します。

戻り値

更新されたバーの数（エラーの場合は -1 ）

注意事項

指定された範囲を超えた*rates[]*配列のバーは無視されます。このようなバーがすでに価格履歴に存在して指定された範囲に入った場合は置き換えられます。現在の価格履歴の指定された範囲外のバーは変更されません。*rates[]*配列データはOHLC価格について正しく、バーが開く時間はM1時間枠に対応するべきです。

例：

```
//+------------------------------------------------------------------+
//|                                            CustomRatesReplace.mq5 |
//|                                  Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
```

© 2000-2025, MetaQuotes Ltd.

```
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"      // カスタム銘柄名
#define   CUSTOM_SYMBOL_PATH     "Forex"            // 銘柄が作成されるグループの名前
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()           // カスタム銘柄の基となる銘柄の名前

#define   DATARATES_COUNT        4                  // 操作ログに送信されるバーの数

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- カスタム銘柄を作成するときにエラーコードを取得する
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH,
CUSTOM_SYMBOL_ORIGIN);

//--- エラーコードが0（銘柄の作成に成功）でも5304（銘柄がすでに作成されている）でもない場合は、終了する
   if(create!=0 && create!=5304)
      return;

//--- 標準銘柄バーの数を取得する
   int bars=Bars(CUSTOM_SYMBOL_ORIGIN, PERIOD_M1);

//--- 標準銘柄の分単位の時間枠のすべてのバーのデータをMql Rates配列に取得する
   MqlRates rates[]={};
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_ORIGIN, PERIOD_M1, 0, bars, rates)!=bars)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d",
CUSTOM_SYMBOL_ORIGIN, bars, GetLastError());
      return;
     }

//--- コピーしたデータをカスタム銘柄の分単位の履歴に設定する
   ResetLastError();
   if(CustomRatesUpdate(CUSTOM_SYMBOL_NAME, rates)<0)
     {
      PrintFormat("CustomRatesUpdate(%s) failed. Error %d", CUSTOM_SYMBOL_NAME,
GetLastError());
      return;
     }

//--- 履歴データを更新した後、カスタム銘柄バーの数を取得する
   bars=Bars(CUSTOM_SYMBOL_NAME, PERIOD_M1);

//--- カスタム銘柄の分単位の時間枠のすべてのバーのデータをMql Rates配列に取得する
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_NAME, PERIOD_M1, 0, bars, rates)!=bars)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d",
CUSTOM_SYMBOL_NAME, bars, GetLastError());
       return;
      }

//--- カスタム銘柄の分単位履歴の最後のDATARATES_COUNTバーを操作ログに出力する
   int digits=(int)SymbolInfoInteger(CUSTOM_SYMBOL_NAME, SYMBOL_DIGITS);
   PrintFormat("Last %d bars of the custom symbol's minute history:",
DATARATES_COUNT);
   ArrayPrint(rates, digits, NULL, bars-DATARATES_COUNT, DATARATES_COUNT);

//--- カスタム銘柄の分単位の履歴の最後から2つのデータバーを変更する
   datetime time_from= rates[bars-3].time;
   datetime time_to  = rates[bars-2].time;

//--- 最後から2番目のバーのすべての価格をrates配列内のこれらのバーの始値と等しくする
   rates[bars-3].high=rates[bars-3].open;
   rates[bars-3].low=rates[bars-3].open;
   rates[bars-3].close=rates[bars-3].open;

   rates[bars-2].high=rates[bars-2].open;
   rates[bars-2].low=rates[bars-2].open;
   rates[bars-2].close=rates[bars-2].open;

//--- 既存のバーを変更されたrates配列のデータに置き換える
   ResetLastError();
   int replaced=CustomRatesUpdate(CUSTOM_SYMBOL_NAME, rates);
   if(replaced<0)
     {
      PrintFormat("CustomRatesUpdate(%s) failed. Error %d", CUSTOM_SYMBOL_NAME,
GetLastError());
      return;
     }

//--- 履歴データの2つのバーを変更した後、カスタム銘柄バーの数を再度取得する
   bars=Bars(CUSTOM_SYMBOL_NAME, PERIOD_M1);

//--- カスタム銘柄の分単位のすべてのバーのデータを再度取得する
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_NAME, PERIOD_M1, 0, bars, rates)!=bars)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d",
CUSTOM_SYMBOL_NAME, bars, GetLastError());
      return;
     }

//--- 更新されたカスタム銘柄の分単位履歴の最後のDATARATES_COUNTバーを操作ログに出力する
   PrintFormat("\nLast %d bars after applying CustomRatesUpdate() with %d replaced
bars:", DATARATES_COUNT, replaced);
```

© 2000-2025, MetaQuotes Ltd.

```
   ArrayPrint(rates, digits, NULL, bars-DATARATES_COUNT, DATARATES_COUNT);

//--- チャートのコメントにスクリプト終了キーに関するヒントを表示する
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and
exit", CUSTOM_SYMBOL_NAME));
//--- EscキーまたはDel キーが押されるまで待機して無限ループを終了する
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- Del キーを押すと、作成したカスタム銘柄とそのデータが削除される
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         //--- バーデータを削除する
         int deleted=CustomRatesDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history bars of the custom symbol '%s' were successfully
deleted", deleted, CUSTOM_SYMBOL_NAME);

         //--- ティックデータを削除する
         deleted=CustomTicksDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history ticks of the custom symbol '%s' were
successfully deleted", deleted, CUSTOM_SYMBOL_NAME);

         //--- 銘柄を削除する
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully",
CUSTOM_SYMBOL_NAME);
         break;
        }
     }
//--- 終了する前にチャートをクリアする
   Comment("");
   /*
   結果:
   Last 4 bars of the custom symbol's minute history:
                     [time]  [open]  [high]   [low] [close] [tick_volume] [spread]
[real_volume]
   [0] 2024.07.29 13:37:00 1.08394 1.08396 1.08388 1.08390            16        1
          0
   [1] 2024.07.29 13:38:00 1.08389 1.08400 1.08389 1.08398            35        1
          0
   [2] 2024.07.29 13:39:00 1.08398 1.08410 1.08394 1.08410            29        1
          0
   [3] 2024.07.29 13:40:00 1.08409 1.08414 1.08408 1.08414            14        1
          0

   Last 4 bars after applying CustomRatesUpdate() with 250820 replaced bars:
                     [time]  [open]  [high]   [low] [close] [tick_volume] [spread]
[real_volume]
```

© 2000-2025, MetaQuotes Ltd.

```
   [0] 2024.07.29 13:37:00 1.08394 1.08396 1.08388 1.08390            16        1
          0
   [1] 2024.07.29 13:38:00 1.08389 1.08389 1.08389 1.08389            35        1
          0
   [2] 2024.07.29 13:39:00 1.08398 1.08398 1.08398 1.08398            29        1
          0
   [3] 2024.07.29 13:40:00 1.08409 1.08414 1.08408 1.08414            14        1
          0
   */
  }
//+------------------------------------------------------------------+
//|  カスタム銘柄ルを作成し、エラーコードを返す                                    |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const
string symbol_origin=NULL)
  {
//--- カスタム銘柄の基となる銘柄の名前をする
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- カスタム銘柄の作成に失敗し、エラー5304ではない場合は、操作ログに報告する
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name,
symbol_path, origin, error);
     }
//--- 成功
   return(error);
  }
//+------------------------------------------------------------------+
//|  カスタム銘柄を削除する                                             |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- 気配値表示ウィンドウから銘柄を非表示にする
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- カスタム銘柄の削除に失敗した場合は、操作ログにこれを報告してfalseを返す
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
```

© 2000–2025, MetaQuotes Ltd.

```
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name,
GetLastError());
      return(false);
     }
//--- 成功
   return(true);
  }
```

参照

CustomRatesDelete、CustomRatesUpdate、CopyRates

© 2000–2025, MetaQuotes Ltd.

CustomRatesUpdate

カスタム銘柄履歴に欠けているバーを追加し、既存のデータをMqlRates型配列のものに置き換えます。

```
int  CustomRatesUpdate(
   const string      symbol,              // カスタム銘柄名
   const MqlRates&   rates[],             // カスタム銘柄に適用されるデータの配列
   uint              count=WHOLE_ARRAY    // 使用されるrates[]配列要素の数
   );
```

パラメータ

symbol

[in] カスタム銘柄名

rates[]

[in] MqlRates型のM1履歴データの配列

count=WHOLE_ARRAY

[in] 更新に使用される*rates[]*配列要素の数。WHOLE_ARRAYはすべての*rates[]*配列要素が使用されることを意味します。

戻り値

更新されたバーの数（エラーの場合は -1 ）

注意事項

rates[] 配列のバーが現在のカスタム銘柄の履歴にない場合は追加され、そのようなバーがすでに存在する場合は置き換えられます。現在の価格履歴の他のバーは変更されません。*rates[]*配列データはOHLC価格について正しく、バーが開く時間はM1時間枠に対応するべきです。

例：

```
//+------------------------------------------------------------------+
//|                                            CustomRatesUpdate.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd.   |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"     // カスタム銘柄名
#define   CUSTOM_SYMBOL_PATH     "Forex"           // 銘柄が作成されるグループの名前
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()          // カスタム銘柄の基となる銘柄の名前

#define   DATARATES_COUNT        4                 // 操作ログに送信されるバーの数

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- カスタム銘柄を作成するときにエラーコードを取得する
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH,
CUSTOM_SYMBOL_ORIGIN);

//--- エラーコードが0（銘柄の作成に成功）でも5304（銘柄がすでに作成されている）でもない場合は、終了する
   if(create!=0 && create!=5304)
      return;

//--- 標準銘柄バーの数を取得して操作ログに出力する
   int bars_origin=Bars(CUSTOM_SYMBOL_ORIGIN, PERIOD_M1);
   PrintFormat("The symbol '%s' from which the custom '%s' was created has %d bars
of minute history.", CUSTOM_SYMBOL_ORIGIN, CUSTOM_SYMBOL_NAME, bars_origin);

//--- カスタム銘柄バーの数を取得して操作ログに出力する
   int bars_custom=Bars(CUSTOM_SYMBOL_NAME, PERIOD_M1);
   PrintFormat("Custom symbol '%s' created from symbol '%s' has %d bars of minute
history", CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_ORIGIN, bars_custom);

//--- 標準銘柄の分単位の時間枠のすべてのバーのデータをMql Rates配列に取得する
   MqlRates rates[]={};
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_ORIGIN, PERIOD_M1, 0, bars_origin,
rates)!=bars_origin)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d",
CUSTOM_SYMBOL_ORIGIN, bars_origin, GetLastError());
      return;
     }

//--- コピーしたデータをカスタム銘柄の分単位の履歴に設定する
   ResetLastError();
   int updated=CustomRatesUpdate(CUSTOM_SYMBOL_NAME, rates);
   if(updated<0)
     {
      PrintFormat("CustomRatesUpdate(%s) failed. Error %d", CUSTOM_SYMBOL_NAME,
GetLastError());
      return;
     }

//--- 履歴を追加した後のカスタム銘柄バーの数を取得して操作ログに出力する
   bars_custom=Bars(CUSTOM_SYMBOL_NAME, PERIOD_M1);
   PrintFormat("\nAfter CustomRatesUpdate(), the custom symbol '%s' has %d bars of
minute history", CUSTOM_SYMBOL_NAME, bars_custom);

//--- カスタム銘柄の分単位の時間枠のすべてのバーのデータをMql Rates配列に取得する
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_NAME, PERIOD_M1, 0, bars_custom, rates)!=bars_custom)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d",
CUSTOM_SYMBOL_NAME, bars_custom, GetLastError());
       return;
      }

//--- カスタム銘柄の分単位の履歴の最後の4つのバーを操作ログに出力する
   int digits=(int)SymbolInfoInteger(CUSTOM_SYMBOL_NAME, SYMBOL_DIGITS);
   PrintFormat("Last %d bars of the custom symbol's minute history:",
DATARATES_COUNT);
   ArrayPrint(rates, digits, NULL, bars_custom-DATARATES_COUNT, DATARATES_COUNT);

//--- MqlRates配列のデータを、式1.0 / Symbol Nameを使用して計算されたデータに置き換える
   for(int i=0; i<bars_custom; i++)
     {
      rates[i].open  =(rates[i].open !=0  ? 1.0 / rates[i].open  : rates[i].open);
      rates[i].high  =(rates[i].high !=0  ? 1.0 / rates[i].high  : rates[i].high);
      rates[i].low   =(rates[i].low  !=0  ? 1.0 / rates[i].low   : rates[i].low);
      rates[i].close =(rates[i].close!=0  ? 1.0 / rates[i].close : rates[i].close);
     }

//--- 変更したデータをカスタム銘柄の分単位の履歴に設定する
   ResetLastError();
   updated=CustomRatesUpdate(CUSTOM_SYMBOL_NAME, rates);
   if(updated<0)
     {
      PrintFormat("CustomRatesUpdate(%s) failed. Error %d", CUSTOM_SYMBOL_NAME,
GetLastError());
      return;
     }

//--- カスタム銘柄の分単位の時間枠のすべてのバーのデータをMqlRates配列に再度取得する
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_NAME, PERIOD_M1, 0, bars_custom, rates)!=bars_custom)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d",
CUSTOM_SYMBOL_NAME, bars_custom, GetLastError());
      return;
     }

//--- 更新されたカスタム銘柄の分単位の履歴の最後の4つのバーを操作ログに出力する
   Print("\nLast %d bars after changing the custom symbol calculation formula:",
DATARATES_COUNT);
   ArrayPrint(rates, digits, NULL, bars_custom-DATARATES_COUNT, DATARATES_COUNT);

//--- チャートのコメントにスクリプト終了キーに関するヒントを表示する
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and
exit", CUSTOM_SYMBOL_NAME));
//--- EscキーまたはDelキーが押されるまで待機して無限ループを終了する
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Sleep(16);
       //--- Del キーを押すと、作成したカスタム銘柄とそのデータが削除される
       if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
         {
          //--- バーデータを削除する
          int deleted=CustomRatesDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
          if(deleted>0)
             PrintFormat("%d history bars of the custom symbol '%s' were successfully
deleted", deleted, CUSTOM_SYMBOL_NAME);

          //--- ティックデータを削除する
          deleted=CustomTicksDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
          if(deleted>0)
             PrintFormat("%d history ticks of the custom symbol '%s' were
successfully deleted", deleted, CUSTOM_SYMBOL_NAME);

          //--- 銘柄を削除する
          if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
             PrintFormat("Custom symbol '%s' deleted successfully",
CUSTOM_SYMBOL_NAME);
          break;
         }
      }
//--- 終了する前にチャートをクリアする
   Comment("");
   /*
   結果:
   The symbol 'EURUSD' from which the custom 'EURUSD.C' was created has 250488 bars
of minute history.
   Custom symbol 'EURUSD.C' created from symbol 'EURUSD' has 0 bars of minute
history

   After CustomRatesUpdate(), the custom symbol 'EURUSD.C' has 250488 bars of minute
 history
   Last 4 bars of the custom symbol's minute history:
                     [time]  [open]  [high]   [low] [close] [tick_volume] [spread]
[real_volume]
   [0] 2024.06.18 11:14:00 1.07235 1.07239 1.07232 1.07239            24        0
         0
   [1] 2024.06.18 11:15:00 1.07238 1.07239 1.07232 1.07235            44        0
         0
   [2] 2024.06.18 11:16:00 1.07234 1.07238 1.07227 1.07234            37        0
         0
   [3] 2024.06.18 11:17:00 1.07234 1.07234 1.07217 1.07225            41        0
         0

   Last 4 bars after changing the custom symbol calculation formula:
                     [time]  [open]  [high]   [low] [close] [tick_volume] [spread]
[real_volume]
```

© 2000-2025, MetaQuotes Ltd.

```
   [0] 2024.06.18 11:14:00 0.93253 0.93250 0.93256 0.93250            24        0
         0
   [1] 2024.06.18 11:15:00 0.93251 0.93250 0.93256 0.93253            44        0
         0
   [2] 2024.06.18 11:16:00 0.93254 0.93251 0.93260 0.93254            37        0
         0
   [3] 2024.06.18 11:17:00 0.93254 0.93254 0.93269 0.93262            41        0
         0
   */
  }
//+------------------------------------------------------------------+
//| カスタム銘柄ルを作成し、エラーコードを返す                                    |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const
string symbol_origin=NULL)
  {
//--- カスタム銘柄の基となる銘柄の名前をする
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- カスタム銘柄の作成に失敗し、エラー5304ではない場合は、操作ログに報告する
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name,
symbol_path, origin, error);
     }
//--- 成功
   return(error);
  }
//+------------------------------------------------------------------+
//| カスタム銘柄を削除する                                                |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- 気配値表示ウィンドウから銘柄を非表示にする
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- カスタム銘柄の削除に失敗した場合は、操作ログにこれを報告してfalseを返す
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
```

© 2000–2025, MetaQuotes Ltd.

```
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name,
GetLastError());
      return(false);
     }
//--- 成功
   return(true);
  }
```

参照

CustomRatesReplace、CustomRatesDelete、CopyRates

© 2000–2025, MetaQuotes Ltd.

CustomTicksAdd

カスタム銘柄の価格履歴にMqlTick型の配列からデータを追加します。カスタム銘柄は気配値ウィンドウで選択されなければなりません。

```
int  CustomTicksAdd(
   const string      symbol,              // 銘柄名
   const MqlTick&    ticks[],             // カスタム銘柄に適用されるティックデータの配列
   uint              count=WHOLE_ARRAY    // 使用されるticks[]配列要素の数
   );
```

パラメータ

symbol

[in] カスタム銘柄名

ticks[]

[in] [in]以前のデータから最近のものまでの時間順に配列されたMqlTick型のティックデータの配列、すなわちk<nの場合はticks[k].time_msc <= ticks[n].time_msc。

count=WHOLE_ARRAY

[in] 追加に使用される*ticks[]*配列要素の数。WHOLE_ARRAYはすべての*ticks[]*配列要素が使用されることを意味します。

戻り値

追加されたティックの数（エラーの場合は -1 ）

さらなる注記

CustomTicksAdd関数は、気配値ウィンドウで開かれているカスタム銘柄に対してのみ機能します。銘柄が気配値ウィンドウで選択されていない場合にはCustomTicksReplaceを使用してティックを追加するべきです。

MqlTick 構造体には、1970年1月1日からカウントされた時間（秒単位のティック時間）とtime_msc（ミリ秒単位のティック時間）の2つの時間値を持つフィールドがあります。追加されたティックのこれらのフィールドは、次の順序で処理されます。

1. ticks[k].time_msc!=0の場合、これはticks[k].timeフィールドへの書き入れに使われます。すなわちticks[k].time=ticks[k].time_msc/1000（整数除算）がtickに設定されます。
2. ticks[k].time_msc==0でticks[k].time!=0の場合、ミリ秒単位の時間は、1000を掛けることによって得られます。すなわちticks[k].time_msc=ticks[k].time*1000です。
3. ticks[k].time_msc==0でticks[k].time==0の場合、これらのフィールドにはCustomTicksAddが呼ばれたときの現在の取引サーバ時間がミリ秒まで書かれます。

ticks[k].bid, ticks[k].ask、ticks[k].lastまたはticks[k].volume の値が正の場合は、適切なフラグの組み合わせがticks[k].flags fieldに書かれます。

- TICK_FLAG_BID - ティックが買値を変更した
- TICK_FLAG_ASK - ティックが売値を変更した
- TICK_FLAG_LAST - ティックが直近の約定値を変更した
- TICK_FLAG_VOLUME - ティックが取引量を変更した

フィールドの値がゼロ以下の場合、対応するフラグはticks[k].flagsフィールドに書き込まれません。

© 2000-2025, MetaQuotes Ltd.

TICK_FLAG_BUYフラグとTICK_FLAG_SELLフラグはカスタム銘柄の履歴には追加されません。

例：

```
//+------------------------------------------------------------------+
//|                                                CustomTicksAdd.mq5 |
//|                                  Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"      // カスタム銘柄名
#define   CUSTOM_SYMBOL_PATH     "Forex"            // 銘柄が作成されるグループの名前
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()           // カスタム銘柄の基となる銘柄の名前

#define   DATATICKS_TO_COPY      UINT_MAX           // コピーされたティックの数
#define   DATATICKS_TO_PRINT     20                 // 操作ログに送信されるティックの数

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- カスタム銘柄を作成するときにエラーコードを取得する
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH,
CUSTOM_SYMBOL_ORIGIN);

//--- エラーコードが0（銘柄の作成に成功）でも5304（銘柄がすでに作成されている）でもない場合は、終了する
   if(create!=0 && create!=5304)
      return;

//--- 標準銘柄ティックデータをMqlTick配列に取得する
   MqlTick array[]={};
   if(!GetTicksToArray(CUSTOM_SYMBOL_ORIGIN, DATATICKS_TO_COPY, array))
      return;

//--- 標準銘柄の最初と最後に受信したティックの時間を出力する
   int total=(int)array.Size();
   PrintFormat("First tick time: %s.%03u, Last tick time: %s.%03u",
               TimeToString(array[0].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS),
array[0].time_msc%1000,
               TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|
TIME_SECONDS), array[total-1].time_msc%1000);

/--- 標準銘柄の最後のDATATICKS_TO_PRINTティックを操作ログに出力する
   PrintFormat("\nThe last %d ticks for the standard symbol '%s':",
DATATICKS_TO_PRINT, CUSTOM_SYMBOL_ORIGIN);
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
     {
      if(i<0)
         continue;
      PrintFormat("  %dth Tick: %s", i, GetTickDescription(array[i]));
     }

//--- 気配値表示ウィンドウにカスタム銘柄を追加する
   ResetLastError();
   if(!SymbolSelect(CUSTOM_SYMBOL_NAME, true))
     {
      Print("SymbolSelect() failed. Error ", GetLastError());
      return;
     }

//--- カスタム銘柄の価格履歴にティック配列データを追加する
   Print("...");
   uint start=GetTickCount();
   PrintFormat("Start of adding %u ticks to the history of the custom symbol '%s'",
array.Size(), CUSTOM_SYMBOL_NAME);
   int added=CustomTicksAdd(CUSTOM_SYMBOL_NAME, array);
   PrintFormat("Added %u ticks to the history of the custom symbol '%s' in %u ms",
added, CUSTOM_SYMBOL_NAME, GetTickCount()-start);

//--- カスタム銘柄のティックデータをMql Tick配列に取得する
   Print("...");
   if(!GetTicksToArray(CUSTOM_SYMBOL_NAME, array.Size(), array))
      return;

//--- カスタム銘柄の最初と最後に受信したティックの時間を出力する
   total=(int)array.Size();
   PrintFormat("First tick time: %s.%03u, Last tick time: %s.%03u",
               TimeToString(array[0].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS),
array[0].time_msc%1000,
               TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|
TIME_SECONDS), array[total-1].time_msc%1000);

//--- カスタム銘柄の最後のDATATICKS_TO_PRINTティックを操作ログに出力する
   PrintFormat("\nThe last %d ticks for the custom symbol '%s':",
DATATICKS_TO_PRINT, CUSTOM_SYMBOL_NAME);
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
     {
      if(i<0)
         continue;
      PrintFormat("  %dth Tick: %s", i, GetTickDescription(array[i]));
     }

//--- チャートのコメントにスクリプト終了キーに関するヒントを表示する
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and
exit", CUSTOM_SYMBOL_NAME));
```

© 2000–2025, MetaQuotes Ltd.

```
//--- EscキーまたはDel キーが押されるまで待機して無限ループを終了する
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- Del キーを押すと、作成したカスタム銘柄とそのデータが削除される
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         //--- バーデータを削除する
         int deleted=CustomRatesDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history bars of the custom symbol '%s' were successfully deleted", deleted, CUSTOM_SYMBOL_NAME);

         //--- ティックデータを削除する
         deleted=CustomTicksDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history ticks of the custom symbol '%s' were successfully deleted", deleted, CUSTOM_SYMBOL_NAME);

         //--- 銘柄を削除する
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME);
         break;
        }
     }
//--- 終了する前にチャートをクリアする
   Comment("");
   /*
   結果:
   Requested 4294967295 ticks to download tick history for the symbol 'EURUSD'
   The tick history for the 'EURUSD' symbol is received in the amount of 351183943 ticks in 56454 ms
   First tick time: 2011.12.19 00:00:08.000, Last tick time: 2024.06.20 21:18:12.010

   The last 20 ticks for the standard symbol 'EURUSD':
     351183923th Tick: 2024.06.20 21:17:46.380 Bid=1.07124 (Info tick)
     351183924th Tick: 2024.06.20 21:17:47.779 Ask=1.07125 Bid=1.07125 (Info tick)
     351183925th Tick: 2024.06.20 21:17:48.584 Ask=1.07124 Bid=1.07124 (Info tick)
     351183926th Tick: 2024.06.20 21:17:49.481 Ask=1.07125 (Info tick)
     351183927th Tick: 2024.06.20 21:17:49.985 Ask=1.07122 Bid=1.07122 (Info tick)
     351183928th Tick: 2024.06.20 21:17:50.482 Ask=1.07124 Bid=1.07124 (Info tick)
     351183929th Tick: 2024.06.20 21:17:51.584 Ask=1.07123 Bid=1.07123 (Info tick)
     351183930th Tick: 2024.06.20 21:17:52.786 Ask=1.07124 Bid=1.07124 (Info tick)
     351183931th Tick: 2024.06.20 21:17:53.487 Ask=1.07125 Bid=1.07125 (Info tick)
     351183932th Tick: 2024.06.20 21:17:53.989 Ask=1.07126 Bid=1.07126 (Info tick)
     351183933th Tick: 2024.06.20 21:17:55.789 Ask=1.07125 Bid=1.07125 (Info tick)
     351183934th Tick: 2024.06.20 21:17:58.495 Ask=1.07126 Bid=1.07126 (Info tick)
     351183935th Tick: 2024.06.20 21:18:00.102 Bid=1.07126 (Info tick)
```

© 2000–2025, MetaQuotes Ltd.

```
     351183936th Tick: 2024.06.20 21:18:00.698 Ask=1.07129 Bid=1.07129 (Info tick)
     351183937th Tick: 2024.06.20 21:18:03.699 Bid=1.07129 (Info tick)
     351183938th Tick: 2024.06.20 21:18:04.699 Ask=1.07128 Bid=1.07128 (Info tick)
     351183939th Tick: 2024.06.20 21:18:05.901 Ask=1.07129 Bid=1.07129 (Info tick)
     351183940th Tick: 2024.06.20 21:18:07.606 Ask=1.07128 Bid=1.07128 (Info tick)
     351183941th Tick: 2024.06.20 21:18:11.512 Ask=1.07127 Bid=1.07127 (Info tick)
     351183942th Tick: 2024.06.20 21:18:12.010 Ask=1.07126 Bid=1.07126 (Info tick)
   ...
   Start of adding 351183943 ticks to the history of the custom symbol 'EURUSD.C'
   Added 351183943 ticks to the history of the custom symbol 'EURUSD.C' in 269890 ms
   ...
   Requested 351183943 ticks to download tick history for the symbol 'EURUSD.C'
   The tick history for the 'EURUSD.C' symbol is received in the amount of 351183943
 ticks in 116407 ms
   First tick time: 2011.12.19 00:00:08.000, Last tick time: 2024.06.20 21:18:12.010

   The last 20 ticks for the custom symbol 'EURUSD.C':
     351183923th Tick: 2024.06.20 21:17:46.380 Ask=1.07124 Bid=1.07124 (Info tick)
     351183924th Tick: 2024.06.20 21:17:47.779 Ask=1.07125 Bid=1.07125 (Info tick)
     351183925th Tick: 2024.06.20 21:17:48.584 Ask=1.07124 Bid=1.07124 (Info tick)
     351183926th Tick: 2024.06.20 21:17:49.481 Ask=1.07125 Bid=1.07125 (Info tick)
     351183927th Tick: 2024.06.20 21:17:49.985 Ask=1.07122 Bid=1.07122 (Info tick)
     351183928th Tick: 2024.06.20 21:17:50.482 Ask=1.07124 Bid=1.07124 (Info tick)
     351183929th Tick: 2024.06.20 21:17:51.584 Ask=1.07123 Bid=1.07123 (Info tick)
     351183930th Tick: 2024.06.20 21:17:52.786 Ask=1.07124 Bid=1.07124 (Info tick)
     351183931th Tick: 2024.06.20 21:17:53.487 Ask=1.07125 Bid=1.07125 (Info tick)
     351183932th Tick: 2024.06.20 21:17:53.989 Ask=1.07126 Bid=1.07126 (Info tick)
     351183933th Tick: 2024.06.20 21:17:55.789 Ask=1.07125 Bid=1.07125 (Info tick)
     351183934th Tick: 2024.06.20 21:17:58.495 Ask=1.07126 Bid=1.07126 (Info tick)
     351183935th Tick: 2024.06.20 21:18:00.102 Ask=1.07126 Bid=1.07126 (Info tick)
     351183936th Tick: 2024.06.20 21:18:00.698 Ask=1.07129 Bid=1.07129 (Info tick)
     351183937th Tick: 2024.06.20 21:18:03.699 Ask=1.07129 Bid=1.07129 (Info tick)
     351183938th Tick: 2024.06.20 21:18:04.699 Ask=1.07128 Bid=1.07128 (Info tick)
     351183939th Tick: 2024.06.20 21:18:05.901 Ask=1.07129 Bid=1.07129 (Info tick)
     351183940th Tick: 2024.06.20 21:18:07.606 Ask=1.07128 Bid=1.07128 (Info tick)
     351183941th Tick: 2024.06.20 21:18:11.512 Ask=1.07127 Bid=1.07127 (Info tick)
     351183942th Tick: 2024.06.20 21:18:12.010 Ask=1.07126 Bid=1.07126 (Info tick)
   */
  }
//+------------------------------------------------------------------+
//|  カスタム銘柄ルを作成し、エラーコードを返す                                    |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const
string symbol_origin=NULL)
  {
//--- カスタム銘柄の基となる銘柄の名前をする
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- カスタム銘柄の作成に失敗し、エラー5304ではない場合は、操作ログに報告する
```

© 2000–2025, MetaQuotes Ltd.

```
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name,
symbol_path, origin, error);
     }
//--- 成功
   return(error);
  }
//+------------------------------------------------------------------+
//| カスタム銘柄を削除する                                           |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- 気配値表示ウィンドウから銘柄を非表示にする
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- カスタム銘柄の削除に失敗した場合は、操作ログにこれを報告してfalseを返す
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name,
GetLastError());
      return(false);
     }
//--- 成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 配列内の指定された数のティックを取得する                         |
//+------------------------------------------------------------------+
bool GetTicksToArray(const string symbol, const uint count, MqlTick &array[])
  {
//--- 履歴データの読み込み開始を通知する
   PrintFormat("Requested %u ticks to get tick history for the symbol '%s'", count,
symbol);

//--- ティックの受信を3回試行する
   int attempts=0;
   while(attempts<3)
     {
```

© 2000–2025, MetaQuotes Ltd.

```
      //--- ティックを受信する前に開始時間を測定する
      uint start=GetTickCount();

      //--- 1970.01.01 00:00.001 以降のティック履歴を要求する（パラメータ from=1 ms）
      int received=CopyTicks(symbol, array, COPY_TICKS_ALL, 1, count);
      if(received!=-1)
        {
         //--- ティック数と経過時間に関する情報を表示する
         PrintFormat("The tick history for the '%s' symbol is received in the amount
of %u ticks in %d ms", symbol, received, GetTickCount()-start);

         //--- ティック履歴が同期されている場合、エラーコードはゼロになる - trueを返す
         if(GetLastError()==0)
            return(true);

         PrintFormat("%s: Ticks are not synchronized yet, %d ticks received for %d
ms. Error=%d",
                     symbol, received, GetTickCount()-start, GetLastError());
        }
      //--- 試行回数を数える
      attempts++;
      //--- ティック データベースの同期が終了するまで1秒間待機する
      Sleep(1000);
     }
//--- 3 回の試行でティックをコピーできない
   return(false);
  }
//+------------------------------------------------------------------+
//| ティックの文字列の説明を返す                                     |
//+------------------------------------------------------------------+
string GetTickDescription(MqlTick &tick)
  {
   string desc=StringFormat("%s.%03u ", TimeToString(tick.time, TIME_DATE|
TIME_MINUTES|TIME_SECONDS),tick.time_msc%1000);

//--- ティックのフラグを確認する
   bool buy_tick   = ((tick.flags &TICK_FLAG_BUY)    == TICK_FLAG_BUY);
   bool sell_tick  = ((tick.flags &TICK_FLAG_SELL)   == TICK_FLAG_SELL);
   bool ask_tick   = ((tick.flags &TICK_FLAG_ASK)    == TICK_FLAG_ASK);
   bool bid_tick   = ((tick.flags &TICK_FLAG_BID)    == TICK_FLAG_BID);
   bool last_tick  = ((tick.flags &TICK_FLAG_LAST)   == TICK_FLAG_LAST);
   bool volume_tick= ((tick.flags &TICK_FLAG_VOLUME)== TICK_FLAG_VOLUME);

//--- まず取引フラグのティックを確認する（CustomTicksAdd()には何も存在しない）
   if(buy_tick || sell_tick)
     {
      //--- 取引ティックの出力を作成する
      desc += (buy_tick ? StringFormat("Buy Tick: Last=%G Volume=%d ", tick.last,
tick.volume)  : "");
```

© 2000-2025, MetaQuotes Ltd.

```
      desc += (sell_tick? StringFormat("Sell Tick: Last=%G Volume=%d ",tick.last,
tick.volume) : "");
      desc += (ask_tick ? StringFormat("Ask=%G ", tick.ask) : "");
      desc += (bid_tick ? StringFormat("Bid=%G ", tick.ask) : "");
      desc += "(Trade tick)";
     }
   else
     {
      //--- 情報ティックの出力を少し異なる方法で作成する
      desc += (ask_tick   ? StringFormat("Ask=%G ",  tick.ask)    : "");
      desc += (bid_tick   ? StringFormat("Bid=%G ",  tick.ask)    : "");
      desc += (last_tick  ? StringFormat("Last=%G ", tick.last)   : "");
      desc += (volume_tick? StringFormat("Volume=%d ",tick.volume): "");
      desc += "(Info tick)";
     }
//--- ティックの説明を返す
   return(desc);
  }
```

参照

CustomRatesDelete、CustomRatesUpdate、CustomTicksReplace、CopyTicks、CopyTicksRange

© 2000–2025, MetaQuotes Ltd.

CustomTicksDelete

指定した時間間隔内のカスタム銘柄の価格履歴からすべてのティックを削除します。

```
int  CustomTicksDelete(
   const string      symbol,           // 銘柄名
   long              from_msc,         // 開始の日付
   long              to_msc            // 終了の日付
   );
```

パラメータ

symbol

[in] カスタム銘柄名

from_msc

[in] 削除する指定範囲内の価格履歴の最初のティックの時間（1970.01.01 から経過したミリ秒）

to_msc

[in] 削除する指定範囲内の価格履歴の最後のティックの時間（1970.01.01 から経過したミリ秒）

戻り値

削除されたティックの数（エラーの場合は -1 ）

例：

```
//+------------------------------------------------------------------+
//|                                              CustomTicksDelete.mq5 |
//|                                  Copyright 2024, MetaQuotes Ltd. |
//|                                           https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"     // カスタム銘柄名
#define   CUSTOM_SYMBOL_PATH     "Forex"           // 銘柄が作成されるグループの名前
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()          // カスタム銘柄の基となる銘柄の名前

#define   DATATICKS_TO_COPY      UINT_MAX          // コピーされたティックの数
#define   DATATICKS_TO_DELETE    10                // number of deleted ticks
#define   DATATICKS_TO_PRINT     20                // 操作ログに送信されるティックの数

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- カスタム銘柄を作成するときにエラーコードを取得する
```

© 2000–2025, MetaQuotes Ltd.

```
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH,
CUSTOM_SYMBOL_ORIGIN);

//--- エラーコードが0（銘柄の作成に成功）でも5304（銘柄がすでに作成されている）でもない場合は、終了する
   if(create!=0 && create!=5304)
      return;

//--- 標準銘柄ティックデータをMql Ti ck配列に取得する
   MqlTick array[]={};
   if(!GetTicksToArray(CUSTOM_SYMBOL_ORIGIN, DATATICKS_TO_COPY, array))
      return;

//--- 標準銘柄の最初と最後に受信したティックの時間を出力する
   int total=(int)array.Size();
   PrintFormat("First tick time: %s.%03u, Last tick time: %s.%03u",
               TimeToString(array[0].time,TIME_DATE|TIME_MINUTES|TIME_SECONDS),
array[0].time_msc%1000,
               TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|
TIME_SECONDS), array[total-1].time_msc%1000);

/--- 標準銘柄の最後のDATATI CKS_ TO_ PRI NTティックを操作ログに出力する
   PrintFormat("\nThe last %d ticks for the standard symbol '%s':",
DATATICKS_TO_PRINT, CUSTOM_SYMBOL_ORIGIN);
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
     {
      if(i<0)
         continue;
      PrintFormat("  %dth Tick: %s", i, GetTickDescription(array[i]));
     }

//--- 気配値表示ウィンドウにカスタム銘柄を追加する
   ResetLastError();
   if(!SymbolSelect(CUSTOM_SYMBOL_NAME, true))
     {
      Print("SymbolSelect() failed. Error ", GetLastError());
      return;
     }

//--- カスタム銘柄の価格履歴にティック配列データを追加する
   Print("...");
   uint start=GetTickCount();
   PrintFormat("Start of adding %u ticks to the history of the custom symbol '%s'",
array.Size(), CUSTOM_SYMBOL_NAME);
   int added=CustomTicksAdd(CUSTOM_SYMBOL_NAME, array);
   PrintFormat("Added %u ticks to the history of the custom symbol '%s' in %u ms",
added, CUSTOM_SYMBOL_NAME, GetTickCount()-start);

//--- get the newly added custom symbol tick data to the MqlTick array
   Print("...");
   if(!GetTicksToArray(CUSTOM_SYMBOL_NAME, array.Size(), array))
```

© 2000-2025, MetaQuotes Ltd.

```
      return;

//--- カスタム銘柄の最初と最後に受信したティックの時間を出力する
   total=(int)array.Size();
   PrintFormat("First tick time: %s.%03u, Last tick time: %s.%03u",
               TimeToString(array[0].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS),
array[0].time_msc%1000,
               TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|
TIME_SECONDS), array[total-1].time_msc%1000);

//--- カスタム銘柄の最後のDATATICKS_TO_PRINTティックを操作ログに出力する
   PrintFormat("\nThe last %d ticks for the custom symbol '%s':",
DATATICKS_TO_PRINT, CUSTOM_SYMBOL_NAME);
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
     {
      if(i<0)
         continue;
      PrintFormat("  %dth Tick: %s", i, GetTickDescription(array[i]));
     }

//--- 履歴からティックの範囲を削除するティック時間をミリ秒単位で取得する
   long time_from=array[total-DATATICKS_TO_DELETE-1].time_msc;

//--- 配列内のカスタム銘柄の最後のティック範囲をDATATICKS_TO_DELETEで削除する
   Print("...");
   start=GetTickCount();
   PrintFormat("Start deleting %u ticks in the history of the custom symbol '%s'",
DATATICKS_TO_DELETE, CUSTOM_SYMBOL_NAME);
   int deleted=CustomTicksDelete(CUSTOM_SYMBOL_NAME, time_from, array[total-
2].time_msc);
   PrintFormat("Deleted %u ticks in the history of the custom symbol '%s' in %u ms",
deleted, CUSTOM_SYMBOL_NAME, GetTickCount()-start);

//--- 新しく変更されたカスタムシンボルのティックデータをMqlTick配列に取得する
   Print("...");
   if(!GetTicksToArray(CUSTOM_SYMBOL_NAME, array.Size(), array))
      return;

//--- ティック範囲を削除したカスタム 銘柄の最初と最後のティックの時間を出力する
   total=(int)array.Size();
   PrintFormat("Time of the first tick from the changed history: %s.%03u, Time of
the last tick from the changed history: %s.%03u",
               TimeToString(array[0].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS),
array[0].time_msc%1000,
               TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|
TIME_SECONDS), array[total-1].time_msc%1000);

//--- カスタム銘柄の最後のDATATICKS_TO_PRINTティックを操作ログに出力する
   PrintFormat("\nThe last %d ticks of custom symbol '%s' with modified history:",
DATATICKS_TO_PRINT, CUSTOM_SYMBOL_NAME);
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       if(i<0)
          continue;
       PrintFormat("  %dth Tick: %s", i, GetTickDescription(array[i]));
      }

//--- チャートのコメントにスクリプト終了キーに関するヒントを表示する
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and exit", CUSTOM_SYMBOL_NAME));
//--- EscキーまたはDel キーが押されるまで待機して無限ループを終了する
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- Del キーを押すと、作成したカスタム銘柄とそのデータが削除される
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         //--- バーデータを削除する
         int deleted=CustomRatesDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history bars of the custom symbol '%s' were successfully deleted", deleted, CUSTOM_SYMBOL_NAME);

         //--- ティックデータを削除する
         deleted=CustomTicksDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history ticks of the custom symbol '%s' were successfully deleted", deleted, CUSTOM_SYMBOL_NAME);

         //--- 銘柄を削除する
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME);
         break;
        }
     }
//--- 終了する前にチャートをクリアする
   Comment("");
   /*
   結果:
   Requested 4294967295 ticks to get tick history for the symbol 'EURUSD'
   The tick history for the 'EURUSD' symbol is received in the amount of 351199027 ticks in 55875 ms
   First tick time: 2011.12.19 00:00:08.000, Last tick time: 2024.06.21 10:10:40.392

   The last 20 ticks for the standard symbol 'EURUSD':
     351199007th Tick: 2024.06.21 10:10:23.045 Bid=1.07032 (Info tick)
     351199008th Tick: 2024.06.21 10:10:24.045 Ask=1.07031 Bid=1.07031 (Info tick)
     351199009th Tick: 2024.06.21 10:10:24.545 Ask=1.07032 (Info tick)
     351199010th Tick: 2024.06.21 10:10:25.146 Bid=1.07032 (Info tick)
     351199011th Tick: 2024.06.21 10:10:25.649 Ask=1.07037 Bid=1.07037 (Info tick)
```

© 2000–2025, MetaQuotes Ltd.

```
    351199012th Tick: 2024.06.21 10:10:27.050 Ask=1.07036 Bid=1.07036 (Info tick)
    351199013th Tick: 2024.06.21 10:10:28.153 Ask=1.07039 Bid=1.07039 (Info tick)
    351199014th Tick: 2024.06.21 10:10:29.157 Ask=1.07037 Bid=1.07037 (Info tick)
    351199015th Tick: 2024.06.21 10:10:29.658 Ask=1.07036 Bid=1.07036 (Info tick)
    351199016th Tick: 2024.06.21 10:10:30.258 Bid=1.07036 (Info tick)
    351199017th Tick: 2024.06.21 10:10:30.872 Ask=1.07035 Bid=1.07035 (Info tick)
    351199018th Tick: 2024.06.21 10:10:31.358 Ask=1.07036 (Info tick)
    351199019th Tick: 2024.06.21 10:10:31.859 Ask=1.07037 Bid=1.07037 (Info tick)
    351199020th Tick: 2024.06.21 10:10:32.377 Ask=1.07039 Bid=1.07039 (Info tick)
    351199021th Tick: 2024.06.21 10:10:32.962 Ask=1.0704 Bid=1.0704 (Info tick)
    351199022th Tick: 2024.06.21 10:10:33.961 Ask=1.07039 Bid=1.07039 (Info tick)
    351199023th Tick: 2024.06.21 10:10:34.667 Ask=1.0704 (Info tick)
    351199024th Tick: 2024.06.21 10:10:35.170 Bid=1.0704 (Info tick)
    351199025th Tick: 2024.06.21 10:10:38.266 Ask=1.07041 Bid=1.07041 (Info tick)
    351199026th Tick: 2024.06.21 10:10:40.392 Ask=1.07042 Bid=1.07042 (Info tick)
  ...
  Start of adding 351199027 ticks to the history of the custom symbol 'EURUSD.C'
  Added 351199027 ticks to the history of the custom symbol 'EURUSD.C' in 261594 ms
  ...
  Requested 351199027 ticks to get tick history for the symbol 'EURUSD.C'
  The tick history for the 'EURUSD.C' symbol is received in the amount of 351199027
ticks in 137156 ms
  First tick time: 2011.12.19 00:00:08.000, Last tick time: 2024.06.21 10:10:40.392

  The last 20 ticks for the custom symbol 'EURUSD.C':
    351199007th Tick: 2024.06.21 10:10:23.045 Ask=1.07032 Bid=1.07032 (Info tick)
    351199008th Tick: 2024.06.21 10:10:24.045 Ask=1.07031 Bid=1.07031 (Info tick)
    351199009th Tick: 2024.06.21 10:10:24.545 Ask=1.07032 Bid=1.07032 (Info tick)
    351199010th Tick: 2024.06.21 10:10:25.146 Ask=1.07032 Bid=1.07032 (Info tick)
    351199011th Tick: 2024.06.21 10:10:25.649 Ask=1.07037 Bid=1.07037 (Info tick)
    351199012th Tick: 2024.06.21 10:10:27.050 Ask=1.07036 Bid=1.07036 (Info tick)
    351199013th Tick: 2024.06.21 10:10:28.153 Ask=1.07039 Bid=1.07039 (Info tick)
    351199014th Tick: 2024.06.21 10:10:29.157 Ask=1.07037 Bid=1.07037 (Info tick)
    351199015th Tick: 2024.06.21 10:10:29.658 Ask=1.07036 Bid=1.07036 (Info tick)
    351199016th Tick: 2024.06.21 10:10:30.258 Ask=1.07036 Bid=1.07036 (Info tick)
    351199017th Tick: 2024.06.21 10:10:30.872 Ask=1.07035 Bid=1.07035 (Info tick)
    351199018th Tick: 2024.06.21 10:10:31.358 Ask=1.07036 Bid=1.07036 (Info tick)
    351199019th Tick: 2024.06.21 10:10:31.859 Ask=1.07037 Bid=1.07037 (Info tick)
    351199020th Tick: 2024.06.21 10:10:32.377 Ask=1.07039 Bid=1.07039 (Info tick)
    351199021th Tick: 2024.06.21 10:10:32.962 Ask=1.0704 Bid=1.0704 (Info tick)
    351199022th Tick: 2024.06.21 10:10:33.961 Ask=1.07039 Bid=1.07039 (Info tick)
    351199023th Tick: 2024.06.21 10:10:34.667 Ask=1.0704 Bid=1.0704 (Info tick)
    351199024th Tick: 2024.06.21 10:10:35.170 Ask=1.0704 Bid=1.0704 (Info tick)
    351199025th Tick: 2024.06.21 10:10:38.266 Ask=1.07041 Bid=1.07041 (Info tick)
    351199026th Tick: 2024.06.21 10:10:40.392 Ask=1.07042 Bid=1.07042 (Info tick)
  ...
  Start deleting 10 ticks in the history of the custom symbol 'EURUSD.C'
  Deleted 10 ticks in the history of the custom symbol 'EURUSD.C' in 188 ms
  ...
```

© 2000–2025, MetaQuotes Ltd.

```
   Requested 351199027 ticks to get tick history for the symbol 'EURUSD.C'
   The tick history for the 'EURUSD.C' symbol is received in the amount of 351199017
 ticks in 138312 ms
   Time of the first tick from the changed history: 2011.12.19 00:00:08.000, Time of
 the last tick from the changed history: 2024.06.21 10:10:40.392

   The last 20 ticks of custom symbol 'EURUSD.C' with modified history:
     351198997th Tick: 2024.06.21 10:10:14.935 Ask=1.07036 Bid=1.07036 (Info tick)
     351198998th Tick: 2024.06.21 10:10:15.533 Ask=1.07035 Bid=1.07035 (Info tick)
     351198999th Tick: 2024.06.21 10:10:17.736 Ask=1.07036 Bid=1.07036 (Info tick)
     351199000th Tick: 2024.06.21 10:10:18.540 Ask=1.07037 Bid=1.07037 (Info tick)
     351199001th Tick: 2024.06.21 10:10:19.046 Ask=1.07038 Bid=1.07038 (Info tick)
     351199002th Tick: 2024.06.21 10:10:19.542 Ask=1.07036 Bid=1.07036 (Info tick)
     351199003th Tick: 2024.06.21 10:10:20.041 Ask=1.07035 Bid=1.07035 (Info tick)
     351199004th Tick: 2024.06.21 10:10:21.041 Ask=1.07035 Bid=1.07035 (Info tick)
     351199005th Tick: 2024.06.21 10:10:21.544 Ask=1.07032 Bid=1.07032 (Info tick)
     351199006th Tick: 2024.06.21 10:10:22.344 Ask=1.07032 Bid=1.07032 (Info tick)
     351199007th Tick: 2024.06.21 10:10:23.045 Ask=1.07032 Bid=1.07032 (Info tick)
     351199008th Tick: 2024.06.21 10:10:24.045 Ask=1.07031 Bid=1.07031 (Info tick)
     351199009th Tick: 2024.06.21 10:10:24.545 Ask=1.07032 Bid=1.07032 (Info tick)
     351199010th Tick: 2024.06.21 10:10:25.146 Ask=1.07032 Bid=1.07032 (Info tick)
     351199011th Tick: 2024.06.21 10:10:25.649 Ask=1.07037 Bid=1.07037 (Info tick)
     351199012th Tick: 2024.06.21 10:10:27.050 Ask=1.07036 Bid=1.07036 (Info tick)
     351199013th Tick: 2024.06.21 10:10:28.153 Ask=1.07039 Bid=1.07039 (Info tick)
     351199014th Tick: 2024.06.21 10:10:29.157 Ask=1.07037 Bid=1.07037 (Info tick)
     351199015th Tick: 2024.06.21 10:10:29.658 Ask=1.07036 Bid=1.07036 (Info tick)
     351199016th Tick: 2024.06.21 10:10:40.392 Ask=1.07042 Bid=1.07042 (Info tick)
   */
  }
//+------------------------------------------------------------------+
//|  カスタム銘柄ルを作成し、エラーコードを返す                                          |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const
string symbol_origin=NULL)
  {
//--- カスタム銘柄の基となる銘柄の名前をする
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- カスタム銘柄の作成に失敗し、エラー5304ではない場合は、操作ログに報告する
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name,
symbol_path, origin, error);
     }
//--- 成功
```

© 2000–2025, MetaQuotes Ltd.

```
   return(error);
  }
//+------------------------------------------------------------------+
//| カスタム銘柄を削除する                                           |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- 気配値表示ウィンドウから銘柄を非表示にする
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- カスタム銘柄の削除に失敗した場合は、操作ログにこれを報告してfalseを返す
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name,
GetLastError());
      return(false);
     }
//--- 成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 配列内の指定された数のティックを取得する                         |
//+------------------------------------------------------------------+
bool GetTicksToArray(const string symbol, const uint count, MqlTick &array[])
  {
//--- 履歴データの読み込み開始を通知する
   PrintFormat("Requested %u ticks to get tick history for the symbol '%s'", count,
symbol);

//--- ティックの受信を3回試行する
   int attempts=0;
   while(attempts<3)
     {
      //--- ティックを受信する前に開始時間を測定する
      uint start=GetTickCount();

      //--- 1970.01.01 00:00.001 以降のティック履歴を要求する（パラメータ from=1 ms）
      int received=CopyTicks(symbol, array, COPY_TICKS_ALL, 1, count);
      if(received!=-1)
        {
         //--- ティック数と経過時間に関する情報を表示する
         PrintFormat("The tick history for the '%s' symbol is received in the amount
of %u ticks in %d ms", symbol, received, GetTickCount()-start);
```

© 2000-2025, MetaQuotes Ltd.

```
         //--- ティック履歴が同期されている場合、エラーコードはゼロになる - trueを返す
         if(GetLastError()==0)
            return(true);

         PrintFormat("%s: Ticks are not synchronized yet, %d ticks received for %d
ms. Error=%d",
                     symbol, received, GetTickCount()-start, GetLastError());
        }
      //--- 試行回数を数える
      attempts++;
      //--- ティック データベースの同期が終了するまで1秒間待機する
      Sleep(1000);
     }
//--- 3 回の試行でティックをコピーできない
   return(false);
  }
//+------------------------------------------------------------------+
//| ティックの文字列の説明を返す                                     |
//+------------------------------------------------------------------+
string GetTickDescription(MqlTick &tick)
  {
   string desc=StringFormat("%s.%03u ", TimeToString(tick.time, TIME_DATE|
TIME_MINUTES|TIME_SECONDS),tick.time_msc%1000);

//--- ティックのフラグを確認する
   bool buy_tick   = ((tick.flags &TICK_FLAG_BUY)    == TICK_FLAG_BUY);
   bool sell_tick  = ((tick.flags &TICK_FLAG_SELL)   == TICK_FLAG_SELL);
   bool ask_tick   = ((tick.flags &TICK_FLAG_ASK)    == TICK_FLAG_ASK);
   bool bid_tick   = ((tick.flags &TICK_FLAG_BID)    == TICK_FLAG_BID);
   bool last_tick  = ((tick.flags &TICK_FLAG_LAST)   == TICK_FLAG_LAST);
   bool volume_tick= ((tick.flags &TICK_FLAG_VOLUME)== TICK_FLAG_VOLUME);

//--- まず取引フラグのティックを確認する（CustomTicksAdd()には何も存在しない）
   if(buy_tick || sell_tick)
     {
      //--- 取引ティックの出力を作成する
      desc += (buy_tick ? StringFormat("Buy Tick: Last=%G Volume=%d ", tick.last,
tick.volume)  : "");
      desc += (sell_tick? StringFormat("Sell Tick: Last=%G Volume=%d ",tick.last,
tick.volume) : "");
      desc += (ask_tick ? StringFormat("Ask=%G ", tick.ask) : "");
      desc += (bid_tick ? StringFormat("Bid=%G ", tick.ask) : "");
      desc += "(Trade tick)";
     }
   else
     {
      //--- 情報ティックの出力を少し異なる方法で作成する
      desc += (ask_tick   ? StringFormat("Ask=%G ",  tick.ask)    : "");
```

© 2000-2025, MetaQuotes Ltd.

```
      desc += (bid_tick   ? StringFormat("Bid=%G ",  tick.ask)    : "");
      desc += (last_tick  ? StringFormat("Last=%G ", tick.last)   : "");
      desc += (volume_tick? StringFormat("Volume=%d ",tick.volume): "");
      desc += "(Info tick)";
     }
//--- ティックの説明を返す
   return(desc);
  }
```

参照

CustomRatesDelete、CustomRatesUpdate、CustomTicksReplace、CopyTicks、CopyTicksRange

© 2000–2025, MetaQuotes Ltd.

CustomTicksReplace

指定された時間間隔内のカスタム銘柄の価格履歴をMqlTick型の配列からのデータで完全に置き換えます。

```
int  CustomTicksReplace(
   const string     symbol,            // 銘柄名
   long             from_msc,          // 開始の日付
   long             to_msc,            // 終了の日付
   const MqlTick&   ticks[],           // カスタム銘柄に適用されるデータの配列
   uint             count=WHOLE_ARRAY  // 使用されるticks[]配列要素の数
   );
```

パラメータ

symbol

[in] カスタム銘柄名

from_msc

[in] 削除する指定範囲内の価格履歴の最初のティックの時間（1970.01.01 から経過したミリ秒）

to_msc

[in] 削除する指定範囲内の価格履歴の最後のティックの時間（1970.01.01 から経過したミリ秒）

ticks[]

[in] 昇順に並び替えられたMqlTick型のティックデータの配列

count=WHOLE_ARRAY

[in] 指定した時間間隔内の置き換えに使用される*ticks[]*配列要素の数。WHOLE_ARRAYはすべての*ticks[]*配列要素が使用されることを意味します。

戻り値

更新されたティックの数（エラーの場合は-1）

注意事項

クォーツの流れの中ではしばしば複数のティックがミリ秒単位で同じ時間を持つことがあるので（正確なティックの時間はMqlTick構造体の*time_msc*フィールドに保存）、CustomTicksReplace関数は自動的に*ticks[]*配列要素を時間順に並び替えることはありません。したがって、ティックの配列は事前に時間の昇順で並び替える必要があります。

ティックは*to_msc*で指定された時間まで、またはエラーが発生するまで、毎日連続して置き換えられます。指定された範囲の日は順番に処理され。ティック時間と昇順（非降順）の間の不一致が検出されると、現在の日のティックの置き換えが停止します。現在の日（間違ったティックの瞬間）と指定された間隔の残りは変更されませんが、その前日までのティックはすべて普通に置き換えられます。

*ticks[]*配列にいずれの日のティックデータも含まれない場合は（一般的に、どの期間でも）、*ticks[]*からのティックデータが適応された後に、カスタム銘柄履歴には欠落したデータに対応する「穴」が表示されます。言い換えれば CustomTicksReplaceを欠陥したティックで呼び出すとティック履歴のその部分を削除され、これはCustomTicksDeleteを「穴」の範囲で呼び出すのと同じです。

© 2000-2025, MetaQuotes Ltd.

例：

```
//+------------------------------------------------------------------+
//|                                              CustomTicksReplace.mq5 |
//|                                  Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"     // カスタム銘柄名
#define   CUSTOM_SYMBOL_PATH     "Forex"           // 銘柄が作成されるグループの名前
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()          // カスタム銘柄の基となる銘柄の名前

#define   DATATICKS_TO_COPY      UINT_MAX          // コピーされたティックの数
#define   DATATICKS_TO_PRINT     20                // 操作ログに送信されるティックの数

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- カスタム銘柄を作成するときにエラーコードを取得する
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH,
CUSTOM_SYMBOL_ORIGIN);

//--- エラーコードが0（銘柄の作成に成功）でも5304（銘柄がすでに作成されている）でもない場合は、終了する
   if(create!=0 && create!=5304)
      return;

//--- 標準銘柄ティックデータをMqlTick配列に取得する
   MqlTick array[]={};
   if(!GetTicksToArray(CUSTOM_SYMBOL_ORIGIN, DATATICKS_TO_COPY, array))
      return;

//--- 標準銘柄の最初と最後に受信したティックの時間を出力する
   int total=(int)array.Size();
   PrintFormat("First tick time: %s.%03u, Last tick time: %s.%03u",
               TimeToString(array[0].time,TIME_DATE|TIME_MINUTES|TIME_SECONDS),
array[0].time_msc%1000,
               TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|
TIME_SECONDS), array[total-1].time_msc%1000);

//--- 標準銘柄の最後のDATATICKS_TO_PRINTティックを操作ログに出力する
   PrintFormat("\nThe last %d ticks for the standard symbol '%s':",
DATATICKS_TO_PRINT, CUSTOM_SYMBOL_ORIGIN);
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
     {
      if(i<0)
```

© 2000-2025, MetaQuotes Ltd.

```
         continue;
       PrintFormat("  %dth Tick: %s", i, GetTickDescription(array[i]));
      }

//--- 気配値表示ウィンドウにカスタム銘柄を追加する
   ResetLastError();
   if(!SymbolSelect(CUSTOM_SYMBOL_NAME, true))
     {
      Print("SymbolSelect() failed. Error ", GetLastError());
      return;
     }

//--- カスタム銘柄の価格履歴にティック配列データを追加する
   Print("...");
   uint start=GetTickCount();
   PrintFormat("Start of adding %u ticks to the history of the custom symbol '%s'",
array.Size(), CUSTOM_SYMBOL_NAME);
   int added=CustomTicksAdd(CUSTOM_SYMBOL_NAME, array);
   PrintFormat("Added %u ticks to the history of the custom symbol '%s' in %u ms",
added, CUSTOM_SYMBOL_NAME, GetTickCount()-start);

//--- get the newly added custom symbol tick data to the MqlTick array
   Print("...");
   if(!GetTicksToArray(CUSTOM_SYMBOL_NAME, array.Size(), array))
      return;

//--- カスタム銘柄の最初と最後に受信したティックの時間を出力する
   total=(int)array.Size();
   PrintFormat("First tick time: %s.%03u, Last tick time: %s.%03u",
               TimeToString(array[0].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS),
array[0].time_msc%1000,
               TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|
TIME_SECONDS), array[total-1].time_msc%1000);

//--- カスタム銘柄の最後のDATATICKS_TO_PRINTティックを操作ログに出力する
   PrintFormat("\nThe last %d ticks for the custom symbol '%s':",
DATATICKS_TO_PRINT, CUSTOM_SYMBOL_NAME);
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
     {
      if(i<0)
         continue;
      PrintFormat("  %dth Tick: %s", i, GetTickDescription(array[i]));
     }

//--- now change Ask and Bid tick values in the array using the equation Ask(Symbol)
= 1.0 / Ask(Symbol), Bid(Symbol) = 1.0 / Bid(Symbol)
   for(int i=0; i<total; i++)
     {
      array[i].ask = (array[i].ask !=0 ? 1.0 / array[i].ask : array[i].ask);
      array[i].bid = (array[i].bid !=0 ? 1.0 / array[i].bid : array[i].bid);
```

© 2000–2025, MetaQuotes Ltd.

```
      }
   Print("\nNow the ticks are changed");

//--- カスタム銘柄のティック履歴を、変更されたティック配列のデータに置き換える
   Print("...");
   start=GetTickCount();
   PrintFormat("Start replacing %u changed ticks in the history of the custom symbol '%s'", array.Size(), CUSTOM_SYMBOL_NAME);
   int replaced=CustomTicksReplace(CUSTOM_SYMBOL_NAME, array[0].time_msc, array[total-1].time_msc, array);
   PrintFormat("Replaced %u ticks in the history of the custom symbol '%s' in %u ms", replaced, CUSTOM_SYMBOL_NAME, GetTickCount()-start);

//--- 新しく置き換えられたカスタム銘柄のティックデータをMql Ti ck配列に取得する
   Print("...");
   if(!GetTicksToArray(CUSTOM_SYMBOL_NAME, array.Size(), array))
      return;

//--- カスタム銘柄の最初に受信した変更されたティックと最後に受信した変更されたティックの時刻を出力する
   total=(int)array.Size();
   PrintFormat("First changed tick time: %s.%03u, Last changed tick time: %s.%03u",
               TimeToString(array[0].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS), array[0].time_msc%1000,
               TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS), array[total-1].time_msc%1000);

/--- カスタム銘柄の最後のDATATI CKS_ TO_ PRI NT変更済みティックを操作ログに出力する
   PrintFormat("\nThe last %d changed ticks for the custom symbol '%s':", DATATICKS_TO_PRINT, CUSTOM_SYMBOL_NAME);
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
     {
      if(i<0)
         continue;
      PrintFormat("  %dth Changed tick: %s", i, GetTickDescription(array[i]));
     }

//--- チャートのコメントにスクリプト終了キーに関するヒントを表示する
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and exit", CUSTOM_SYMBOL_NAME));
//--- Escキーまたは Del キーが押されるまで待機して無限ループを終了する
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- Del キーを押すと、作成したカスタム銘柄とそのデータが削除される
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         //--- バーデータを削除する
         int deleted=CustomRatesDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
```

© 2000–2025, MetaQuotes Ltd.

```
            PrintFormat("%d history bars of the custom symbol '%s' were successfully
deleted", deleted, CUSTOM_SYMBOL_NAME);

         //--- ティックデータを削除する
         deleted=CustomTicksDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history ticks of the custom symbol '%s' were
successfully deleted", deleted, CUSTOM_SYMBOL_NAME);

         //--- 銘柄を削除する
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully",
CUSTOM_SYMBOL_NAME);
         break;
        }
     }
//--- 終了する前にチャートをクリアする
   Comment("");
   /*
   結果:
   Requested 4294967295 ticks to get tick history for the symbol 'EURUSD'
   The tick history for the 'EURUSD' symbol is received in the amount of 351195822
ticks in 55735 ms
   First tick time: 2011.12.19 00:00:08.000, Last tick time: 2024.06.21 08:39:03.113

   The last 20 ticks for the standard symbol 'EURUSD':
     351195802th Tick: 2024.06.21 08:38:10.076 Ask=1.07194 (Info tick)
     351195803th Tick: 2024.06.21 08:38:13.162 Ask=1.07195 (Info tick)
     351195804th Tick: 2024.06.21 08:38:13.872 Bid=1.07195 (Info tick)
     351195805th Tick: 2024.06.21 08:38:14.866 Ask=1.07194 Bid=1.07194 (Info tick)
     351195806th Tick: 2024.06.21 08:38:17.374 Bid=1.07194 (Info tick)
     351195807th Tick: 2024.06.21 08:38:18.883 Bid=1.07194 (Info tick)
     351195808th Tick: 2024.06.21 08:38:19.771 Bid=1.07194 (Info tick)
     351195809th Tick: 2024.06.21 08:38:20.873 Ask=1.07195 Bid=1.07195 (Info tick)
     351195810th Tick: 2024.06.21 08:38:22.278 Ask=1.07196 Bid=1.07196 (Info tick)
     351195811th Tick: 2024.06.21 08:38:22.775 Bid=1.07196 (Info tick)
     351195812th Tick: 2024.06.21 08:38:23.477 Bid=1.07196 (Info tick)
     351195813th Tick: 2024.06.21 08:38:38.194 Ask=1.07197 (Info tick)
     351195814th Tick: 2024.06.21 08:38:38.789 Ask=1.07196 (Info tick)
     351195815th Tick: 2024.06.21 08:38:39.290 Ask=1.07197 (Info tick)
     351195816th Tick: 2024.06.21 08:38:43.695 Ask=1.07196 (Info tick)
     351195817th Tick: 2024.06.21 08:38:52.203 Ask=1.07195 Bid=1.07195 (Info tick)
     351195818th Tick: 2024.06.21 08:38:55.105 Ask=1.07196 Bid=1.07196 (Info tick)
     351195819th Tick: 2024.06.21 08:38:57.607 Ask=1.07195 Bid=1.07195 (Info tick)
     351195820th Tick: 2024.06.21 08:39:00.512 Ask=1.07196 Bid=1.07196 (Info tick)
     351195821th Tick: 2024.06.21 08:39:03.113 Ask=1.07195 Bid=1.07195 (Info tick)
   ...
   Start of adding 351195822 ticks to the history of the custom symbol 'EURUSD.C'
   Added 351195822 ticks to the history of the custom symbol 'EURUSD.C' in 349407 ms
```

© 2000–2025, MetaQuotes Ltd.

```
   ...
   Requested 351195822 ticks to get tick history for the symbol 'EURUSD.C'
   The tick history for the 'EURUSD.C' symbol is received in the amount of 351195822
 ticks in 190203 ms
   First tick time: 2011.12.19 00:00:08.000, Last tick time: 2024.06.21 08:39:03.113

   The last 20 ticks for the custom symbol 'EURUSD.C':
     351195802th Tick: 2024.06.21 08:38:10.076 Ask=1.07194 Bid=1.07194 (Info tick)
     351195803th Tick: 2024.06.21 08:38:13.162 Ask=1.07195 Bid=1.07195 (Info tick)
     351195804th Tick: 2024.06.21 08:38:13.872 Ask=1.07195 Bid=1.07195 (Info tick)
     351195805th Tick: 2024.06.21 08:38:14.866 Ask=1.07194 Bid=1.07194 (Info tick)
     351195806th Tick: 2024.06.21 08:38:17.374 Ask=1.07194 Bid=1.07194 (Info tick)
     351195807th Tick: 2024.06.21 08:38:18.883 Ask=1.07194 Bid=1.07194 (Info tick)
     351195808th Tick: 2024.06.21 08:38:19.771 Ask=1.07194 Bid=1.07194 (Info tick)
     351195809th Tick: 2024.06.21 08:38:20.873 Ask=1.07195 Bid=1.07195 (Info tick)
     351195810th Tick: 2024.06.21 08:38:22.278 Ask=1.07196 Bid=1.07196 (Info tick)
     351195811th Tick: 2024.06.21 08:38:22.775 Ask=1.07196 Bid=1.07196 (Info tick)
     351195812th Tick: 2024.06.21 08:38:23.477 Ask=1.07196 Bid=1.07196 (Info tick)
     351195813th Tick: 2024.06.21 08:38:38.194 Ask=1.07197 Bid=1.07197 (Info tick)
     351195814th Tick: 2024.06.21 08:38:38.789 Ask=1.07196 Bid=1.07196 (Info tick)
     351195815th Tick: 2024.06.21 08:38:39.290 Ask=1.07197 Bid=1.07197 (Info tick)
     351195816th Tick: 2024.06.21 08:38:43.695 Ask=1.07196 Bid=1.07196 (Info tick)
     351195817th Tick: 2024.06.21 08:38:52.203 Ask=1.07195 Bid=1.07195 (Info tick)
     351195818th Tick: 2024.06.21 08:38:55.105 Ask=1.07196 Bid=1.07196 (Info tick)
     351195819th Tick: 2024.06.21 08:38:57.607 Ask=1.07195 Bid=1.07195 (Info tick)
     351195820th Tick: 2024.06.21 08:39:00.512 Ask=1.07196 Bid=1.07196 (Info tick)
     351195821th Tick: 2024.06.21 08:39:03.113 Ask=1.07195 Bid=1.07195 (Info tick)

   Now the ticks are changed
   ...
   Start replacing 351195822 changed ticks in the history of the custom symbol
'EURUSD.C'
   Replaced 351195822 ticks in the history of the custom symbol 'EURUSD.C' in 452266
 ms
   ...
   Requested 351195822 ticks to get tick history for the symbol 'EURUSD.C'
   The tick history for the 'EURUSD.C' symbol is received in the amount of 351195822
 ticks in 199812 ms
   First changed tick time: 2011.12.19 00:00:08.000, Last changed tick time:
2024.06.21 08:39:03.113

   The last 20 changed ticks for the custom symbol 'EURUSD.C':
     351195802th Changed tick: 2024.06.21 08:38:10.076 Ask=0.93289 Bid=0.93289 (Info
 tick)
     351195803th Changed tick: 2024.06.21 08:38:13.162 Ask=0.93288 Bid=0.93288 (Info
 tick)
     351195804th Changed tick: 2024.06.21 08:38:13.872 Ask=0.93288 Bid=0.93288 (Info
 tick)
     351195805th Changed tick: 2024.06.21 08:38:14.866 Ask=0.93289 Bid=0.93289 (Info
 tick)
```

© 2000–2025, MetaQuotes Ltd.

```
    351195806th Changed tick: 2024.06.21 08:38:17.374 Ask=0.93289 Bid=0.93289 (Info
 tick)
    351195807th Changed tick: 2024.06.21 08:38:18.883 Ask=0.93289 Bid=0.93289 (Info
 tick)
    351195808th Changed tick: 2024.06.21 08:38:19.771 Ask=0.93289 Bid=0.93289 (Info
 tick)
    351195809th Changed tick: 2024.06.21 08:38:20.873 Ask=0.93288 Bid=0.93288 (Info
 tick)
    351195810th Changed tick: 2024.06.21 08:38:22.278 Ask=0.93287 Bid=0.93287 (Info
 tick)
    351195811th Changed tick: 2024.06.21 08:38:22.775 Ask=0.93287 Bid=0.93287 (Info
 tick)
    351195812th Changed tick: 2024.06.21 08:38:23.477 Ask=0.93287 Bid=0.93287 (Info
 tick)
    351195813th Changed tick: 2024.06.21 08:38:38.194 Ask=0.93286 Bid=0.93286 (Info
 tick)
    351195814th Changed tick: 2024.06.21 08:38:38.789 Ask=0.93287 Bid=0.93287 (Info
 tick)
    351195815th Changed tick: 2024.06.21 08:38:39.290 Ask=0.93286 Bid=0.93286 (Info
 tick)
    351195816th Changed tick: 2024.06.21 08:38:43.695 Ask=0.93287 Bid=0.93287 (Info
 tick)
    351195817th Changed tick: 2024.06.21 08:38:52.203 Ask=0.93288 Bid=0.93288 (Info
 tick)
    351195818th Changed tick: 2024.06.21 08:38:55.105 Ask=0.93287 Bid=0.93287 (Info
 tick)
    351195819th Changed tick: 2024.06.21 08:38:57.607 Ask=0.93288 Bid=0.93288 (Info
 tick)
    351195820th Changed tick: 2024.06.21 08:39:00.512 Ask=0.93287 Bid=0.93287 (Info
 tick)
    351195821th Changed tick: 2024.06.21 08:39:03.113 Ask=0.93288 Bid=0.93288 (Info
 tick)
  */
 }
//+------------------------------------------------------------------+
//| カスタム銘柄ルを作成し、エラーコードを返す                                |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const
string symbol_origin=NULL)
 {
//--- カスタム銘柄の基となる銘柄の名前をする
  string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- カスタム銘柄の作成に失敗し、エラー5304ではない場合は、操作ログに報告する
  ResetLastError();
  int error=0;
  if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
    {
     error=GetLastError();
     if(error!=5304)
        PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name,
symbol_path, origin, error);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- 成功
   return(error);
  }
//+------------------------------------------------------------------+
//| カスタム銘柄を削除する                                           |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- 気配値表示ウィンドウから銘柄を非表示にする
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- カスタム銘柄の削除に失敗した場合は、操作ログにこれを報告してfalseを返す
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name,
GetLastError());
      return(false);
     }
//--- 成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 配列内の指定された数のティックを取得する                         |
//+------------------------------------------------------------------+
bool GetTicksToArray(const string symbol, const uint count, MqlTick &array[])
  {
//--- 履歴データの読み込み開始を通知する
   PrintFormat("Requested %u ticks to get tick history for the symbol '%s'", count,
symbol);

//--- ティックの受信を3回試行する
   int attempts=0;
   while(attempts<3)
     {
      //--- ティックを受信する前に開始時間を測定する
      uint start=GetTickCount();

      //--- 1970.01.01 00:00.001 以降のティック履歴を要求する（パラメータ from=1 ms）
      int received=CopyTicks(symbol, array, COPY_TICKS_ALL, 1, count);
      if(received!=-1)
        {
         //--- ティック数と経過時間に関する情報を表示する
```

© 2000-2025, MetaQuotes Ltd.

```
         PrintFormat("The tick history for the '%s' symbol is received in the amount
of %u ticks in %d ms", symbol, received, GetTickCount()-start);

         //--- ティック履歴が同期されている場合、エラーコードはゼロになる - trueを返す
         if(GetLastError()==0)
            return(true);

         PrintFormat("%s: Ticks are not synchronized yet, %d ticks received for %d
ms. Error=%d",
                     symbol, received, GetTickCount()-start, GetLastError());
        }
      //--- 試行回数を数える
      attempts++;
      //--- ティック データベースの同期が終了するまで1秒間待機する
      Sleep(1000);
     }
//--- 3 回の試行でティックをコピーできない
   return(false);
  }
//+------------------------------------------------------------------+
//| ティックの文字列の説明を返す                                     |
//+------------------------------------------------------------------+
string GetTickDescription(MqlTick &tick)
  {
   string desc=StringFormat("%s.%03u ", TimeToString(tick.time, TIME_DATE|
TIME_MINUTES|TIME_SECONDS),tick.time_msc%1000);

//--- ティックのフラグを確認する
   bool buy_tick   = ((tick.flags &TICK_FLAG_BUY)    == TICK_FLAG_BUY);
   bool sell_tick  = ((tick.flags &TICK_FLAG_SELL)   == TICK_FLAG_SELL);
   bool ask_tick   = ((tick.flags &TICK_FLAG_ASK)    == TICK_FLAG_ASK);
   bool bid_tick   = ((tick.flags &TICK_FLAG_BID)    == TICK_FLAG_BID);
   bool last_tick  = ((tick.flags &TICK_FLAG_LAST)   == TICK_FLAG_LAST);
   bool volume_tick= ((tick.flags &TICK_FLAG_VOLUME)== TICK_FLAG_VOLUME);

//--- まず取引フラグのティックを確認する（CustomTicksAdd()には何も存在しない）
   if(buy_tick || sell_tick)
     {
      //--- 取引ティックの出力を作成する
      desc += (buy_tick ? StringFormat("Buy Tick: Last=%G Volume=%d ", tick.last,
tick.volume)  : "");
      desc += (sell_tick? StringFormat("Sell Tick: Last=%G Volume=%d ",tick.last,
tick.volume) : "");
      desc += (ask_tick ? StringFormat("Ask=%G ", tick.ask) : "");
      desc += (bid_tick ? StringFormat("Bid=%G ", tick.ask) : "");
      desc += "(Trade tick)";
     }
   else
     {
      //--- 情報ティックの出力を少し異なる方法で作成する
```

© 2000-2025, MetaQuotes Ltd.

```
      desc += (ask_tick   ? StringFormat("Ask=%G ",  tick.ask)    : "");
      desc += (bid_tick   ? StringFormat("Bid=%G ",  tick.ask)    : "");
      desc += (last_tick  ? StringFormat("Last=%G ", tick.last)   : "");
      desc += (volume_tick? StringFormat("Volume=%d ",tick.volume): "");
      desc += "(Info tick)";
     }
//--- ティックの説明を返す
   return(desc);
  }
```

参照

CustomRatesDelete、CustomRatesUpdate、CustomTicksDelete、CopyTicks、CopyTicksRange

© 2000-2025, MetaQuotes Ltd.

CustomBookAdd

銘柄の板情報の構造体を渡します。ブローカーサーバーから価格が到着したかのように板情報をブロードキャストできます。

```
bool  CustomBookAdd(
   const string         symbol,             // 銘柄名
   const MqlBookInfo&   books[]             // 板情報要素の説明を持つ配列
   uint                 count=WHOLE_ARRAY   // 使用される要素の数
   );
```

パラメータ

symbol

[in] カスタム銘柄名。

books[]

[in] The array of MqlBookInfo型データの配列で板情報のステータスを完全に記述します(すべての売買リクエスト)。渡された板情報は以前の者を完全に置き換えます。

count=WHOLE_ARRAY

[in] 関数に渡される'books'配列要素の数。デフォルトでは配列全体が使用されます。

戻り値

成功の場合はtrue、それ以外の場合はfalse。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

注意事項

CustomBookAdd関数は板情報が開かれたカスタム銘柄とのみ作動します(プラットフォームインターフェイスまたはMarketBookAdd関数による)。

板情報を投入したとき、銘柄の買値と売値は更新されません。CustomTicksAddを使用して、最良価格の変更を制御してティックを投入する必要があります。

この関数は、送信されたデータの正確性を検証します。各要素の種類、価格、量を示す必要があります。さらに、MqlBookInfo.volumeとMqlBookInfo.volume_realはゼロまたは負であってはなりません。 両方のボリュームが負の場合はエラーと見なされます。ボリュームのいずれかまたは両方を指定できます。指定されたボリュームまたは正のボリュームが使用されます。

```
volume=-1 && volume_real=2 — volume_real=2 will be used,
a volume=3 && volume_real=0 — volume=3 will be used.
```

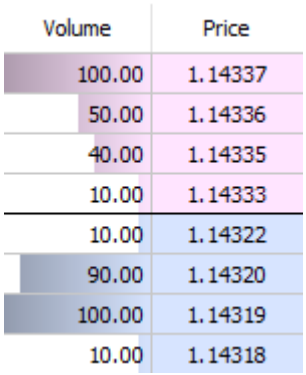
'books'配列内のMqlBookInfo要素の順序は関係ありません。データを保存するとき、端末は独自に価格によってデータを分類します。

データを保存するとき、受け取り側のカスタム銘柄の"Book depth" (SYMBOL_TICKS_BOOKDEPTH)パラメータが確認されます。渡された板情報で売りリクエストの数がこの値を超えると、超過レベルは破棄されます。買いリクエストについても同様です。

'books'配列の書き込み例:

© 2000–2025, MetaQuotes Ltd.

板情報ステータス	books[]の書き込み

	books[0].type=BOOK_TYPE_SELL;
	books[0].price=1.14337;
	books[0].volume=100;
	books[1].type=BOOK_TYPE_SELL;
	books[1].price=1.14330;
	books[1].volume=50;
	books[2].type=BOOK_TYPE_SELL;
	books[2].price=1.14335;
	books[2].volume=40;
	books[3].type=BOOK_TYPE_SELL;
	books[3].price=1.14333;
	books[3].volume=10;
	books[4].type=BOOK_TYPE_BUY;
	books[4].price=1.14322;
	books[4].volume=10;
	books[5].type=BOOK_TYPE_BUY;
	books[5].price=1.14320;
	books[5].volume=90;
	books[6].type=BOOK_TYPE_BUY;
	books[6].price=1.14319;
	books[6].volume=100;
	books[7].type=BOOK_TYPE_BUY;
	books[7].price=1.14318;
	books[7].volume=10;

例：

```
//+------------------------------------------------------------------+
//|  エキスパート初期化関数                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- データ源となる銘柄の板情報を有効にする
   MarketBookAdd(Symbol());
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//|  エキスパート初期化解除関数                                      |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
  }
//+------------------------------------------------------------------+
//|  ティック関数                                                    |
//+------------------------------------------------------------------+
void OnTick(void)
  {
   MqlTick ticks[];
   ArrayResize(ticks,1);
//--- 現在の価格を共通銘柄からカスタム銘柄にコピーする
   if(SymbolInfoTick(Symbol(),ticks[0]))
     {
```

© 2000–2025, MetaQuotes Ltd.

```
      string symbol_name=Symbol()+".SYN";
      CustomTicksAdd(symbol_name,ticks);
     }
  }
//+------------------------------------------------------------------+
//| Book function                                                    |
//+------------------------------------------------------------------+
void OnBookEvent(const string &book_symbol)
  {
//--- 現在の板情報ステータスを共通銘柄からカスタム銘柄にコピーする
   if(book_symbol==Symbol())
     {
      MqlBookInfo book_array[];
      if(MarketBookGet(Symbol(),book_array))
        {
         string symbol_name=Symbol()+".SYN";
         CustomBookAdd(symbol_name,book_array);
        }
     }
  }
//+------------------------------------------------------------------+
```

参照

MarketBookAdd、CustomTicksAdd、OnBookEvent

© 2000-2025, MetaQuotes Ltd.

チャート操作

チャートプロパティを設定するための関数（ChartSetInteger、ChartSetDouble、 ChartSetString）は非同期で、チャートに更新コマンドを送信するために使用されます。これらの関数が正常に実行された場合、コマンドはチャートイベントの共通キューに含まれます。チャートプロパティの変更は、このチャートのイベントキューの処理とともに実装されます。

したがって、非同期関数を呼び出した後のチャートの即座の更新は期待できません。チャートの外観とプロパティを強制的に更新するにはChartRedraw()関数を使用します。

関数	アクション
ChartApplyTemplate	指定されたファイルの特定のテンプレートをチャートに適用します。
ChartSaveTemplate	チャートの設定を指定された名称のテンプレートに保存します。
ChartWindowFind	指標が描画されるサブウィンドウの数を返します。
ChartTimePriceToXY	チャートの座標を時間/価格表現からXとYの座標に変換します。
ChartXYToTimePrice	チャートの座標をXとYの座標から時間/価格表現に変換します。
ChartOpen	指定された銘柄と期間で新しいチャートを開きます。
ChartClose	指定されたチャートを閉じます。
ChartFirst	クライアント端末の最初のチャートのIDを返します。
ChartNext	指定されたチャートの次のチャート識別子を返します。
ChartSymbol	指定されたチャートの銘柄名を返します。
ChartPeriod	指定されたチャートの期間値を返します。
ChartRedraw	指定されたチャートの強制再描画を呼び出します。
ChartSetDouble	指定されたチャートのプロパティにdouble値を設定します。
ChartSetInteger	指定されたチャートのプロパティに整数値（datetime、int、color、boolまたはchar）を設定します。
ChartSetString	指定されたチャートのプロパティに文字列値を設定します。
ChartGetDouble	指定されたチャートのプロパティのdouble値を返します。
ChartGetInteger	指定されたチャートのプロパティの整数値を返します。
ChartGetString	指定されたチャートのプロパティの文字列値を返します。
ChartNavigate	チャート内の指定された位置を基準に指定された足数によって指定されたチャートのシフトを実行します。
ChartID	現在のチャートのIDを返します。
ChartIndicatorAdd	指定されたハンドルと指標を指定されたチャートウィンドウに追加します。
ChartIndicatorDelete	指定された名称の指標を指定されたチャートウィンドウから削除します。

© 2000–2025, MetaQuotes Ltd.

関数	アクション
ChartIndicatorGet	指定されたチャートウィンドウで短縮名によって指定された指標ハンドルを返します。
ChartIndicatorName	指定されたチャートの指標のリスト内の数によって指標の短縮名を返します。
ChartIndicatorsTotal	指定されたチャートウィンドウに適用された全ての指標の数を返します。
ChartWindowOnDropped	エキスパートアドバイザーやスクリプトがドロップされたサブウィンドウチャートの番号（インデックス）を返します。
ChartPriceOnDropped	エキスパートアドバイザーやスクリプトがドロップされたチャートのポイントの価格座標を返します。
ChartTimeOnDropped	エキスパートアドバイザーやスクリプトがドロップされたチャートのポイントの時間座標を返します。
ChartXOnDropped	エキスパートアドバイザーやスクリプトがドロップされたチャートのポイントのX座標を返します。
ChartYOnDropped	エキスパートアドバイザーやスクリプトがドロップされたチャートのポイントのY座標を返します。
ChartSetSymbolPeriod	指定されたチャートの銘柄値と期間を変更します。
ChartScreenShot	指定された拡張子に応じてGIF、PNGまたはBMP形式で現在のチャートのスクリーンショットを提供します。

© 2000–2025, MetaQuotes Ltd.

ChartApplyTemplate

チャートに指定されたファイルから特定のテンプレートを適用します。コマンドは、チャートメッセージキューに追加され、以前の全てのコマンドの処理後に実行されます。

```
bool  ChartApplyTemplate(
   long           chart_id,      // チャート識別子
   const string   filename       // テンプレートファイル名
   );
```

パラメータ

chart_id

[in] チャート識別子。(0 は現在のチャート)

filename

[in] テンプレートを含むファイルの名称

戻り値

コマンドがチャートキューに追加された場合は true、そうでない場合は false。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

注意事項

エキスパートアドバイザーは、接続されているチャートへの新しいテンプレートの読み込みが成功した場合には、アンロードされ、動作を継続することは出来ません。

チャートにテンプレートを適用する場合、セキュリティ上の理由のために取引権限が制限されることがあります。

> **ライブ取引許可は ChartApplyTemplate() 関数を使用してテンプレートを適用することによって開始したエキスパートアドバイザーには拡張することは出来ません。**

ChartApplyTemplate() 関数を呼び出した MQL5 プログラムに取引権限がない場合は、テンプレートから起動されたエキスパートアドバイザーは、テンプレートの設定に関係なく取引することは出来ません。

ChartApplyTemplate() 関数を呼び出し MQL5 プログラムに取引権限がある場合でも、テンプレートの設定にそのような権限はなければ、テンプレートから起動されたエキスパートアドバイザーは、取引することが出来ません。

テンプレートの使用

MQL5 言語のリソースは、ChartSetInteger() 関数を使用して、色に関した複数のチャートのプロパティの設定を許可します。

- チャートの背景色
- 軸、スケールとOHLCラインの色
- グリッドの色
- ボリュームやポジションオープンレベルの色
- アップバー、影と強気ローソク足の縁の色
- ダウンバー、影や弱気ローソク足の縁の色

© 2000–2025, MetaQuotes Ltd.

- チャートの線と同時線の色
- 強気ローソク足の実体の色
- 弱気ローソク足の実体の色
- 売値ラインの色
- 買値ラインの色
- 直近の約定価格（Last）の線の色
- 逆指値（決済逆指値及び決済指値）レベルの色

また、チャートには、複数のグラフィカルオブジェクトと指標が存在出来ます。全ての必要な指標とチャートを設定した後、テンプレートとして保存することが出来ます。このようなテンプレートは、任意のチャートに適用することが出来ます。

ChartApplyTemplate()関数は、以前に保存したテンプレートを使用するためにあり、任意のMQL5 プログラムで使用することが出来ます。テンプレートを格納するファイルへのパスは ChartApplyTemplate() の2番目のパラメータとして受け渡しされます。テンプレートファイルは、次のルールに従って検索されます。

- バックスラッシュ"¥" セパレータ（"¥"として記述）がパスの先頭に配置されている場合、テンプレートは terminal_data_directory¥MQL5 との相対パスで検索されます。
- バックスラッシュがない場合、テンプレートは ChartApplyTemplate() を呼び出した実行可能な EX5 ファイルへの相対位置から検索されます。
- テンプレートが上記の2つで見つからない場合は、検索が terminal_directory¥Profiles¥Templates¥ で行われます。

ここで terminal_directory はメタトレーダー5クライアント端末が実行されているフォルダで terminal_data_directory が編集可能なファイルが保存されているフォルダで、その場所は、オペレーティングシステム、ユーザ名、及びコンピュータのセキュリティ設定に依存します。通常、これらは別々のフォルダですが、場合によって、同じフォルダであっても可能です。

terminal_data_directory と terminal_directory フォルダの場所は TerminalInfoString() 関数で習得出来ます。

```
//--- 端末が起動されたディレクトリ
   string terminal_path=TerminalInfoString(TERMINAL_PATH);
   Print("Terminal directory:",terminal_path);
//--- EAと指標を含むMQL5フォルダが配置されている端末のデータディレクトリ
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
   Print("Terminal data directory:",terminal_data_path);
```

例えば、

```
//--- terminal_data_directory¥MQL5¥ でテンプレートを探す
ChartApplyTemplate(0,"\\first_template.tpl"))
//--- directory_of_EX5_file¥ に続いて terminal_data_directory¥Profiles¥Templates¥でテンプレートを探す
ChartApplyTemplate(0,"second_template.tpl"))
//--- directory_of_EX5_file¥My_templates¥ に続いて
terminal_directory¥Profiles¥Templates¥My_templates¥ でテンプレートを探す
ChartApplyTemplate(0,"My_templates\\third_template.tpl"))
```

テンプレートは実行可能なEX5ファイルに含むことは出来ないのでリソースではありません。

例：

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- ¥MQL5¥Files にあるテンプレートを適用する例
   if(FileIsExist("my_template.tpl"))
     {
      Print("The file my_template.tpl found in \Files'");
      //--- テンプレートを適用する
      if(ChartApplyTemplate(0,"\\Files\\my_template.tpl"))
        {
         Print("The template 'my_template.tpl' applied successfully");
         //--- チャートを再描画
         ChartRedraw();
        }
      else
         Print("Failed to apply 'my_template.tpl', error code ",GetLastError());
     }
   else
     {
      Print("File 'my_template.tpl' not found in "
            +TerminalInfoString(TERMINAL_PATH)+"\\MQL5\\Files");
     }
  }
```

参照

リソース

© 2000-2025, MetaQuotes Ltd.

ChartSaveTemplate

チャートの設定を指定された名称のテンプレートに保存します。

```
bool  ChartSaveTemplate(
   long           chart_id,      // チャート識別子
   const string   filename       // テンプレートを保存するファイル名
   );
```

パラメータ

chart_id

[in] チャート識別子。(0 は現在のチャート)

filename

[in] テンプレートを保存するファイル名「.tpl」拡張子は自動的にファイル名に追加されるので、指定する必要はありません。テンプレートは **terminal_directory¥Profiles¥Templates¥** に保存され、端末での手動適用のために使用することが出来ます。同名のテンプレートが既存する場合、このファイルの内容は上書きされます。

戻り値

成功の場合は true、それ以外の場合は false。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

注意事項

テンプレートを使用すると、適用される指標やグラフィカルオブジェクトとチャートの設定を全て保存し、後に別のチャートに適用出来ます。

例:

```
//+------------------------------------------------------------------+
//|                                      Test_ChartSaveTemplate.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property script_show_inputs
//--- 入力パラメータ
input string                symbol="GBPUSD";  // 新しいチャートのシンボル
input ENUM_TIMEFRAMES       period=PERIOD_H3; // 新しいチャートの時間軸
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 最初のチャートに指標を取り付ける
   int handle;
//--- 指標を使用に向けて準備する
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!PrepareZigzag(NULL,0,handle)) return; // 失敗に続く終了
//--- チャートに指標を別ウィンドウで取り付ける
   if(!ChartIndicatorAdd(0,1,handle))
     {
      PrintFormat("Failed to attach to chart %s/%s an indicator with the handle=%d.
Error code %d",
                  _Symbol,
                  EnumToString(_Period),
                  handle,
                  GetLastError());
      //--- プログラム動作を終了する
      return;
     }
//--- チャートを更新して指標を表示する
   ChartRedraw();
//--- ジグザグの最後の 2 つの切れ目を探す
   double two_values[];
   datetime two_times[];
   if(!GetLastTwoFractures(two_values,two_times,handle))
     {
      PrintFormat("Failed to find two last fractures in the Zigzag!");
      //--- プログラム動作を終了する
      return;
     }
//--- 標準偏差チャンネルを取り付ける
   string channel="StdDeviation Channel";
   if(!ObjectCreate(0,channel,OBJ_STDDEVCHANNEL,0,two_times[1],0))
     {
      PrintFormat("Failed to create object %s. Error code %d",
                  EnumToString(OBJ_STDDEVCHANNEL),GetLastError());
      return;
     }
   else
     {
      //--- チャンネルが作成されたので2 番目の点を定義する
      ObjectSetInteger(0,channel,OBJPROP_TIME,1,two_times[0]);
      //--- チャンネルのツールヒントテキストを設定する
      ObjectSetString(0,channel,OBJPROP_TOOLTIP,"Demo from MQL5 Help");
      //--- チャートを更新する
      ChartRedraw();
     }
//--- テンプレートに結果を保存する
   ChartSaveTemplate(0,"StdDevChannelOnZigzag");
//--- 新しいチャートを開いて、保存されたテンプレートを適用する
   long new_chart=ChartOpen(symbol,period);
   //--- グラフィックオブジェクトのツールヒントを有効にする
   ChartSetInteger(new_chart,CHART_SHOW_OBJECT_DESCR,true);
   if(new_chart!=0)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 保存されたテンプレートをチャートに適用する
      ChartApplyTemplate(new_chart,"StdDevChannelOnZigzag");
     }
   Sleep(10000);
  }
//+------------------------------------------------------------------+
//| ジグザグのハンドルを作成し、データの準備を整える                 |
//+------------------------------------------------------------------+
bool PrepareZigzag(string sym,ENUM_TIMEFRAMES tf,int &h)
  {
   ResetLastError();
//--- ジグザグ指標は terminal_data_folder¥MQL5¥Examples に位置する必要がある
   h=iCustom(sym,tf,"Examples\\Zigzag");
   if(h==INVALID_HANDLE)
     {
      PrintFormat("%s: Failed to create the handle of the Zigzag indicator. Error code %d",
                  __FUNCTION__,GetLastError());
      return false;
     }
//--- 指標ハンドルを作成時には、値を計算するための時間が必要となる
   int k=0; // 指標計算を待機する試みの回数
//--- 各ループで50ミリ秒停止して計算を待機する
   while(BarsCalculated(h)<=0)
     {
      k++;
      //--- 試みの回数を表示する
      PrintFormat("%s: k=%d",__FUNCTION__,k);
      //--- 指標が計算されるまで 50 ミリ秒待機する
      Sleep(50);
      //--- 100 回以上試みた場合は、何かがおかしい
      if(k>100)
        {
         //--- 問題を報告する
         PrintFormat("Failed to calculate the indicator for %d attempts!");
         //--- プログラム動作を終了する
         return false;
        }
     }
//--- 指標が作成され、値が計算されて、準備が整う
   return true;
  }
//+------------------------------------------------------------------+
//| ジグザグの最後の 2 つの切れ目を探して配列に書き込む              |
//+------------------------------------------------------------------+
bool GetLastTwoFractures(double &get_values[],datetime &get_times[],int handle)
  {
   double values[];          // ジグザグの値の配列
   datetime times[];         // 時間取得の配列
```

© 2000-2025, MetaQuotes Ltd.

```
   int size=100;             // 配列サイズ
   ResetLastError();
//--- 指標の最後の100 個の値を複製する
   int copied=CopyBuffer(handle,0,0,size,values);
//--- 複製された値の数をチェックする
   if(copied<100)
     {
      PrintFormat("%s: Failed to copy %d values of the indicator with the handle=%d.
Error code %d",
                  __FUNCTION__,size,handle,GetLastError());
      return false;
     }
//--- 配列へのアクセスの順序を時系列のように定義する
   ArraySetAsSeries(values,true);
//--- 切れ目が見つかったバーの数を書く
   int positions[];
//--- 配列サイズを設定する
   ArrayResize(get_values,3); ArrayResize(get_times,3); ArrayResize(positions,3);
//---カウンタ
   int i=0,k=0;
//--- 切れ目を探し始める
   while(i<100)
     {
      double v=values[i];
      //--- 空の値には興味がない
      if(v!=0.0)
        {
         //--- バー番号を覚える
         positions[k]=i;
         //--- 切れ目でのジグザグの値を覚える
         get_values[k]=values[i];
         PrintFormat("%s: Zigzag[%d]=%G",__FUNCTION__,i,values[i]);
         //--- カウンタを増加する
         k++;
         //--- 切れ目が 2 つ見つかったら、ループから出る
         if(k>2) break;
        }
      i++;
     }
//--- 配列へのアクセスの順序を時系列のように定義する
   ArraySetAsSeries(times,true);   ArraySetAsSeries(get_times,true);
   if(CopyTime(_Symbol,_Period,0,size,times)<=0)
     {
      PrintFormat("%s: Failed to copy %d values from CopyTime(). Error code %d",
                  __FUNCTION__,size,GetLastError());
      return false;
     }
//--- 最後の 2 つの折れ目が発生したバーのオープン時間を開く
   get_times[0]=times[positions[1]];// 最後から2 番目の値が初めの折れ目として書かれる
```

© 2000-2025, MetaQuotes Ltd.

```
   get_times[1]=times[positions[2]];// 最後から3 番目の値が2 番目の折れ目になる
   PrintFormat("%s: first=%s,  second=%
s",__FUNCTION__,TimeToString(get_times[1]),TimeToString(get_times[0]));
//--- 成功
   return true;
  }
```

参照

ChartApplyTemplate()、リソース

© 2000–2025, MetaQuotes Ltd.

ChartWindowFind

この関数は指標が描画されるサブウィンドウの数を返します。この関数には 2 つのバージョンがあります。

1. 指標の指定された「短縮名」(短縮名はサブウィンドウの左側上部に表示されています)を検索し、成功した場合にはサブウィンドウ番号を返します。

```
int  ChartWindowFind(
   long     chart_id,                  // チャート識別子
   string   indicator_shortname        // 短縮名 INDICATOR_SHORTNAMEを参照
```

2. この関数はカスタム指標から呼び出される必要があります。指標が位置するサブウィンドウ番号が返されます。

```
int  ChartWindowFind();
```

パラメータ

chart_id

[in] チャート識別子。(0 は現在のチャート)

indicator_shortname

[in] 指標の短縮名

戻り値

サブウィンドウ番号 (失敗の場合は -1)

注意事項

関数の2 つ目のバージョン(パラメータなし) がスクリプトまたはエキスパートアドバイザーから呼び出された場合、関数は-1を返します。

指標の短縮名と iCustom() や IndicatorCreate() 関数で指標が作成された時に指定されるファイル名を混同しないでください。指標の短縮名が明示的に設定されていない場合、コンパイル時に指標のソースコードを含むファイルの名称が指定されます。

IndicatorSetString()関数でINDICATOR_SHORTNAME プロパティに設定される指標の短縮名を正しく形成することは重要です。 ChartIndicatorDelete() 関数によってチャートから削除される指標は短縮名によって識別されるので、短縮名に指標の全ての入力パラメータの値を含むようお勧めします。

例 :

```
#property script_show_inputs
//--- 入力パラメータ
input string   shortname="MACD(12,26,9)";
//+------------------------------------------------------------------+
//|  この指標のチャートウィンドウの番号を返す                        |
//+------------------------------------------------------------------+
int GetIndicatorSubWindowNumber(long chartID=0,string short_name="")
  {
   int window=-1;
//---
   if((ENUM_PROGRAM_TYPE)MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- 関数が指標から呼び出されているので名称は必須でない
       window=ChartWindowFind();
      }
    else
      {
       //--- 関数がエキスパートアドバイザーやスクリプトから呼び出されている
       window=ChartWindowFind(0,short_name);
       if(window==-1) Print(__FUNCTION__+"(): Error = ",GetLastError());
      }
//---
    return(window);
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
    int window=GetIndicatorSubWindowNumber(0,shortname);
    if(window!=-1)
       Print("Indicator "+shortname+" is in the window #"+(string)window);
    else
       Print("Indicator "+shortname+" is not found. window = "+(string)window);
  }
```

参照

ObjectCreate()、ObjectFind()

© 2000-2025, MetaQuotes Ltd.

ChartTimePriceToXY

チャートの座標を時間/価格表現からXとYの座標に変換します。

```
bool  ChartTimePriceToXY(
   long           chart_id,      // チャート識別子
   int            sub_window,    // サブウィンドウ番号
   datetime       time,          // チャートの時間
   double         price,         // チャートの価格
   int&           x,             // X 座標は時間
   int&           y              // Y 座標は価格
   );
```

パラメータ

chart_id

[in] チャート識別子。(0 は現在のチャート)

sub_window

[in] チャートサブウィンドウの番号 (0 はメインチャートウィンドウ)

time

[in] X 軸方向のピクセルの値が受け取られるチャートの時間値。原点はメインチャートウィンドウの左上隅にあります。

価格

[in] Y 軸方向のピクセルの値が受け取られるチャートの価格値。原点はメインチャートウィンドウの左上隅にあります。

x

[out] 時間からXへの変換が受け取られる変数

y

[out] 価格からYへの変換が受け取られる変数

戻り値

成功の場合は true、それ以外の場合は false 。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

例：

```
#define   SYMBOL    "GBPUSD"
#define   PERIOD    PERIOD_H1

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 現在のチャートI D、銘柄、期間を取得する
   long           chart_id= ChartID();
```

© 2000-2025, MetaQuotes Ltd.

```
   string           symbol  = Symbol();
   ENUM_TIMEFRAMES period  = Period();

//--- 現在のチャートの銘柄と期間を SYMBOL と PERIOD で設定されたものに置き換えたことを操作ログ報告する
   PrintFormat("Change the %s symbol and the %s period of the chart %I64u to %s %s",
               symbol, TimeframeDescrioption(period), chart_id, SYMBOL,
TimeframeDescrioption(PERIOD));

//--- チャートの銘柄と期間を変更する
   ChartSetSymbolPeriod(chart_id, SYMBOL, PERIOD);
   /*
   結果:
   Change the EURUSD symbol and the M1 period of the chart 133246248352168440 to
GBPUSD H1
   */
  }
//+------------------------------------------------------------------+
//| チャートの期間の説明を返す                                       |
//+------------------------------------------------------------------+
string TimeframeDescrioption(const ENUM_TIMEFRAMES timeframe)
  {
   return(StringSubstr(EnumToString(timeframe), 7));
  }
```

参照

ChartXYToTimePrice()

© 2000-2025, MetaQuotes Ltd.

ChartXYToTimePrice

チャートの座標をXとYの座標から時間/価格表現に変換します。

```
bool  ChartXYToTimePrice(
   long            chart_id,      // チャート識別子
   int             x,             // チャートの X 座標
   int             y,             // チャートの Y 座標
   int&            sub_window,    // サブウィンドウ番号
   datetime&       time,          // チャートの時間
   double&         price          // チャートの価格
   );
```

パラメータ

chart_id

[in] チャート識別子。(0は現在のチャート)

x

[in] X座標

y

[in] Y座標

sub_window

[out] チャートサブウィンドウの番号を示す変数(0はメインチャートウィンドウ)

time

[out] X軸方向のピクセルの値が受け取られるチャートの時間値原点はメインチャートウィンドウの左上隅にあります。

価格

[out] Y軸方向のピクセルの値が受け取られるチャートの価格値原点はメインチャートウィンドウの左上隅にあります。

戻り値

成功の場合はtrue、それ以外の場合はfalse。エラー情報を取得するには、GetLastError()関数が呼ばれます。

例:

```
//+------------------------------------------------------------------+
//| ChartEvent 関数                                                  |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- チャートのイベントパラメータを表示する
   Comment(__FUNCTION__,": id=",id," lparam=",lparam," dparam=",dparam,"
sparam=",sparam);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- チャートのマウスクリックイベント？
   if(id==CHARTEVENT_CLICK)
     {
      //--- 変数を準備する
      int      x     =(int)lparam;
      int      y     =(int)dparam;
      datetime dt    =0;
      double   price =0;
      int      window=0;
      //--- X とYの座標を日付/時間的に変更する
      if(ChartXYToTimePrice(0,x,y,window,dt,price))
        {
         PrintFormat("Window=%d X=%d  Y=%d  =>  Time=%s  Price=%
G",window,x,y,TimeToString(dt),price);
         //--- 逆変換を実行する:( X, Y) => ( Ti me, Pri ce)
         if(ChartTimePriceToXY(0,window,dt,price,x,y))
            PrintFormat("Time=%s  Price=%G  =>  X=%d  Y=%
d",TimeToString(dt),price,x,y);
         else
            Print("ChartTimePriceToXY return error code: ",GetLastError());
         //--- 行を削除する
         ObjectDelete(0,"V Line");
         ObjectDelete(0,"H Line");
         //--- 十字線の水平線と垂直線を作成します。
         ObjectCreate(0,"H Line",OBJ_HLINE,window,dt,price);
         ObjectCreate(0,"V Line",OBJ_VLINE,window,dt,price);
         ChartRedraw(0);
        }
      else
         Print("ChartXYToTimePrice return error code: ",GetLastError());
      Print("+------------------------------------------------------------------+");
     }
  }
```

参照

ChartTimePriceToXY()

© 2000-2025, MetaQuotes Ltd.

ChartOpen

指定された銘柄と期間で新しいチャートを開きます。

```
long  ChartOpen(
   string           symbol,     // 銘柄名
   ENUM_TIMEFRAMES  period      // 期間
   );
```

パラメータ

symbol

[in] チャートシンボル(NULL はエキスパートアドバイザーがに接続されている現在のチャートのシンボル)

period

[in] チャート期間(時間軸)。ENUM_TIMEFRAMES のいずれかの値。(0 は現在のチャート期間)

戻り値

成功した場合は開かれたチャートのID、その他の場合は 0 。

注意事項

端末内に同時に開いているチャートの数は CHARTS_MAXを超えることは出来ません。

例：

```
#define CHART_SYMBOL  NULL
#define CHART_PERIOD  PERIOD_CURRENT

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 新しいチャートシンボルと時間枠を設定する
   string symbol=CHART_SYMBOL;
   if(symbol==NULL)
      symbol=Symbol();
   ENUM_TIMEFRAMES timeframe = (CHART_PERIOD==PERIOD_CURRENT ? Period() :
CHART_PERIOD);

//--- 指定された銘柄と期間で新しいチャートを開く
   long chart_id=ChartOpen(symbol, timeframe);
   if(chart_id==0)
     {
      Print("ChartOpen() failed. Error ", GetLastError());
      return;
     }

//--- 開いているチャートのパラメータを操作ログにプリントする
```

© 2000-2025, MetaQuotes Ltd.

```
   PrintFormat("A new chart of the %s symbol has been opened with a period of %s and
ChartID %I64u",
               symbol, StringSubstr(EnumToString(timeframe), 7), chart_id);
   /*
   結果:
   A new chart of the GBPUSD symbol has been opened with a period of M1 and ChartID
133346697706632016
   */
  }
```

© 2000–2025, MetaQuotes Ltd.

ChartFirst

クライアント端末の最初のチャートのIDを返します。

```
long  ChartFirst();
```

戻り値

チャート識別子

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- クライアント端末で最初に開かれたチャートのID を取得する
   long              chart_id= ChartFirst();

//--- 取得したID を使用してチャートのシンボルと期間も取得する
   string            symbol  = ChartSymbol(chart_id);
   ENUM_TIMEFRAMES   period  = ChartPeriod(chart_id);

//--- クライアント端末の最初のチャートの説明を操作ログ内に表示する
   PrintFormat("ID of the first chart of the client terminal: %I64d, chart symbol: %s, chart period: %s", chart_id, symbol, StringSubstr(EnumToString(period), 7));
   /*
   結果:
   ID of the first chart of the client terminal: 133246248352168440, chart symbol: EURUSD, chart period: M1
   */
  }
```

© 2000–2025, MetaQuotes Ltd.

ChartNext

指定されたチャートの次のチャート識別子を返します。

```
long  ChartNext(
   long  chart_id      // チャート識別子
   );
```

パラメータ

chart_id

[in] チャート識別子。0 は、現在のチャートを意味せず、「1 番目のチャート識別子を返す」という意味です。

戻り値

Chart ID。これがチャートリストの最後である場合は -1。

例:

```
//--- チャート識別子の変数
   long currChart,prevChart=ChartFirst();
   int i=0,limit=100;
   Print("ChartFirst =",ChartSymbol(prevChart)," ID =",prevChart);
   while(i<limit)// チャートは 100 以上は開かれていないはず
     {
      currChart=ChartNext(prevChart); // 前のチャート識別子を使用して、新しいチャート識別子を取得する
      if(currChart<0) break;           // チャートリストの終わりに達した
      Print(i,ChartSymbol(currChart)," ID =",currChart);
      prevChart=currChart;// ChartNext() のために現在のチャート識別子を保存する
      i++;// カウンタを増加させることを忘れない
     }
```

© 2000-2025, MetaQuotes Ltd.

ChartClose

指定されたチャートを閉じます。

```
bool  ChartClose(
   long  chart_id=0      // チャート識別子
   );
```

パラメータ

chart_id=0

[in] チャート識別子。（0は現在のチャート）

戻り値

成功の場合はtrue、それ以外の場合はfalse

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 現在の銘柄と期間で新しいチャートを開く
   long chart_id=ChartOpen(_Symbol, _Period);
   if(chart_id==0)
     {
      Print("ChartOpen() failed. Error ", GetLastError());
      return;
     }

//--- 開いているチャートのパラメータを操作ログにプリントする
   PrintFormat("A new chart of the %s symbol has been opened with a period of %s and
ChartID %I64u",
               _Symbol, StringSubstr(EnumToString(_Period), 7), chart_id);

//--- 2秒待って、新しく開いたチャートを閉じる
   PrintFormat("Waiting 3 seconds before closing a newly opened chart with ID %I64d
...", chart_id);
   Sleep(3000);
   ResetLastError();
   if(!ChartClose(chart_id))
     {
      Print("ChartClose() failed. Error ", GetLastError());
      return;
     }

//--- チャートが正常に閉じられたというメッセージを操作ログにプリントする
   PrintFormat("The chart with ID %I64d was successfully closed", chart_id);
   /*
```

© 2000-2025, MetaQuotes Ltd.

```
   結果:
   A new chart of the GBPUSD symbol has been opened with a period of M1 and ChartID 133346697706632016
   Waiting 3 seconds before closing a newly opened chart with ID 133346697706632016
...
   The chart with ID 133346697706632016 was successfully closed
   */
  }
```

© 2000–2025, MetaQuotes Ltd.

ChartSymbol

指定されたチャートの銘柄名を返します。

```
string  ChartSymbol(
   long  chart_id=0      // チャート識別子
   );
```

パラメータ

chart_id=0

[in] チャート識別子。（0は現在のチャート）

戻り値

チャートが存在しない場合は空の文字列が返されます。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 現在のチャート銘柄を取得し、取得した値を操作ログに表示する
   string chart_symbol = ChartSymbol();
   Print("Current chart symbol: ", chart_symbol);

//--- 既存の(この場合は現在の)チャートID を取得する
   long chart_id=ChartID();
   chart_symbol=ChartSymbol(chart_id);
   PrintFormat("Chart symbol with ID %I64d: %s", chart_id, chart_symbol);

//--- 銘柄を受け取ったときにランダムなチャートID を設定する
   chart_symbol = ChartSymbol(1234567890);
   if(chart_symbol=="")
      Print("The chart with ID 1234567890 does not exist");
   else
      Print("Chart symbol with ID 1234567890: ", chart_symbol);
   /*
   結果:
   Current chart symbol: GBPUSD
   Chart symbol with ID 132966427583395104: GBPUSD
   The chart with ID 1234567890 does not exist
   */
  }
```

参照

ChartSetSymbolPeriod

© 2000-2025, MetaQuotes Ltd.

ChartPeriod

指定されたチャートの時間軸期間を返します。

```
ENUM_TIMEFRAMES  ChartPeriod(
   long  chart_id=0      // チャート識別子
   );
```

パラメータ

chart_id=0

[in] チャート識別子。（0は現在のチャート）

戻り値

ENUM_TIMEFRAMES のうちの1つの値（チャートが存在しない場合は0）

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 現在のチャート期間を取得し、取得した値を操作ログに表示する
   ENUM_TIMEFRAMES period=ChartPeriod();
   Print("Current chart period: ", EnumToString(period));

//--- 既存の(この場合は現在の)チャートID を取得する
   long chart_id=ChartID();
   period=ChartPeriod(chart_id);
   PrintFormat("Chart period with ID %I64d: %s", chart_id, EnumToString(period));

//--- ランダムなチャートID を設定する
   period=ChartPeriod(1234567890);
   if(period==0)
      Print("The chart with ID 1234567890 does not exist");
   else
      Print("Chart period with ID 1234567890: ", EnumToString(period));
   /*
   結果:
   Current chart period: PERIOD_M15
   Chart period with ID 133510090240498505: PERIOD_M15
   The chart with ID 1234567890 does not exist
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

ChartRedraw

この関数は指定されたチャートの強制再描画を呼び出します。

```
void  ChartRedraw(
   long  chart_id=0      // チャート識別子
   );
```

パラメータ

chart_id=0

[in] チャート識別子。(0 は現在のチャート)

注意事項

通常 object propertiesが更新された後で使用されます。

例 :

```
#define   WIDTH      50    // rectangle width in bars

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 現在のチャートI Dを取得し、グラフィカルオブジェクト名を設定する
   long   chart_id = ChartID();
   string obj_name = MQLInfoString(MQL_PROGRAM_NAME)+"_RectLabel";

//--- 2つの長方形の時間座標を取得する
   datetime time1  = iTime(_Symbol, _Period, WIDTH);
   datetime time2  = iTime(_Symbol, _Period, 0);
   if(time1==0 || time2==0)
     {
      Print("Error getting time ", GetLastError());
      return;
     }

//--- 長方形の幅の範囲内の最大価格と最小価格を取得する
   double price1 = HighestHigh(_Symbol, _Period, 0, WIDTH);
   double price2 = LowestLow(_Symbol, _Period, 0, WIDTH);
   if(price1==EMPTY_VALUE || price2==EMPTY_VALUE)
      return;

//--- 長方形オブジェクトを作成する
   Print("Create a wheat-colored rectangle");
   if(!ObjectCreate(chart_id, obj_name, OBJ_RECTANGLE, 0, time1, price1, time2,
price2))
     {
      Print("ObjectCreate() failed. Error ", GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }

//--- 元の色で四角形を塗りつぶす
   ObjectSetInteger(chart_id, obj_name, OBJPROP_FILL, true);
   ObjectSetInteger(chart_id, obj_name, OBJPROP_BACK, true);
   ObjectSetInteger(chart_id, obj_name, OBJPROP_COLOR, clrWheat);
   ChartRedraw();

//--- 1秒待って四角形をDodgerBlue色で塗りつぶし、グラフを更新する
   Sleep(1000);
   Print("Change color to DodgerBlue");
   ObjectSetInteger(chart_id, obj_name, OBJPROP_COLOR, clrDodgerBlue);
   ChartRedraw();

//--- 1秒待って四角形をLimeGreen色で塗りつぶし、グラフを更新する
   Sleep(1000);
   Print("Change color to LimeGreen");
   ObjectSetInteger(chart_id, obj_name, OBJPROP_COLOR, clrLimeGreen);
   ChartRedraw();

//--- 1秒待って四角形をOrangeRed色で塗りつぶし、グラフを更新する
   Sleep(1000);
   Print("Change color to OrangeRed");
   ObjectSetInteger(chart_id, obj_name, OBJPROP_COLOR, clrOrangeRed);
   ChartRedraw();

//--- 1秒待って四角形をWheat色で塗りつぶし、グラフを更新する
   Sleep(1000);
   Print("Reset color to original");
   ObjectSetInteger(chart_id, obj_name, OBJPROP_COLOR, clrWheat);
   ChartRedraw();

//--- 3秒後、チャートからオブジェクトを削除する
   Sleep(3000);
   Print("Delete the rectangle");
   ObjectDelete(chart_id, obj_name);
  }
//+------------------------------------------------------------------+
//|  指定されたバーの範囲内の最大の高値を返す                                |
//+------------------------------------------------------------------+
double HighestHigh(const string symbol, const ENUM_TIMEFRAMES timeframe, const uint
start, const uint count)
  {
   ResetLastError();
   int index=iHighest(symbol, timeframe, MODE_HIGH, count, start);
   if(index==-1)
     {
      PrintFormat("%s: iHighest() failed. Error %d",__FUNCTION__, GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
      return(EMPTY_VALUE);
     }
   GetLastError();
   double price=iHigh(symbol, timeframe, index);
   if(price==0)
     {
      PrintFormat("%s: iHigh() failed. Error %d",__FUNCTION__, GetLastError());
      return(EMPTY_VALUE);
     }
   return(price);
  }
//+------------------------------------------------------------------+
//|  指定されたバーの範囲内の最小の安値を返す                                  |
//+------------------------------------------------------------------+
double LowestLow(const string symbol, const ENUM_TIMEFRAMES timeframe, const uint
start, const uint count)
  {
   ResetLastError();
   int index=iLowest(symbol, timeframe, MODE_LOW, count, start);
   if(index==-1)
     {
      PrintFormat("%s: iLowest() failed. Error %d",__FUNCTION__, GetLastError());
      return(EMPTY_VALUE);
     }
   GetLastError();
   double price=iLow(symbol, timeframe, index);
   if(price==0)
     {
      PrintFormat("%s: iLow() failed. Error %d",__FUNCTION__, GetLastError());
      return(EMPTY_VALUE);
     }
   return(price);
  }
```

参照

オブジェクト関数

© 2000–2025, MetaQuotes Ltd.

ChartSetDouble

指定されたチャートのプロパティに値を設定します。チャートプロパティは double 型でなければなりません。コマンドは、チャートメッセージキューに追加され、以前の全てのコマンドの処理後に実行されます。

```
bool  ChartSetDouble(
   long                          chart_id,      // チャート識別子
   ENUM_CHART_PROPERTY_DOUBLE    prop_id,       // プロパティ識別子
   double                        value          // 値
   );
```

パラメータ

chart_id

[in] チャート識別子。(0 は現在のチャート)

prop_id

[in] チャートプロパティ識別子。ENUM_CHART_PROPERTY_DOUBLE の値の 1 つ (読み込み専用のプロパティを除いて)。

value

[in] プロパティ値

戻り値

コマンドがチャートキューに追加された場合は true、そうでない場合は false。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

注意事項

これは非同期関数です。つまり、この関数は、指定されたチャートのキューに正常に追加されたコマンドの実行を待たずに、すぐに制御を返します。プロパティは、チャートキューから適切なコマンドが処理された後にのみ変更されます。チャートキューからコマンドをすぐに実行するには ChartRedraw関数を呼び出します。

複数のチャートプロパティをすぐに一度に変更したい場合は、対応する関数 (ChartSetString、ChartSetDouble、ChartSetString) を1つのコードブロックで実行してからChartRedrawを一回呼び出す必要があります。

コマンドの実行結果を確認するには、関数を使用して、指定されたチャートプロパティ (ChartGetInteger、 ChartGetDouble、ChartSetString) をリクエストしますI。ただし、これらは同期関数であり、実行結果を待つことに注意してください。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 現在のチャートI Dを取得する
   long chart_id = ChartID();

//--- ゼロバーの右端からの初期オフセットを割合で保存し、自動スクロールフラグを設定する
```

© 2000-2025, MetaQuotes Ltd.

```
   double shift = ChartGetDouble(chart_id, CHART_SHIFT_SIZE);
   bool   scroll= ChartGetInteger(chart_id,CHART_AUTOSCROLL);

//--- 自動スクロールチャートフラグをリセットする
   ChartSetInteger(chart_id, CHART_AUTOSCROLL, true);
   PrintFormat("Initial chart shift size: %.2f", shift);

//--- ループ内で2.0%から50.0%まで0.5%ずつ増加
   for(int i=20; i<=500; i+=5)
     {
      //--- シフト サイズを i / 10 に設定し、指定された値を操作ログに出力する
      ChartSetDouble(chart_id, CHART_SHIFT_SIZE, i/10.0);
      PrintFormat("Set chart shift size to %.1f%%", i/10.0);

      //--- チャートのゼロバーを指定されたシフト距離に維持する
      ChartNavigate(chart_id, CHART_END, 0);

      //--- 待機する
      Sleep(16);
     }

//--- 初期のチャートシフトと自動スクロールを設定する
   Print("Set the chart shift size to the initial value of ", shift);
   ChartSetDouble(ChartID(), CHART_SHIFT_SIZE, shift);
   ChartSetInteger(chart_id, CHART_AUTOSCROLL, scroll);
  }
```

© 2000-2025, MetaQuotes Ltd.

ChartSetInteger

指定されたチャートのプロパティに値を設定します。チャートプロパティは datetime、int、bool または char型でなければなりません。コマンドは、チャートメッセージキューに追加され、以前の全てのコマンドの処理後に実行されます。

```
bool  ChartSetInteger(
   long                           chart_id,      // チャート識別子
   ENUM_CHART_PROPERTY_INTEGER    prop_id,       // プロパティ識別子
   long                           value          // 値
   );
```

```
bool  ChartSetInteger(
   long                           chart_id,      // チャート識別子
   ENUM_CHART_PROPERTY_INTEGER    prop_id,       // プロパティ識別子
   int                            sub_window,    // サブウィンドウ番号
   long                           value          // 値
   );
```

パラメータ

chart_id

[in] チャート識別子。(0 は現在のチャート)

prop_id

[in] チャートプロパティ識別子。ENUM_CHART_PROPERTY_INTEGER の値の 1 つ (読み込み専用のプロパティを除いて)。

sub_window

[in] チャートサブウィンドウの番号。最初のケースの場合、初期値は0(メインチャートウィンドウ)です。プロパティのほとんどは、サブウィンドウの番号を必要としません。

value

[in] プロパティ値

戻り値

コマンドがチャートキューに追加された場合は true、そうでない場合は false。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

注意事項

これは非同期関数です。つまり、この関数は、指定されたチャートのキューに正常に追加されたコマンドの実行を待たずに、すぐに制御を返します。プロパティは、チャートキューから適切なコマンドが処理された後にのみ変更されます。チャートキューからコマンドをすぐに実行するには ChartRedraw関数を呼び出します。

複数のチャートプロパティをすぐに一度に変更したい場合は、対応する関数 (ChartSetString、ChartSetDouble、ChartSetString) を1つのコードブロックで実行してからChartRedrawを一回呼び出す必要があります。

© 2000–2025, MetaQuotes Ltd.

コマンドの実行結果を確認するには、関数を使用して、指定されたチャートプロパティ（ChartGetInteger、 ChartGetDouble、ChartSetString）をリクエストします。ただし、これらは同期関数であり、実行結果を待つことに注意してください。

例：

```
//+------------------------------------------------------------------+
//|  エキスパート初期化関数                                          |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- チャートウィンドウでのマウス移動イベントの有効化
   ChartSetInteger(0,CHART_EVENT_MOUSE_MOVE,1);
//--- チャートプロパティを強制的に更新して、イベント処理の準備を整える
   ChartRedraw();
  }
//+------------------------------------------------------------------+
//| MouseState                                                       |
//+------------------------------------------------------------------+
string MouseState(uint state)
  {
   string res;
   res+="\nML: "   +(((state& 1)== 1)?"DN":"UP");   // 左
   res+="\nMR: "   +(((state& 2)== 2)?"DN":"UP");   // 右
   res+="\nMM: "   +(((state&16)==16)?"DN":"UP");   // 中央
   res+="\nMX: "   +(((state&32)==32)?"DN":"UP");   // 初めのXキー
   res+="\nMY: "   +(((state&64)==64)?"DN":"UP");   // 2番目のXキー
   res+="\nSHIFT: "+(((state& 4)== 4)?"DN":"UP");   // shiftキー
   res+="\nCTRL: " +(((state& 8)== 8)?"DN":"UP");   // Ctrlキー
   return(res);
  }
//+------------------------------------------------------------------+
//|  ChartEvent関数                                                  |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,const long &lparam,const double &dparam,const string
&sparam)
  {
   if(id==CHARTEVENT_MOUSE_MOVE)
      Comment("POINT: ",(int)lparam,",",(int)dparam,"\n",MouseState((uint)sparam));
  }
```

© 2000-2025, MetaQuotes Ltd.

ChartSetString

指定されたチャートのプロパティに値を設定します。チャートプロパティは string 型でなければなりません。コマンドは、チャートメッセージキューに追加され、以前の全てのコマンドの処理後に実行されます。

```
bool  ChartSetString(
   long                          chart_id,      // チャート識別子
   ENUM_CHART_PROPERTY_STRING    prop_id,       // プロパティ識別子
   string                        str_value      // 値
   );
```

パラメータ

chart_id

[in] チャート識別子。(0 は現在のチャート)

prop_id

[in] チャートプロパティ識別子。値は ENUM_CHART_PROPERTY_STRING のいずれかです(読み込み専用のプロパティ以外)。

str_value

[in] プロパティ値の文字列文字列の長さは2045文字を超えることは出来ません。(余分な文字は切り捨てられます)。

戻り値

コマンドがチャートキューに追加された場合は true、そうでない場合は false。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

注意事項

ChartSetString はコメントをチャートに出力するのに Comment 関数の代わりに使用することが出来ます。

これは非同期関数です。つまり、この関数は、指定されたチャートのキューに正常に追加されたコマンドの実行を待たずに、すぐに制御を返します。プロパティは、チャートキューから適切なコマンドが処理された後にのみ変更されます。チャートキューからコマンドをすぐに実行するには ChartRedraw関数を呼び出します。

複数のチャートプロパティをすぐに一度に変更したい場合は、対応する関数 (ChartSetString、ChartSetDouble、ChartSetString) を1つのコードブロックで実行してからChartRedrawを一回呼び出す必要があります。

コマンドの実行結果を確認するには、関数を使用して、指定されたチャートプロパティ (ChartGetInteger、 ChartGetDouble、ChartSetString) をリクエストしますI。ただし、これらは同期関数であり、実行結果を待つことに注意してください。

例 :

```
void OnTick()
  {
//---
   double Ask,Bid;
   int Spread;
```

© 2000-2025, MetaQuotes Ltd.

```
   Ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
   Bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   Spread=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD);
   string comment=StringFormat("Выводим цены:\nAsk = %G\nBid = %G\nSpread = %d",
                               Ask,Bid,Spread);
   ChartSetString(0,CHART_COMMENT,comment);
  }
```

参照

コメント、ChartGetString

© 2000–2025, MetaQuotes Ltd.

ChartGetDouble

指定されたチャートの対応するプロパティの値を返します。チャートプロパティは double 型でなければなりません。この関数の呼び出し方は 2 つあります。

1. 直接プロパティ値を返します。

```
double  ChartGetDouble(
   long                          chart_id,         // チャート識別子
   ENUM_CHART_PROPERTY_DOUBLE    prop_id,          // プロパティ識別子
   int                           sub_window=0      // サブウィンドウ番号（必要に応じて）
   );
```

2. 関数の成功に応じて、true または false を返します。成功した場合、プロパティの値は、参照によって渡された目的変数 double_var に配置されます。

```
bool  ChartGetDouble(
   long                          chart_id,         // チャート識別子
   ENUM_CHART_PROPERTY_DOUBLE    prop_id,          // プロパティ識別子
   int                           sub_window,       // サブウィンドウ番号
   double&                       double_var        // チャートのプロパティのターゲット変数
   );
```

パラメータ

chart_id

[in] チャート識別子。（0 は現在のチャート）

prop_id

[in] チャートプロパティ識別子。値は ENUM_CHART_PROPERTY_DOUBLE のいずれかです。

sub_window

[in] チャートサブウィンドウの番号。最初のケースの場合、初期値は0（メインチャートウィンドウ）です。プロパティのほとんどは、サブウィンドウの番号を必要としません。

double_var

[out] 要求されたプロパティの double 型のターゲット変数

戻り値

double 型の値

2 番目の呼び方では、指定されたプロパティが利用可能であり、その値が double_var 変数に置かれている場合には、true を返します。そうでない場合は false を返します。エラーについての追加情報を取得するには、GetLastError() 関数を呼び出します。

注意事項

この関数は同期型です。つまり、呼び出し前にチャートキューに追加されたすべてのコマンドの実行を待機します。

例：

```
void OnStart()
```

© 2000–2025, MetaQuotes Ltd.

```
{
 double priceMin=ChartGetDouble(0,CHART_PRICE_MIN,0);
 double priceMax=ChartGetDouble(0,CHART_PRICE_MAX,0);
 Print("CHART_PRICE_MIN =",priceMin);
 Print("CHART_PRICE_MAX =",priceMax);
}
```

© 2000–2025, MetaQuotes Ltd.

ChartGetInteger

指定されたチャートの対応するプロパティの値を返します。チャートプロパティは datetime、int または bool 型でなければなりません。この関数の呼び出し方は 2 つあります。

1. 直接プロパティ値を返します。

```
long  ChartGetInteger(
   long                          chart_id,         // チャート識別子
   ENUM_CHART_PROPERTY_INTEGER   prop_id,          // プロパティ識別子
   int                           sub_window=0      // サブウィンドウ番号（必要に応じて）
   );
```

2. 関数の成功に応じて、true または false を返します。成功した場合、プロパティの値は、参照によって渡された目的変数 long_var に配置されます。

```
bool  ChartGetInteger(
   long                          chart_id,         // チャート識別子
   ENUM_CHART_PROPERTY_INTEGER   prop_id,          // プロパティ識別子
   int                           sub_window,       // サブウィンドウ番号
   long&                         long_var          // チャートのプロパティのターゲット変数
   );
```

パラメータ

chart_id

[in] チャート識別子。（0 は現在のチャート）

prop_id

[in] チャートプロパティ識別子。値は ENUM_CHART_PROPERTY_INTEGER のいずれかです。

sub_window

[in] チャートサブウィンドウの番号。最初のケースの場合、初期値は0（メインチャートウィンドウ）です。プロパティのほとんどは、サブウィンドウの番号を必要としません。

long_var

[out] 要求されたプロパティの long 型のターゲット変数

戻り値

long 型の値

2 番目の呼び方では、指定されたプロパティが利用可能であり、その値が long_var 変数に置かれている場合には、true を返します。そうでない場合は false を返します。エラーについての追加情報を取得するには、GetLastError() 関数を呼び出します。

注意事項

この関数は同期型です。つまり、呼び出し前にチャートキューに追加されたすべてのコマンドの実行を待機します。

例：

```
void OnStart()
```

© 2000–2025, MetaQuotes Ltd.

```
{
 int height=ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0);
 int width=ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0);
 Print("CHART_HEIGHT_IN_PIXELS =",height,"pixels");
 Print("CHART_WIDTH_IN_PIXELS =",width,"pixels");
}
```

© 2000–2025, MetaQuotes Ltd.

ChartGetString

指定されたチャートの対応するプロパティの値を返します。チャートプロパティは string 型でなければなりません。この関数の呼び出し方は 2 つあります。

1. 直接プロパティ値を返します。

```
string  ChartGetString(
   long                          chart_id,         // チャート識別子
   ENUM_CHART_PROPERTY_STRING    prop_id           // プロパティ識別子
   );
```

2. 関数の成功に応じて、true または false を返します。成功した場合、プロパティの値は、参照によって渡された目的変数 string_var に配置されます。

```
bool  ChartGetString(
   long                          chart_id,         // チャート識別子
   ENUM_CHART_PROPERTY_STRING    prop_id,          // プロパティ識別子
   string&                       string_var        // プロパティのターゲット変数 property
   );
```

パラメータ

chart_id

[in] チャート識別子。(0 は現在のチャート)

prop_id

[in] チャートプロパティ識別子。値は ENUM_CHART_PROPERTY_STRING のいずれかです。

string_var

[out] 要求されたプロパティの string 型のターゲット変数

戻り値

string 型の値

2 番目の呼び方では、指定されたプロパティが利用可能であり、その値が string_var 変数に置かれている場合には、trueを返します。そうでない場合は falseを返します。エラーについての追加情報を取得するには、GetLastError()関数を呼び出します。

注意事項

ChartGetString は Comment または ChartSetString 関数を使用してチャートにプロットされたコメントを読むために使用されます。

この関数は同期型です。つまり、呼び出し前にチャートキューに追加されたすべてのコマンドの実行を待機します。

例:

```
void OnStart()
  {
   ChartSetString(0,CHART_COMMENT,"Test comment.\nSecond line.\nThird!");
   ChartRedraw();
```

© 2000-2025, MetaQuotes Ltd.

```
   Sleep(1000);
   string comm=ChartGetString(0,CHART_COMMENT);
   Print(comm);
  }
```

参照

コメント、ChartSetString

© 2000-2025, MetaQuotes Ltd.

ChartNavigate

チャート内の指定された位置を基準に指定された足数によって指定されたチャートのシフトを実行します。

```
bool  ChartNavigate(
   long                  chart_id,      // チャート識別子
   ENUM_CHART_POSITION   position,      // 位置
   int                   shift=0        // シフト値
   );
```

パラメータ

chart_id

[in] チャート識別子。(0 は現在のチャート)

position

[in] シフトを実行するチャートの位置 ENUM_CHART_POSITION のいずれかの値を持ちます。

shift=0

[in] チャートをシフトするバーの数 正の値は(チャートの最後に向けた)右シフトを意味し、負の値は(グラフの先頭に向けた)左シフトを意味します。ゼロの値はチャートの端に移動するために使用することが出来ます。

戻り値

成功の場合は true、それ以外の場合は false

例:

```
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 現在のチャートのハンドルを取得
   long handle=ChartID();
   string comm="";
   if(handle>0) // 成功した場合、追加的にチャートを設定する
     {
      //--- オートスクロールを無効にする
      ChartSetInteger(handle,CHART_AUTOSCROLL,false);
      //--- チャートの右の境界からのシフトを設定する
      ChartSetInteger(handle,CHART_SHIFT,true);
      //--- ローソク足を描画する
      ChartSetInteger(handle,CHART_MODE,CHART_CANDLES);
      //--- ティックボリュームの表示モードを設定する
      ChartSetInteger(handle,CHART_SHOW_VOLUMES,CHART_VOLUME_TICK);

      //--- Comment() に出力するテキストを準備する
      comm="Scroll 10 bars to the right of the history start";
      //--- コメントを表示する
      Comment(comm);
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 履歴開始の右に10足分スクロールする
      ChartNavigate(handle,CHART_BEGIN,10);
      //--- チャート上で最初に見えるバーの番号を得る( 番号付けは時系列と同様)
      long first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
      //--- 改行文字を追加する
      comm=comm+"\r\n";
      //--- コメントに追加する
      comm=comm+"The first bar on the chart is number "+IntegerToString(first_bar)
+"\r\n";
      //--- コメントを表示する
      Comment(comm);
      //--- 5秒待ってチャートの動きを見る
      Sleep(5000);

      //--- コメントテキストに追加する
      comm=comm+"\r\n"+"Scroll 10 bars to the left of the right chart border";
      Comment(comm);
      //--- チャートの右の境界の左に10足分スクロールする
      ChartNavigate(handle,CHART_END,-10);
      //--- チャート上で最初に見えるバーの番号を得る( 番号付けは時系列と同様)
      first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
      comm=comm+"\r\n";
      comm=comm+"The first bar on the chart is number "+IntegerToString(first_bar)
+"\r\n";
      Comment(comm);
      //--- 5秒待ってチャートの動きを見る
      Sleep(5000);

      //--- チャートのスクロールの新しいブロック
      comm=comm+"\r\n"+"Scroll 300 bars to the right of the history start";
      Comment(comm);
      //--- 履歴開始の右に 300 足分スクロールする
      ChartNavigate(handle,CHART_BEGIN,300);
      first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
      comm=comm+"\r\n";
      comm=comm+"The first bar on the chart is number "+IntegerToString(first_bar)
+"\r\n";
      Comment(comm);
      //--- 5秒待ってチャートの動きを見る
      Sleep(5000);

      //--- チャートのスクロールの新しいブロック
      comm=comm+"\r\n"+"Scroll 300 bars to the left of the right chart border";
      Comment(comm);
      //--- チャートの右の境界の左に 300 足分スクロールする
      ChartNavigate(handle,CHART_END,-300);
      first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
      comm=comm+"\r\n";
```

© 2000-2025, MetaQuotes Ltd.

```
      comm=comm+"The first bar on the chart is number "+IntegerToString(first_bar)
+"\r\n";
      Comment(comm);
     }
  }
```

© 2000–2025, MetaQuotes Ltd.

ChartID

現在のチャートの ID を返します。

```
long  ChartID();
```

戻り値

long 型の値

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- チャート識別子の変数
   long curr_chart=ChartFirst();
   int i=0;
   //--- 最初のチャートのデータを操作ログに出力する
   PrintFormat("Chart[%d] ID: %I64d,  symbol: %s", i, curr_chart,
ChartSymbol(curr_chart));

//--- 開いているチャートの制限に達するまで（CHARTS_MAX）
   while(!IsStopped() && i < CHARTS_MAX)
     {
      //--- チャートカウンタを増加する
      i++;
      //--- 前のチャートのＩＤに基づいて次のチャートのＩＤを取得する
      curr_chart=ChartNext(curr_chart);

      //--- チャートリストの最後に達したらループを終了する
      if(curr_chart<0)
         break;

   //--- 次のチャートのデータを操作ログに出力する
      PrintFormat("Chart[%d] ID: %I64d,  symbol: %s", i, curr_chart,
ChartSymbol(curr_chart));
     }
   /*
  結果:
   Chart[0] ID: 133246248352168440,  symbol: EURUSD
   Chart[1] ID: 133346697706632015,  symbol: USDJPY
   Chart[2] ID: 133246248352168439,  symbol: GBPUSD
   Chart[3] ID: 133346697706632009,  symbol: RU000A103661
   Chart[4] ID: 133346697706632010,  symbol: AEM4
   Chart[5] ID: 133346697706632011,  symbol: AA.SPB
   Chart[6] ID: 133346697706632012,  symbol: ALLFUTMIX
   Chart[7] ID: 133346697706632013,  symbol: EURUSD
   Chart[8] ID: 133346697706632014,  symbol: SBER
```

© 2000-2025, MetaQuotes Ltd.

```
   */
  }
```

© 2000–2025, MetaQuotes Ltd.

ChartIndicatorAdd

指定されたチャートウィンドウに指定されたハンドルと指標を追加します。指標とチャートは同一のシンボルと時間軸に生成されるべきです。

```
bool  ChartIndicatorAdd(
   long  chart_id,                  // チャート識別子
   int   sub_window                 // サブウィンドウ番号
   int   indicator_handle           // 指標ハンドル
   );
```

パラメータ

chart_id

[in] チャート識別子。(0 は現在のチャート)

sub_window

[in] チャートサブウィンドウの番号 (0 はメインチャートウィンドウ) 新しいウィンドウで指標を追加するには、パラメータは、CHART_WINDOWS_TOTALに等しく、最後の既存のウィンドウの指標より 1 大きくなければなりません。パラメータの値が CHART_WINDOWS_TOTALよりも大きい場合、新しいウィンドウが作成されず、表示が追加されません。

indicator_handle

[in] 指標ハンドル

戻り値

成功の場合は true、それ以外の場合は false。エラー情報を取得するには、GetLastError() 関数が呼ばれます。エラー 4114 は、チャートとコメントを追加した指標の記号または時間軸が異なることを意味します。

注意事項

サブウィンドウに描画されるべき指標がメインチャートウィンドウに適応されると、指標はリスト内には存在しても表示されないことがあります (例えば、内蔵された iMACD または #property indicator_separate_window プロパティの指定されたカスタム指標) これは、指標の規模は価格チャートの規模とは異なり、適用された指標の値は、価格チャートの表示範囲に収まらないことを意味します。この場合、GetLastError() はエラーが存在しないことを示すゼロコードを返します。このような「見えない」指標の値は、データウィンドウに表示し、他のMQL5アプリケーションから受け取ることが出来ます。

例 :

```
#property description "Expert Advisor demonstrating the work with
ChartIndicatorAdd() function."
#property description "After launching on the chart (and receiving the error in
Journal), open"
#property description "the Expert Advisor's properties and specify correct <symbol>
and <period> parameters."
#property description "MACD indicator will be added on the chart."

//--- 入力パラメータ
input string          symbol="AUDUSD";     // 銘柄名
input ENUM_TIMEFRAMES period=PERIOD_M12;   // 時間軸
input int     fast_ema_period=12;          // MACDが高速期間
```

© 2000-2025, MetaQuotes Ltd.

```
input int     slow_ema_period=26;           // MACDが低速期間
input int       signal_period=9;            // シグナル期間
input ENUM_APPLIED_PRICE apr=PRICE_CLOSE; // MACD計算に使用する値の型

int indicator_handle=INVALID_HANDLE;
//+------------------------------------------------------------------+
//| エキスパート初期化に使用される関数                                |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   indicator_handle=iMACD
(symbol,period,fast_ema_period,slow_ema_period,signal_period,apr);
//--- チャートに指標を追加しようとする
   if(!AddIndicator())
     {
      //--- AddIndicator() 関数が、チャートに指標を追加することを拒否した
      int answer=MessageBox("Do you want to add MACD on the chart anyway?",
                            "Incorrect symbol and/or time frame for adding the
indicator",
                            MB_YESNO //「はい(Y)」と「いいえ(N)」ボタンが表示される
                            );
      //--- ユーザが ChartIndicatorAdd() の不正な使用方法を主張する場合
      if(answer==IDYES)
        {
         //--- まず、操作ログにこのことを記載する
         PrintFormat("Attention! %s: Trying to add MACD(%s/%s) indicator on %s/%s
chart. Receiving error 4114",
                     __FUNCTION__
,symbol,EnumToString(period),_Symbol,EnumToString(_Period));
         //--- 新たなサブウィンドウの数を受け取り指標を追加しようとする
         int subwindow=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
         //--- 試んでエラーを発生する
         if(!ChartIndicatorAdd(0,subwindow,indicator_handle))
            PrintFormat("Failed to add MACD indicator on %d chart window. Error code
 %d",
                        subwindow,GetLastError());
        }
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| エキスパートティック関数                                         |
//+------------------------------------------------------------------+
void OnTick()
  {
// エキスパートアドバイザーは何もしない
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|  チェックしてチャートに指標を追加する関数                                  |
//+------------------------------------------------------------------+
bool AddIndicator()
  {
//--- 表示メッセージ
   string message;
//--- 指標シンボルとチャートシンボルが一致するかどうかを確認する
   if(symbol!=_Symbol)
     {
      message="Displaying the use of Demo_ChartIndicatorAdd() function:";
      message=message+"\r\n";
      message=message+"Unable to add the indicator calculated on another symbol on the chart.";
      message=message+"\r\n";
      message=message+"Specify the chart symbol in Expert Advisor's property - "+_Symbol+".";
      Alert(message);
      //--- 早期の終了。指標はチャート上に追加されない
      return false;
     }
//---指標とチャートの時間軸が一致しているかどうかを確認する
   if(period!=_Period)
     {
      message="Unable to add the indicator calculated on another time frame on the chart.";
      message=message+"\r\n";
      message=message+"Specify the chart time frame in Expert Advisor properties - "+EnumToString(_Period)+".";
      Alert(message);
      //--- 早期の終了。指標はチャート上に追加されない
      return false;
     }
//--- 全てのチェックが完了し、シンボルと指標時間軸はチャートに一致する
   if(indicator_handle==INVALID_HANDLE)
     {
      Print(__FUNCTION__,"  Creating MACD indicator");
      indicator_handle=iMACD(symbol,period,fast_ema_period,slow_ema_period,signal_period,apr);
      if(indicator_handle==INVALID_HANDLE)
        {
         Print("Failed to create MACD indicator. Error code ",GetLastError());
        }
     }
//--- エラーコードをリセットする
   ResetLastError();
//--- 指標をチャートに適用する
   Print(__FUNCTION__,"  Adding MACD indicator on the chart");
   Print("MACD is generated on ",symbol,"/",EnumToString(period));
//--- MACD指標が付加された新たなサブウィンドウの番号を受け取る
   int subwindow=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
```

© 2000-2025, MetaQuotes Ltd.

```
    PrintFormat("Adding MACD indicator on %d chart window",subwindow);
    if(!ChartIndicatorAdd(0,subwindow,indicator_handle))
      {
       PrintFormat("Failed to add MACD indicator on %d chart window. Error code  %d",
                   subwindow,GetLastError());
      }
//--- 指標が正常に追加された
    return(true);
   }
```

参照

ChartIndicatorDelete(), ChartIndicatorName(), ChartIndicatorsTotal(), iCustom(), IndicatorCreate()

© 2000–2025, MetaQuotes Ltd.

ChartIndicatorDelete

指定された名称の指標を指定されたチャートウィンドウから削除します。

```
bool  ChartIndicatorDelete(
   long           chart_id,              // チャート識別子
   int            sub_window             // サブウィンドウ番号
   const string   indicator_shortname    // 指標短縮名
   );
```

パラメータ

chart_id

[in] チャート識別子。(0 は現在のチャート)

sub_window

[in] チャートサブウィンドウの番号。(0 はメインチャートウィンドウ)

const indicator_shortname

[in] IndicatorSetString() 関数で INDICATOR_SHORTNAMEプロパティに設定した指標の短縮名指標の短縮名を取得するには ChartIndicatorName() 関数が呼ばれます。

戻り値

指標の削除が成功した場合は true、その他の場合は false。エラーの詳細を取得するには GetLastError() 関数を使用します。

注意事項

同一の短縮名を持つ指標がチャートサブウィンドウに 2 つ存在する場合、行の最初のものは削除されます。

このチャート上の他の指標が削除される指標の値に基づいている場合は、そのような指標も削除されます。

iCustom() 及び IndicatorCreate() 関数を使用して指標を作成する際、指標の短縮名とファイル名を混同しないでください。指標の短縮名が明示的に設定されていない場合、指標のソースコードを含むファイルの名称は、コンパイル時に指定されます。

指標のチャートからの削除は、その演算部が端末のメモリから削除されることを意味するものではありません。指標ハンドルを解除するには IndicatorRelease() 関数が使用されます。

指標の短縮名は正しく形成されなければなりません。短縮名は IndicatorSetString() 関数を使用して INDICATOR_SHORTNAMEプロパティに書かれます。ChartIndicatorDelete() 関数によってチャートから削除される指標は短縮名によって識別されるので、短縮名に指標の全ての入力パラメータの値を含むようお勧めします。

初期化の失敗後に指標を削除する例：

```
//+------------------------------------------------------------------+
//|                                   Demo_ChartIndicatorDelete.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
```

© 2000–2025, MetaQuotes Ltd.

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- ヒストグラムをプロットする
#property indicator_label1  "Histogram"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 入力パラメータ
input int      first_param=1;
input int      second_param=2;
input int      third_param=3;
input bool     wrong_init=true;
//--- 指標バッファ
double         HistogramBuffer[];
string         shortname;
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
   int res=INIT_SUCCEEDED;
//--- HistogramBuffer 配列を指標バッファに関係づける
   SetIndexBuffer(0,HistogramBuffer,INDICATOR_DATA);
//--- 入力パラメータに基づいて、指標短縮名を構築する
   shortname=StringFormat("Demo_ChartIndicatorDelete(%d,%d,%d)",
                          first_param,second_param,third_param);
   IndicatorSetString(INDICATOR_SHORTNAME,shortname);
//--- 指標の強制終了が設定されている場合、ゼロ以外の値を返す
   if(wrong_init) res=INIT_FAILED;
   return(res);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- ループを操作するための開始位置
   int start=prev_calculated-1;
   if(start<0) start=0;
//--- 指標バッファに値を記入する
   for(int i=start;i<rates_total;i++)
     {
      HistogramBuffer[i]=close[i];
     }
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Deinit イベントハンドラ                                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   PrintFormat("%s: Deinitialization reason code=%d",__FUNCTION__,reason);
   if(reason==REASON_INITFAILED)
     {
      PrintFormat("An indicator with a short name %s (file %s) deletes itself from
the chart",shortname,__FILE__);
      int window=ChartWindowFind();
      bool res=ChartIndicatorDelete(0,window,shortname);
      //--- ChartIndicatorDelete() 呼び出しの結果を分析する
      if(!res)
        {
         PrintFormat("Failed to delete indicator %s from window #%d. Error code %d",
                     shortname,window,GetLastError());
        }
     }
  }
```

参照

ChartIndicatorAdd(), ChartIndicatorName(), ChartIndicatorsTotal(), iCustom(), IndicatorCreate(), IndicatorSetString()

© 2000-2025, MetaQuotes Ltd.

ChartIndicatorGet

指定されたチャート]ウィンドウで指定された短縮名を持つ指標ハンドルを返します。

```
int  ChartIndicatorGet(
   long           chart_id,              // チャート識別子
   int            sub_window             // サブウィンドウ番号
   const string   indicator_shortname    // 指標短縮名
   );
```

パラメータ

chart_id

[in] チャート識別子。(0 は現在のチャート)

sub_window

[in] チャートサブウィンドウの番号 (0 はメインチャートウィンドウ)

const indicator_shortname

[in] IndicatorSetString() 関数を使用して INDICATOR_SHORTNAME プロパティに設定された指標の短縮名 指標の短縮名を取得するには ChartIndicatorName() 関数が使用されます。

戻り値

成功した場合は指標ハンドル、そうでない場合は INVALID_HANDLEを返します。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

注意事項

ChartIndicatorGet()関数を使用して取得した指標ハンドルは、内部指標使用カウンターを増やします。カウンターがゼロより大きいすべての指標は、ターミナルランタイムシステムに読み込まれたままになります。したがって、以下の例に示すように、不要になった指標ハンドルは、IndicatorRelease()を使用して、それを受け取ったのと同じプログラムですぐに明示的に解放する必要があります。別のプログラムから「放棄された」ハンドルを見つけて正しく解放することはできません。

指標を作成する場合はIndicatorSetString() 関数を使用して INDICATOR_SHORTNAME プロパティに書かれる短縮名を形成するようお気をつけください。指標は ChartIndicatorGet() 関数で短縮名によって識別されるので、.短縮名に指標の全ての入力パラメータの値を含むようお勧めします。

指標を識別するためのあと 1 つの方法は、IndicatorParameters() 関数を使用して、指定されたハンドルのパラメータリストを取得し、その後、取得された値を分析することです。

例 :

```
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
   //--- チャート上のウィンドウの数(少なくとも1つのメインウィンドウは常に存在する)
   int windows=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
   //--- 全てのウィンドウをチェックする
   for(int w=0;w<windows;w++)
```

© 2000–2025, MetaQuotes Ltd.

```
      {
       //--- このウィンドウ/サブウィンドウ内の指標の数
       int total=ChartIndicatorsTotal(0,w);
       //--- ウィンドウ内の全ての指標をみる
       for(int i=0;i<total;i++)
         {
          //--- 指標の短縮名を取得する
          string name=ChartIndicatorName(0,w,i);
          //--- 指標ハンドルを取得する
          int handle=ChartIndicatorGet(0,w,name);
          //--- ログに加える
          PrintFormat("Window=%d,  index=%d,  name=%s,  handle=%d",w,i,name,handle);
          //--- 不要になった指標ハンドルは解放するべき
          IndicatorRelease(handle);
         }
      }
  }
```

参照

ChartIndicatorAdd()、ChartIndicatorName()、ChartIndicatorsTotal()、IndicatorParameters()

© 2000-2025, MetaQuotes Ltd.

ChartIndicatorName

指定されたチャートウィンドウの指標リスト内の番号で指標の短縮名を返します。

```
string  ChartIndicatorName(
   long   chart_id,       // チャート識別子
   int    sub_window      // サブウィンドウ番号
   int    index           // チャートサブウィンドウに追加された指標のリスト内の指標のインデックス
   );
```

パラメータ

chart_id

[in] チャート識別子。(0 は現在のチャート)

sub_window

[in] チャートサブウィンドウの番号。(0 はメインチャートウィンドウ)

index

[in] 指標のリスト内の指標のインデックスインデックスの命数は、ゼロで始まります。すなわち、リストの最初の指標が 0 のインデックスを持ちます。リスト内の指標の数を得るためには ChartIndicatorsTotal() 関数が使用されます。

戻り値

IndicatorSetString() 関数でINDICATOR_SHORTNAME プロパティに設定した指標の短縮名。エラーの詳細を取得するには GetLastError() 関数を使用します。

注意事項

iCustom() 及び IndicatorCreate() 関数を使用して指標を作成する際、指標の短縮名とファイル名を混同しないでください。指標の短縮名が明示的に設定されていない場合、コンパイル時に指標のソースコードを含むファイルの名称が指定されます。

指標の短縮名は正しく形成されなければなりません。短縮名は IndicatorSetString() 関数を使用してINDICATOR_SHORTNAMEプロパティに書かれます。ChartIndicatorDelete() 関数によってチャートから削除される指標は短縮名によって識別されるので、短縮名に指標の全ての入力パラメータの値を含むようお勧めします。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 現在のチャートI Dと、メインウィンドウを含むサブウィンドウの数を取得する
   long chart_id = ChartID();
   int  wnd_total= (int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);

//--- 現在のチャートウィンドウすべてをループで繰り返し処理する
   for(int w=0; w<wnd_total; w++)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- ループインデックスで指定されたチャートウィンドウにアタッチされているインジケーターの数を取得する
      int ind_total=ChartIndicatorsTotal(chart_id, w);

      //--- 選択したチャートウィンドウのヘッダーを出力する
      PrintFormat("Chart window %d indicators: ", w);

      //--- ループに入り、選択されたウィンドウにアタッチされているすべてのインジケーターの名前を変数に書き込む
      string ind_names="";
      for(int i=0; i<ind_total; i++)
        {
         ind_names+="  "+ChartIndicatorName(chart_id, w, i)+(i<ind_total-1 ? "\n":
"");
        }
      //--- 指定されたチャートウィンドウに添付されているすべてのインジケーターの名前のリストを操作ログに出力する
      Print(ind_names);
     }
   /*
   結果:
   Chart window 0 indicators:
     AMA(9,2,30)
     SAR(0.02,0.20)
     Fractals
   Chart window 1 indicators:
     RSI(14)
     AMA(9,2,30)
   Chart window 2 indicators:
     MFI(14)
   */
  }
```

参照

ChartIndicatorAdd()、ChartIndicatorDelete()、ChartIndicatorsTotal()、iCustom()、IndicatorCreate()、IndicatorSetString()

© 2000-2025, MetaQuotes Ltd.

ChartIndicatorsTotal

指定されたチャートウィンドウに適用された全ての指標の数を返します。

```
int  ChartIndicatorsTotal(
   long  chart_id,       // チャート識別子
   int   sub_window      // サブウィンドウ番号
   );
```

パラメータ

chart_id

[in] チャート識別子。(0は現在のチャート)

sub_window

[in] チャートサブウィンドウの番号。(0はメインチャートウィンドウ)

戻り値

指定されたチャートウィンドウの指標の数。エラーの詳細を取得するにはGetLastError() 関数を使用します。

注意事項

この関数は、チャートに接続された全ての指標を検索することが出来ます。チャートのウィンドウの数の合計は ChartGetInteger() 関数を使用してCHART_WINDOWS_TOTAL プロパティから得られます。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 現在のチャートI Dと、メインウィンドウを含むサブウィンドウの数を取得する
   long chart_id = ChartID();
   int  wnd_total= (int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);

//--- 現在のチャートウィンドウすべてをループで繰り返し処理する
   for(int w=0; w<wnd_total; w++)
     {
      //--- ループインデックスで指定されたチャートウィンドウにアタッチされているインジケーターの数を取得する
      int ind_total=ChartIndicatorsTotal(chart_id, w);

      //--- 指定されたチャートウィンドウに添付されているインジケーターの数を取得する
      PrintFormat("Chart ID %I64d, subwindow %d. Attached indicators: %d", chart_id,
w, ind_total);
     }
   /*
   結果:
   Chart ID 133246248352168439, subwindow 0. Attached indicators: 3
   Chart ID 133246248352168439, subwindow 1. Attached indicators: 2
```

© 2000-2025, MetaQuotes Ltd.

```
   Chart ID 133246248352168439, subwindow 2. Attached indicators: 1
   */
  }
```

参照

ChartIndicatorAdd()、ChartIndicatorDelete()、iCustom()、IndicatorCreate()、IndicatorSetString()

© 2000–2025, MetaQuotes Ltd.

ChartWindowOnDropped

エキスパート アド バイザーやスクリプト がドロップされたチャート サブウィンドウチャート の番号（インデックス）を返します。（ 0 はメインチャート ウィンドウ）

```
int  ChartWindowOnDropped();
```

戻り値

int 型の値

例 :

```
   int myWindow=ChartWindowOnDropped();
   int windowsTotal=ChartGetInteger(0,CHART_WINDOWS_TOTAL);
   Print("Script is running on the window #"+myWindow+
         ". Total windows on the chart "+ChartSymbol()+":",windowsTotal);
```

参照

ChartPriceOnDropped、ChartTimeOnDropped、ChartXOnDropped、ChartYOnDropped

© 2000–2025, MetaQuotes Ltd.

ChartPriceOnDropped

エキスパート アド バイザーやスクリプト がドロップされたチャート 点 の価 格 座 標 を返します。

```
double  ChartPriceOnDropped();
```

戻り値

double 型 の値

例 :

```
   double p=ChartPriceOnDropped();
   Print("ChartPriceOnDropped() = ",p);
```

参 照

ChartXOnDropped、ChartYOnDropped

© 2000–2025, MetaQuotes Ltd.

ChartTimeOnDropped

エキスパート アド バイザーやスクリプト がドロップされたチャート 点 の価 格 座 標 を返します。

```
datetime  ChartTimeOnDropped();
```

戻り値

datetime 型の値

例 :

```
   datetime t=ChartTimeOnDropped();
   Print("Script was dropped on the "+t);
```

参照

ChartXOnDropped、ChartYOnDropped

© 2000–2025, MetaQuotes Ltd.

ChartXOnDropped

エキスパート アドバイザーやスクリプトがドロップされたチャートのポイントの X 座標を返します。

```
int  ChartXOnDropped();
```

戻り値

X 座標値

注意事項

X 軸方向は左から右です。

例:

```
   int X=ChartXOnDropped();
   int Y=ChartYOnDropped();
   Print("(X,Y) = ("+X+","+Y+")");
```

参照

ChartWindowOnDropped、ChartPriceOnDropped、ChartTimeOnDropped

© 2000-2025, MetaQuotes Ltd.

ChartYOnDropped

エキスパートアドバイザーやスクリプトがドロップされたチャート点のY座標を返します。

```
int  ChartYOnDropped();
```

戻り値

Y座標値

注意事項

Y軸方向は上から下です。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- マウスを使用してスクリプトがドロップされたチャートポイントの X 座標と Y 座標を取得して出力する
   int x=ChartXOnDropped();
   int y=ChartYOnDropped();
   PrintFormat("Script dropped to coordinates X = %d, Y = %d", x, y);
   /*
   結果:
   Script dropped to coordinates X = 429, Y = 114
   */
  }
```

参照

ChartWindowOnDropped、ChartPriceOnDropped、ChartTimeOnDropped

© 2000-2025, MetaQuotes Ltd.

ChartSetSymbolPeriod

指定されたチャートの銘柄値と期間を変更します。この関数は非同期的です。すなわち、コマンドを送信後に実行完了は待たれません。コマンドは、チャートメッセージキューに追加され、以前の全てのコマンドの処理後に実行されます。

```
bool  ChartSetSymbolPeriod(
   long             chart_id,      // チャート識別子
   string           symbol,        // 銘柄名
   ENUM_TIMEFRAMES  period         // 期間
   );
```

パラメータ

chart_id

[in] チャート識別子。(0 は現在のチャート)

symbol

[in] チャートシンボル(NULL はエキスパートアドバイザーが接続されている現在のチャートのシンボル)

period

[in] チャート期間(時間軸)。ENUM_TIMEFRAMES のいずれかの値。(0 は現在のチャート期間)

戻り値

コマンドがチャートキューに追加された場合は true、そうでない場合は false。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

注意事項

銘柄/期間の変更はチャートに取り付けられたエキスパートアドバイザーの再初期化につながります。

同じシンボルと時間枠でのChartSetSymbolPeriodの呼び出しはチャートの更新に使われることができます(ターミナルのRefreshコマンドと同じ)。 チャートの更新によって、取り付けられているインディケータの再計算がトリガされます。 したがって、ティックがない場合でも(たとえば、週末に)チャート上のインディケータを計算することができます。

例：

```
#define   SYMBOL    "GBPUSD"
#define   PERIOD    PERIOD_H1

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 現在のチャートID、銘柄、期間を取得する
   long             chart_id= ChartID();
   string           symbol  = Symbol();
   ENUM_TIMEFRAMES period  = Period();

//--- 現在のチャートの銘柄と期間を SYMBOL と PERIOD で設定されたものに置き換えたことを操作ログ報告する
```

© 2000-2025, MetaQuotes Ltd.

```
   PrintFormat("Change the %s symbol and the %s period of the chart %I64u to %s %s",
               symbol, TimeframeDescrioption(period), chart_id, SYMBOL,
TimeframeDescrioption(PERIOD));

//--- チャートの銘柄と期間を変更する
   ChartSetSymbolPeriod(chart_id, SYMBOL, PERIOD);
   /*
   結果:
   Change the EURUSD symbol and the M1 period of the chart 133246248352168440 to
GBPUSD H1
   */
  }
//+------------------------------------------------------------------+
//| チャートの期間の説明を返す                                       |
//+------------------------------------------------------------------+
string TimeframeDescrioption(const ENUM_TIMEFRAMES timeframe)
  {
   return(StringSubstr(EnumToString(timeframe), 7));
  }
```

参照

ChartSymbol、ChartPeriod

© 2000–2025, MetaQuotes Ltd.

ChartScreenShot

この関数は、指定された拡張子に応じてGIF、PNGまたはBMP形式で現在のチャートのスクリーンショットを提供します。

```
bool  ChartScreenShot(
   long             chart_id,                  // チャート識別子
   string           filename,                  // 銘柄名
   int              width,                     // 幅
   int              height,                    // 高さ
   ENUM_ALIGN_MODE  align_mode=ALIGN_RIGHT     // 整列の種類
   );
```

パラメータ

chart_id

[in] チャート識別子。(0 は現在のチャート)

filename

[in] スクリーンショットファイル名。63 文字を超えることは出来ません。スクリーンショットファイルは ¥Files ディレクトリに置かれます。

幅

[in] ピクセル単位でのスクリーンショットの幅

height

[in] ピクセル単位でのスクリーンショットの高さ

align_mode=ALIGN_RIGHT

[in] スクリーンショットの出力モード値は ENUM_ALIGN_MODE 列挙のいずれかです。ALIGN_RIGHT は右マージンに揃えることを意味します。ALIGN_LEFT は左マージンに揃えることです。

戻り値

成功の場合は true、それ以外の場合は false。

注意事項

特定の位置からチャートからスクリーンショットを取る必要がある場合は、最初に ChartNavigate() 関数を使用してグラフを配置する必要があります。スクリーンショットの水平方向のサイズがウィンドウよりも小さい場合には、align_modeの設定に応じてチャートウィンドウの右または左の部分が出力されます。

例:

```
#property description "The Expert Advisor demonstrates how to create a series of
screenshots of the current"
#property description "chart using the ChartScreenShot() function. For convenience,
the file name is"
#property description "shown on the chart. The height and width of images is defined
through macros."

#define        WIDTH  800     // ChartScreenShot() を呼ぶ画像幅
#define        HEIGHT 600     // ChartScreenShot() を呼ぶ画像の縦幅
```

© 2000–2025, MetaQuotes Ltd.

```
//--- 入力パラメータ
input int      pictures=5;     // シリーズ内の画像の数
int            mode=-1;        // -1 は左へ、1は右端へのシフトを示す
int            bars_shift=300;// ChartNavigate() を使用してチャートをスクロールする際のバーの数
//+------------------------------------------------------------------+
//| エキスパート初期化に使用される関数                                |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- チャートの自動スクロールを無効にする
   ChartSetInteger(0,CHART_AUTOSCROLL,false);
//--- チャートの右端のシフトを設定する
   ChartSetInteger(0,CHART_SHIFT,true);
//--- ローソク足チャートを表示する
   ChartSetInteger(0,CHART_MODE,CHART_CANDLES);
//---
   Print("Preparation of the Expert Advisor is completed");
  }
//+------------------------------------------------------------------+
//| エキスパートティック関数                                         |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| ChartEvent 関数                                                  |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- 関数の名称、呼び出し時間とイベント識別子を表示する
   Print(__FUNCTION__,TimeCurrent(),"   id=",id,"   mode=",mode);
//--- CHARTEVENT_CLICK イベントをハンドルする（"チャート上でのマウスクリック"）
   if(id==CHARTEVENT_CLICK)
     {
      //--- チャートの縁から初期シフト
      int pos=0;
      //--- チャートの左縁での動作
      if(mode>0)
        {
         //--- チャートを左縁にスクロールする
         ChartNavigate(0,CHART_BEGIN,pos);
         for(int i=0;i<pictures;i++)
           {
            //--- チャートに表示するテキスト及びファイル名を準備する
```

© 2000-2025, MetaQuotes Ltd.

```
            string name="ChartScreenShot"+"CHART_BEGIN"+string(pos)+".gif";
            //--- チャート名をコメントとして表示する
            Comment(name);
            //--- terminal_directory¥MQL5¥Files¥ にチャートのスクリーンショットファイルを保存する
            if(ChartScreenShot(0,name,WIDTH,HEIGHT,ALIGN_LEFT))
               Print("We've saved the screenshot ",name);
            //---
            pos+=bars_shift;
            //--- ユーザにチャートの新しい部分を見る時間を与える
            Sleep(3000);
            //--- 現在位置 bars_shift バーからチャートを右にスクロールする
            ChartNavigate(0,CHART_CURRENT_POS,bars_shift);
           }
         //--- モードを反対に変更する
         mode*=-1;
        }
      else // チャートの右縁での動作
        {
         //--- チャートを右縁にスクロールする
         ChartNavigate(0,CHART_END,pos);
         for(int i=0;i<pictures;i++)
           {
            //--- チャートに表示するテキスト及びファイル名を準備する
            string name="ChartScreenShot"+"CHART_END"+string(pos)+".gif";
            //--- チャート名をコメントとして表示する
            Comment(name);
            //--- terminal_directory¥MQL5¥Files¥ にチャートのスクリーンショットファイルを保存する
            if(ChartScreenShot(0,name,WIDTH,HEIGHT,ALIGN_RIGHT))
               Print("We've saved the screenshot ",name);
            //---
            pos+=bars_shift;
            //--- ユーザにチャートの新しい部分を見る時間を与える
            Sleep(3000);
            //--- 現在位置 bars_shift バーからチャートを右にスクロールする
            ChartNavigate(0,CHART_CURRENT_POS,-bars_shift);
           }
         //--- モードを反対に変更する
         mode*=-1;
        }
     } // CHARTEVENT_CLICK イベント処理の終了
//--- OnChartEvent() ハンドラの終わり
  }
```

参照

ChartNavigate()、リソース

© 2000-2025, MetaQuotes Ltd.

取引関数

これは取引アクティビティを管理する関数群です。

プラットフォームの取引機能の研究に進む前に、注文、約定、及びポジションといった基本的な用語を明確に理解しておく必要があります。

- まず注文とはブローカーに出された金融製品の売買の指示です。注文の種類には、主に成行と未決の2種類があります。更に、特別な決済指値と決済逆指値レベルがあります。
- 約定とは有価証券の商業的な交換(売買)です。買いは買値(アスク)、売りは売値(ビッド)で行われます。約定は成行注文の実行または未決注文のトリガによって起こります。成行注文の実行は複数の約定につながることもあります。
- ポジションとは取引の義務、つまり金融製品の買いまたは売りの契約です。ロングポジションとは、価格の上昇の期待によって買われた有価証券です。ショートポジションは、将来的な価格の下落を期待した有価証券を提供する義務です。

取引操作に関する一般的な情報は、クライアントターミナルのヘルプでご覧になれます。

取引関数はエキスパートアドバイザーと指標で動作出来ます。取引関数は、エキスパートアドバイザーまたはスクリプトのプロパティで「自動トレーディングを許可する」チェックボックスがチェックされている時のみ呼ぶことが出来ます。

取引は取引許可セクションで説明されている様々な要因に応じて許可または禁止されます。

関数	アクション
OrderCalcMargin	指定された注文の種類に必要な証拠金を預金通貨で計算します。
OrderCalcProfit	渡されたパラメータに基づいて、利益を預金通貨で計算します。
OrderCheck	取引操作を実行するための充分な資金があるかをチェックします。
OrderSend	サーバに取引リクエストを送ります。
OrderSendAsync	取引サーバの返答を待たずに取引リクエストを非同期的に送ります。
PositionsTotal	ポジションの数を返します。
PositionGetSymbol	ポジションに対応するシンボルを返します。
PositionSelect	操作を続けるためにポジションを選択します。
PositionSelectByTicket	Selects a position to work with by the ticket number specified in it
PositionGetDouble	リクエストされたポジションプロパティ(double)を返します。
PositionGetInteger	リクエストされたポジションプロパティ(datetime または int)を返します。
PositionGetString	リクエストされたポジションプロパティ(string)を返します。
PositionGetTicket	Returns the ticket of the position with the specified index in the list of open positions
OrdersTotal	注文数を返します。
OrderGetTicket	注文のチケットを返します。
OrderSelect	操作を続けるために注文を選択します。

© 2000–2025, MetaQuotes Ltd.

関数	アクション
OrderGetDouble	リクエストされた注文プロパティ(double)を返します。
OrderGetInteger	リクエストされた注文プロパティ(datetime または int)を返します。
OrderGetString	リクエストされた注文プロパティ(string)を返します。
HistorySelect	指定されたサーバタイム期間のトランザクションと注文履歴を取得します。
HistorySelectByPosition	指定されたポジション識別子を持った取引履歴をリクエストします。
HistoryOrderSelect	操作を続けるために履歴内の注文を選択します。
HistoryOrdersTotal	履歴内の注文の数を返します。
HistoryOrderGetTicket	履歴内の注文に対応するチケットを返します。
HistoryOrderGetDouble	履歴内で指定された double 注文プロパティを返します。
HistoryOrderGetInteger	履歴内で指定された datetime または int 注文プロパティを返します。
HistoryOrderGetString	履歴内で指定された string 注文プロパティを返します。
HistoryDealSelect	更に適切な関数で呼び出すために履歴内の取引を選択します。
HistoryDealsTotal	履歴内の取引数を返します。
HistoryDealGetTicket	履歴内の取引に対応するチケットを返します。
HistoryDealGetDouble	履歴内で指定された約定の double 型プロパティを返します。
HistoryDealGetInteger	履歴内で指定された約定の datetime または int プロパティを返します。
HistoryDealGetString	履歴内で指定された約定の string 型プロパティを返します。

© 2000–2025, MetaQuotes Ltd.

OrderCalcMargin

この関数は、現在未決算の注文と未決済ポジションを考慮せずに、現在の口座で、指定された注文の種類に現在の市場環境で必要な証拠金を預金通貨で計算します。これは、計画された取引操作のための証拠金の評価を可能にします。値は口座の預金通貨で返されます。

```
bool  OrderCalcMargin(
   ENUM_ORDER_TYPE       action,           // 注文の種類
   string                symbol,           // 銘柄名
   double                volume,           // ボリューム
   double                price,            // 始値
   double&               margin            // 証拠金取得に使用される変数
   );
```

パラメータ

action

[in] ENUM_ORDER_TYPE 列挙のいずれかの値を持つ注文の種類

symbol

[in] 銘柄名

volume

[in] 取引操作のボリューム

price

[in] 始値

margin

[out] 関数の実行が成功した場合にリクエストされた証拠金の値が配置される変数。計算は、現在未決算の注文と未決済ポジションを考慮せずに行われます。証拠金の値は多くの要因に依存し、市場環境によって異なることがあります。

戻り値

成功の場合は true、それ以外の場合は false。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

例：

```
#define   VOLUME      1.0   // 注文量
#define   DEVIATION   100   //予約注文を設定するための距離
#define   STOP_LIMIT  50    // ストップリミット注文の距離

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string currency=AccountInfoString(ACCOUNT_CURRENCY);
   int array_types[8]={ORDER_TYPE_BUY,
                       ORDER_TYPE_SELL,
```

© 2000-2025, MetaQuotes Ltd.

```
                        ORDER_TYPE_BUY_LIMIT,
                        ORDER_TYPE_SELL_LIMIT,
                        ORDER_TYPE_BUY_STOP,
                        ORDER_TYPE_SELL_STOP,
                        ORDER_TYPE_BUY_STOP_LIMIT,
                        ORDER_TYPE_SELL_STOP_LIMIT };

//--- 注文タイプによるループ
   for(int i=0; i<(int)array_types.Size(); i++)
     {
      //--- 注文タイプに応じて、ORDER_TYPE_BUYまたはORDER_TYPE_SELLタイプを取得する
      ENUM_ORDER_TYPE type=MarketOrderByOrderType((ENUM_ORDER_TYPE)array_types[i]);

      //--- 注文タイプに応じて、価格を取得する
      double price=PriceByOrderType(_Symbol, type);

      //--- actionで指定された注文タイプに必要な証拠金の金額を取得する
      double margin=EMPTY_VALUE;
      ResetLastError();
      if(!OrderCalcMargin(type, _Symbol, VOLUME, price, margin))
        {
         Print("OrderCalcMargin() failed. Error ",GetLastError());
         continue;
        }
      //--- 結果を操作ログに出力する
      PrintFormat("Margin required for %.2f %s order at price %.*f on %s symbol:
%.2f %s", VOLUME, OrderTypeDescription((ENUM_ORDER_TYPE)i), _Digits, price, _Symbol,
margin, currency);
     }
   /*
   結果:
   Margin required for 1.00 Buy order at price 1.31668 on GBPUSD symbol: 1316.68 USD
   Margin required for 1.00 Sell order at price 1.31661 on GBPUSD symbol: 1316.61
USD
   Margin required for 1.00 Buy Limit order at price 1.31568 on GBPUSD symbol:
1315.68 USD
   Margin required for 1.00 Sell Limit order at price 1.31761 on GBPUSD symbol:
1317.61 USD
   Margin required for 1.00 Buy Stop order at price 1.31768 on GBPUSD symbol:
1317.68 USD
   Margin required for 1.00 Sell Stop order at price 1.31561 on GBPUSD symbol:
1315.61 USD
   Margin required for 1.00 Buy Stop Limit order at price 1.31718 on GBPUSD symbol:
1317.18 USD
   Margin required for 1.00 Sell Stop Limit order at price 1.31611 on GBPUSD symbol:
1316.11 USD
   */
  }
//+------------------------------------------------------------------+
//|  BuyまたはSellの注文タイプを返す                                 |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
ENUM_ORDER_TYPE MarketOrderByOrderType(ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY  : case ORDER_TYPE_BUY_LIMIT  : case ORDER_TYPE_BUY_STOP
 : case ORDER_TYPE_BUY_STOP_LIMIT  :
        return(ORDER_TYPE_BUY);
      case ORDER_TYPE_SELL : case ORDER_TYPE_SELL_LIMIT : case ORDER_TYPE_SELL_STOP
: case ORDER_TYPE_SELL_STOP_LIMIT :
        return(ORDER_TYPE_SELL);
     }
   return(WRONG_VALUE);
  }
//+------------------------------------------------------------------+
//|  注文タイプによる始値を返す                                      |
//+------------------------------------------------------------------+
double PriceByOrderType(const string symbol, const ENUM_ORDER_TYPE order_type)
  {
   int     digits=0;
   double  point=0;
   MqlTick tick={};

//--- 銘柄のPoint値を取得する
   ResetLastError();
   if(!SymbolInfoDouble(symbol, SYMBOL_POINT, point))
     {
      Print("SymbolInfoDouble() failed. Error ", GetLastError());
      return 0;
     }

//--- 銘柄のDigits値を取得する
   long value=0;
   if(!SymbolInfoInteger(symbol, SYMBOL_DIGITS, value))
     {
      Print("SymbolInfoInteger() failed. Error ", GetLastError());
      return 0;
     }
   digits=(int)value;

//--- 銘柄ごとの最新価格を取得する
   if(!SymbolInfoTick(symbol, tick))
     {
      Print("SymbolInfoTick() failed. Error ", GetLastError());
      return 0;
     }

//--- 注文タイプに応じて価格を返す
   switch(order_type)
```

© 2000–2025, MetaQuotes Ltd.

```
     {
      case ORDER_TYPE_BUY              :  return(tick.ask);
      case ORDER_TYPE_SELL             :  return(tick.bid);
      case ORDER_TYPE_BUY_LIMIT        :  return(NormalizeDouble(tick.ask -
DEVIATION * point, digits));
      case ORDER_TYPE_SELL_LIMIT       :  return(NormalizeDouble(tick.bid +
DEVIATION * point, digits));
      case ORDER_TYPE_BUY_STOP         :  return(NormalizeDouble(tick.ask +
DEVIATION * point, digits));
      case ORDER_TYPE_SELL_STOP        :  return(NormalizeDouble(tick.bid -
DEVIATION * point, digits));
      case ORDER_TYPE_BUY_STOP_LIMIT   :  return(NormalizeDouble(tick.ask +
DEVIATION * point - STOP_LIMIT * point, digits));
      case ORDER_TYPE_SELL_STOP_LIMIT  :  return(NormalizeDouble(tick.bid -
DEVIATION * point + STOP_LIMIT * point, digits));
      default                          :  return(0);
     }
  }
//+------------------------------------------------------------------+
//|  注文タイプの説明を返す                                          |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE order_type)
  {
   switch(order_type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      case ORDER_TYPE_BUY_LIMIT        :  return("Buy Limit");
      case ORDER_TYPE_SELL_LIMIT       :  return("Sell Limit");
      case ORDER_TYPE_BUY_STOP         :  return("Buy Stop");
      case ORDER_TYPE_SELL_STOP        :  return("Sell Stop");
      case ORDER_TYPE_BUY_STOP_LIMIT   :  return("Buy Stop Limit");
      case ORDER_TYPE_SELL_STOP_LIMIT  :  return("Sell Stop Limit");
      case ORDER_TYPE_CLOSE_BY         :  return("Close By");
      default                          :  return("Unknown order type");
     }
  }
```

参照

OrderSend()、注文プロパティ、取引操作の種類

© 2000–2025, MetaQuotes Ltd.

OrderCalcProfit

この関数は、渡されたパラメータに基づいて、現在の銘柄での現在の口座の利益を計算します。これは、取引操作の結果の事前評価のために使用されます。値は口座の預金通貨で返されます。

```
bool  OrderCalcProfit(
   ENUM_ORDER_TYPE       action,          // 注文の種類（ORDER_TYPE_BUY または ORDER_TYPE_SELL）
   string                symbol,          // 銘柄名
   double                volume,          // ボリューム
   double                price_open,      // 始値
   double                price_close,     // 終値
   double&               profit           // 利益値の取得に使用される変数
   );
```

パラメータ

動作

[in] ENUM_ORDER_TYPE 列挙のいずれか（ORDER_TYPE_BUY または ORDER_TYPE_SELL）の値を持つ注文の種類

symbol

[in] 銘柄名

ボリューム

[in] 取引操作のボリューム

price_open

[in] 始値

price_close

[in] 終値

profit

[out] 関数の実行が成功した場合に計算された利益の値を格納する変数。見積りされた利益の値は多くの要因に依存し、市場環境によって異なることがあります。

戻り値

成功の場合は true、それ以外の場合は false。無効な注文の種類が指定された場合、この関数は falseを返します。エラー情報を取得するには GetLastError() 関数が呼ばれます。

例：

```
#define   VOLUME      1.0   // 注文量
#define   DEVIATION   100   // 終値までのポイント数

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   string currency_profit=SymbolInfoString(_Symbol,SYMBOL_CURRENCY_PROFIT);   // 利益通貨
   double close_price=ChartPriceOnDropped(); // the price coordinate, corresponding to the point where the script was dropped using the mouse, serves as the close price

//---
   for(int i=0; i<=ORDER_TYPE_SELL; i++)
     {
      ENUM_ORDER_TYPE order_type=(ENUM_ORDER_TYPE)i;                  // order type: 0 - Buy, 1 - Sell
      double open_price=PriceOpenByOrderType(_Symbol, order_type);    // open price: for Buy - Ask, for Sell - Bid

      //--- 始値が取得できない場合は、ループを続ける
      if(open_price==0)
         continue;

      //--- 終値がゼロの場合（スクリプトがチャートにドラッグされて起動されなかった場合）、価格を計算する
      if(close_price==0)
         close_price=(order_type==ORDER_TYPE_BUY ? open_price + DEVIATION * _Point : open_price - DEVIATION * _Point);

      //--- 渡されたパラメータに基づいて、現在の口座と市場環境に対するおおよその利益サイズを計算する
      double profit=0;
      ResetLastError();
      if(!OrderCalcProfit(order_type, _Symbol, VOLUME, open_price, close_price, profit))
        {
         Print("OrderCalcProfit() failed. Error ", GetLastError());
         continue;
        }

      //--- 受け取った利益値を操作ログに表示する
      PrintFormat("Estimated profit for %.2f %s position on %s with opening price of %.*f and closing price of %.*f: %.2f %s",
                  VOLUME, OrderTypeDescription(order_type), _Symbol, _Digits, open_price, _Digits, close_price, profit, currency_profit);

     }
   /*
   結果:
   Estimated profit for 1.00 Buy position on EURUSD with opening price of 1.10757 and closing price of 1.10881: 124.00 USD
   Estimated profit for 1.00 Sell position on EURUSD with opening price of 1.10753 and closing price of 1.10881: -128.00 USD
   */
  }
//+------------------------------------------------------------------+
//|  注文タイプによる始値を返す                                      |
//+------------------------------------------------------------------+
double PriceOpenByOrderType(const string symbol, const ENUM_ORDER_TYPE order_type)
```

© 2000–2025, MetaQuotes Ltd.

```
  {
   MqlTick tick={};

//--- 銘柄ごとの最新価格を取得する
   if(!SymbolInfoTick(symbol, tick))
     {
      Print("SymbolInfoTick() failed. Error ", GetLastError());
      return 0;
     }

//--- 注文タイプに応じて価格を返す
   switch(order_type)
     {
      case ORDER_TYPE_BUY              :  return(tick.ask);
      case ORDER_TYPE_SELL             :  return(tick.bid);
      default                          :  return(0);
     }
  }
//+------------------------------------------------------------------+
//|  注文タイプの説明を返す                                          |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE order_type)
  {
   switch(order_type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      default                          :  return("Unknown order type");
     }
  }
```

参照

OrderSend()、注文プロパティ、取引操作の種類

© 2000-2025, MetaQuotes Ltd.

OrderCheck

OrderCheck() 関数は、取引操作を実行するための充分な資金があるかをチェックします。チェックの結果は MqlTradeCheckResult 構造体のフィールドに配置されます。

```
bool  OrderCheck(
   MqlTradeRequest&        request,      // リクエスト構造体
   MqlTradeCheckResult&    result        // 結果構造体
   );
```

パラメータ

request

[in] 必要な取引アクションを記述する MqlTradeRequest 型の構造体へのポインタ

結果

[in, out] チェック結果が格納される MqlTradeCheckResult 型の構造体

戻り値

資金が足りない場合、またパラメータが不正な場合は false。基本的な構造体チェック(ポインタチェック)が成功した場合は true。しかし、これは、リクエストされた取引操作が確実に正常に実行されることの保障ではありません。関数実行結果の詳細には、*result* 構造体のフィールドが分析されるべきです。

エラー情報を取得するには GetLastError() 関数が呼ばれます。

例：

```
#define   DEVIATION      5              // allowed deviation from the price
#define   VOLUME         1.0            // order volume
#define   EXPERT_MAGIC   123            // MagicNumber
#define   DIRECTION      ORDER_TYPE_BUY // opened position direction (ORDER_TYPE_BUY
or ORDER_TYPE_SELL)

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- リクエスト、検証、および結果の構造体を宣言して初期化する
   MqlTradeRequest      request={};
   MqlTradeCheckResult check  ={};
   MqlTradeResult       result ={};

//--- リクエストパラメータを準備する
   PrepareRequest(_Symbol, DIRECTION, VOLUME, request);

//--- 取引リクエストパラメータを確認する
   ResetLastError();
   bool res=OrderCheck(request, check);
   if(!res)
```

© 2000–2025, MetaQuotes Ltd.

```
      {
       PrintFormat("Trade request verification completed with error %d\nServer
retcode: %u, comment: %s", GetLastError(), check.retcode, check.comment);
       return;
      }

//--- 取引リクエストの確認が成功した - 取引リクエスト検証構造体のフィールドの説明を表示する
   Print("Trade request verification completed successfully");
   MqlTradeCheckResultPrint(check, 14);

//--- 取引リクエストを送る
   if(!OrderSend(request, result))
      Print("OrderSend error ", GetLastError());    // if unable to send the
request, display the error code

//--- 操作についての情報
   PrintFormat("Trade request result: retcode=%u, deal=%I64u, order=%I64u",
result.retcode, result.deal, result.order);
   /*
   result with disabled auto trading in the client terminal:
   Trade request verification completed with error 4752
   Server retcode: 10027, comment: AutoTrading disabled by client

   enable auto trading and check again on a closed market:
   Experts   automated trading is enabled
   Trade request verification completed successfully
   Retcode:      0
   Balance:      10779.50 USD
   Equity:       10779.50 USD
   Profit:       0.00 USD
   Margin:       1104.79 USD
   Margin free:  9674.71 USD
   Margin level: 975.71 %
   Comment:      Done
   OrderSend error 4756
   Trade request result: retcode=10018, deal=0, order=0

   check on the open market:
   Trade request verification completed successfully
   Retcode:      0
   Balance:      10779.50 USD
   Equity:       10779.50 USD
   Profit:       0.00 USD
   Margin:       110.46 USD
   Margin free:  10669.04 USD
   Margin level: 9758.74 %
   Comment:      Done
   Trade request result: retcode=10009, deal=2777010968, order=2802818813
   */
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//|  取引リクエストのパラメータを準備する                                |
//+------------------------------------------------------------------+
void PrepareRequest(const string symbol, const ENUM_ORDER_TYPE order_type, const
double volume, MqlTradeRequest &request)
  {
   ENUM_ORDER_TYPE type=(DIRECTION !=ORDER_TYPE_BUY ? ORDER_TYPE_SELL : DIRECTION);
   double price=(DIRECTION==ORDER_TYPE_BUY ? SymbolInfoDouble(Symbol(),
SYMBOL_ASK) : SymbolInfoDouble(Symbol(), SYMBOL_BID));
//--- request parameters
   request.action    = TRADE_ACTION_DEAL; // trading operation type
   request.symbol    = symbol;            // symbol
   request.volume    = volume;            // volume
   request.type      = type;              // order type
   request.price     = price;             // open price
   request.deviation = DEVIATION;         // allowed deviation from the price
   request.magic     = EXPERT_MAGIC;      // order MagicNumber
  }
//+------------------------------------------------------------------+
//|  取引リクエストの検証結果のフィールドを                              |
//|  操作ログに表示する                                                |
//+------------------------------------------------------------------+
void MqlTradeCheckResultPrint(const MqlTradeCheckResult &check, const uint
header_width=0)
  {
//--- 口座の通貨とその小数点以下の桁数を取得する
   string currency=AccountInfoString(ACCOUNT_CURRENCY);
   int    digits  =(int)AccountInfoInteger(ACCOUNT_CURRENCY_DIGITS);

//--- ヘッダーのテキストとヘッダーフィールドの幅を定義する
//--- ヘッダーの幅が関数に渡され、ゼロと等しい場合、幅はヘッダー行のサイズ+1になる
   string header="Retcode:";
   uint w=(header_width==0 ? header.Length()+1 : header_width);
//--- 指定された幅のヘッダーで戻りコードを操作ログに表示する
   PrintFormat("%-*s%-u", w, header, check.retcode);

//--- 取引操作を実行した後の残高値を操作ログに表示する
   header="Balance:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   PrintFormat("%-*s%-.*f %s", w, header, digits, check.balance, currency);

//--- 取引操作を実行した後の資産の値を操作ログに表示する
   header="Equity:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   PrintFormat("%-*s%-.*f %s", w, header, digits, check.equity, currency);

//--- 取引操作を実行した後の浮動利益値を操作ログに表示する
   header="Profit:";
```

© 2000-2025, MetaQuotes Ltd.

```
   w=(header_width==0 ? header.Length()+1 : header_width);
   PrintFormat("%-*s%-.*f %s", w, header, digits, check.profit, currency);

//--- 必要な取引操作に必要な証拠金の金額を操作ログに表示する
   header="Margin:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   PrintFormat("%-*s%-.*f %s", w, header, digits, check.margin, currency);

//--- 取引操作を行った後に残る資本の値を操作ログに表示する
   header="Margin free:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   PrintFormat("%-*s%-.*f %s", w, header, digits, check.margin_free, currency);


//--- 必要な取引操作を完了した後に設定される証拠金レベルを操作ログに表示する
   header="Margin level:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   PrintFormat("%-*s%-.2f %%", w, header, check.margin_level);

//--- レスポンスコードとエラー説明のコメントを操作ログに表示する
   header="Comment:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   PrintFormat("%-*s%-s", w, header, check.comment);
  }
```

参照

OrderSend()、取引操作の種類、取引リクエスト結果、リクエストチェック結果の構造体、取引リクエスト結果の構造体

© 2000-2025, MetaQuotes Ltd.

OrderSend

OrderSend() 関数は取引操作の実行に使用され、リクエストを取引サーバに送信します。

```
bool  OrderSend(
   MqlTradeRequest&  request,       // 問の構造体
   MqlTradeResult&   result         // 答えの構造体
   );
```

パラメータ

request

[in] クライアントの取引アクティビティを記述する MqlTradeRequest 型の構造体へのポインタ

result

[in,out] 成功した場合（true が返される場合）取引操作の結果を表す MqlTradeResult 型の構造体へのポインタ

戻り値

基本的な構造体チェック（ポインタチェック）が成功した場合、trueが戻されます。**しかし、これは取引操作が成功することを表すものではありません。** 関数実行結果の詳細には、*result* 構造体のフィールドが分析されるべきです。

注意事項

取引リクエストは、サーバ上で複数の段階を踏みます。初めに、*request* パラメータの必須のフィールドが正しく書き入れられているかどうかがチェックされます。エラーがない場合、サーバは、更なる処理のために注文を受け付けます。注文が正常に取引サーバによって受け入れられると、OrderSend() 関数が true を返します。

取引サーバに送信する前に、リクエストをチェックすることが推奨されます。リクエストをチェックするには OrderCheck() 関数が使用されます。それは、取引を実行するための充分な資金があるかをチェックし、複数の役立つパラメータを 取引リクエストチェックの結果 に返します。

- チェックされたリクエストに含まれるエラーの情報を含むリターンコード
- 取引操作が実行された後に表示される残高値
- 取引操作が実行された後に表示される株式価値
- 取引操作が実行された後に表示される浮動小数点値
- 取引操作に必要な証拠金
- 取引操作が実行された後に残る自由資本の金額
- 取引操作が実行された後に設定される証拠金レベル
- 返信コードコメント（エラーの説明）

成行注文を送信する場合（MqlTradeRequest.action=TRADE_ACTION_DEAL）、OrderSend() 関数の正常結果は、注文が実行されたことを意味しません。この場合、'true'によって意味されるのは、注文がその後の実行のために取引システムに正常に配置されたことだけです。トレードサーバーは、OrderSend()への応答を形成する場合にデータがある場合、返された結果構造体の*deal*または*order*フィールド値に書き入れます。一般に、注文に対応する取引を実行するイベントは、OrderSend() 呼び出しに対する応答を送信した後に発生する可能性があります。したがって、いずれの取引リクエストに対しても、OrderSend() の実行結果を受け取ったときには、まず、結果構造体の*retcode* t取引サーバリレスポンスコードと*retcode_external*外部システムレスポンスコードを初めに確認するべきです。

© 2000-2025, MetaQuotes Ltd.

受け入れられた注文は、実行に必要な条件の 1 つが準備出来るまで処理を待って取引サーバに格納されます。

- 期限切れ
- 反方向のリクエストの出現
- 指定された実行価格が出現した場合の注文の実行
- 注文を取り消すリクエストの受信

注文処理時に、取引サーバは端末に取引イベント発生についてのメッセージを送り、これは OnTrade() 関数で処理出来ます。

サーバ上で OrderSend() 関数で送られたリクエストを実行した結果は OnTradeTransaction ハンドラで追跡出来ます。1 つの取引リクエストを実行する際に OnTradeTransaction ハンドラは複数回呼ばれることには留意されるべきです。

例えば、市場の買い注文を送信する際に、注文が処理され、買い注文が口座に記録されます。その後注文が実行されオープン注文の表から削除されて注文履歴に追加されます。約定履歴に追加され新しいポジションが作成されます。OnTradeTransaction 関数はこれらのイベント全部に呼び出されます。

例:

```
//--- ORDER_MAGIC の値
input long order_magic=55555;
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- これがデモ口座であることを確かめる
   if(AccountInfoInteger(ACCOUNT_TRADE_MODE)==ACCOUNT_TRADE_MODE_REAL)
     {
      Alert("Script operation is not allowed on a live account!");
      return;
     }
//--- 注文を出すか削除する
   if(GetOrdersTotalByMagic(order_magic)==0)
     {
      //--- 現在注文はないので、注文を出す
      uint res=SendRandomPendingOrder(order_magic);
      Print("Return code of the trade server ",res);
     }
   else // 注文があるので削除する
     {
      DeleteAllOrdersByMagic(order_magic);
     }
//---
  }
//+------------------------------------------------------------------+
//|  指定された ORDER_MAGIC で現在の注文数を受け取る                   |
//+------------------------------------------------------------------+
int GetOrdersTotalByMagic(long const magic_number)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   ulong order_ticket;
   int total=0;
//--- 未決注文を全部見る
   for(int i=0;i<OrdersTotal();i++)
      if((order_ticket=OrderGetTicket(i))>0)
         if(magic_number==OrderGetInteger(ORDER_MAGIC)) total++;
//---
   return(total);
  }
//+------------------------------------------------------------------+
//| 指定された ORDER_MAGIC の未決注文を全て作成する                  |
//+------------------------------------------------------------------+
void DeleteAllOrdersByMagic(long const magic_number)
  {
   ulong order_ticket;
//--- 未決注文を全部見る
   for(int i=OrdersTotal()-1;i>=0;i--)
      if((order_ticket=OrderGetTicket(i))>0)
         //--- 適切な ORDER_MAGIC を持った注文
         if(magic_number==OrderGetInteger(ORDER_MAGIC))
           {
            MqlTradeResult result={};
            MqlTradeRequest request={};
            request.order=order_ticket;
            request.action=TRADE_ACTION_REMOVE;
            OrderSend(request,result);
            //--- サーバ返答をログに書く
            Print(__FUNCTION__,": ",result.comment," reply code ",result.retcode);
           }
//---
  }
//+------------------------------------------------------------------+
//| 未決注文をランダムに設定する                                     |
//+------------------------------------------------------------------+
uint SendRandomPendingOrder(long const magic_number)
  {
//--- リクエストを準備する
   MqlTradeRequest request={};
   request.action=TRADE_ACTION_PENDING;          // 未決注文を設定する
   request.magic=magic_number;                   // ORDER_MAGIC
   request.symbol=_Symbol;                       // シンボル
   request.volume=0.1;                           // 0.1 ロットのボリューム
   request.sl=0;                                 // 決済逆指の指定なし
   request.tp=0;                                 // 決済指値の指定なし
//--- 注文の種類を形成する
   request.type=GetRandomType();                 // 注文の種類
//--- 未決注文の価格を形成する
   request.price=GetRandomPrice(request.type);   // 始値
//--- 取引リクエストを送る
```

© 2000-2025, MetaQuotes Ltd.

```
   MqlTradeResult result={};
   OrderSend(request,result);
//--- サーバ返答をログに書く
   Print(__FUNCTION__,":",result.comment);
   if(result.retcode==10016) Print(result.bid,result.ask,result.price);
//--- 取引サーバの返答のコードを返す
   return result.retcode;
  }
//+------------------------------------------------------------------+
//| 未決注文の種類をランダムに返す                                   |
//+------------------------------------------------------------------+
ENUM_ORDER_TYPE GetRandomType()
  {
   int t=MathRand()%4;
//---   0<=t<4
   switch(t)
     {
      case(0):return(ORDER_TYPE_BUY_LIMIT);
      case(1):return(ORDER_TYPE_SELL_LIMIT);
      case(2):return(ORDER_TYPE_BUY_STOP);
      case(3):return(ORDER_TYPE_SELL_STOP);
     }
//--- 不正な値
   return(WRONG_VALUE);
  }
//+------------------------------------------------------------------+
//| ランダムな価格を返す                                             |
//+------------------------------------------------------------------+
double GetRandomPrice(ENUM_ORDER_TYPE type)
  {
   int t=(int)type;
//--- シンボルのストップレベル
   int distance=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_STOPS_LEVEL);
//--- 最終ティックのデータを受け取る
   MqlTick last_tick={};
   SymbolInfoTick(_Symbol,last_tick);
//--- 種類に応じて価格を計算する
   double price;
   if(t==2 || t==5) // ORDER_TYPE_BUY_LIMIT または ORDER_TYPE_SELL_STOP
     {
      price=last_tick.bid; // 買値から離れる
      price=price-(distance+(MathRand()%10)*5)*_Point;
     }
   else             // ORDER_TYPE_SELL_LIMIT または ORDER_TYPE_BUY_STOP
     {
      price=last_tick.ask; // 売値から離れる
      price=price+(distance+(MathRand()%10)*5)*_Point;
     }
//---
```

© 2000-2025, MetaQuotes Ltd.

```
   return(price);
  }
```

参照

取引操作の種類、取引リクエストの構造体、リクエストチェック結果の構造体、取引リクエスト結果の構造体

© 2000–2025, MetaQuotes Ltd.

OrderSendAsync

OrderSendAsync() 関数は、取引サーバの返答を待たずに非同期的な 取引操作を行うための リクエスト の送信に使用されます。この関数は、サーバからの応答を待つ時間の無駄を容認出来ない取引アルゴリズムの条件下での高頻度取引のために設計されています。

```
bool  OrderSendAsync(
   MqlTradeRequest&  request,       // リクエスト構造体
   MqlTradeResult&   result         // 返答構造体
   );
```

パラメータ

request

[in] クライエントの取引アクションを記述する MqlTradeRequest 型の構造体へのポインタ

result

[in,out] 成功した場合（true が返される場合）取引操作の結果を表す MqlTradeResult 型の構造体へのポインタ

戻り値

リクエストが取引サーバに送信された場合は true、それ以外の場合は false。リクエストが送信された場合 *result* 変数内で、レスポンスコードが TRADE_RETCODE_PLACED 値（コード 10008- order placed（注文が出された））を含みます。正常な実行は送信の事実のみを意味し、リクエストが取引サーバに到達し処理のために受け入れられている保証を与えるものではありません。受信したリクエストを処理する際、取引サーバは、ポジション、注文や約定の状態の変化を通知する応答をクライアント端末に送信し、これが Trade イベント生成につながります。

OrderSend() 関数で送られたリクエストをサーバ上で実行した結果は OnTradeTransaction ハンドラで追跡出来ます。1 つの取引リクエストを実行する際に OnTradeTransaction ハンドラは複数回呼ばれることには留意されるべきです。

例えば、市場の買い注文を送信する際に、注文が処理され、買い注文が口座に記録されます。その後注文が実行されオープン注文の表から削除されて注文履歴に追加されます。約定履歴に追加され新しいポジションが作成されます。OnTradeTransaction 関数はこれらのイベント全部に呼び出されます。この様なデータの取得には、関数パラメータが分析されるべきです。

- **trans** – このパラメータは取引口座に適用される取引トランザクションを記述する MqlTradeTransaction 構造体を取得します。
- **request** – このパラメータは取引トランザクションにつながった取引リクエストを記述する MqlTradeRequest 構造体を取得します。
- **result** – このパラメータは取引リクエスト実行の結果を記述する MqlTradeResult 構造体を取得します。

注意事項

パラメータの用途の点では、この関数は OrderSend() に似ていますが、それとは異なって非同期的です。つまり、関数の実行結果を待っている間にプログラム動作は止められません。サンプルのエキスパートアドバイザーを使用して、これら 2 つの関数の取引操作の速度を比較することが出来ます。

例：

```
#property description "Expert Advisor for sending trade requests "
                      " using OrderSendAsync() function.\r\n"
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "Handling trading events using"
                      " OnTrade() and OnTradeTransaction() handler functions is
displayed\r\n"
#property description "Expert Advisor parameters allow setting Magic Number"
                      " (unique ID) "
#property description "and the mode of displaying messages in Experts log. All
details are displayed by default.\r\n"
//--- 入力パラメータ
input int  MagicNumber=1234567;      // エキスパートアドバイザー ＩＤ
input bool DescriptionModeFull=true; // 詳細出力モード
//--- HistorySelect() 呼び出しに使用される変数
datetime history_start;
//+------------------------------------------------------------------+
//| エキスパート初期化に使用される関数                                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 自動売買が許可されているかどうかをチェックする
   if(!TerminalInfoInteger(TERMINAL_TRADE_ALLOWED))
     {
      Alert("Autotrading in the terminal is disabled, Expert Advisor will be
removed.");
      ExpertRemove();
      return(-1);
     }
//--- 実際の口座での取引が不可能
   if(AccountInfoInteger(ACCOUNT_TRADE_MODE)==ACCOUNT_TRADE_MODE_REAL)
     {
      Alert("Expert Advisor cannot trade on a real account!");
      ExpertRemove();
      return(-2);
     }
//--- この口座で取引が可能かどうかをみる（例えば、投資家のパスワードを使用しての取引は不可能）
   if(!AccountInfoInteger(ACCOUNT_TRADE_ALLOWED))
     {
      Alert("Trading on this account is disabled");
      ExpertRemove();
      return(-3);
     }
//--- 取引履歴を受信するためにエキスパートアドバイザーを起動する時間を節約する
   history_start=TimeCurrent();
//---
   CreateBuySellButtons();
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| エキスパート初期化解除に使用される関数                             |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- 全てのグラフィックオブジェクトを削除する
   ObjectDelete(0,"Buy");
   ObjectDelete(0,"Sell");
//---
  }
//+------------------------------------------------------------------+
//| TradeTransaction 関数                                            |
//+------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction &trans,
                        const MqlTradeRequest &request,
                        const MqlTradeResult &result)
  {
//--- 取引イベントのハンドラ関数名にちなんだ名付け見出し
   Print("=> ",__FUNCTION__," at ",TimeToString(TimeCurrent(),TIME_SECONDS));
//--- トランザクションタイプを列挙値として取得する
   ENUM_TRADE_TRANSACTION_TYPE type=trans.type;
//--- トランザクションがリクエスト処理の結果の場合
   if(type==TRADE_TRANSACTION_REQUEST)
     {
      //--- トランザクション名を表示する
      Print(EnumToString(type));
      //--- 処理されたリクエストを記述する文字列を表示する
      Print("------------RequestDescription\r\n",
            RequestDescription(request,DescriptionModeFull));
      //--- そしてリクエスト結果の記述を表示する
      Print("------------ ResultDescription\r\n",
            TradeResultDescription(result,DescriptionModeFull));
     }
   else // トランザクションの他のトランザクションのための完全な記述を表示する
     {
      Print("------------ TransactionDescription\r\n",
            TransactionDescription(trans,DescriptionModeFull));
     }
//---
  }
//+------------------------------------------------------------------+
//| 取引関数                                                         |
//+------------------------------------------------------------------+
void OnTrade()
  {
//--- 取引口座の状態を格納する静的メンバ
   static int prev_positions=0,prev_orders=0,prev_deals=0,prev_history_orders=0;
//--- 取引履歴をリクエストする
   bool update=HistorySelect(history_start,TimeCurrent());
   PrintFormat("HistorySelect(%s , %s) = %s",
               TimeToString(history_start),TimeToString(TimeCurrent()),(string)
update);
//--- 取引イベントのハンドラ関数名にちなんだ名付け見出し
```

© 2000-2025, MetaQuotes Ltd.

```
   Print("=> ",__FUNCTION__," at ",TimeToString(TimeCurrent(),TIME_SECONDS));
//--- ハンドラの名称と処理の瞬間の注文数を表示する
   int curr_positions=PositionsTotal();
   int curr_orders=OrdersTotal();
   int curr_deals=HistoryOrdersTotal();
   int curr_history_orders=HistoryDealsTotal();
//--- 注文、ポジション、及び約定の数、またカッコ内に変更を表示
   PrintFormat("PositionsTotal() = %d (%+d)",
               curr_positions,(curr_positions-prev_positions));
   PrintFormat("OrdersTotal() = %d (%+d)",
               curr_orders,curr_orders-prev_orders);
   PrintFormat("HistoryOrdersTotal() = %d (%+d)",
               curr_deals,curr_deals-prev_deals);
   PrintFormat("HistoryDealsTotal() = %d (%+d)",
               curr_history_orders,curr_history_orders-prev_history_orders);
//--- ログをより便利に表示する文字列の区切りを挿入する
   Print("");
//--- 口座の状態を保存する
   prev_positions=curr_positions;
   prev_orders=curr_orders;
   prev_deals=curr_deals;
   prev_history_orders=curr_history_orders;
//---
  }
//+------------------------------------------------------------------+
//| ChartEvent 関数                                                  |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- CHARTEVENT_CLICK イベント(「チャートクリック」)の処理
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      Print("=> ",__FUNCTION__,": sparam = ",sparam);
      //---約定の最小ボリューム
      double volume_min=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
      //---「Buy」ボタンが押されたら、買う
      if(sparam=="Buy")
        {
         PrintFormat("Buy %s %G lot",_Symbol,volume_min);
         BuyAsync(volume_min);
         //--- ボタンを放す
         ObjectSetInteger(0,"Buy",OBJPROP_STATE,false);
        }
      //---「Sell」ボタンが押されたら、売る
      if(sparam=="Sell")
        {
```

© 2000-2025, MetaQuotes Ltd.

```
         PrintFormat("Sell %s %G lot",_Symbol,volume_min);
         SellAsync(volume_min);
         //--- ボタンを放す
         ObjectSetInteger(0,"Sell",OBJPROP_STATE,false);
        }
      ChartRedraw();
     }
//---
  }
//+------------------------------------------------------------------+
//| トランザクションの記述を文字列として返す                         |
//+------------------------------------------------------------------+
string TransactionDescription(const MqlTradeTransaction &trans,
                              const bool detailed=true)
  {
//--- 関数が返す文字列を準備する
   string desc=EnumToString(trans.type)+"\r\n";
//--- detailed(繊細)モードでは繊細が全て追加される
   if(detailed)
     {
      desc+="Symbol: "+trans.symbol+"\r\n";
      desc+="Deal ticket: "+(string)trans.deal+"\r\n";
      desc+="Deal type: "+EnumToString(trans.deal_type)+"\r\n";
      desc+="Order ticket: "+(string)trans.order+"\r\n";
      desc+="Order type: "+EnumToString(trans.order_type)+"\r\n";
      desc+="Order state: "+EnumToString(trans.order_state)+"\r\n";
      desc+="Order time type: "+EnumToString(trans.time_type)+"\r\n";
      desc+="Order expiration: "+TimeToString(trans.time_expiration)+"\r\n";
      desc+="Price: "+StringFormat("%G",trans.price)+"\r\n";
      desc+="Price trigger: "+StringFormat("%G",trans.price_trigger)+"\r\n";
      desc+="Stop Loss: "+StringFormat("%G",trans.price_sl)+"\r\n";
      desc+="Take Profit: "+StringFormat("%G",trans.price_tp)+"\r\n";
      desc+="Volume: "+StringFormat("%G",trans.volume)+"\r\n";
     }
//--- 受信された文字列を返す
   return desc;
  }
//+------------------------------------------------------------------+
//| 取引リクエストのテキスト記述を返す                               |
//+------------------------------------------------------------------+
string RequestDescription(const MqlTradeRequest &request,
                          const bool detailed=true)
  {
//--- 関数が返す文字列を準備する
   string desc=EnumToString(request.action)+"\r\n";
//--- detailed(繊細)モードでは、使用可能のデータを全て追加する
   if(detailed)
     {
      desc+="Symbol: "+request.symbol+"\r\n";
```

© 2000-2025, MetaQuotes Ltd.

```
      desc+="Magic Number: "+StringFormat("%d",request.magic)+"\r\n";
      desc+="Order ticket: "+(string)request.order+"\r\n";
      desc+="Order type: "+EnumToString(request.type)+"\r\n";
      desc+="Order filling: "+EnumToString(request.type_filling)+"\r\n";
      desc+="Order time type: "+EnumToString(request.type_time)+"\r\n";
      desc+="Order expiration: "+TimeToString(request.expiration)+"\r\n";
      desc+="Price: "+StringFormat("%G",request.price)+"\r\n";
      desc+="Deviation points: "+StringFormat("%G",request.deviation)+"\r\n";
      desc+="Stop Loss: "+StringFormat("%G",request.sl)+"\r\n";
      desc+="Take Profit: "+StringFormat("%G",request.tp)+"\r\n";
      desc+="Stop Limit: "+StringFormat("%G",request.stoplimit)+"\r\n";
      desc+="Volume: "+StringFormat("%G",request.volume)+"\r\n";
      desc+="Comment: "+request.comment+"\r\n";
     }
//--- 受信された文字列を返す
   return desc;
  }
//+------------------------------------------------------------------+
//| リクエスト処理結果の記述ををテキスト形式で返す                                |
//+------------------------------------------------------------------+
string TradeResultDescription(const MqlTradeResult &result,
                              const bool detailed=true)
  {
//--- 関数が返す文字列を準備する
   string desc="Retcode "+(string)result.retcode+"\r\n";
//--- detailed(繊細)モードでは、使用可能のデータを全て追加する
   if(detailed)
     {
      desc+="Request ID: "+StringFormat("%d",result.request_id)+"\r\n";
      desc+="Order ticket: "+(string)result.order+"\r\n";
      desc+="Deal ticket: "+(string)result.deal+"\r\n";
      desc+="Volume: "+StringFormat("%G",result.volume)+"\r\n";
      desc+="Price: "+StringFormat("%G",result.price)+"\r\n";
      desc+="Ask: "+StringFormat("%G",result.ask)+"\r\n";
      desc+="Bid: "+StringFormat("%G",result.bid)+"\r\n";
      desc+="Comment: "+result.comment+"\r\n";
     }
//--- 受信された文字列を返す
   return desc;
  }
//+------------------------------------------------------------------+
//| 売買に必要な 2 つのボタンを作成する                                      |
//+------------------------------------------------------------------+
void CreateBuySellButtons()
  {
//--- 「Buy」オブジェクトをチェックする
   if(ObjectFind(0,"Buy")>=0)
     {
      //--- 見つかったオブジェクトがボタンでない場合は削除する
```

© 2000-2025, MetaQuotes Ltd.

```
      if(ObjectGetInteger(0,"Buy",OBJPROP_TYPE)!=OBJ_BUTTON)
         ObjectDelete(0,"Buy");
      }
   else
      ObjectCreate(0,"Buy",OBJ_BUTTON,0,0,0); //「Buy」ボタンを作成する
//---「Buy」ボタンを設定する
   ObjectSetInteger(0,"Buy",OBJPROP_CORNER,CORNER_RIGHT_UPPER);
   ObjectSetInteger(0,"Buy",OBJPROP_XDISTANCE,100);
   ObjectSetInteger(0,"Buy",OBJPROP_YDISTANCE,50);
   ObjectSetInteger(0,"Buy",OBJPROP_XSIZE,70);
   ObjectSetInteger(0,"Buy",OBJPROP_YSIZE,30);
   ObjectSetString(0,"Buy",OBJPROP_TEXT,"Buy");
   ObjectSetInteger(0,"Buy",OBJPROP_COLOR,clrRed);
//---「Sell」オブジェクトの存在をチェックする
   if(ObjectFind(0,"Sell")>=0)
     {
      //--- 見つかったオブジェクトがボタンでない場合は削除する
      if(ObjectGetInteger(0,"Sell",OBJPROP_TYPE)!=OBJ_BUTTON)
         ObjectDelete(0,"Sell");
     }
   else
      ObjectCreate(0,"Sell",OBJ_BUTTON,0,0,0); //「 Sell 」ボタンを作成する
//---「Sell」ボタンを設定する
   ObjectSetInteger(0,"Sell",OBJPROP_CORNER,CORNER_RIGHT_UPPER);
   ObjectSetInteger(0,"Sell",OBJPROP_XDISTANCE,100);
   ObjectSetInteger(0,"Sell",OBJPROP_YDISTANCE,100);
   ObjectSetInteger(0,"Sell",OBJPROP_XSIZE,70);
   ObjectSetInteger(0,"Sell",OBJPROP_YSIZE,30);
   ObjectSetString(0,"Sell",OBJPROP_TEXT,"Sell");
   ObjectSetInteger(0,"Sell",OBJPROP_COLOR,clrBlue);
//--- ボタンをすぐに表示するためにチャートの強制アップデートを実行する
   ChartRedraw();
//---
  }
//+------------------------------------------------------------------+
//| OrderSendAsync() 非同期的関数を使用して買う                      |
//+------------------------------------------------------------------+
void BuyAsync(double volume)
  {
//--- リクエストを準備する
   MqlTradeRequest req={};
   req.action      =TRADE_ACTION_DEAL;
   req.symbol      =_Symbol;
   req.magic       =MagicNumber;
   req.volume      =0.1;
   req.type        =ORDER_TYPE_BUY;
   req.price       =SymbolInfoDouble(req.symbol,SYMBOL_ASK);
   req.deviation   =10;
   req.comment     ="Buy using OrderSendAsync()";
```

© 2000-2025, MetaQuotes Ltd.

```
   MqlTradeResult  res={};
   if(!OrderSendAsync(req,res))
     {
      Print(__FUNCTION__,": error ",GetLastError(),", retcode = ",res.retcode);
     }
//---
  }
//+------------------------------------------------------------------+
//| OrderSendAsync() 非同期的関数を使用して売る                      |
//+------------------------------------------------------------------+
void SellAsync(double volume)
  {
//--- リクエストを準備する
   MqlTradeRequest req={};
   req.action      =TRADE_ACTION_DEAL;
   req.symbol      =_Symbol;
   req.magic       =MagicNumber;
   req.volume      =0.1;
   req.type        =ORDER_TYPE_SELL;
   req.price       =SymbolInfoDouble(req.symbol,SYMBOL_BID);
   req.deviation   =10;
   req.comment     ="Sell using OrderSendAsync()";
   MqlTradeResult  res={};
   if(!OrderSendAsync(req,res))
     {
      Print(__FUNCTION__,": error ",GetLastError(),", retcode = ",res.retcode);
     }
//---
  }
//+------------------------------------------------------------------+
```

「エキスパート」操作ログでのメッセージ表示の例：

```
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnChartEvent: sparam = Sell
12:52:52   ExpertAdvisor (EURUSD,H1)   Sell EURUSD 0.01 lot
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_REQUEST
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------RequestDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_ACTION_DEAL
12:52:52   ExpertAdvisor (EURUSD,H1)   Symbol: EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   Magic Number: 1234567
12:52:52   ExpertAdvisor (EURUSD,H1)   Order ticket: 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Order type: ORDER_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Order filling: ORDER_FILLING_FOK
12:52:52   ExpertAdvisor (EURUSD,H1)   Order time type: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Order expiration: 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Price: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Deviation points: 10
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss: 0
```

© 2000–2025, MetaQuotes Ltd.

```
12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Limit: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0.1
12:52:52   ExpertAdvisor (EURUSD,H1)   Comment: Sell using OrderSendAsync()
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ ResultDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   Retcode 10009
12:52:52   ExpertAdvisor (EURUSD,H1)   Request ID: 2
12:52:52   ExpertAdvisor (EURUSD,H1)   Order ticket: 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal ticket: 15048668
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0.1
12:52:52   ExpertAdvisor (EURUSD,H1)   Price: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Ask: 1.29319
12:52:52   ExpertAdvisor (EURUSD,H1)   Bid: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Comment:
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistorySelect( 09:34 , 09:52) = true
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTrade at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   PositionsTotal() = 1 (+1)
12:52:52   ExpertAdvisor (EURUSD,H1)   OrdersTotal() = 0 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryOrdersTotal() = 2 (+2)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryDealsTotal() = 2 (+2)
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ TransactionDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_ORDER_ADD
12:52:52   ExpertAdvisor (EURUSD,H1)   Symbol: EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal ticket: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal type: DEAL_TYPE_BUY
12:52:52   ExpertAdvisor (EURUSD,H1)   Order ticket: 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Order type: ORDER_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Order state: ORDER_STATE_STARTED
12:52:52   ExpertAdvisor (EURUSD,H1)   Order time type: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Order expiration: 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Price: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Price trigger: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0.1
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ TransactionDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_ORDER_DELETE
12:52:52   ExpertAdvisor (EURUSD,H1)   Symbol: EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal ticket: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal type: DEAL_TYPE_BUY
12:52:52   ExpertAdvisor (EURUSD,H1)   Order ticket: 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Order type: ORDER_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Order state: ORDER_STATE_STARTED
```

© 2000–2025, MetaQuotes Ltd.

```
12:52:52   ExpertAdvisor (EURUSD,H1)   Order time type: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Order expiration: 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Price: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Price trigger: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0.1
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistorySelect( 09:34 , 09:52) = true
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTrade at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   PositionsTotal() = 1 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   OrdersTotal() = 0 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryOrdersTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryDealsTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ TransactionDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_HISTORY_ADD
12:52:52   ExpertAdvisor (EURUSD,H1)   Symbol: EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal ticket: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal type: DEAL_TYPE_BUY
12:52:52   ExpertAdvisor (EURUSD,H1)   Order ticket: 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Order type: ORDER_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Order state: ORDER_STATE_FILLED
12:52:52   ExpertAdvisor (EURUSD,H1)   Order time type: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Order expiration: 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Price: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Price trigger: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistorySelect( 09:34 , 09:52) = true
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTrade at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   PositionsTotal() = 1 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   OrdersTotal() = 0 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryOrdersTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryDealsTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ TransactionDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_DEAL_ADD
12:52:52   ExpertAdvisor (EURUSD,H1)   Symbol: EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal ticket: 15048668
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal type: DEAL_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Order ticket: 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Order type: ORDER_TYPE_BUY
12:52:52   ExpertAdvisor (EURUSD,H1)   Order state: ORDER_STATE_STARTED
12:52:52   ExpertAdvisor (EURUSD,H1)   Order time type: ORDER_TIME_GTC
```

© 2000–2025, MetaQuotes Ltd.

```
12:52:52   ExpertAdvisor (EURUSD,H1)   Order expiration: 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Price: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Price trigger: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0.1
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistorySelect( 09:34 , 09:52) = true
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTrade at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   PositionsTotal() = 1 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   OrdersTotal() = 0 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryOrdersTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryDealsTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)
```

© 2000–2025, MetaQuotes Ltd.

PositionsTotal

未決済ポジションの数を返します。

```
int  PositionsTotal();
```

戻り値

int 型の値

注意事項

ポジションの『ネッティング』計算時（ACCOUNT_MARGIN_MODE_RETAIL_NETTINGとACCOUNT_MARGIN_MODE_EXCHANGE）各シンボルにつき常に一つのポジションのみ（1つ以上の取引の結果である）保有することができます。ポジションと『ツールボックス』パネルの『取引』タブに表示される有効な未決注文と混同しないようにしてください。

ポジションに制限がない場合（ACCOUNT_MARGIN_MODE_RETAIL_HEDGING）各シンボルごとに同時に複数のポジションを保有することができます。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 口座のポジションの数を取得して操作ログに出力する
   int total=PositionsTotal();
   Print("Number of open positions on account: ", total);
   /*
   結果:
   Number of open positions on account: 2
   */
  }
```

参照

PositionGetSymbol()、PositionSelect()、ポジションプロパティ

© 2000-2025, MetaQuotes Ltd.

PositionGetSymbol

未決済ポジションに対応するシンボルを返し、PositionGetDouble、PositionGetInteger、PositionGetString関数を使用しての操作のために自動的にポジションを選択します。

```
string  PositionGetSymbol(
   int  index      // ポジションリスト内の番号
   );
```

パラメータ

index

[in] 未決済ポジションリスト内での番号

戻り値

string 型の変数ポジションが見つからなかった場合、空の文字列 が戻されます。エラーコード取得にはGetLastError() 関数が呼ばれます。

注意事項

ポジションの『ネッティング』計算時（ACCOUNT_MARGIN_MODE_RETAIL_NETTINGとACCOUNT_MARGIN_MODE_EXCHANGE）各シンボルにつき常に一つのポジションのみ（1つ以上の取引の結果である）保有することができます。ポジションと『ツールボックス』パネルの『取引』タブに表示される有効な未決注文と混同しないようにしてください。

ポジションに制限がない場合（ACCOUNT_MARGIN_MODE_RETAIL_HEDGING）各シンボルごとに同時に複数のポジションを保有することができます。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 口座のポジションの数を取得する
   int total=PositionsTotal();
   for(int i=0; i<total; i++)
     {
      //--- i ループインデックスでポジション銘柄を取得する
      ResetLastError();
      string symbol=PositionGetSymbol(i);

      //--- ポジションの銘柄が正常に受信されると、i インデックスのポジションが自動的に選択される
      //--- PositionGetDouble、PositionGetInteger、PositionGetStringを使用してそのプロパティを
取得できる
      if(symbol!="")
        {
         ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)
PositionGetInteger(POSITION_TYPE);
         PrintFormat("Position symbol at index %d: %s, position type: %s", i,
symbol, StringSubstr(EnumToString(type), 14));
```

© 2000-2025, MetaQuotes Ltd.

```
         }
       else
         {
          PrintFormat("PositionGetSymbol(%d) failed. Error %d", i, GetLastError());
          continue;
         }
     }
   /*
   結果:
   Position symbol at index 0: GBPUSD, position type: SELL
   Position symbol at index 1: EURUSD, position type: BUY
   */
  }
```

参照

PositionsTotal()、PositionSelect()、ポジションプロパティ

© 2000–2025, MetaQuotes Ltd.

PositionSelect

今後の作業で使用する保有ポジションを選択します。関数の実行に成功した場合、trueを返します。関数の実行に失敗した場合、falseを返します。エラーに関する情報を取得するには、GetLastError()関数を呼び出す必要があります。

```
bool  PositionSelect(
   string  symbol     // 商品名
   );
```

パラメータ

symbol

[in] 金融商品の名称。

戻り値

bool型の値。

注意事項

ポジションの『ネッティング』計算時（ACCOUNT_MARGIN_MODE_RETAIL_NETTINGとACCOUNT_MARGIN_MODE_EXCHANGE）各シンボルにつき常に一つのポジションのみ（1つ以上の取引の結果である）保有することができます。ポジションと『ツールボックス』パネルの『取引』タブに表示される有効な未決注文と混同しないようにしてください。

ポジションに制限がない場合（ACCOUNT_MARGIN_MODE_RETAIL_HEDGING）各シンボルごとに同時に複数のポジションを保有することができます。この場合、PositionSelectは最小チケットのポジションを選択します。

PositionSelect()関数はソフトウェア環境でのポジションについての情報をコピーし、後続のPositionGetDouble()、PositionGetInteger()、PositionGetString()の呼び出しは前にコピーしたデータを返します。これはつまり、ポジション自体はすでになくても（またはポジションのボリュームや方向が変わったなど）、このポジションのデータは取得できるということを意味しています。ポジションの最新データの取得を確実なものにする為には、直接PositionSelect()関数を適用前に呼び出すことをお勧めします。

例：

```
#define    SYMBOL_NAME    "EURUSD"

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 指定された銘柄でポジションを選択する
   if(!PositionSelect(SYMBOL_NAME))
     {
      PrintFormat("PositionSelect(%s) failed. Error %d",SYMBOL_NAME,
GetLastError());
      return;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- ポジションが選択されている場合、PositionGetDouble()、PositionGetInteger()、
PositionGetString() を使用してそのデータを取得できる
//--- 選択されたポジションチケットを取得する
   ResetLastError();
   long ticket=PositionGetInteger(POSITION_TICKET);
   if(ticket==0)
     {
      PrintFormat("Failed to get %s position ticket. Error %d", SYMBOL_NAME,
GetLastError());
      return;
     }

//--- チケットが正常に受信された場合、選択されたポジションの銘柄とチケットを操作ログに出力する
   PrintFormat("The position that is selected on the %s symbol has ticket %I64d",
SYMBOL_NAME, ticket);
   /*
   結果:
   The position that is selected on the EURUSD symbol has ticket 2810846623
   */
  }
```

参照

PositionGetSymbol(), PositionsTotal(), ポジションプロパティ

© 2000-2025, MetaQuotes Ltd.

PositionSelectByTicket

指定したチケットによって今後の作業で使用する保有ポジションを選択します。関数の実行に成功した場合、trueを返します。関数の実行に失敗した場合、falseを返します。エラーに関する情報を取得するには、GetLastError()関数を呼び出す必要があります。

```
bool  PositionSelectByTicket(
   ulong    ticket      // ポジションチケット
   );
```

パラメータ

ticket

[in] ポジションチケット。

戻り値

bool型の値。

注意事項

PositionSelectByTicket()関数はソフトウェア環境でのポジションについての情報をコピーし、後続のPositionGetDouble()、PositionGetInteger()、PositionGetString()の呼び出しは前にコピーしたデータを返します。これはつまり、ポジション自体はすでになくても（またはポジションのボリュームや方向が変わったなど）、このポジションのデータは取得できるということを意味しています。ポジションの最新データの取得を確実なものにする為には、直接PositionSelectByTicket()関数を適用前に呼び出すことをお勧めします。

例：

```
#define   EXPERT_MAGIC  123456   // マジックナンバー

//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- リクエストおよび結果の構造体を宣言して初期化する
   MqlTradeRequest request={};
   MqlTradeResult  result ={};

//--- ロング（買い）注文を出す際のパラメータを入力する
   request.action     = TRADE_ACTION_DEAL;                       // 取引操作の種類
   request.symbol     = Symbol();                                // 銘柄
   request.volume     = 0.1;                                     // 0.1ロットの数量
   request.type       = ORDER_TYPE_BUY;                          // 注文の種類
   request.price      = SymbolInfoDouble(Symbol(), SYMBOL_ASK); // 始値
   request.deviation = 5;                                        // 価格から許容される偏差
   request.magic      = EXPERT_MAGIC;                            // 注文のマジックnアンバー

//--- リクエストを送信する。リクエストの送信に失敗した場合、エラーコードを表示し、操作を完了する
   if(!OrderSend(request, result))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("OrderSend error ", GetLastError());
      return;
     }

//--- 操作データを表示する
   PrintFormat("Trade request result: retcode: %u, deal: %I64u, order: %I64u",
result.retcode, result.deal, result.order);

//--- 取引操作の結果からポジションチケットを取得し、チケットでポジションを選択する
//--- 新しく開かれたポジションのチケットは、その取引を生成した注文のチケットに対応する
   ulong ticket=result.order;
   ResetLastError();
   if(!PositionSelectByTicket(ticket))
     {
      PrintFormat("PositionSelectByTicket(%I64u) failed. Error %d", ticket,
GetLastError());
      return;
     }

//--- チケットで選択されたポジションのデータを操作ログに表示する
   ENUM_POSITION_TYPE type  = (ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
   long               time  = PositionGetInteger(POSITION_TIME_MSC);
   double             price = PositionGetDouble(POSITION_PRICE_OPEN);
   double             volume= PositionGetDouble(POSITION_VOLUME);
   string             symbol= PositionGetString(POSITION_SYMBOL);
   int                digits= (int)SymbolInfoInteger(symbol, SYMBOL_DIGITS);
   PrintFormat("Current selected position: %s %.2f %s #%I64u at %.*f, %s",
               symbol, volume, (type==POSITION_TYPE_BUY ? "Buy" : "Sell"), ticket,
digits, price, TimeMscToString(time));
   /*
   結果:
   Trade request result: retcode: 10009, deal: 2778100901, order: 2803905975
   Current selected position: EURUSD 0.10 Buy #2803905975 at 1.10672, 2024.09.02
12:09:51.239
   */
  }
//+------------------------------------------------------------------+
//| ミリ秒で時間を返す                                               |
//+------------------------------------------------------------------+
string TimeMscToString(const long time_msc, int flags=TIME_DATE|TIME_MINUTES|
TIME_SECONDS)
  {
   return(TimeToString(time_msc/1000, flags) + "." + IntegerToString(time_msc %1000,
3, '0'));
  }
```

参照

PositionGetSymbol(), PositionsTotal(), ポジションプロパティ

© 2000-2025, MetaQuotes Ltd.

PositionGetDouble

この関数は PositionGetSymbol または PositionSelect で事前に選択された未決済ポジションのリクエストされたプロパティを返します。ポジションプロパティは double 型でなければなりません。この関数には 2 つのバージョンがあります。

1. すぐにプロパティ値を返します。

```
double  PositionGetDouble(
   ENUM_POSITION_PROPERTY_DOUBLE  property_id      // プロパティ識別子
   );
```

2. 関数実行の成功に応じて true または false を返す。成功した場合、プロパティの値は、参照によって渡された最後のパラメータに配置される。

```
bool  PositionGetDouble(
   ENUM_POSITION_PROPERTY_DOUBLE  property_id,     // プロパティ識別子
   double&                        double_var       // プロパティ値を受け取る
   );
```

パラメータ

property_id

[in] ポジションプロパティの識別子。値は ENUM_POSITION_PROPERTY_DOUBLE 列挙のいずれかです。

double_var

[out] リクエストされたプロパティの値を受け取るdouble 型の変数

戻り値

double 型の値（関数が失敗した場合は 0）

注意事項

ポジションの『ネッティング』計算時（ACCOUNT_MARGIN_MODE_RETAIL_NETTINGと ACCOUNT_MARGIN_MODE_EXCHANGE）各シンボルにつき常に一つのポジションのみ（1つ以上の取引の結果である）保有することができます。ポジションと『ツールボックス』パネルの『取引』タブに表示される有効な未決注文と混同しないようにしてください。

ポジションに制限がない場合（ACCOUNT_MARGIN_MODE_RETAIL_HEDGING）各シンボルごとに同時に複数のポジションを保有することができます。

取得したポジション情報が最新であるように、PositionSelect() を参照直前に呼び出すことが推奨されています。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---  すべての口座ポジションのループ
```

© 2000-2025, MetaQuotes Ltd.

```
   int total=PositionsTotal();
   for(int i=0; i<total; i++)
     {
      //--- 自動的にポジションを選択して、そのプロパティにアクセスし、次のポジションのチケットを取得する
      ulong ticket=PositionGetTicket(i);
      if(ticket==0)
         continue;

      //--- ポジションタイプを取得し、ポジションの実数プロパティー覧のヘッダーを表示する
      string type=(PositionGetInteger(POSITION_TYPE)==POSITION_TYPE ? "Buy" :
"Sell");
      PrintFormat("Double properties of an open position %s #%I64u:", type, ticket);

      //--- 選択した取引のすべての実数プロパティをヘッダーの下に表示する
      PositionPropertiesDoublePrint(15);
     }
   /*
   結果:
   Double properties of an open position Buy #2807075208:
   Volume:         1.00
   Price open:     1.10516
   StopLoss:       0.00000
   TakeProfit:     0.00000
   Price current: 1.10518
   Swap:           0.00
   Profit:         2.00 USD
   */
  }
//+------------------------------------------------------------------+
//|  操作ログに選択したポジションの実数プロパティを表示する                |
//+------------------------------------------------------------------+
void PositionPropertiesDoublePrint(const uint header_width=0)
  {
   uint   w=0;
   string header="";
   double value=0;

//--- 口座の通貨、ポジションの銘柄、およびその銘柄の小数点以下の桁数を取得する
   string currency=AccountInfoString(ACCOUNT_CURRENCY);
   string symbol  =PositionGetString(POSITION_SYMBOL);
   int    digits  =(int)SymbolInfoInteger(symbol, SYMBOL_DIGITS);

//--- ヘッダーのテキストとヘッダーフィールドの幅を定義する
//--- ヘッダーの幅が関数に渡され、ゼロと等しい場合、幅はヘッダー行のサイズ+1になる
   header="Volume:";
   w=(header_width==0 ? header.Length()+1 : header_width);
//--- ポジションボリュームを取得し、指定されたヘッダー幅で操作ログに表示する
   if(!PositionGetDouble(POSITION_VOLUME, value))
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
   PrintFormat("%-*s%-.2f", w, header, value);

//--- 操作ログにポジション価格を表示する
   header="Price open:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!PositionGetDouble(POSITION_PRICE_OPEN, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- 操作ログにストップロス値を表示する
   header="StopLoss:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!PositionGetDouble(POSITION_SL, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- 操作ログにテイクプロフィット値を表示する
   header="TakeProfit:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!PositionGetDouble(POSITION_TP, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- 操作ログに現行価格を表示する
   header="Price current:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!PositionGetDouble(POSITION_PRICE_CURRENT, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- 蓄積されたスワップ値を操作ログに表示する
   header="Swap:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!PositionGetDouble(POSITION_SWAP, value))
      return;
   PrintFormat("%-*s%-.2f", w, header, value);

//--- 操作ログに現在の利益値を表示する
   header="Profit:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!PositionGetDouble(POSITION_PROFIT, value))
      return;
   PrintFormat("%-*s%-.2f %s", w, header, value, currency);
  }
```

参照

PositionGetSymbol()、PositionSelect()、ポジションプロパティ

© 2000-2025, MetaQuotes Ltd.

PositionGetInteger

この関数は PositionGetSymbol または PositionSelect で事前に選択された未決済ポジションのリクエストされたプロパティを返します。ポジションプロパティは datetime または int 型でなければいけません。この関数には 2 つのバージョンがあります。

1. すぐにプロパティ値を返します。

```
long  PositionGetInteger(
   ENUM_POSITION_PROPERTY_INTEGER  property_id      // プロパティ識別子
   );
```

2. 関数実行の成功に応じて true または false を返す。成功した場合、プロパティの値は、参照によって渡された最後のパラメータに配置されます。

```
bool  PositionGetInteger(
   ENUM_POSITION_PROPERTY_INTEGER  property_id,     // プロパティ識別子
   long&                           long_var         // プロパティ値を受け取る
   );
```

パラメータ

property_id

[in] ポジションプロパティの識別子。値は ENUM_POSITION_PROPERTY_INTEGER 列挙のいずれかです。

long_var

[out] リクエストされたプロパティの値を受け取る long 型の変数

戻り値

long 型の値（関数が失敗した場合は 0）

注意事項

ポジションの『ネッティング』計算時（ACCOUNT_MARGIN_MODE_RETAIL_NETTINGと ACCOUNT_MARGIN_MODE_EXCHANGE）各シンボルにつき常に一つのポジションのみ（1つ以上の取引の結果である）保有することができます。ポジションと『ツールボックス』パネルの『取引』タブに表示される有効な未決注文と混同しないようにしてください。

ポジションに制限がない場合（ACCOUNT_MARGIN_MODE_RETAIL_HEDGING）各シンボルごとに同時に複数のポジションを保有することができます。

取得したポジション情報が最新であるように、PositionSelect() を参照直前に呼び出すことが推奨されています。

例：

```
//+------------------------------------------------------------------+
//|  取引関数                                                        |
//+------------------------------------------------------------------+
void OnTrade()
  {
//--- ポジションが存在するかどうかをチェックし、変化の時刻を表示する
```

© 2000-2025, MetaQuotes Ltd.

```
   if(PositionSelect(_Symbol))
     {
//--- 更に操作するためにポジション識別子を受け取る
      ulong position_ID=PositionGetInteger(POSITION_IDENTIFIER);
      Print(_Symbol," position #",position_ID);
//--- 01.01.1970 から経過したミリ秒で表されたポジション形成の時刻を受け取る
      long create_time_msc=PositionGetInteger(POSITION_TIME_MSC);
      PrintFormat("Position #%d  POSITION_TIME_MSC = %i64 milliseconds => %
s",position_ID,
                  create_time_msc,TimeToString(create_time_msc/1000));
//--- 01.01.1970 から経過した秒数で表されたポジションの最終変化時刻を受け取る
      long update_time_sec=PositionGetInteger(POSITION_TIME_UPDATE);
      PrintFormat("Position #%d  POSITION_TIME_UPDATE = %i64 seconds => %s",
                  position_ID,update_time_sec,TimeToString(update_time_sec));
//--- 01.01.1970 から経過したミリ秒数で表されたポジションの最終変化時刻を受け取る
      long update_time_msc=PositionGetInteger(POSITION_TIME_UPDATE_MSC);
      PrintFormat("Position #%d  POSITION_TIME_UPDATE_MSC = %i64 milliseconds => %
s",
                  position_ID,update_time_msc,TimeToString(update_time_msc/1000));
     }
//---
  }
```

参照

PositionGetSymbol()、PositionSelect()、ポジションプロパティ

© 2000-2025, MetaQuotes Ltd.

PositionGetString

この関数は PositionGetSymbol または PositionSelect で事前に選択された未決済ポジションのリクエストされたプロパティを返します。ポジションプロパティは string 型でなければなりません。この関数には 2 つのバージョンがあります。

1. すぐにプロパティ値を返します。

```
string  PositionGetString(
   ENUM_POSITION_PROPERTY_STRING  property_id      // プロパティ識別子
   );
```

2. 関数実行の成功に応じて true または false を返す。成功した場合、プロパティの値は、参照によって渡された最後のパラメータに配置されます。

```
bool  PositionGetString(
   ENUM_POSITION_PROPERTY_STRING  property_id,     // プロパティ識別子
   string&                        string_var       // プロパティ値を受け取る
   );
```

パラメータ

property_id

[in] ポジションプロパティの識別子。値は ENUM_POSITION_PROPERTY_STRING 列挙のいずれかです。

string_var

[out] リクエストされたプロパティ値を受け取るstring 型の変数

戻り値

string 型の変数関数が失敗した場合、空の文字列 が戻されます。

注意事項

ポジションの『ネッティング』計算時（ACCOUNT_MARGIN_MODE_RETAIL_NETTINGとACCOUNT_MARGIN_MODE_EXCHANGE）各シンボルにつき常に一つのポジションのみ（1つ以上の取引の結果である）保有することができます。ポジションと『ツールボックス』パネルの『取引』タブに表示される有効な未決注文と混同しないようにしてください。

ポジションに制限がない場合（ACCOUNT_MARGIN_MODE_RETAIL_HEDGING）各シンボルごとに同時に複数のポジションを保有することができます。

取得したポジション情報が最新であるように、PositionSelect() を参照直前に呼び出すことが推奨されています。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- すべての口座ポジションのループ
   int total=PositionsTotal();
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0; i<total; i++)
     {
      //--- 自動的にポジションを選択して、そのプロパティにアクセスし、次のポジションのチケットを取得する
      ulong ticket=PositionGetTicket(i);
      if(ticket==0)
         continue;

      //--- ポジションタイプを取得し、ポジションの文字列プロパティ一覧のヘッダーを表示する
      string type=(PositionGetInteger(POSITION_TYPE)==POSITION_TYPE ? "Buy" :
"Sell");
      PrintFormat("String properties of an open position %s #%I64u:", type, ticket);

      //--- ヘッダーの下に選択したポジションのすべての文字列プロパティを表示する
      PositionPropertiesStringPrint(15);
     }
   /*
   結果:
   String properties of an open position Buy #2810798881:
   Symbol:        EURUSD
   Comment:       Test PositionGetString
   External ID:
   */
  }
//+------------------------------------------------------------------+
//| 操作ログに選択したポジションの文字列プロパティを表示する                  |
//+------------------------------------------------------------------+
void PositionPropertiesStringPrint(const uint header_width=0)
  {
   uint   w=0;
   string header="";
   string value="";

//--- ヘッダーのテキストとヘッダーフィールドの幅を定義する
//--- ヘッダーの幅が関数に渡され、ゼロと等しい場合、幅はヘッダー行のサイズ+1になる
   header="Symbol:";
   w=(header_width==0 ? header.Length()+1 : header_width);
//--- ポジション銘柄を取得し、指定されたヘッダー幅で操作ログに表示する
   if(!PositionGetString(POSITION_SYMBOL, value))
      return;
   PrintFormat("%-*s%-s", w, header, value);

//--- 操作ログにポジションコメントを表示する
   header="Comment:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!PositionGetString(POSITION_COMMENT, value))
      return;
   PrintFormat("%-*s%-s", w, header, value);

//--- 操作ログに外部取引システムでのポジションIDを表示する
```

© 2000-2025, MetaQuotes Ltd.

```
   header="External ID:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!PositionGetString(POSITION_EXTERNAL_ID, value))
      return;
   PrintFormat("%-*s%-s", w, header, value);
  }
```

参照

PositionGetSymbol()、PositionSelect()、ポジションプロパティ

© 2000-2025, MetaQuotes Ltd.

PositionGetTicket

関数は保有ポジションの一覧のインデックスでポジションチケットを返し、PositionGetDouble、PositionGetInteger、PositionGetString関数を使用した今後の作業の為に自動的にこのポジションを選択します。

```
ulong  PositionGetTicket(
   int  index      // ポジション一覧の番号
   );
```

パラメータ

index

[in] 0から始まる保有ポジション一覧のポジションインデックス。

戻り値

ポジションチケット。実行に失敗した場合は、0を返します。

注意事項

ポジションの『ネッティング』計算時（ACCOUNT_MARGIN_MODE_RETAIL_NETTINGとACCOUNT_MARGIN_MODE_EXCHANGE）各シンボルにつき常に一つのポジションのみ（1つ以上の取引の結果である）保有することができます。ポジションと『ツールボックス』パネルの『取引』タブに表示される有効な未決注文と混同しないようにしてください。

ポジションに制限がない場合（ACCOUNT_MARGIN_MODE_RETAIL_HEDGING）各シンボルごとに同時に複数のポジションを保有することができます。

ポジションの最新データの取得を確実なものにする為には、直接PositionSelect()関数を適用前に呼び出すことをお勧めします。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- すべての口座ポジションのループ
   int total=PositionsTotal();
   for(int i=0; i<total; i++)
     {
      //--- 自動的にポジションを選択して、そのプロパティにアクセスし、次のポジションのチケットを取得する
      ulong ticket=PositionGetTicket(i);
      if(ticket==0)
         continue;

      //--- ポジションタイプを取得し、選択されたポジションの説明を操作ログに表示する
      string type=(PositionGetInteger(POSITION_TYPE)==POSITION_TYPE ? "Buy" :
"Sell");
      PrintFormat("[%d] Selected position %s #%I64u", i, type, ticket);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   /*
   結果:
   [0] Selected position Sell #2810802718
   [1] Selected position Buy #2810802919
   */
  }
```

参照

PositionGetSymbol(), PositionSelect(), ポジションプロパティ

© 2000-2025, MetaQuotes Ltd.

OrdersTotal

現在の注文数を返します。

```
int  OrdersTotal();
```

戻り値

int 型の値

注意事項

現在の未決注文はクライアント端末の「ツールボックス」の「取引」タブに表示されているポジションと混乱されてはなりません。注文とはトランザクションを実行するリクエストです。ポジションは 1 つまたは複数の約定の結果です。

ポジションの『ネッティング』計算時（ACCOUNT_MARGIN_MODE_RETAIL_NETTINGと ACCOUNT_MARGIN_MODE_EXCHANGE）各シンボルにつき常に一つのポジションのみ（1つ以上の取引の結果である）保有することができます。ポジションと『ツールボックス』パネルの『取引』タブに表示される有効な未決注文と混同しないようにしてください。

ポジションに制限がない場合（ACCOUNT_MARGIN_MODE_RETAIL_HEDGING）各シンボルごとに同時に複数のポジションを保有することができます。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 口座のアクティブな未決済注文の数を取得し、操作ログに表示する
   int total=OrdersTotal();
   Print("Number of active pending orders on the account: ", total);
   /*
   結果:
   Number of active pending orders on the account: 2
   */
  }
```

参照

OrderSelect()、OrderGetTicket()、注文プロパティ

© 2000-2025, MetaQuotes Ltd.

OrderGetTicket

さらに関数を使用して作業するための注文を自動的に選択して対応するチケットを返します。

```
ulong  OrderGetTicket(
   int  index      // 注文リスト内の番号
   );
```

パラメータ

index

[in] 現在の注文リスト内での注文の番号

戻り値

ulong 型の値（関数が失敗した場合は 0 ）

注意事項

現在の未決注文はクライアント端末の「ツールボックス」の「取引」タブに表示されているポジションと混乱されてはなりません。注文とはトランザクションを実行するリクエストです。ポジションは 1 つまたは複数の約定の結果です。

ポジションの『ネッティング』計算時（ACCOUNT_MARGIN_MODE_RETAIL_NETTINGと ACCOUNT_MARGIN_MODE_EXCHANGE）各シンボルにつき常に一つのポジションのみ（1つ以上の取引の結果である）保有することができます。ポジションと『ツールボックス』パネルの『取引』タブに表示される有効な未決注文と混同しないようにしてください。

ポジションに制限がない場合（ACCOUNT_MARGIN_MODE_RETAIL_HEDGING）各シンボルごとに同時に複数のポジションを保有することができます。

OrderGetTicket() 関数は、注文に関するデータをプログラム環境にコピーし、OrderGetDouble()、OrderGetInteger()、OrderGetString() の呼び出しは既に複製されたデータを返します。つまり、注文自体はもはや存在しなくても（または始値、決済逆指及び決済指レベルまたは期限が変更されても）、注文に関するデータはまだ取得することが出来ます。取得した注文情報が最新であるように、OrderGetTicket() を参照直前に呼び出すことが推奨されています。

例:

```
void OnStart()
  {
//--- 注文プロパティから値を返す変数
   ulong    ticket;
   double   open_price;
   double   initial_volume;
   datetime time_setup;
   string   symbol;
   string   type;
   long     order_magic;
   long     positionID;
//--- 現在の未決注文数
   uint     total=OrdersTotal();
//--- ループで注文をみる
```

© 2000–2025, MetaQuotes Ltd.

```
   for(uint i=0;i<total;i++)
     {
      //--- リスト内の位置によって注文を返す
      if((ticket=OrderGetTicket(i))>0)
        {
         //--- 注文プロパティを返す
         open_price    =OrderGetDouble(ORDER_PRICE_OPEN);
         time_setup    =(datetime)OrderGetInteger(ORDER_TIME_SETUP);
         symbol        =OrderGetString(ORDER_SYMBOL);
         order_magic   =OrderGetInteger(ORDER_MAGIC);
         positionID    =OrderGetInteger(ORDER_POSITION_ID);
         initial_volume=OrderGetDouble(ORDER_VOLUME_INITIAL);
         type          =EnumToString(ENUM_ORDER_TYPE(OrderGetInteger(ORDER_TYPE)));
         //--- 注文についての情報を準備して表示する
         printf("#ticket %d %s %G %s at %G was set up at %s",
                ticket,                  // 注文チケット
                type,                    // 種類
                initial_volume,          // 注文のボリューム
                symbol,                  // シンボル
                open_price,              // 指定された始値
                TimeToString(time_setup)// 注文の出された時刻
                );
        }
     }
//---
   }
```

参照

OrdersTotal()、OrderSelect()、OrderGetInteger()

© 2000-2025, MetaQuotes Ltd.

OrderSelect

操作に使用する注文を選択します。関数実行が成功した場合は true を返します。関数実行が失敗した場合 false を返します。更なるエラー情報を取得するには、GetLastError() が呼ばれます。

```
bool  OrderSelect(
   ulong    ticket      // 注文チケット
   );
```

パラメータ

ticket

[in] 注文チケット

戻り値

bool 型の値

注意事項

現在の未決注文はクライアント端末の「ツールボックス」の「取引」タブに表示されているポジションと混乱されてはなりません。

ポジションの『ネッティング』計算時（ACCOUNT_MARGIN_MODE_RETAIL_NETTINGとACCOUNT_MARGIN_MODE_EXCHANGE）各シンボルにつき常に一つのポジションのみ（1つ以上の取引の結果である）保有することができます。ポジションと『ツールボックス』パネルの『取引』タブに表示される有効な未決注文と混同しないようにしてください。

ポジションに制限がない場合（ACCOUNT_MARGIN_MODE_RETAIL_HEDGING）各シンボルごとに同時に複数のポジションを保有することができます。

OrderSelect() 関数は、注文に関するデータをプログラム環境にコピーし、OrderGetDouble()、OrderGetInteger()、OrderGetString() の呼び出しは既に複製されたデータを返します。つまり、注文自体はもはや存在しなくても（または始値、決済逆指及び決済指レベルまたは期限が変更されても）、注文に関するデータはまだ取得することが出来ます。取得した注文情報が最新であるように、OrderGetTicket() を参照直前に呼び出すことが推奨されています。

例：

```
#define   EXPERT_MAGIC  123456
#define   OFFSET        50                    // 注文を出すための現在の価格からのオフセット（ポイント単位）
#define   DIRECTION     ORDER_TYPE_BUY_LIMIT  // 注文タイプ
#define   VOLUME        1.0                   // 数量
#define   DEVIATION     2                     // 価格から許容される偏差

//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 取引リクエスト、結果、変数を宣言して初期化する
   MqlTradeRequest request={};
   MqlTradeResult  result ={};
```

© 2000–2025, MetaQuotes Ltd.

```
   double          order_price=0;

//--- 未決注文パラメータ
   request.action    = TRADE_ACTION_PENDING;                                 //
trading operation type
   request.symbol    = _Symbol;                                              // symbol
   request.volume    = VOLUME;                                               // volume
   request.deviation = DEVIATION;                                            //
allowed deviation from the price
   request.magic     = EXPERT_MAGIC;                                         // order
MagicNumber

//--- 操作タイプを確認する
   switch(DIRECTION)
     {
      case ORDER_TYPE_BUY_LIMIT :
        request.type = ORDER_TYPE_BUY_LIMIT;                                 // order
type
        order_price  = SymbolInfoDouble(_Symbol, SYMBOL_ASK)-OFFSET*_Point;// open
price
        request.price= NormalizeDouble(order_price, _Digits);               //
normalized open price
        break;
      case ORDER_TYPE_SELL_LIMIT :
        request.type = ORDER_TYPE_SELL_LIMIT;                                // order
type
        order_price  = SymbolInfoDouble(_Symbol, SYMBOL_BID)+OFFSET*_Point;// open
price
        request.price= NormalizeDouble(order_price,_Digits);                //
normalized open price
        break;
      case ORDER_TYPE_BUY_STOP :
        request.type = ORDER_TYPE_BUY_STOP;                                  // order
type
        order_price  = SymbolInfoDouble(_Symbol, SYMBOL_ASK)+OFFSET*_Point;// open
price
        request.price= NormalizeDouble(order_price,_Digits);                //
normalized open price
        break;
      case ORDER_TYPE_SELL_STOP :
        request.type = ORDER_TYPE_SELL_STOP;                                 // order
type
        order_price  = SymbolInfoDouble(_Symbol, SYMBOL_BID)-OFFSET*_Point;// open
price
        request.price= NormalizeDouble(order_price,_Digits);                //
normalized open price
        break;
      default: // if non-pending or StopLimit order is selected
        Alert("This example is only for placing pending orders BuyLimit, SellLimit,
BuyStop and SellStop");
        break;
     }
```

© 2000–2025, MetaQuotes Ltd.

```
//--- リクエストを送信する。リクエストの送信に失敗した場合、エラーコードを表示し、操作を完了する
   if(!OrderSend(request, result))
     {
      Print("OrderSend error ", GetLastError());
      return;
     }

//--- 操作データを表示する
   PrintFormat("Trade request result: retcode=%u, order=%I64u", result.retcode,
result.order);

//--- 取引操作の結果から注文チケットを取得し、チケットで注文を選択する
   ulong ticket=result.order;
   ResetLastError();
   if(!OrderSelect(ticket))
     {
      PrintFormat("OrderSelect(%I64u) failed. Error %d", ticket, GetLastError());
      return;
     }

//--- チケットで選択された注文のデータを操作ログに表示する
   ENUM_ORDER_TYPE   type   = (ENUM_ORDER_TYPE)OrderGetInteger(ORDER_TYPE);
   long              time   = OrderGetInteger(ORDER_TIME_SETUP_MSC);
   double            price = OrderGetDouble(ORDER_PRICE_OPEN);
   double            volume= OrderGetDouble(ORDER_VOLUME_CURRENT);
   string            symbol= OrderGetString(ORDER_SYMBOL);
   int               digits= (int)SymbolInfoInteger(symbol, SYMBOL_DIGITS);
   PrintFormat("Current selected order: %s %.2f %s #%I64u at %.*f, %s",
               symbol, volume, OrderTypeDescription(type), ticket, digits, price,
TimeMscToString(time));
   /*
   結果:
   Trade request result: retcode=10009, order=2811006719
   Current selected order: EURUSD 1.00 Buy Limit #2811006719 at 1.10550, 2024.09.04
10:38:28.563
   */
  }
//+------------------------------------------------------------------+
//|  ミリ秒で時間を返す                                              |
//+------------------------------------------------------------------+
string TimeMscToString(const long time_msc, int flags=TIME_DATE|TIME_MINUTES|
TIME_SECONDS)
  {
   return(TimeToString(time_msc/1000, flags) + "." + IntegerToString(time_msc %1000,
3, '0'));
  }
//+------------------------------------------------------------------+
//|  注文タイプの説明を返す                                          |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      case ORDER_TYPE_BUY_LIMIT        :  return("Buy Limit");
      case ORDER_TYPE_SELL_LIMIT       :  return("Sell Limit");
      case ORDER_TYPE_BUY_STOP         :  return("Buy Stop");
      case ORDER_TYPE_SELL_STOP        :  return("Sell Stop");
      case ORDER_TYPE_BUY_STOP_LIMIT   :  return("Buy Stop Limit");
      case ORDER_TYPE_SELL_STOP_LIMIT  :  return("Sell Stop Limit");
      default                          :  return("Unknown order type");
     }
  }
```

参照

OrderGetInteger()、OrderGetDouble()、OrderGetString()、OrderCalcProfit()、OrderGetTicket()、注文プロパティ

© 2000–2025, MetaQuotes Ltd.

OrderGetDouble

OrderGetTicket または OrderSelectで事前に選択された注文のプロパティを返します。注文プロパティはdouble 型でなければなりません。この関数には 2 つのバージョンがあります。

1. すぐにプロパティ値を返します。

```
double  OrderGetDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  property_id         // プロパティ識別子
   );
```

2. 関数の成功に応じて、true または false を返します。成功した場合、プロパティの値は、参照によって渡された最後のパラメータに配置されます。

```
bool  OrderGetDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  property_id,       // プロパティ識別子
   double&                     double_var         // プロパティ値を受け取る
   );
```

パラメータ

property_id

[in] 注文プロパティの識別子。値は ENUM_ORDER_PROPERTY_DOUBLE 列挙のいずれかです。

double_var

[out] リクエストされたプロパティの値を受け取るdouble 型の変数

戻り値

double 型の値（関数が失敗した場合は 0）

注意事項

現在の未決注文はクライアント端末の「ツールボックス」の「取引」タブに表示されているポジションと混乱されてはなりません。

ポジションの『ネッティング』計算時（ACCOUNT_MARGIN_MODE_RETAIL_NETTINGとACCOUNT_MARGIN_MODE_EXCHANGE）各シンボルにつき常に一つのポジションのみ（1つ以上の取引の結果である）保有することができます。ポジションと『ツールボックス』パネルの『取引』タブに表示される有効な未決注文と混同しないようにしてください。

ポジションに制限がない場合（ACCOUNT_MARGIN_MODE_RETAIL_HEDGING）各シンボルごとに同時に複数のポジションを保有することができます。

取得した注文情報が最新であるように、OrderSelect()　を参照直前に呼び出すことが推奨されています。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- すべての口座注文のリストのループ
```

© 2000-2025, MetaQuotes Ltd.

```
   int total=OrdersTotal();
   for(int i=0; i<total; i++)
     {
      //--- ループインデックスでリストから注文チケットを取得する
      ulong ticket=OrderGetTicket(i);
      if(ticket==0)
         continue;

      //--- 注文タイプを取得し、選択した注文の実数プロパティのリストのヘッダーを表示する
      string type=OrderTypeDescription((ENUM_ORDER_TYPE)
OrderGetInteger(ORDER_TYPE));
      PrintFormat("Double properties of an active pending order %s #%I64u:", type,
ticket);

      //--- ヘッダーの下に選択した注文のすべての実数プロパティを表示する
      OrderPropertiesDoublePrint(16);
     }
   /*
   結果:
   Double properties of an active pending order Sell Limit #2812000714:
   Volume initial: 1.00
   Volume current: 1.00
   Price open:     145.282
   StopLoss:       0.000
   TakeProfit:     0.000
   Price current:  145.044
   StopLimit:      0.000
   Double properties of an active pending order Buy Limit #2812001112:
   Volume initial: 1.00
   Volume current: 1.00
   Price open:     144.836
   StopLoss:       0.000
   TakeProfit:     0.000
   Price current:  145.051
   StopLimit:      0.000
   Double properties of an active pending order Buy Stop #2812001488:
   Volume initial: 0.50
   Volume current: 0.50
   Price open:     1.10642
   StopLoss:       0.00000
   TakeProfit:     0.00000
   Price current:  1.10530
   StopLimit:      0.00000
   Double properties of an active pending order Sell Stop #2812001712:
   Volume initial: 0.50
   Volume current: 0.50
   Price open:     1.10374
   StopLoss:       0.00000
   TakeProfit:     0.00000
```

© 2000–2025, MetaQuotes Ltd.

```
   Price current:  1.10525
   StopLimit:      0.00000
   */
  }
//+------------------------------------------------------------------+
//|  注文タイプの説明を返す                                          |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      case ORDER_TYPE_BUY_LIMIT        :  return("Buy Limit");
      case ORDER_TYPE_SELL_LIMIT       :  return("Sell Limit");
      case ORDER_TYPE_BUY_STOP         :  return("Buy Stop");
      case ORDER_TYPE_SELL_STOP        :  return("Sell Stop");
      case ORDER_TYPE_BUY_STOP_LIMIT   :  return("Buy Stop Limit");
      case ORDER_TYPE_SELL_STOP_LIMIT  :  return("Sell Stop Limit");
      default                          :  return("Unknown order type");
     }
  }
//+------------------------------------------------------------------+
//|  選択した注文の実数プロパティを操作ログに表示する               |
//+------------------------------------------------------------------+
void OrderPropertiesDoublePrint(const uint header_width=0)
  {
//--- 注文の銘柄とその小数点以下の桁数を取得する
   string symbol = OrderGetString(ORDER_SYMBOL);
   int    digits = (int)SymbolInfoInteger(symbol, SYMBOL_DIGITS);

//--- 注文を出す際に初期ボリュームを指定されたヘッダー幅で操作ログに表示する
   OrderPropertyPrint("Volume initial:",header_width,2,ORDER_VOLUME_INITIAL);

//--- 操作ログに未履行の注文ボリュームを表示する
   OrderPropertyPrint("Volume current:",header_width,2,ORDER_VOLUME_CURRENT);

//--- 操作ログに注文で指定された価格を表示する
   OrderPropertyPrint("Price open:",header_width,digits,ORDER_PRICE_OPEN);

//--- 操作ログにストップロスレベルを表示する
   OrderPropertyPrint("StopLoss:",header_width,digits,ORDER_SL);

//--- 操作ログにテイクプロフィットレベルを表示する
   OrderPropertyPrint("TakeProfit:",header_width,digits,ORDER_TP);

//--- 操作ログに注文の銘柄による現在の価格を表示する
   OrderPropertyPrint("Price current:",header_width,digits,ORDER_PRICE_CURRENT);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 操作ログにストップリミット注文がアクティブになったときのリミット注文価格を表示する
   OrderPropertyPrint("StopLimit:",header_width,digits,ORDER_PRICE_STOPLIMIT);
  }
//+------------------------------------------------------------------+
//|  操作ログに外部取引システムでの注文の実数プロパティを表示する                          |
//+------------------------------------------------------------------+
void OrderPropertyPrint(const string header, uint header_width, int digits,
ENUM_ORDER_PROPERTY_DOUBLE property)
  {
   double value=0;
   if(!OrderGetDouble(property, value))
      PrintFormat("Cannot get property %s, error=%d", EnumToString(property),
GetLastError());
   else
     {
//--- ヘッダーの幅が関数に渡され、ゼロと等しい場合、幅はヘッダー行のサイズ+1になる
      uint w=(header_width==0 ? header.Length()+1 : header_width);
      PrintFormat("%-*s%-.*f", w, header, digits, value);
     }
  }
```

参照

OrdersTotal()、OrderGetTicket()、注文プロパティ

© 2000-2025, MetaQuotes Ltd.

OrderGetInteger

OrderGetTicket または OrderSelect で事前に選択された注文のリクエストされたプロパティを返します。注文プロパティは datetime または int 型でなければいけません。この関数には 2 つのバージョンがあります。

1. すぐにプロパティ値を返します。

```
long  OrderGetInteger(
   ENUM_ORDER_PROPERTY_INTEGER  property_id         // プロパティ識別子
   );
```

2. 関数実行の成功に応じて true または false を返します。成功した場合、プロパティの値は、参照によって渡された最後のパラメータに配置されます。

```
bool  OrderGetInteger(
   ENUM_ORDER_PROPERTY_INTEGER  property_id,        // プロパティ識別子
   long&                        long_var            // プロパティ値を受け取る
   );
```

パラメータ

property_id

[in] 注文プロパティの識別子。値は ENUM_ORDER_PROPERTY_INTEGER 列挙のいずれかです。

long_var

[out] リクエストされたプロパティの値を受け取る long 型の変数

戻り値

long 型の値（関数が失敗した場合は 0）

注意事項

現在の未決注文はクライアント端末の「ツールボックス」の「取引」タブに表示されているポジションと混乱されてはなりません。

ポジションの『ネッティング』計算時（ACCOUNT_MARGIN_MODE_RETAIL_NETTINGと ACCOUNT_MARGIN_MODE_EXCHANGE）各シンボルにつき常に一つのポジションのみ（1つ以上の取引の結果である）保有することができます。ポジションと『ツールボックス』パネルの『取引』タブに表示される有効な未決注文と混同しないようにしてください。

ポジションに制限がない場合（ACCOUNT_MARGIN_MODE_RETAIL_HEDGING）各シンボルごとに同時に複数のポジションを保有することができます。

取得した注文情報が最新であるように、OrderSelect() を参照直前に呼び出すことが推奨されています。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- すべての口座注文のリストのループ
```

© 2000-2025, MetaQuotes Ltd.

```
   int total=OrdersTotal();
   for(int i=0; i<total; i++)
     {
      //--- ループインデックスでリストから注文チケットを取得する
      ulong ticket=OrderGetTicket(i);
      if(ticket==0)
         continue;

      //--- 注文タイプを取得し、選択した注文の実数プロパティのリストのヘッダーを表示する
      string type=OrderTypeDescription((ENUM_ORDER_TYPE)
OrderGetInteger(ORDER_TYPE));
      PrintFormat("Integer properties of an active pending order %s #%I64u:", type,
ticket);

      //--- ヘッダーの下に選択した注文のすべての整数プロパティを表示する
      OrderPropertiesIntegerPrint(17);
     }
   /*
   結果:
   Integer properties of an active pending order Buy Limit #2812945317:
   Ticket:           2812945317
   Time setup:       2024.09.04 19:17:16
   Type:             Buy Limit
   State:            Placed
   Time expiration:  0
   Time done:        0
   Time setup msc:   2024.09.04 19:17:16.686
   Time done msc:    0
   Type filling:     Return
   Type time:        Time GTC
   Magic:            0
   Reason:           Client
   Position ID:      0
   Position By ID:   0
   */
  }
//+------------------------------------------------------------------+
//|  選択した注文の整数プロパティを操作ログに表示する                |
//+------------------------------------------------------------------+
void OrderPropertiesIntegerPrint(const uint header_width=0)
  {
   uint   w=0;
   string header="";
   long   value=0;

//--- 操作ログに注文チケットを表示する
   OrderPropertyPrint("Ticket:", header_width, ORDER_TICKET);

//--- 操作ログに注文時刻を表示する
```

© 2000–2025, MetaQuotes Ltd.

```
   OrderPropertyPrint("Time setup:", header_width, ORDER_TIME_SETUP);

//--- 操作ログに注文タイプを表示する
   OrderPropertyPrint("Type:", header_width, ORDER_TYPE);

//--- 操作ログに注文ステータスを表示する
   OrderPropertyPrint("State:", header_width, ORDER_STATE);

//--- 操作ログに注文の有効期限を表示する
   OrderPropertyPrint("Time expiration:", header_width, ORDER_TIME_EXPIRATION);

//--- 操作ログに注文の実行/期限切れの時間を表示する
   OrderPropertyPrint("Time done:", header_width, ORDER_TIME_DONE);

//--- 操作ログに1970年1月1日からのミリ秒単位での注文実行時間を表示する
   OrderPropertyPrint("Time setup msc:", header_width, ORDER_TIME_SETUP_MSC);

//--- 操作ログに1970年1月1日からのミリ秒単位での注文の実行/期限切れ時間を表示する
   OrderPropertyPrint("Time done msc:", header_width, ORDER_TIME_DONE_MSC);

//--- 操作ログに残高による実行タイプを表示する
   OrderPropertyPrint("Type filling:", header_width, ORDER_TYPE_FILLING);

//--- 操作ログに注文の有効期限を表示する
   OrderPropertyPrint("Type time:", header_width, ORDER_TYPE_TIME);

//--- 操作ログに注文を出したEAのIDを表示する
   OrderPropertyPrint("Magic:", header_width, ORDER_MAGIC);

//--- 操作ログに注文の理由または注文元を表示する
   OrderPropertyPrint("Reason:", header_width, ORDER_REASON);

//--- 操作ログに注文の実行中に設定されたポジションのIDを表示する
   OrderPropertyPrint("Position ID:", header_width, ORDER_POSITION_ID);

//--- 操作ログにORDER_TYPE_CLOSE_BYタイプの注文に対する反対のポジションIDを表示する
   OrderPropertyPrint("Position By ID:", header_width, ORDER_POSITION_BY_ID);
  }
//+------------------------------------------------------------------+
//|  操作ログに注文の整数プロパティを表示する                                |
//+------------------------------------------------------------------+
void OrderPropertyPrint(const string header, uint header_width,
ENUM_ORDER_PROPERTY_INTEGER property)
  {
   string svalue="";
   long   lvalue=0;
   if(!OrderGetInteger(property, lvalue))
      PrintFormat("Cannot get property %s, error=%d", EnumToString(property),
GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
   else
     {
      switch(property)
        {
         case ORDER_TICKET          :
         case ORDER_MAGIC           :
         case ORDER_POSITION_ID     :
         case ORDER_POSITION_BY_ID  :
           svalue=(string)lvalue;
           break;

         case ORDER_TIME_SETUP      :
         case ORDER_TIME_EXPIRATION :
         case ORDER_TIME_DONE       :
           svalue=(lvalue!=0 ? TimeToString((datetime)lvalue, TIME_DATE|
TIME_MINUTES|TIME_SECONDS) : "0");
           break;

         case ORDER_TIME_SETUP_MSC  :
         case ORDER_TIME_DONE_MSC   :
           svalue=(lvalue!=0 ? TimeMscToString(lvalue) : "0");
           break;

         case ORDER_TYPE            :
           svalue=OrderTypeDescription((ENUM_ORDER_TYPE)lvalue);
           break;
         case ORDER_STATE           :
           svalue=OrderStateDescription((ENUM_ORDER_STATE)lvalue);
           break;
         case ORDER_TYPE_FILLING    :
           svalue=OrderTypeFillingDescription((ENUM_ORDER_TYPE_FILLING)lvalue);
           break;
         case ORDER_TYPE_TIME       :
           svalue=OrderTypeTimeDescription((ENUM_ORDER_TYPE_TIME)lvalue);
           break;
         case ORDER_REASON          :
           svalue=OrderReasonDescription((ENUM_ORDER_REASON)lvalue);
           break;

         default                    :
           svalue="Unknown property";
           break;
        }

//---  ヘッダーの幅が関数に渡され、ゼロと等しい場合、幅はヘッダー行のサイズ+1になる
      uint w=(header_width==0 ? header.Length()+1 : header_width);
      PrintFormat("%-*s%-s", w, header, svalue);
     }
  }
```

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  注文タイプの説明を返す                                          |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      case ORDER_TYPE_BUY_LIMIT        :  return("Buy Limit");
      case ORDER_TYPE_SELL_LIMIT       :  return("Sell Limit");
      case ORDER_TYPE_BUY_STOP         :  return("Buy Stop");
      case ORDER_TYPE_SELL_STOP        :  return("Sell Stop");
      case ORDER_TYPE_BUY_STOP_LIMIT   :  return("Buy Stop Limit");
      case ORDER_TYPE_SELL_STOP_LIMIT  :  return("Sell Stop Limit");
      default                          :  return("Unknown order type: "+(string)
type);
     }
  }
//+------------------------------------------------------------------+
//|  注文ステータスの説明を返す                                      |
//+------------------------------------------------------------------+
string OrderStateDescription(ENUM_ORDER_STATE state)
  {
   switch(state)
     {
      case ORDER_STATE_STARTED         :  return("Started");
      case ORDER_STATE_PLACED          :  return("Placed");
      case ORDER_STATE_CANCELED        :  return("Canceled");
      case ORDER_STATE_PARTIAL         :  return("Partial");
      case ORDER_STATE_FILLED          :  return("Filled");
      case ORDER_STATE_REJECTED        :  return("Rejected");
      case ORDER_STATE_EXPIRED         :  return("Expired");
      case ORDER_STATE_REQUEST_ADD     :  return("Request Add");
      case ORDER_STATE_REQUEST_MODIFY  :  return("Request Modify");
      case ORDER_STATE_REQUEST_CANCEL  :  return("Request Cancel");
      default                          :  return("Unknown state: "+(string)state);
     }
  }
//+------------------------------------------------------------------+
//|  注文量充足ポリシーの説明を返す                                  |
//+------------------------------------------------------------------+
string OrderTypeFillingDescription(const ENUM_ORDER_TYPE_FILLING type)
  {
   switch(type)
     {
      case ORDER_FILLING_FOK     :  return("Fill or Kill");
      case ORDER_FILLING_IOC     :  return("Immediate or Cancel");
      case ORDER_FILLING_BOC     :  return("Book or Cancel");
```

© 2000–2025, MetaQuotes Ltd.

```
      case ORDER_FILLING_RETURN  :  return("Return");
      default                    :  return("Unknown type filling: "+(string)type);
     }
  }
//+------------------------------------------------------------------+
//| 注文の有効期限の説明を返す                                       |
//+------------------------------------------------------------------+
string OrderTypeTimeDescription(const ENUM_ORDER_TYPE_TIME type)
  {
   switch(type)
     {
      case ORDER_TIME_GTC           :   return("Time GTC");
      case ORDER_TIME_DAY           :   return("Time Day");
      case ORDER_TIME_SPECIFIED     :   return("Time Specified");
      case ORDER_TIME_SPECIFIED_DAY :   return("Time Specified Day");
      default                       :  return("Unknown type time: "+(string)type);
     }
  }
//+------------------------------------------------------------------+
//| 注文理由の説明を返す                                             |
//+------------------------------------------------------------------+
string OrderReasonDescription(const ENUM_ORDER_REASON reason)
  {
   switch(reason)
     {
      case ORDER_REASON_CLIENT    :  return("Client");
      case ORDER_REASON_MOBILE    :  return("Mobile");
      case ORDER_REASON_WEB       :  return("Web");
      case ORDER_REASON_EXPERT    :  return("Expert");
      case ORDER_REASON_SL        :  return("Stop Loss");
      case ORDER_REASON_TP        :  return("Take Profit");
      case ORDER_REASON_SO        :  return("Stop Out");
      default                     :  return("Unknown reason: "+(string)reason);
     }
  }
//+------------------------------------------------------------------+
//| ミリ秒で時間を返す                                               |
//+------------------------------------------------------------------+
string TimeMscToString(const long time_msc, int flags=TIME_DATE|TIME_MINUTES|
TIME_SECONDS)
  {
   return(TimeToString(time_msc/1000, flags) + "." + IntegerToString(time_msc %1000,
3, '0'));
  }
```

参照

OrdersTotal()、OrderGetTicket()、注文プロパティ

© 2000–2025, MetaQuotes Ltd.

OrderGetString

OrderGetTicket または OrderSelect で事前に選択された注文のリクエストされたプロパティを返します。注文プロパティは string 型でなければなりません。この関数には 2 つのバージョンがあります。

1. すぐにプロパティ値を返します。

```
string  OrderGetString(
   ENUM_ORDER_PROPERTY_STRING  property_id         // プロパティ識別子
   );
```

2. 関数実行の成功に応じて true または false を返します。成功した場合、プロパティの値は、参照によって渡された最後のパラメータに配置されます。

```
bool  OrderGetString(
   ENUM_ORDER_PROPERTY_STRING  property_id,        // プロパティ識別子
   string&                     string_var          // プロパティ値を受け取る
   );
```

パラメータ

property_id

[in] 注文プロパティの識別子。値は ENUM_ORDER_PROPERTY_STRING 列挙のいずれかです。

string_var

[out] リクエストされたプロパティ値を受け取るstring 型の変数

戻り値

string 型の変数

注意事項

現在の未決注文はクライアント端末の「ツールボックス」の「取引」タブに表示されているポジションと混乱されてはなりません。

ポジションの『ネッティング』計算時（ACCOUNT_MARGIN_MODE_RETAIL_NETTINGとACCOUNT_MARGIN_MODE_EXCHANGE）各シンボルにつき常に一つのポジションのみ（1つ以上の取引の結果である）保有することができます。ポジションと『ツールボックス』パネルの『取引』タブに表示される有効な未決注文と混同しないようにしてください。

ポジションに制限がない場合（ACCOUNT_MARGIN_MODE_RETAIL_HEDGING）各シンボルごとに同時に複数のポジションを保有することができます。

取得した注文情報が最新であるように、OrderSelect()　を参照直前に呼び出すことが推奨されています。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- すべての口座注文のリストのループ
```

© 2000-2025, MetaQuotes Ltd.

```
   int total=OrdersTotal();
   for(int i=0; i<total; i++)
     {
      //--- ループインデックスでリストから注文チケットを取得する
      ulong ticket=OrderGetTicket(i);
      if(ticket==0)
         continue;

      //--- 注文タイプを取得し、選択した注文の文字列プロパティのリストのヘッダーを表示する
      string type=OrderTypeDescription((ENUM_ORDER_TYPE)
OrderGetInteger(ORDER_TYPE));
      PrintFormat("String properties of an active pending order %s #%I64u:", type,
ticket);

      //--- ヘッダーの下に選択した注文のすべての文字列プロパティを表示する
      OrderPropertiesStringPrint(13);
     }
   /*
   結果:
   String properties of an active pending order Sell Limit #2813781342:
   Comment:      Test OrderGetString
   Symbol:       EURUSD
   External ID:
   */
  }
//+------------------------------------------------------------------+
//|  操作ログに選択した注文の文字列プロパティを表示する                |
//+------------------------------------------------------------------+
void OrderPropertiesStringPrint(const uint header_width=0)
  {
//--- 操作ログにコメントを表示する
   OrderPropertyPrint("Comment:", header_width, ORDER_COMMENT);

//--- 操作ログに注文が出された銘柄を表示する
   OrderPropertyPrint("Symbol:", header_width, ORDER_SYMBOL);

//--- 操作ログに外部取引システムでの注文I Dを表示する
   OrderPropertyPrint("External ID:", header_width, ORDER_EXTERNAL_ID);
  }
//+------------------------------------------------------------------+
//   操作ログに注文の文字列プロパティを表示する                      |
//+------------------------------------------------------------------+
void OrderPropertyPrint(const string header, uint header_width,
ENUM_ORDER_PROPERTY_STRING property)
  {
   string value="";
   if(!OrderGetString(property, value))
      PrintFormat("Cannot get property %s, error=%d", EnumToString(property),
GetLastError());
```

© 2000–2025, MetaQuotes Ltd.

```
   else
     {
//--- ヘッダーの幅が関数に渡され、ゼロと等しい場合、幅はヘッダー行のサイズ+1になる
      uint w=(header_width==0 ? header.Length()+1 : header_width);
      PrintFormat("%-*s%-s", w, header, value);
     }
  }
//+------------------------------------------------------------------+
//|  注文タイプの説明を返す                                          |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      case ORDER_TYPE_BUY_LIMIT        :  return("Buy Limit");
      case ORDER_TYPE_SELL_LIMIT       :  return("Sell Limit");
      case ORDER_TYPE_BUY_STOP         :  return("Buy Stop");
      case ORDER_TYPE_SELL_STOP        :  return("Sell Stop");
      case ORDER_TYPE_BUY_STOP_LIMIT   :  return("Buy Stop Limit");
      case ORDER_TYPE_SELL_STOP_LIMIT  :  return("Sell Stop Limit");
      default                          :  return("Unknown order type: "+(string)
type);
     }
  }
```

参照

OrdersTotal()、OrderGetTicket()、注文プロパティ

© 2000-2025, MetaQuotes Ltd.

HistorySelect

指定されたサーバタイム期間の約定と注文履歴を取得します。

```
bool  HistorySelect(
   datetime  from_date,     // 開始日
   datetime  to_date        // 終了日
   );
```

パラメータ

from_date

[in] リクエストの開始日

to_date

[in] リクエストの終了日

戻り値

成功の場合は true、それ以外の場合は false

注意事項

HistorySelect() は MQL5 プログラムの注文と取引のリストを作成し、対応する関数での要素参照を可能にします。約定リストのサイズは HistoryDealsTotal() 関数、注文リストのサイズは HistoryOrdersTotal() 関数を使用して取得することが出来ます。注文リストでの選択には HistoryOrderGetTicket() が使用されるべきです。約定リストの要素には HistoryDealGetTicket() が適切です。

MQL5 プログラムで使用可能な注文履歴のリストは HistoryOrderSelect() の使用後にリセットされ、チケットによる注文の検索が成功した場合、見つかった注文で書き入れられます。MQL5 プログラムで使用可能な約定履歴のリストについても同じことが言えます。それは HistoryDealSelect() の使用後にリセットされ、チケットによる約定が受け取られた場合、それで書き込まれます。

例:

```
void OnStart()
  {
   color BuyColor =clrBlue;
   color SellColor=clrRed;
//--- 取引履歴をリクエストする
   HistorySelect(0,TimeCurrent());
//--- オブジェクトを作成する
   string   name;
   uint     total=HistoryDealsTotal();
   ulong    ticket=0;
   double   price;
   double   profit;
   datetime time;
   string   symbol;
   long     type;
   long     entry;
//--- 全ての約定に対して
```

© 2000-2025, MetaQuotes Ltd.

```
   for(uint i=0;i<total;i++)
     {
      //--- 約定チケットの取得を試みる
      if((ticket=HistoryDealGetTicket(i))>0)
        {
         //--- 約定プロパティを取得する
         price =HistoryDealGetDouble(ticket,DEAL_PRICE);
         time  =(datetime)HistoryDealGetInteger(ticket,DEAL_TIME);
         symbol=HistoryDealGetString(ticket,DEAL_SYMBOL);
         type  =HistoryDealGetInteger(ticket,DEAL_TYPE);
         entry =HistoryDealGetInteger(ticket,DEAL_ENTRY);
         profit=HistoryDealGetDouble(ticket,DEAL_PROFIT);
         //--- 現在のシンボルのみ
         if(price && time && symbol==Symbol())
           {
            //--- 価格オブジェクトを作成する
            name="TradeHistory_Deal_"+string(ticket);
            if(entry) ObjectCreate(0,name,OBJ_ARROW_RIGHT_PRICE,0,time,price,0,0);
            else      ObjectCreate(0,name,OBJ_ARROW_LEFT_PRICE,0,time,price,0,0);
            //--- オブジェクトプロパティを設定する
            ObjectSetInteger(0,name,OBJPROP_SELECTABLE,0);
            ObjectSetInteger(0,name,OBJPROP_BACK,0);
            ObjectSetInteger(0,name,OBJPROP_COLOR,type?BuyColor:SellColor);
            if(profit!=0) ObjectSetString(0,name,OBJPROP_TEXT,"Profit:
"+string(profit));
           }
        }
     }
//--- チャートに適応する
   ChartRedraw();
  }
```

参照

HistoryOrderSelect()、HistoryDealSelect()

© 2000-2025, MetaQuotes Ltd.

HistorySelectByPosition

指定されたポジション識別子を有する取引や注文履歴を取得します。

```
bool  HistorySelectByPosition(
   long   position_id     // ポジション識別子 - POSITION_IDENTIFIER
   );
```

パラメータ

position_id

[in] 全ての実行された注文と約定に設定されるポジション識別子

戻り値

成功の場合は true、それ以外の場合は false

注意事項

取引履歴の注文を「ツールボックス」バーの「取引」タブに表示される未決注文 と混乱してはいけません。取り消されたまたはトランザクションにつながった 注文のリストはクライアント端末の「ツールボックス」バーの「履歴」タブで見られます。

HistorySelectByPosition() は MQL5 プログラムに指定されたポジション識別子 を持った注文リストと約定リストを作成し、対応する関数での要素参照を可能にします。約定リストのサイズは HistoryDealsTotal() 関数、注文リストのサイズは HistoryOrdersTotal() 関数を使用して取得することが出来ます。注文リストの要素を全部参照するには HistoryOrderGetTicket()、約定リストの要素を全部参照するには HistoryDealGetTicket() が使用されるべきです。

MQL5 プログラムで使用可能な注文履歴のリストは HistoryOrderSelect() の使用後にリセットされ チケットによる注文の検索 が成功した場合、見つかった注文で書き入れられます。MQL5 プログラムで使用可能な取引履歴のリストについても同じことが言えます。それは HistoryDealSelect() にリセットされ、チケットによる取引が受け取られた場合、それで書き込まれます。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   long  pos_id_array[];          // array for storing position IDs

//--- 履歴全部をリクエストする
   if(!HistorySelect(0, TimeCurrent()))
     {
      Print("HistorySelect() failed. Error ", GetLastError());
      return;
     }

//--- 配列内の未決注文からすべてのポジションIDを収集する
   int total=HistoryOrdersTotal();
   for(int i=0; i<total; i++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       ulong ticket=HistoryOrderGetTicket(i);
       if(ticket==0)
          continue;
       ENUM_ORDER_TYPE type=(ENUM_ORDER_TYPE)HistoryOrderGetInteger(ticket,
ORDER_TYPE);
       long pos_id=HistoryOrderGetInteger(ticket, ORDER_POSITION_ID);
       if(type<=ORDER_TYPE_SELL || pos_id==0)
          continue;

       int size=ArraySize(pos_id_array);
       if(ArrayResize(pos_id_array, size+1)==size+1)
          pos_id_array[size]=pos_id;
      }

//--- 配列内のポジションIDのリストによる
   total=ArraySize(pos_id_array);
   for(int i=0; i<total; i++)
      {
       //--- ヘッダーとポジション、注文、取引のリストを表示する
       long position_id=pos_id_array[i];
       Print("List of orders and deals for position with ID: ", position_id);
       HistorySelectByPositionProcess(position_id);
      }
   /*
   結果:
   List of orders and deals for position with ID: 1819629924
      [0] Order Sell Limit #1819629924
      [1] Order Buy #1819633194
      [0] Entry In Deal Sell #1794972472
      [1] Entry Out Deal Buy #1794975589
   List of orders and deals for position with ID: 1841753970
      [0] Order Sell Stop #1841753970
      [1] Order Buy #1842322160
      [0] Entry In Deal Sell #1817242142
      [1] Entry Out Deal Buy #1817765341
   */
  }
//+------------------------------------------------------------------+
//| ポジションIDによって注文と取引の履歴を選択し、                                |
//| ポジションの注文と取引のリストを操作ログに表示する                              |
//+------------------------------------------------------------------+
bool HistorySelectByPositionProcess(const long position_id)
  {
//--- 指定されたポジションIDを持つ取引と注文の履歴をリクエストする
   if(!HistorySelectByPosition(position_id))
      {
       PrintFormat("HistorySelectByPosition(%I64d) failed. Error %d", position_id,
GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
     }

//--- ポジション注文のリストを表示する
   int orders_total=HistoryOrdersTotal();
   for(int i=0; i<orders_total; i++)
     {
      ulong ticket=HistoryOrderGetTicket(i);
      if(ticket==0)
         continue;
      ENUM_ORDER_TYPE order_type=(ENUM_ORDER_TYPE)HistoryOrderGetInteger(ticket,
ORDER_TYPE);
      PrintFormat("  [%d] Order %s #%I64u", i, OrderTypeDescription(order_type),
ticket);
     }

//--- 操作ログにポジション取引のリストを表示する
   int deals_total =HistoryDealsTotal();
   for(int i=0; i<deals_total; i++)
     {
      ulong ticket=HistoryDealGetTicket(i);
      if(ticket==0)
         continue;
      ENUM_DEAL_ENTRY deal_entry=(ENUM_DEAL_ENTRY)HistoryDealGetInteger(ticket,
DEAL_ENTRY);
      ENUM_DEAL_TYPE  deal_type= (ENUM_DEAL_TYPE)HistoryDealGetInteger(ticket,
DEAL_TYPE);
      if(deal_type!=DEAL_TYPE_BUY && deal_type!=DEAL_TYPE_SELL)
         continue;
      PrintFormat("  [%d] Entry %s Deal %s #%I64u", i,
DealEntryDescription(deal_entry), DealTypeDescription(deal_type), ticket);
     }
   return(true);
  }
//+------------------------------------------------------------------+
//|  注文タイプの説明を返す                                          |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      case ORDER_TYPE_BUY_LIMIT        :  return("Buy Limit");
      case ORDER_TYPE_SELL_LIMIT       :  return("Sell Limit");
      case ORDER_TYPE_BUY_STOP         :  return("Buy Stop");
      case ORDER_TYPE_SELL_STOP        :  return("Sell Stop");
      case ORDER_TYPE_BUY_STOP_LIMIT   :  return("Buy Stop Limit");
      case ORDER_TYPE_SELL_STOP_LIMIT  :  return("Sell Stop Limit");
```

© 2000–2025, MetaQuotes Ltd.

```
      default                          :  return("Unknown order type: "+(string)
type);
     }
  }
//+------------------------------------------------------------------+
//| ポジション取引のタイプ説明を返す                                 |
//+------------------------------------------------------------------+
string DealTypeDescription(const ENUM_DEAL_TYPE type)
  {
   switch(type)
     {
      //--- BuyおよびSellの取引の説明のみを返す
      //--- （他のすべてのタイプはポジションに適用されないため）
      case DEAL_TYPE_BUY    :  return("Buy");
      case DEAL_TYPE_SELL   :  return("Sell");
      default               :  return("Unknown deal type: "+(string)type);
     }
  }
//+------------------------------------------------------------------+
//| ポジション変更法を返す                                           |
//+------------------------------------------------------------------+
string DealEntryDescription(const ENUM_DEAL_ENTRY entry)
  {
   switch(entry)
     {
      case DEAL_ENTRY_IN      :  return("In");
      case DEAL_ENTRY_OUT     :  return("Out");
      case DEAL_ENTRY_INOUT   :  return("InOut");
      case DEAL_ENTRY_OUT_BY  :  return("Out by");
      case DEAL_ENTRY_STATE   :  return("Status record");
      default                 :  return("Unknown deal entry: "+(string)entry);
     }
  }
```

参照

HistorySelect()、HistoryOrderGetTicket()、注文プロパティ

© 2000-2025, MetaQuotes Ltd.

HistoryOrderSelect

更に適切な関数で呼び出すために履歴内の注文を選択します。関数実行が成功した場合は true を返します。関数実行が失敗した場合 false を返します。エラーの詳細については GetLastError() を呼びます。

```
bool  HistoryOrderSelect(
   ulong  ticket      // 注文チケット
   );
```

パラメータ

ticket

[in] 注文チケット

戻り値

成功の場合は true、それ以外の場合は false。

注意事項

取引履歴の注文を「ツールボックス」バーの「取引」タブに表示される未決注文 と混乱してはいけません。取り消されたまたはトランザクションにつながった 注文のリストはクライアント端末の「ツールボックス」バーの「履歴」タブで見られます。

HistoryOrderSelect() はMQL5 プログラムで参照出来る注文のリストをクリアして、成功した場合、1 つの注文を複製します。HistorySelect()関数で選ばれた約定を1つずつ参照する必要がある場合には HistoryOrderGetTicket() が使用されるべきです。

例：

```
#define   TICKET    1819621374   // 端末の口座履歴などからの、既知の注文のチケット
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- TICKETで指定されたチケットに基づいて、過去の注文を選択する
   if(!HistoryOrderSelect(TICKET))
     {
      PrintFormat("HistoryOrderSelect(%I64u) failed. Error %d", TICKET,
GetLastError());
      return;
     }

//--- 注文が正常に選択された場合、そのデータを取得し、注文の説明を操作ログに表示する
   ENUM_ORDER_TYPE   order_type  = (ENUM_ORDER_TYPE)HistoryOrderGetInteger(TICKET,
ORDER_TYPE);
   ENUM_ORDER_STATE  order_state = (ENUM_ORDER_STATE)HistoryOrderGetInteger(TICKET,
ORDER_STATE);
   ENUM_ORDER_REASON order_reason= (ENUM_ORDER_REASON)HistoryOrderGetInteger(TICKET,
ORDER_REASON);
```

© 2000-2025, MetaQuotes Ltd.

```
   long              order_time  = HistoryOrderGetInteger(TICKET,
ORDER_TIME_SETUP_MSC);
   string            order_symbol=HistoryOrderGetString(TICKET, ORDER_SYMBOL);
   double            order_vol_init=HistoryOrderGetDouble(TICKET,
ORDER_VOLUME_INITIAL);
   double            order_vol_curr=HistoryOrderGetDouble(TICKET,
ORDER_VOLUME_CURRENT);
   PrintFormat("%s Order %.2f/%s %s #%I64u %s by %s at %s",
               order_symbol, order_vol_init, (order_vol_curr>0 ?
DoubleToString(order_vol_curr,2) : "0"),
               OrderTypeDescription(order_type), TICKET,
OrderStateDescription(order_state),
               OrderReasonDescription(order_reason), TimeMscToString(order_time));
   /*
   result for various specified tickets:
   EURUSD Order 0.50/0.50 Buy Limit #2812894647 Canceled by Client at 2024.09.04
19:02:31.793
   EURUSD Order 0.10/0 Sell #1753011743 Filled by Take Profit at 2023.06.12
17:04:20.353
   GBPUSD Order 0.10/0 Buy #1819621374 Filled by Client at 2023.07.24 06:16:25.746
   */
  }
//+------------------------------------------------------------------+
//| 注文タイプの説明を返す                                                  |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      case ORDER_TYPE_BUY_LIMIT        :  return("Buy Limit");
      case ORDER_TYPE_SELL_LIMIT       :  return("Sell Limit");
      case ORDER_TYPE_BUY_STOP         :  return("Buy Stop");
      case ORDER_TYPE_SELL_STOP        :  return("Sell Stop");
      case ORDER_TYPE_BUY_STOP_LIMIT   :  return("Buy Stop Limit");
      case ORDER_TYPE_SELL_STOP_LIMIT  :  return("Sell Stop Limit");
      default                          :  return("Unknown order type: "+(string)
type);
     }
  }
//+------------------------------------------------------------------+
//| 注文ステータスの説明を返す                                               |
//+------------------------------------------------------------------+
string OrderStateDescription(ENUM_ORDER_STATE state)
  {
   switch(state)
     {
      case ORDER_STATE_STARTED         :  return("Started");
      case ORDER_STATE_PLACED          :  return("Placed");
```

© 2000–2025, MetaQuotes Ltd.

```
      case ORDER_STATE_CANCELED        :  return("Canceled");
      case ORDER_STATE_PARTIAL         :  return("Partial");
      case ORDER_STATE_FILLED          :  return("Filled");
      case ORDER_STATE_REJECTED        :  return("Rejected");
      case ORDER_STATE_EXPIRED         :  return("Expired");
      case ORDER_STATE_REQUEST_ADD     :  return("Request Add");
      case ORDER_STATE_REQUEST_MODIFY  :  return("Request Modify");
      case ORDER_STATE_REQUEST_CANCEL  :  return("Request Cancel");
      default                          :  return("Unknown state: "+(string)state);
     }
  }
//+------------------------------------------------------------------+
//| 注文理由の説明を返す                                             |
//+------------------------------------------------------------------+
string OrderReasonDescription(const ENUM_ORDER_REASON reason)
  {
   switch(reason)
     {
      case ORDER_REASON_CLIENT    :  return("Client");
      case ORDER_REASON_MOBILE    :  return("Mobile");
      case ORDER_REASON_WEB       :  return("Web");
      case ORDER_REASON_EXPERT    :  return("Expert");
      case ORDER_REASON_SL        :  return("Stop Loss");
      case ORDER_REASON_TP        :  return("Take Profit");
      case ORDER_REASON_SO        :  return("Stop Out");
      default                     :  return("Unknown reason: "+(string)reason);
     }
  }
//+------------------------------------------------------------------+
//| ミリ秒で時間を返す                                               |
//+------------------------------------------------------------------+
string TimeMscToString(const long time_msc, int flags=TIME_DATE|TIME_MINUTES|
TIME_SECONDS)
  {
   return(TimeToString(time_msc/1000, flags) + "." + IntegerToString(time_msc %1000,
3, '0'));
  }
```

参照

HistorySelect()、HistoryOrderGetTicket()、注文プロパティ

© 2000-2025, MetaQuotes Ltd.

HistoryOrdersTotal

履歴内の注文の数を返します。HistoryOrdersTotal() を呼び出す前に HistorySelect() または HistorySelectByPosition() 関数を使用して約定と注文の履歴を受信するべきです。

```
int  HistoryOrdersTotal();
```

戻り値

int 型の値

注意事項

取引履歴の注文を「ツールボックス」バーの「取引」タブに表示される未決注文 と混乱してはいけません。取り消されたまたはトランザクションにつながった 注文のリストはクライアント端末の「ツールボックス」バーの「履歴」タブで見られます。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 口座の既存のすべての履歴をリクエストする
   if(!HistorySelect(0, TimeCurrent()))
     {
      Print("HistorySelect() failed. Error ", GetLastError());
      return;
     }

//--- リスト内の注文の数を取得し、操作ログに表示する
   int total=HistoryOrdersTotal();
   Print("Number of historical orders on the account: ", total);
   /*
   結果:
   Number of historical orders on the account: 496
   */
  }
```

参照

HistorySelect()、HistoryOrderSelect()、HistoryOrderGetTicket()、注文プロパティ

© 2000-2025, MetaQuotes Ltd.

HistoryOrderGetTicket

履歴内の注文に対応するチケットを返します。HistoryOrderGetTicket() を呼び出す前に HistorySelect() または HistorySelectByPosition() 関数を使用して取引と注文の履歴を受信するべきです。

```
ulong  HistoryOrderGetTicket(
   int  index      // 注文リスト内の番号
   );
```

パラメータ

index

[in] 注文リスト内での注文の番号

戻り値

ulong 型の値（関数が失敗した場合は 0）

注意事項

取引履歴の注文を「ツールボックス」バーの「取引」タブに表示される未決注文と混乱してはいけません。取り消されたまたはトランザクションにつながった注文のリストはクライアント端末の「ツールボックス」バーの「履歴」タブで見られます。

例:

```
void OnStart()
  {
   datetime from=0;
   datetime to=TimeCurrent();
//--- 履歴全部をリクエストする
   HistorySelect(from,to);
//--- 注文プロパティから値を返す変数
   ulong    ticket;
   double   open_price;
   double   initial_volume;
   datetime time_setup;
   datetime time_done;
   string   symbol;
   string   type;
   long     order_magic;
   long     positionID;
//--- 現在の未決注文数
   uint     total=HistoryOrdersTotal();
//--- ループで注文をみる
   for(uint i=0;i<total;i++)
     {
      //--- リスト内の位置によって注文を返す
      if((ticket=HistoryOrderGetTicket(i))>0)
        {
         //--- 注文プロパティを返す
         open_price    =HistoryOrderGetDouble(ticket,ORDER_PRICE_OPEN);
```

© 2000-2025, MetaQuotes Ltd.

```
         time_setup     =(datetime)HistoryOrderGetInteger(ticket,ORDER_TIME_SETUP);
         time_done      =(datetime)HistoryOrderGetInteger(ticket,ORDER_TIME_DONE);
         symbol         =HistoryOrderGetString(ticket,ORDER_SYMBOL);
         order_magic    =HistoryOrderGetInteger(ticket,ORDER_MAGIC);
         positionID     =HistoryOrderGetInteger(ticket,ORDER_POSITION_ID);
         initial_volume=HistoryOrderGetDouble(ticket,ORDER_VOLUME_INITIAL);
         type           =GetOrderType(HistoryOrderGetInteger(ticket,ORDER_TYPE));
         //--- 注文についての情報を準備して表示する
         printf("#ticket %d %s %G %s at %G was set up at %s => done at %s, pos ID=%
d",
                ticket,                    // 注文チケット
                type,                      // 種類
                initial_volume,            // 注文のボリューム
                symbol,                    // シンボル
                open_price,                // 指定された始値
                TimeToString(time_setup),// 注文の出された時刻
                TimeToString(time_done), // 注文実行または削除の時刻
                positionID                 // 注文の約定が含まれたポジションの識別子
                );
        }
      }
//---
  }
//+------------------------------------------------------------------+
//|  注文の種類を文字列として返す                                    |
//+------------------------------------------------------------------+
string GetOrderType(long type)
  {
   string str_type="unknown operation";
   switch(type)
     {
      case (ORDER_TYPE_BUY):             return("buy");
      case (ORDER_TYPE_SELL):            return("sell");
      case (ORDER_TYPE_BUY_LIMIT):       return("buy limit");
      case (ORDER_TYPE_SELL_LIMIT):      return("sell limit");
      case (ORDER_TYPE_BUY_STOP):        return("buy stop");
      case (ORDER_TYPE_SELL_STOP):       return("sell stop");
      case (ORDER_TYPE_BUY_STOP_LIMIT): return("buy stop limit");
      case (ORDER_TYPE_SELL_STOP_LIMIT):return("sell stop limit");
     }
   return(str_type);
  }
```

参照

HistorySelect()、HistoryOrdersTotal()、HistoryOrderSelect()、注文プロパティ

© 2000-2025, MetaQuotes Ltd.

HistoryOrderGetDouble

リクエストされた注文プロパティを返します。注文プロパティは double 型でなければなりません。この関数には 2 つのバージョンがあります。

1. すぐにプロパティ値を返します。

```
double  HistoryOrderGetDouble(
   ulong                        ticket_number,      // チケット
   ENUM_ORDER_PROPERTY_DOUBLE   property_id         // プロパティ識別子
   );
```

2. 関数実行の成功に応じて true または false を返します。成功した場合、プロパティの値は、参照によって渡された最後のパラメータに配置されます。

```
bool  HistoryOrderGetDouble(
   ulong                        ticket_number,      // チケット
   ENUM_ORDER_PROPERTY_DOUBLE   property_id,        // プロパティ識別子
   double&                      double_var          // プロパティ値を受け取る
   );
```

パラメータ

ticket_number

[in] 注文チケット

property_id

[in] 注文プロパティの識別子。値は ENUM_ORDER_PROPERTY_DOUBLE 列挙のいずれかです。

double_var

[out] リクエストされたプロパティの値を受け取るdouble 型の変数

戻り値

double 型の値

注意事項

取引履歴の注文を「ツールボックス」バーの「取引」タブに表示される未決注文 と混乱してはいけません。取り消されたまたはトランザクションにつながった 注文のリストはクライアント端末の「ツールボックス」バーの「履歴」タブで見られます。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 取引及び注文履歴をリクエストする
   if(!HistorySelect(0, TimeCurrent()))
     {
      Print("HistorySelect() failed. Error ", GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }

//--- 口座のすべての履歴注文のリストのループ
   int total=HistoryOrdersTotal();
   for(int i=0; i<total; i++)
     {
      //--- ループインデックスでリストから注文チケットを取得する
      ulong ticket=HistoryOrderGetTicket(i);
      if(ticket==0)
         continue;

      //--- 注文タイプを取得し、選択した注文の実数プロパティのリストのヘッダーを表示する
      string type=OrderTypeDescription((ENUM_ORDER_TYPE)
HistoryOrderGetInteger(ticket, ORDER_TYPE));
      PrintFormat("Double properties of an history order %s #%I64u:", type, ticket);

      //--- ヘッダーの下に選択した注文のすべての実数プロパティを表示する
      HistoryOrderPropertiesDoublePrint(ticket, 16);
     }
   /*
   結果:
   Double properties of an history order Sell #2810847541:
   Volume initial: 0.50
   Volume current: 0.00
   Price open:     1.10491
   StopLoss:       0.00000
   TakeProfit:     0.00000
   Price current:  1.10491
   StopLimit:      0.00000
   Double properties of an history order Buy Limit #2811003507:
   Volume initial: 1.00
   Volume current: 1.00
   Price open:     1.10547
   StopLoss:       0.00000
   TakeProfit:     0.00000
   Price current:  1.10591
   StopLimit:      0.00000
   */
  }
//+------------------------------------------------------------------+
//|  選択した過去の注文の実数プロパティを                                |
//|  操作ログに表示する                                                |
//+------------------------------------------------------------------+
void HistoryOrderPropertiesDoublePrint(const long ticket, const uint header_width=0)
  {
   uint   w=0;
   string header="";
   double value=0;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 注文の銘柄とその小数点以下の桁数を取得する
   string symbol = HistoryOrderGetString(ticket, ORDER_SYMBOL);
   int    digits = (int)SymbolInfoInteger(symbol, SYMBOL_DIGITS);

//--- ヘッダーのテキストとヘッダーフィールドの幅を定義する
//--- ヘッダーの幅が関数に渡され、ゼロと等しい場合、幅はヘッダー行のサイズ+1になる
   header="Volume initial:";
   w=(header_width==0 ? header.Length()+1 : header_width);
//--- 注文を出す際に初期ボリュームを取得し、指定されたヘッダー幅で操作ログに表示する
   if(!HistoryOrderGetDouble(ticket, ORDER_VOLUME_INITIAL, value))
      return;
   PrintFormat("%-*s%-.2f", w, header, value);

//--- 操作ログに未履行の注文ボリュームを表示する
   header="Volume current:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetDouble(ticket, ORDER_VOLUME_CURRENT, value))
      return;
   PrintFormat("%-*s%-.2f", w, header, value);

//--- 操作ログに注文で指定された価格を表示する
   header="Price open:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetDouble(ticket, ORDER_PRICE_OPEN, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- 操作ログにストップロスレベルを表示する
   header="StopLoss:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetDouble(ticket, ORDER_SL, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- 操作ログにテイクプロフィットレベルを表示する
   header="TakeProfit:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetDouble(ticket, ORDER_TP, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- 操作ログに注文の銘柄による現在の価格を表示する
   header="Price current:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetDouble(ticket, ORDER_PRICE_CURRENT, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 操作ログにストップリミット注文がアクティブになったときのリミット注文価格を表示する
   header="StopLimit:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetDouble(ticket, ORDER_PRICE_STOPLIMIT, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);
  }
//+------------------------------------------------------------------+
//| 注文タイプの説明を返す                                           |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      case ORDER_TYPE_BUY_LIMIT        :  return("Buy Limit");
      case ORDER_TYPE_SELL_LIMIT       :  return("Sell Limit");
      case ORDER_TYPE_BUY_STOP         :  return("Buy Stop");
      case ORDER_TYPE_SELL_STOP        :  return("Sell Stop");
      case ORDER_TYPE_BUY_STOP_LIMIT   :  return("Buy Stop Limit");
      case ORDER_TYPE_SELL_STOP_LIMIT  :  return("Sell Stop Limit");
      default                          :  return("Unknown order type");
     }
  }
```

参照

HistorySelect()、HistoryOrdersTotal()、HistoryOrderSelect()、注文プロパティ

© 2000–2025, MetaQuotes Ltd.

HistoryOrderGetInteger

リクエストされた注文プロパティを返します。注文プロパティは datetime または int 型でなければいけません。この関数には 2 つのバージョンがあります。

1. すぐにプロパティ値を返します。

```
long  HistoryOrderGetInteger(
   ulong                         ticket_number,     // チケット
   ENUM_ORDER_PROPERTY_INTEGER   property_id        // プロパティ識別子
   );
```

2. 関数実行の成功に応じて true または false を返します。成功した場合、プロパティの値は、参照によって渡された最後のパラメータに配置されます。

```
bool  HistoryOrderGetInteger(
   ulong                         ticket_number,     // チケット
   ENUM_ORDER_PROPERTY_INTEGER   property_id,       // プロパティ識別子
   long&                         long_var           // プロパティ値を受け取る
   );
```

パラメータ

ticket_number

[in] 注文チケット

property_id

[in] 注文プロパティの識別子。値は ENUM_ORDER_PROPERTY_INTEGER 列挙のいずれかです。

long_var

[out] リクエストされたプロパティの値を受け取る long 型の変数

戻り値

long 型の値

注意事項

取引履歴の注文を「ツールボックス」バーの「取引」タブに表示される未決注文 と混乱してはいけません。取り消されたまたはトランザクションにつながった 注文のリストはクライアント端末の「ツールボックス」バーの「履歴」タブで見られます。

例:

```
//+------------------------------------------------------------------+
//|  取引関数                                                        |
//+------------------------------------------------------------------+
void OnTrade()
  {
//--- 今週の取引履歴から最終注文のチケットを受け取る
   ulong last_order=GetLastOrderTicket();
   if(HistoryOrderSelect(last_order))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 01.01.1970から経過したミリ秒数で表された注文が出された時刻
      long time_setup_msc=HistoryOrderGetInteger(last_order,ORDER_TIME_SETUP_MSC);
      PrintFormat("Order #%d ORDER_TIME_SETUP_MSC=%i64 => %s",
                  last_order,time_setup_msc,TimeToString(time_setup_msc/1000));
      //--- 01.01.1970 から経過したミリ秒数で表された注文の実行/取り消し時刻
      long  time_done_msc=HistoryOrderGetInteger(last_order,ORDER_TIME_DONE_MSC);
      PrintFormat("Order #%d ORDER_TIME_DONE_MSC=%i64 => %s",
                  last_order,time_done_msc,TimeToString(time_done_msc/1000));
     }
   else // 失敗を通知する
      PrintFormat("HistoryOrderSelect() failed for #%d. Eror code=%d",
                  last_order,GetLastError());

//---
  }
//+------------------------------------------------------------------+
//| 履歴内の最終注文チケットまたは -1 を返す                                |
//+------------------------------------------------------------------+
ulong GetLastOrderTicket()
  {
//--- 過去7日の履歴をリクエストする
   if(!GetTradeHistory(7))
     {
      //--- 失敗した呼び出しを通知して -1 を返す
      Print(__FUNCTION__," HistorySelect() returned false");
      return -1;
     }
//---
   ulong first_order,last_order,orders=HistoryOrdersTotal();
//--- あれば注文を操作する
   if(orders>0)
     {
      Print("Orders = ",orders);
      first_order=HistoryOrderGetTicket(0);
      PrintFormat("first_order = %d",first_order);
      if(orders>1)
        {
         last_order=HistoryOrderGetTicket((int)orders-1);
         PrintFormat("last_order = %d",last_order);
         return last_order;
        }
      return first_order;
     }
//--- 注文が見つからないので -1 を返す
   return -1;
  }
//+------------------------------------------------------------------+
//| 過去数日の履歴をリクエストし、障害が発生した場合にはfalseを返す                  |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool GetTradeHistory(int days)
  {
//--- 取引履歴を要求する週の期間を設定する
   datetime to=TimeCurrent();
   datetime from=to-days*PeriodSeconds(PERIOD_D1);
   ResetLastError();
//--- リクエストして結果をチェックする
   if(!HistorySelect(from,to))
     {
      Print(__FUNCTION__," HistorySelect=false. Error code=",GetLastError());
      return false;
     }
//--- 履歴が正常に受信された
   return true;
  }
```

参照

HistorySelect()、HistoryOrdersTotal()、HistoryOrderSelect()、注文プロパティ

© 2000-2025, MetaQuotes Ltd.

HistoryOrderGetString

リクエストされた注文プロパティを返します。注文プロパティは string 型でなければなりません。この関数には 2 つのバージョンがあります。

1. すぐにプロパティ値を返します。

```
string  HistoryOrderGetString(
   ulong                      ticket_number,     // チケット
   ENUM_ORDER_PROPERTY_STRING property_id        // プロパティ識別子
   );
```

2. 関数実行の成功に応じて true または false を返します。成功した場合、プロパティの値は、参照によって渡された最後のパラメータに配置されます。

```
bool  HistoryOrderGetString(
   ulong                      ticket_number,     // チケット
   ENUM_ORDER_PROPERTY_STRING property_id,       // プロパティ識別子
   string&                    string_var         // プロパティ値を受け取る
   );
```

パラメータ

ticket_number

[in] 注文チケット

property_id

[in] 注文プロパティの識別子。値は ENUM_ORDER_PROPERTY_STRING 列挙のいずれかです。

string_var

[out] string 型の変数

戻り値

string 型の変数

注意事項

取引履歴の注文を「ツールボックス」バーの「取引」タブに表示される未決注文 と混乱してはいけません。取り消されたまたはトランザクションにつながった 注文のリストはクライアント端末の「ツールボックス」バーの「履歴」タブで見られます。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 取引及び注文履歴をリクエストする
   if(!HistorySelect(0, TimeCurrent()))
     {
      Print("HistorySelect() failed. Error ", GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }

//--- 口座のすべての履歴注文のリストのループ
   int total=HistoryOrdersTotal();
   for(int i=0; i<total; i++)
     {
      //--- ループインデックスでリストから注文チケットを取得する
      ulong ticket=HistoryOrderGetTicket(i);
      if(ticket==0)
         continue;

      //--- 注文タイプを取得し、選択した注文の文字列プロパティのリストのヘッダーを表示する
      string type=OrderTypeDescription((ENUM_ORDER_TYPE)
HistoryOrderGetInteger(ticket, ORDER_TYPE));
      PrintFormat("String properties of an history order %s #%I64u:", type, ticket);

      //--- ヘッダーの下に選択した注文のすべての文字列プロパティを表示する
      HistoryOrderPropertiesStringPrint(ticket, 16);
     }
   /*
   結果:
   String properties of an history order Buy #2646074112:
   Comment:         [tp 1.09137]
   Symbol:          EURUSD
   External ID:
   String properties of an history order Buy #2646131906:
   Comment:
   Symbol:          EURUSD
   External ID:
   */
  }
//+------------------------------------------------------------------+
//|  選択した過去の注文の文字列プロパティを                                  |
//|  操作ログに表示する                                                 |
//+------------------------------------------------------------------+
void HistoryOrderPropertiesStringPrint(const long ticket, const uint header_width=0)
  {
   uint    w=0;
   string header="";
   string value="";

//--- ヘッダーのテキストとヘッダーフィールドの幅を定義する
//--- ヘッダーの幅が関数に渡され、ゼロと等しい場合、幅はヘッダー行のサイズ+1になる
   header="Comment:";
   w=(header_width==0 ? header.Length()+1 : header_width);
//--- コメントを取得し、指定されたヘッダー幅で操作ログに表示する
   if(!HistoryOrderGetString(ticket, ORDER_COMMENT, value))
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
   PrintFormat("%-*s%-s", w, header, value);

//--- 操作ログに注文が出された銘柄を表示する
   header="Symbol:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetString(ticket, ORDER_SYMBOL, value))
      return;
   PrintFormat("%-*s%-s", w, header, value);

//--- 操作ログに外部取引システムでの注文I Dを表示する
   header="External ID:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetString(ticket, ORDER_EXTERNAL_ID, value))
      return;
   PrintFormat("%-*s%-s", w, header, value);
  }
//+------------------------------------------------------------------+
//|  注文タイプの説明を返す                                          |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      case ORDER_TYPE_BUY_LIMIT        :  return("Buy Limit");
      case ORDER_TYPE_SELL_LIMIT       :  return("Sell Limit");
      case ORDER_TYPE_BUY_STOP         :  return("Buy Stop");
      case ORDER_TYPE_SELL_STOP        :  return("Sell Stop");
      case ORDER_TYPE_BUY_STOP_LIMIT   :  return("Buy Stop Limit");
      case ORDER_TYPE_SELL_STOP_LIMIT  :  return("Sell Stop Limit");
      default                          :  return("Unknown order type: "+(string)
type);
     }
  }
```

参照

HistorySelect()、HistoryOrdersTotal()、HistoryOrderSelect()、注文プロパティ

© 2000-2025, MetaQuotes Ltd.

HistoryDealSelect

更に適切な関数で呼び出すために履歴内の約定を選択します。関数実行が成功した場合は true を返します。関数実行が失敗した場合 false を返します。エラーの詳細については GetLastError() を呼びます。

```
bool  HistoryDealSelect(
   ulong  ticket      // 約定チケット
   );
```

パラメータ

ticket

[in] 約定チケット

戻り値

成功の場合は true、それ以外の場合は false。

注意事項

注文、約定 及び ポジションは混乱されてはなりません。約定は注文実行の結果で、ポジションは 1 つ以上の約定の結果の概要です。

HistoryDealSelect() はMQL5 プログラムで参照出来る約定のリストをクリアして、成功した場合、1 つの約定を複製します。HistorySelect() 関数で選ばれた約定をひとつずつ参照する必要がある場合には HistoryDealGetTicket()が使用されるべきです。

例：

```
#define    TICKET     2620919264    // 端末の口座履歴などからの、既知の取引のチケット

long       ExtTicket=TICKET;       // 指定されたチケットを、スクリプトのテスト用にマクロ置換から変数に設定する
                                   // または、EAのOnTradeTransaction()ハンドラ内で取引を処理する
//+------------------------------------------------------------------+
//| Expert TradeTransactionハンドラ                                  |
//+------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction& trans,
                        const MqlTradeRequest& request,
                        const MqlTradeResult& result)
  {
   //--- 取引が履歴に約定を追加している場合
   if(trans.type==TRADE_TRANSACTION_DEAL_ADD)
     {
      //--- チケットで取引を選択し、そのデータを取得して取引の説明を操作ログに表示する
      HistoryDealSelectProcess(trans.deal);
     }
  }

//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//--- チケットで取引を選択し、そのデータを取得して取引の説明を操作ログに表示する
   HistoryDealSelectProcess(ExtTicket);
   /*
   結果:
   (Position ID #2645974677) EURUSD Deal Out 0.10 Buy #2620919264 by order
#2646028969 at 1.09178, 2024.07.15 18:16:32.570
   */
  }
//+------------------------------------------------------------------+
//|  チケットで取引を選択し、取引データを操作ログに表示する                |
//+------------------------------------------------------------------+
void HistoryDealSelectProcess(const ulong deal_ticket)
  {
//--- deal_ticket で指定されたチケットに基づいて、履歴の取引を選択する
   ResetLastError();
   if(!HistoryDealSelect(deal_ticket))
     {
      PrintFormat("HistoryDealSelect(%I64u) failed. Error %d", deal_ticket,
GetLastError());
      return;
     }

//--- 取引が正常に選択された場合、そのデータを取得し、取引の説明を操作ログに表示する
   ENUM_DEAL_TYPE     deal_type   = (ENUM_DEAL_TYPE)HistoryDealGetInteger(ExtTicket,
DEAL_TYPE);
   ENUM_DEAL_ENTRY    deal_entry  = (ENUM_DEAL_ENTRY)HistoryDealGetInteger(ExtTicket,
DEAL_ENTRY);
   ENUM_DEAL_REASON   deal_reason= (ENUM_DEAL_REASON)HistoryDealGetInteger(ExtTicket,
DEAL_REASON);
   long               deal_time   = HistoryDealGetInteger(ExtTicket, DEAL_TIME_MSC);
   long               deal_order  = HistoryDealGetInteger(ExtTicket, DEAL_ORDER);
   long               deal_pos_id= HistoryDealGetInteger(ExtTicket,
DEAL_POSITION_ID);
   string             deal_symbol= HistoryDealGetString(ExtTicket, DEAL_SYMBOL);
   double             deal_volume= HistoryDealGetDouble(ExtTicket, DEAL_VOLUME);
   double             deal_price  = HistoryDealGetDouble(ExtTicket, DEAL_PRICE);
   int                digits      = (int)SymbolInfoInteger(deal_symbol,
SYMBOL_DIGITS);

   PrintFormat("(Position ID #%I64d) %s Deal %s %.2f %s #%I64u by order #%I64d at
%.*f, %s",
               deal_pos_id, deal_symbol, DealEntryDescription(deal_entry),
deal_volume,
               DealTypeDescription(deal_type), ExtTicket, deal_order, digits,
deal_price,
               TimeMscToString(deal_time));
  }
//+------------------------------------------------------------------+
//|  取引タイプの説明を返す                                          |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
string DealTypeDescription(const ENUM_DEAL_TYPE type)
  {
   switch(type)
     {
      case DEAL_TYPE_BUY                    :  return("Buy");
      case DEAL_TYPE_SELL                   :  return("Sell");
      case DEAL_TYPE_BALANCE                :  return("Balance");
      case DEAL_TYPE_CREDIT                 :  return("Credit");
      case DEAL_TYPE_CHARGE                 :  return("Additional charge");
      case DEAL_TYPE_CORRECTION             :  return("Correction");
      case DEAL_TYPE_BONUS                  :  return("Bonus");
      case DEAL_TYPE_COMMISSION             :  return("Additional commission");
      case DEAL_TYPE_COMMISSION_DAILY       :  return("Daily commission");
      case DEAL_TYPE_COMMISSION_MONTHLY     :  return("Monthly commission");
      case DEAL_TYPE_COMMISSION_AGENT_DAILY :  return("Daily agent commission");
      case DEAL_TYPE_COMMISSION_AGENT_MONTHLY: return("Monthly agent commission");
      case DEAL_TYPE_INTEREST               :  return("Interest rate");
      case DEAL_TYPE_BUY_CANCELED           :  return("Canceled buy deal");
      case DEAL_TYPE_SELL_CANCELED          :  return("Canceled sell deal");
      case DEAL_DIVIDEND                    :  return("Dividend operations");
      case DEAL_DIVIDEND_FRANKED            :  return("Franked (non-taxable)
dividend operations");
      case DEAL_TAX                         :  return("Tax charges");
      default                               :  return("Unknown deal type:
"+(string)type);
     }
  }
//+------------------------------------------------------------------+
//| ポジション変更法を返す                                           |
//+------------------------------------------------------------------+
string DealEntryDescription(const ENUM_DEAL_ENTRY entry)
  {
   switch(entry)
     {
      case DEAL_ENTRY_IN      :  return("In");
      case DEAL_ENTRY_OUT     :  return("Out");
      case DEAL_ENTRY_INOUT   :  return("Reverce");
      case DEAL_ENTRY_OUT_BY  :  return("Out by");
      case DEAL_ENTRY_STATE   :  return("Status record");
      default                 :  return("Unknown deal entry: "+(string)entry);
     }
  }
//+------------------------------------------------------------------+
//| ミリ秒で時間を返す                                               |
//+------------------------------------------------------------------+
string TimeMscToString(const long time_msc, int flags=TIME_DATE|TIME_MINUTES|
TIME_SECONDS)
  {
```

© 2000–2025, MetaQuotes Ltd.

```
   return(TimeToString(time_msc/1000, flags) + "." + IntegerToString(time_msc %1000,
3, '0'));
  }
```

参照

HistorySelect()、HistoryDealGetTicket()、約定プロパティ

© 2000–2025, MetaQuotes Ltd.

HistoryDealsTotal

履歴内の約定数を返します。HistoryDealsTotal() を呼び出す前に HistorySelect() または HistorySelectByPosition() 関数を使用して約定と注文の履歴を受信するべきです。

```
int  HistoryDealsTotal();
```

戻り値

int 型の値

注意事項

注文、約定 及び ポジションは混乱されてはなりません。約定は注文実行の結果で、ポジションは 1 つ以上の約定の結果の概要です。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 口座の既存のすべての履歴をリクエストする
   if(!HistorySelect(0, TimeCurrent()))
     {
      Print("HistorySelect() failed. Error ", GetLastError());
      return;
     }

//--- リスト内の取引の数を取得し、操作ログに表示する
   int total=HistoryDealsTotal();
   Print("Number of historical deals on the account: ", total);
   /*
   結果:
   Number of historical deals on the account: 339
   */
  }
```

参照

HistorySelect()、HistoryDealGetTicket()、約定プロパティ

© 2000-2025, MetaQuotes Ltd.

HistoryDealGetTicket

この関数は、さらなる処理のための約定を選択し、履歴中の約定チケットを返します。
HistoryDealGetTicket() を呼び出す前に HistorySelect() または HistorySelectByPosition() 関数を使用して約定と注文の履歴を受信するべきです。

```
ulong  HistoryDealGetTicket(
   int  index      // 約定チケット
   );
```

パラメータ

index

[in] リスト中の約定番号

戻り値

ulong 型の値（関数が失敗した場合は 0）

注意事項

注文、約定 及び ポジションは混乱されてはなりません。約定は注文実行の結果で、ポジションは 1 つ以上の約定の結果の概要です。

例:

```
void OnStart()
  {
   ulong deal_ticket;             // 約定チケット
   ulong order_ticket;            // 約定が実行された注文のチケット
   datetime transaction_time;     // 約定実行時刻
   long deal_type ;               // 取引操作の種類
   long position_ID;              // ポジション識別子
   string deal_description;       // 操作の説明
   double volume;                 // 操作のボリューム
   string symbol;                 // 約定シンボル
//--- 約定履歴をリクエストするのに開始日と終了日を設定する
   datetime from_date=0;          // 一番初めから
   datetime to_date=TimeCurrent();// 今まで
//--- 指定された期間の約定履歴をリクエストする
   HistorySelect(from_date,to_date);
//--- リスト中の約定の数の合計
   int deals=HistoryDealsTotal();
//--- 取引をひとつづつ処理する
   for(int i=0;i<deals;i++)
     {
      deal_ticket=               HistoryDealGetTicket(i);
      volume=                    HistoryDealGetDouble(deal_ticket,DEAL_VOLUME);
      transaction_time=(datetime)HistoryDealGetInteger(deal_ticket,DEAL_TIME);
      order_ticket=              HistoryDealGetInteger(deal_ticket,DEAL_ORDER);
      deal_type=                 HistoryDealGetInteger(deal_ticket,DEAL_TYPE);
      symbol=                    HistoryDealGetString(deal_ticket,DEAL_SYMBOL);
```

© 2000-2025, MetaQuotes Ltd.

```
      position_ID=
HistoryDealGetInteger(deal_ticket,DEAL_POSITION_ID);
      deal_description=
GetDealDescription(deal_type,volume,symbol,order_ticket,position_ID);
      //--- 約定番号をフォーマットする
      string print_index=StringFormat("% 3d",i);
      //--- 約定の情報を表示する
      Print(print_index+": deal #",deal_ticket," at
",transaction_time,deal_description);
     }
  }
//+------------------------------------------------------------------+
//| 操作の説明を文字列として返す                                     |
//+------------------------------------------------------------------+
string GetDealDescription(long deal_type,double volume,string symbol,long
ticket,long pos_ID)
  {
   string descr;
//---
   switch(deal_type)
     {
      case DEAL_TYPE_BALANCE:                  return ("balance");
      case DEAL_TYPE_CREDIT:                   return ("credit");
      case DEAL_TYPE_CHARGE:                   return ("charge");
      case DEAL_TYPE_CORRECTION:               return ("correction");
      case DEAL_TYPE_BUY:                      descr="buy"; break;
      case DEAL_TYPE_SELL:                     descr="sell"; break;
      case DEAL_TYPE_BONUS:                    return ("bonus");
      case DEAL_TYPE_COMMISSION:               return ("additional commission");
      case DEAL_TYPE_COMMISSION_DAILY:         return ("daily commission");
      case DEAL_TYPE_COMMISSION_MONTHLY:       return ("monthly commission");
      case DEAL_TYPE_COMMISSION_AGENT_DAILY:   return ("daily agent commission");
      case DEAL_TYPE_COMMISSION_AGENT_MONTHLY: return ("monthly agent commission");
      case DEAL_TYPE_INTEREST:                 return ("interest rate");
      case DEAL_TYPE_BUY_CANCELED:             descr="cancelled buy deal"; break;
      case DEAL_TYPE_SELL_CANCELED:            descr="cancelled sell deal"; break;
     }
   descr=StringFormat("%s %G %s (order #%d, position ID %d)",
                      descr,  // 現在の説明
                      volume, // 約定高
                      symbol, // 約定シンボル
                      ticket, // 約定が実行された注文のチケット
                      pos_ID  // 約定が含まれたポジションの識別子
                      );
   return(descr);
//---
  }
```

© 2000–2025, MetaQuotes Ltd.

参照

HistorySelect()、HistoryDealsTotal()、HistoryDealSelect()、約定プロパティ

© 2000–2025, MetaQuotes Ltd.

HistoryDealGetDouble

約定のリクエストされたプロパティを返します。約定プロパティは double 型でなければいけません。この関数には 2 つのバージョンがあります。

1. すぐにプロパティ値を返します。

```
double  HistoryDealGetDouble(
   ulong                      ticket_number,     // チケット
   ENUM_DEAL_PROPERTY_DOUBLE  property_id        // プロパティ識別子
   );
```

2. 関数実行の成功に応じて true または false を返します。成功した場合、プロパティの値は、参照によって渡された最後のパラメータに配置されます。

```
bool  HistoryDealGetDouble(
   ulong                      ticket_number,     // チケット
   ENUM_DEAL_PROPERTY_DOUBLE  property_id,       // プロパティ識別子
   double&                    double_var         // プロパティ値を受け取る
   );
```

パラメータ

ticket_number

[in] 約定チケット

property_id

[in] 約定プロパティ識別子。値は ENUM_DEAL_PROPERTY_DOUBLE 列挙のいずれかです。

double_var

[out] リクエストされたプロパティの値を受け取るdouble 型の変数

戻り値

double 型の値

注意事項

注文、約定 及び ポジションは混乱されてはなりません。約定は注文実行の結果で、ポジションは 1 つ以上の約定の結果の概要です。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 取引及び注文履歴をリクエストする
   if(!HistorySelect(0, TimeCurrent()))
     {
      Print("HistorySelect() failed. Error ", GetLastError());
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
      }

//--- 口座履歴の取引リストによるループ
   int total=HistoryDealsTotal();
   for(int i=0; i<total; i++)
     {
      //--- 次の取引のチケットを取得する(取引はプロパティを取得するために自動的に選択される)
      ulong ticket=HistoryDealGetTicket(i);
      if(ticket==0)
         continue;

      //--- 選択した取引の実数プロパティのリストのヘッダーを表示するために、取引のタイプと方向を取得する
      string type=DealTypeDescription((ENUM_DEAL_TYPE)HistoryDealGetInteger(ticket,
DEAL_TYPE));
      string entry=DealEntryDescription((ENUM_DEAL_ENTRY)
HistoryDealGetInteger(ticket, DEAL_ENTRY));
      PrintFormat("Double properties of an deal %s entry %s #%I64u:", type, entry,
ticket);

      //--- 選択した取引のすべての実数プロパティをヘッダーの下に表示する
      HistoryDealPropertiesDoublePrint(ticket, 12);
     }
   /*
   real:
   Double properties of an deal Buy entry In #2785070622:
   Volume:      0.50
   Price:       1.10480
   Commission: 0.00
   Swap:        0.00
   Profit:      0.00 USD
   Fee:         0.00
   StopLoss:    0.00000
   TakeProfit: 0.00000
   Double properties of an deal Sell entry Out #2785071538:
   Volume:      0.50
   Price:       1.10491
   Commission: 0.00
   Swap:        0.00
   Profit:      5.50 USD
   Fee:         0.00
   StopLoss:    0.00000
   TakeProfit: 0.00000
   */
  }
//+------------------------------------------------------------------+
//|  選択した取引の実数プロパティを操作ログに表示する                                |
//+------------------------------------------------------------------+
void HistoryDealPropertiesDoublePrint(const ulong ticket, const uint header_width=0)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   uint    w=0;
   string header="";
   double value=0;

//--- 銘柄、利益通貨、銘柄の小数点以下の桁数を取得する
   string symbol  = HistoryDealGetString(ticket, DEAL_SYMBOL);
   string currency= SymbolInfoString(symbol, SYMBOL_CURRENCY_PROFIT);
   int    digits  = (int)SymbolInfoInteger(symbol, SYMBOL_DIGITS);

//--- ヘッダーのテキストとヘッダーフィールドの幅を定義する
//--- ヘッダーの幅が関数に渡され、ゼロと等しい場合、幅はヘッダー行のサイズ+1になる
   header="Volume:";
   w=(header_width==0 ? header.Length()+1 : header_width);
//--- 取引量を取得し、指定されたヘッダー幅で操作ログに表示する
   if(!HistoryDealGetDouble(ticket, DEAL_VOLUME, value))
      return;
   PrintFormat("%-*s%-.2f", w, header, value);

//--- 操作ログに取引価格を表示する
   header="Price:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetDouble(ticket, DEAL_PRICE, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- 操作ログに取引手数料を表示する
   header="Commission:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetDouble(ticket, DEAL_COMMISSION, value))
      return;
   PrintFormat("%-*s%-.2f", w, header, value);

//--- 閉じる際に、操作ログに累積スワップを表示する
   header="Swap:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetDouble(ticket, DEAL_SWAP, value))
      return;
   PrintFormat("%-*s%-.2f", w, header, value);

//--- 操作ログに取引の財務結果を表示する
   header="Profit:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetDouble(ticket, DEAL_PROFIT, value))
      return;
   PrintFormat("%-*s%-.2f %s", w, header, value, currency);

//--- 操作ログに取引の手数料を表示する
   header="Fee:";
   w=(header_width==0 ? header.Length()+1 : header_width);
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!HistoryDealGetDouble(ticket, DEAL_FEE, value))
      return;
   PrintFormat("%-*s%-.2f", w, header, value);

//--- 操作ログにストップロスレベルを表示する
   header="StopLoss:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetDouble(ticket, DEAL_SL, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- 操作ログにテイクプロフィットレベルを表示する
   header="TakeProfit:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetDouble(ticket, DEAL_TP, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);
  }
//+------------------------------------------------------------------+
//|  取引タイプの説明を返す                                          |
//+------------------------------------------------------------------+
string DealTypeDescription(const ENUM_DEAL_TYPE type)
  {
   switch(type)
     {
      case DEAL_TYPE_BUY                     :  return("Buy");
      case DEAL_TYPE_SELL                    :  return("Sell");
      case DEAL_TYPE_BALANCE                 :  return("Balance");
      case DEAL_TYPE_CREDIT                  :  return("Credit");
      case DEAL_TYPE_CHARGE                  :  return("Additional charge");
      case DEAL_TYPE_CORRECTION              :  return("Correction");
      case DEAL_TYPE_BONUS                   :  return("Bonus");
      case DEAL_TYPE_COMMISSION              :  return("Additional commission");
      case DEAL_TYPE_COMMISSION_DAILY        :  return("Daily commission");
      case DEAL_TYPE_COMMISSION_MONTHLY      :  return("Monthly commission");
      case DEAL_TYPE_COMMISSION_AGENT_DAILY  :  return("Daily agent commission");
      case DEAL_TYPE_COMMISSION_AGENT_MONTHLY:  return("Monthly agent commission");
      case DEAL_TYPE_INTEREST                :  return("Interest rate");
      case DEAL_TYPE_BUY_CANCELED            :  return("Canceled buy deal");
      case DEAL_TYPE_SELL_CANCELED           :  return("Canceled sell deal");
      case DEAL_DIVIDEND                     :  return("Dividend operations");
      case DEAL_DIVIDEND_FRANKED             :  return("Franked (non-taxable)
dividend operations");
      case DEAL_TAX                          :  return("Tax charges");
      default                                :  return("Unknown deal type:
"+(string)type);
     }
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|  ポジション変更法を返す                                                  |
//+------------------------------------------------------------------+
string DealEntryDescription(const ENUM_DEAL_ENTRY entry)
  {
   switch(entry)
     {
      case DEAL_ENTRY_IN      :  return("In");
      case DEAL_ENTRY_OUT     :  return("Out");
      case DEAL_ENTRY_INOUT   :  return("Reverce");
      case DEAL_ENTRY_OUT_BY  :  return("Out by");
      case DEAL_ENTRY_STATE   :  return("Status record");
      default                 :  return("Unknown deal entry: "+(string)entry);
     }
  }
```

参照

HistorySelect()、HistoryDealsTotal()、HistoryDealGetTicket()、約定プロパティ

© 2000–2025, MetaQuotes Ltd.

HistoryDealGetInteger

約定のリクエストされたプロパティを返します。約定プロパティは datetime または int 型でなければいけません。この関数には 2 つのバージョンがあります。

1. すぐにプロパティ値を返します。

```
long  HistoryDealGetInteger(
   ulong                      ticket_number,     // チケット
   ENUM_DEAL_PROPERTY_INTEGER property_id        // プロパティ識別子
   );
```

2. 関数実行の成功に応じて true または false を返します。成功した場合、プロパティの値は、参照によって渡された最後のパラメータに配置されます。

```
bool  HistoryDealGetInteger(
   ulong                      ticket_number,     // チケット
   ENUM_DEAL_PROPERTY_INTEGER property_id,       // プロパティ識別子
   long&                      long_var           // プロパティ値を受け取る
   );
```

パラメータ

ticket_number

[in] 取引チケット

property_id

[in] 約定プロパティ識別子値は ENUM_DEAL_PROPERTY_INTEGER 列挙のいずれかです。

long_var

[out] リクエストされたプロパティの値を受け取る long 型の変数

戻り値

long 型の値

注意事項

注文、約定 及び ポジションは混乱されてはなりません。約定は注文実行の結果で、ポジションは 1 つ以上の約定の結果の概要です。

例:

```
//+------------------------------------------------------------------+
//|  取引関数                                                        |
//+------------------------------------------------------------------+
void OnTrade()
  {
//--- 今週の取引履歴から直近約定のチケットを受け取る
   ulong last_deal=GetLastDealTicket();
   if(HistoryDealSelect(last_deal))
     {
      //--- 01.01.1970から経過したミリ秒数で表された約定の実行時刻
```

© 2000-2025, MetaQuotes Ltd.

```
      long deal_time_msc=HistoryDealGetInteger(last_deal,DEAL_TIME_MSC);
      PrintFormat("Deal #%d DEAL_TIME_MSC=%i64 => %s",
                  last_deal,deal_time_msc,TimeToString(deal_time_msc/1000));
     }
   else
      PrintFormat("HistoryDealSelect() failed for #%d. Eror code=%d",
                  last_deal,GetLastError());
//---
  }
//+------------------------------------------------------------------+
//| 履歴内の最終約定チケットまたは -1 を返す                          |
//+------------------------------------------------------------------+
ulong GetLastDealTicket()
  {
//--- 過去7日の履歴をリクエストする
   if(!GetTradeHistory(7))
     {
      //--- 失敗した呼び出しを通知して -1 を返す
      Print(__FUNCTION__," HistorySelect() returned false");
      return -1;
     }
//---
   ulong first_deal,last_deal,deals=HistoryDealsTotal();
//--- あれば注文を操作する
   if(deals>0)
     {
      Print("Deals = ",deals);
      first_deal=HistoryDealGetTicket(0);
      PrintFormat("first_deal = %d",first_deal);
      if(deals>1)
        {
         last_deal=HistoryDealGetTicket((int)deals-1);
         PrintFormat("last_deal = %d",last_deal);
         return last_deal;
        }
      return first_deal;
     }
//--- 約定が見つからないので -1 を返す
   return -1;
  }
//+------------------------------------------------------------------+
//| 過去数日の履歴をリクエストし、障害が発生した場合にはfalseを返す  |
//+------------------------------------------------------------------+
bool GetTradeHistory(int days)
  {
//--- 取引履歴を要求する週の期間を設定する
   datetime to=TimeCurrent();
   datetime from=to-days*PeriodSeconds(PERIOD_D1);
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- リクエストして結果をチェックする
   if(!HistorySelect(from,to))
     {
      Print(__FUNCTION__," HistorySelect=false. Error code=",GetLastError());
      return false;
     }
//--- 履歴が正常に受信された
   return true;
  }
```

参照

HistoryDealsTotal()、HistorySelect()、HistoryDealGetTicket()、約定プロパティ

© 2000-2025, MetaQuotes Ltd.

HistoryDealGetString

約定のリクエストされたプロパティを返します。約定プロパティは string 型でなければなりません。この関数には 2 つのバージョンがあります。

1. すぐにプロパティ値を返します。

```
string  HistoryDealGetString(
   ulong                      ticket_number,     // チケット
   ENUM_DEAL_PROPERTY_STRING  property_id        // プロパティ識別子
   );
```

2. 関数実行の成功に応じて true または false を返します。成功した場合、プロパティの値は、参照によって渡された最後のパラメータに配置されます。

```
bool  HistoryDealGetString(
   ulong                      ticket_number,     // チケット
   ENUM_DEAL_PROPERTY_STRING  property_id,       // プロパティ識別子
   string&                    string_var         // プロパティ値を受け取る
   );
```

パラメータ

ticket_number

[in] 約定チケット

property_id

[in] 約定プロパティ識別子値は ENUM_DEAL_PROPERTY_STRING 列挙のいずれかです。

string_var

[out] リクエストされたプロパティ値を受け取るstring 型の変数

戻り値

string 型の変数

注意事項

注文、約定 及び ポジションは混乱されてはなりません。約定は注文実行の結果で、ポジションは 1 つ以上の約定の結果の概要です。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 取引及び注文履歴をリクエストする
   if(!HistorySelect(0, TimeCurrent()))
     {
      Print("HistorySelect() failed. Error ", GetLastError());
      return;
```

© 2000–2025, MetaQuotes Ltd.

```
      }

//--- 口座履歴の取引リストによるループ
   int total=HistoryDealsTotal();
   for(int i=0; i<total; i++)
     {
      //--- 次の取引のチケットを取得する(取引はプロパティを取得するために自動的に選択される)
      ulong ticket=HistoryDealGetTicket(i);
      if(ticket==0)
         continue;

      //--- 選択した取引の実数プロパティのリストのヘッダーを表示するために、取引のタイプと方向を取得する
      string type=DealTypeDescription((ENUM_DEAL_TYPE)HistoryDealGetInteger(ticket,
DEAL_TYPE));
      string entry=DealEntryDescription((ENUM_DEAL_ENTRY)
HistoryDealGetInteger(ticket, DEAL_ENTRY));
      PrintFormat("String properties of an deal %s entry %s #%I64u:", type, entry,
ticket);

      //--- 選択した取引のすべての実数プロパティをヘッダーの下に表示する
      HistoryDealPropertiesStringPrint(ticket, 13);
     }
   /*
   結果:
   String properties of an deal Buy entry In #2785021084:
   Symbol:       EURUSD
   Comment:      Test PositionGetString
   Extarnal ID:
   String properties of an deal Buy entry Out #2497993663:
   Symbol:       EURUSD
   Comment:      [tp 1.08639]
   Extarnal ID:
   */
  }
//+------------------------------------------------------------------+
//|  選択した取引の文字列プロパティを操作ログに表示する                                |
//+------------------------------------------------------------------+
void HistoryDealPropertiesStringPrint(const ulong ticket, const uint header_width=0)
  {
   uint    w=0;
   string header="";
   string value ="";

//--- ヘッダーのテキストとヘッダーフィールドの幅を定義する
//--- ヘッダーの幅が関数に渡され、ゼロと等しい場合、幅はヘッダー行のサイズ+1になる
   header="Symbol:";
   w=(header_width==0 ? header.Length()+1 : header_width);
//--- 取引銘柄を取得し、指定されたヘッダー幅で操作ログに表示する
   if(!HistoryDealGetString(ticket, DEAL_SYMBOL, value))
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
   PrintFormat("%-*s%-s", w, header, value);

//--- 操作ログに取引コメントを表示する
   header="Comment:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetString(ticket, DEAL_COMMENT, value))
      return;
   PrintFormat("%-*s%-s", w, header, value);

//--- 外部取引システムでの約定ID を表示する
   header="Extarnal ID:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetString(ticket, DEAL_EXTERNAL_ID, value))
      return;
   PrintFormat("%-*s%-s", w, header, value);
  }
//+------------------------------------------------------------------+
//|  取引タイプの説明を返す                                          |
//+------------------------------------------------------------------+
string DealTypeDescription(const ENUM_DEAL_TYPE type)
  {
   switch(type)
     {
      case DEAL_TYPE_BUY                    :  return("Buy");
      case DEAL_TYPE_SELL                   :  return("Sell");
      case DEAL_TYPE_BALANCE                :  return("Balance");
      case DEAL_TYPE_CREDIT                 :  return("Credit");
      case DEAL_TYPE_CHARGE                 :  return("Additional charge");
      case DEAL_TYPE_CORRECTION             :  return("Correction");
      case DEAL_TYPE_BONUS                  :  return("Bonus");
      case DEAL_TYPE_COMMISSION             :  return("Additional commission");
      case DEAL_TYPE_COMMISSION_DAILY       :  return("Daily commission");
      case DEAL_TYPE_COMMISSION_MONTHLY     :  return("Monthly commission");
      case DEAL_TYPE_COMMISSION_AGENT_DAILY :  return("Daily agent commission");
      case DEAL_TYPE_COMMISSION_AGENT_MONTHLY: return("Monthly agent commission");
      case DEAL_TYPE_INTEREST               :  return("Interest rate");
      case DEAL_TYPE_BUY_CANCELED           :  return("Canceled buy deal");
      case DEAL_TYPE_SELL_CANCELED          :  return("Canceled sell deal");
      case DEAL_DIVIDEND                    :  return("Dividend operations");
      case DEAL_DIVIDEND_FRANKED            :  return("Franked (non-taxable)
dividend operations");
      case DEAL_TAX                         :  return("Tax charges");
      default                               :  return("Unknown deal type:
"+(string)type);
     }
  }
//+------------------------------------------------------------------+
//|  ポジション変更法を返す                                          |
```

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
string DealEntryDescription(const ENUM_DEAL_ENTRY entry)
  {
   switch(entry)
     {
      case DEAL_ENTRY_IN      :  return("In");
      case DEAL_ENTRY_OUT     :  return("Out");
      case DEAL_ENTRY_INOUT   :  return("Reverce");
      case DEAL_ENTRY_OUT_BY  :  return("Out by");
      case DEAL_ENTRY_STATE   :  return("Status record");
      default                 :  return("Unknown deal entry: "+(string)entry);
     }
  }
```

参照

HistoryDealsTotal()、HistorySelect()、HistoryDealGetTicket()、約定プロパティ

© 2000–2025, MetaQuotes Ltd.

売買シグナル

これは売買シグナルを管理する関数群です。これらの関数は下記を可能にします。

- コピー可能な売買シグナル情報の取得
- シグナルコピー設定の取得と設定
- MQL5 言語関数を使用したシグナルコピーへのサブスクライブ/アンサブスクライブ

関数	アクション
SignalBaseGetDouble	選択されたシグナルの double 型プロパティの値を返します。
SignalBaseGetInteger	選択されたシグナルの整数型プロパティの値を返します。
SignalBaseGetString	選択されたシグナルの string 型プロパティの値を返します。
SignalBaseSelect	端末で利用可能なシグナルから操作を続けるためにシグナルを選択します。
SignalBaseTotal	端末で使用可能なシグナルの合計量を返します。
SignalInfoGetDouble	シグナルコピー設定の double 型プロパティの値を返します。
SignalInfoGetInteger	シグナルコピー設定の整数型プロパティの値を返します。
SignalInfoGetString	シグナルコピー設定の string 型プロパティの値を返します。
SignalInfoSetDouble	シグナルコピー設定の double 型プロパティの値を返します。
SignalInfoSetInteger	シグナルコピー設定の整数型プロパティの値を設定します。
SignalSubscribe	売買シグナルにサブスクライブします。
SignalUnsubscribe	サブスクライブをキャンセルします。

© 2000–2025, MetaQuotes Ltd.

SignalBaseGetDouble

選択されたシグナルの double 型プロパティの値を返します。

```
double  SignalBaseGetDouble(
   ENUM_SIGNAL_BASE_DOUBLE     property_id,     // プロパティ識別子
   );
```

パラメータ

property_id

[in] シグナルプロパティ識別子。値は ENUM_SIGNAL_BASE_DOUBLE 列挙のいずれかです。

戻り値

選択されたシグナルの double 型プロパティの値

© 2000–2025, MetaQuotes Ltd.

SignalBaseGetInteger

選択されたシグナルの整数型プロパティの値を返します。

```
long  SignalBaseGetInteger(
   ENUM_SIGNAL_BASE_INTEGER     property_id,     // プロパティ識別子
   );
```

パラメータ

property_id

[in] シグナルプロパティ識別子。値は ENUM_SIGNAL_BASE_INTEGER 列挙のいずれかです。

戻り値

選択されたシグナルの整数型プロパティの値

© 2000–2025, MetaQuotes Ltd.

SignalBaseGetString

選択されたシグナルの string 型プロパティの値を返します。

```
string  SignalBaseGetString(
   ENUM_SIGNAL_BASE_STRING     property_id,     // プロパティ識別子
   );
```

パラメータ

property_id

[in] シグナルプロパティ識別子。値は ENUM_SIGNAL_BASE_STRING 列挙のいずれかです。

戻り値

選択されたシグナルのstring 型プロパティの値

© 2000–2025, MetaQuotes Ltd.

SignalBaseSelect

端末で利用可能なシグナルから操作を続けるためにシグナルを選択します。

```
bool  SignalBaseSelect(
   int      index      // シグナルインデックス
   );
```

パラメータ

index

[in] 売買シグナルのベースにおけるシグナルインデックス

戻り値

成功の場合は true、それ以外の場合は false。エラーの詳細を読むには GetLastError() を呼びます。

例:

```
void OnStart()
  {
//--- 端末のシグナルの総量を取得する
   int total=SignalBaseTotal();
//--- 全てのシグナルを処理する
   for(int i=0;i<total;i++)
     {
      //--- シグナルをインデックスで選択する
      if(SignalBaseSelect(i))
        {
         //--- シグナルプロパティを取得する
         long   id    =SignalBaseGetInteger(SIGNAL_BASE_ID);           // シグナルＩＤ
         long   pips  =SignalBaseGetInteger(SIGNAL_BASE_PIPS);         // ピップス単位での
利益
         long   subscr=SignalBaseGetInteger(SIGNAL_BASE_SUBSCRIBERS); // サブスクライバの
数
         string name  =SignalBaseGetString(SIGNAL_BASE_NAME);          // シグナル名
         double price =SignalBaseGetDouble(SIGNAL_BASE_PRICE);         // シグナル価格
         string curr  =SignalBaseGetString(SIGNAL_BASE_CURRENCY);      // シグナル通貨
         //--- 全ての収益性の高い無料の信号をサブスクライバとともに出力する
         if(price==0.0 && pips>0 && subscr>0)
            PrintFormat("id=%d, name=\"%s\", currency=%s, pips=%d, subscribers=%
d",id,name,curr,pips,subscr);
        }
      else PrintFormat("Error in call of SignalBaseSelect. Error code=%
d",GetLastError());
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

SignalBaseTotal

端末で使用可能なシグナルの総量を返します。

```
int  SignalBaseTotal();
```

戻り値

端末で使用可能なシグナルの総量

© 2000-2025, MetaQuotes Ltd.

SignalInfoGetDouble

シグナルコピー設定の double 型プロパティの値を返します。

```
double  SignalInfoGetDouble(
   ENUM_SIGNAL_INFO_DOUBLE     property_id,     // プロパティ識別子
   );
```

パラメータ

property_id

[in] シグナルコピー設定プロパティ識別子。値は ENUM_SIGNAL_INFO_DOUBLE 列挙のいずれかです。

戻り値

シグナルコピー設定の double 型プロパティの値

© 2000-2025, MetaQuotes Ltd.

SignalInfoGetInteger

シグナルコピー設定の整数型プロパティの値を返します。

```
long  SignalInfoGetInteger(
   ENUM_SIGNAL_INFO_INTEGER     property_id,      // プロパティ識別子
   );
```

パラメータ

property_id

[in] シグナルコピー設定プロパティ識別子。値は ENUM_SIGNAL_INFO_INTEGER 列挙のいずれかです。

戻り値

シグナルコピー設定の整数型プロパティの値

© 2000–2025, MetaQuotes Ltd.

SignalInfoGetString

シグナルコピー設定の string 型プロパティの値を返します。

```
string  SignalInfoGetString(
   ENUM_SIGNAL_INFO_STRING     property_id,     // プロパティ識別子
   );
```

パラメータ

property_id

[in] シグナルコピー設定プロパティ識別子。値は ENUM_SIGNAL_INFO_STRING 列挙のいずれかです。

戻り値

シグナルコピー設定の string 型プロパティの値

© 2000–2025, MetaQuotes Ltd.

SignalInfoSetDouble

シグナルコピー設定の double 型プロパティの値を設定します。

```
bool  SignalInfoSetDouble(
   ENUM_SIGNAL_INFO_DOUBLE       property_id,      // プロパティ識別子
   double                        value             // 新しい値
   );
```

パラメータ

property_id

[in] シグナルコピー設定プロパティ識別子。値は ENUM_SIGNAL_INFO_DOUBLE 列挙のいずれかです。

value

[in] シグナルコピー設定プロパティの値

戻り値

プロパティが変更された場合は true、それ以外の場合は false。エラーの詳細を読むには GetLastError() を呼びます。

© 2000–2025, MetaQuotes Ltd.

SignalInfoSetInteger

シグナルコピー設定の整数型プロパティの値を設定します。

```
bool  SignalInfoSetInteger(
   ENUM_SIGNAL_INFO_INTEGER     property_id,     // プロパティ識別子
   long                         value            // 新しい値
   );
```

パラメータ

property_id

[in] シグナルコピー設定プロパティ識別子。値は ENUM_SIGNAL_INFO_INTEGER 列挙のいずれかです。

value

[in] シグナルコピー設定プロパティの値

戻り値

プロパティが変更された場合は true、それ以外の場合は false。エラーの詳細を読むには GetLastError() を呼びます。

© 2000–2025, MetaQuotes Ltd.

SignalSubscribe

売買シグナルにサブスクライブします。

```
bool  SignalSubscribe(
   long     signal_id     // シグナル識別子
   );
```

パラメータ

signal_id

[in] シグナル識別子

戻り値

サブスクライブが成功の場合は true、それ以外の場合は false。エラーの詳細を読むには GetLastError() を呼びます。

© 2000-2025, MetaQuotes Ltd.

SignalUnsubscribe

サブスクライブをキャンセルします。

```
bool  SignalUnsubscribe();
```

戻り値

サブスクライブのキャンセルが成功の場合は true、それ以外の場合は false。エラーの詳細を読むには GetLastError() を呼びます。

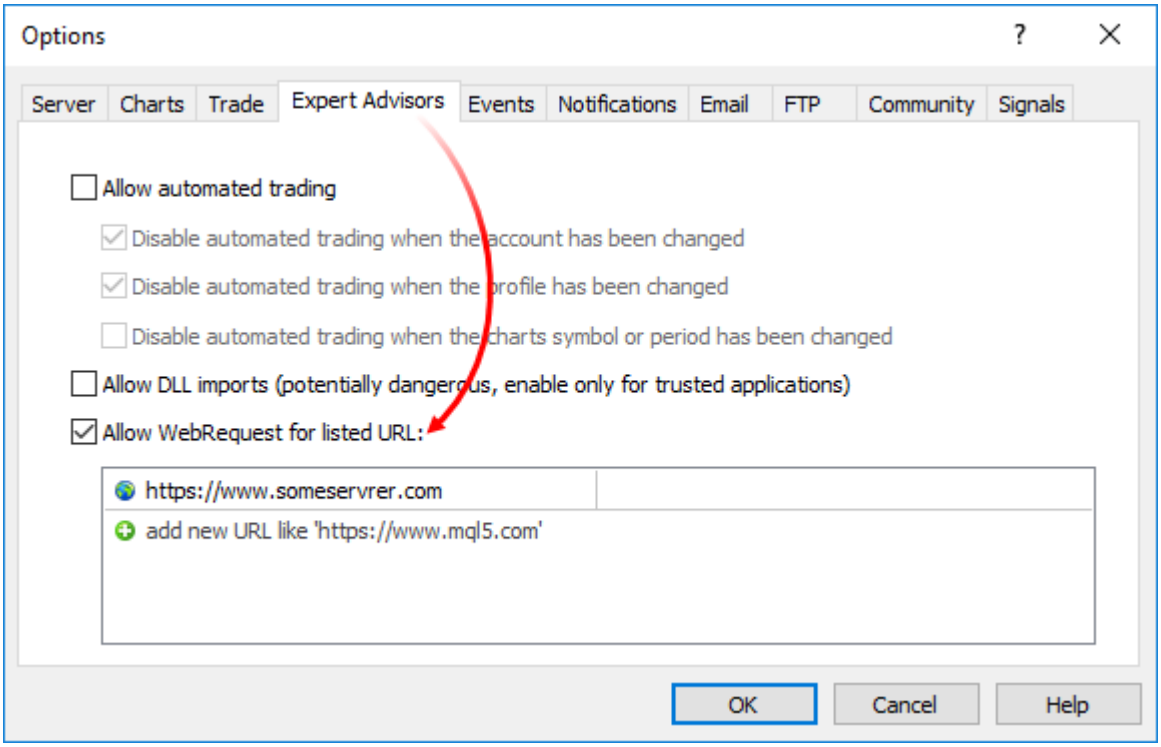
© 2000–2025, MetaQuotes Ltd.

ネットワーク関数

MQL5プログラムはリモートサーバとデータを交換したり、プッシュ通知、メールやデータをFTP経由で送信することができます。

- Socket*関数群により、システムソケット経由でリモートホストとのTCP接続（安全なTLSを含む）を確立することができます。操作の原則は簡単です: ソケットを作成し、サーバに接続し、データの 読み書きを開始します。
- WebRequest関数はWebリソースの使用のために設計されており、HTTPリクエスト（GETやPOSTを含む）を簡単に送信することができます。
- SendFTP、SendMail、SendNotificationはファイル、メール、モバイル通知を送信するより簡潔な関数です。

エンドユーザのセキュリティのために、許可されたIPアドレスのリストはクライアント端末側で実装されています。このリストには、MQL5プログラムがSocket *およびWebRequest関数を介して接続することを許可されているIPアドレスが含まれています。例えば、プログラムがhttps://www.someserver.comに接続する必要がある場合、このアドレスはリスト内の端末ユーザーによって明示的に示されるべきです。プログラムでアドレスを追加することはできません。



追加の設定が必要であることをユーザに通知するためにMQL5プログラムに明示的なメッセージを追加します。これは、#property description、Alert、 Printのいずれかで行うことができます。

関数	アクション
SocketCreate	指定されたフラグを使ってソケットを作成し、そのハンドルを返す
SocketClose	ソケットを閉じる
SocketConnect	タイムアウト管理付きでサーバに接続する
SocketIsConnected	ソケットが現在接続されているかどうかを確認します

© 2000–2025, MetaQuotes Ltd.

関数	アクション
SocketIsReadable	ソケットから読み込めるバイト数を取得する
SocketIsWritable	現在の時点でデータをソケットに書き込めるかどうかを確認する
SocketTimeouts	ソケットシステムオブジェクトのデータを送受信するためのタイムアウトを設定する
SocketRead	ソケットからデータを読む
SocketSend	ソケットにデータを書く
SocketTlsHandshake	TLS Handshakeプロトコルを介して指定されたホストへの安全なTLS（SSL）接続を開始する
SocketTlsCertificate	ネットワーク接続を保護するために使用される証明書に関するデータを取得する
SocketTlsRead	安全なTLS接続からデータを読み込む
SocketTlsReadAvailable	安全なTLS接続からすべてのデータを読み込む
SocketTlsSend	安全なTLS接続を介してデータを送信する
WebRequest	指定されたサーバにHTTP要求を送信する
SendFTP	[FTP]タブで指定されたアドレスにファイルを送信する
SendMail	オプションウィンドウの[メール]タブで指定されたアドレスにファイルを送信する
SendNotification	MetaQuotes ID が[通知]タブで指定されているモバイル端末にプッシュ通知を送信する

© 2000–2025, MetaQuotes Ltd.

SocketCreate

指定されたフラグを使ってソケットを作成し、そのハンドルを返します。

```
int  SocketCreate(
   uint     flags          // フラグ
   );
```

パラメータ

フラグ

[in] ソケットを扱うモードを定義するフラグの組み合わせです。現在サポートされているフラグは、「SOCKET_DEFAULT」1つだけです。

戻り値

ソケット作成が成功した場合はハンドル、それ以外の場合はINVALID_HANDLE

注意事項

未使用のソケットからコンピュータのメモリを解放するにはSocketCloseを呼び出します。

1つのMQL5プログラムから最大128個のソケットを作成できます。この上限が達すると、エラー5271（ERR_NETSOCKET_TOO_MANY_OPENED）が _LastErrorに書かれます。

この関数は独自のスレッド内で実行されるエキスパートアドバイザーやスクリプトのみから呼び出すことが出来ます。指標から呼び出すと、GetLastError()はエラー4014「Function is not allowed for call（関数呼び出しの許可がありません）」を返します。

例：

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Add Address to the list of allowed ones in the terminal
settings to let the example work"
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| サーバにコマンドを送信                                           |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- 安全なTLS接続がポート443を使用する場合
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- 標準TCP接続が使用されている場合
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| サーバ返答を読む                                                 |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- ソケットからデータが存在する間タイムアウトまで読み取る
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- 接続が安全かどうかに応じたさまざまな読み取りコマンド
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- 応答を分析する
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- 応答ヘッダーのみを印刷する
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("HTTP answer header received:");
               Print(StringSubstr(result,0,header_end));
               return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
               }
            }
         }
      }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- ハンドルを確認する
   if(socket!=INVALID_HANDLE)
     {
      //--- すべて正常な場合は接続する
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Established connection to ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- 接続が証明書によって保護されている場合は、そのデータを表示する
         if
(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS certificate:");
            Print("   Owner:  ",subject);
            Print("   Issuer:  ",issuer);
            Print("   Number:     ",serial);
            Print("   Print: ",thumbprint);
            Print("   Expiration: ",expiration);
            ExtTLS=true;
           }
         //--- サーバにGET要求を送信する
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent:
MT5\r\n\r\n"))
           {
            Print("GET request sent");
            //--- 応答を読む
            if(!HTTPRecv(socket,1000))
               Print("Failed to get a response, error ",GetLastError());
           }
         else
            Print("Failed to send GET request, error ",GetLastError());
        }
      else
        {
         Print("Connection to ",Address,":",Port," failed, error ",GetLastError());
        }
      //--- 使用後にソケットを閉じる
      SocketClose(socket);
     }
   else
      Print("Failed to create a socket, error ",GetLastError());
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

SocketClose

ソケットを閉じます。

```
bool  SocketClose(
   const int  socket      // ソケットのハンドル
   );
```

パラメータ

socket

[in] 閉じられるソケットのハンドルです。ハンドルはSocketCreate関数で戻されます。無効なハンドルが渡されると、5270 (ERR_NETSOCKET_INVALIDHANDLE)が_LastErrorに書かれます。

戻り値

成功の場合はtrue、それ以外の場合はfalse

注意事項

ソケットに対してSocketConnectを介した接続が以前に作成されている場合は、停止されます。

この関数は独自のスレッド内で実行されるエキスパートアドバイザーやスクリプトのみから呼び出すことが出来ます。指標から呼び出すと、GetLastError()はエラー4014「Function is not allowed for call(関数呼び出しの許可がありません)」を返します。

例：

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Add Address to the list of allowed ones in the terminal
settings to let the example work"
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| サーバにコマンドを送信                                           |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- 安全なTLS接続がポート443を使用する場合
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- 標準TCP接続が使用されている場合
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| サーバ返答を読む                                                 |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- ソケットからデータが存在する間タイムアウトまで読み取る
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- 接続が安全かどうかに応じたさまざまな読み取りコマンド
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- 応答を分析する
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- 応答ヘッダーのみを印刷する
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("HTTP answer header received:");
               Print(StringSubstr(result,0,header_end));
               return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
               }
            }
         }
      }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- ハンドルを確認する
   if(socket!=INVALID_HANDLE)
     {
      //--- すべて正常な場合は接続する
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Established connection to ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- 接続が証明書によって保護されている場合は、そのデータを表示する
         if
(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS certificate:");
            Print("   Owner:  ",subject);
            Print("   Issuer:  ",issuer);
            Print("   Number:     ",serial);
            Print("   Print: ",thumbprint);
            Print("   Expiration: ",expiration);
            ExtTLS=true;
           }
         //--- サーバにGET要求を送信する
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent:
MT5\r\n\r\n"))
           {
            Print("GET request sent");
            //--- 応答を読む
            if(!HTTPRecv(socket,1000))
               Print("Failed to get a response, error ",GetLastError());
           }
         else
            Print("Failed to send GET request, error ",GetLastError());
        }
      else
        {
         Print("Connection to ",Address,":",Port," failed, error ",GetLastError());
        }
      //--- 使用後にソケットを閉じる
      SocketClose(socket);
     }
   else
      Print("Failed to create a socket, error ",GetLastError());
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

SocketConnect

タイムアウト管理付きでサーバに接続します。

```
bool  SocketConnect(
   int           socket,              // ソケット
   const string  server,              // 接続アドレス
   uint          port,                // 接続ポート
   uint          timeout_receive_ms   // 接続タイムアウト
   );
```

パラメータ

socket

[in] SocketCreate関数で返されるソケットハンドルです。無効なハンドルが渡されると、5270 (ERR_NETSOCKET_INVALIDHANDLE)が_LastErrorに書かれます。

server

[in] 接続するサーバのドメイン名またはIPアドレス

port

[in] 接続ポート番号

timeout_receive_ms

[in] ミリ秒単位の接続タイムアウトです。その時間内に接続が確立されないと、試行は中止されます。

戻り値

接続が成功の場合はtrue、それ以外の場合はfalse

注意事項

接続アドレスは、クライアント端末側(ツール、オプション、エキスパートアドバイザー)で許可されているアドレスのリストに追加する必要があります。

接続が失敗すると、5272 (ERR_NETSOCKET_CANNOT_CONNECT)が_LastErrorに書かれます。

この関数は独自のスレッド内で実行されるエキスパートアドバイザーやスクリプトのみから呼び出すことが出来ます。指標から呼び出すと、GetLastError()はエラー4014「Function is not allowed for call(関数呼び出しの許可がありません)」を返します。

例:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Add Address to the list of allowed ones in the terminal
settings to let the example work"
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| サーバにコマンドを送信                                           |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- 安全なTLS接続がポート443を使用する場合
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- 標準TCP接続が使用されている場合
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| サーバ返答を読む                                                 |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- ソケットからデータが存在する間タイムアウトまで読み取る
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- 接続が安全かどうかに応じたさまざまな読み取りコマンド
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- 応答を分析する
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- 応答ヘッダーのみを印刷する
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("HTTP answer header received:");
               Print(StringSubstr(result,0,header_end));
               return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
               }
            }
         }
      }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- ハンドルを確認する
   if(socket!=INVALID_HANDLE)
     {
      //--- すべて正常な場合は接続する
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Established connection to ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- 接続が証明書によって保護されている場合は、そのデータを表示する
         if
(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS certificate:");
            Print("   Owner:  ",subject);
            Print("   Issuer:  ",issuer);
            Print("   Number:     ",serial);
            Print("   Print: ",thumbprint);
            Print("   Expiration: ",expiration);
            ExtTLS=true;
           }
         //--- サーバにGET要求を送信する
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent:
MT5\r\n\r\n"))
           {
            Print("GET request sent");
            //--- 応答を読む
            if(!HTTPRecv(socket,1000))
               Print("Failed to get a response, error ",GetLastError());
           }
         else
            Print("Failed to send GET request, error ",GetLastError());
        }
      else
        {
         Print("Connection to ",Address,":",Port," failed, error ",GetLastError());
        }
      //--- 使用後にソケットを閉じる
      SocketClose(socket);
     }
   else
      Print("Failed to create a socket, error ",GetLastError());
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

SocketIsConnected

ソケットが現在接続されているかどうかを確認します。

```
bool  SocketIsConnected(
   const int  socket      // ソケットのハンドル
   );
```

パラメータ

socket

[in] SocketCreate()関数で返されるソケットハンドルです。 _LastErrorに無効なハンドルが渡されると、5270（ERR_NETSOCKET_INVALIDHANDLE）がアクティブになります。

戻り値

ソケットが接続された場合はtrue、それ以外の場合はfalse

注意事項

SocketIsConnected()関数によって、現在のソケット接続状態を確認することができます。

この関数は独自のスレッド内で実行されるエキスパートアドバイザーやスクリプトのみから呼び出すことが出来ます。指標から呼び出すと、GetLastError()はエラー4014「Function is not allowed for call（関数呼び出しの許可がありません）」を返します。

例：

```
//+------------------------------------------------------------------+
//|                                            SocketIsConnected.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd.   |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"
#property description "Add Address to the list of allowed ones in the terminal
settings to let the example work"
#property script_show_inputs

input string Address    ="www.mql5.com";
input int    Port       =80;
input bool   CloseSocket=true;
bool         ExtTLS     =false;
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart(void)
  {
//--- ソケットを作成し、そのハンドルを取得する
   int socket=SocketCreate();
//--- ハンドルを確認する
```

© 2000–2025, MetaQuotes Ltd.

```
   if(socket!=INVALID_HANDLE)
     {
      //--- 成功したら接続する
      if(SocketConnect(socket,Address,Port,1000))
        {
         PrintFormat("Established connection to %s:%d",Address,Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- 接続が証明書で保護されている場合は、そのデータを表示する
         if
(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS certificate:");
            Print("   Owner:       ",subject);
            Print("   Issuer:      ",issuer);
            Print("   Number:      ",serial);
            Print("   Print:       ",thumbprint);
            Print("   Expiration: ",expiration);
            ExtTLS=true;
           }
         //--- サーバにGETリクエストを送信する
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent:
MT5\r\n\r\n"))
           {
            Print("GET request sent");
            //--- 応答を読む
            if(!HTTPRecv(socket,1000))
               Print("Failed to get a response, error ",GetLastError());
           }
         else
            Print("Failed to send GET request, error ",GetLastError());
        }
      else
        {
         PrintFormat("Connection to %s:%d failed, error %
d",Address,Port,GetLastError());
        }
      //--- フラグが設定されている場合は、使用後にソケットを閉じる
      if(CloseSocket)
         SocketClose(socket);
     }
   else
      Print("Failed to create a socket, error ",GetLastError());

//--- サーバ接続を確認する
   bool connected=SocketIsConnected(socket);
//--- 接続がない場合には操作を終了する
   if(!connected)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print("No connection to server");
      }
//--- サーバ接続がある場合
   else
      {
       Print("Connection to the server is available\nThe connection needs to be
closed. Closing");
       //--- ソケットを閉じて接続状態を再確認する
       SocketClose(socket);
       connected=SocketIsConnected(socket);
      }

//--- 現在のサーバ接続ステータスを表示する
   Print("Currently connected: ",(connected ? "opened" : "closed"));
   /*
   result in case CloseSocket = true:
   No connection to server
   Currently connected: closed

   result in case CloseSocket = false:
   Connection to the server is available
   The connection needs to be closed. Closing
   Currently connected: closed
   */
  }
//+------------------------------------------------------------------+
//|  サーバにコマンドを送信する                                      |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
//--- 文字列を文字配列に変換し、終端のゼロを破棄する
   char req[];
   int  len=StringToCharArray(request,req)-1;

   if(len<0)
      return(false);
//--- ポート443経由の安全なTLS接続が使用されている場合
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- 通常のTCP接続が使用される場合
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//|  サーバの応答を読む                                              |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout_ms)
  {
   char   rsp[];
```

© 2000-2025, MetaQuotes Ltd.

```
   string result;
   ulong  timeout_check=GetTickCount64()+timeout_ms;
//--- ソケットからデータを読み出すが、タイムアウトを超えない範囲でデータを読み出す
   do
     {
      uint len=SocketIsReadable(socket);

      if(len)
        {
         int rsp_len;
         //--- 接続が安全かどうかに応じて異なる読み取りコマンド
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout_ms);
         //--- 応答を解析する
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- 応答ヘッダーのみを表示する
            int header_end=StringFind(result,"\r\n\r\n");

            if(header_end>0)
              {
               Print("HTTP answer header received:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
            //--- 読み取りタイムアウトの有効期限を更新する
            timeout_check=GetTickCount64()+timeout_ms;
           }
        }
     }
   while(GetTickCount64()<timeout_check && !IsStopped());

   return(false);
  }
```

参照

SocketConnect、SocketIsWritable、SocketCreate、SocketClose

© 2000-2025, MetaQuotes Ltd.

SocketIsReadable

ソケットから読み込めるバイト数を取得します。

```
uint  SocketIsReadable(
   const int  socket      // ソケットのハンドル
   );
```

パラメータ

socket

[in] SocketCreate関数で返されるソケットハンドルです。 _LastErrorに無効なハンドルが渡されると、5270（ERR_NETSOCKET_INVALIDHANDLE）がアクティブになります。

戻り値

計算できるバイト数（エラーの場合は0）

注意事項

関数の実行時にシステムソケットでエラーが発生した場合、SocketConnectによって確立された接続は中断されます。

SocketReadを呼び出す前に、ソケットが読み取り用のデータを備えているかどうかを確認してください。さもないと、SocketRead関数はデータがない場合にtimeout_ms間データを待つので、プログラムの実行が遅れます。

この関数は独自のスレッド内で実行されるエキスパートアドバイザーやスクリプトのみから呼び出すことが出来ます。指標から呼び出すと、GetLastError()はエラー4014「Function is not allowed for call（関数呼び出しの許可がありません）」を返します。

例：

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Add Address to the list of allowed ones in the terminal
settings to let the example work"
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| サーバにコマンドを送信                                           |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- 安全なTLS接続がポート443を使用する場合
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- 標準TCP接続が使用されている場合
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| サーバ返答を読む                                                 |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- ソケットからデータが存在する間タイムアウトまで読み取る
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- 接続が安全かどうかに応じたさまざまな読み取りコマンド
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- 応答を分析する
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- 応答ヘッダーのみを印刷する
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("HTTP answer header received:");
               Print(StringSubstr(result,0,header_end));
               return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
               }
            }
         }
      }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- ハンドルを確認する
   if(socket!=INVALID_HANDLE)
     {
      //--- すべて正常な場合は接続する
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Established connection to ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- 接続が証明書によって保護されている場合は、そのデータを表示する
         if
(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS certificate:");
            Print("   Owner:  ",subject);
            Print("   Issuer:  ",issuer);
            Print("   Number:     ",serial);
            Print("   Print: ",thumbprint);
            Print("   Expiration: ",expiration);
            ExtTLS=true;
           }
         //--- サーバにGET要求を送信する
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent:
MT5\r\n\r\n"))
           {
            Print("GET request sent");
            //--- 応答を読む
            if(!HTTPRecv(socket,1000))
               Print("Failed to get a response, error ",GetLastError());
           }
         else
            Print("Failed to send GET request, error ",GetLastError());
        }
      else
        {
         Print("Connection to ",Address,":",Port," failed, error ",GetLastError());
        }
      //--- 使用後にソケットを閉じる
      SocketClose(socket);
     }
   else
      Print("Failed to create a socket, error ",GetLastError());
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

SocketIsWritable

現在の時点でデータをソケットに書き込めるかどうかを確認します。

```
bool  SocketIsWritable(
   const int  socket      // ソケットのハンドル
   );
```

パラメータ

socket

[in] SocketCreate関数で返されるソケットハンドルです。無効なハンドルが渡されると、5270（ERR_NETSOCKET_INVALIDHANDLE）が_LastErrorに書かれます。

戻り値

書き込みが可能の場合はtrue、それ以外の場合はfalse

注意事項

この関数を使用すると、ソケットにデータを書き込めるかどうかを確認できます。

関数の実行時にシステムソケットでエラーが発生した場合、SocketConnectによって確立された接続は中断されます。

この関数は独自のスレッド内で実行されるエキスパートアドバイザーやスクリプトのみから呼び出すことが出来ます。指標から呼び出すと、GetLastError()はエラー4014「Function is not allowed for call（関数呼び出しの許可がありません）」を返します。

例：

```
//+------------------------------------------------------------------+
//|                                              SocketIsWritable.mq5 |
//|                                  Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"
#property description "Add Address to the list of allowed ones in the terminal
settings to let the example work"
#property script_show_inputs

input string Address    ="www.mql5.com";
input int    Port       =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart(void)
  {
//--- ソケットを作成し、そのハンドルを取得する
   int socket=SocketCreate();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- ハンドルを確認する
   if(socket!=INVALID_HANDLE)
     {
      //--- 成功したら接続する
      if(SocketConnect(socket,Address,Port,1000))
        {
         PrintFormat("Established connection to %s:%d",Address,Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- 接続が証明書で保護されている場合は、そのデータを表示する
         if
(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS certificate:");
            Print("   Owner:       ",subject);
            Print("   Issuer:      ",issuer);
            Print("   Number:      ",serial);
            Print("   Print:       ",thumbprint);
            Print("   Expiration: ",expiration);
            ExtTLS=true;
           }
         //--- サーバにGETリクエストを送信する
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent:
MT5\r\n\r\n"))
           {
            Print("GET request sent");
            //--- 応答を読む
            if(!HTTPRecv(socket,1000))
               Print("Failed to get a response, error ",GetLastError());
           }
         else
            Print("Failed to send GET request, error ",GetLastError());
        }
      else
        {
         PrintFormat("Connection to %s:%d failed, error %
d",Address,Port,GetLastError());
        }
      //--- 操作ログの現在の時点でソケットにデータを書き込む機能を表示する
      if(SocketIsWritable(socket))
         Print("At the current moment in time, writing data to the socket is
possible");
      else
         Print("It is not possible to write data to the socket at the current
time");
      //--- 使用後はソケットを閉じる
      if(SocketClose(socket))
         Print("Now the socket is closed");
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   else
      Print("Failed to create a socket, error ",GetLastError());
   /*
  結果:
   At the current moment in time, writing data to the socket is possible
   Socket is closed now
   */
  }
//+------------------------------------------------------------------+
//|  サーバにコマンドを送信する                                      |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
//--- 文字列を文字配列に変換し、終端のゼロを破棄する
   char req[];
   int  len=StringToCharArray(request,req)-1;

   if(len<0)
      return(false);
//--- ポート443経由の安全なTLS接続が使用されている場合
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- 通常のTCP接続が使用される場合
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//|  サーバの応答を読む                                              |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout_ms)
  {
   char   rsp[];
   string result;
   ulong  timeout_check=GetTickCount64()+timeout_ms;
//--- ソケットからデータを読み出すが、タイムアウトを超えない範囲でデータを読み出す
   do
     {
      uint len=SocketIsReadable(socket);

      if(len)
        {
         int rsp_len;
         //--- 接続が安全かどうかに応じて異なる読み取りコマンド
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout_ms);
         //--- 応答を解析する
         if(rsp_len>0)
           {
```

© 2000-2025, MetaQuotes Ltd.

```
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- 応答ヘッダーのみを表示する
            int header_end=StringFind(result,"\r\n\r\n");

            if(header_end>0)
              {
               Print("HTTP answer header received:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
              //--- 読み取りタイムアウトの有効期限を更新する
            timeout_check=GetTickCount64()+timeout_ms;
           }
        }
     }
  while(GetTickCount64()<timeout_check && !IsStopped());

  return(false);
 }
```

© 2000–2025, MetaQuotes Ltd.

SocketTimeouts

ソケットシステムオブジェクトのデータを送受信するためのタイムアウトを設定します。

```
bool  SocketTimeouts(
   int            socket,               // ソケット
   uint           timeout_send_ms,      // データ送信のタイムアウト
   uint           timeout_receive_ms    // データ受信のタイムアウト
   );
```

パラメータ

socket

[in] SocketCreate関数で返されるソケットハンドルです。無効なハンドルが渡されると、5270（ERR_NETSOCKET_INVALIDHANDLE）が_LastErrorに書かれます。

timeout_send_ms

[in] ミリ秒単位のデータ送信タイムアウト

timeout_receive_ms

[in] ミリ秒単位のデータ受信タイムアウト<.

戻り値

成功の場合はtrue、それ以外の場合はfalse

注意事項

システムオブジェクトのタイムアウトをSocketReadで設定されたデータ受信タイムアウトと混乱しないでください。SocketTimeoutはオペレーティングシステムのソケットオブジェクトに対して1回タイムアウトを設定します。これらのタイムアウトは、このソケットを介してデータを読み書きするためのすべての関数に適用されます。SocketReadでは、タイムアウトは特定のデータ読み取り操作に対して設定されます。

この関数は独自のスレッド内で実行されるエキスパートアドバイザーやスクリプトのみから呼び出すことが出来ます。指標から呼び出すと、GetLastError()はエラー4014「Function is not allowed for call（関数呼び出しの許可がありません）」を返します。

例：

```
//+------------------------------------------------------------------+
//|                                                SocketTimeouts.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd.   |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"

#define   TIMEOUT_SEND     5000
#define   TIMEOUT_RECEIVE  5000
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnStart(void)
  {
//--- ソケットを作成し、そのハンドルを取得する
   int socket=SocketCreate();

   if(socket==INVALID_HANDLE)
     {
      Print("SocketCreate() failed. Error ",GetLastError());
      return;
     }
//--- ソケットシステムオブジェクトのデータの受信と送信のタイムアウトを設定する
   if(SocketTimeouts(socket,TIMEOUT_SEND,TIMEOUT_RECEIVE))
      PrintFormat("timeouts were successfully
set",socket,TIMEOUT_SEND,TIMEOUT_RECEIVE);
   else
      PrintFormat("SocketTimeouts(%d, %I64d, %I64d) failed. Error %
d",socket,TIMEOUT_SEND,TIMEOUT_RECEIVE,GetLastError());
//--- ソケットを閉じる
   SocketClose(socket);
  }
```

© 2000–2025, MetaQuotes Ltd.

SocketRead

ソケットからデータを読みます。

```
int  SocketRead(
   int           socket,              // ソケット
   uchar&        buffer[],            // ソケットからデータを読むためのバッファ
   uint          buffer_maxlen,       // 読まれるバイト数
   uint          timeout_ms           // 読み込みのタイムアウト
   );
```

パラメータ

socket

[in] SocketCreate関数で返されるソケットハンドルです。 _LastErrorに無効なハンドルが渡されると、5270 (ERR_NETSOCKET_INVALIDHANDLE)がアクティブになります。

buffer

[out] データが読まれるuchar型配列への参照。動的配列サイズは、読み取られたバイト数だけ増えます。配列サイズはINT_MAX (2147483647)を超えることはできません。

buffer_maxlen

[in] *buffer[]*配列に読まれるバイト数。配列に収まらないデータはソケットに残り、次のSocketReadの呼び出しで受信できます。*buffer_maxlen*はINT_MAX (2147483647)を超えることはできません。

timeout_ms

[in] ミリ秒単位の読み込みタイムアウトです。この時間内にデータが取得されない場合、試行は中止され、関数は-1を返します。

戻り値

成功の場合、読まれたバイト数。エラーの場合は -1 。

注意事項

関数の実行時にシステムソケットでエラーが発生した場合、SocketConnectによって確立された接続は中断されます。

データ読み込みエラーの場合は、エラー5273 (ERR_NETSOCKET_IO_ERROR)が _LastErrorに書かれます。

この関数は独自のスレッド内で実行されるエキスパートアドバイザーやスクリプトのみから呼び出すことが出来ます。指標から呼び出すと、GetLastError()はエラー4014「Function is not allowed for call(関数呼び出しの許可がありません)」を返します。

例：

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Add Address to the list of allowed ones in the terminal
settings to let the example work"
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| サーバにコマンドを送信                                           |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- 安全なTLS接続がポート443を使用する場合
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- 標準TCP接続が使用されている場合
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| サーバ返答を読む                                                 |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- ソケットからデータが存在する間タイムアウトまで読み取る
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- 接続が安全かどうかに応じたさまざまな読み取りコマンド
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- 応答を分析する
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- 応答ヘッダーのみを印刷する
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("HTTP answer header received:");
               Print(StringSubstr(result,0,header_end));
               return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
               }
            }
         }
      }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- ハンドルを確認する
   if(socket!=INVALID_HANDLE)
     {
      //--- すべて正常な場合は接続する
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Established connection to ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- 接続が証明書によって保護されている場合は、そのデータを表示する
         if
(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS certificate:");
            Print("   Owner:  ",subject);
            Print("   Issuer:  ",issuer);
            Print("   Number:     ",serial);
            Print("   Print: ",thumbprint);
            Print("   Expiration: ",expiration);
            ExtTLS=true;
           }
         //--- サーバにGET要求を送信する
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent:
MT5\r\n\r\n"))
           {
            Print("GET request sent");
            //--- 応答を読む
            if(!HTTPRecv(socket,1000))
               Print("Failed to get a response, error ",GetLastError());
           }
         else
            Print("Failed to send GET request, error ",GetLastError());
        }
      else
        {
         Print("Connection to ",Address,":",Port," failed, error ",GetLastError());
        }
      //--- 使用後にソケットを閉じる
      SocketClose(socket);
     }
   else
      Print("Failed to create a socket, error ",GetLastError());
  }
//+------------------------------------------------------------------+
```

参照

© 2000-2025, MetaQuotes Ltd.

SocketTimeouts、MathSwap

© 2000–2025, MetaQuotes Ltd.

SocketSend

ソケットにデータを書きます。

```
int  SocketSend(
   int            socket,              // ソケット
   const uchar&   buffer[],            // データバッファ
   uint           buffer_len           // バッファサイズ
   );
```

パラメータ

socket

[in] SocketCreate関数で返されるソケットハンドルです。無効なハンドルが渡されると、5270 (ERR_NETSOCKET_INVALIDHANDLE)が_LastErrorに書かれます。

buffer

[in] データがソケットに送信されるuchar型配列への参照

buffer_len

[in]「buffer」配列サイズ

戻り値

成功の場合、ソケットに書かれたバイト数。エラーの場合は -1 。

注意事項

関数の実行時にシステムソケットでエラーが発生した場合、SocketConnectによって確立された接続は中断されます。

データ書き込みエラーの場合は、エラー5273 (ERR_NETSOCKET_IO_ERROR)が _LastErrorに書かれます。

この関数は独自のスレッド内で実行されるエキスパートアドバイザーやスクリプトのみから呼び出すことが出来ます。指標から呼び出すと、GetLastError()はエラー4014「Function is not allowed for call(関数呼び出しの許可がありません)」を返します。

例：

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Add Address to the list of allowed ones in the terminal
settings to let the example work"
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| サーバにコマンドを送信                                           |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- 安全なTLS接続がポート443を使用する場合
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- 標準TCP接続が使用されている場合
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| サーバ返答を読む                                                 |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- ソケットからデータが存在する間タイムアウトまで読み取る
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- 接続が安全かどうかに応じたさまざまな読み取りコマンド
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- 応答を分析する
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- 応答ヘッダーのみを印刷する
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("HTTP answer header received:");
               Print(StringSubstr(result,0,header_end));
               return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
               }
            }
         }
      }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- ハンドルを確認する
   if(socket!=INVALID_HANDLE)
     {
      //--- すべて正常な場合は接続する
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Established connection to ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- 接続が証明書によって保護されている場合は、そのデータを表示する
         if
(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS certificate:");
            Print("   Owner:  ",subject);
            Print("   Issuer:  ",issuer);
            Print("   Number:     ",serial);
            Print("   Print: ",thumbprint);
            Print("   Expiration: ",expiration);
            ExtTLS=true;
           }
         //--- サーバにGET要求を送信する
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent:
MT5\r\n\r\n"))
           {
            Print("GET request sent");
            //--- 応答を読む
            if(!HTTPRecv(socket,1000))
               Print("Failed to get a response, error ",GetLastError());
           }
         else
            Print("Failed to send GET request, error ",GetLastError());
        }
      else
        {
         Print("Connection to ",Address,":",Port," failed, error ",GetLastError());
        }
      //--- 使用後にソケットを閉じる
      SocketClose(socket);
     }
   else
      Print("Failed to create a socket, error ",GetLastError());
  }
//+------------------------------------------------------------------+
```

参照

© 2000-2025, MetaQuotes Ltd.

SocketTimeouts、MathSwap、StringToCharArray

© 2000–2025, MetaQuotes Ltd.

SocketTlsHandshake

TLS Handshakeプロトコルを介して指定されたホストへの安全なTLS(SSL)接続を開始します。ハンドシェイク中、クライアントとサーバは接続パラメータ、つまり適用されるプロトコルのバージョンとデータの暗号化方式について合意します。

```
bool  SocketTlsHandshake(
   int           socket,               // ソケット
   const string  host                  // ホストアドレス
   );
```

パラメータ

socket

[in] SocketCreate関数で返されるソケットハンドルです。無効なハンドルが渡されると、5270(ERR_NETSOCKET_INVALIDHANDLE)が_LastErrorに書かれます。

host

[in] 安全な接続が確立されているホストのアドレス

戻り値

成功の場合はtrue、それ以外の場合はfalse

注意事項

安全な接続の前に、プログラムはSocketConnectを使用してホストとの標準TCP接続を確立する必要があります。

安全な接続が失敗すると、エラー5274(ERR_NETSOCKET_HANDSHAKE_FAILED)が_LastErrorに書かれます。

ポート443に接続する場合はこの関数を呼び出す必要はありません。これは、安全なTLS(SSL)接続に使用される標準のTCPポートです。

この関数は独自のスレッド内で実行されるエキスパートアドバイザーやスクリプトのみから呼び出すことが出来ます。指標から呼び出すと、GetLastError()はエラー4014「Function is not allowed for call(関数呼び出しの許可がありません)」を返します。

例:

```
//+------------------------------------------------------------------+
//|                                            SocketTlsHandshake.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd.   |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"

#define   SERVER    "smtp.gmail.com"
#define   PORT      587
//+------------------------------------------------------------------+
//|  安全な接続に手動で切り替える                                    |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool TlsHandshake(int socket)
  {
//--- サーバの挨拶を取得する
   string rsp;

   if(!RecvString(socket,rsp))
      return(false);
//--- サーバに挨拶する
   if(!SendString(socket,"EHLO my.domain.com\r\n"))
      return(false);
//--- サポートされているコマンドのリストを含むサーバ応答を取得する
   if(!RecvString(socket,rsp))
      return(false);
//--- 挨拶をプリントする
   Print("SERVER: ",rsp);
//--- 安全でない接続から TLS を使用して安全な接続に切り替えることをサーバーに通知する
   if(!SendString(socket,"STARTTLS\r\n"))
      return(false);
//--- サーバの応答を取得する
   if(!RecvString(socket,rsp))
      return(false);
//--- この例では、TLSへの切り替えの準備状況に関するサーバの応答(「Ready to start TLS」)は確認しない

//--- TLS Handshakeプロトコルを介して指定されたホストへの安全なTLS (SSL)接続を開始する
   if(SocketTlsHandshake(socket,InpTimeout))
      return(true);

   Print("SocketTlsHandshake() failed. Error ",GetLastError());
   return(false);
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart(void)
  {
//--- ソケットを作成し、そのハンドルを取得する
   int socket=SocketCreate();

   if(socket==INVALID_HANDLE)
     {
      Print("SocketCreate() failed. Error ",GetLastError());
      return;
     }
//--- ポート経由でサーバーに接続する
   if(!SocketConnect(socket,SERVER,PORT,10000))
     {
      Print("SocketConnect() failed. Error ",GetLastError());
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   else
     {
      //--- 安全でない接続が確立された
      PrintFormat("%s connection has been established to %s:%d",(PORT==443 ? "A
secured" : "An unsecured"),SERVER,PORT);
      //--- 安全な接続に切り替える
      if(PORT!=443 && TlsHandshake(socket))
        {
         PrintFormat("Unsecured connection to %s:%d switched to
secured",SERVER,PORT);
         //--- 接続が証明書で保護されている場合は、そのデータを表示する
         string   subject,issuer,serial,thumbprint;
         datetime expiration;

         if
(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS certificate:");
            Print("   Owner:       ",subject);
            Print("   Issuer:      ",issuer);
            Print("   Number:      ",serial);
            Print("   Print:       ",thumbprint);
            Print("   Expiration: ",expiration);
           }
        }
     }
//--- 使用後はソケットを閉じる</t1>
   SocketClose(socket);
   /*
   結果:
   An unsecured connection has been established to smtp.gmail.com:587
   SERVER: 220 smtp.gmail.com ESMTP a640c23a62f3a-a9b1f298319sm82305866b.105 - gsmtp

   SERVER: 250-smtp.gmail.com at your service, [37.193.40.122]
   250-SIZE 35882577
   250-8BITMIME
   250-STARTTLS
   250-ENHANCEDSTATUSCODES
   250-PIPELINING
   250-CHUNKING
   250 SMTPUTF8

   SERVER: 220 2.0.0 Ready to start TLS

   SocketTlsHandshake(): A secure connection to smtp.gmail.com:587 is now
established
   TLS certificate:
      Owner:  /CN=smtp.gmail.com
      Issuer:  /C=US/O=Google Trust Services/CN=WR2
```

© 2000–2025, MetaQuotes Ltd.

```
      Number:     1f:f4:db:2a:5a:e6:dc:52:0a:4c:05:ce:81:cc:c3:f7
      Print: d6be8af229b5329cd3d4c2789c02aa94f89b421c
      Expiration: 2024.12.30 08:25:30
   */
  }
//+------------------------------------------------------------------+
//| サーバに文字列を送信する                                         |
//+------------------------------------------------------------------+
bool SendString(int socket,const string str)
  {
//--- 文字列を文字配列に変換する
   uchar data[];
   int   size=StringToCharArray(str,data,0,str.Length(),CP_UTF8);
//--- ソケットにデータを送信する
   ResetLastError();
   if(SocketSend(socket,data,size)==size)
      return(true);
//--- データ送信エラー
   Print("Failed to send data to server. Error ",GetLastError());
   return false;
  }
//+------------------------------------------------------------------+
//| サーバから文字列を取得する                                       |
//+------------------------------------------------------------------+
bool RecvString(int socket,string& result,uint timeout_ms=1000)
  {
//--- ソケットにデータが現れるのを待機する
   ulong wait_time_end=GetMicrosecondCount()+timeout_ms*1000;

   while(!SocketIsReadable(socket))
     {
      Sleep(10);
      //--- データの待機中にタイムアウトしたため、応答としてNULLを返す
      if(wait_time_end<GetMicrosecondCount())
        {
         Print("ERROR: No response from server");
         return(false);
        }
     }
//--- ソケットからデータを読む
   uchar  data[128];
   uint   size=0;
   string resp=NULL;

   do
     {
      uchar b[1];
      int   n=SocketRead(socket,b,1,1000);
```

© 2000-2025, MetaQuotes Ltd.

```
      if(n < 0)
         break;

      if(n)
        {
         data[size++]=b[0];

         if(size==data.Size())
           {
            resp += CharArrayToString(data,0,data.Size(),CP_UTF8);
            size = 0;
           }
        }
     }
   while(SocketIsReadable(socket));
//--- 読み取ったデータを文字列にコピーする
   if(size)
      resp+=CharArrayToString(data,0,size,CP_UTF8);
//--- 文字列が空の場合はエラー
   if(!resp.Length())
     {
      Print("ERROR: No response from server");
      return(false);
     }
//--- 文字列を返す
   result=resp;
   return(true);
  }
```

© 2000–2025, MetaQuotes Ltd.

SocketTlsCertificate

ネットワーク接続を保護するために使用される証明書に関するデータを取得します。

```
bool  SocketTlsCertificate(
   int           socket,              // ソケット
   string&       subject,             // 証明書の所有者
   string&       issuer,              // 証明書の発行者
   string&       serial,              // 証明書のシリアル番号
   string&       thumbprint,          // 証明書プリント
   datetime&     expiration           // 証明書期限
   );
```

パラメータ

socket

[in] SocketCreate関数で返されるソケットハンドルです。無効なハンドルが渡されると、5270（ERR_NETSOCKET_INVALIDHANDLE）が_LastErrorに書かれます。

subject

[in] 証明書所有者名。Subjectフィールドに対応します。

issuer

[in] 証明書所有者名。Issuerフィールドに対応します。

serial

[in] 証明書のシリアル番号。SerialNumberフィールドに対応します。

thumbprint

[in] 証明書プリント。証明書ファイル全体（発行者の署名を含むすべてのフィールド）のSHA-1ハッシュに対応します。

expiration

[in] datetime形式での証明書の有効期限。

戻り値

成功の場合はtrue、それ以外の場合はfalse

注意事項

証明書データはSocketTlsHandshakeを使用して安全な接続を確立した後にのみ要求できます。

証明書取得エラーの場合、エラー5275（ERR_NETSOCKET_NO_CERTIFICATE）が_LastErrorに書かれます。

この関数は独自のスレッド内で実行されるエキスパートアドバイザーやスクリプトのみから呼び出すことが出来ます。指標から呼び出すと、GetLastError()はエラー4014「Function is not allowed for call（関数呼び出しの許可がありません）」を返します。

例：

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Add Address to the list of allowed ones in the terminal
settings to let the example work"
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| サーバにコマンドを送信                                           |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- 安全なTLS接続がポート443を使用する場合
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- 標準TCP接続が使用されている場合
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| サーバ返答を読む                                                 |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- ソケットからデータが存在する間タイムアウトまで読み取る
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- 接続が安全かどうかに応じたさまざまな読み取りコマンド
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- 応答を分析する
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- 応答ヘッダーのみを印刷する
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("HTTP answer header received:");
               Print(StringSubstr(result,0,header_end));
               return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
               }
            }
         }
      }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- ハンドルを確認する
   if(socket!=INVALID_HANDLE)
     {
      //--- すべて正常な場合は接続する
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Established connection to ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- 接続が証明書によって保護されている場合は、そのデータを表示する
         if
(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS certificate:");
            Print("   Owner:  ",subject);
            Print("   Issuer:  ",issuer);
            Print("   Number:     ",serial);
            Print("   Print: ",thumbprint);
            Print("   Expiration: ",expiration);
            ExtTLS=true;
           }
         //--- サーバにGET要求を送信する
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent:
MT5\r\n\r\n"))
           {
            Print("GET request sent");
            //--- 応答を読む
            if(!HTTPRecv(socket,1000))
               Print("Failed to get a response, error ",GetLastError());
           }
         else
            Print("Failed to send GET request, error ",GetLastError());
        }
      else
        {
         Print("Connection to ",Address,":",Port," failed, error ",GetLastError());
        }
      //--- 使用後にソケットを閉じる
      SocketClose(socket);
     }
   else
      Print("Failed to create a socket, error ",GetLastError());
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

SocketTlsRead

安全なTLS接続からデータを読み込みます。

```
int  SocketTlsRead(
   int           socket,              // ソケット
   uchar&        buffer[],            // ソケットからデータを読むためのバッファ
   uint          buffer_maxlen        // 読まれるバイト数
   );
```

パラメータ

socket

[in] SocketCreate関数で返されるソケットハンドルです。 _LastErrorに無効なハンドルが渡されると、5270 (ERR_NETSOCKET_INVALIDHANDLE)がアクティブになります。

buffer

[out] データが読まれるuchar型配列への参照。動的配列サイズは、読み取られたバイト数だけ増えます。配列サイズはINT_MAX (2147483647)を超えることはできません。

buffer_maxlen

[in] *buffer[]*配列に読まれるバイト数。配列に収まらないデータはソケットに残り、次のSocketTLSReadの呼び出しで受信できます。*buffer_maxlen*はINT_MAX (2147483647)を超えることはできません。

戻り値

成功の場合、読まれたバイト数。エラーの場合は -1 。

注意事項

関数の実行時にシステムソケットでエラーが発生した場合、SocketConnectによって確立された接続は中断されます。

この関数は、指定された量のデータを受信するかタイムアウトに達するまで実行されます (SocketTimeouts)。

データ読み込みエラーの場合は、エラー5273 (ERR_NETSOCKET_IO_ERROR)が _LastErrorに書かれます。

この関数は独自のスレッド内で実行されるエキスパートアドバイザーやスクリプトのみから呼び出すことが出来ます。指標から呼び出すと、GetLastError()はエラー4014「Function is not allowed for call(関数呼び出しの許可がありません)」を返します。

例：

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Add Address to the list of allowed ones in the terminal
settings to let the example work"
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| サーバにコマンドを送信                                           |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- 安全なTLS接続がポート443を使用する場合
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- 標準TCP接続が使用されている場合
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| サーバ返答を読む                                                 |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- ソケットからデータが存在する間タイムアウトまで読み取る
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- 接続が安全かどうかに応じたさまざまな読み取りコマンド
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- 応答を分析する
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- 応答ヘッダーのみを印刷する
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("HTTP answer header received:");
               Print(StringSubstr(result,0,header_end));
               return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
               }
            }
         }
      }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- ハンドルを確認する
   if(socket!=INVALID_HANDLE)
     {
      //--- すべて正常な場合は接続する
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Established connection to ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- 接続が証明書によって保護されている場合は、そのデータを表示する
         if
(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS certificate:");
            Print("   Owner:  ",subject);
            Print("   Issuer:  ",issuer);
            Print("   Number:     ",serial);
            Print("   Print: ",thumbprint);
            Print("   Expiration: ",expiration);
            ExtTLS=true;
           }
         //--- サーバにGET要求を送信する
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent:
MT5\r\n\r\n"))
           {
            Print("GET request sent");
            //--- 応答を読む
            if(!HTTPRecv(socket,1000))
               Print("Failed to get a response, error ",GetLastError());
           }
         else
            Print("Failed to send GET request, error ",GetLastError());
        }
      else
        {
         Print("Connection to ",Address,":",Port," failed, error ",GetLastError());
        }
      //--- 使用後にソケットを閉じる
      SocketClose(socket);
     }
   else
      Print("Failed to create a socket, error ",GetLastError());
  }
//+------------------------------------------------------------------+
```

参照

© 2000-2025, MetaQuotes Ltd.

SocketTimeouts、MathSwap

© 2000–2025, MetaQuotes Ltd.

SocketTlsReadAvailable

安全なTLS接続からすべてのデータを読み込みます。

```
int  SocketTlsReadAvailable(
   int           socket,              // ソケット
   uchar&        buffer[],            // ソケットからデータを読むためのバッファ
   const uint    buffer_maxlen        // 読まれるバイト数
   );
```

パラメータ

socket

[in] SocketCreate関数で返されるソケットハンドルです。 _LastErrorに無効なハンドルが渡されると、5270 (ERR_NETSOCKET_INVALIDHANDLE)がアクティブになります。

buffer

[out] データが読まれるuchar型配列への参照。動的配列サイズは、読み取られたバイト数だけ増えます。配列サイズはINT_MAX (2147483647)を超えることはできません。

buffer_maxlen

[in] buffer[]配列に読まれるバイト数。配列に収まらないデータはソケットに残り、次のSocketTlsReadAvailableまたはSocketTlsReadの呼び出しで受信できます。buffer_maxlenはINT_MAX (2147483647)を超えることはできません。

戻り値

成功の場合、読まれたバイト数。エラーの場合は -1 。

注意事項

関数の実行時にシステムソケットでエラーが発生した場合、SocketConnectによって確立された接続は中断されます。

データ読み込みエラーの場合は、エラー5273 (ERR_NETSOCKET_IO_ERROR)が _LastErrorに書かれます。

この関数は独自のスレッド内で実行されるエキスパートアドバイザーやスクリプトのみから呼び出すことが出来ます。指標から呼び出すと、GetLastError()はエラー4014「Function is not allowed for call(関数呼び出しの許可がありません)」を返します。

例：

```
//+------------------------------------------------------------------+
//|                                      SocketTlsReadAvailable.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"
#property script_show_inputs
//+------------------------------------------------------------------+
//| スクリプトの入力                                                 |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
input string InpMethod ="GET";             // メソッド( HEAD, GET)
input string InpServer ="www.google.com"; // サーバ
input uint   InpPort   =443;               // ポート
input uint   InpTimeout=5000;              // タイムアウト
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart(void)
  {
   Print("Server: ",InpServer);
   Print("Port: ",InpPort);
//--- ソケットを作成し、そのハンドルを取得する
   const int socket=SocketCreate();

   if(socket==INVALID_HANDLE)
     {
      Print("SocketCreate() failed. Error ",GetLastError());
      return;
     }
//--- ソケットシステムオブジェクトのデータの受信と送信のタイムアウトを設定する
   if(!SocketTimeouts(socket,InpTimeout,InpTimeout))
     {
      PrintFormat("SocketTimeouts(%u, %u) failed. Error %
d",InpTimeout,InpTimeout,GetLastError());
      SocketClose(socket);
      return;
     }
//--- ポート経由でサーバーに接続する
   if(!SocketConnect(socket,InpServer,InpPort,InpTimeout))
     {
      PrintFormat("SocketConnect('%s', %u, %u) failed. Error %
d",InpServer,InpPort,InpTimeout,GetLastError());
      SocketClose(socket);
      return;
     }
//--- ネットワーク接続を保護するために使用される証明書に関するデータを取得する
   string   subject,issuer,serial,thumbprint;
   datetime expiration;

   if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
     {
      Print("TLS certificate:");
      Print("   Owner:      ",subject);
      Print("   Issuer:     ",issuer);
      Print("   Number:     ",serial);
      Print("   Print:      ",thumbprint);
      Print("   Expiration: ",expiration);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   else
     {
      //--- サーバーが証明書を提供していないため、安全でない接続を報告して終了する
      Print("The connection is not secured by a certificate");
      SocketClose(socket);
      return;
     }
//--- サーバにリクエストを送信する
   string request=StringFormat("%s / HTTP/1.1\r\nHost: %s\r\nUser-Agent: MetaTrader
5\r\n\r\n",InpMethod,InpServer);

   if(HTTPSendTLS(socket,request))
     {
      //--- リクエストが送信され、応答が返される
      Print("\nRequest sent. Starting page loading...");
      uchar response[]; // all received data (document header and body)

      if(!HTTPRecvTLS(socket,response,InpTimeout))
        {
         Print("There were errors while reading the page");
         SocketClose(socket);
         return;
        }
      //--- 受信したデータのバイト数を報告する
      PrintFormat("%u bytes received",response.Size());
      //--- 取得したページヘッダーのみを表示する
      string result     =CharArrayToString(response,0,WHOLE_ARRAY,CP_UTF8);
      int    header_end=StringFind(result,"\r\n\r\n");

      if(header_end>0)
        {
         Print("\nHTTP answer header received:");
         Print(StringSubstr(result,0,header_end));
        }
     }
//--- 使用後はソケットを閉じる</t1>
   SocketClose(socket);
   /*
   結果:
   Server: www.google.com
   Port: 443
   TLS certificate:
      Owner: /CN=www.google.com
      Issuer: /C=US/O=Google Trust Services/CN=WR2
      Number: 0d:43:b1:4a:bb:9c:15:96:10:e1:3d:55:23:9f:25:4e
      Print: 89167618e5017f813aff981c88ce422dc1016bdf
      Expiration: 2024.12.30 08:26:35

   Request sent. Starting page loading...
```

© 2000-2025, MetaQuotes Ltd.

```
   HTTPRecvTLS: Document received within 27 attempts
   25185 bytes received

   HTTP answer header received:
   HTTP/1.1 200 OK
   Date: Fri, 25 Oct 2024 17:12:42 GMT
   Expires: -1
   Cache-Control: private, max-age=0
   Content-Type: text/html; charset=ISO-8859-1
   Content-Security-Policy-Report-Only: object-src 'none';base-uri 'self';script-src
'nonce-CUL2rdUOeAN7xIV6v0WUuQ' 'strict-dynamic' 'report-sample' 'unsafe-eval'
'unsafe-inline' https: http:;report-uri https://csp.withgoogle.com/csp/gws/other-hp
   Accept-CH: Sec-CH-Prefers-Color-Scheme
   P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
   Server: gws
   X-XSS-Protection: 0
   X-Frame-Options: SAMEORIGIN
   Set-Cookie: AEC=AVYB7coyYMCdweTDTaWeGYzmRnxzKGqsOEosH_VkbCn8xhWkFz6v0kxQFw;
expires=Wed, 23-Apr-2025 17:12:42 GMT; path=/; domain=.google.com; Secure; HttpOnly;
SameSite=lax
   Set-Cookie: NID=518=J02X02Ff4v_9sMcNoUz-1SolmuG08E26Gs438ik0J_SOJUMy7of-P-qup-
LaNSWVXUL8OjhOXpGIGuJQGIoEPBnzqDKCH-
46_FN4J2foHeWTlGG8bVVvQ44AHWLg1OXjrGp3CUBexYdczLWNy3LxEcb7eh6mxSvFzOelPC6-
vpXkaumLQ80x9gF_RpLcAYfN4ehT; expires=Sat, 26-Apr-2025 17:12:42 GMT; path=/;
domain=.google.com; HttpOnly
   Alt-Svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000
   Accept-Ranges: none
   Vary: Accept-Encoding
   Transfer-Encoding: chunked
   */
  }
//+------------------------------------------------------------------+
//|  安全な接続を介して HTTP リクエストを送信する
//+------------------------------------------------------------------+
bool HTTPSendTLS(int socket,const string request)
  {
//--- 文字列を文字配列に変換し、終端のゼロを破棄する
   char req[];
   int  len=StringToCharArray(request,req,0,WHOLE_ARRAY,CP_UTF8)-1;

   if(len<0)
      return false;

   return(SocketTlsSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//|  安全な接続を介して Web ページを取得する                                  |
//+------------------------------------------------------------------+
bool HTTPRecvTLS(int socket,uchar &response[],const uint timeout_ms)
  {
```

© 2000–2025, MetaQuotes Ltd.

```
//--- タイムアウトが切れる前に、安全なTLS接続から利用可能なデータを読み取る
   ulong timeout_check=GetTickCount64()+timeout_ms;
   uchar block[1024];   // ソケットからブロックデータを読み取るためのバッファ
   uint  attempt=0;     // 要求されたデータブロックの数
   int   err    =0;     // エラーコード

   ResetLastError();

   do
     {
      //--- ブロック単位で読み取り、最大1024バイト
      int len=SocketTlsReadAvailable(socket,block,1024);

      if(len>0)
        {
         attempt++;
         //--- 取得したデータブロックを結合する
         ArrayCopy(response,block,response.Size());
         //--- 取得したデータを分析し、ヘッダー、ページ本文、終了、読み込みエラーなどを定義する
         //...
         //...
         //...
         timeout_check=GetTickCount64()+timeout_ms;
        }
      else
         Sleep(10);

      err=GetLastError();
     }
   while(!IsStopped() && GetTickCount()<timeout_check && !err);
//--- 読み取り中にエラーは発生したか
   if(err)
     {
      Print("Error ",err);
      return(false);
     }

   PrintFormat("%s: Document received within %d attempts",__FUNCTION__,attempt);
   return(true);
  }
```

参照

SocketTimeouts、MathSwap

© 2000-2025, MetaQuotes Ltd.

SocketTlsSend

安全なTLS接続を介してデータを送信します。

```
int  SocketTlsSend(
   int           socket,              // ソケット
   const uchar&  buffer[],            // データバッファ
   uint          buffer_len           // バッファサイズ
   );
```

パラメータ

socket

[in] SocketCreate関数で返されるソケットハンドルです。無効なハンドルが渡されると、5270（ERR_NETSOCKET_INVALIDHANDLE）がLastErrorに書かれます。

buffer

[in] データが送信されるuchar型配列への参照

buffer_len

[in]「buffer」配列サイズ

戻り値

成功の場合、ソケットに書かれたバイト数。エラーの場合は -1 。

注意事項

関数の実行時にシステムソケットでエラーが発生した場合、SocketConnectによって確立された接続は中断されます。

データ書き込みエラーの場合は、エラー5273（ERR_NETSOCKET_IO_ERROR）が LastErrorに書かれます。

この関数は独自のスレッド内で実行されるエキスパートアドバイザーやスクリプトのみから呼び出すことが出来ます。指標から呼び出すと、GetLastError()はエラー4014「Function is not allowed for call（関数呼び出しの許可がありません）」を返します。

例：

```
//+------------------------------------------------------------------+
//|                                                SocketTlsSend.mq5 |
//|                                  Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"
#property description "Add Address to the list of allowed ones in the terminal
settings to let the example work"
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port    =443;
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart(void)
  {
   int socket=SocketCreate();
//--- ハンドルを確認する
   if(socket!=INVALID_HANDLE)
     {
      //--- 成功したら接続する
      if(SocketConnect(socket,Address,Port,1000))
        {
         PrintFormat("Established connection to %s:%d",Address,Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- 接続が証明書で保護されている場合は、そのデータを表示する
         if
(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS certificate:");
            Print("   Owner:       ",subject);
            Print("   Issuer:      ",issuer);
            Print("   Number:      ",serial);
            Print("   Print:       ",thumbprint);
            Print("   Expiration: ",expiration);
           }
         //--- サーバにGETリクエストを送信する
         string request="GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent:
MT5\r\n\r\n";
         char   req[];
         int    len=StringToCharArray(request,req)-1;

         if(len<0)
           {
            Print("StringToCharArray() failed. Error ", GetLastError());
            SocketClose(socket);
            return;
           }
         //--- if a secure TLS connection via port 443 is used
         if(SocketTlsSend(socket,req,len)==len)
           {
            Print("GET request sent");
            //--- 応答を読む
            if(!HTTPRecv(socket,1000))
               Print("Failed to get a response, error ",GetLastError());
           }
         else
            Print("Failed to send GET request, error ",GetLastError());
```

© 2000–2025, MetaQuotes Ltd.

```
         }
       else
         {
          PrintFormat("Connection to %s:%d failed, error %
d",Address,Port,GetLastError());
         }
       //--- 使用後はソケットを閉じる
       SocketClose(socket);
      }
   else
      Print("Failed to create a socket, error ",GetLastError());
  }
//+------------------------------------------------------------------+
//| サーバの応答を読む                                               |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout_ms)
  {
//--- ソケットからデータを読み出すが、タイムアウトを超えない範囲でデータを読み出す
   char   rsp[];
   string result;
   ulong  timeout_check=GetTickCount64()+timeout_ms;

   do
     {
      uint len=SocketIsReadable(socket);

      if(len)
        {
         //--- read and parse the data of a secure TLS connection
         int rsp_len=SocketTlsRead(socket,rsp,len);

         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- 応答ヘッダーのみを表示する
            int header_end=StringFind(result,"\r\n\r\n");

            if(header_end>0)
              {
               Print("HTTP answer header received:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
            //--- 読み取りタイムアウトの有効期限を更新する
            timeout_check=GetTickCount64()+timeout_ms;
           }
        }
     }
   while(GetTickCount64()<timeout_check && !IsStopped());
```

© 2000–2025, MetaQuotes Ltd.

```
   return(false);
  }
```

参照

SocketTimeouts、MathSwap、StringToCharArray

© 2000-2025, MetaQuotes Ltd.

WebRequest

この関数は指定されたサーバにHTTP要求を送信します。この関数には 2 つのバージョンがあります。

1ヘッダの Content-Type: application/x-www-form-urlencoded を使用して型「key=value」の単純なリクエストを送信します。

```
int  WebRequest(
   const string      method,           // HTTP メソッド
   const string      url,              // URL
   const string      cookie,           // クッキー
   const string      referer,          // リファラ
   int               timeout,          // 時間切れ
   const char        &data[],          // HTTPメッセージ本文の配列
   int               data_size,        // data[] 配列のバイトサイズ
   char              &result[],        // サーバ応答データを含む配列
   string            &result_headers   // サーバ応答ヘッダ
   );
```

2. 様々なWebサービスとのより柔軟な相互作用のためにカスタムヘッダセットを指定して任意の型のリクエストを送ります

```
int  WebRequest(
   const string      method,           // HTTP メソッド
   const string      url,              // URL
   const string      headers,          // ヘッダ
   int               timeout,          // 時間切れ
   const char        &data[],          // HTTPメッセージ本文の配列
   char              &result[],        // サーバ応答データを含む配列
   string            &result_headers   // サーバ応答ヘッダ
   );
```

パラメータ

method

[in] HTTP メソッド

url

[in] URL

headers

[in]「¥¥n」で改行された「key: value」型のリクエストヘッダ

cookie

[in] クッキー値

referer

[in] HTTPリクエストのReferrerヘッダー値

timeout

[in] ミリ秒単位のタイムアウト

© 2000-2025, MetaQuotes Ltd.

data[]

[in] HTTPメッセージ本文の配列

data_size

[in] data[] 配列のサイズ

result[]

[out] サーバ応答データを含む配列

result_headers

[out] サーバ応答ヘッダ

戻り値

エラーのHTTPサーバ応答コードまたは-1

注意事項

WebRequest() 関数を使用するには、「オプション」ウィンドウの「エキスパートアドバイザー(EA)」タブで許可されたURLのリストに必要なサーバのアドレスを追加します。サーバのポートは指定されたプロトコルに基づいて「http://」には 80、「https://」には 443 が自動的に選択されます。

WebRequest() 関数は同期的で、プログラムの実行を停止してリクエストされたサーバからの応答を待機します。指標は 1 つのシンボルのに使用される全ての指標やチャートによって共有される共通のスレッドで実行されるので、応答までに長時間を必要とする場合があるこの関数は、指標からの呼び出しには使用出来ません。シンボルのチャートの 1 つの指標パフォーマンス遅延が、同じシンボルの全てのグラフの更新を停止することになります。

この関数は独自のスレッド内で実行されるエキスパートアドバイザーやスクリプトのみから呼び出すことが出来ます。この関数が指標から呼ばれた場合、GetLastError() はエラー 4014「Function is not allowed for call(関数呼び出しの許可がありません)」を返します。

WebRequest() はストラテジーテスター内では実行出来ません。

例:

```
void OnStart()
  {
   string cookie=NULL,headers;
   char   post[],result[];
   string url="https://finance.yahoo.com";
//--- サーバへのアクセスを有効にするには、URL "https://finance.yahoo.com"を
//--- メインメニュー->ツール->オプションの「エキスパートアドバイザー(EA)」タブの許可されたURLのリストに追加する必要がある
//--- 最後のエラーコードのリセット
   ResetLastError();
//--- Yahoo! ファイナンスからhtml ページを読み込む
   int res=WebRequest("GET",url,cookie,NULL,500,post,0,result,headers);
   if(res==-1)
     {
      Print("Error in WebRequest. Error code  =",GetLastError());
      //--- おそらくURLがリストされていないので、アドレスを追加する必要についてのメッセージを表示する
      MessageBox("Add the address '"+url+"' to the list of allowed URLs on tab 'Expert Advisors'","Error",MB_ICONINFORMATION);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
    else
      {
       if(res==200)
         {
          //--- ダウンロード成功
          PrintFormat("The file has been successfully downloaded, File size %d
byte.",ArraySize(result));
          //PrintFormat("Server headers: %s",headers);
          //--- ファイルでのデータ保存
          int filehandle=FileOpen("url.htm",FILE_WRITE|FILE_BIN);
          if(filehandle!=INVALID_HANDLE)
            {
             //--- result[] 配列の内容のファイルでの保存
             FileWriteArray(filehandle,result,0,ArraySize(result));
             //--- ファイルを閉じる
             FileClose(filehandle);
            }
          else
             Print("Error in FileOpen. Error code =",GetLastError());
         }
       else
          PrintFormat("Downloading '%s' failed, error code %d",url,res);
      }
  }
```

© 2000–2025, MetaQuotes Ltd.

SendFTP

「FTP」タブの設定ウィンドウで指定されたアドレスにファイルを送信します。

```
bool  SendFTP(
   string  filename,           // ftp で送信されるファイル
   string  ftp_path=NULL       // ftp カタログ
   );
```

パラメータ

filename

[in] 送られたファイルの名称

ftp_path=NULL

[in] FTP カタログディレクトリが指定されていない場合は、設定に記載されたディレクトリが使用されます。

戻り値

失敗の場合 false

注意事項

送信されるファイルは*terminal_directory\MQL5\files* フォルダまたはそのサブディレクトリに位置しなければいけません。FTP アドレス及び/またはアクセスパスワードが設定で指定されていない場合は、送信が実行されません。

SendFTP() 関数はストラテジーテスター内では使用できません。

例：

```
//+------------------------------------------------------------------+
//|                                                      SendFTP.mq5 |
//|                                  Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"

#define   FILENAME  "SomeFile.bin"
#define   PATH      NULL
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart(void)
  {
//--- 端末のFTPアドレスにファイルを送信する権限を確認する
   if(!TerminalInfoInteger(TERMINAL_FTP_ENABLED))
     {
      Print("Error. The client terminal does not have permission to send messages to
the FTP-address");
```

© 2000–2025, MetaQuotes Ltd.

```
      return;
     }
//--- ファイルを送信する
   ResetLastError();
   if(!SendFTP(FILENAME, PATH))
      Print("SendFTP() failed. Error ",GetLastError());
  }
```

© 2000–2025, MetaQuotes Ltd.

SendMail

「メール」タブの設定ウィンドウで指定されたアドレスに電子メールを送信します。

```
bool  SendMail(
   string  subject,        // ヘッダ
   string  some_text       // メール本文
   );
```

パラメータ

subject

[in] メールヘッダ

some_text

[in] メール本文

戻り値

電子メールが送信キューに入れられた場合は true、それ以外の場合は false

注意事項

送信は設定で禁止することができ、電子メールアドレスも同様に省略することが出来ます。エラー情報を取得するには、GetLastError() が呼ばれます。

SendMail() 関数はストラテジーテスター内では使用できません。

例：

```
//+------------------------------------------------------------------+
//|                                                     SendMail.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd.   |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"

#define   SUBJECT   "Test SendMail"
#define   TEXT      "Text for SendMail() function"
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart(void)
  {
//--- 端末でメールを送信する権限を確認する
   if(!TerminalInfoInteger(TERMINAL_EMAIL_ENABLED))
     {
      Print("Error. The client terminal does not have permission to send email
messages");
      return;
     }
```

© 2000–2025, MetaQuotes Ltd.

```
//--- メールを送信する
   ResetLastError();
   if(!SendMail(SUBJECT, TEXT))
      Print("SendMail() failed. Error ",GetLastError());
  }
```

© 2000–2025, MetaQuotes Ltd.

SendNotification

MetaQuotes ID が「通知」タブで指定されているモバイル端末にプッシュ通知を送信します。

```
bool  SendNotification(
   string  text          // 通知テキスト
   );
```

パラメータ

text

[in] 通知テキストメッセージの長さは 255 文字を超えることは出来ません。

戻り値

通知が正常に端末から送信された場合は true、失敗なら false。失敗したプッシュ後に通知をチェックした際、GetLastError () は次のいずれかのエラーを返します。

- 4515 - ERR_NOTIFICATION_SEND_FAILED,
- 4516 - ERR_NOTIFICATION_WRONG_PARAMETER,
- 4517 - ERR_NOTIFICATION_WRONG_SETTINGS,
- 4518 - ERR_NOTIFICATION_TOO_FREQUENT.

注意事項

SendNotification() 関数には厳格な使用制限が設定されており、呼び出しは毎秒 2 以下と毎分 10 以下に限られています。 使用頻度は動的に監視されます。制限違反の場合にはこの関数は無効にされることが出来ます。

SendNotification() 関数はストラテジーテスター内では使用できません。

例：

```
//+------------------------------------------------------------------+
//|                                                SendNotification.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"

#define   MESSAGE   "Test Message"
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                                |
//+------------------------------------------------------------------+
void OnStart(void)
  {
//--- 端末で通知を送信する権限を確認する
   if(!TerminalInfoInteger(TERMINAL_NOTIFICATIONS_ENABLED))
     {
      Print("Error. The client terminal does not have permission to send
notifications");
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }
//--- 通知を送信する
   ResetLastError();
   if(!SendNotification(MESSAGE))
      Print("SendNotification() failed. Error ",GetLastError());
  }
```

© 2000–2025, MetaQuotes Ltd.

クライアント端末のグローバル変数

これは、グローバル変数を操作するための関数群です。

クライアント端末のグローバル変数は MQL5 プログラムのグローバルスコープで宣言された変数と混乱されてはいけません。

グローバル変数は、最終アクセス後4週間クライアント端末に保存されてから自動的に削除されます。グローバル変数へのアクセスは、新しい値の設定だけではなく読み込みでもあります。

クライアント端末のグローバル変数は、クライアント端末で実行されている全ての MQL5 プログラムから同時にアクセス出来ます。

関数	アクション
GlobalVariableCheck	指定された名称のグローバル変数の存在をチェックします。
GlobalVariableTime	グローバル変数の最終アクセス時刻を返します。
GlobalVariableDel	グローバル変数を削除します。
GlobalVariableGet	グローバル変数の値を返します。
GlobalVariableName	グローバル変数の名称をグローバル変数のリスト内の序数で返します。
GlobalVariableSet	グローバル変数に新しい値を設定します。
GlobalVariablesFlush	強制的にディスクに全てのグローバル変数の内容を保存します。
GlobalVariableTemp	グローバル変数に端末の現在のセッションでのみ存在する新しい値を設定します。
GlobalVariableSetOnCondition	条件付きで既存するグローバル変数に新しい値を設定します。
GlobalVariablesDeleteAll	指定されたプレフィックスを持つグローバル変数を削除します。
GlobalVariablesTotal	グローバル変数の合計数を返します。

© 2000–2025, MetaQuotes Ltd.

GlobalVariableCheck

指定された名称のグローバル変数の存在をチェックします。

```
bool  GlobalVariableCheck(
   string  name      // グローバル変数名
   );
```

パラメータ

name

[in] グローバル変数名

戻り値

グローバル変数が存在すれば true、それ以外の場合は false。

グローバル変数は、最終アクセス後 4 週間クライアント端末に保存されてから自動的に削除されます。

参照

GlobalVariableTime()

© 2000–2025, MetaQuotes Ltd.

GlobalVariableTime

グローバル変数の最終アクセス時刻を返します。

```
datetime  GlobalVariableTime(
   string  name      // 名称
   );
```

パラメータ

name

[in] グローバル変数名

戻り値

指定されたグローバル変数の最終アクセス時刻。GlobalVariableGet() 及び GlobalVariableCheck() 関数などを使用して変数を値によって参照する場合にもアクセス時刻が変更されます。エラーの詳細を取得するには GetLastError() 関数が呼ばれます。

注意事項

グローバル変数は、最終アクセス後4週間クライアント端末に保存されます。その後、自動的に削除されます。

参照

GlobalVariableCheck()

© 2000-2025, MetaQuotes Ltd.

GlobalVariableDel

クライアント端末からグローバル変数を削除します。

```
bool  GlobalVariableDel(
   string  name      // グローバル変数名
   );
```

パラメータ

name

[in] グローバル変数名

戻り値

成功の場合は true、それ以外の場合は false。エラー情報を取得するには GetLastError() 関数を呼び出します。

注意事項

グローバル変数は、最終アクセス後 4 週間クライアント端末に保存されてから自動的に削除されます。

© 2000–2025, MetaQuotes Ltd.

GlobalVariableGet

クライアント端末に存在するグローバル変数の値を返します。この関数には 2 つのバージョンがあります。

1. すぐにグローバル変数の値を返します。

```
double  GlobalVariableGet(
   string  name      // グローバル変数名
   );
```

2. 関数実行の成功に応じて true または false を返します。成功した場合、クライアント端末のグローバル変数は、2 番目のパラメータで参照によって渡された変数に格納されます。

```
bool  GlobalVariableGet(
   string  name,              // グローバル変数名
   double& double_var         // グローバル変数の値を含む変数
   );
```

パラメータ

name

[in] グローバル変数名

double_var

[out] クライアント端末のグローバル変数に格納された値を受け入れる double 型のターゲット変数

戻り値

既存するグローバル変数の値（エラーの場合は 0）。エラーの詳細については GetLastError() を呼びます。

注意事項

グローバル変数は、最終アクセス後 4 週間クライアント端末に保存されてから自動的に削除されます。

© 2000–2025, MetaQuotes Ltd.

GlobalVariableName

グローバル変数の名称を序数で返します。

```
string  GlobalVariableName(
   int  index      // グローバル変数のリスト内の序数
   );
```

パラメータ

index

[in] グローバル変数のリスト内の序数 0 以上で GlobalVariablesTotal() 未満

戻り値

グローバル変数のリスト内の序数としてのグローバル変数の名称。エラーの詳細については GetLastError() を呼びます。

注意事項

グローバル変数は、最終アクセス後 4 週間クライアント端末に保存されてから自動的に削除されます。

© 2000–2025, MetaQuotes Ltd.

GlobalVariableSet

グローバル変数に新しい値を設定します。変数が存在しない場合、システムは新しいグローバル変数を作成します。

```
datetime  GlobalVariableSet(
   string  name,       // グローバル変数名
   double  value       // 設定する値
   );
```

パラメータ

name

[in] グローバル変数名

value

[in] 新しい数値

戻り値

成功の場合は最終更新時刻、それ以外の場合は 0。エラーの詳細については GetLastError() を呼びます。

注意事項

グローバル変数名は 63 字を超えられません。グローバル変数は、最終使用後4週間クライアント端末に保存されてから自動的に削除されます。

© 2000–2025, MetaQuotes Ltd.

GlobalVariablesFlush

強制的にディスクに全てのグローバル変数の内容を保存します。

```
void  GlobalVariablesFlush();
```

戻り値

なし

注意事項

端末は、作業が終わった時に全てのグローバル変数を書き込みますが、突然のコンピュータ操作の失敗でデータが失われる可能性があります。この関数は、有事の場合に独立してグローバル変数を保存するプロセスを制御することが出来ます。

© 2000-2025, MetaQuotes Ltd.

GlobalVariableTemp

この関数は、一時的なグローバル変数の作成を試みます。変数が存在しない場合、システムは新しい一時的なグローバル変数を作成します。

```
bool  GlobalVariableTemp(
   string  name      // グローバル変数名
   );
```

パラメータ

name

[in] 一時的なグローバル変数の名称

戻り値

成功の場合は true、それ以外の場合は false。エラーの詳細には GetLastError() 関数が呼ばれます。

注意事項

一時的なグローバル変数はクライアント端末の実行中ににのみ存在し、端末シャットダウン後、自動的に削除されます。GlobalVariablesFlush() 実行の際に一時的なグローバル変数はディスクに書きだされないことにご注意下さい。

一時的なグローバル変数は、作成後はクライアント端末のグローバル変数と同じようにアクセスまた変更出来ます。

© 2000–2025, MetaQuotes Ltd.

GlobalVariableSetOnCondition

現在の値が3番目のcheck_value パラメータに等しい場合、既存のグローバル変数の新しい値を設定します。グローバル変数が存在しない場合、ERR_GLOBALVARIABLE_NOT_FOUND（4501）エラーが発生しfalseが戻されます。

```
bool  GlobalVariableSetOnCondition(
   string  name,              // グローバル変数名
   double  value,             // 条件がtrueの場合の変数の新しい値
   double  check_value        // 値の条件をチェック
   );
```

パラメータ

name

[in] グローバル変数名

value

[in] 新しい値

check_value

[in] グローバルの現在の値をチェックする値

戻り値

成功の場合はtrue、それ以外の場合はfalse。エラーの詳細については GetLastError() を呼びます。グローバル変数の現在の値がcheck_valueと異なる場合 falseが戻されます。

注意事項

この関数は、グローバル変数へのアトミックなアクセスを提供し、1つのクライアント端末内で同時に作業するエキスパートアドバイザーへのミューテックスの提供のために使用することが出来ます。

© 2000-2025, MetaQuotes Ltd.

GlobalVariablesDeleteAll

クライアント端末のグローバル変数を削除します。

```
int  GlobalVariablesDeleteAll(
   string      prefix_name=NULL,      // プレフィックスで始まる名称を持つ全てのグローバル変数
   datetime    limit_data=0           // この日付より前に変更された全てのグローバル変数
   );
```

パラメータ

prefix_name=NULL

[in] 削除に使用されるプレフィックス。プレフィックスが、NULLまたは空の文字列に指定されると、データの基準を満たす全ての変数が削除されます。

limit_data=0

[in] 最終変更時刻によってでグローバル変数を選択する時に使用される日付。この関数はこの日付け以前に変更されたグローバル変数を削除します。パラメータがゼロの場合は、最初の基準（プレフィックス）を満たす全ての変数が削除されます。

戻り値

削除された変数の数

注意事項

両方のオプションがゼロに等しい場合（prefix_name= NULLとlimit_data=0）、この関数は端末の全てのグローバル変数を削除します。両方のパラメータが指定されている場合、両方のパラメータに対応するグローバル変数が削除されます。

グローバル変数は、最終アクセス後4週間クライアント端末に保存されてから自動的に削除されます。

© 2000–2025, MetaQuotes Ltd.

GlobalVariablesTotal

クライアント端末に存在するグローバル変数の数を返します。

```
int  GlobalVariablesTotal();
```

戻り値

グローバル変数の数

注意事項

グローバル変数は、最終アクセス後4週間クライアント端末に保存されてから自動的に削除されます。グローバル変数へのアクセスは、新しい値の設定だけではなく読み込みでもあります。

© 2000–2025, MetaQuotes Ltd.

ファイル関数

これは、ファイルを操作するための関数群です。

セキュリティ上の理由から、MQL5 言語ではファイルの扱いは厳しく制御されています。MQL5 手段を用いて操作されるファイルは、ファイルサンドボックスの外に存在することは出来ません。

作業ファイルを配置出来るディレクトリ（サブディレクトリで）は 2 つあります。

- terminal_data_folder¥MQL5¥FILES¥（端末メニューで、「ファイル」→「データディレクトリを開く」を選択）
- コンピュータにインストールされている全ての端末のための共通のフォルダ（通常 C:¥Documents and Settings\All Users\Application Data\MetaQuotes\Terminal\Common\Files ）

ENUM_TERMINAL_INFO_STRING 列挙と TerminalInfoString() 関数を使用してこれらのカタログ名を取得することが出来ます。

```
//--- 端末データが格納されるフォルダ
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
//--- 全てのクライアント端末に共通なフォルダー
   string common_data_path=TerminalInfoString(TERMINAL_COMMONDATA_PATH);
```

他のディレクトリにあるファイルの操作は禁止されています。

ファイル関数は、いわゆる「名前付きパイプ」での作業を可能にします。そのためには FileOpen() 関数が適切なパラメータと呼ばれるべきです。

関数	アクション
FileSelectDialog	ファイルまたはフォルダーを開く/作成ダイアログを作成します。
FileFindFirst	指定されたフィルタに基づいて、ディレクトリ内のファイルの検索を開始します。
FileFindNext	FileFindFirst() 関数で始めた検索を続けます。
FileFindClose	検索ハンドルを閉じます。
FileOpen	指定された名称とフラグでファイルを開きます。
FileDelete	指定されたファイルを削除します。
FileFlush	入力/出力ファイルのバッファに残っている全てのデータをディスクに書き込みます。
FileGetInteger	ファイルの整数プロパティを取得します。
FileIsEnding	読み出し処理において、ファイルの終わりを定義します。
FileIsLineEnding	読み出し処理において、テキストファイル内の行の終わりを定義します。
FileClose	以前に開いたファイルを閉じます。
FileIsExist	ファイルの存在をチェックします。
FileCopy	ファイルをローカルまたは共有フォルダから別のファイルに複製します。
FileMove	ファイルを移動または名称を変更します。

© 2000–2025, MetaQuotes Ltd.

関数	アクション
FileReadArray	BIN形式のファイルから文字列を除く全ての型の配列を読み込みます。
FileReadBool	CSV ファイルから、現在の位置から区切り文字までの文字列（またはテキスト行の終わり）を読み込み、読んだ文字列を bool 型の値に変換します。
FileReadDatetime	CSV ファイルから、「YYYY.MM.DD HH:MM:SS」、「YYYY.MM.DD」または「HH:MM:SS」フォーマットの文字列を読み込みdatetime 値に変換します。
FileReadDouble	ファイルポインタの現在位置から double 値を読み込みます。
FileReadFloat	ファイルポインタの現在位置から float 値を読み込みます。
FileReadInteger	ファイルポインタの現在位置から int、short または char 値を読み込みます。
FileReadLong	ファイルポインタの現在位置から long 値を読み込みます。
FileReadNumber	CSV ファイルから、現在の位置から区切り文字までの文字列（またはテキスト行の終わり）を読み込み、読んだ文字列を double 型の値に変換します。
FileReadString	ファイルポインタの現在位置から string 値を読み込みます。
FileReadStruct	バイナリファイルのファイルポインタの現在の位置から、パラメータとして渡される構造体にコンテンツを読み込みます。
FileSeek	ファイルポインタの位置を指定された位置から指定されたバイト数で移動します。
FileSize	対応する開かれているファイルのサイズを返します。
FileTell	対応する開かれているファイルのファイルポインタの現在位置を返します。
FileWrite	データを CSV または TXT 形式のファイルに書き込みます。
FileWriteArray	BIN 形式のファイルに文字列を除いた任意の型の配列を書き込みます。
FileWriteDouble	バイナリファイルのファイルポインタの現在位置にdouble 型の値を書き込みます。
FileWriteFloat	バイナリファイルのファイルポインタの現在位置にfloat 型の値を書き込みます。
FileWriteInteger	バイナリファイルのファイルポインタの現在位置に int 型の値を書き込みます。
FileWriteLong	バイナリファイルのファイルポインタの現在位置に long 型の値を書き込みます。
FileWriteString	BIN または TXT ファイルのファイルポインタの現在位置に string パラメータの値を書き込みます。

© 2000–2025, MetaQuotes Ltd.

関数	アクション
FileWriteStruct	バイナリファイルのファイルのファイルポインタの現在位置にパラメータとして割らされた構造体のコンテンツを書き込みます。
FileLoad	指定されたバイナリファイルのすべてのデータを渡された数値型または単純な構造体型の配列に読み込みます
FileSave	配列のすべての要素をパラメータとして渡されたバイナリファイルに書き込みます
FolderCreate	Files ディレクトリ内にフォルダを作成します。
FolderDelete	選択されたディレクトリを削除します。フォルダが空でない場合は、削除出来ません。
FolderClean	指定されたフォルダ内の全てのファイルを削除します。

ファイルが FileOpen() を使用して書き込みのために開かれている場合、パスで指定された全てのサブフォルダは存在しない場合でも作成されます。

© 2000–2025, MetaQuotes Ltd.

FileSelectDialog

ファイルまたはフォルダーを開く/作成ダイアログを作成します。

```
int  FileSelectDialog(
   string   caption,              // ウィンドウヘッダ
   string   initial_dir,          // デフォルトディレクトリ
   string   filter,               // 拡張子フィルタ
   uint     flags,                // フラグの組み合わせ
   string&  filenames[],          // ファイル名の配列
   string   default_filename      // デフォルトファイル名
   );
```

パラメータ

caption

[in] ダイアログウィンドウヘッダ

initial_dir

[in] MQL5¥Filesに早退した初期ディレクトリ名。その内容はダイアログボックスに表示されます。値がNULLの場合、ダイアログにはMQL5¥Filesが表示されます。

filter

[in] 選択ダイアログウィンドウに表示されるファイルの拡張子フィルタ。他の形式のファイルは非表示です。

フラグ

[in] ダイアログウィンドウモードを定義するフラグの組み合わせ。フラグは次のように定義されます。
FSD_WRITE_FILE – ファイルを開くダイアログ
FSD_SELECT_FOLDER – フォルダのみの選択を許可する
FSD_ALLOW_MULTISELECT – 複数ファイルの選択を許可する
FSD_FILE_MUST_EXIST – 選択したファイルは存在するべきである
FSD_COMMON_FOLDER – ファイルはすべてのクライアント端末の共通フォルダ¥Terminal¥Common¥Filesにある

filenames[]

[out] 選択したファイル/フォルダーの名前が配置される文字列の配列。

default_filename

[in] デフォルトファイル/フォルダー名。指定されている場合、名前は自動的に開いているダイアログに追加され、テスト時に*filenames[]*配列に返されます。

戻り値

正常に完了した場合、関数は、*filenames[]*で名前を取得できる選択されたファイルの数を返します。ユーザーがファイルを選択せずにダイアログを閉じると、関数は0を返します。実行に失敗した場合、 0未満の値が返されます。エラーコードは、GetLastError()を使用して取得できます。

注意事項

© 2000–2025, MetaQuotes Ltd.

セキュリティ上の理由から、MQL5 言語ではファイルの扱いは厳しく制御されています。MQL5言語を用いて操作されるファイルは、ファイルサンドボックス(MQL5¥Filesディレクトリ)の外に存在することは出来ません。

*initial_dir*の名前は、MQL5¥Files(またはテストの場合はtesting_agent_directory¥MQL5 ¥Files)のクライアント端末ディレクトリで検索されます。If FSD_COMMON_FOLDERがフラグ間で設定されている場合、初期ディレクトリの検索は、すべてのクライアント端末の共通フォルダ¥Terminal¥Common¥Filesで行われます。

*filter*パラメータは有効なファイルを示すもので、「<description 1>|<extension 1>|<description 2>|<extension 2>...」形式(例:「Text files (*.txt)|*.txt|All files (*.*)|*.*」)に設定されるべきです。初めの拡張し「Text files (*.txt)|*.txt」はデフォルトファイルタイプとして選択されています。
filter=NULLの場合、ダイアログウィンドウでのファイル選択のマスクは「All Files (*.*)|*.*」です。

*default_filename*パラメータが設定されている場合は、FileSelectDialog()の呼び出しは1を返し、*default_filename*値辞退はビジュアルテスト以外では*filenames[]*配列にコピーされます。

FileSelectDialog() を呼び出すと、ユーザーの応答を待っている間実行スレッドがずっと中断されるため、この関数をカスタム指標で使用することはできません。各銘柄のすべての指標は単一のスレッドで実行されるため、この銘柄のすべての時間枠のすべてのチャートが一時停止されます。

例:

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- クライアント端末の共通フォルダからダウンロードするテキストファイルの名前を取得する
   string filenames[];
   if(FileSelectDialog("Select files to download", NULL,
                       "Text files (*.txt)|*.txt|All files (*.*)|*.*",
                       FSD_ALLOW_MULTISELECT|FSD_COMMON_FOLDER, filenames,
"data.txt")>0)
     {
      //--- 選択した各ファイルの名前を表示する
      int total=ArraySize(filenames);
      for(int i=0; i<total; i++)
         Print(i, ": ", filenames[i]);
     }
   else
     {
     Print("Files not selected");
     }
//---
  }
```

参照

FileOpen、FileIsExist、FileDelete、FileMove、FolderCreate、FolderDelete、FolderClean、Flags of opening files

© 2000-2025, MetaQuotes Ltd.

FileFindFirst

この関数は指定されたフィルタに基づいてディレクトリ内のファイルまたはサブディレクトリの検索を開始します。

```
long  FileFindFirst(
   const string   file_filter,          // 文字列の検索フィルタ
   string&        returned_filename,    // 見つかったファイルまたはサブディレクトリの名称
   int            common_flag=0         // 検索を定義する
   );
```

パラメータ

file_filter

[in] 検索フィルタ。ファイル検索に使用される、¥Files ディレクトリに相対したサブディレクトリ（またはネストされたサブディレクトリのシーケンス）が指定されます。

returned_filename

[out] 戻されたパラメータ。成功の場合には、最初に見つかったファイルまたはサブディレクトリの名称が配置されます。拡張子を含んだファイル名のみが返されます。ディレクトリ及びサブディレクトリは検索フィルタに含まれている場合でも結果に含まれません。

common_flag

[in] ファイルの場所を決めるフラグ。common_flag = FILE_COMMON の場合、ファイルは全てのクライアント端末の共有フォルダ ¥Terminal¥Common¥Files 内に存在します。その他の場合、ファイルはローカルフォルダに存在します。

戻り値

FileFindNext() 関数によってファイルとサブディレクトリのさらなるソートに使用される検索オブジェクトのハンドル、または、フィルタに対応するファイルとサブディレクトリが存在しない場合（特別なケースではディレクトリが空の場合）はINVALID_HANDLE。検索後はFileFindClose() 関数を使用してハンドルを閉じるべきです。

注意事項

セキュリティ上の理由から、MQL5 言語ではファイルの扱いは厳しく制御されています。MQL5 手段を用いて操作されるファイルは、ファイルサンドボックスの外に存在することは出来ません。

例：

© 2000–2025, MetaQuotes Ltd.

```
//--- スクリプト起動時に入力パラメータウィンドウを表示する
#property script_show_inputs
//--- フィルタ
input string InpFilter="Dir1\\*";
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
   string file_name;
   string int_dir="";
   int    i=1,pos=0,last_pos=-1;
//--- 最後のバックスラッシュを探す
   while(!IsStopped())
     {
      pos=StringFind(InpFilter,"\\",pos+1);
      if(pos>=0)
         last_pos=pos;
      else
         break;
     }
//--- フィルタにフォルダ名が含まれる
   if(last_pos>=0)
      int_dir=StringSubstr(InpFilter,0,last_pos+1);
//--- ローカルフォルダのルートで検索ハンドルを取得する
   long search_handle=FileFindFirst(InpFilter,file_name);
//--- FileFindFirst() の実行が成功したかをチェックする
   if(search_handle!=INVALID_HANDLE)
     {
      //--- 渡された文字列がファイル名かディレクトリ名かをループでチェックする
      do
        {
         ResetLastError();
         //--- ファイルの場合、関数が true を返し、ディレクトリの場合 ERR_FILE_IS_DIRECTORY エラーを返す
         FileIsExist(int_dir+file_name);
         PrintFormat("%d : %s name = %s",i,GetLastError()
==ERR_FILE_IS_DIRECTORY ?"Directory" : "File",file_name);
         i++;
        }
      while(FileFindNext(search_handle,file_name));
      //--- 検索ハンドルを閉じる
      FileFindClose(search_handle);
     }
   else
      Print("Files not found!");
  }
```

参照

FileFindNext、FileFindClose

© 2000-2025, MetaQuotes Ltd.

FileFindNext

この関数は FileFindFirst() 関数で始めた検索を続けます。

```
bool  FileFindNext(
   long      search_handle,          // 検索ハンドル
   string&   returned_filename       // 見つかったファイルまたはサブディレクトリの名称
   );
```

パラメータ

search_handle

[in] FileFindFirst() で取得される検索ハンドル

returned_filename

[out] 次に見つかったファイルまたはサブディレクトリの名称拡張子を含んだファイル名のみが返されます。ディレクトリ及びサブディレクトリは検索フィルタに含まれている場合でも結果に含まれません。

戻り値

成功の場合は true、それ以外の場合は false

例：

```
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- フィルタ
input string InpFilter="*";
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                 |
//+------------------------------------------------------------------+
void OnStart()
  {
   string file_name;
   int    i=1;
//--- ローカルフォルダのルートで検索ハンドルを受け取る
   long search_handle=FileFindFirst(InpFilter,file_name);
//--- FileFindFirst() の実行が成功したかをチェックする
   if(search_handle!=INVALID_HANDLE)
     {
      //--- 渡された文字列がファイル名かディレクトリ名かをループでチェックする
      do
        {
         ResetLastError();
         //--- ファイルであれば trueを返し、ディレクトリであれば ERR_FILE_IS_DIRECTORY エラーを生成する
         FileIsExist(file_name);
         PrintFormat("%d : %s name = %s",i,GetLastError()==ERR_FILE_IS_DIRECTORY ?
"Directory" : "File",file_name);
         i++;
        }
      while(FileFindNext(search_handle,file_name));
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 検索ハンドルを閉じる
      FileFindClose(search_handle);
     }
   else
      Print("Files not found!");
  }
```

参照

FileFindFirst、FileFindClose

© 2000-2025, MetaQuotes Ltd.

FileFindClose

この関数は検索ハンドルを閉じます。

```
void  FileFindClose(
   long  search_handle      //  検索ハンドル
   );
```

パラメータ

search_handle

[in] FileFindFirst() で取得される検索ハンドル

戻り値

なし

注意事項

この関数は、システムリソースを解放するために呼び出されなければなりません。

例:

```
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- フィルタ
input string InpFilter="*";
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
   string file_name;
   int    i=1;
//--- ローカルフォルダのルートで検索ハンドルを受け取る
   long search_handle=FileFindFirst(InpFilter,file_name);
//--- FileFindFirst() の実行が成功したかをチェックする
   if(search_handle!=INVALID_HANDLE)
     {
      //--- 渡された文字列がファイル名かディレクトリ名かをループでチェックする
      do
        {
         ResetLastError();
         //--- ファイルであれば trueを返し、ディレクトリであればエラーを生成する
         FileIsExist(file_name);
         PrintFormat("%d : %s name = %s",i,GetLastError()==5018 ? "Directory" :
"File",file_name);
         i++;
        }
      while(FileFindNext(search_handle,file_name));
      //--- 検索ハンドルを閉じる
      FileFindClose(search_handle);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   else
      Print("Files not found!");
  }
```

参照

FileFindFirst、FileFindNext

© 2000–2025, MetaQuotes Ltd.

FileIsExist

ファイルの存在をチェックします。

```
bool  FileIsExist(
   const string  file_name,        // ファイル名
   int           common_flag=0     // 検索の場所
   );
```

パラメータ

file_name

[in] チェックされているファイルの名称

common_flag=0

[in] ファイルの場所を決めるフラグ。common_flag = FILE_COMMON の場合、ファイルは全てのクライアント端末の共有フォルダ ¥Terminal¥Common¥Files 内に存在します。その他の場合、ファイルはローカルフォルダに存在します。

戻り値

指定されたファイルが存在する場合は true

注意事項

チェックされたファイルはサブディレクトリである可能性があります。この場合 FileIsExist() 関数は false を戻し error 5018 - This is a directory, not a file（これはディレクトリでファイルではありません）が _LastError 変数にログされます。（FileFindFirst 関数の例を参照）。

セキュリティ上の理由から、MQL5 言語ではファイルの扱いは厳しく制御されています。MQL5 手段を用いて操作されるファイルは、ファイルサンドボックスの外に存在することは出来ません。

common_flag = FILE_COMMON の場合、この関数はファイルを全てのクライアント端末の共有フォルダ \Terminal\Common\Files で探します。他の場合、ローカルフォルダ（MQL5¥Files またはテストの際には MQL5¥Tester¥Files）が探されます。

例：

```
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- 古いファイルの日付
input datetime InpFilesDate=D'2013.01.01 00:00';
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
   string   file_name;       // ファイル名を格納する変数
   string   filter="*.txt"; // ファイル検索フィルタ
   datetime create_date;     // ファイル作成日
   string   files[];         // ファイル名のリスト
   int      def_size=25;     // デフォルトの配列サイズ
   int      size=0;          // ファイル数
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 配列へのメモリ追加
   ArrayResize(files,def_size);
//--- ローカルフォルダのルートで検索ハンドルを受け取る
   long search_handle=FileFindFirst(filter,file_name);
//--- FileFindFirst() の実行が成功したかをチェックする
   if(search_handle!=INVALID_HANDLE)
     {
      //--- ループでファイルを検索する
      do
        {
         files[size]=file_name;
         //--- 配列サイズを増やす
         size++;
         if(size==def_size)
           {
            def_size+=25;
            ArrayResize(files,def_size);
           }
         //--- エラー値をリセットする
         ResetLastError();
         //--- 作成日を受け取る
         create_date=(datetime)FileGetInteger(file_name,FILE_CREATE_DATE,false);
         //--- ファイルが古いかをチェックする
         if(create_date<InpFilesDate)
           {
            PrintFormat("%s file deleted!",file_name);
            //--- 古いファイルを削除する
            FileDelete(file_name);
           }
        }
      while(FileFindNext(search_handle,file_name));
      //--- 検索ハンドルを閉じる
      FileFindClose(search_handle);
     }
   else
     {
      Print("Files not found!");
      return;
     }
//--- 何のファイルが残っているかをチェックする
   PrintFormat("Results:");
   for(int i=0;i<size;i++)
     {
      if(FileIsExist(files[i]))
         PrintFormat("%s file exists!",files[i]);
      else
         PrintFormat("%s file deleted!",files[i]);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

参照

FileFindFirst

© 2000–2025, MetaQuotes Ltd.

FileOpen

この関数は、指定された名称とフラグでファイルを開きます。

```
int  FileOpen(
   string   file_name,           // ファイル名
   int      open_flags,          // フラグの組み合わせ
   short    delimiter='\t',      // デリミタ
   uint     codepage=CP_ACP      // コードページ
   );
```

パラメータ

file_name

[in] ファイルの名称はサブフォルダを含むことが出来ます。ファイルが書き込み用に開かれている場合は、存在しないサブフォルダは作成されます。

open_flags

[in] ファイル操作のモードを決めるフラグの組み合わせフラグは次の通りです。
FILE_READ – ファイルが読み込みのために開かれる
FILE_WRITE – ファイルが書き込みのために開かれる
FILE_BIN – バイナリ読み書きモード（文字列から/への変換はなし）
FILE_CSV – csv ファイル（unicode または ansi 文字列への変換され区切り文字で区切られる）
FILE_TXT – 単純なテキストファイル（ csv と同じだが区切り文字がない）
FILE_ANSI – ANSI 文字列（1 バイト文字）
FILE_UNICODE – UNICODE 文字列（2 バイト文字）
FILE_SHARE_READ – 複数プログラムから読み込まれる
FILE_SHARE_WRITE – 複数プログラムから書き込まれる
FILE_COMMON – 全てのクライエント端末に共通なファイルの位置（ ¥Terminal¥Common¥Files ）

delimiter='\t'

[in] txt または csv ファイルでセパレータとして使用される値。csv ファイルのデリミタが指定されていない場合、デフォルトはタブです。txt ファイルのデリミタが指定されていない場合、セパレータは使用されません。セパレータが明らかに 0 に設定されている場合、セパレータは何も使用されません。

codepage=CP_ACP

[in] コードページの値。よく使用される コードページ では定数が提供されています。

戻り値

ファイルが正常に開かれた場合、この関数は、ファイルデータのアクセスに使用されるファイルハンドルが返されます。失敗した場合 INVALID_HANDLE を返します。

注意事項

セキュリティ上の理由から、MQL5 言語ではファイルの扱いは厳しく制御されています。MQL5 手段を用いて操作されるファイルは、ファイルサンドボックスの外に存在することは出来ません。

ファイルを特定のエンコーディング（コードページパラメータにはコードページ値を指定されます）で読み込む必要がある場合は、FILE_ANSIフラグを必ず設定してください。指定されたFILE_ANSIフラグがない場合、テキストファイルは変換なしでUnicodeで読み込まれます。

© 2000–2025, MetaQuotes Ltd.

ファイルは MQL5¥files サブフォルダ（テスト時は testing_agent_directory¥MQL5¥files ）内のクライアント端末のフォルダで開かれます。FILE_COMMON フラグが指定されている場合、ファイルは、全てのメタトレーダー5クライアント端末が使用する共有フォルダで開かれます。

「名前付きパイプ」は以下の規則に従って開くことが出来ます。

- パイプ名は文字列で「¥¥servername¥pipe¥pipename」の様な外観を持つべきです。ここで servername はネットワーク内のサーバ名でpipename はパイプ名です。パイプが同じコンピュータ上で使用されている場合はサーバ名は省略出来ますが「¥¥.¥pipe¥pipename」のように点を挿入する必要があります。パイプに接続しようとしているクライアントはその名称を知っているべきです。
- FileFlush() 及び FileSeek() は、パイプから読み込み、それへ書き込みという一連の動作の間に、ファイルの先頭に呼び出す必要があります。

特殊記号「¥」が表示される文字列で使用されています。従って MQL5 アプリケーションで名称を書く際には「¥」は二重にされるべきです。つまり上の例はコード内では "¥¥¥¥servername¥¥pipe¥¥pipename" になる必要があります。

名前付きパイプでの作業の詳細は「Communicating With MetaTrader 5 Using Named Pipes Without Using DLLs」でみられます。

例:

```
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 不正なファイルオープンメソッド
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
   string filename=terminal_data_path+"\\MQL5\\Files\\"+"fractals.csv";
   int filehandle=FileOpen(filename,FILE_WRITE|FILE_CSV);
   if(filehandle<0)
     {
      Print("Failed to open the file by the absolute path ");
      Print("Error code ",GetLastError());
     }

//--- ファイルサンドボックスでの正しい作業の仕方
   ResetLastError();
   filehandle=FileOpen("fractals.csv",FILE_WRITE|FILE_CSV);
   if(filehandle!=INVALID_HANDLE)
     {
      FileWrite(filehandle,TimeCurrent(),Symbol(), EnumToString(_Period));
      FileClose(filehandle);
      Print("FileOpen OK");
     }
   else Print("Operation FileOpen failed, error ",GetLastError());
//--- MQL5¥Files¥で囲まれたディレクトリの作成の例
   string subfolder="Research";
   filehandle=FileOpen(subfolder+"\\fractals.txt",FILE_WRITE|FILE_CSV);
      if(filehandle!=INVALID_HANDLE)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      FileWrite(filehandle,TimeCurrent(),Symbol(), EnumToString(_Period));
      FileClose(filehandle);
      Print("The file must be created in the folder "+terminal_data_path+"\
\"+subfolder);
     }
   else Print("File open failed, error ",GetLastError());
  }
```

参照

コードページの利用、FileFindFirst、FolderCreate、ファイルを開く際のフラグ

© 2000–2025, MetaQuotes Ltd.

FileClose

FileOpen() で既に開いたファイルを閉じます。

```
void  FileClose(
   int  file_handle      // ファイルハンドル
   );
```

パラメータ

file_handle

[in] FileOpen() から戻されたファイル記述子

戻り値

なし

例:

```
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- 入力パラメータ
input string InpFileName="file.txt";     // ファイル名
input string InpDirectoryName="Data";    // ディレクトリ名
input int    InpEncodingType=FILE_ANSI; // ANSI=32 または UNICODE=64
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 使用するファイルへのパスを出力する
   PrintFormat("Working %s\\Files\\ folder",TerminalInfoString(TERMINAL_DATA_PATH));
//--- エラー値をリセットする
   ResetLastError();
//--- ファイルを読み込みのために開く(ファイルが存在しない場合、エラーが発生する)
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|
InpEncodingType);
   if(file_handle!=INVALID_HANDLE)
     {
      //--- ファイルの内容を出力
      while(!FileIsEnding(file_handle))
         Print(FileReadString(file_handle));
      //--- ファイルを閉じる
      FileClose(file_handle);
     }
   else
      PrintFormat("Error, code = %d",GetLastError());
  }
```

© 2000-2025, MetaQuotes Ltd.

FileCopy

この関数は、ファイルをローカルまたは共有フォルダから別のファイルに複製します。

```
bool  FileCopy(
   const string  src_file_name,      // ソースファイル名
   int           common_flag,        // 場所
   const string  dst_file_name,      // ターゲットファイル名
   int           mode_flags          // アクセスモード
   );
```

パラメータ

src_file_name

[in] 複製するファイルの名称

common_flag

[in] ファイルの場所を決めるフラグ。common_flag = FILE_COMMON の場合、ファイルは全てのクライアント端末の共有フォルダ ¥Terminal¥Common¥Files 内に存在します。その他の場合、ファイルはローカルフォルダに存在します（例えば*common_flag=0*）。

dst_file_name

[in] 結果ファイル名

mode_flags

[in] アクセスフラグ。パラメータは FILE_REWRITE 及び/または FILE_COMMON の 2 つのフラグのみを含むことができ、他のフラグは無視されます。宇ファイルが既存し FILE_REWRITE フラグが指定されていない場合、ファイルは書き換えされず、関数は false を返します。

戻り値

失敗の場合 false

注意事項

セキュリティ上の理由から、MQL5 言語ではファイルの扱いは厳しく制御されています。MQL5 手段を用いて操作されるファイルは、ファイルサンドボックスの外に存在することは出来ません。

新しいファイルが既存する場合、コピーは mode_flags パラメータに FILE_REWRITE フラグがあるかどうかに応じて行われます。

例：

```
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- 入力パラメータ
input string InpSrc="source.txt";        // ソース
input string InpDst="destination.txt";   // コピー
input int    InpEncodingType=FILE_ANSI; // ANSI=32 または UNICODE=64
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- ソースの内容を表示する(存在が必要)
   if(!FileDisplay(InpSrc))
      return;
//--- コピーファイルが既に存在するかどうかを確認(作成されない場合がある)
   if(!FileDisplay(InpDst))
     {
      //--- コピーファイルが既に存在しないので FILE_REWRITE フラグなしで複製する(正しいやり方)
      if(FileCopy(InpSrc,0,InpDst,0))
         Print("File is copied!");
      else
         Print("File is not copied!");
     }
   else
     {
      //--- コピーファイルが既に存在するので FILE_REWRITE フラグなしでのコピーを試みる(不正)
      if(FileCopy(InpSrc,0,InpDst,0))
         Print("File is copied!");
      else
         Print("File is not copied!");
      //--- InpDst file の内容は変わらない
      FileDisplay(InpDst);
      //--- FILE_REWRITE フラグで再コピー(ファイルが既存する場合の正しいやり方)
      if(FileCopy(InpSrc,0,InpDst,FILE_REWRITE))
         Print("File is copied!");
      else
         Print("File is not copied!");
     }
//--- InpSrc ファイルコピーを受け取る
   FileDisplay(InpDst);
  }
//+------------------------------------------------------------------+
//| ファイルの内容を読む                                             |
//+------------------------------------------------------------------+
bool FileDisplay(const string file_name)
  {
//--- エラー値をリセットする
   ResetLastError();
//--- ファイルを開く
   int file_handle=FileOpen(file_name,FILE_READ|FILE_TXT|InpEncodingType);
   if(file_handle!=INVALID_HANDLE)
     {
      //--- ファイルの内容をループで表示する
      Print("+----------------------+");
      PrintFormat("File name = %s",file_name);
      while(!FileIsEnding(file_handle))
         Print(FileReadString(file_handle));
      Print("+----------------------+");
      //--- ファイルを閉じる
```

© 2000-2025, MetaQuotes Ltd.

```
      FileClose(file_handle);
      return(true);
     }
//--- ファイルを開くのに失敗
   PrintFormat("%s is not opened, error = %d",file_name,GetLastError());
   return(false);
  }
```

© 2000–2025, MetaQuotes Ltd.

FileDelete

指定されたクライアント端末のローカルフォルダ内のファイルを削除します。

```
bool  FileDelete(
   const string  file_name,      // 削除するファイルの名称
   int           common_flag=0   // 削除するファイルの場所
   );
```

パラメータ

file_name

[in] ファイル名

common_flag=0

[in] ファイルの場所を決定するフラグcommon_flag = FILE_COMMON の場合、ファイルは全てのクライアント端末の共有フォルダ ¥Terminal¥Common¥Files 内に存在します。その他の場合、ファイルはローカルフォルダに存在します。

戻り値

失敗の場合 false

注意事項

セキュリティ上の理由から、MQL5 言語ではファイルの扱いは厳しく制御されています。MQL5 手段を用いて操作されるファイルは、ファイルサンドボックスの外に存在することは出来ません。

指定されたファイルをローカルまたは共有フォルダから削除します。(MQL5¥Files またはテストの場合 MQL5¥Tester¥Files) common_flag = FILE_COMMON の場合、この関数はファイルを全てのクライアント端末の共有フォルダ \Terminal\Common\Files 内から削除します。

例:

```
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- 古いファイルの日付
input datetime InpFilesDate=D'2013.01.01 00:00';
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
   string   file_name;       // ファイル名を格納する変数
   string   filter="*.txt"; // ファイル検索フィルタ
   datetime create_date;     // ファイル作成日
   string   files[];         // ファイル名のリスト
   int      def_size=25;     // デフォルトの配列サイズ
   int      size=0;          // ファイル数
//--- 配列へのメモリ追加
   ArrayResize(files,def_size);
//--- ローカルフォルダのルートで検索ハンドルを受け取る
   long search_handle=FileFindFirst(filter,file_name);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- FileFindFirst() の実行が成功したかをチェックする
   if(search_handle!=INVALID_HANDLE)
     {
      //--- ループでファイルを検索する
      do
        {
         files[size]=file_name;
         //--- 配列サイズを増やす
         size++;
         if(size==def_size)
           {
            def_size+=25;
            ArrayResize(files,def_size);
           }
         //--- エラー値をリセットする
         ResetLastError();
         //--- 作成日を受け取る
         create_date=(datetime)FileGetInteger(file_name,FILE_CREATE_DATE,false);
         //--- ファイルが古いかをチェックする
         if(create_date<InpFilesDate)
           {
            PrintFormat("%s file deleted!",file_name);
            //--- 古いファイルを削除する
            FileDelete(file_name);
           }
        }
      while(FileFindNext(search_handle,file_name));
      //--- 検索ハンドルを閉じる
      FileFindClose(search_handle);
     }
   else
     {
      Print("Files not found!");
      return;
     }
//--- 何のファイルが残っているかをチェックする
   PrintFormat("Results:");
   for(int i=0;i<size;i++)
     {
      if(FileIsExist(files[i]))
         PrintFormat("%s file exists!",files[i]);
      else
         PrintFormat("%s file deleted!",files[i]);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

FileMove

ファイルをローカルまたは共有フォルダから別のフォルダに移動します。

```
bool  FileMove(
   const string  src_file_name,     // 移動されるファイルの名称
   int           common_flag,       // 場所
   const string  dst_file_name,     // ターゲットファイルの名称
   int           mode_flags         // アクセスモード
   );
```

パラメータ

src_file_name

[in] 移動/名称変更されるファイル名

common_flag

[in] ファイルの場所を決めるフラグ。common_flag = FILE_COMMON の場合、ファイルは全てのクライアント端末の共有フォルダ ¥Terminal¥Common¥Files 内に存在します。その他の場合、ファイルはローカルフォルダに存在します（*common_flag=0*）。

dst_file_name

[in] 操作後のファイル名

mode_flags

[in] アクセスフラグ。パラメータは FILE_REWRITE 及び/または FILE_COMMON の 2 つのフラグのみを含むことができ、他のフラグは無視されます。宇ファイルが既存し FILE_REWRITE フラグが指定されていない場合、ファイルは書き換えされず、関数は false を返します。

戻り値

失敗の場合 false

注意事項

セキュリティ上の理由から、MQL5 言語ではファイルの扱いは厳しく制御されています。MQL5 手段を用いて操作されるファイルは、ファイルサンドボックスの外に存在することは出来ません。

新しいファイルが既存する場合、コピーは mode_flags パラメータに FILE_REWRITE フラグがあるかどうかに応じて行われます。

例：

```
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- 入力パラメータ
input string InpSrcName="data.txt";
input string InpDstName="newdata.txt";
input string InpSrcDirectory="SomeFolder";
input string InpDstDirectory="OtherFolder";
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
   string local=TerminalInfoString(TERMINAL_DATA_PATH);
   string common=TerminalInfoString(TERMINAL_COMMONDATA_PATH);
//--- ファイルパスを受け取る
   string src_path;
   string dst_path;
   StringConcatenate(src_path,InpSrcDirectory,"//",InpSrcName);
   StringConcatenate(dst_path,InpDstDirectory,"//",InpDstName);
//--- ソースファイルが存在するかをチェック(存在しなければ終了する)
   if(FileIsExist(src_path))
      PrintFormat("%s file exists in the %s\\Files\\%s
folder",InpSrcName,local,InpSrcDirectory);
   else
     {
      PrintFormat("Error, %s source file not found",InpSrcName);
      return;
     }
//--- ターゲットファイルが既存するかをチェックする
   if(FileIsExist(dst_path,FILE_COMMON))
     {
      PrintFormat("%s file exists in the %s\\Files\\%s
folder",InpDstName,common,InpDstDirectory);
      //--- ファイルが存在するので移動は FILE_REWRITE フラグで行うべき
      ResetLastError();
      if(FileMove(src_path,0,dst_path,FILE_COMMON|FILE_REWRITE))
         PrintFormat("%s file moved",InpSrcName);
      else
         PrintFormat("Error! Code = %d",GetLastError());
     }
   else
     {
      PrintFormat("%s file does not exist in the %s\\Files\\%s
folder",InpDstName,common,InpDstDirectory);
      //--- ファイルが存在しないので移動は FILE_REWRITE フラグなしで行うべき
      ResetLastError();
      if(FileMove(src_path,0,dst_path,FILE_COMMON))
         PrintFormat("%s file moved",InpSrcName);
      else
         PrintFormat("Error! Code = %d",GetLastError());
     }
//--- ファイルが移動されたのでみてみる
   if(FileIsExist(dst_path,FILE_COMMON) && !FileIsExist(src_path,0))
      Print("Success!");
   else
      Print("Error!");
  }
```

参照

© 2000-2025, MetaQuotes Ltd.

FileIsExist

© 2000–2025, MetaQuotes Ltd.

FileFlush

入力/出力ファイルのバッファに残っている全てのデータをディスクに書き込みます。

```
void  FileFlush(
   int  file_handle      // ファイルハンドル
   );
```

パラメータ

file_handle

[in] FileOpen() から戻されたファイル記述子

戻り値

なし

注意事項

ファイルに書き込んだデータは実際にはすぐには見つからないかもしれません。即座にファイルにデータを保存するには FileFlush() 関数を使用します。この関数を使用しない場合、まだディスクに格納されていないデータは FileClose() 関数でファイルが閉じられた時のみに強制的に書き込まれます。

書き込まれたデータが特定の値である時にこの関数を使用する必要があります。この関数の頻繁な呼び出しは、プログラムの動作速度に影響を与える可能性があることに留意すべきです。

FileFlush () 関数はファイルからの読み込みと書き込み操作の間に呼び出される必要があります。

例:

```
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- 書き込みするファイル名
input string InpFileName="example.csv"; // ファイル名
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- エラー値をリセットする
   ResetLastError();
//--- ファイルを開く
   int file_handle=FileOpen(InpFileName,FILE_READ|FILE_WRITE|FILE_CSV);
   if(file_handle!=INVALID_HANDLE)
     {
      //--- データをファイルに書く
      for(int i=0;i<1000;i++)
        {
         //--- write 関数を呼び出す
         FileWrite(file_handle,TimeCurrent(),SymbolInfoDouble
(Symbol(),SYMBOL_BID),SymbolInfoDouble(Symbol(),SYMBOL_ASK));
         //--- 各128番目の繰り返しでディスク上のデータを保存する
         if((i & 127)==127)
```

© 2000-2025, MetaQuotes Ltd.

```
            {
             //--- データはファイルに配置され、重大なエラーの場合に失われることはない
             FileFlush(file_handle);
             PrintFormat("i = %d, OK",i);
            }
          //--- 0.01 秒の休止
          Sleep(10);
         }
       //--- ファイルを閉じる
       FileClose(file_handle);
      }
   else
      PrintFormat("Error, code = %d",GetLastError());
  }
```

参照

FileClose

© 2000-2025, MetaQuotes Ltd.

FileGetInteger

ファイルの整数プロパティを取得します。この関数には 2 つのバージョンがあります。

1. ファイルハンドルでプロパティを取得

```
long  FileGetInteger(
   int                          file_handle,    // ファイルハンドル
   ENUM_FILE_PROPERTY_INTEGER   property_id     // プロパティ識別子
   );
```

2. ファイル名でプロパティを取得

```
long  FileGetInteger(
   const string                 file_name,              // ファイル名
   ENUM_FILE_PROPERTY_INTEGER   property_id,            // プロパティ ID
   bool                         common_folder=false     // ファイルがローカルフォルダでみられる
(false)
   );                                                   // または全ての端末の共有フォルダでみられ
る(true)
```

パラメータ

file_handle

[in] FileOpen() から戻されたファイル記述子

file_name

[in] ファイル名

property_id

[in] ファイルプロパティ識別子値は ENUM_FILE_PROPERTY_INTEGER 列挙のいずれかです。関数の2番目のバージョンが使用されている場合は以下のプロパティ値のみを受け取ることが出来ます。FILE_EXISTS、FILE_CREATE_DATE、FILE_MODIFY_DATE、FILE_ACCESS_DATE 及び FILE_SIZE。

common_folder=false

[in] ファイルの位置にポイントするパラメータが false の場合、端末データフォルダがみられます。さもなければ、ファイルは全てのクライアント端末の共有フォルダ ¥Terminal¥Common¥Files （FILE_COMMON）に位置すると想定されます。

戻り値

プロパティ値（失敗した場合は -1 ）。エラーコードを受け取るには GetLastError() 関数を使用します。

名称でプロパティを取得する際にフォルダを指定された場合、この関数は、戻り値は正しくありながらもエラー5018（ERR_MQL_FILE_IS_DIRECTORY）を発します。

注意事項

この関数は常にエラーコードを変更します。正常に完了した場合はエラーコードが NULL にリセットされます。

例：

```
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 入力パラメータ
input string InpFileName="data.csv";
input string InpDirectoryName="SomeFolder";
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
   string path=InpDirectoryName+"//"+InpFileName;
   long   l=0;
//--- ファイルを開く
   ResetLastError();
   int handle=FileOpen(path,FILE_READ|FILE_CSV);
   if(handle!=INVALID_HANDLE)
     {
      //--- ファイルの情報を全て出力する
      Print(InpFileName," file info:");
      FileInfo(handle,FILE_EXISTS,l,"bool");
      FileInfo(handle,FILE_CREATE_DATE,l,"date");
      FileInfo(handle,FILE_MODIFY_DATE,l,"date");
      FileInfo(handle,FILE_ACCESS_DATE,l,"date");
      FileInfo(handle,FILE_SIZE,l,"other");
      FileInfo(handle,FILE_POSITION,l,"other");
      FileInfo(handle,FILE_END,l,"bool");
      FileInfo(handle,FILE_IS_COMMON,l,"bool");
      FileInfo(handle,FILE_IS_TEXT,l,"bool");
      FileInfo(handle,FILE_IS_BINARY,l,"bool");
      FileInfo(handle,FILE_IS_CSV,l,"bool");
      FileInfo(handle,FILE_IS_ANSI,l,"bool");
      FileInfo(handle,FILE_IS_READABLE,l,"bool");
      FileInfo(handle,FILE_IS_WRITABLE,l,"bool");
      //--- ファイルを閉じる
      FileClose(handle);
     }
   else
      PrintFormat("%s file is not opened, ErrorCode = %
d",InpFileName,GetLastError());
  }
//+------------------------------------------------------------------+
//|  ファイルプロパティ値を表示する                                    |
//+------------------------------------------------------------------+
void FileInfo(const int handle,const ENUM_FILE_PROPERTY_INTEGER id,
              long l,const string type)
  {
//--- プロパティ値を受け取る
   ResetLastError();
   if((l=FileGetInteger(handle,id))!=-1)
     {
      //--- 値を受け取ったので、正しいフォーマットで表示する
```

© 2000-2025, MetaQuotes Ltd.

```
      if(!StringCompare(type,"bool"))
         Print(EnumToString(id)," = ",l ? "true" : "false");
      if(!StringCompare(type,"date"))
         Print(EnumToString(id)," = ",(datetime)l);
      if(!StringCompare(type,"other"))
         Print(EnumToString(id)," = ",l);
     }
   else
      Print("Error, Code = ",GetLastError());
  }
```

参照

ファイル操作、ファイルプロパティ

© 2000–2025, MetaQuotes Ltd.

FileIsEnding

読み出し処理において、ファイルの終わりを定義します。

```
bool  FileIsEnding(
   int  file_handle      // ファイルハンドル
   );
```

パラメータ

file_handle

[in] FileOpen() から戻されたファイル記述子

戻り値

読み出し処理またはファイルポインタの移動時において、ファイルの終わりに達した場合は true

注意事項

ファイルの終わりを定義するには、関数は次の文字列を読み取ろうとします。文字列が存在しない場合、この関数は true を戻し、それ以外の場合は false を返します。

例:

```
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- 入力パラメータ
input string InpFileName="file.txt";     // ファイル名
input string InpDirectoryName="Data";    // ディレクトリ名
input int    InpEncodingType=FILE_ANSI; // ANSI=32 または UNICODE=64
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 使用するファイルへのパスを出力する
   PrintFormat("Working %s\\Files\\ folder",TerminalInfoString(TERMINAL_DATA_PATH));
//--- エラー値をリセットする
   ResetLastError();
//--- ファイルを読み込みのために開く(ファイルが存在しない場合、エラーが発生する)
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|
InpEncodingType);
   if(file_handle!=INVALID_HANDLE)
     {
      //--- ファイルの内容を出力
      while(!FileIsEnding(file_handle))
         Print(FileReadString(file_handle));
      //--- ファイルを閉じる
      FileClose(file_handle);
     }
   else
      PrintFormat("Error, code = %d",GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
   }
```

© 2000–2025, MetaQuotes Ltd.

FileIsLineEnding

読み出し処理において、テキストファイルの行の終わりを定義します。

```
bool  FileIsLineEnding(
   int  file_handle      // ファイルハンドル
   );
```

パラメータ

file_handle

[in] FileOpen() から戻されたファイル記述子

戻り値

txt または csv ファイルが読み出し処理において行の終わり(CR-LF) に達したなら true を返します。

例 （ FileWriteString 関数の例の実行によって取得されたファイルが使用されます）

```
//+------------------------------------------------------------------+
//|                                        Demo_FileIsLineEnding.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                             https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   1
//---- Label 1 をプロットする
#property indicator_label1  "Overbought & Oversold"
#property indicator_type1   DRAW_COLOR_BARS
#property indicator_color1  clrRed, clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- データ読み込みのパラメータ
input string InpFileName="RSI.csv";   // ファイル名
input string InpDirectoryName="Data"; // ディレクトリ名
//--- 指標バッファ
double   open_buff[];
double   high_buff[];
double   low_buff[];
double   close_buff[];
double   color_buff[];
//--- 買いすぎ変数
int      ovb_ind=0;
int      ovb_size=0;
datetime ovb_time[];
//--- 売りすぎ変数
int      ovs_ind=0;
```

© 2000-2025, MetaQuotes Ltd.

```
int      ovs_size=0;
datetime ovs_time[];
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- デフォルトの配列サイズの変数
   int ovb_def_size=100;
   int ovs_def_size=100;
//--- 変数にメモリを割り当てる
   ArrayResize(ovb_time,ovb_def_size);
   ArrayResize(ovs_time,ovs_def_size);
//--- ファイルを開く
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV|
FILE_ANSI);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s file is available for reading",InpFileName);
      PrintFormat("File path: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      double value;
      //--- ファイルからデータを読む
      while(!FileIsEnding(file_handle))
        {
         //--- 文字列の初めの値を読む
         value=FileReadNumber(file_handle);
         //--- 関数の結果に応じて異なる配列を読む
         if(value>=70)
            ReadData(file_handle,ovb_time,ovb_size,ovb_def_size);
         else
            ReadData(file_handle,ovs_time,ovs_size,ovs_def_size);
        }
      //--- ファイルを閉じる
      FileClose(file_handle);
      PrintFormat("Data is written, %s file is closed",InpFileName);
     }
   else
     {
      PrintFormat("Failed to open %s file, Error code = %
d",InpFileName,GetLastError());
      return(INIT_FAILED);
     }
//--- 配列の結合
   SetIndexBuffer(0,open_buff,INDICATOR_DATA);
   SetIndexBuffer(1,high_buff,INDICATOR_DATA);
   SetIndexBuffer(2,low_buff,INDICATOR_DATA);
   SetIndexBuffer(3,close_buff,INDICATOR_DATA);
   SetIndexBuffer(4,color_buff,INDICATOR_COLOR_INDEX);
```

© 2000–2025, MetaQuotes Ltd.

```
//---- チャートでは表示されない指標値を設定する
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| ファイルの文字列データを読む                                     |
//+------------------------------------------------------------------+
void ReadData(const int file_handle,datetime &arr[],int &size,int &def_size)
  {
   bool flag=false;
//--- 文字列の終わりかファイルの終わりまで読む
   while(!FileIsLineEnding(file_handle) && !FileIsEnding(file_handle))
     {
      //--- 数字を読んで位置をシフトする
      if(flag)
         FileReadNumber(file_handle);
      //--- 現在の日付を格納する
      arr[size]=FileReadDatetime(file_handle);
      size++;
      //--- 必要な場合配列サイズを増やす
      if(size==def_size)
        {
         def_size+=100;
         ArrayResize(arr,def_size);
        }
      //--- 最初の反復をすり抜ける
      flag=true;
     }
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(close,false);
```

© 2000–2025, MetaQuotes Ltd.

```
//--- まだ処理されてないバーのループ
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- デフォルトでは 0
      open_buff[i]=0;
      high_buff[i]=0;
      low_buff[i]=0;
      close_buff[i]=0;
      color_buff[i]=0;
      //--- まだデータがあるかをチェック
      if(ovb_ind<ovb_size)
         for(int j=ovb_ind;j<ovb_size;j++)
           {
            //--- 日付が一致した場合、バーが買われ過ぎ領域にある
            if(time[i]==ovb_time[j])
              {
               open_buff[i]=open[i];
               high_buff[i]=high[i];
               low_buff[i]=low[i];
               close_buff[i]=close[i];
               //--- 0 - 赤
               color_buff[i]=0;
               //--- カウンタを増加する
               ovb_ind=j+1;
               break;
              }
           }
      //--- データがまだあるかをチェック
      if(ovs_ind<ovs_size)
         for(int j=ovs_ind;j<ovs_size;j++)
           {
            //--- 日付が一致した場合、バーが売られ過ぎ領域にある
            if(time[i]==ovs_time[j])
              {
               open_buff[i]=open[i];
               high_buff[i]=high[i];
               low_buff[i]=low[i];
               close_buff[i]=close[i];
               //--- 1 - 青
               color_buff[i]=1;
               //--- カウンタを増加する
               ovs_ind=j+1;
               break;
              }
           }
     }
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  ChartEvent イベントハンドラ                                     |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam
                 )
  {
//--- スケールに合わせて指標の幅を変更する
   if(ChartGetInteger(0,CHART_SCALE)>3)
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,2);
   else
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,1);
  }
```

参照

FileWriteString

© 2000-2025, MetaQuotes Ltd.

FileReadArray

string 以外の任意型のBIN 型配列（文字列を含まない構造体の配列、動的配列）のファイルから読み込みを行います。

```
uint  FileReadArray(
   int     file_handle,              // ファイルハンドル
   void&   array[],                  // 記録するための配列
   int     start=0,                  // 配列の書き初めの位置
   int     count=WHOLE_ARRAY         // 読みのカウント
   );
```

パラメータ

file_handle

[in] FileOpen() から戻されたファイル記述子

array[]

[out] データが読み込まれる配列

start=0

[in] 配列に書く開始位置

count=WHOLE_ARRAY

[in] 読まれる要素数。デフォルトでは配列全体が読まれます（count=WHOLE_ARRAY）。

戻り値

読まれた要素数

注意事項

文字列の配列は TXT 形式のファイルからのみ読み込むことが出来ます。この関数は、必要であれば配列のサイズを増やそうとします。

例（FileWriteArray 関数の例の実行によって取得されたファイルが使用されます）

```
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- 入力パラメータ
input string InpFileName="data.bin";
input string InpDirectoryName="SomeFolder";
//+------------------------------------------------------------------+
//|  価格データを格納する構造体                                      |
//+------------------------------------------------------------------+
struct prices
  {
   datetime          date; // 日付
   double            bid;  // 売値
   double            ask;  // 買値
  };
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                              |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 構造体配列
   prices arr[];
//--- ファイルパス
   string path=InpDirectoryName+"//"+InpFileName;
//--- ファイルを開く
   ResetLastError();
   int file_handle=FileOpen(path,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      //--- ファイルのデータを全部配列に読見込む
      FileReadArray(file_handle,arr);
      //--- 配列サイズを受け取る
      int size=ArraySize(arr);
      //--- 配列からデータを出力する
      for(int i=0;i<size;i++)
         Print("Date = ",arr[i].date," Bid = ",arr[i].bid," Ask = ",arr[i].ask);
      Print("Total data = ",size);
      //--- ファイルを閉じる
      FileClose(file_handle);
     }
   else
      Print("File open failed, error ",GetLastError());
  }
```

参照

変数、FileWriteArray

© 2000-2025, MetaQuotes Ltd.

FileReadBool

CSV 形式のファイルの現在位置から区切り文字までの文字列（またはテキスト行の終わり）を読み込み、読んだ文字列を bool 型の値に変換します。

```
bool  FileReadBool(
   int  file_handle    // ファイルハンドル
   );
```

パラメータ

file_handle

[in] FileOpen() から戻されたファイル記述子

戻り値

読み取られた行は「true」「false」または整数の「0」「1」にも設定出来ます。ゼロ以外の値は、論理 true に変換されます。この関数は変換された値を返します。

例（FileWrite 関数の例の実行によって取得されたファイルが使用されます）

```
//+------------------------------------------------------------------+
//|                                            Demo_FileReadBool.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//---- Label 1 をプロットする
#property indicator_label1  "UpSignal"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  4
//---- Label 2 をプロットする
#property indicator_label2  "DownSignal"
#property indicator_type2   DRAW_ARROW
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  4
//--- データ読み込みのパラメータ
input string InpFileName="MACD.csv";  // ファイル名
input string InpDirectoryName="Data"; // ディレクトリ名
//--- グローバル変数
int       ind=0;        // インデックス
double    upbuff[];     // 上向き矢印の指標バッファ
double    downbuff[];   // 下向き矢印の指標バッファ
```

© 2000–2025, MetaQuotes Ltd.

```
bool     sign_buff[]; // シグナル配列(true - 買、false - 売)
datetime time_buff[]; // シグナル到着時の配列
int      size=0;      // シグナル配列のサイズ
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- ファイルを開く
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s file is open for reading",InpFileName);
      //--- 初めにシグナルの数を読む
      size=(int)FileReadNumber(file_handle);
      //--- 配列へメモリを割り当てる
      ArrayResize(sign_buff,size);
      ArrayResize(time_buff,size);
      //--- ファイルからデータを読む
      for(int i=0;i<size;i++)
        {
         //--- 時間シグナル
         time_buff[i]=FileReadDatetime(file_handle);
         //--- 値シグナル
         sign_buff[i]=FileReadBool(file_handle);
        }
      //--- ファイルを閉じる
      FileClose(file_handle);
     }
   else
     {
      PrintFormat("Failed to open %s file, Error code = %
d",InpFileName,GetLastError());
      return(INIT_FAILED);
     }
//--- 配列の結合
   SetIndexBuffer(0,upbuff,INDICATOR_DATA);
   SetIndexBuffer(1,downbuff,INDICATOR_DATA);
//--- PLOT_ARROW での描画のためにシンボルコードを設定
   PlotIndexSetInteger(0,PLOT_ARROW,241);
   PlotIndexSetInteger(1,PLOT_ARROW,242);
//---- チャートでは表示されない指標値を設定
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
   PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|  カスタム指標の反復関数                                                  |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(high,false);
//--- まだ処理されてないバーのループ
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- デフォルトでは 0
      upbuff[i]=0;
      downbuff[i]=0;
      //--- まだデータがあるかをチェック
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- 日付が同じならファイルの値を使用する
            if(time[i]==time_buff[j])
              {
               //--- シグナルに合わせて矢印を描画する
               if(sign_buff[j])
                  upbuff[i]=high[i];
               else
                  downbuff[i]=low[i];
               //--- カウンタを増加する
               ind=j+1;
               break;
              }
           }
        }
     }
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
```

参照

ブール型、FileWrite

© 2000-2025, MetaQuotes Ltd.

FileReadDatetime

CSV 形式のファイルから、「YYYY.MM.DD HH:MM:SS」、「YYYY.MM.DD」または「HH:MM:SS」フォーマットの文字列を読み込みdatetime 型の値に変換します。

```
datetime  FileReadDatetime(
   int  file_handle    // ファイルハンドル
   );
```

パラメータ

file_handle

[in] FileOpen() から戻されたファイル記述子

戻り値

datetime 型の値

例（FileWrite 関数の例の実行によって取得されたファイルが使用されます）

```
//+------------------------------------------------------------------+
//|                                        Demo_FileReadDateTime.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                             https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//---- Label 1 をプロットする
#property indicator_label1  "UpSignal"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  4
//---- Label 2 をプロットする
#property indicator_label2  "DownSignal"
#property indicator_type2   DRAW_ARROW
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  4
//--- データ読み込みのパラメータ
input string InpFileName="MACD.csv";  // ファイル名
input string InpDirectoryName="Data"; // ディレクトリ名
//--- グローバル変数
int       ind=0;        // インデックス
double    upbuff[];     // 上向き矢印の指標バッファ
double    downbuff[];   // 下向き矢印の指標バッファ
bool      sign_buff[]; // シグナル配列(true - 買、false - 売)
```

© 2000-2025, MetaQuotes Ltd.

```
datetime time_buff[]; // シグナル到着時の配列
int      size=0;      // シグナル配列のサイズ
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- ファイルを開く
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s file is open for reading",InpFileName);
      //--- 初めにシグナルの数を読む
      size=(int)FileReadNumber(file_handle);
      //--- 配列へメモリを割り当てる
      ArrayResize(sign_buff,size);
      ArrayResize(time_buff,size);
      //--- ファイルからデータを読む
      for(int i=0;i<size;i++)
        {
         //--- 時間シグナル
         time_buff[i]=FileReadDatetime(file_handle);
         //--- 値シグナル
         sign_buff[i]=FileReadBool(file_handle);
        }
      //--- ファイルを閉じる
      FileClose(file_handle);
     }
   else
     {
      PrintFormat("Failed to open %s file, Error code = %
d",InpFileName,GetLastError());
      return(INIT_FAILED);
     }
//--- 配列の結合
   SetIndexBuffer(0,upbuff,INDICATOR_DATA);
   SetIndexBuffer(1,downbuff,INDICATOR_DATA);
//--- PLOT_ARROW での描画のためにシンボルコードを設定
   PlotIndexSetInteger(0,PLOT_ARROW,241);
   PlotIndexSetInteger(1,PLOT_ARROW,242);
//---- チャートでは表示されない指標値を設定
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
   PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(high,false);
//--- まだ処理されてないバーのループ
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- デフォルトでは 0
      upbuff[i]=0;
      downbuff[i]=0;
      //--- まだデータがあるかをチェック
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- 日付が同じならファイルの値を使用する
            if(time[i]==time_buff[j])
              {
               //--- シグナルに合わせて矢印を描画する
               if(sign_buff[j])
                  upbuff[i]=high[i];
               else
                  downbuff[i]=low[i];
               //--- カウンタを増加する
               ind=j+1;
               break;
              }
           }
        }
     }
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
```

参照

日付時刻型、StringToTime、TimeToString、FileWrite

© 2000-2025, MetaQuotes Ltd.

FileReadDouble

バイナリファイルの現在位置から倍精度浮動小数点数（double）を読み込みます。

```
double  FileReadDouble(
   int  file_handle    // ファイルハンドル
   );
```

パラメータ

file_handle

[in] FileOpen() から戻されたファイル記述子

戻り値

double 型の値

注意事項

エラーの詳細については GetLastError()を呼びます。

例（FileWriteDouble 関数の例の実行によって取得されたファイルが使用されます）

```
//+------------------------------------------------------------------+
//|                                          Demo_FileReadDouble.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- Label 1 をプロットする
#property indicator_label1  "MA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
#property indicator_separate_window
//--- データ読みのパラメータ
input string InpFileName="MA.csv";     // ファイル名
input string InpDirectoryName="Data"; // ディレクトリ名
//--- グローバル変数
int      ind=0;
int      size=0;
double   ma_buff[];
datetime time_buff[];
//--- 指標バッファ
double   buff[];
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|  カスタム指標を初期化する関数                                                |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- ファイルを開く
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s file is available for reading",InpFileName);
      PrintFormat("File path: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- 初めに、ファイルのデータ量を読み込む
      size=(int)FileReadDouble(file_handle);
      //--- 配列へメモリを割り当てる
      ArrayResize(ma_buff,size);
      ArrayResize(time_buff,size);
      //--- ファイルからデータを読む
      for(int i=0;i<size;i++)
        {
         time_buff[i]=(datetime)FileReadDouble(file_handle);
         ma_buff[i]=FileReadDouble(file_handle);
        }
      //--- ファイルを閉じる
      FileClose(file_handle);
      PrintFormat("Data is written, %s file is closed",InpFileName);
     }
   else
     {
      PrintFormat("Failed to open %s file, Error code = %
d",InpFileName,GetLastError());
      return(INIT_FAILED);
     }
//--- 配列をインデックス0で指標バッファと結合する
   SetIndexBuffer(0,buff,INDICATOR_DATA);
//---- チャートでは表示されない指標値を設定する
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//|  カスタム指標の反復関数                                                |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
```

© 2000–2025, MetaQuotes Ltd.

```
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
//--- まだ処理されてないバーのループ
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- デフォルトでは 0
      buff[i]=0;
      //--- データがまだあるかをチェック
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- 日付が同じならファイルの値を使用する
            if(time[i]==time_buff[j])
              {
               buff[i]=ma_buff[j];
               //--- カウンタを増加する
               ind=j+1;
               break;
              }
           }
        }
     }
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
```

参照

浮動小数点数型(ダブル、フロート)、StringToDouble、DoubleToString、FileWriteDouble

© 2000-2025, MetaQuotes Ltd.

FileReadFloat

バイナリファイルの現在位置から単精度浮動小数点数（float）を読み込みます。

```
float  FileReadFloat(
   int  file_handle    // ファイルハンドル
   );
```

パラメータ

file_handle

[in] FileOpen() から戻されたファイル記述子

戻り値

float 型の値

注意事項

エラーの詳細については GetLastError()を呼びます。

例（FileWriteFloat 関数の例の実行によって取得されたファイルが使用されます）

```
//+------------------------------------------------------------------+
//|                                            Demo_FileReadFloat.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//---- Label 1 をプロットする
#property indicator_label1  "CloseLine"
#property indicator_type1   DRAW_COLOR_LINE
#property indicator_color1  clrRed,clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- データ読み込みのパラメータ
input string InpFileName="Close.bin"; // file name
input string InpDirectoryName="Data"; // ディレクトリ名
//--- グローバル変数
int      ind=0;
int      size=0;
double   close_buff[];
datetime time_buff[];
//--- 指標バッファ
double   buff[];
double   color_buff[];
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|  カスタム指標を初期化する関数                                                |
//+------------------------------------------------------------------+
int OnInit()
  {
   int def_size=100;
//--- 配列へメモリを割り当てる
   ArrayResize(close_buff,def_size);
   ArrayResize(time_buff,def_size);
//--- ファイルを開く
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s file is available for reading",InpFileName);
      PrintFormat("File path: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- ファイルからデータを読む
      while(!FileIsEnding(file_handle))
        {
         //--- 時間と価格値を読む
         time_buff[size]=(datetime)FileReadDouble(file_handle);
         close_buff[size]=(double)FileReadFloat(file_handle);
         size++;
         //--- オーバーフローしている場合は、配列のサイズを増やす
         if(size==def_size)
           {
            def_size+=100;
            ArrayResize(close_buff,def_size);
            ArrayResize(time_buff,def_size);
           }
        }
      //--- ファイルを閉じる
      FileClose(file_handle);
      PrintFormat("Data is read, %s file is closed",InpFileName);
     }
   else
     {
      PrintFormat("Failed to open %s file, Error code = %
d",InpFileName,GetLastError());
      return(INIT_FAILED);
     }
//--- 配列と指標バッファを関連付ける
   SetIndexBuffer(0,buff,INDICATOR_DATA);
   SetIndexBuffer(1,color_buff,INDICATOR_COLOR_INDEX);
//---- チャートでは表示されない指標値を設定する
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|  カスタム指標の反復関数                                                  |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
//--- まだ処理されてないバーのループ
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- デフォルトでは 0
      buff[i]=0;
      color_buff[i]=0; // デフォルトでは赤
      //--- まだデータがあるかをチェック
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- 日付が同じならファイルの値を使用する
            if(time[i]==time_buff[j])
              {
               //--- 価格を受け取る
               buff[i]=close_buff[j];
               //--- 現在の価格は以前の価格を超えた場合、色は青
               if(buff[i-1]>buff[i])
                  color_buff[i]=1;
               //--- カウンタを増加する
               ind=j+1;
               break;
              }
           }
        }
     }
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
```

参照

浮動小数点数型(ダブル、フロート)、FileReadDouble、FileWriteFloat

© 2000-2025, MetaQuotes Ltd.

FileReadInteger

この関数は、ファイルポインタの現在位置から、バイト単位で指定された長差に従って int、short または char 値を読み込みます。

```
int  FileReadInteger(
   int  file_handle,          // ファイルハンドル
   int  size=INT_VALUE        // バイト単位での整数のサイズ
   );
```

パラメータ

file_handle

[in] FileOpen() から戻されたファイル記述子

size=INT_VALUE

[in] 読まれるべきバイト数（4 以下）。関数が char、short または int 型の値を読めるように、CHAR_VALUE=1、SHORT_VALUE=2 t及び INT_VALUE=4 と対応する定数が用意されています。

戻り値

int 型の値この関数の結果は、読まれるために必要なデータの型であるターゲットの型に明確的にキャストする必要があります。int 型の値が返されるので、任意の整数値に変換することは容易です。ファイルポインタは、読み込まれたバイト数によってシフトされます。

注意事項

4 バイト未満が読み取られた場合、結果は常に正です。1 または 2 バイトが読み取られた場合、数値の符号は、char（1 バイト）又は short（2 バイト）に明示的にキャストすることによって決定出来ます。基になる型が存在しないため、3バイトの数の記号を取得することは簡単ではありません。

例（FileWriteInteger 関数の例の実行によって取得されたファイルが使用されます）

```
//+------------------------------------------------------------------+
//|                                        Demo_FileReadInteger.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- Label 1 をプロットする
#property indicator_label1  "Trends"
#property indicator_type1   DRAW_SECTION
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- データ読み込みのパラメータ
input string InpFileName="Trend.bin"; // ファイル名
```

© 2000-2025, MetaQuotes Ltd.

```
input string InpDirectoryName="Data"; // ディレクトリ名
//--- グローバル変数
int      ind=0;
int      size=0;
datetime time_buff[];
//--- 指標バッファ
double   buff[];
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
   int def_size=100;
//--- 配列へのメモリ追加
   ArrayResize(time_buff,def_size);
//--- ファイルを開く
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s file is available for reading",InpFileName);
      PrintFormat("File path: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- 追加の変数
      int    arr_size;
      uchar  arr[];
      //--- ファイルからデータを読む
      while(!FileIsEnding(file_handle))
        {
         //--- 時間を書くのに使用されるシンボルの数を見つける
         arr_size=FileReadInteger(file_handle,INT_VALUE);
         ArrayResize(arr,arr_size);
         for(int i=0;i<arr_size;i++)
            arr[i]=(char)FileReadInteger(file_handle,CHAR_VALUE);
         //--- 時間値を格納する
         time_buff[size]=StringToTime(CharArrayToString(arr));
         size++;
         //--- オーバーフローしている場合は、配列のサイズを増やす
         if(size==def_size)
           {
            def_size+=100;
            ArrayResize(time_buff,def_size);
           }
        }
      //--- ファイルを閉じる
      FileClose(file_handle);
      PrintFormat("Data is read, %s file is closed",InpFileName);
     }
   else
     {
```

© 2000–2025, MetaQuotes Ltd.

```
      PrintFormat("Failed to open %s file, Error code = %
d",InpFileName,GetLastError());
      return(INIT_FAILED);
     }
//--- 配列と指標バッファを関連付ける
   SetIndexBuffer(0,buff,INDICATOR_DATA);
//---- チャートでは表示されない指標値を設定する
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(close,false);
//--- まだ処理されてないバーのループ
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- デフォルトでは 0
      buff[i]=0;
      //--- まだデータがあるかをチェック
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- 日付が同じならファイルの値を使用する
            if(time[i]==time_buff[j])
              {
               //--- 価格を受け取る
               buff[i]=close[i];
               //--- カウンタを増加する
               ind=j+1;
               break;
              }
           }
        }
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
```

参照

IntegerToString、StringToInteger、整数型、FileWriteInteger

© 2000-2025, MetaQuotes Ltd.

FileReadLong

この関数は、バイナリファイルのファイルポインタの現在位置から long 型（8 バイト）の値を読み込みます。

```
long  FileReadLong(
   int  file_handle    // ファイルハンドル
   );
```

パラメータ

file_handle

[in] FileOpen() から戻されたファイル記述子

戻り値

long 型の値

例（FileWriteLong 関数の例の実行によって取得されたファイルが使用されます）

```
//+------------------------------------------------------------------+
//|                                            Demo_FileReadLong.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                             https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_buffers 1
#property indicator_plots   1
//---- Label 1 をプロットする
#property indicator_label1  "Volume"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrYellow
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
#property indicator_separate_window
//--- データ読み込みのパラメータ
input string InpFileName="Volume.bin"; // ファイル名
input string InpDirectoryName="Data";  // ディレクトリ名
//--- グローバル変数
int      ind=0;
int      size=0;
long     volume_buff[];
datetime time_buff[];
//--- 指標バッファ
double   buff[];
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
```

© 2000–2025, MetaQuotes Ltd.

```
//--- ファイルを開く
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s file is open for writing",InpFileName);
      PrintFormat("File path: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- 初めに、ファイルのデータ量を読み込む
      size=(int)FileReadLong(file_handle);
      //--- 配列へメモリを割り当てる
      ArrayResize(volume_buff,size);
      ArrayResize(time_buff,size);
      //--- ファイルからデータを読む
      for(int i=0;i<size;i++)
        {
         time_buff[i]=(datetime)FileReadLong(file_handle);
         volume_buff[i]=FileReadLong(file_handle);
        }
      //--- ファイルを閉じる
      FileClose(file_handle);
      PrintFormat("Data is read, %s file is closed",InpFileName);
     }
   else
     {
      PrintFormat("Failed to open %s file, Error code = %
d",InpFileName,GetLastError());
      return(INIT_FAILED);
     }
//--- 配列をインデックス0で指標バッファと結合するx
   SetIndexBuffer(0,buff,INDICATOR_DATA);
//---- チャートで表示される指標値を設定する
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   ArraySetAsSeries(time,false);
//--- まだ処理されてないバーのループ
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- デフォルトでは 0
      buff[i]=0;
      //--- まだデータがあるかをチェック
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- 日付が同じならファイルの値を使用する
            if(time[i]==time_buff[j])
              {
               buff[i]=(double)volume_buff[j];
               ind=j+1;
               break;
              }
           }
        }
     }
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
```

参照

整数型、FileReadInteger、FileWriteLong

© 2000-2025, MetaQuotes Ltd.

FileReadNumber

CSV ファイルから、現在の位置から区切り文字までの文字列（またはテキスト行の終わり）を読み込み、読んだ文字列をdouble 型の値に変換します。

```
double  FileReadNumber(
   int  file_handle    // ファイルハンドル
   );
```

パラメータ

file_handle

[in] FileOpen() から戻されたファイル記述子

戻り値

double 型の値

例（FileWriteString 関数の例の実行によって取得されたファイルが使用されます）

```
//+------------------------------------------------------------------+
//|                                          Demo_FileReadNumber.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   1
//---- Label 1 をプロットする
#property indicator_label1  "Overbought & Oversold"
#property indicator_type1   DRAW_COLOR_BARS
#property indicator_color1  clrRed, clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- データ読み込みのパラメータ
input string InpFileName="RSI.csv";   // ファイル名
input string InpDirectoryName="Data"; // ディレクトリ名
//--- 指標バッファ
double   open_buff[];
double   high_buff[];
double   low_buff[];
double   close_buff[];
double   color_buff[];
//--- 買いすぎ変数
int      ovb_ind=0;
int      ovb_size=0;
datetime ovb_time[];
//--- 売りすぎ変数
```

© 2000–2025, MetaQuotes Ltd.

```
int      ovs_ind=0;
int      ovs_size=0;
datetime ovs_time[];
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- デフォルトの配列サイズの変数
   int ovb_def_size=100;
   int ovs_def_size=100;
//--- 変数にメモリを割り当てる
   ArrayResize(ovb_time,ovb_def_size);
   ArrayResize(ovs_time,ovs_def_size);
//--- ファイルを開く
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV|
FILE_ANSI);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s file is available for reading",InpFileName);
      PrintFormat("File path: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      double value;
      //--- ファイルからデータを読む
      while(!FileIsEnding(file_handle))
        {
         //--- 文字列の初めの値を読む
         value=FileReadNumber(file_handle);
         //--- 関数の結果に応じて異なる配列を読む
         if(value>=70)
            ReadData(file_handle,ovb_time,ovb_size,ovb_def_size);
         else
            ReadData(file_handle,ovs_time,ovs_size,ovs_def_size);
        }
      //--- ファイルを閉じる
      FileClose(file_handle);
      PrintFormat("Data is written, %s file is closed",InpFileName);
     }
   else
     {
      PrintFormat("Failed to open %s file, Error code = %
d",InpFileName,GetLastError());
      return(INIT_FAILED);
     }
//--- 配列の結合
   SetIndexBuffer(0,open_buff,INDICATOR_DATA);
   SetIndexBuffer(1,high_buff,INDICATOR_DATA);
   SetIndexBuffer(2,low_buff,INDICATOR_DATA);
   SetIndexBuffer(3,close_buff,INDICATOR_DATA);
```

© 2000–2025, MetaQuotes Ltd.

```
   SetIndexBuffer(4,color_buff,INDICATOR_COLOR_INDEX);
//---- チャートでは表示されない指標値を設定する
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| ファイルの文字列データを読む                                     |
//+------------------------------------------------------------------+
void ReadData(const int file_handle,datetime &arr[],int &size,int &def_size)
  {
   bool flag=false;
//--- 文字列の終わりかファイルの終わりまで読む
   while(!FileIsLineEnding(file_handle) && !FileIsEnding(file_handle))
     {
      //--- 数を読んだ後にキャリッジをシフトする
      if(flag)
         FileReadNumber(file_handle);
      //--- 現在の日付を格納する
      arr[size]=FileReadDatetime(file_handle);
      size++;
      //--- 必要な場合配列サイズを増やす
      if(size==def_size)
        {
         def_size+=100;
         ArrayResize(arr,def_size);
        }
      //--- 最初の反復をすり抜ける
      flag=true;
     }
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
```

© 2000–2025, MetaQuotes Ltd.

```
   ArraySetAsSeries(close,false);
//--- まだ処理されてないバーのループ
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- デフォルトでは 0
      open_buff[i]=0;
      high_buff[i]=0;
      low_buff[i]=0;
      close_buff[i]=0;
      color_buff[i]=0;
      //--- まだデータがあるかをチェック
      if(ovb_ind<ovb_size)
         for(int j=ovb_ind;j<ovb_size;j++)
           {
            //--- 日付が一致した場合、バーが買われ過ぎ領域にある
            if(time[i]==ovb_time[j])
              {
               open_buff[i]=open[i];
               high_buff[i]=high[i];
               low_buff[i]=low[i];
               close_buff[i]=close[i];
               //--- 0 - 赤
               color_buff[i]=0;
               //--- カウンタを増加する
               ovb_ind=j+1;
               break;
              }
           }
      //--- データがまだあるかをチェック
      if(ovs_ind<ovs_size)
         for(int j=ovs_ind;j<ovs_size;j++)
           {
            //--- 日付が一致した場合、バーが売られ過ぎ領域にある
            if(time[i]==ovs_time[j])
              {
               open_buff[i]=open[i];
               high_buff[i]=high[i];
               low_buff[i]=low[i];
               close_buff[i]=close[i];
               //--- 1 - 青
               color_buff[i]=1;
               //--- カウンタを増加する
               ovs_ind=j+1;
               break;
              }
           }
     }
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| ChartEvent イベントハンドラ                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam
                 )
  {
//--- スケールに合わせて指標の幅を変更する
   if(ChartGetInteger(0,CHART_SCALE)>3)
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,2);
   else
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,1);
  }
```

参照

FileWriteString

© 2000-2025, MetaQuotes Ltd.

FileReadString

この関数は、ファイルポインタの現在位置から文字列を読み込みます。

```
string  FileReadString(
   int  file_handle,     // ファイルハンドル
   int  length=-1        // 文字列の長さ
   );
```

パラメータ

file_handle

[in] FileOpen() から戻されたファイル記述子

length=-1

[in] 読まれる文字数

戻り値

読まれた行（string）

注意事項

バイナリファイルからの読み込みでは読まれる文字列の長さが指定されるべきです。TXTファイルから読み込む場合は文字列の長さが必要とされず、文字列は現在位置から改行文字「¥r¥n」まで読み込まれます。CSVファイルから読み込む場合は文字列の長さが必要とされず、文字列は現在位置から最寄りの区切り文字もしくははテキスト文字列の最後の文字まで読み込まれます。

ファイルが FILE_ANSI フラグで開かれた場合は、読まれた行はUnicode に変換されます。

例（FileWriteInteger 関数の例の実行によって取得されたファイルが使用されます）

```
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- データ読み込みのパラメータ
input string InpFileName="Trend.bin"; // ファイル名
input string InpDirectoryName="Data"; // ディレクトリ名
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- ファイルを開く
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN|
FILE_ANSI);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s file is available for reading",InpFileName);
      PrintFormat("File path: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- 追加の変数
      int    str_size;
      string str;
```

© 2000–2025, MetaQuotes Ltd.

```
      //--- ファイルからデータを読む
      while(!FileIsEnding(file_handle))
        {
         //--- 時間を書くのに使用されるシンボルの数を見つける
         str_size=FileReadInteger(file_handle,INT_VALUE);
         //--- 文字列を読む
         str=FileReadString(file_handle,str_size);
         //--- 文字列を出力する
         PrintFormat(str);
        }
      //--- ファイルを閉じる
      FileClose(file_handle);
      PrintFormat("Data is read, %s file is closed",InpFileName);
     }
   else
      PrintFormat("Failed to open %s file, Error code = %
d",InpFileName,GetLastError());
  }
```

参照

String Type, Conversion Functions, FileWriteInteger

© 2000-2025, MetaQuotes Ltd.

FileReadStruct

バイナリファイルのファイルポインタの現在位置から、パラメータとして渡される構造体にコンテンツを読み込みます。

```
uint  FileReadStruct(
   int           file_handle,        // ファイルハンドル
   const void&   struct_object,      // コンテンツが読み込まれる構造体
   int           size=-1             // バイト単位の構造体サイズ
   );
```

パラメータ

file_handle

[in] バイナリファイルのファイル記述子

struct_object

[out] 構造体オブジェクト構造体は文字列、動的配列 また 仮想関数を含むことは出来ません。

size=-1

[in] 読まれるバイト数サイズが無指定、または示された値が構造体のサイズよりも大きい場合は、指定された構造のサイズが使用されます。

戻り値

成功の場合、読み込まれたバイト数ファイルポインタは、読み込まれたバイト数によって動きます。

例（FileWriteStruct 関数の例の実行によって取得されたファイルが使用されます）

```
//+------------------------------------------------------------------+
//|                                          Demo_FileReadStruct.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   1
//---- Label 1 をプロットする
#property indicator_label1  "Candles"
#property indicator_type1   DRAW_CANDLES
#property indicator_color1  clrOrange
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
#property indicator_separate_window
//--- データ受信のパラメータ
input string  InpFileName="EURUSD.txt"; // ファイル名
input string  InpDirectoryName="Data";  // ディレクトリ名
//+------------------------------------------------------------------+
//|  ローソク足データを格納する構造体                                |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
struct candlesticks
  {
   double            open;  // 始値
   double            close; // 終値
   double            high;  // 高値
   double            low;   // 安値
   datetime          date;  // 日付
  };
//--- 指標バッファ
double        open_buff[];
double        close_buff[];
double        high_buff[];
double        low_buff[];
//--- グローバル変数
candlesticks cand_buff[];
int           size=0;
int           ind=0;
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
   int default_size=100;
   ArrayResize(cand_buff,default_size);
//--- ファイルを開く
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN|
FILE_COMMON);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s file is available for reading",InpFileName);
      PrintFormat("File path: %s\\Files\
\",TerminalInfoString(TERMINAL_COMMONDATA_PATH));
      //--- ファイルからデータを読む
      while(!FileIsEnding(file_handle))
        {
         //--- データを配列に書く
         FileReadStruct(file_handle,cand_buff[size]);
         size++;
         //--- 配列がオーバーフローしているかチェックする
         if(size==default_size)
           {
            //--- 配列サイズを増やす
            default_size+=100;
            ArrayResize(cand_buff,default_size);
           }
        }
      //--- ファイルを閉じる
```

© 2000–2025, MetaQuotes Ltd.

```
      FileClose(file_handle);
      PrintFormat("Data is read, %s file is closed",InpFileName);
     }
   else
     {
      PrintFormat("Failed to open %s file, Error code = %
d",InpFileName,GetLastError());
      return(INIT_FAILED);
     }
//--- 指標バッファマッピング
   SetIndexBuffer(0,open_buff,INDICATOR_DATA);
   SetIndexBuffer(1,high_buff,INDICATOR_DATA);
   SetIndexBuffer(2,low_buff,INDICATOR_DATA);
   SetIndexBuffer(3,close_buff,INDICATOR_DATA);
//--- 空の値
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
//--- まだ処理されてないローソク足のループ
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- デフォルトでは 0
      open_buff[i]=0;
      close_buff[i]=0;
      high_buff[i]=0;
      low_buff[i]=0;
      //--- まだデータがあるかをチェック
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- 日付が同じならファイルの値を使用する
            if(time[i]==cand_buff[j].date)
```

© 2000-2025, MetaQuotes Ltd.

```
                {
                 open_buff[i]=cand_buff[j].open;
                 close_buff[i]=cand_buff[j].close;
                 high_buff[i]=cand_buff[j].high;
                 low_buff[i]=cand_buff[j].low;
                 //--- カウンタを増加する
                 ind=j+1;
                 break;
                }
            }
         }
      }
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
```

参照

構造体とクラス、FileWriteStruct

© 2000–2025, MetaQuotes Ltd.

FileSeek

この関数は、ファイルポインタの位置を指定された位置から指定されたバイト数で移動します。

```
bool  FileSeek(
   int                  file_handle,     // ファイルハンドル
   long                 offset,          // バイト単位
   ENUM_FILE_POSITION   origin           // 参考位置
   );
```

パラメータ

file_handle

[in] FileOpen() から戻されたファイル記述子

offset

[in] バイト単位でのシフト（負の値も可能）

origin

[in] シフトの開始点。ENUM_FILE_POSITION 列挙のいずれかの値。

戻り値

成功の場合は true、それ以外の場合は false。エラー情報を取得するにはGetLastError() 関数が呼ばれます。

注意事項

FileSeek() 関数の実行が（ファイルの「レベルの境界」を超えて）負のシフトを生じる場合は、ファイルポインタはファイルの先頭に設定されます。

位置が、ファイルサイズより大きい、ファイルの「右境界」を超えて設定されている場合は、次のファイルへの書き込みは、ファイルの末尾ではなく、設定された位置から行われます。この場合、前のファイルの終わりと位置設定に不定値が書き込まれます。

例：

```
//+------------------------------------------------------------------+
//|                                                Demo_FileSeek.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- 入力パラメータ
input string InpFileName="file.txt";     // ファイル名
input string InpDirectoryName="Data";    // ディレクトリ名
input int    InpEncodingType=FILE_ANSI; // ANSI=32 または UNICODE=64
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 乱数を生成するための変数の値を指定する
   _RandomSeed=GetTickCount();
//--- 文字列の開始位置のための変数
   ulong pos[];
   int   size;
//--- エラー値をリセットする
   ResetLastError();
//--- ファイルを開く
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|
InpEncodingType);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s file is available for reading",InpFileName);
      //--- ファイル内の各文字列の開始位置を受け取る
      GetStringPositions(file_handle,pos);
      //--- ファイル内の文字列の数を定義する
      size=ArraySize(pos);
      if(!size)
        {
         //--- ファイルに文字列がない場合、停止する
         PrintFormat("%s file is empty!",InpFileName);
         FileClose(file_handle);
         return;
        }
      //--- 文字列番号をランダムに選ぶ
      int ind=MathRand()%size;
      //--- 位置を文字列の始めにシフトする
      if(FileSeek(file_handle,pos[ind],SEEK_SET)==true)
        {
      //--- ind 番号で文字列を読んで出力する
         PrintFormat("String text with %d number: \"%
s\"",ind,FileReadString(file_handle));
        }
      //--- ファイルを閉じる
      FileClose(file_handle);
      PrintFormat("%s file is closed",InpFileName);
     }
   else
      PrintFormat("Failed to open %s file, Error code = %
d",InpFileName,GetLastError());
  }
//+------------------------------------------------------------------------------+
//| この関数はファイル内の文字列の開始点を定義して                                  |
//| 配列に格納する                                                              |
//+------------------------------------------------------------------------------+
void GetStringPositions(const int handle,ulong &arr[])
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- デフォルトの配列サイズ
   int def_size=127;
//--- 配列へのメモリ追加
   ArrayResize(arr,def_size);
//--- 文字列カウンタ
   int i=0;
//--- これがファイルの最後でない場合、少なくても 1 つの文字列がある
   if(!FileIsEnding(handle))
     {
      arr[i]=FileTell(handle);
      i++;
     }
   else
      return; // ファイルが空なので終了する
//--- エンコーディングに応じてバイト単位のシフトを定義する
   int shift;
   if(FileGetInteger(handle,FILE_IS_ANSI))
      shift=1;
   else
      shift=2;
//--- ループで文字列をみる
   while(1)
     {
      //--- 文字列を読む
      FileReadString(handle);
      //--- ファイルの終わりをチェックする
      if(!FileIsEnding(handle))
        {
         //--- 文字列の次の位置を格納する
         arr[i]=FileTell(handle)+shift;
         i++;
         //--- オーバーフローしている場合は、配列のサイズを増やす
         if(i==def_size)
           {
            def_size+=def_size+1;
            ArrayResize(arr,def_size);
           }
        }
      else
         break; // ファイルの終わり。終了する。
     }
//--- 配列の実際のサイズを定義する
   ArrayResize(arr,i);
  }
```

© 2000-2025, MetaQuotes Ltd.

FileSize

この関数は、ファイルサイズをバイト単位で返します。

```
ulong  FileSize(
   int  file_handle    // ファイルハンドル
   );
```

パラメータ

file_handle

[in] FileOpen() から戻されたファイル記述子

戻り値

int 型の値

注意事項

エラー情報を取得するにはGetLastError() が呼ばれます。

例:

```
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- 入力パラメータ
input ulong  InpThresholdSize=20;         // キロバイト単位でのファイルのしきい値サイズ
input string InpBigFolderName="big";      // 大きいファイルのフォルダ
input string InpSmallFolderName="small"; // 小さいファイルのフォルダ
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
   string   file_name;       // ファイル名を格納する変数
   string   filter="*.csv"; // ファイル検索フィルタ
   ulong    file_size=0;     // バイト単位のファイルサイズ
   int      size=0;          // ファイル数
//--- 操作するファイルへのパスを出力する
   PrintFormat("Working in %s\\Files\\
folder",TerminalInfoString(TERMINAL_COMMONDATA_PATH));
//--- 全ての端末に共通なフォルダのルートで検索ハンドルを受け取る
   long search_handle=FileFindFirst(filter,file_name,FILE_COMMON);
//--- FileFindFirst() の実行が成功したかをチェックする
   if(search_handle!=INVALID_HANDLE)
     {
      //--- サイズによってファイルをループで移動する
      do
        {
         //--- ファイルを開く
         ResetLastError();
         int file_handle=FileOpen(file_name,FILE_READ|FILE_CSV|FILE_COMMON);
```

© 2000-2025, MetaQuotes Ltd.

```
         if(file_handle!=INVALID_HANDLE)
           {
            //--- ファイルサイズを受け取る
            file_size=FileSize(file_handle);
            //--- ファイルを閉じる
            FileClose(file_handle);
           }
         else
           {
            PrintFormat("Failed to open %s file, Error code = %
d",file_name,GetLastError());
            continue;
           }
         //--- ファイルサイズを出力する
         PrintFormat("Size of %s file is equal to %d bytes",file_name,file_size);
         //--- ファイル移動のパスを定義する
         string path;
         if(file_size>InpThresholdSize*1024)
            path=InpBigFolderName+"//"+file_name;
         else
            path=InpSmallFolderName+"//"+file_name;
         //--- ファイルを移動する
         ResetLastError();
         if(FileMove(file_name,FILE_COMMON,path,FILE_REWRITE|FILE_COMMON))
            PrintFormat("%s file is moved",file_name);
         else
            PrintFormat("Error, code = %d",GetLastError());
        }
      while(FileFindNext(search_handle,file_name));
      //--- 検索ハンドルを閉じる
      FileFindClose(search_handle);
     }
   else
      Print("Files not found!");
  }
```

© 2000–2025, MetaQuotes Ltd.

FileTell

この関数は、開かれているファイルのファイルポインタの現在位置を返します。

```
ulong  FileTell(
   int  file_handle    // ファイルハンドル
   );
```

パラメータ

file_handle

[in] FileOpen() から戻されたファイル記述子

戻り値

ファイル記述子の現在位置のファイルの先頭からのバイト単位での距離

注意事項

エラー情報を取得するにはGetLastError() が呼ばれます。

例:

```
//+------------------------------------------------------------------+
//|                                                Demo_FileTell.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- 入力パラメータ
input string InpFileName="file.txt";     // ファイル名
input string InpDirectoryName="Data";    // ディレクトリ名
input int    InpEncodingType=FILE_ANSI; // ANSI=32 または UNICODE=64
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 乱数を生成するための変数の値を指定する
   _RandomSeed=GetTickCount();
//--- 文字列の開始位置のための変数
   ulong pos[];
   int   size;
//--- エラー値をリセットする
   ResetLastError();
//--- ファイルを開く
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|
InpEncodingType);
```

© 2000-2025, MetaQuotes Ltd.

```
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s file is available for reading",InpFileName);
      //--- ファイル内の各文字列の開始位置を受け取る
      GetStringPositions(file_handle,pos);
      //--- ファイル内の文字列の数を定義する
      size=ArraySize(pos);
      if(!size)
        {
         //--- ファイルに文字列がない場合、停止する
         PrintFormat("%s file is empty!",InpFileName);
         FileClose(file_handle);
         return;
        }
      //--- 文字列番号をランダムに選ぶ
      int ind=MathRand()%size;
      //--- 位置を文字列の始めにシフトする
      FileSeek(file_handle,pos[ind],SEEK_SET);
      //--- ind 番号で文字列を読んで出力する
      PrintFormat("String text with %d number: \"%
s\"",ind,FileReadString(file_handle));
      //--- ファイルを閉じる
      FileClose(file_handle);
      PrintFormat("%s file is closed",InpFileName);
     }
   else
      PrintFormat("Failed to open %s file, Error code = %
d",InpFileName,GetLastError());
  }
//+------------------------------------------------------------------------------+
//| この関数はファイル内の文字列の開始点を定義して                                |
//| 配列に格納する                                                               |
//+------------------------------------------------------------------------------+
void GetStringPositions(const int handle,ulong &arr[])
  {
//--- デフォルトの配列サイズ
   int def_size=127;
//--- 配列へのメモリ追加
   ArrayResize(arr,def_size);
//--- 文字列カウンタ
   int i=0;
//--- これがファイルの最後でない場合、少なくても 1 つの文字列がある
   if(!FileIsEnding(handle))
     {
      arr[i]=FileTell(handle);
      i++;
     }
   else
      return; // ファイルが空なので終了する
```

© 2000-2025, MetaQuotes Ltd.

```
//--- エンコーディングに応じてバイト単位のシフトを定義する
   int shift;
   if(FileGetInteger(handle,FILE_IS_ANSI))
      shift=1;
   else
      shift=2;
//--- ループで文字列をみる
   while(1)
     {
      //--- 文字列を読む
      FileReadString(handle);
      //--- ファイルの終わりをチェックする
      if(!FileIsEnding(handle))
        {
         //--- 文字列の次の位置を格納する
         arr[i]=FileTell(handle)+shift;
         i++;
         //--- オーバーフローしている場合は、配列のサイズを増やす
         if(i==def_size)
           {
            def_size+=def_size+1;
            ArrayResize(arr,def_size);
           }
        }
      else
         break; // ファイルの終わり。終了する。
     }
//--- 配列の実際のサイズを定義する
   ArrayResize(arr,i);
  }
```

© 2000–2025, MetaQuotes Ltd.

FileWrite

この関数は、0である場合以外は区切りを自動的に挿入してのCSVファイルへのデータの書き込みのために意図されています。ファイル書き込みの後で「¥¥n」改行文字が加えられます。

```
uint  FileWrite(
   int  file_handle,   // ファイルハンドル
   ...                 // 記録されたパラメータのリスト
   );
```

パラメータ

file_handle

[in] FileOpen() から戻されたファイル記述子

...

[in] ファイルに書かれる、コンマで区切られたパラメータのリスト（パラメータは 63 まで）

戻り値

書かれたバイト数

注意事項

数値は出力の際テキストに変換されます（Print() 関数をご参照ください）。double 型のデータは小数点以下最大 16 桁の精度で出力され、どちらの表記がよりコンパクトになるかに応じて、伝統的または科学形式で出力されます。float 型のデータは小数点以下 5 桁で出力されます。実数を異なる精度または科学形式で出力するには DoubleToString() 関数が使用されます。

bool 型の数は「true」または「false」文字列として表示されます。datetime 型の数は「YYYY.MM.DD HH:MI:SS」として表示されます。

例：

```
//+------------------------------------------------------------------+
//|                                                Demo_FileWrite.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- 端末からデータを受け取るパラメータ
input string             InpSymbolName="EURUSD";           // 通貨ペア
input ENUM_TIMEFRAMES    InpSymbolPeriod=PERIOD_H1;        // 時間軸
input int                InpFastEMAPeriod=12;              // 高速 EMA 期間
input int                InpSlowEMAPeriod=26;              // 低速 EMA 期間
input int                InpSignalPeriod=9;                // 差分平均期間
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;      // 価格の種類
input datetime           InpDateStart=D'2012.01.01 00:00'; // データコピー開始日
//--- ファイルにデータを書き込むためのパラメータ
```

© 2000–2025, MetaQuotes Ltd.

```
input string             InpFileName="MACD.csv";            // ファイル名
input string             InpDirectoryName="Data";           // ディレクトリ名
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish; // データコピー終了日
   bool     sign_buff[]; // シグナル配列(true - 買、false - 売)
   datetime time_buff[]; // シグナル到着時の配列
   int      sign_size=0; // シグナル配列サイズ
   double   macd_buff[]; // 指標値の配列
   datetime date_buff[]; // 指標日付の配列
   int      macd_size=0; // 指標配列のサイズ
//--- 終了時刻が現在時刻
   date_finish=TimeCurrent();
//--- MACD 指標ハンドルを受け取る
   ResetLastError();
   int
macd_handle=iMACD
(InpSymbolName
,InpSymbolPeriod,InpFastEMAPeriod,InpSlowEMAPeriod,InpSignalPeriod,InpAppliedPrice);
   if(macd_handle==INVALID_HANDLE)
     {
      //--- 指標ハンドルの受け取りに失敗
      PrintFormat("Error when receiving indicator handle. Error code = %
d",GetLastError());
      return;
     }
//--- 全ての指標値が計算されるまでループに留まる
   while(BarsCalculated(macd_handle)==-1)
      Sleep(10); // 指標が全ての値を計算出来るように一時停止
//--- 一定期間の指標値を複製する
   ResetLastError();
   if(CopyBuffer(macd_handle,0,InpDateStart,date_finish,macd_buff)==-1)
     {
      PrintFormat("Failed to copy indicator values. Error code = %
d",GetLastError());
      return;
     }
//--- 指標値に適切な時間を複製する
   ResetLastError();
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,date_buff)==-
1)
     {
      PrintFormat("Failed to copy time values. Error code = %d",GetLastError());
      return;
     }
//--- 指標で使用されたメモリを解放する
   IndicatorRelease(macd_handle);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- バッファサイズを受け取る
   macd_size=ArraySize(macd_buff);
//--- データを分析し指標シグナルを配列に保存する
   ArrayResize(sign_buff,macd_size-1);
   ArrayResize(time_buff,macd_size-1);
   for(int i=1;i<macd_size;i++)
     {
      //--- 買いシグナル
      if(macd_buff[i-1]<0 && macd_buff[i]>=0)
        {
         sign_buff[sign_size]=true;
         time_buff[sign_size]=date_buff[i];
         sign_size++;
        }
      //--- 売りシグナル
      if(macd_buff[i-1]>0 && macd_buff[i]<=0)
        {
         sign_buff[sign_size]=false;
         time_buff[sign_size]=date_buff[i];
         sign_size++;
        }
     }
//--- 指標値を書き込むためにファイルを開く(ファイルが存在しない場合は自動的に作成される)
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|
FILE_CSV);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s file is available for writing",InpFileName);
      PrintFormat("File path: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- 初めにシグナル数を書く
      FileWrite(file_handle,sign_size);
      //--- シグナルの時間と値をファイルに書く
      for(int i=0;i<sign_size;i++)
         FileWrite(file_handle,time_buff[i],sign_buff[i]);
      //--- ファイルを閉じる
      FileClose(file_handle);
      PrintFormat("Data is written, %s file is closed",InpFileName);
     }
   else
      PrintFormat("Failed to open %s file, Error code = %
d",InpFileName,GetLastError());
  }
```

参照

コメント、Print、StringFormat

© 2000-2025, MetaQuotes Ltd.

FileWriteArray

この関数は string 以外の任意の型の配列を BIN ファイルに書きます（文字列や動的配列を含まない構造体の配列も可能）

```
uint  FileWriteArray(
   int           file_handle,          // ファイルハンドル
   const void&   array[],              // 配列
   int           start=0,              // 配列の開始インデックス
   int           count=WHOLE_ARRAY     // 要素数
   );
```

パラメータ

file_handle

[in] FileOpen() から戻されたファイル記述子

array[]

[out] 記録する配列

start=0

[in] 配列内の最初のインデックス（最初に記録された要素の番号）

count=WHOLE_ARRAY

[in] 書かれる項目数（WHOLE_ARRAY は.開始点から配列の最後までの全ての項目が書き込まれることを意味します）。

戻り値

記録された項目数

注意事項

文字列配列は TXT ファイルに記録出来ます。この場合、文字列は自動的に行末文字「¥¥n」で終了されます。ANSI または UNICODE のファイルタイプに応じて、文字列は ansi エンコーディングに変換されます。

例：

```
//+------------------------------------------------------------------+
//|                                          Demo_FileWriteArray.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- 入力パラメータ
input string InpFileName="data.bin";
input string InpDirectoryName="SomeFolder";
//+------------------------------------------------------------------+
//|  価格データを格納する構造体                                      |
//+------------------------------------------------------------------+
```

© 2000–2025, MetaQuotes Ltd.

```
struct prices
  {
   datetime          date; // 日付
   double            bid;  // 売値
   double            ask;  // 買値
  };
//--- グローバル変数
int    count=0;
int    size=20;
string path=InpDirectoryName+"//"+InpFileName;
prices arr[];
//+------------------------------------------------------------------+
//| エキスパート初期化に使用される関数                                |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列へのメモリ追加
   ArrayResize(arr,size);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| エキスパート初期化解除に使用される関数                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- count<n の場合、残り数を書く
   WriteData(count);
  }
//+------------------------------------------------------------------+
//| エキスパートティック関数                                          |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- データを配列に保存する
   arr[count].date=TimeCurrent();
   arr[count].bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   arr[count].ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
//--- 現在のデータを表示する
   Print("Date = ",arr[count].date," Bid = ",arr[count].bid," Ask =
",arr[count].ask);
//--- カウンタを増やす
   count++;
//--- 配列がすでに記入済みなら、データをファイルに書いてゼロで書きなおす
   if(count==size)
     {
      WriteData(size);
      count=0;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//|  配列の n 要素をファイルに書く                                   |
//+------------------------------------------------------------------+
void WriteData(const int n)
  {
//--- ファイルを開く
   ResetLastError();
   int handle=FileOpen(path,FILE_READ|FILE_WRITE|FILE_BIN);
   if(handle!=INVALID_HANDLE)
     {
      //--- 配列データをファイルの終わりに書く
      FileSeek(handle,0,SEEK_END);
      FileWriteArray(handle,arr,0,n);
      //--- ファイルを閉じる
      FileClose(handle);
     }
   else
      Print("Failed to open the file, error ",GetLastError());
  }
```

参照

変数、FileSeek

© 2000-2025, MetaQuotes Ltd.

FileWriteDouble

この関数は、ファイルポインタの現在位置から開始して、ファイルに double パラメータの値を書き込みます。

```
uint  FileWriteDouble(
   int     file_handle,     // ファイルハンドル
   double  value            // 書かれる値
   );
```

パラメータ

file_handle

[in] FileOpen() から戻されたファイル記述子

value

[in] double 型の値

戻り値

成功の場合、書かれたバイト数（sizeof(double)=8）。ファイルポインタは、書かれたバイト数によって動きます。

例：

```
//+------------------------------------------------------------------+
//|                                          Demo_FileWriteDouble.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- 端末からデータを受け取るパラメータ
。input string              InpSymbolName="EURJPY";          // 通貨ペア
input ENUM_TIMEFRAMES     InpSymbolPeriod=PERIOD_M15;       // 時間軸
input int                 InpMAPeriod=10;                   // 平滑化期間
input int                 InpMAShift=0;                     // 指標のシフト shift
input ENUM_MA_METHOD      InpMAMethod=MODE_SMA;             // 平滑化の種類
input ENUM_APPLIED_PRICE  InpAppliedPrice=PRICE_CLOSE;      // 価格の種類
input datetime            InpDateStart=D'2013.01.01 00:00'; // データコピー開始日
//--- データをファイルに書くパラメータ
input string              InpFileName="MA.csv";             // ファイル名
input string              InpDirectoryName="Data";          // ディレクトリ名
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish=TimeCurrent();
```

© 2000–2025, MetaQuotes Ltd.

```
   double   ma_buff[];
   datetime time_buff[];
   int      size;
//--- MA 指標ハンドルを受け取る
   ResetLastError();
   int
ma_handle=iMA
(InpSymbolName,InpSymbolPeriod,InpMAPeriod,InpMAShift,InpMAMethod,InpAppliedPrice);
   if(ma_handle==INVALID_HANDLE)
     {
      //--- 指標ハンドルの受け取りに失敗
      PrintFormat("Error when receiving indicator handle. Error code = %
d",GetLastError());
      return;
     }
//--- 全ての指標値が計算されるまでループに留まる
   while(BarsCalculated(ma_handle)==-1)
      Sleep(20); // 指標が全ての値を計算出来るように一時停止
   PrintFormat("Indicator values starting from %s will be written to the
file",TimeToString(InpDateStart));
//--- 指標値を複製する
   ResetLastError();
   if(CopyBuffer(ma_handle,0,InpDateStart,date_finish,ma_buff)==-1)
     {
      PrintFormat("Failed to copy the indicator values. Error code = %
d",GetLastError());
      return;
     }
//--- 適切なバーの到着時間を複製する
   ResetLastError();
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-
1)
     {
      PrintFormat("Failed to copy time values. Error code = %d",GetLastError());
      return;
     }
//--- バッファサイズを受け取る
   size=ArraySize(ma_buff);
//--- 指標で使用されたメモリを解放する
   IndicatorRelease(ma_handle);
//--- 指標値を書き込むためにファイルを開く(ファイルが存在しない場合は自動的に作成される)
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|
FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s file is available for writing",InpFileName);
      PrintFormat("File path: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- 初めにデータサンプルのサイズを書く
      FileWriteDouble(file_handle,(double)size);
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 指標の時間と値をファイルに書く
      for(int i=0;i<size;i++)
        {
         FileWriteDouble(file_handle,(double)time_buff[i]);
         FileWriteDouble(file_handle,ma_buff[i]);
        }
      //--- ファイルを閉じる
      FileClose(file_handle);
      PrintFormat("Data is written, %s file is closed",InpFileName);
     }
   else
      PrintFormat("Failed to open %s file, Error code = %
d",InpFileName,GetLastError());
  }
```

参照

浮動小数点数型(ダブル、フロート)

© 2000-2025, MetaQuotes Ltd.

FileWriteFloat

この関数は、ファイルポインタの現在位置から開始して、バイナリファイルに float パラメータの値を書き込みます。

```
uint  FileWriteFloat(
   int    file_handle,     // ファイルハンドル
   float  value            // 書かれる値
   );
```

パラメータ

file_handle

[in] FileOpen() から戻されたファイル記述子

value

[in] float 型の値

戻り値

成功の場合、書かれたバイト数（sizeof(float)=4）。ファイルポインタは、書かれたバイト数によって動きます。

例：

```
//+------------------------------------------------------------------+
//|                                          Demo_FileWriteFloat.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- 端末からデータを受け取るパラメータ
input string           InpSymbolName="EURUSD";           // 通貨ペア
input ENUM_TIMEFRAMES  InpSymbolPeriod=PERIOD_M15;       // 時間軸
input datetime         InpDateStart=D'2013.01.01 00:00'; // データコピー開始日
//--- データをファイルに書くパラメータ
input string           InpFileName="Close.bin";          // ファイル名
input string           InpDirectoryName="Data";          // ディレクトリ名
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish=TimeCurrent();
   double   close_buff[];
   datetime time_buff[];
   int      size;
```

© 2000–2025, MetaQuotes Ltd.

```
//--- エラー値をリセットする
   ResetLastError();
//--- バーに終値を複製する
   if(CopyClose(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,close_buff)
==-1)
     {
      PrintFormat("Failed to copy close price values. Error code = %
d",GetLastError());
      return;
     }
//--- バーに時間を複製する
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-
1)
     {
      PrintFormat("Failed to copy the time values. Error code = %d",GetLastError());
      return;
     }
//--- バッファサイズを受け取る
   size=ArraySize(close_buff);
//--- 値を書き込むためにファイルを開く（ファイルが存在しない場合は自動的に作成される）
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|
FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s file is open for writing",InpFileName);
      PrintFormat("File path: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- 終値の時間と値をファイルに書く
      for(int i=0;i<size;i++)
        {
         FileWriteDouble(file_handle,(double)time_buff[i]);
         FileWriteFloat(file_handle,(float)close_buff[i]);
        }
      //--- ファイルを閉じる
      FileClose(file_handle);
      PrintFormat("Data is written, %s file is closed",InpFileName);
     }
   else
      PrintFormat("Failed to open %s file, Error code = %
d",InpFileName,GetLastError());
  }
```

参照

浮動小数点数型（ダブル、フロート）、FileWriteDouble

© 2000-2025, MetaQuotes Ltd.

FileWriteInteger

この関数は、ファイルポインタの現在位置から開始して、バイナリファイルに int パラメータの値を書き込みます。

```
uint  FileWriteInteger(
   int  file_handle,         // ファイルハンドル
   int  value,               // 書かれる値
   int  size=INT_VALUE       // バイト単位でのサイズ
   );
```

パラメータ

file_handle

[in] FileOpen() から戻されたファイル記述子

value

[in] 整数値

size=INT_VALUE

[in] 書かれるバイト数（4以下）。関数が char、short または int 型の値を書けるように、CHAR_VALUE=1、SHORT_VALUE=2 t及び INT_VALUE=4 と対応する定数が用意されています。

戻り値

成功の場合、書かれたバイト数。ファイルポインタは、書かれたバイト数によって動きます。

例：

```
//+------------------------------------------------------------------+
//|                                      Demo_FileWriteInteger.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- 端末からデータを受け取るパラメータ
input string              InpSymbolName="EURUSD";            // 通貨ペア
input ENUM_TIMEFRAMES     InpSymbolPeriod=PERIOD_H1;         // 時間軸
input datetime            InpDateStart=D'2013.01.01 00:00'; // データコピー開始日
//--- データをファイルに書くパラメータ
input string              InpFileName="Trend.bin"; // ファイル名
input string              InpDirectoryName="Data"; // ディレクトリ名
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish=TimeCurrent();
```

© 2000-2025, MetaQuotes Ltd.

```
   double   close_buff[];
   datetime time_buff[];
   int      size;
//--- エラー値をリセットする
   ResetLastError();
//--- バーに終値を複製する
   if(CopyClose(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,close_buff)
==-1)
     {
      PrintFormat("Failed to copy the values of close prices. Error code = %
d",GetLastError());
      return;
     }
//--- バーに時間を複製する
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-
1)
     {
      PrintFormat("Failed to copy time values. Error code = %d",GetLastError());
      return;
     }
//--- バッファサイズを受け取る
   size=ArraySize(close_buff);
//--- 値を書き込むためにファイルを開く（ファイルが存在しない場合は自動的に作成される）
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|
FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s file is available for writing",InpFileName);
      PrintFormat("File path: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //---
      int   up_down=0; // トレンドフラグ
      int   arr_size;  // arr 配列サイズ
      uchar arr[];     // uchar 型の配列
      //--- 時間値をファイルに書く
      for(int i=0;i<size-1;i++)
        {
         //--- 現在と次のバーの終値を比べる
         if(close_buff[i]<=close_buff[i+1])
           {
            if(up_down!=1)
              {
               //--- FileWriteInteger を使用してデータ値をファイルに書く
               StringToCharArray(TimeToString(time_buff[i]),arr);
               arr_size=ArraySize(arr);
               //--- 初めにシンボル数を配列に書く
               FileWriteInteger(file_handle,arr_size,INT_VALUE);
               //--- シンボルを書く
               for(int j=0;j<arr_size;j++)
```

© 2000-2025, MetaQuotes Ltd.

```
                  FileWriteInteger(file_handle,arr[j],CHAR_VALUE);
               //--- トレンドフラグを変更する
               up_down=1;
              }
           }
         else
           {
            if(up_down!=-1)
              {
               //--- FileWriteInteger を使用してデータ値をファイルに書く
               StringToCharArray(TimeToString(time_buff[i]),arr);
               arr_size=ArraySize(arr);
               //--- 初めにシンボル数を配列に書く
               FileWriteInteger(file_handle,arr_size,INT_VALUE);
               //--- シンボルを書く
               for(int j=0;j<arr_size;j++)
                  FileWriteInteger(file_handle,arr[j],CHAR_VALUE);
               //--- トレンドフラグを変更する
               up_down=-1;
              }
           }
        }
      //--- ファイルを閉じる
      FileClose(file_handle);
      PrintFormat("Data is written, %s file is closed",InpFileName);
     }
   else
      PrintFormat("Failed to open %s file, Error code = %
d",InpFileName,GetLastError());
  }
```

参照

IntegerToString、StringToInteger、整数型

© 2000-2025, MetaQuotes Ltd.

FileWriteLong

この関数は、ファイルポインタの現在位置から開始して、バイナリファイルに long 型のパラメータの値を書き込みます。

```
uint  FileWriteLong(
   int   file_handle,     // ファイルハンドル
   long  value            // 書かれる値
   );
```

パラメータ

file_handle

[in] FileOpen() から戻されたファイル記述子

value

[in] long 型の値

戻り値

成功の場合、書かれたバイト数（sizeof(long)=8）。ファイルポインタは、書かれたバイト数によって動きます。

例：

```
//+------------------------------------------------------------------+
//|                                              Demo_FileWriteLong.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- 端末からデータを受け取るパラメータ
input string          InpSymbolName="EURUSD";            // 通貨ペア
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_H1;         // 時間軸
input datetime        InpDateStart=D'2013.01.01 00:00'; // データコピー開始日
//--- データをファイルに書くパラメータ
input string          InpFileName="Volume.bin"; // ファイル名
input string          InpDirectoryName="Data";  // ディレクトリ名
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish=TimeCurrent();
   long     volume_buff[];
   datetime time_buff[];
   int      size;
```

© 2000–2025, MetaQuotes Ltd.

```
//--- エラー値をリセットする
   ResetLastError();
//--- バーにティックボリュームを複製する
   if
(CopyTickVolume(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,volume_buff)
==-1)
     {
      PrintFormat("Failed to copy values of the tick volume. Error code = %
d",GetLastError());
      return;
     }
//--- バーに時間を複製する
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-
1)
     {
      PrintFormat("Failed to copy time values. Error code = %d",GetLastError());
      return;
     }
//--- バッファサイズを受け取る
   size=ArraySize(volume_buff);
//--- 指標値を書き込むためにファイルを開く(ファイルが存在しない場合は自動的に作成される)
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|
FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s file is available for writing",InpFileName);
      PrintFormat("File path: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- 初めにデータサンプルサイズを書く
      FileWriteLong(file_handle,(long)size);
      //--- 時間とボリューム値をファイルに書く
      for(int i=0;i<size;i++)
        {
         FileWriteLong(file_handle,(long)time_buff[i]);
         FileWriteLong(file_handle,volume_buff[i]);
        }
      //--- ファイルを閉じる
      FileClose(file_handle);
      PrintFormat("Data is written, %s file is closed",InpFileName);
     }
   else
      PrintFormat("Failed to open %s file, Error code = %
d",InpFileName,GetLastError());
  }
```

参照

整数型、FileWriteInteger

© 2000-2025, MetaQuotes Ltd.

FileWriteString

この関数は、BIN、CSV または TXT ファイルのファイルポインタの現在位置に string 型のパラメータの値を書き込みます。CSV や TXT ファイルに書き込む場合、文字列が、事前に「¥r」(CR) がない「¥n」(LF) を含む場合，「¥n」の前に「¥r」が加えられます。

```
uint  FileWriteString(
   int           file_handle,     // ファイルハンドル
   const string  text_string,     // 書かれる文字列
   int           length=-1        // シンボル数
   );
```

パラメータ

file_handle

[in] FileOpen() から戻されたファイル記述子

text_string

[in] 文字列

length=-1

[in] 書かれる文字数。このオプションはBINファイルに文字列を書き込むために必要です。サイズが指定されていない場合には、トレーラー 0 を含まない文字列全体が書かれます。文字列の長さよりも小さいサイズが指定された場合は、その部分がトレーラー 0 を含まずに書かれます。文字列の長さよりも大きいサイズが指定された場合は、適切な数のゼロが書き足されます。CSV 及び TXT 形式のファイルではこのパラメータは無視され、文字列全体が書かれます。

戻り値

成功の場合、書かれたバイト数。ファイルポインタは、書かれたバイト数によって動きます。

注意事項

FILE_UNICODE フラグ を使用して、もしくは FILE_ANSI フラグなしで開かれたファイルに書き込む場合、書かれたバイト数は書かれた文字数の 2 倍になることにご注意下さい。FILE_ANSI フラグを使用して開かれたファイルに書き込む場合書かれたバイト数は書かれた文字数と同じです。

例:

```
//+------------------------------------------------------------------+
//|                                        Demo_FileWriteString.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                             https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- 端末からデータを受け取るパラメータ
input string              InpSymbolName="EURUSD";           // 通貨ペア
input ENUM_TIMEFRAMES     InpSymbolPeriod=PERIOD_H1;        // 時間軸
input int                 InpMAPeriod=14;                   // MA 期間
```

© 2000–2025, MetaQuotes Ltd.

```
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;       // 価格の種類
input datetime           InpDateStart=D'2013.01.01 00:00'; // データコピー開始日
//--- データをファイルに書くパラメータ
input string             InpFileName="RSI.csv";    // ファイル名
input string             InpDirectoryName="Data"; // ディレクトリ名
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish; // データコピー終了日
   double   rsi_buff[];  // 指標値の配列
   datetime date_buff[]; // 指標日付の配列
   int      rsi_size=0;  // 指標配列のサイズ
//--- 終了時刻が現在の時刻
   date_finish=TimeCurrent();
//--- RSI  指標ハンドルを受け取る
   ResetLastError();
   int rsi_handle=iRSI(InpSymbolName,InpSymbolPeriod,InpMAPeriod,InpAppliedPrice);
   if(rsi_handle==INVALID_HANDLE)
     {
      //--- 指標ハンドルの受け取りに失敗
      PrintFormat("Error when receiving indicator handle. Error code = %
d",GetLastError());
      return;
     }
//--- 指標が全ての値を計算するまでループ内にとどまる
   while(BarsCalculated(rsi_handle)==-1)
      Sleep(10); // 指標が全ての値を計算出来るように一時停止
//--- 一定期間の指標値を複製する
   ResetLastError();
   if(CopyBuffer(rsi_handle,0,InpDateStart,date_finish,rsi_buff)==-1)
     {
      PrintFormat("Failed to copy indicator values. Error code = %
d",GetLastError());
      return;
     }
//--- 指標値に適切な時間を複製する
   ResetLastError();
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,date_buff)==-
1)
     {
      PrintFormat("Failed to copy time values. Error code = %d",GetLastError());
      return;
     }
//--- 指標で使用されたメモリを解放する
   IndicatorRelease(rsi_handle);
//--- バッファサイズを受け取る
   rsi_size=ArraySize(rsi_buff);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 指標値を書き込むためにファイルを開く(ファイルが存在しない場合は自動的に作成される)
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|
FILE_CSV|FILE_ANSI);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s file is available for writing",InpFileName);
      PrintFormat("File path: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- 追加の変数を準備する
      string str="";
      bool   is_formed=false;
      //--- 買われ過ぎと売られ過ぎ領域を作る日付を書く
      for(int i=0;i<rsi_size;i++)
        {
         //--- 指標値をチェックする
         if(rsi_buff[i]>=70 || rsi_buff[i]<=30)
           {
            //--- これが領域の初めの値の場合
            if(!is_formed)
              {
               //--- 値と日付を加える
               str=(string)rsi_buff[i]+"\t"+(string)date_buff[i];
               is_formed=true;
              }
            else
               str+="\t"+(string)rsi_buff[i]+"\t"+(string)date_buff[i];
            //--- 次のループ反復に移る
            continue;
           }
         //--- フラグをチェックする
         if(is_formed)
           {
            //--- 文字列が用意できたのでファイルに書く
            FileWriteString(file_handle,str+"\r\n");
            is_formed=false;
           }
        }
      //--- ファイルを閉じる
      FileClose(file_handle);
      PrintFormat("Data is written, %s file is closed",InpFileName);
     }
   else
      PrintFormat("Failed to open %s file, Error code = %
d",InpFileName,GetLastError());
  }
```

参照

String Type, StringFormat

© 2000-2025, MetaQuotes Ltd.

FileWriteStruct

この関数は、バイナリファイルのファイルポインタの現在の位置から、パラメータとして渡される構造体のコンテンツを書き込みます。

```
uint  FileWriteStruct(
   int           file_handle,       // ファイルハンドル
   const void&   struct_object,     // オブジェクトへのリンク
   int           size=-1            // 書かれるバイト数
   );
```

パラメータ

file_handle

[in] FileOpen() から戻されたファイル記述子

struct_object

[in] 構造体オブジェクトへの参照。構造体は文字列、動的配列 また 仮想関数を含むことは出来ません。

size=-1

[in] 書かれるバイト数サイズが指定されない、または指定されたバイト数が構造のサイズよりも大きい場合は、構造体全体が書かれます。

戻り値

成功の場合、書かれたバイト数。ファイルポインタは、書かれたバイト数によって動きます。

例:

```
//+------------------------------------------------------------------+
//|                                          Demo_FileWiteStruct.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- 端末からデータを受け取るパラメータ
input string          InpSymbolName="EURUSD";            // 通貨ペア
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_H1;         // 時間軸
input datetime        InpDateStart=D'2013.01.01 00:00'; // データコピー開始日
//--- データをファイルに書くパラメータ
input string          InpFileName="EURUSD.txt";          // ファイル名
input string          InpDirectoryName="Data";           // ディレクトリ名
//+------------------------------------------------------------------+
//|  ローソク足データを格納する構造体                                |
//+------------------------------------------------------------------+
struct candlesticks
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   double            open;  // 始値
   double            close; // 終値
   double            high;  // 高値
   double            low;   // 安値
   datetime          date;  // 日付
  };
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime     date_finish=TimeCurrent();
   int          size;
   datetime     time_buff[];
   double       open_buff[];
   double       close_buff[];
   double       high_buff[];
   double       low_buff[];
   candlesticks cand_buff[];
//--- エラー値をリセットする
   ResetLastError();
//--- 範囲内のバーの到着時間を受信
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-
1)
     {
      PrintFormat("Failed to copy time values. Error code = %d",GetLastError());
      return;
     }
//--- 範囲内の足の高値を受信
   if(CopyHigh(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,high_buff)==-
1)
     {
      PrintFormat("Failed to copy values of high prices. Error code = %
d",GetLastError());
      return;
     }
//--- 範囲内のバーの安値を受信
   if(CopyLow(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,low_buff)==-1)
     {
      PrintFormat("Failed to copy values of low prices. Error code = %
d",GetLastError());
      return;
     }
//--- 範囲内のバーの始値を受信
   if(CopyOpen(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,open_buff)==-
1)
     {
      PrintFormat("Failed to copy values of open prices. Error code = %
d",GetLastError());
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- 範囲内のバーの終値を受信
   if(CopyClose(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,close_buff)
==-1)
      {
       PrintFormat("Failed to copy values of close prices. Error code = %
d",GetLastError());
       return;
      }
//--- 配列の次元を定義
   size=ArraySize(time_buff);
//--- データを構造体配列に保存
   ArrayResize(cand_buff,size);
   for(int i=0;i<size;i++)
     {
      cand_buff[i].open=open_buff[i];
      cand_buff[i].close=close_buff[i];
      cand_buff[i].high=high_buff[i];
      cand_buff[i].low=low_buff[i];
      cand_buff[i].date=time_buff[i];
     }

//--- 構造体配列を書き込むためにファイルを開く(ファイルが存在しない場合は自動的に作成される)
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|
FILE_BIN|FILE_COMMON);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s file is open for writing",InpFileName);
      PrintFormat("File path: %s\\Files\
\",TerminalInfoString(TERMINAL_COMMONDATA_PATH));
      //--- バイト数カウンタを準備する
      uint counter=0;
      //--- ループで配列値を書く
      for(int i=0;i<size;i++)
         counter+=FileWriteStruct(file_handle,cand_buff[i]);
      PrintFormat("%d bytes of information is written to %s
file",InpFileName,counter);
      PrintFormat("Total number of bytes: %d * %d * %d = %d, %
s",size,5,8,size*5*8,size*5*8==counter ? "Correct" : "Error");
      //--- ファイルを閉じる
      FileClose(file_handle);
      PrintFormat("Data is written, %s file is closed",InpFileName);
     }
   else
      PrintFormat("Failed to open %s file, Error code = %
d",InpFileName,GetLastError());
  }
```

参照

© 2000-2025, MetaQuotes Ltd.

構造体とクラス

© 2000–2025, MetaQuotes Ltd.

FileLoad

指定されたバイナリファイルのすべてのデータを渡された数値型または単純な構造体型の配列に読み込みます。この関数を使用すると、既知の型のデータをすぐに適切な配列に読み取ることができます。

```
long  FileLoad(
   const string  file_name,          // ファイル名
   void&         buffer[],           // 数値型または単純な構造体型の配列
   int           common_flag=0       // ファイルフラグ、デフォルトでは <data_folder>¥MQL5¥Files¥
で検索される
   );
```

パラメータ

file_name

[in] 含まれるデータが読まれるファイル名。

buffer

[out] 数値型または単純な構造体型の配列。

common_flag=0

[in] 操作モードを示すファイルフラグ。このパラメータが指定されていない場合、ファイルは MQL5¥Files サブフォルダ（または検証の場合 <testing_agent_directory>¥MQL5¥Files ）で検索されます。

戻り値

読み取られた要素数。エラーの場合は-1。

注意事項

FileLoad()関数は、ファイルから配列の要素サイズの倍数のバイト数を読み取ります。ファイルサイズが10バイトで関数がdouble（sizeof(double)=8）型の配列にデータを読み込むとします。この場合、FileLoad()は8バイトのみを読んでファイルの終わりに残る2バイトを無視して、（1要素が読み込まれたため）1を戻します。

例：

```
//+------------------------------------------------------------------+
//|                                                Demo_FileLoad.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property script_show_inputs
//--- 入力パラメータ
input int      bars_to_save=10; // バー数
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                  |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
   string  filename=_Symbol+"_rates.bin";
   MqlRates rates[];
//---
   int copied=CopyRates(_Symbol,_Period,0,bars_to_save,rates);
   if(copied!=-1)
     {
      PrintFormat(" CopyRates(%s) copied %d bars",_Symbol,copied);
      //---   相場をファイルに書く
      if(!FileSave(filename,rates,FILE_COMMON))
         PrintFormat("FileSave() failed, error=%d",GetLastError());
     }
   else
      PrintFormat("Failed CopyRates(%s), error=",_Symbol,GetLastError());
//--- 相場をファイルに読み返す
   ArrayFree(rates);
   long count=FileLoad(filename,rates,FILE_COMMON);
   if(count!=-1)
     {
      Print("Time\tOpen\tHigh\tLow\tClose\tTick Voulme\tSpread\tReal Volume");
      for(int i=0;i<count;i++)
        {
         PrintFormat("%s\t%G\t%G\t%G\t%G\t%I64u\t%d\t%I64u",
                     TimeToString(rates[i].time,TIME_DATE|TIME_SECONDS),
                     rates[i].open,rates[i].high,rates[i].low,rates[i].close,
                     rates[i].tick_volume,rates[i].spread,rates[i].real_volume);
        }
     }
  }
```

参照

構造体とクラス、FileReadArray、FileReadStruct、FileSave

© 2000–2025, MetaQuotes Ltd.

FileSave

配列のすべての要素をパラメータとして渡されたバイナリファイルに書き込みます。この関数を使用すると、数値型または単純な構造体型の配列をすぐに単一の文字列として書くことができます。

```
bool  FileSave(
   const string  file_name,          // ファイル名
   void&         buffer[],           // 数値型または単純な構造体型の配列
   int           common_flag=0       // ファイルフラグ、デフォルトでは <data_folder>¥MQL5¥Files¥
に書かれる
   );
```

パラメータ

file_name

[in] データ配列の書き込み先のファイル名

buffer

[in] 数値型または単純な構造体型の配列。

common_flag=0

[in] 操作モードを示すファイルフラグ。このパラメータが指定されていない場合、ファイルは MQL5¥Files サブフォルダ（または検証の場合 <testing_agent_directory>¥MQL5¥Files）に書かれます。

戻り値

失敗の場合 false

例：

```
//+------------------------------------------------------------------+
//|                                                Demo_FileSave.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property script_show_inputs
//--- 入力パラメータ
input int      ticks_to_save=1000; // ティック数
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
   string  filename=_Symbol+"_ticks.bin";
   MqlTick ticks[];
u//---
   int copied=CopyTicks(_Symbol,ticks,COPY_TICKS_ALL,0,ticks_to_save);
   if(copied!=-1)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat(" CopyTicks(%s) copied %d ticks",_Symbol,copied);
      //--- ティック履歴が同期されている場合はエラーコードは0に等しい
      if(!GetLastError()==0)
         PrintFormat("%s: Ticks are not synchronized, error=%
d",_Symbol,copied,_LastError);
      //---  ティックをファイルに書く
      if(!FileSave(filename,ticks,FILE_COMMON))
         PrintFormat("FileSave() failed, error=%d",GetLastError());
     }
   else
      PrintFormat("Failed CopyTicks(%s), Error=",_Symbol,GetLastError());
//--- ティックをファイルに読み返す
   ArrayFree(ticks);
   long count=FileLoad(filename,ticks,FILE_COMMON);
   if(count!=-1)
     {
      Print("Time\tBid\tAsk\tLast\tVolume\tms\tflags");
      for(int i=0;i<count;i++)
        {
         PrintFormat("%s.%03I64u:\t%G\t%G\t%G\t%I64u\t0x%04x",
         TimeToString(ticks[i].time,TIME_DATE|TIME_SECONDS),ticks[i].time_msc%1000,
         ticks[i].bid,ticks[i].ask,ticks[i].last,ticks[i].volume,ticks[i].flags);
        }
     }
  }
```

参照

構造体とクラス、FileWriteArray、FileWriteStruct, FileLoad, FileWrite

© 2000-2025, MetaQuotes Ltd.

FolderCreate

この関数は、common_flag の値によって、Files ディレクトリ内にフォルダを作成します。

```
bool  FolderCreate(
   string  folder_name,        // 新しいフォルダの名称
   int     common_flag=0       // 範囲
   );
```

パラメータ

folder_name

[in] Name of the directory to be created. Contains the relative path to the folder.

common_flag=0

[in] ディレクトリの場所を決めるフラグ。common_flag=FILE_COMMON の場合、ディレクトリは全てのクライアント端末の共有フォルダ ¥Terminal¥Common¥Filesに位置します。その他の場合、ディレクトリはローカルフォルダ（MQL5¥Files またはテストの場合 MQL5¥Tester¥Files）に位置します。

戻り値

成功の場合は true、それ以外の場合は false

注意事項

セキュリティ上の理由から、MQL5 言語ではファイルの扱いは厳しく制御されています。MQL5 手段を用いて操作されるファイルは、ファイルサンドボックスの外に存在することは出来ません。

例：

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- description
#property description "The script shows FolderCreate() application sample."
#property description "The external parameter defines the directory for creating
folders."
#property description "The folder structure is created after executing the script"

//--- display window of the input parameters during the script's launch
#property script_show_inputs
//--- the input parameter defines the folder, in which the script works
input bool     common_folder=false; // common folder for all terminals
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- folder to be created in MQL5\Files
   string root_folder="Folder_A";
   if(CreateFolder(root_folder,common_folder))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- create the Child_Folder_B1 sub-folder in it
      string folder_B1="Child_Folder_B1";
      string path=root_folder+"\\"+folder_B1;          // create the folder name
considering the structure
      if(CreateFolder(path,common_folder))
        {
         //--- create 3 more sub-directories in this folder
         string folder_C11="Child_Folder_C11";
         string child_path=root_folder+"\\"+folder_C11;// create the folder name
considering the structure
         CreateFolder(child_path,common_folder);
         //--- second sub-directory
         string folder_C12="Child_Folder_C12";
         child_path=root_folder+"\\"+folder_C12;
         CreateFolder(child_path,common_folder);

         //--- third sub-directory
         string folder_C13="Child_Folder_C13";
         child_path=root_folder+"\\"+folder_C13;
         CreateFolder(child_path,common_folder);
        }
     }
//---
  }
//+------------------------------------------------------------------+
//| Try creating a folder and display a message about that           |
//+------------------------------------------------------------------+
bool CreateFolder(string folder_path,bool common_flag)
  {
   int flag=common_flag?FILE_COMMON:0;
   string working_folder;
//--- define the full path depending on the common_flag parameter
   if(common_flag)
      working_folder=TerminalInfoString(TERMINAL_COMMONDATA_PATH)+"\\MQL5\\Files";
   else
      working_folder=TerminalInfoString(TERMINAL_DATA_PATH)+"\\MQL5\\Files";
//--- debugging message
   PrintFormat("folder_path=%s",folder_path);
//--- attempt to create a folder relative to the MQL5\Files path
   if(FolderCreate(folder_path,flag))
     {
      //--- display the full path for the created folder
      PrintFormat("Created the folder %s",working_folder+"\\"+folder_path);
      //--- reset the error code
      ResetLastError();
      //--- successful execution
      return true;
     }
   else
```

© 2000–2025, MetaQuotes Ltd.

```
      PrintFormat("Failed to create the folder %s. Error code %
d",working_folder+folder_path,GetLastError());
//--- execution failed
   return false;
  }
```

参照

FileOpen()、FolderClean()、FileCopy()

© 2000–2025, MetaQuotes Ltd.

FolderDelete

この関数は、指定されたディレクトリを削除します。フォルダが空でない場合は、削除出来ません。

```
bool  FolderDelete(
   string  folder_name,        // 削除するフォルダの名称
   int     common_flag=0       // 範囲
   );
```

パラメータ

folder_name

[in] 削除するディレクトリ名。フォルダの絶対パスを含みます。

common_flag=0

[in] ディレクトリの場所を決めるフラグ。common_flag=FILE_COMMONの場合、ディレクトリは全てのクライアント端末の共有フォルダ ¥Terminal¥Common¥Filesに位置します。その他の場合、ディレクトリはローカルフォルダ（MQL5¥Files またはテストの場合 MQL5¥Tester¥Files）に位置します。

戻り値

成功の場合はtrue、それ以外の場合はfalse。

注意事項

セキュリティ上の理由から、MQL5 言語ではファイルの扱いは厳しく制御されています。MQL5 手段を用いて操作されるファイルは、ファイルサンドボックスの外に存在することは出来ません。

ディレクトリが少なくとも1つのファイル及び/またはサブディレクトリを含む場合は、ディレクトリの削除が出来ないので、最初にこれらが削除される必要があります。FolderClean() は、フォルダの全てのファイルやサブフォルダを削除するのに使用されます。

例：

```
//+------------------------------------------------------------------+
//|                                            Demo_FolderDelete.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- 説明
#property description "The script shows a sample use of FolderDelete()."
#property description "First two folders are created; one of them is empty, the
second one contains a file."
#property description "When trying to delete a non-empty folder, an error is
returned and a warning is shown."

//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- 入力パラメータ
input string   firstFolder="empty";     // 空のフォルダ
```

© 2000-2025, MetaQuotes Ltd.

```
input string   secondFolder="nonempty";// ファイルが作成されるフォルダ
string filename="delete_me.txt";       // secondFolder フォルダに作成されるファイルの名称
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- ここにファイルハンドルを書く
   int handle;
//--- 作業を行うフォルダを見つける
   string working_folder=TerminalInfoString(TERMINAL_DATA_PATH)+"\\MQL5\\Files";
//--- デバッグメッセージ
   PrintFormat("working_folder=%s",working_folder);
//--- MQL5¥Files パスに相対した空フォルダ作成を試みる
   if(FolderCreate(firstFolder,0)) // 0 は端末のローカルフォルダでの作業を意味する
     {
      //--- 作成されたフォルダの絶対パスを書く
      PrintFormat("Folder %s has been created",working_folder+"\\"+firstFolder);
      //--- エラーコードをリセットする
      ResetLastError();
     }
   else
      PrintFormat("Failed to create folder %s. Error code %d",working_folder+"\
\"+firstFolder, GetLastError());

//--- FileOpen() 関数で空でないフォルダを作成する
   string filepath=secondFolder+"\\"+filename;  // 存在しないフォルダに作成して書き込むファイルのパスを作る
   handle=FileOpen(filepath,FILE_WRITE|FILE_TXT); // この場合 FILE_WRITE は必須です。FileOpen のヘルプをご参照ください。
   if(handle!=INVALID_HANDLE)
      PrintFormat("File %s has been opened for reading",working_folder+"\
\"+filepath);
   else
      PrintFormat("Failed to create file %s in folder %s. Error
code=",filename,secondFolder, GetLastError());

   Comment(StringFormat("Prepare to delete folders %s and %s", firstFolder,
secondFolder));
//--- チャートのメッセージを読む5秒の休止
   Sleep(5000); // Sleep() cannot be used in indicators!

//--- ダイアログを表示してユーザに聞く
   int choice=MessageBox(StringFormat("Do you want to delete folders %s and %s?",
firstFolder, secondFolder),
                         "Deleting folders",
                         MB_YESNO|MB_ICONQUESTION); //  「はい(Y)」と「いいえ(N)」の 2 つのボタン

//--- 選択されたバージョンに応じてアクションを実行する
   if(choice==IDYES)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- チャートからコメントを削除
      Comment("");
      //--- エキスパート操作ログにメッセージを加える
      PrintFormat("Trying to delete folders %s and %s",firstFolder, secondFolder);
      ResetLastError();
      //--- 空のフォルダを削除
      if(FolderDelete(firstFolder))
         //--- フォルダが空なので次のメッセージが表示されるべき
         PrintFormat("Folder %s has been successfully deleted",firstFolder);
      else
         PrintFormat("Failed to delete folder %s. Error code=%d", firstFolder,
GetLastError());

      ResetLastError();
      //--- ファイルを含むフォルダを削除
      if(FolderDelete(secondFolder))
         PrintFormat("Folder %s has been successfully deleted", secondFolder);
      else
         //--- フォルダにファイルが存在するので次のメッセージが表示されるべき
         PrintFormat("Failed to delete folder %s. Error code=%d", secondFolder,
GetLastError());
     }
   else
      Print("Deletion canceled");
//---
  }
```

参照

FileOpen()、FolderClean()、FileMove()

© 2000-2025, MetaQuotes Ltd.

FolderClean

この関数は、指定されたフォルダ内の全てのファイルを削除します。

```
bool  FolderClean(
   string  folder_name,        // 削除されるフォルダの名称
   int     common_flag=0       // 範囲
   );
```

パラメータ

folder_name

[in] 削除されるファイルのディレクトリ名。フォルダの絶対パスを含みます。

common_flag=0

[in] ディレクトリの場所を決めるフラグ。common_flag = FILE_COMMON の場合、ディレクトリは全てのクライアント端末の共有フォルダ ¥Terminal¥Common¥Filesに位置します。その他の場合、ディレクトリはローカルフォルダ（MQL5¥Files またはテストの場合 MQL5¥Tester¥Files）に位置します。

戻り値

成功の場合は true、それ以外の場合は false。

注意事項

セキュリティ上の理由から、MQL5 言語ではファイルの扱いは厳しく制御されています。MQL5 手段を用いて操作されるファイルは、ファイルサンドボックスの外に存在することは出来ません。

全てのファイルとサブディレクトリが削除されて回復不能なので、この関数の使用には注意が必要です。

例：

```
//+------------------------------------------------------------------+
//|                                                Demo_FolderClean.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- 説明
#property description "The script shows a sample use of FolderClean()."
#property description "First, files are created in the specified folder using the FileOpen() function."
#property description "Then, before the files are deleted, a warning is shown using MessageBox()."

//--- スクリプトの起動時に入力パラメータのウィンドウを表示する
#property script_show_inputs
//--- 入力パラメータ
input string   foldername="demo_folder";  // MQL5/Files/ にフォルダを作成
input int      files=5;                   // 作成及び削除するファイルの数
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|  スクリプトプログラムを開始する関数                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
   string name="testfile";
//--- 初めに端末データフォルダでファイルを開けるか作成する
   for(int N=0;N<files;N++)
     {
      //--- ファイル名は「demo_folder¥testfileN.txt」
      string filemane=StringFormat("%s\\%s%d.txt",foldername,name,N);
      //--- ファイルを書き入れフラグで開く。この場合「demo_folder」が自動的に作成される
      int handle=FileOpen(filemane,FILE_WRITE);
      //--- FileOpen() 関数が成功したか見つける
      if(handle==INVALID_HANDLE)
        {
         PrintFormat("Failed to create file %s. Error
code",filemane,GetLastError());
         ResetLastError();
        }
      else
        {
         PrintFormat("File %s has been successfully opened",filemane);
         //--- 開かれたファイルがもう必要ないので閉じる
         FileClose(handle);
        }
     }

//--- フォルダ内のファイル数をチェックする
   int k=FilesInFolder(foldername+"\\*.*",0);
   PrintFormat("Totally the folder %s contains %d files",foldername,k);

//--- ダイアログを表示してユーザに聞く
   int choice=MessageBox(StringFormat("You are going to delete %d files from folder
%s. Do you want to continue?",foldername,k),
                         "Deleting files from the folder",
                         MB_YESNO|MB_ICONQUESTION); //  「はい(Y)」と「いいえ(N)」の 2 つの
ボタン
   ResetLastError();

//--- 選択されたバージョンによってアクションを実行する
   if(choice==IDYES)
     {
      //--- ファイル削除を開始
      PrintFormat("Trying to delete all files from folder %s",foldername);
      if(FolderClean(foldername,0))
         PrintFormat("Files have been successfully deleted, %d files left in folder
%s",
                     foldername,
                     FilesInFolder(foldername+"\\*.*",0));
      else
```

© 2000-2025, MetaQuotes Ltd.

```
         PrintFormat("Failed to delete files from folder %s. Error code %
d",foldername,GetLastError());
      }
   else
      PrintFormat("Deletion canceled");
//---
  }
//+------------------------------------------------------------------+
//| 指定されたフォルダ内のファイル数を返す                                |
//+------------------------------------------------------------------+
int FilesInFolder(string path,int flag)
  {
   int count=0;
   long handle;
   string filename;
//---
   handle=FileFindFirst(path,filename,flag);
//--- ファイルが見つかったらもっと探す
   if(handle!=INVALID_HANDLE)
     {
      //--- ファイル名を表示
      PrintFormat("File %s found",filename);
      //--- 見つかったファイル・フォルダのカウンタを増やす
      count++;
      //--- 全てのファイル・フォルダで検索を開始
      while(FileFindNext(handle,filename))
        {
         PrintFormat("File %s found",filename);
         count++;
        }
      //--- 終了後に忘れずに検索ハンドルを閉じる
      FileFindClose(handle);
     }
   else // ハンドル取得に失敗
     {
      PrintFormat("Files search in folder %s failed",path);
     }
//--- 結果を返す
   return count;
  }
```

参照

FileFindFirst、FileFindNext、FileFindClose

© 2000-2025, MetaQuotes Ltd.

カスタム指標

これは、カスタム指標の作成に使用される関数群です。これれの関数はエキスパートアドバイザーとスクリプトでは使用できません。

関数	アクション
SetIndexBuffer	指定された指標バッファをdouble 型の 1 次元の動的配列と関係づけます。
IndicatorSetDouble	double 型の指標プロパティの値を設定します。
IndicatorSetInteger	int 型の指標プロパティの値を設定します。
IndicatorSetString	string 型の指標プロパティの値を設定します。
PlotIndexSetDouble	double 型の指標線プロパティの値を設定します。
PlotIndexSetInteger	int 型の指標線プロパティの値を設定します。
PlotIndexSetString	string 型の指標線プロパティの値を設定します。
PlotIndexGetInteger	integer 型の指標線プロパティの値を返します。

指標プロパティは、コンパイラディレクティブか関数を使用して設定することが出来ます。よりよく理解するためには指標スタイルの例を勉強することをお勧めします。

カスタム指標に必要な全ての計算は、所定の OnCalculate() 関数に配置されなければなりません。OnCalculate() 関数呼び出しの短い形式を使用する場合、下記の様になります。

```
int OnCalculate (const int rates_total, const int prev_calculated, const int begin,
const double& price[])
```

すると *rates_total* 変数は、指標値計算のための入力パラメータとして渡されたprice[] 配列の要素の合計数を含みます。

prev_calculated パラメータは前の呼び出しでの OnCalculate() の実行の結果です。これは、指標値を計算するためのアルゴリズムの保存や整理を可能にします。例えば、現在の値 rates_total = 1000 で prev_calculated = 999、各指標バッファの 1 つの値の計算を行うので充分です。

price 入力配列のサイズに関する情報が利用出来ない場合は、それぞれの指標バッファの1000 個の値の計算を行うことが必要になります。OnCalculate() の最初の呼び出しで、prev_calculated = 0 です。price[] 配列が変わった場合 prev_calculated も 0 です。

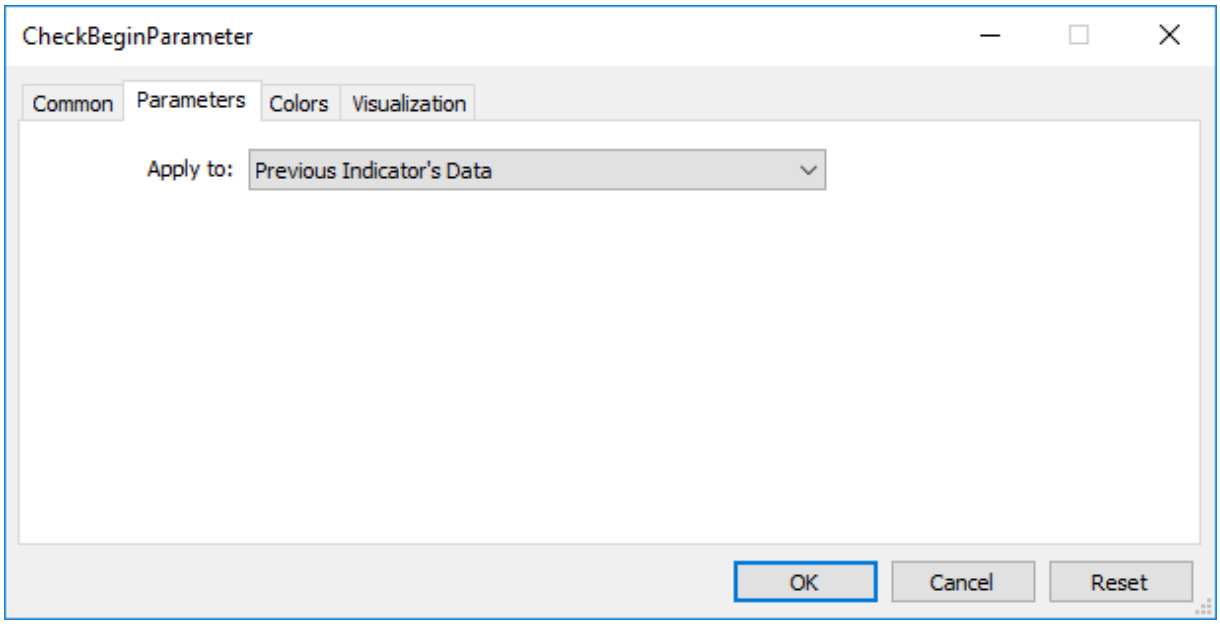
begin パラメータは、計算のためのデータを含まない price 配列の初期値の数を示します。例えば、(初めの37の値が計算されていない) ACオシレーターの値を入力パラメータとして使用された場合、begin = 37 です。例として、単純な指標を見てみましょう。

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- Label 1 をプロットする
#property indicator_label1  "Label1"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
```

© 2000–2025, MetaQuotes Ltd.

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 指標バッファ
double         Label1Buffer[];
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- 指標バッファマッピング
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
//---
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double &price[])

  {
//---
   Print("begin = ",begin,"  prev_calculated = ",prev_calculated,"  rates_total =
",rates_total);
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
```

「ナビゲータ」ウィンドウからACオシレーター指標のウィンドウにドラッグして、計算が前の指標の値に基づいて行われることを示します。



© 2000–2025, MetaQuotes Ltd.

結果として、OnCalculate() の最初の呼び出しで prev_calculated の値はゼロに等しく、以降の呼び出しでは（価格チャートでバーの数が増えるまで）rates_total 値に等しくなります。



begin パラメータの値は初期のバーの数と等しくなります。この指標の論理に従って、アクセラレータ指標の値は計算されていません。Accelerator.mq5 のカスタム指標のコードをみると、OnInit() 関数にて以下の行が見られます。

```
//--- インデックスが描画される最初のバーを設定する
   PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,37);
```

PlotIndexSetInteger(0, PLOT_DRAW_BEGIN, empty_first_values) 関数を使用して、計算のために受け入れる必要のないカスタム指標のゼロ指標配列に架空の初期値を設定します（empty_first_values）。従って、下記のメカニズムがあります。

1. 別のカスタム指標の計算に使用されるべきではない指標の初期値を設定する
2. 別のカスタム指標を呼び出した時、計算の論理を通さずに無視される初期値についての情報を得る

© 2000–2025, MetaQuotes Ltd.

指標スタイルの例

MetaTrader 5 クライアント端末は適切な関数を使用して MQL5 プログラムで使用出来る 38 のテクニカル指標を含みます。しかし、MQL5 言語の主な利点は、エキスパートアドバイザーで使用するか、単に技術的な分析のために価格チャートに適用出来るカスタム指標を作成することです。

指標のセット全体はプロットとして知られるいくつかの基本的な描画スタイルから派生出来ます。プロットとは、指標が計算、保存、またリクエストに応じて提供するデータの表示の方法です。基礎的なプロットには 7 種類あります。

1. 線
2. セクション（セグメント）
3. ヒストグラム
4. 矢印（シンボル）
5. 塗りつぶされた領域（チャンネル）
6. バー
7. 日本製のローソク足

プロットは指標値を格納する1～ 5 のdouble 型の 配列を必要とします。便宜のために、これらの配列は、指標バッファに関連しています。指標内のバッファの数は、コンパイラディレクティブを使用して事前に宣言される必要があります。例として

```
#property indicator_buffers 3 // バッファの数
#property indicator_plots   2 // プロットの数
```

指標内のバッファ数は、常に指標におけるプロットの数以上です。

それぞれの基本プロットタイプはカラーバリエーションや建設仕様を持つことが出来るので、MQL5　での実際のプロットの種類の数は18です。

Plotting	説明	値バッファ	色バッファ
DRAW_NONE	チャートには非表示ですが、対応するバッファの値は、データウィンドウに表示することが出来ます。	1	-
DRAW_LINE	線は対応するバッファの値をプロットしています	1	-

© 2000–2025, MetaQuotes Ltd.

Plotting	説明	値バッファ	色バッファ
	（バッファ内の空の値は望ましくありません）。		
DRAW_SECTION	対応するバッファの値の間の線分として描画されます（通常は空の値をたくさん持っています）。	1	-
DRAW_HISTOGRAM	対応するバッファの値にゼロラインからヒストグラムとして描画されます（空の値を有していてもよいです）。	1	-
DRAW_HISTOGRAM2	2 つの指標バッファに基づいてヒストグラムとして描画されています（空の値を有していてもよいです）。	2	-
DRAW_ARROW	シンボルとして描画されています（空の値を有して	1	-

© 2000–2025, MetaQuotes Ltd.

Plotting	説明	値バッファ	色バッファ
	いてもよいです）。		
DRAW_ZIGZAG	DRAW_SECTION スタイルと似ていますが、1つのバーに垂直なセグメントをプロット出来ます。	2	-
DRAW_FILLING	日本製の線の間を塗りつぶします。対応する2つのバッファの値は、データウィンドウに表示されます。	2	-
DRAW_BARS	バーとして描画されます。対応する4つのバッファの値は、データウィンドウに表示されます。	4	-
DRAW_CANDLES	日本製のローソク足として描画されます。対応する4つのバッファの値は、データウィンドウに表示	4	-

© 2000–2025, MetaQuotes Ltd.

Plotting	説明	値バッファ	色バッファ
	されます。		
DRAW_COLOR_LINE	それぞれのバーに違う色を利用したり、任意の時点で色を変更出来ます。	1	1
DRAW_COLOR_SECTION	DRAW_SECTION スタイルと似ていますが、セクションの色はそれぞれ設定でき、動的に設定することも出来ます。	1	1
DRAW_COLOR_HISTOGRAM	DRAW_HISTOGRAM スタイルと似ていますが、ストリップの色はそれぞれ設定でき、動的に設定することも出来ます。	1	1
DRAW_COLOR_HISTOGRAM2	DRAW_HISTOGRAM2 スタイルと似ていますが、ストリップの色はそれぞれ設定	2	1

© 2000–2025, MetaQuotes Ltd.

Plotting	説明	値バッファ	色バッファ
	できる、動的に設定することも出来ます。		
DRAW_COLOR_ARROW	DRAW_ARROW スタイルと似ていますが、各シンボルの色をそれぞれ設定出来ます色は動的に設定することが出来ます。	1	1
DRAW_COLOR_ZIGZAG	セクションの個別な色の選択と動的な色の設定が可能な DRAW_ZIGZAG スタイル。	2	1
DRAW_COLOR_BARS	バーの個別な色の選択と動的な色の設定が可能な DRAW_BARS スタイル。	4	1
DRAW_COLOR_CANDLES	ローソク足の個別な色の選択と動的な色の設定が可能な DRAW_C	4	1

© 2000–2025, MetaQuotes Ltd.

Plotting	説明	値バッファ	色バッファ
	ANDLES スタイル。		

指標バッファと配列の違い

指標のグローバルレベルでは double 型の配列を宣言し、SetIndexBuffer() 関数を使用して指標バッファとして使用するべきです。指標プロットを描画するには、指標バッファの値のみが使用可能で一般の配列は使用出来ません。加えて、バッファ値はデータウィンドウに表示されます。

指標バッファは動的でサイズの指定を必要としません。指標バッファとして使用される配列のサイズは、自動的に端末の実行サブシステムによって設定されます。

配列が指標バッファと関係づけられた後、索引付けの方向がデフォルトでは通常の配列の様に設定されますが ArraySetAsSeries() 関数を使用して配列要素のアクセス方法を変更することが出来ます。デフォルトでは、指標バッファはプロットに使用されるデータ(INDICATOR_DATA)を格納します。

指標値の計算に中間計算の保持が必要、また各足に追加の値を格納する必要がある場合は、指標バッファは結合時に計算用バッファとして宣言することが出来ます(INDICATOR_CALCULATIONS)。中間値には普通の配列も使用することが出来ますが、この場合には、プログラマは、配列のサイズを管理しなければなりません。

プロットによっては、各足の色を個別に設定することが可能です。色に関する情報を格納するには、カラーバッファが使用されます(INDICATOR_COLOR_INDEX)。色は整数型の color ですが、指標バッファは double 型でなければなりません。色及び補助(INDICATOR_CALCULATIONS)バッファの値は CopyBuffer() で得ることが出来ません。

指標バッファの数はコンパイラディレクティブ #property indicator_buffers バッファ数 で指定されなければなりません。

```
#property indicator_buffers 3  //  指標には 3 つのバッファがある
```

1 つの指標のバッファの最大許容数は 512 です。

指標バッファとプロットの関連性

各プロットは、1つ以上の指標バッファに基づいています。単純なローソク足を表示するためには、始値、高値、安値、終値の 4 つの値が必要とされています。従って、指標をローソク足として表示するには、4 つの指標バッファとそのための 4 つの double 型の配列を宣言することが必要です。例えば、

```
//--- 指標は4つの指標のバッファを持つ
#property indicator_buffers 4
//--- 指標に 1 つのプロットがある
#property indicator_plots   1
//--- グラフィックプロット 1 はローソク足
#property indicator_type1   DRAW_CANDLES
//--- ローソク足が clrDodgerBlue に描画される
#property indicator_color1  clrDodgerBlue
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 指標バッファに使用される 4 つの配列
double OBuffer[];
double HBuffer[];
double LBuffer[];
double CBuffer[];
```

グラフィックプロットは自動的にプロット数に応じて指標バッファを使用します。プロット番号は1から始まり、バッファ番号はゼロから始まります。最初のプロットが 4 つの指標バッファを必要とする場合、最初の 4 つの指標バッファはそれを描画するために使用されます。これらの 4 つのバッファは、SetIndexBuffer() 関数を使用して、正しいインデックス作成と適切な配列と関連付けられる必要があります。

```
//--- 配列と指標バッファを関連付ける
   SetIndexBuffer(0,OBuffer,INDICATOR_DATA);  // インデックス0にあたる1 番目のバッファ
   SetIndexBuffer(1,HBuffer,INDICATOR_DATA);  // インデックス1にあたる2 番目のバッファ
   SetIndexBuffer(2,LBuffer,INDICATOR_DATA);  // インデックス2にあたる3 番目のバッファ
   SetIndexBuffer(3,CBuffer,INDICATOR_DATA);  // インデックス3にあたる4 番目のバッファ
```

「ローソク足」のプロットは、最初の番号の下で発表されたため、指標は、単に最初の 4 つのバッファを使用します。

例を変えて、簡単な線のプロットを加えましょう(DRAW_LINE)。仮に、線が 1 と番号付けされて、ローソク足が 2 だとします。バッファ数とプロット数が増加しました。

```
//--- 指標は 5 つの指標のバッファを持つ
#property indicator_buffers 5
//--- 指標に 2 つのプロットがある
#property indicator_plots   2
//--- プロット1は線
#property indicator_type1   DRAW_LINE
//--- 線の色は clrDodgerRed
#property indicator_color1  clrDodgerRed
//--- プロット 2 は日本製のローソク足
#property indicator_type2   DRAW_CANDLES
//--- ローソク足の色は clrDodgerBlue
#property indicator_color2  clrDodgerBlue
//--- 指標バッファである 5 つの配列
double LineBuffer[];
double OBuffer[];
double HBuffer[];
double LBuffer[];
double CBuffer[];
```

プロットの順序が変更され、線が初めで日本製のローソク足が続きます。そのため、バッファの順序は適切です。線のためバッファをゼロインデックスで宣言し、ローソク足のための 4 つのバッファを宣言します。

```
   SetIndexBuffer(0,LineBuffer,INDICATOR_DATA);  // インデックス0にあたる1 番目のバッファ
//--- ローソク足のために配列と指標バッファを関連付ける
   SetIndexBuffer(1,OBuffer,INDICATOR_DATA);     // インデックス1にあたる2 番目のバッファ
   SetIndexBuffer(2,HBuffer,INDICATOR_DATA);     // インデックス2にあたる3 番目のバッファ
   SetIndexBuffer(3,LBuffer,INDICATOR_DATA);     // インデックス3にあたる4 番目のバッファ
```

© 2000-2025, MetaQuotes Ltd.

```
   SetIndexBuffer(4,CBuffer,INDICATOR_DATA);      // インデックス4にあたる5 番目のバッファ
```

バッファ数とプロット数はコンパイラディレクティブを使用してのみ設定することができ、これらのプロパティを関数で動的に変更することは不可能です。

スタイルのカラーバージョン

表から分かるように、スタイルは、2 つのグループに分割されます。最初のグループでは、名称に COLOR が付きません。これらは基本スタイルと呼ばれます。

- DRAW_LINE
- DRAW_SECTION
- DRAW_HISTOGRAM
- DRAW_HISTOGRAM2
- DRAW_ARROW
- DRAW_ZIGZAG
- DRAW_FILLING
- DRAW_BARS
- DRAW_CANDLES

2 つ目のグループでは、名称に COLOR が付きます。これらを、カラーバージョンと呼びます。

- DRAW_COLOR_LINE
- DRAW_COLOR_SECTION
- DRAW_COLOR_HISTOGRAM
- DRAW_COLOR_HISTOGRAM2
- DRAW_COLOR_ARROW
- DRAW_COLOR_ZIGZAG
- DRAW_COLOR_BARS
- DRAW_COLOR_CANDLES

スタイルのカラーバージョンは、プロットの各部分の色の指定を可能にし、基本スタイルとは異なります。プロットの最小部分はバーなので、カラーバージョンは、各足の色の設定を可能にすると言うことが出来ます。

例外はカラーバージョンを持たない DRAW_NONE 及び DRAW_FILLING スタイルです。

各足のプロットの色を設定するために、カラーバージョンに、カラーインデックスを格納するためのバッファが追加されました。これらのインデックスは、前もって定義された色のセットが含まれている特殊な配列内の色番号を示します。色の配列のサイズは64です。これはスタイルの各カラーバージョンが64の異なる色のプロットを描画出来ることを意味します。

セットや色の特殊な配列内の色のセットと数は、コンパイラディレクティブ #property indicator_color でコンマで区切って、必要な全てのものを指定することが出来ます。指標のエントリの例として

```
//---- ローソク足を塗るために(特殊な配列に格納されている)8 色を定義する
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_color1
clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrLime,clrOrange
```

これは、1をプロットするために、8 色が設定され、これらが特殊な配列に配置されることを述べています。ここからはプログラムの中で、プロットの色の代わりにインデックスを指定します。バー番号 K に赤を指定されたい場合、配列での赤のインデックス値が指標のカラーバッファに設定される必要があります。赤は最初のディレクティブで指定されており、インデックス番号 0 に対応しています。

```
   //--- ローソク足の色を clrRed に設定
   col_buffer[buffer_index]=0;
```

色のセットは不変ではなく、PlotIndexSetInteger() を使用して動的に変更することが出来ます。例:

```
      //--- 各インデックスの色をPLOT_LINE_COLOR として設定する
      PlotIndexSetInteger(0,                          //   グラフィックスタイルの番号
                          PLOT_LINE_COLOR,            //   プロパティの識別子
                          plot_color_ind,             //   色を書き込む色インデックス
                          color_array[i]);            //   新しい色
```

指標プロパティとプロット

指標のプロットでは、プロパティはコンパイラディレクティブまたは適切な関数を使用して設定出来ます。詳しくは指標プロパティと関数との関係をご参照ください。特別な関数を使用しての指標プロパティの動的な変更は、より柔軟なカスタム指標作成を可能にします。

チャート上での指標描画の開始

多くの例では、アルゴリズムの条件に応じて、最小限の履歴バーを提供する必要があるため、現在足のみを使用してすぐに指標値の計算を開始することは不可能です。例えば、平滑化の多くの種類は、以前の N バーの価格配列の使用を暗示し、これらの値に基づいて、現在足の指標値を算出します。

このような場合には、最初の N バーの指標値を計算する方法がないか、これらの値は計算のみに使用されるものでチャート上での表示は意図外です。履歴中の最初の N 個のバーでの指標のプロットを回避するためには、プロットのN を PLOT_DRAW_BEGIN プロパティに設定します。

```
//--- ローソク足のために配列と指標バッファを関連付ける
PlotIndexSetInteger(number_of_plot,PLOT_DRAW_BEGIN,N);
```

ここで

- number_of_plot – ゼロから indicator_plots-1 までの値(プロット番号はゼロから始まる)
- N – 指標がチャート上に表示されるべきではない履歴中の最初のバーの数

© 2000-2025, MetaQuotes Ltd.

DRAW_NONE

DRAW_NONE スタイルは、バッファ値の計算とデータウィンドウでの表示の必要があるがチャートでのプロットが必要とされない場合の使用に設計されています。精度を設定するには OnInit() 関数の iIndicatorSetInteger(INDICATOR_DIGITS,num_chars) を使用します。

```
int OnInit()
  {
//--- 指標バッファマッピング
   SetIndexBuffer(0,InvisibleBuffer,INDICATOR_DATA);
//--- データウィンドウに表示される値の精度を設定する
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
   return(INIT_SUCCEEDED);
  }
```

DRAW_NONE のプロットに必要なバッファの数は 1 です。

データウィンドウでマウスが現在ホバーしているバーの番号を示す指標の例です。番号は時系列に対応し、現在未完成のバーがゼロのインデックスを持ち、最も古いバーが最大のインデックスを持ちます。



#1のプロットが赤に設定されているにもかかわらずチャート上では指標が描画されないことにご注意下さい。

```
//+------------------------------------------------------------------+
//|                                                   DRAW_NONE.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
```

© 2000–2025, MetaQuotes Ltd.

```
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- 目に見えないプロット
#property indicator_label1  "Bar Index"
#property indicator_type1   DRAW_NONE
#property indicator_style1  STYLE_SOLID
#property indicator_color1  clrRed
#property indicator_width1  1
//--- 指標バッファ
double         InvisibleBuffer[];
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列と指標バッファを関連付ける
   SetIndexBuffer(0,InvisibleBuffer,INDICATOR_DATA);
//--- データウィンドウに表示される値の精度を設定する
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static datetime lastbar=0;
//--- 指標の初めての計算の場合
   if(prev_calculated==0)
     {
      //--- バーの番号を始めて変更
      CalcValues(rates_total,close);
      //--- lastbar で現在足のオープン時間を保存
      lastbar=(datetime)SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE);
     }
   else
     {
```

© 2000–2025, MetaQuotes Ltd.

```
      //--- 新しいバーが出来た場合、オープン時間が lastbar と異なる
      if(lastbar!=SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE))
        {
         //--- バーの番号をまた変更
         CalcValues(rates_total,close);
         //--- 現在足のオープン時間を lastbar で更新
         lastbar=(datetime)SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE);
        }
      }
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| 時系列のようにバーを番号付け                                     |
//+------------------------------------------------------------------+
void CalcValues(int total,double const  &array[])
  {
//--- 時系列のように指標バッファの索引付けを設定
   ArraySetAsSeries(InvisibleBuffer,true);
//--- それぞれのバーに番号を書き込む
   for(int i=0;i<total;i++) InvisibleBuffer[i]=i;
  }
```

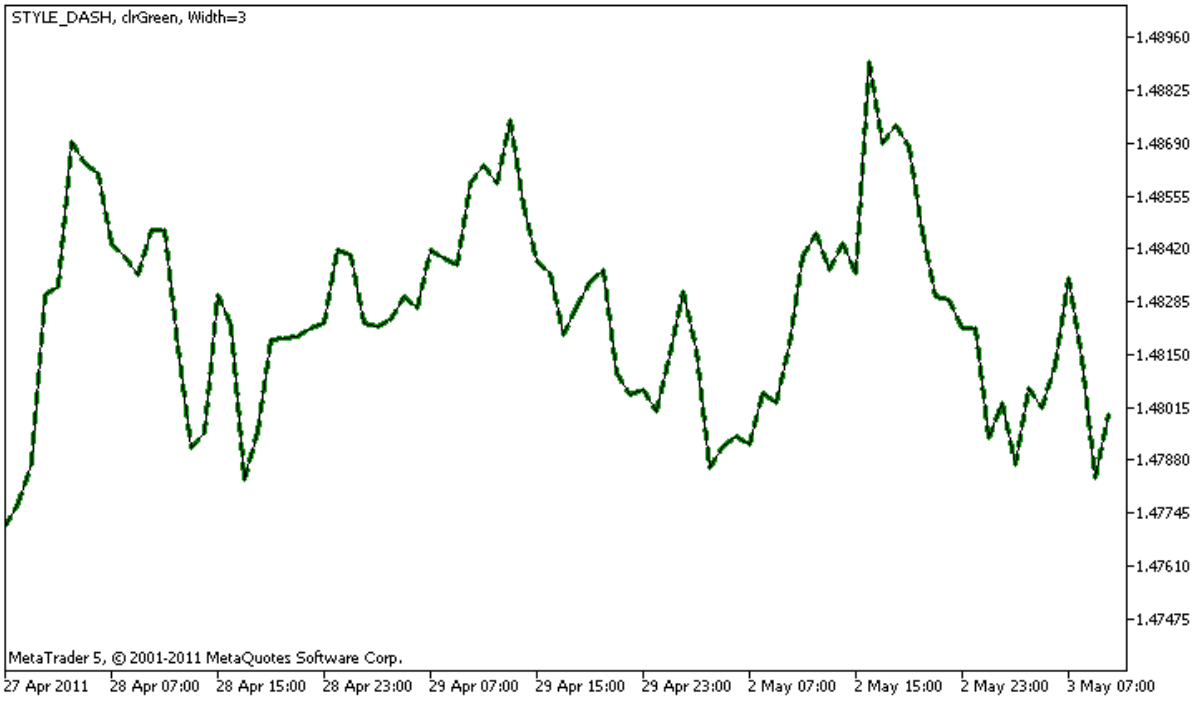
© 2000-2025, MetaQuotes Ltd.

DRAW_LINE

DRAW_LINE は指標バッファの値によって指定された色の線を描画します。線の幅、スタイルと色はコンパイラディレクティブまたは動的に PlotIndexSetInteger() 関数を使用して指定することが出来ます。プロットのプロパティの動的な変更は、指標の外観が現在の状況に応じて変わるようにして「盛り上げる」ことが出来ます。

DRAW_LINE のプロットに必要なバッファの数は 1 です。

バーの終値を使用して線を描画する指標の例です。線の色、幅とスタイルは N=5 ティックごとにランダムに変更されます。



DRAW_LINE を持つ **plot1**では、3 つのプロパティは #property コンパイラディレクティブを使用して指定され、後に OnCalculate() 関数でランダムに設定されることにご注意ください。N パラメータは手動設定の可能性を持って(指標の「プロパティ」ウィンドウの「パラメータ」タブ)指標外部パラメータに設定されています。

```
//+------------------------------------------------------------------+
//|                                                    DRAW_LINE.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "An indicator to demonstrate DRAW_LINE"
#property description "It draws a line of a specified color at Close prices"
#property description "Color, width and style of lines is changed randomly"
#property description "after every N ticks"

#property indicator_chart_window
```

© 2000–2025, MetaQuotes Ltd.

```
#property indicator_buffers 1
#property indicator_plots   1
//--- 線のプロパティはコンパイラディレクティブで設定される
#property indicator_label1  "Line"       // データウィンドウでのプロット名
#property indicator_type1   DRAW_LINE    // プロットの種類は「線」
#property indicator_color1  clrRed       // 線の色
#property indicator_style1  STYLE_SOLID // 線のスタイル
#property indicator_width1  1            // 線の幅
//--- 入力パラメータ
input int       N=5;          // 変化をもたらすティックの数
//--- プロットの指標バッファ
double          LineBuffer[];
//--- 色を格納する配列
color colors[]={clrRed,clrBlue,clrGreen};
//--- 線のスタイルを格納する配列
ENUM_LINE_STYLE styles[]
={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOTDOT};
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列と指標バッファを関連付ける
   SetIndexBuffer(0,LineBuffer,INDICATOR_DATA);
//--- 擬似乱数生成器を初期化
   MathSrand(GetTickCount());
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//|  カスタム指標の反復関数                                          |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- 線のスタイル、色、及び幅を変更するティックを計算する
   ticks++;
//--- 充分なティックの数が蓄積されている場合
   if(ticks>=N)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 線のプロパティを変更する
      ChangeLineAppearance();
      //--- ティックカウンタをゼロにリセットする
      ticks=0;
     }

//--- 指標安倍を計算するブロック
   for(int i=0;i<rates_total;i++)
     {
      LineBuffer[i]=close[i];
     }

//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| 指標で描画された線の外観を変更する                                |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- 線のプロパティに関する情報を形成するための文字列
   string comm="";
//--- 線の色を変更するブロック
//--- 乱数を取得
   int number=MathRand();
//--- 除数は colors[] 配列のサイズと同じ
   int size=ArraySize(colors);
//--- 新しい色を選択するためのインデックスを整数除算の余りから取得
   int color_index=number%size;
//--- 色をPLOT_LINE_COLOR プロパティとして設定
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- 線の色を書く
   comm=comm+(string)colors[color_index];

//--- 線の幅を変更するブロック
   number=MathRand();
//--- 整数除算の余りの幅を取得
   int width=number%5; // 幅は 0～4 に設定される
//--- 色を PLOT_LINE_WIDTH プロパティに設定
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- 線の幅を書く
   comm=comm+", Width="+IntegerToString(width);

//--- 線のスタイルを変更するブロック
   number=MathRand();
//--- 除数は、スタイルの配列の大きさに等しい
   size=ArraySize(styles);
//--- 新しいスタイルを選択するためのインデックスを整数除算の余りから取得
   int style_index=number%size;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 色をPLOT_LINE_COLOR プロパティとして設定
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- 線のスタイルを書く
   comm=EnumToString(styles[style_index])+", "+comm;
//--- コメントを使用して、チャート上の情報を表示する
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_SECTION

DRAW_SECTION はセクションを指標バッファの値によって指定された色で描画します。線の幅、色とスタイルは DRAW_LINEスタイルのようにコンパイラディレクティブまたは 動的に PlotIndexSetInteger() 関数を使用して指定することが出来ます。プロットのプロパティの動的な変更は、指標の外観が現在の状況に応じて変わるようにして「盛り上げる」ことが出来ます。

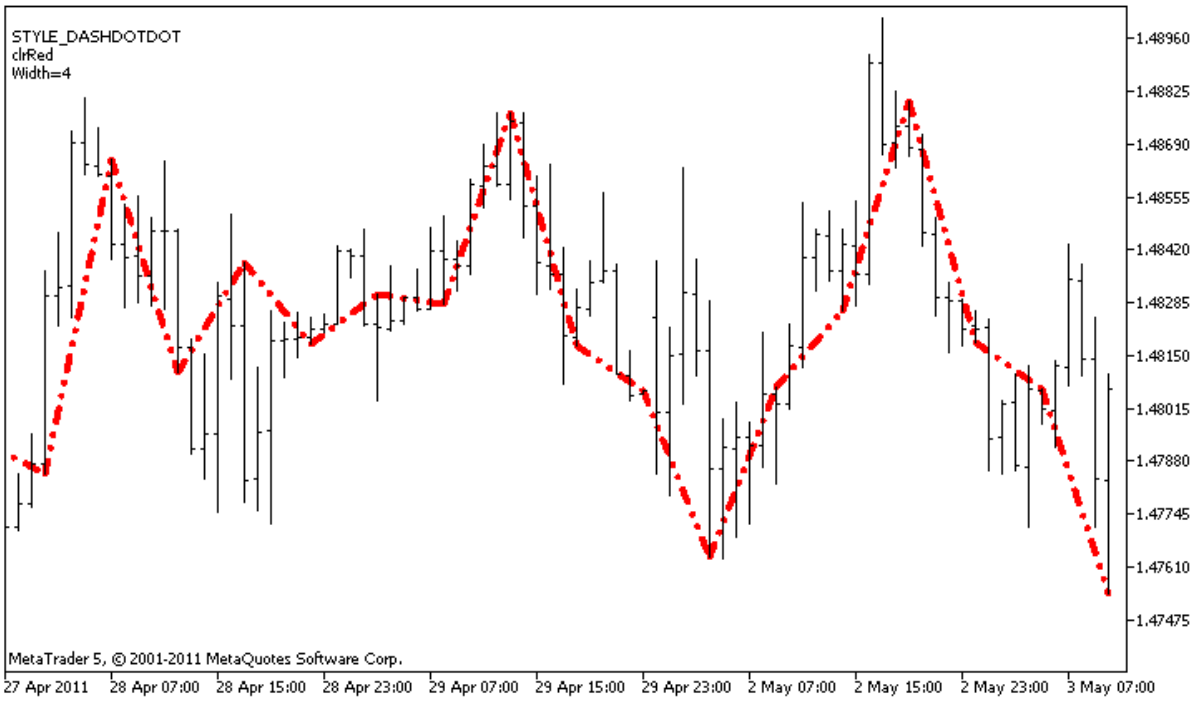
セクションは指標バッファの空でない値からあと 1 つの空でない値まで描かれ、空の値は無視されます。「空」値を指定するには PLOT_EMPTY_VALUE プロパティで値を設定します。例えば、指標をゼロ以外の値のセクションのシーケンスとして描画する必要がある場合、ゼロを空値として設定する必要があります。

```
//--- 0(空)値は描画されない
   PlotIndexSetDouble(index_of_plot_DRAW_SECTION,PLOT_EMPTY_VALUE,0);
```

バッファの値は常に明示的に入力し、描画されないべき要素ではバッファに空の値を設定します。

DRAW_SECTION のプロットに必要なバッファの数は 1 です。

高値と安値の間のセクションを描画する指標の例です。セクションの色、幅とスタイルは全て N ティックごとにランダムに変更されます。



DRAW_SECTION を持つ**plot1** では、3 つのプロパティは #property コンパイラディレクティブを使用して指定され、後に OnCalculate() 関数でランダムに設定されることにご注意ください。N パラメータは手動設定の可能性を持って(指標の「プロパティ」ウィンドウの「パラメータ」タブ)指標外部パラメータに設定されています。

```
//+------------------------------------------------------------------+
//|                                                DRAW_SECTION.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
```

© 2000-2025, MetaQuotes Ltd.

```
#property version   "1.00"

#property description "An indicator to demonstrate DRAW_SECTION"
#property description "Draws straight sections every bars bars"
#property description "The color, width and style of sections are changed randomly"
#property description "after every N ticks"

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- セクションをプロットする
#property indicator_label1  "Section"
#property indicator_type1   DRAW_SECTION
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 入力パラメータ
input int      bars=5;           // バー単位でのセクションの長さ
input int      N=5;              // セクションスタイルを変更するティック数
//--- プロットの指標バッファ
double         SectionBuffer[];
//--- セクションの端を計算する補助変数
int            divider;
//--- 色を格納する配列
color colors[]={clrRed,clrBlue,clrGreen};
//--- 線のスタイルを格納する配列
ENUM_LINE_STYLE styles[]
={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOTDOT};
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列と指標バッファを関連付ける
   SetIndexBuffer(0,SectionBuffer,INDICATOR_DATA);
//--- 0(空)値は描画されない
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- 指標パラメータをチェックする
   if(bars<=0)
     {
      PrintFormat("Invalid value of parameter bar=%d",bars);
      return(INIT_PARAMETERS_INCORRECT);
     }
   else divider=2*bars;
//---+
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//|  カスタム指標の反復関数                                          |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- 線のスタイル、色、及び幅を変更するティックを計算する
   ticks++;
//--- 充分なティックの数が蓄積されている場合
   if(ticks>=N)
     {
      //--- 線のプロパティを変更する
      ChangeLineAppearance();
      //--- ティックカウンタをゼロにリセットする
      ticks=0;
     }

//--- 指標値の算出が開始されるバーの番号
   int start=0;
//--- 指標が以前に計算されている場合には、開始点を前のバーに設定する
   if(prev_calculated>0) start=prev_calculated-1;
//--- 指標値の計算
   for(int i=start;i<rates_total;i++)
     {
      //--- バー番号を 2*bars で割って余りを取得
      int rest=i%divider;
      //--- バー番号が 2*bars で割りきれる場合
      if(rest==0)
        {
         //--- この足の高値でセクションの終わりを設定する
         SectionBuffer[i]=high[i];
        }
      //---除算の剰余は、バーに等しい場合
      else
        {
         //--- この足の高値でセクションの終わりを設定する
         if(rest==bars) SectionBuffer[i]=low[i];
         //--- 何もない場合、バーを無視して 0 に設定
         else SectionBuffer[i]=0;
        }
     }
//--- prev_calculated 値を次の関数呼び出しのために返す
```

© 2000-2025, MetaQuotes Ltd.

```
   return(rates_total);
  }
//+------------------------------------------------------------------+
//|  指標のセクションの外観を変更する                                |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- 線のプロパティに関する情報を形成するための文字列
   string comm="";
//--- 線の色を変更するブロック
   int number=MathRand(); // 乱数を取得
//--- 除数は colors[] 配列のサイズと同じ
   int size=ArraySize(colors);
//--- 新しい色を選択するためのインデックスを整数除算の余りから取得
   int color_index=number%size;
//--- 色をPLOT_LINE_COLOR プロパティとして設定
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- 線の色を書く
   comm=comm+"\r\n"+(string)colors[color_index];

//--- 線の幅を変更するブロック
   number=MathRand();
//--- 整数除算の余りの幅を取得
   int width=number%5;    // 幅は 0～4 に設定される
//--- 幅を設定
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- 線の幅を書く
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- 線のスタイルを変更するブロック
   number=MathRand();
//--- 除数は、スタイルの配列の大きさに等しい
   size=ArraySize(styles);
//--- 新しいスタイルを選択するためのインデックスを整数除算の余りから取得
   int style_index=number%size;
//--- 線のスタイルを設定
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- 線のスタイルを書く
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- コメントを使用して、チャート上の情報を表示する
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

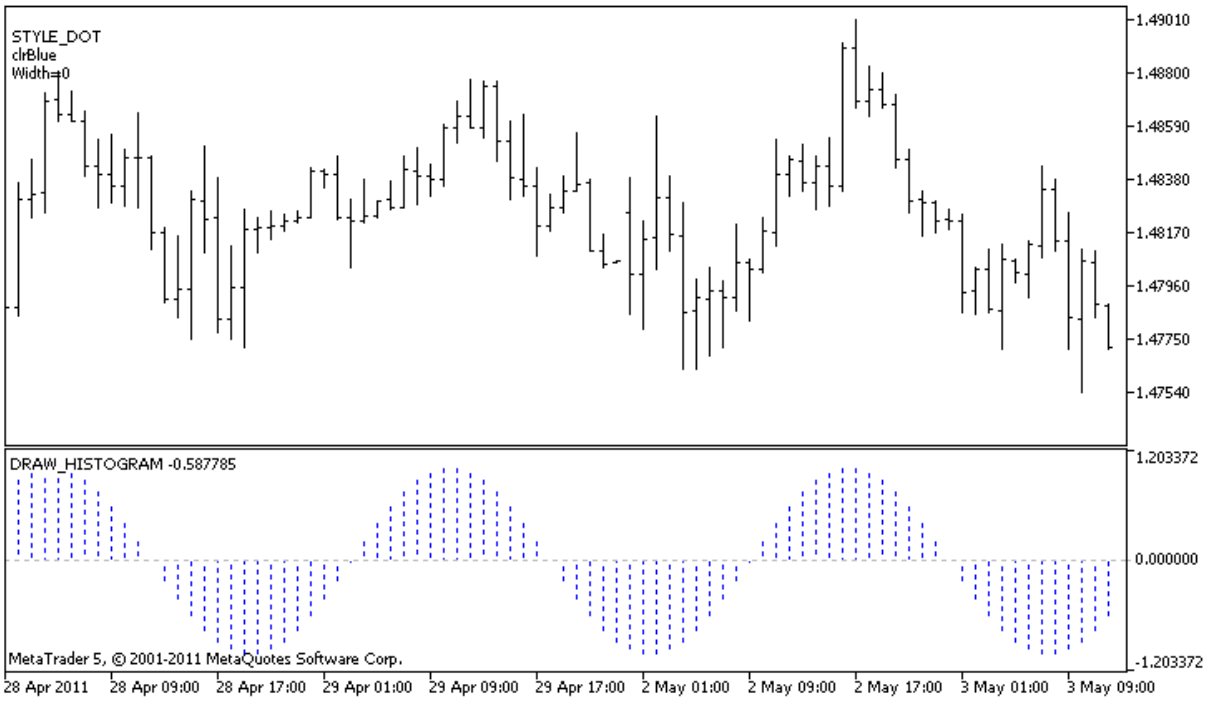
DRAW_HISTOGRAM

DRAW_HISTOGRAM スタイルはゼロから指定された値に列の順序として、指定された色のヒストグラムを描画します。値は指標バッファから取られます。列の幅、色とスタイルは DRAW_LINE スタイルのようにコンパイラディレクティブまたは動的に PlotIndexSetInteger() 関数を使用して指定することが出来ます。プロットプロパティの動的な変更は、現在の状況に基づいてのヒストグラムの外観の変更を可能にします。

ゼロレベルからの列が各足の上に描画されているので、DRAW_HISTOGRAM は、別のチャートウィンドウで使用される必要があります。ほとんどの場合、この種類のプロットは、Bears Power または OsMA のようなオシレーター型の指標を作成するために使用されます。空の表示不可能な値には、ゼロ値を指定する必要があります。

DRAW_HISTOGRAM のプロットに必要なバッファの数は 1 です。

MathSin() 関数に基づいて指定された色の正弦波を描く指標の例です。ヒストグラムの列の色、幅とスタイルは全て N ティックごとにランダムに変更されます。bars パラメータは正弦波の期間を指定し、バーの指定された数の後で、正弦波のサイクルが繰り返すことになります。



DRAW_HISTOGRAM を持つplot1 では、3 つのプロパティは #propertyコンパイラディレクティブを使用して指定され、後にOnCalculate() 関数でランダムに設定されることにご注意ください。N パラメータは手動設定の可能性を持って(指標の「プロパティ」ウィンドウの「パラメータ」タブ)指標外部パラメータに設定されています。

```
//+------------------------------------------------------------------+
//|                                                DRAW_HISTOGRAM.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

© 2000–2025, MetaQuotes Ltd.

```
#property description "An indicator to demonstrate DRAW_HISTOGRAM"
#property description "It draws a sinusoid as a histogram in a separate window"
#property description "The color and width of columns are changed randomly"
#property description "after every N ticks"
#property description "The bars parameter sets the number of bars in the cycle of
the sinusoid"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- ヒストグラムをプロットする
#property indicator_label1  "Histogram"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 入力パラメータ
input int      bars=30;          // バーでの正弦波の期間
input int      N=5;              // ヒストグラムを変更するティック数
//--- 指標バッファ
double         HistogramBuffer[];
//--- bars パラメータを掛けた時ラジアンで2PI 角を取得する要因
double    multiplier;
//--- 色を格納する配列
color colors[]={clrRed,clrBlue,clrGreen};
//--- 線のスタイルを格納する配列
ENUM_LINE_STYLE styles[]
={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOTDOT};
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 指標バッファマッピング
   SetIndexBuffer(0,HistogramBuffer,INDICATOR_DATA);
//--- 乗数を計算する
   if(bars>1)multiplier=2.*M_PI/bars;
   else
     {
      PrintFormat("Set the value of bars=%d greater than 1",bars);
      //--- 指標の早期終了
      return(INIT_PARAMETERS_INCORRECT);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- 線のスタイル、色、及び幅を変更するティックを計算する
   ticks++;
//--- 充分なティックの数が蓄積されている場合
   if(ticks>=N)
     {
      //--- 線のプロパティを変更する
      ChangeLineAppearance();
      //--- ティックカウンタをゼロにリセットする
      ticks=0;
     }

//--- 指標値を計算する
   int start=0;
//--- すでにOnCalculate の前回の開始時に算出した場合
   if(prev_calculated>0) start=prev_calculated-1; // 最後から2 つ目のバーから計算する
//--- 指標バッファに値を記入する
   for(int i=start;i<rates_total;i++)
     {
      HistogramBuffer[i]=sin(i*multiplier);
     }
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//|  指標での線の外観を変更する                                      |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- 線のプロパティに関する情報を形成するための文字列
   string comm="";
//--- 線の色を変更するブロック
   int number=MathRand(); // 乱数を取得
//--- 除数は colors[] 配列のサイズと同じ
   int size=ArraySize(colors);
//--- 新しい色を選択するためのインデックスを整数除算の余りから取得
   int color_index=number%size;
//--- 色をPLOT_LINE_COLOR プロパティとして設定
```

© 2000-2025, MetaQuotes Ltd.

```
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- 線の色を書く
   comm=comm+"\r\n"+(string)colors[color_index];

//--- 線の幅を変更するブロック
   number=MathRand();
//--- 整数除算の余りの幅を取得
   int width=number%5;    // 幅は 0～4 に設定される
//--- 幅を設定
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- 線の幅を書く
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- 線のスタイルを変更するブロック
   number=MathRand();
//--- 除数は、スタイルの配列の大きさに等しい
   size=ArraySize(styles);
//--- 新しいスタイルを選択するためのインデックスを整数除算の余りから取得
   int style_index=number%size;
//--- 線のスタイルを設定
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- 線のスタイルを書く
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- コメントを使用して、チャート上の情報を表示する
   Comment(comm);
  }
```

© 2000–2025, MetaQuotes Ltd.

DRAW_HISTOGRAM2

DRAW_HISTOGRAM2 スタイルは、指定された色のヒストグラム（2 つの指標バッファの値を使用した垂直セグメント）を描画します。セグメントの幅、色とスタイルは DRAW_LINE スタイルのようにコンパイラディレクティブまたは 動的に PlotIndexSetInteger() 関数を使用して指定することが出来ます。プロットプロパティの動的な変更は、現在の状況に基づいてのヒストグラムの外観の変更を可能にします。

DRAW_HISTOGRAM2 スタイルはチャートの別々のサブウィンドウとメインウィンドウで使用することが出来ます。空の値は描画されません。指標バッファ内の全ての値は明示的に設定する必要があります。バッファはゼロで初期化されません。

DRAW_HISTOGRAM2 のプロットに必要なバッファの数は 2 です。

各足の始値と終値の間を指定された色と幅をもった垂直セグメントとしてプロットする指標の例です。ヒストグラムの列の色、幅とスタイルは 全て N ティックごとにランダムに変更されます。指標の始めに OnInit() 関数内で、ヒストグラムが描画されない invisible_day の数がランダムに設定されます。このために、空の値は PLOT_EMPTY_VALUE=0 に設定されます。

```
//--- 空の値を設定
   PlotIndexSetDouble(index_of_plot_DRAW_SECTION,PLOT_EMPTY_VALUE,0);
```



DRAW_HISTOGRAM2 を持つ**plot1** では、3 つのプロパティは #property コンパイラディレクティブを使用して指定され、後に OnCalculate() 関数でランダムに設定されることにご注意ください。N パラメータは手動設定の可能性を持って（指標の「プロパティ」ウィンドウの「パラメータ」タブ）指標外部パラメータに設定されています。

```
//+------------------------------------------------------------------+
//|                                                DRAW_HISTOGRAM2.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
```

© 2000–2025, MetaQuotes Ltd.

```
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "An indicator to demonstrate DRAW_HISTOGRAM2"
#property description "It draws a segment between Open and Close on each bar"
#property description "The color, width and style are changed randomly"
#property description "after every N ticks"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- Histogram_2 をプロットする
#property indicator_label1  "Histogram_2"
#property indicator_type1   DRAW_HISTOGRAM2
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 入力パラメータ
input int       N=5;              // ヒストグラムを変更するティック数
//--- 指標バッファ
double          Histogram_2Buffer1[];
double          Histogram_2Buffer2[];
//--- 指標がプロットされない曜日
int invisible_day;
//--- 色を格納する配列
color colors[]={clrRed,clrBlue,clrGreen};
//--- 線のスタイルを格納する配列
ENUM_LINE_STYLE styles[]
={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOTDOT};
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 指標バッファマッピング
   SetIndexBuffer(0,Histogram_2Buffer1,INDICATOR_DATA);
   SetIndexBuffer(1,Histogram_2Buffer2,INDICATOR_DATA);
//--- 空の値を設定
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- 0～ 5 の乱数を取得
   invisible_day=MathRand()%6;
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
```

© 2000–2025, MetaQuotes Ltd.

```
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- 線のスタイル、色、及び幅を変更するティックを計算する
   ticks++;
//--- 充分なティックの数が蓄積されている場合
   if(ticks>=N)
     {
      //--- 線のプロパティを変更する
      ChangeLineAppearance();
      //--- ティックカウンタをゼロにリセットする
      ticks=0;
     }

//--- 指標値を計算する
   int start=0;
//--- 各足の始値で曜日を取得する
   MqlDateTime dt;
//--- すでにOnCalculate の前回の開始時に算出した場合
   if(prev_calculated>0) start=prev_calculated-1; // 最後から2 つ目のバーから計算する
//--- 指標バッファに値を記入する
   for(int i=start;i<rates_total;i++)
     {
      TimeToStruct(time[i],dt);
      if(dt.day_of_week==invisible_day)
        {
         Histogram_2Buffer1[i]=0;
         Histogram_2Buffer2[i]=0;
        }
      else
        {
         Histogram_2Buffer1[i]=open[i];
         Histogram_2Buffer2[i]=close[i];
        }
     }
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//|  指標での線の外観を変更する                                      |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
```

© 2000–2025, MetaQuotes Ltd.

```
  {
//--- 線のプロパティに関する情報を形成するための文字列
   string comm="";
//--- 線の色を変更するブロック
   int number=MathRand(); // 乱数を取得
//--- 除数は colors[] 配列のサイズと同じ
   int size=ArraySize(colors);
//--- 新しい色を選択するためのインデックスを整数除算の余りから取得
   int color_index=number%size;
//--- 色をPLOT_LINE_COLOR プロパティとして設定
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- 線の色を書く
   comm=comm+"\r\n"+(string)colors[color_index];

//--- 線の幅を変更するブロック
   number=MathRand();
//--- 整数除算の余りの幅を取得
   int width=number%5;    // 幅は 0～4 に設定される
//--- 線の幅を設定
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- 線の幅を書く
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- 線のスタイルを変更するブロック
   number=MathRand();
//--- 除数は、スタイルの配列の大きさに等しい
   size=ArraySize(styles);
//--- 新しいスタイルを選択するためのインデックスを整数除算の余りから取得
   int style_index=number%size;
//--- 線のスタイルを設定
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- 線のスタイルを書く
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- 計算では省略された日についての情報を追加する
   comm="\r\nNot plotted day - "+EnumToString((ENUM_DAY_OF_WEEK)invisible_day)+comm;
//--- コメントを使用して、チャート上の情報を表示する
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_ARROW

DRAW_ARROW スタイルは指標バッファの値に基づいて指定された色の矢印（Wingdingsフォントシンボルのセット）を描画します。シンボルの幅と色はDRAW_LINE スタイルの様にコンパイラディレクティブまたは 動的にPlotIndexSetInteger() 関数を使用して指定することが出来ます。プロットプロパティの動的な変更は、現在の状況に基づいての指標の外観の変更を可能にします。

シンボルコードは PLOT_ARROW プロパティを使用して設定されます。

```
//--- PLOT_ARROW の描画のためにWingdings フォントからのシンボルコードを定義する
   PlotIndexSetInteger(0,PLOT_ARROW,code);
```

PLOT_ARROW=159（丸）がデフォルトです。

矢印は、実際には、高さとアンカーポイントを持つシンボルであり、チャート上のいくつかの重要な情報（例えばバーの終値）をカバーすることが出来ます。従って、チャートの規模に依存しないピクセル単位での垂直シフトの指定が可能です。指標値はそのままですが、矢印が特定のピクセル数だけ下に移動します。

```
//--- 矢印の垂直シフトをピクセル単位で設定する
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
```

PLOT_ARROW_SHIFT の負の値は矢印の上向きのシフト、正の値は矢印の下向きのシフトを意味します。

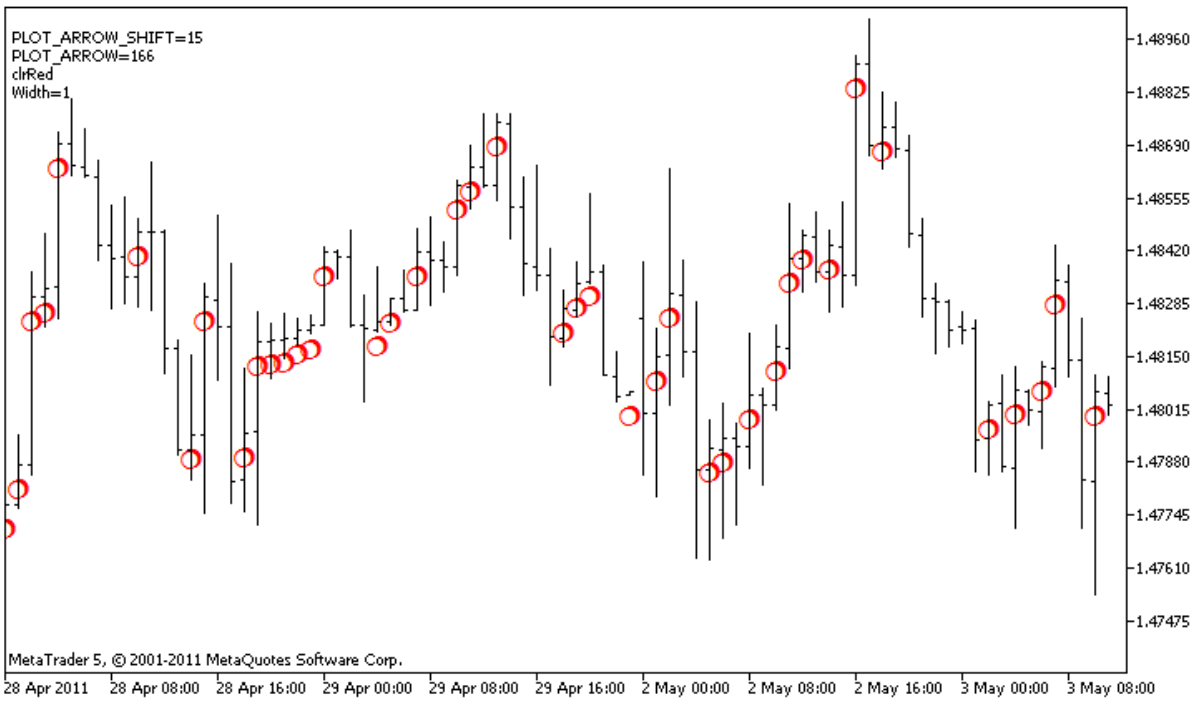
DRAW_ARROW スタイルはチャートの別々のサブウィンドウとメインウィンドウで使用することが出来ます。空の値は描画されず「データウィンドウ」に表示されません。指標バッファ内の全ての値は明示的に設定する必要があります。バッファはゼロで初期化されません。

```
//--- 空の値を設定
   PlotIndexSetDouble(index_of_plot_DRAW_ARROW,PLOT_EMPTY_VALUE,0);
```

DRAW_ARROW のプロットに必要なバッファの数は 1 です。

前のバーの終値よりも高い終値で各足の矢印を描画する指標の例です。全ての矢印の色、幅、シフトとシンボルコードは、N ティックごとにランダムに変更されます。

© 2000-2025, MetaQuotes Ltd.



この例では、DRAW_ARROW スタイルを持つ**plot1** では、プロパティ、色とサイズは #property コンパイラディレクティブを使用して指定され、プロパティは OnCalculate() 関数でランダムに設定されます。N パラメータは手動設定の可能性を持って（指標の「プロパティ」ウィンドウの「パラメータ」タブ）指標外部パラメータに設定されています。

```
//+------------------------------------------------------------------+
//|                                                   DRAW_ARROW.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "An indicator to demonstrate DRAW_ARROW"
#property description "Draws arrows set by Unicode characters, on a chart"
#property description "The color, size, shift and symbol code of the arrow are
changed in a random way"
#property description "after every N ticks"
#property description "The code parameter sets the base value: code=159 (a circle)"

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- 矢印をプロットする
#property indicator_label1  "Arrows"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrGreen
#property indicator_width1  1
//--- 入力パラメータ
```

© 2000–2025, MetaQuotes Ltd.

```
input int       N=5;          // 変化をもたらすティックの数
input ushort    code=159;     // DRAW_ARROW で描画するシンボルコード
//--- プロットの指標バッファ
double          ArrowsBuffer[];
//--- 色を格納する配列
color colors[]={clrRed,clrBlue,clrGreen};
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 指標バッファマッピング
   SetIndexBuffer(0,ArrowsBuffer,INDICATOR_DATA);
//--- PLOT_ARROW の描画のためにシンボルコードを定義する
   PlotIndexSetInteger(0,PLOT_ARROW,code);
//--- 矢印の垂直シフトをピクセル単位で設定する
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,5);
//--- 空の値の 0 を設定
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- 矢印の色、大きさ、及びシフトコードを変更するティックを計算する
   ticks++;
//--- 充分なティックの数が蓄積されている場合
   if(ticks>=N)
     {
      //--- 線のプロパティを変更する
      ChangeLineAppearance();
      //--- ティックカウンタをゼロにリセットする
      ticks=0;
     }

//--- 指標安倍を計算するブロック
```

© 2000-2025, MetaQuotes Ltd.

```
   int start=1;
   if(prev_calculated>0) start=prev_calculated-1;
//--- 計算ループ
   for(int i=1;i<rates_total;i++)
     {
      //--- 現在の終値が以前の終値より高い場合、矢印を描画する
      if(close[i]>close[i-1])
         ArrowsBuffer[i]=close[i];
      //--- 他の場合、ゼロを指定する
      else
         ArrowsBuffer[i]=0;
     }
//--- 次の呼び出しのためにprev_calculated の値を返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| 指標でのシンボルの外観を変更する                                 |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- 指標プロパティに関する情報を形成するための文字列
   string comm="";
//--- 矢印の色を変更するブロック
   int number=MathRand(); // 乱数を取得
//--- 除数はcolors[] 配列のサイズと同じ
   int size=ArraySize(colors);
//--- 新しい色を選択するためのインデックスを整数除算の余りから取得
   int color_index=number%size;
//--- 色をPLOT_LINE_COLOR プロパティとして設定
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- 線の色を書く
   comm=comm+"\r\n"+(string)colors[color_index];

//--- 矢印のサイズを変更するブロック
   number=MathRand();
//--- 整数除算の余りの幅を取得
   int width=number%5;    // サイズは0～ 4 に設定
//--- 色を PLOT_LINE_WIDTH プロパティに設定
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- 矢印のサイズを書く
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- 矢印のコードを変更するブロック(PLOT_ARROW)
   number=MathRand();
//--- 矢印の新しいコードを計算するために、整数の除算の剰余を取得(0～19)
   int code_add=number%20;
//--- 新しいシンボルコードをcode+code_add に設定
   PlotIndexSetInteger(0,PLOT_ARROW,code+code_add);
//--- PLOT_ARROW シンボルコードを書く
```

© 2000-2025, MetaQuotes Ltd.

```
   comm="\r\n"+"PLOT_ARROW="+IntegerToString(code+code_add)+comm;

//--- 矢印の垂直シフトをピクセル単位で変更するブロック
   number=MathRand();
//--- 整数除算の余りとしてシフトを取得
   int shift=20-number%41;
//--- 新しいシフトを-20～20で取得
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
//--- PLOT_ARROW_SHIFT シフトを書く
   comm="\r\n"+"PLOT_ARROW_SHIFT="+IntegerToString(shift)+comm;

//--- コメントを使用して、チャート上の情報を表示する
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_ZIGZAG

DRAW_ZIGZAG スタイルは 2 つの指標バッファの値に基づいてセグメントを指定された色で描画します。このスタイルは DRAW_SECTION ととても似ていますが、それと違って、両方の指標バッファの値が、このバーに設定されている場合には、1つのバー内の垂直セグメントを描画することが出来ます。セグメントは、1 番目のバッファの値から2 番目の指標バッファの値にプロットされます。いずれのバッファも空の値のみを含むことは出来ません。この場合プロットがなされません。

線の幅、色とスタイルは DRAW_SECTION スタイルのようにコンパイラディレクティブまたは 動的に PlotIndexSetInteger() 関数を使用して指定することが出来ます。プロットのプロパティの動的な変更は、指標の外観が現在の状況に応じて変わるようにして「盛り上げる」ことが出来ます。

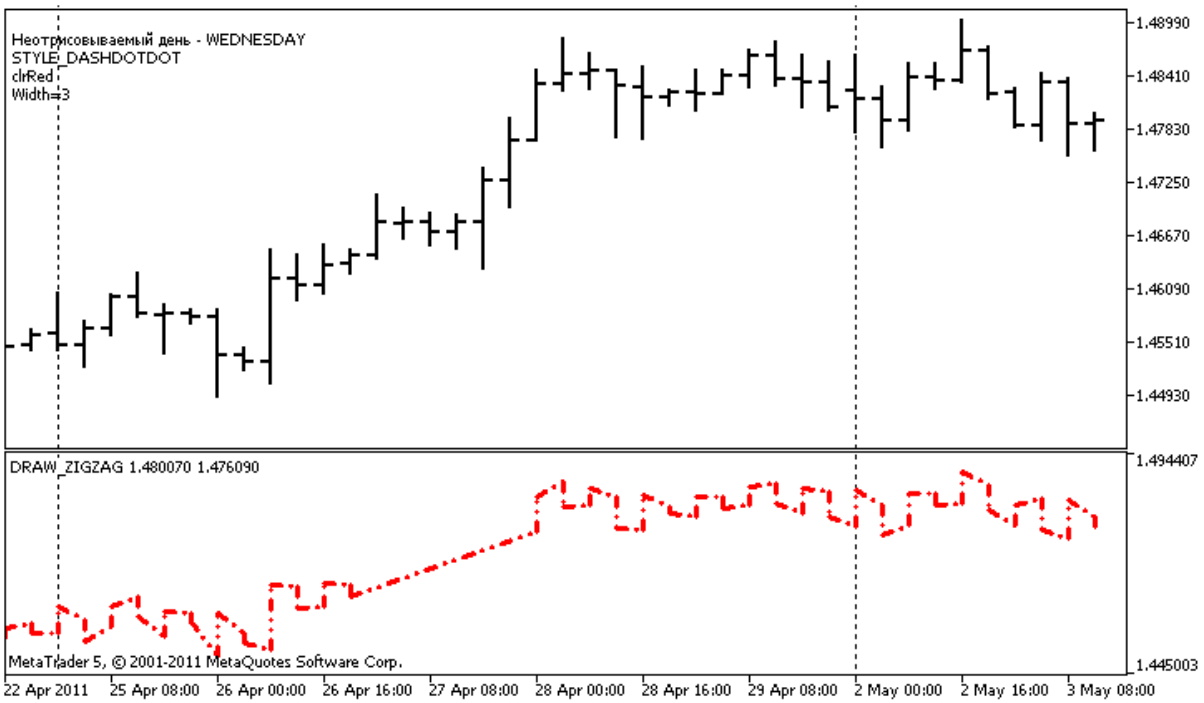
セクションは指標バッファの空でない値からあと 1 つの空でない値まで描かれます。「空」値を指定するには PLOT_EMPTY_VALUE プロパティで値を設定します。

```
//--- 0(空)値は描画されない
   PlotIndexSetDouble(index_of_plot_DRAW_ZIGZAG,PLOT_EMPTY_VALUE,0);
```

バッファの値は常に明示的に入力し、バーをスキップするにはバッファに空の値を設定します。

DRAW_ZIGZAG のプロットに必要なバッファの数は 2 です。

高値と安値に基づいてのこぎりをプロットする指標の例。ジグザグ線の色、幅とスタイルは N ティックごとにランダムに変更されます。



DRAW_ZIGZAG を持つ **plot1** では、プロパティは #property コンパイラディレクティブを使用して指定され、後に OnCalculate() 関数でランダムに設定されることにご注意ください。N パラメータは手動設定の可能性を持って(指標の「プロパティ」ウィンドウの「パラメータ」タブ)指標外部パラメータに設定されています。

```
//+------------------------------------------------------------------+
//|                                                  DRAW_ZIGZAG.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
```

© 2000–2025, MetaQuotes Ltd.

```
//|                                                 https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "An indicator to demonstrate DRAW_ZIGZAG"
#property description "It draws a \"saw\" as straight segments, skipping the bars of one day"
#property description "The day to skip is selected randomly during indicator start"
#property description "The color, width and style of segments are changed randomly"
#property description " every N ticks"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- ZigZag をプロットする
#property indicator_label1  "ZigZag"
#property indicator_type1   DRAW_ZIGZAG
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 入力パラメータ
input int       N=5;              // 変化をもたらすティックの数
//--- 指標バッファ
double          ZigZagBuffer1[];
double          ZigZagBuffer2[];
//--- 指標がプロットされない曜日
int invisible_day;
//--- 色を格納する配列
color colors[]={clrRed,clrBlue,clrGreen};
//--- 線のスタイルを格納する配列
ENUM_LINE_STYLE styles[]
={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOTDOT};
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列と指標バッファを関連付ける
   SetIndexBuffer(0,ZigZagBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ZigZagBuffer2,INDICATOR_DATA);
//--- プロットしない曜日を決めるのに0～6の乱数を取得
   invisible_day=MathRand()%6;
//--- 0(空)値は描画されない
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- 0(空)値は描画されない
   PlotIndexSetString(0,PLOT_LABEL,"ZigZag1;ZigZag2");
//---
```

© 2000-2025, MetaQuotes Ltd.

```
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- 線のスタイル、色、及び幅を変更するティックを計算する
   ticks++;
//--- 充分なティックの数が蓄積されている場合
   if(ticks>=N)
     {
      //--- 線のプロパティを変更する
      ChangeLineAppearance();
      //--- ティックカウンタをゼロにリセットする
      ticks=0;
     }

//--- 各足の曜日を取得するのに時間の構造体が必要
   MqlDateTime dt;

//--- 計算開始位置
   int start=0;
//--- 指標が以前に計算されている場合には、計算開始点を最後から2 つ目のバーに設定する
   if(prev_calculated!=0) start=prev_calculated-1;
//--- 計算ループ
   for(int i=start;i<rates_total;i++)
     {
      //--- バーのオープン時間を書く
      TimeToStruct(time[i],dt);
      //--- このバーの曜日が invisible_day に等しい場合
      if(dt.day_of_week==invisible_day)
        {
         //--- このバーでは、空の値をバッファに書く
         ZigZagBuffer1[i]=0;
         ZigZagBuffer2[i]=0;
        }
      //--- 曜日が大丈夫なら、バッファに書き込む
      else
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          //--- バー番号が偶数の場合
          if(i%2==0)
            {
             //---  1 番目のバッファの高値と2 番目のバッファの安値を書く
             ZigZagBuffer1[i]=high[i];
             ZigZagBuffer2[i]=low[i];
            }
          //--- バー番号が奇数の場合
          else
            {
             //--- 逆の順序でバーに記入
             ZigZagBuffer1[i]=low[i];
             ZigZagBuffer2[i]=high[i];
            }
         }
      }
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//|  ジグザグセグメントの外観を変更する                                |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- ジグザグのプロパティに関する情報を形成するための文字列
   string comm="";
//--- ジグザグの色を変更するブロック
   int number=MathRand(); // 乱数を取得
//--- 除数は colors[] 配列のサイズと同じ
   int size=ArraySize(colors);
//--- 新しい色を選択するためのインデックスを整数除算の余りから取得
   int color_index=number%size;
//--- 色をPLOT_LINE_COLOR プロパティとして設定
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- 線の色を書く
   comm=comm+"\r\n"+(string)colors[color_index];

//--- 線の幅を変更するブロック
   number=MathRand();
//--- 整数除算の余りの幅を取得
   int width=number%5;    // 幅は 0～4 に設定される
//--- 色を PLOT_LINE_WIDTH プロパティに設定
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- 線の幅を書く
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- 線のスタイルを変更するブロック
   number=MathRand();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 除数は、スタイルの配列の大きさに等しい
   size=ArraySize(styles);
//--- 新しいスタイルを選択するためのインデックスを整数除算の余りから取得
   int style_index=number%size;
//--- 色をPLOT_LINE_COLOR プロパティとして設定
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- 線のスタイルを書く
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- 計算では省略された日についての情報を追加する
   comm="\r\nNot plotted day - "+EnumToString((ENUM_DAY_OF_WEEK)invisible_day)+comm;
//--- コメントを使用して、チャート上の情報を表示する
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_FILLING

DRAW_FILLING スタイルは、2個の指標バッファの値の間の色の領域をプロットします。実際に、このスタイルは 2 つの線を描画しそれらの間の空間を指定された 2 つの色のいずれかで塗りつぶします。これは、チャンネルを描く指標の作成に使用されます。いずれのバッファも空の値のみを含むことは出来ません。この場合プロットがなされません。

塗りつぶしの色は 2 つ設定することが出来ます。

- 最初の色は、1 番目のバッファの値が2 番目の指標バッファの値よりも大きい領域に使用されます。
- 2 番目の色は、2 番目のバッファの値が1 番目の指標バッファの値よりも大きい領域に使用されます。

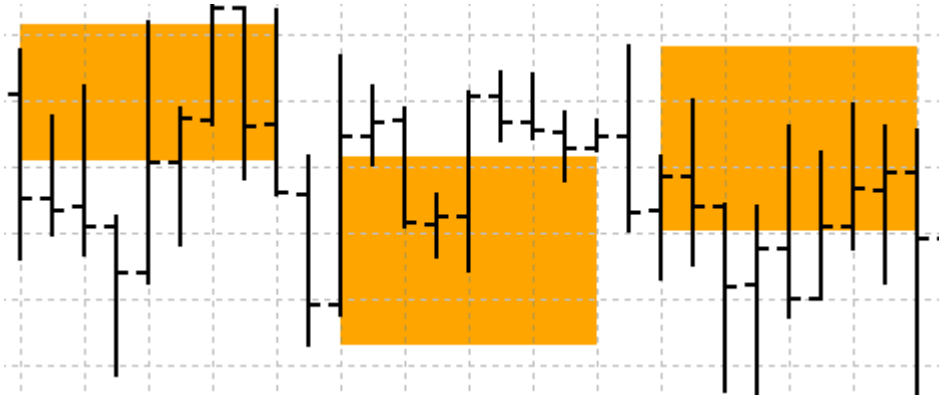
塗りつぶしの色はコンパイラディレクティブまたは動的に PlotIndexSetInteger() 関数で設定出来ます。プロットのプロパティの動的な変更は、指標の外観が現在の状況に応じて変わるようにして「盛り上げる」ことが出来ます。

指標は、両方の指標バッファの値が 0 でも空値でもない全てのバーに計算されます。「空」値を指定するには PLOT_EMPTY_VALUE プロパティで値を設定します。

```
   #define INDICATOR_EMPTY_VALUE -1.0
   ...
//--- INDICATOR_EMPTY_VALUE (空)値は以下に使用されない
   PlotIndexSetDouble
(DRAW_FILLING_creation_index,PLOT_EMPTY_VALUE,INDICATOR_EMPTY_VALUE);
```

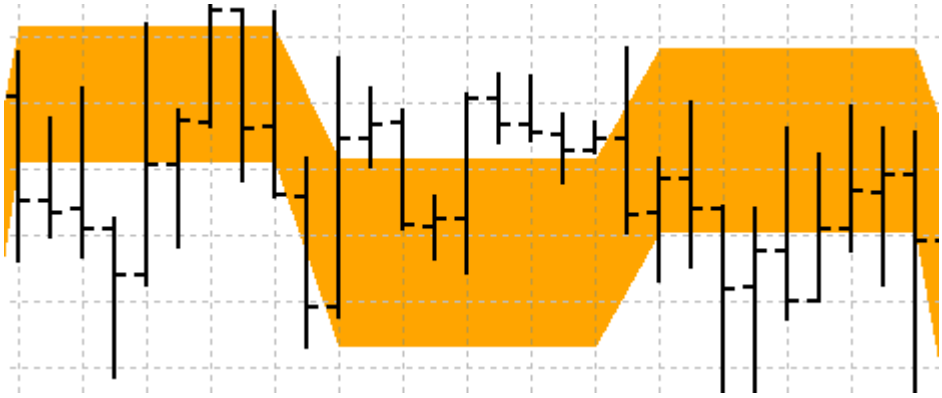
指標の計算に関与しないバーの描画は指標バッファ内の値に依存します。

- 両方の指標バッファの値が 0 に等しいバーは、指標描画に関与しません。これは、ゼロの値を持つ領域が記入されていないことを意味します。



- 指標バッファの値が「空の値」に等しいバーは、指標描画に関与します。空の値を持つ領域が有意な値と領域を接続するように記入されます。

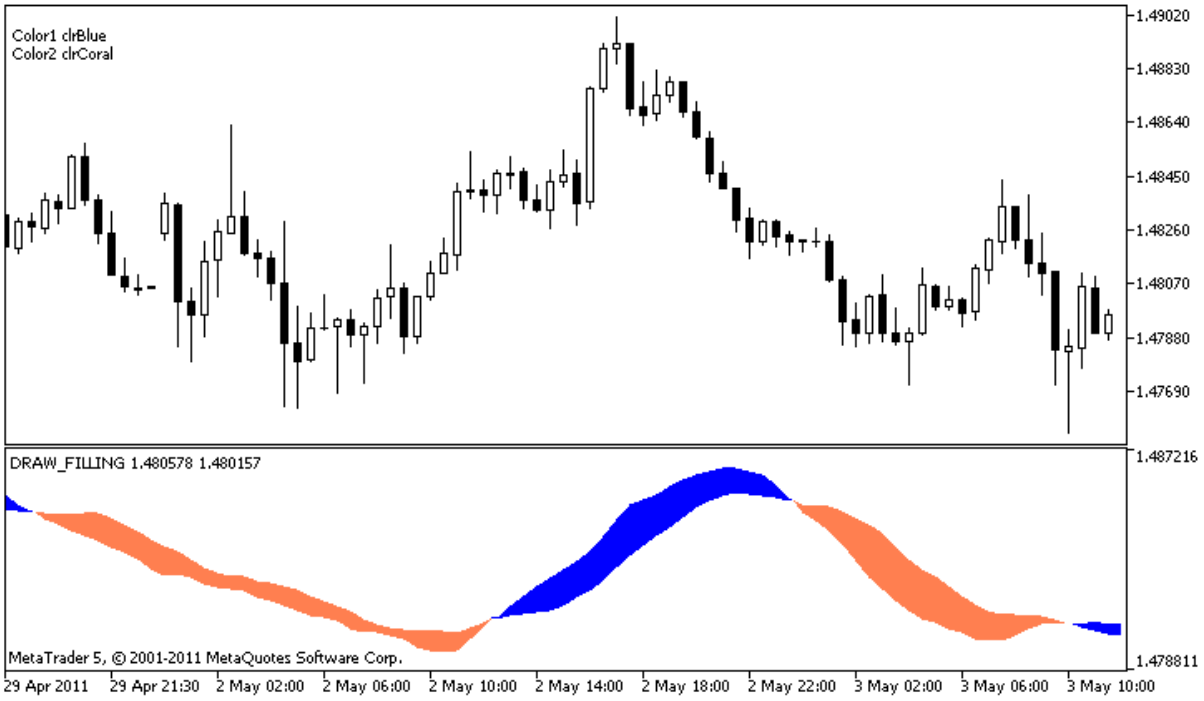
© 2000-2025, MetaQuotes Ltd.



「空の値」がゼロに等しい場合、指標の計算に関与しないバーも記入されていることに留意すべきです。

DRAW_FILLING のプロットに必要なバッファの数は 2 です。

別々のウィンドウで別々の平均期間を持つ 2 つのMAの間のチャンネルを描画する指標の例です。移動平均の交差点での色の変化は視覚的に上向きと下向きのトレンドの変化を示しています。色は N ティックごとにランダムに変わります。パラメータは手動設定の可能性を持って（指標の「プロパティ」ウィンドウの「パラメータ」タブ）指標外部パラメータに設定されています。



DRAW_FILLING を持つ **plot1** では、プロパティは #property コンパイラディレクティブを使用して指定され、後に OnCalculate() 関数で新しい色がランダムに設定されます。

```
//+------------------------------------------------------------------+
//|                                                DRAW_FILLING.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
```

© 2000–2025, MetaQuotes Ltd.

```
#property version   "1.00"

#property description "An indicator to demonstrate DRAW_FILLING"
#property description "It draws a channel between two MAs in a separate window"
#property description "The fill color is changed randomly"
#property description "after every N ticks"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- 交差点をプロットする
#property indicator_label1  "Intersection"
#property indicator_type1   DRAW_FILLING
#property indicator_color1  clrRed,clrBlue
#property indicator_width1  1
//--- 入力パラメータ
input int       Fast=13;          // 高速 MA 期間
input int       Slow=21;          // 低速 MA 期間
input int       shift=1;          // 将来(正)に向けた MA のシフト
input int       N=5;              // 変化をもたらすティックの数
//--- 指標バッファ
double          IntersectionBuffer1[];
double          IntersectionBuffer2[];
int fast_handle;
int slow_handle;
//--- 色を格納する配列
color colors[]
={clrRed
,clrBlue
,clrGreen,clrAquamarine,clrBlanchedAlmond,clrBrown,clrCoral,clrDarkSlateGray};
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 指標バッファマッピング
   SetIndexBuffer(0,IntersectionBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,IntersectionBuffer2,INDICATOR_DATA);
//---
   PlotIndexSetInteger(0,PLOT_SHIFT,shift);
//---
   fast_handle=iMA(_Symbol,_Period,Fast,0,MODE_SMA,PRICE_CLOSE);
   slow_handle=iMA(_Symbol,_Period,Slow,0,MODE_SMA,PRICE_CLOSE);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- 線のスタイル、色、及び幅を変更するティックを計算する
   ticks++;
//--- 充分なティックの数が蓄積されている場合
   if(ticks>=N)
     {
      //--- 線のプロパティを変更する
      ChangeLineAppearance();
      //--- ティックカウンタをゼロにリセットする
      ticks=0;
     }

//--- 指標の最初の計算を行う。またはデータが変更されており、完全な再計算が必要。
   if(prev_calculated==0)
     {
      //--- 指標値の全てをバッファに複製する
      int copied1=CopyBuffer(fast_handle,0,0,rates_total,IntersectionBuffer1);
      int copied2=CopyBuffer(slow_handle,0,0,rates_total,IntersectionBuffer2);
     }
   else // 更新されたデータのみを書き込む
     {
      //--- OnCalculate() の現在及び以前の開始点の間でのバーの違いを取得
      int to_copy=rates_total-prev_calculated;
      //--- 違いがなければゼロバーに値をひとつ複製する
      if(to_copy==0) to_copy=1;
      //--- to_copy 値を指標バッファの終わりに複製する
      int copied1=CopyBuffer(fast_handle,0,0,to_copy,IntersectionBuffer1);
      int copied2=CopyBuffer(slow_handle,0,0,to_copy,IntersectionBuffer2);
     }
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//|  チャンネル塗りつぶしの色を変更                                  |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- 線のプロパティに関する情報を形成するための文字列
```

© 2000-2025, MetaQuotes Ltd.

```
   string comm="";
//--- 線の色を変更するブロック
   int number=MathRand(); // 乱数を取得
//--- 除数は colors[] 配列のサイズと同じ
   int size=ArraySize(colors);

//--- 新しい色を選択するためのインデックスを整数除算の余りから取得
   int color_index1=number%size;
//--- 最初の色を PLOT_LINE_COLOR プロパティとして設定
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,colors[color_index1]);
//--- 最初の色を書く
   comm=comm+"\r\nColor1 "+(string)colors[color_index1];

//--- 新しい色を選択するためのインデックスを整数除算の余りから取得
   number=MathRand(); // 乱数を取得
   int color_index2=number%size;
//--- 2 番目の色を PLOT_LINE_COLOR プロパティとして設定
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,colors[color_index2]);
//--- 2 番目の色を書く
   comm=comm+"\r\nColor2 "+(string)colors[color_index2];
//--- コメントを使用して、チャート上の情報を表示する
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_BARS

DRAW_BARS スタイルは、始値、高値、安値、終値の値を含む 4 つの指標バッファからバーを描画します。これは、チャートの別々のサブウィンドウ内及びその他の金融商品のカスタム指標をバーとして作成するために使用されます。

バーの色はコンパイラディレクティブまたは動的に PlotIndexSetInteger() 関数で設定出来ます。プロットのプロパティの動的な変更は、指標の外観が現在の状況に応じて変わるようにして「盛り上げる」ことが出来ます。

指標は 4 つの指標バッファ 全部に空以外の値があるバーにのみ描画されます。「空」値を指定するには PLOT_EMPTY_VALUE プロパティで値を設定します。

```
//--- 0(空)値は描画されない
   PlotIndexSetDouble(index_of_plot_DRAW_BARS,PLOT_EMPTY_VALUE,0);
```

バッファの値は常に明示的に入力し、バーをスキップするにはバッファに空の値を設定します。

DRAW_BARS のプロットに必要なバッファの数は 4 です。プロットバッファは、始値、高値、安値、終値の順番であるべきです。いずれのバッファも空の値のみを含むことは出来ません。この場合プロットがなされません。

別のウィンドウで選択された金融商品のバーを描画する指標の例です。バーの色は N ティックごとにランダムに変わります。パラメータは手動設定の可能性を持って(指標の「プロパティ」ウィンドウの「パラメータ」タブ)指標外部パラメータに設定されています。



DRAW_BARS スタイルを持つ plot1 では、色が #propertyコンパイラディレクティブによって指定された後、OnCalculate() 関数で事前に準備されたリストからランダムに設定されます。

```
//+------------------------------------------------------------------+
//|                                                    DRAW_BARS.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
```

© 2000-2025, MetaQuotes Ltd.

```
//|                                                 https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "An indicator to demonstrate DRAW_BARS"
#property description "It draws bars of a selected symbol in a separate window"
#property description "The color and width of bars, as well as the symbol are changed randomly"
#property description "every N ticks"

#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   1
//--- バーをプロットする
#property indicator_label1  "Bars"
#property indicator_type1   DRAW_BARS
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 入力パラメータ
input int      N=5;              // 種類を変更するティック数
input int      bars=500;         // 表示するバーの数
input bool     messages=false;   // エキスパートアドバイザーログにメッセージを表示する
//--- 指標バッファ
double         BarsBuffer1[];
double         BarsBuffer2[];
double         BarsBuffer3[];
double         BarsBuffer4[];
//--- 銘柄名
string symbol;
//--- 色を格納する配列
color colors[]={clrRed,clrBlue,clrGreen,clrPurple,clrBrown,clrIndianRed};
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- バーが小さい場合は、前もって完成する
   if(bars<50)
     {
      Comment("Please specify a larger number of bars! The operation of the indicator has been terminated");
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- 指標バッファマッピング
   SetIndexBuffer(0,BarsBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,BarsBuffer2,INDICATOR_DATA);
```

© 2000–2025, MetaQuotes Ltd.

```
   SetIndexBuffer(2,BarsBuffer3,INDICATOR_DATA);
   SetIndexBuffer(3,BarsBuffer4,INDICATOR_DATA);
//--- バーが描画される銘柄名
   symbol=_Symbol;
//--- シンボルの表示を設定
   PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;"+symbol+" High;"+symbol+"
Low;"+symbol+" Close");
   IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_BARS("+symbol+")");
//--- 空の値
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0.0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- 線のスタイル、色、及び幅を変更するティックを計算する
   ticks++;
//--- 充分なティックの数が蓄積されている場合
   if(ticks>=N)
     {
      //---「気配値表示」ウィンドウで新しいシンボルを選ぶ
      symbol=GetRandomSymbolName();
      //--- 線のプロパティを変更する
      ChangeLineAppearance();

      int tries=0;
      //--- シンボルからの価格でバッファを埋めることを五回試みる
      while(!CopyFromSymbolToBuffers(symbol,rates_total) && tries<5)
        {
         //--- CopyFromSymbolToBuffers() 関数呼び出しのカウンタ
         tries++;
        }
      //--- ティックカウンタをゼロにリセットする
      ticks=0;
     }
//--- 次の呼び出しのために prev_calculated の値を返す
```

© 2000-2025, MetaQuotes Ltd.

```
   return(rates_total);
  }
//+------------------------------------------------------------------+
//|  指標バッファに価格を記入する                                    |
//+------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,int total)
  {
//--- rates[] 配列で、始値、高値、安値、終値を複製する
   MqlRates rates[];
//--- 試みのカウンタ
   int attempts=0;
//--- 複製された量
   int copied=0;
//--- 希望する記号の時系列を取得するための25の試みをする
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) attempts=%
d",__FUNCTION__,name,attempts);
     }
//--- 充分な数のバーの複製が失敗した場合
   if(copied!=bars)
     {
      //--- メッセージ文字列を作る
      string comm=StringFormat("For the symbol %s, managed to receive only %d bars
of %d requested ones",
                               name,
                               copied,
                               bars
                               );
      //--- メインチャートウィンドウでコメントを表示する
      Comment(comm);
      //--- メッセージを表示する
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- シンボルの表示を設定する
      PlotIndexSetString(0,PLOT_LABEL,name+" Open;"+name+" High;"+name+"
Low;"+name+" Close");
      IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_BARS("+name+")");
     }
//--- バッファを空の値で初期化する
   ArrayInitialize(BarsBuffer1,0.0);
   ArrayInitialize(BarsBuffer2,0.0);
   ArrayInitialize(BarsBuffer3,0.0);
   ArrayInitialize(BarsBuffer4,0.0);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 価格をバッファに複製する
   for(int i=0;i<copied;i++)
     {
      //--- バッファの適切なインデックスを計算する
      int buffer_index=total-copied+i;
      //--- 価格をバッファに書く
      BarsBuffer1[buffer_index]=rates[i].open;
      BarsBuffer2[buffer_index]=rates[i].high;
      BarsBuffer3[buffer_index]=rates[i].low;
      BarsBuffer4[buffer_index]=rates[i].close;
     }
   return(true);
  }
//+------------------------------------------------------------------+
//| 「気配値表示」からランダムにシンボルを返す                                  |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//---「気配値表示」ウィンドウに表示されるシンボルの数
   int symbols=SymbolsTotal(true);
//--- リスト内のシンボルの位置 - 0～シンボル数の乱数
   int number=MathRand()%symbols;
//--- 指定された位置のシンボルの名称を返す
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//| バーの外観を変更する                                                |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- バーのプロパティに関する情報を形成するための文字列
   string comm="";
//--- バーの色を変更するブロック
   int number=MathRand(); // 乱数を取得
//--- 除数は colors[] 配列のサイズと同じ
   int size=ArraySize(colors);
//--- 新しい色を選択するためのインデックスを整数除算の余りから取得
   int color_index=number%size;
//--- 色をPLOT_LINE_COLOR プロパティとして設定
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- 線の色を書く
   comm=comm+"\r\n"+(string)colors[color_index];

//--- バーの幅を変更するブロック
   number=MathRand();
//--- 整数除算の余りの幅を取得
   int width=number%5;    // 幅は 0～4 に設定される
//--- 色を PLOT_LINE_WIDTH プロパティに設定
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 線の幅を書く
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- 銘柄名を書く
   comm="\r\n"+symbol+comm;

//--- コメントを使用して、チャート上の情報を表示する
   Comment(comm);
  }
```

© 2000–2025, MetaQuotes Ltd.

DRAW_CANDLES

DRAW_CANDLES スタイルは、始値、高値、安値、終値の値を含む 4 つの指標バッファからローソク足を描画します。これは、チャートの別々のサブウィンドウ内及びその他の金融商品のカスタム指標をローソク足として作成するために使用されます。

ローソク足の色はコンパイラディレクティブ または動的に PlotIndexSetInteger() 関数で設定出来ます。プロットのプロパティの動的な変更は、指標の外観が現在の状況に応じて変わるようにして「盛り上げる」ことが出来ます。

指標は 4 つの指標バッファ 全部に空以外の値があるバーにのみ描画されます。「空」値を指定するには PLOT_EMPTY_VALUE プロパティで値を設定します。

```
//--- 0(空)値は描画されない
   PlotIndexSetDouble(index_of_plot_DRAW_CANDLES,PLOT_EMPTY_VALUE,0);
```

バッファの値は常に明示的に入力し、バーをスキップするにはバッファに空の値を設定します。

DRAW_CANDLES のプロットに必要なバッファの数は 4 です。プロット バッファは、始値、高値、安値、終値の順番であるべきです。いずれのバッファも空の値のみを含むことは出来ません。この場合プロットがなされません。

DRAW_CANDLESスタイルでは3色までを設定してローソク足の外観を変えることができます。一色のみが設定された場合、それがチャート上のすべてのローソク足に適用されます。

```
//--- 単一の色が適用された同一のローソク足
#property indicator_label1  "One color candles"
#property indicator_type1   DRAW_CANDLES
//--- 1色のみが設定されたのでローソク足の色はすべて同じ
#property indicator_color1  clrGreen
```

2つのカンマで区切られた色が指定された場合は、1番目の色がローソク足の輪郭、2番目が実体に適用されます。<

```
//--- ローソク足と芯での異なる色
#property indicator_label1  "Two color candles"
#property indicator_type1   DRAW_CANDLES
//--- 緑が芯と輪郭、白が実体に適応される
#property indicator_color1  clrGreen,clrWhite
```

上昇および下降のローソク足が異なって表示されるためにはコンマで区切られた3つの色を指定します。その場合は、1番目の色がローソク足の輪郭、2番目と3番目は強気と弱気ローソク足に使われます。

```
//--- ローソク足と芯での異なる色
#property indicator_label1  "One color candles"
#property indicator_type1   DRAW_CANDLES
//--- 緑が芯と輪郭、強気ローソク足の実体は白で弱気ローソク足の実体は赤
#property indicator_color1  clrGreen,clrWhite,clrRed
```

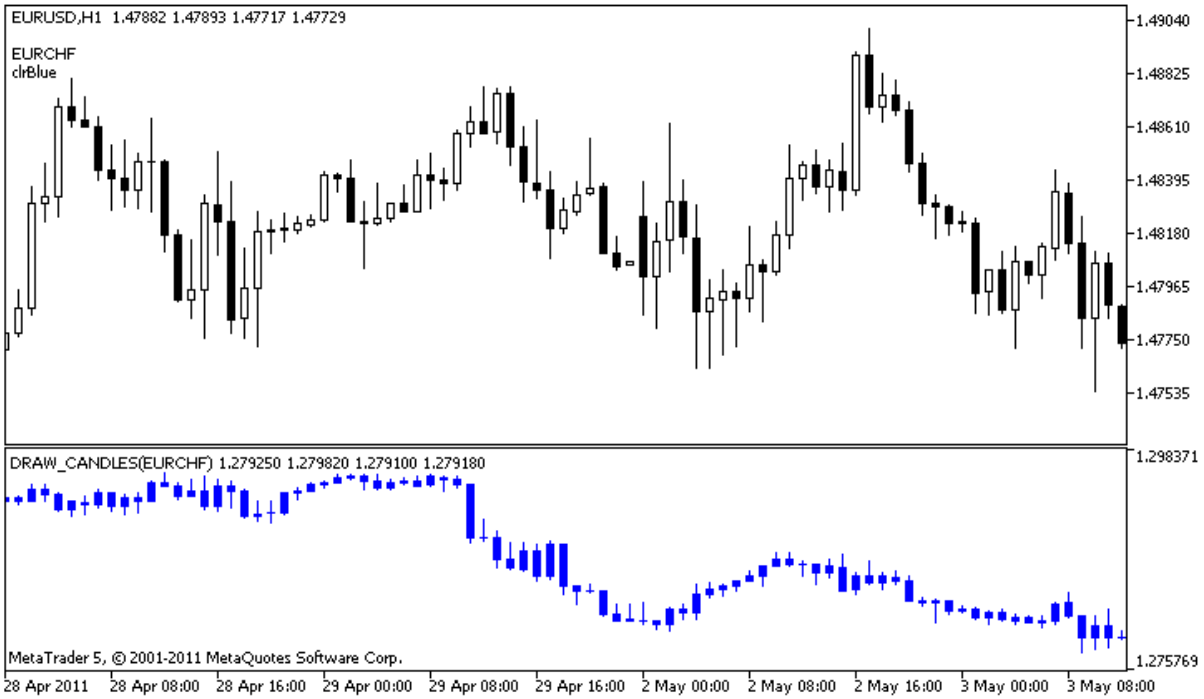
よって、DRAW_CANDLESスタイルでカスタムキャンドル着色オプションを作成することができます。また、すべての色はPlotIndexSetInteger function (composition_index_DRAW_CANDLES, PLOT_LINE_COLOR, modifier_index, color)関数を使用してインディケータの動作中に動的に変更することができ、ここでmodifier_indexの持つ値は次のいずれかです。

© 2000-2025, MetaQuotes Ltd.

- 0 – 輪郭と芯の色
- 1– 強気のローソク足の実体の色
- 2 – 弱気のローソク足の実体の色

```
//--- 輪郭と芯の色を設定する
PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrBlue);
//--- 強気のローソク足の実体の色を設定する
PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrGreen);
//--- 弱気のローソク足の実体の色を設定する
PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrRed);
```

別のウィンドウで選択された金融商品のローソク足を描画する指標の例です。ローソク足の色はN ティックごとにランダムに変わります。パラメータは手動設定の可能性を持って（指標の「プロパティ」ウィンドウの「パラメータ」タブ）指標外部パラメータに設定されています。



plot1では、色が #property コンパイラディレクティブによって指定された後、OnCalculate() 関数で事前に準備されたリストからランダムに設定されます。

```
//+------------------------------------------------------------------+
//|                                                 DRAW_CANDLES.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "An indicator to demonstrate DRAW_CANDLES."
#property description "It draws candlesticks of a selected symbol in a separate window"
#property description " "
```

© 2000–2025, MetaQuotes Ltd.

```
#property description "The color and width of candlesticks, as well as the symbol are changed"
#property description "randomly every N ticks"

#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   1
//--- バーをプロットする
#property indicator_label1  "DRAW_CANDLES1"
#property indicator_type1   DRAW_CANDLES
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1

//--- 入力パラメータ
input int      N=5;              // 種類を変更するティック数
input int      bars=500;         // 表示するバーの数
input bool     messages=false;   // エキスパートアドバイザー操作ログにメッセージを表示する
//--- 指標バッファ
double         Candle1Buffer1[];
double         Candle1Buffer2[];
double         Candle1Buffer3[];
double         Candle1Buffer4[];
//--- 銘柄名
string symbol;
//--- 色を格納する配列
color colors[]={clrRed,clrBlue,clrGreen,clrPurple,clrBrown,clrIndianRed};
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- バーが小さい場合は、前もって完成する
   if(bars<50)
     {
      Comment("Please specify a larger number of bars!The operation of the indicator has been terminated");
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- 指標バッファマッピング
   SetIndexBuffer(0,Candle1Buffer1,INDICATOR_DATA);
   SetIndexBuffer(1,Candle1Buffer2,INDICATOR_DATA);
   SetIndexBuffer(2,Candle1Buffer3,INDICATOR_DATA);
   SetIndexBuffer(3,Candle1Buffer4,INDICATOR_DATA);
//--- 空の値
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- バーが描画される銘柄名
   symbol=_Symbol;
//--- シンボルの表示を設定
```

© 2000-2025, MetaQuotes Ltd.

```
   PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;"+symbol+" High;"+symbol+"
Low;"+symbol+" Close");
   IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_CANDLES("+symbol+")");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=INT_MAX-100;
//--- 線のスタイル、色、及び幅を変更するティックを計算する
   ticks++;
//--- 充分なティックの数が蓄積されている場合
   if(ticks>=N)
     {
      //---「気配値表示」 ウィンドウで新しいシンボルを選ぶ
      symbol=GetRandomSymbolName();
      //--- 形を変更する
      ChangeLineAppearance();
      //---「気配値表示」 ウィンドウで新しいシンボルを選ぶ
      int tries=0;
      //--- シンボルからの価格で plot1 バッファを埋めることを五回試みる
      while(!CopyFromSymbolToBuffers(symbol,rates_total,0,
            Candle1Buffer1,Candle1Buffer2,Candle1Buffer3,Candle1Buffer4)
            && tries<5)
        {
         //--- CopyFromSymbolToBuffers() 関数呼び出しのカウンタ
         tries++;
        }
      //--- ティックカウンタをゼロにリセットする
      ticks=0;
     }
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| 指定されたローソク足に書き込む                                   |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool CopyFromSymbolToBuffers(string name,
                             int total,
                             int plot_index,
                             double &buff1[],
                             double &buff2[],
                             double &buff3[],
                             double &buff4[]
                            )
  {
//--- rates[] 配列で、始値、高値、安値、終値を複製する
   MqlRates rates[];
//--- 試みのカウンタ
   int attempts=0;
//--- 複製された量
   int copied=0;
//--- 希望する記号の時系列を取得するための25の試みをする
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) attempts=%
d",__FUNCTION__,name,attempts);
     }
//--- 充分な数のバーの複製が失敗した場合
   if(copied!=bars)
     {
      //--- メッセージ文字列を作る
      string comm=StringFormat("For the symbol %s, managed to receive only %d bars
of %d requested ones",
                               name,
                               copied,
                               bars
                               );
      //--- メインチャートウィンドウでコメントを表示する
      Comment(comm);
      //--- メッセージを表示する
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- シンボルの表示を設定する
      PlotIndexSetString(plot_index,PLOT_LABEL,name+" Open;"+name+" High;"+name+"
Low;"+name+" Close");
     }
//--- バッファを空の値で初期化する
   ArrayInitialize(buff1,0.0);
   ArrayInitialize(buff2,0.0);
   ArrayInitialize(buff3,0.0);
```

© 2000-2025, MetaQuotes Ltd.

```
   ArrayInitialize(buff4,0.0);
//--- ティックのたびに価格をバッファに複製する
   for(int i=0;i<copied;i++)
     {
      //--- バッファの適切なインデックスを計算する
      int buffer_index=total-copied+i;
      //--- 価格をバッファに書く
      buff1[buffer_index]=rates[i].open;
      buff2[buffer_index]=rates[i].high;
      buff3[buffer_index]=rates[i].low;
      buff4[buffer_index]=rates[i].close;
     }
   return(true);
  }
//+------------------------------------------------------------------+
//| 「気配値表示」からランダムにシンボルを返す                                |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- 「気配値表示」ウィンドウに表示されるシンボルの数
   int symbols=SymbolsTotal(true);
//--- リスト内のシンボルの位置 - 0～シンボル数の乱数
   int number=MathRand()%symbols;
//--- 指定された位置のシンボルの名称を返す
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//| バーの外観を変更する                                                |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- バーのプロパティに関する情報を形成するための文字列
   string comm="";
//--- バーの色を変更するブロック
   int number=MathRand(); // 乱数を取得
//--- 除数は colors[] 配列のサイズと同じ
   int size=ArraySize(colors);
//--- 新しい色を選択するためのインデックスを整数除算の余りから取得
   int color_index=number%size;
//--- 色をPLOT_LINE_COLOR プロパティとして設定
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- 色を書く
   comm=comm+"\r\n"+(string)colors[color_index];
//--- 銘柄名を書く
   comm="\r\n"+symbol+comm;
//--- コメントを使用して、チャート上の情報を表示する
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

© 2000–2025, MetaQuotes Ltd.

DRAW_COLOR_LINE

DRAW_COLOR_LINE は DRAW_LINE スタイルの色付きバージョンで、指標バッファの値を使用して線を描きます。しかし、このスタイルは、タイトルにCOLOR がついている全てのカラースタイルと同じく、特別に設けられた色の配列のカラーインデックス(番号)を格納する追加の特別な指標バッファを持っています。従って、それぞれの線分の色は、そのバーで線を引くための、カラーインデックスのインデックスを指定することによって定義することが出来ます。

線の幅、スタイルと色はコンパイラディレクティブまたは 動的に PlotIndexSetInteger() 関数を使用して指定することが出来ます。プロットのプロパティの動的な変更は、指標の外観が現在の状況に応じて変わるようにして「盛り上げる」ことが出来ます。

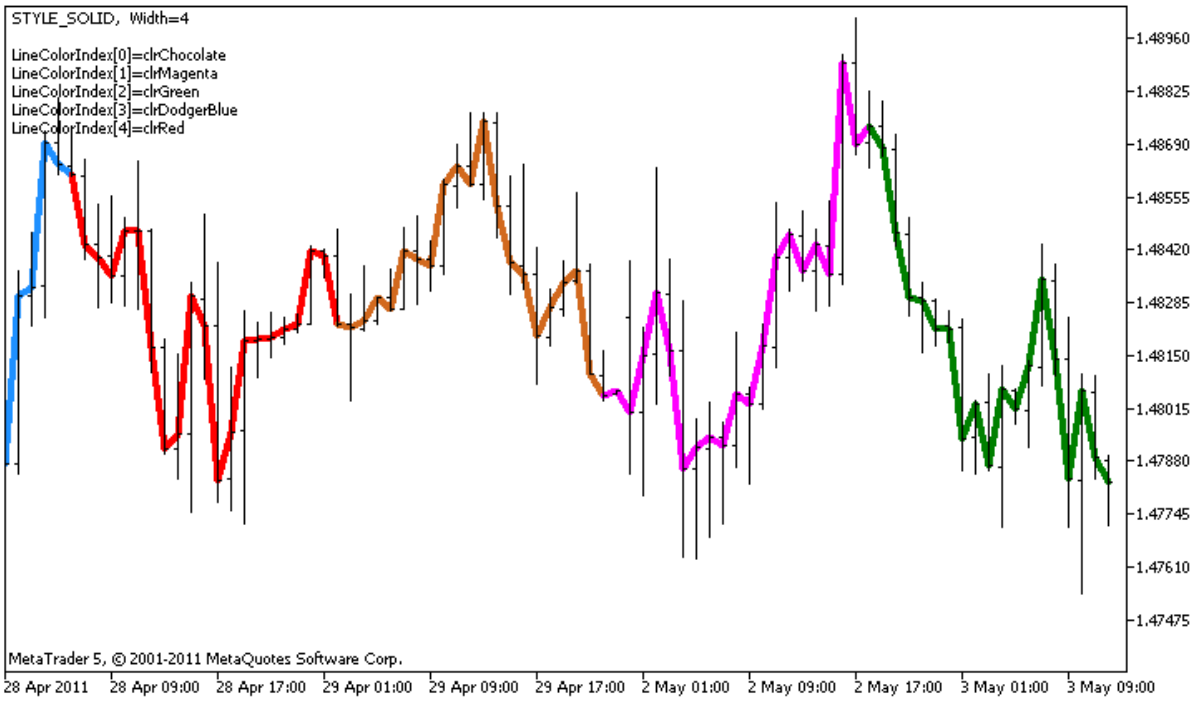
DRAW_COLOR_LINE のプロットに必要なバッファの数は 2 です。

- 1 つ目のバッファは線を描くために使用される指標値を格納します。
- 2 つ目のバッファは各足の線の色のインデックスを格納します。

色は #property indicator_color1 コンパイラディレクティブでコンマで区切って指定することが出来ます。色の数は 64 を超えることは出来ません。

```
//--- バーを塗るために(特殊な配列に格納されている)5色を定義する
#property indicator_color1  clrRed,clrBlue,clrGreen,clrOrange,clrDeepPink // (64色まで指定可能)
```

バーの終値を使用して線を描画する指標の例です。線幅とスタイルは N=5 ティックごとにランダムに変更されます。



線分の色も ChangeColors() カスタム関数でランダムに変わります。

```
//+------------------------------------------------------------------+
//|  線分の色を変更する                                              |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- 色の数
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- それぞれのカラーインデックに新しい色をランダムに定義する
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- 乱数を取得
      int number=MathRand();
      //--- 整数除算の余りから col[] 配列のインデックスを取得
      int i=number%size;
      //--- 各インデックスの色をPLOT_LINE_COLOR として設定する
      PlotIndexSetInteger(0,                    //  グラフィックスタイルの番号
                          PLOT_LINE_COLOR,      //  プロパティの識別子
                          plot_color_ind,       //  色を書き込む色インデックス
                          cols[i]);             //  新しい色
      //--- 色を書く
      comm=comm+StringFormat("LineColorIndex[%d]=%s
\r\n",plot_color_ind,ColorToString(cols[i],true));
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
```

例では、指標の「色」バージョンの特徴を示します。線分の色を変更するには、(カラーインデックスを含む) ColorLineColors[] バッファ内の値を変更する必要はありません。特殊配列に新しい色を設定するだけです。よって、PlotIndexSetInteger() 関数を用いて、小規模の色配列を変更するだけで、プロット全体の色を一時に迅速に変更することが出来ます。

DRAW_COLOR_LINE を持つ**plot1** では、3 つのプロパティは #property コンパイラディレクティブを使用して指定され、後に OnCalculate() 関数でランダムに設定されることにご注意ください。

N パラメータと長さ(バーの色セグメントの長さ)は手動設定の可能性を持って(指標の「プロパティ」ウィンドウの「パラメータ」タブ)指標 外部パラメータ に設定されています。

```
//+------------------------------------------------------------------+
//|                                                DRAW_COLOR_LINE.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "An indicator to demonstrate DRAW_COLOR_LINE"
#property description "It draws a line on Close price in colored pieces of 20 bars
each"
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "The width, style and color of the line parts are changed randomly"
#property description "every N ticks"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- ColorLine をプロットする
#property indicator_label1  "ColorLine"
#property indicator_type1   DRAW_COLOR_LINE
//--- バーを塗るために(特殊な配列に格納されている)5色を定義する
#property indicator_color1  clrRed,clrBlue,clrGreen,clrOrange,clrDeepPink // (64色まで指定可能)
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 入力パラメータ
input int       N=5;            // 変化をもたらすティックの数
input int       Length=20;      // バーの色部分の長さ
int             line_colors=5; // 色の数は 5   ― #property indicator_color1 を参照
//--- プロットに使用されるバッファ
double          ColorLineBuffer[];
//--- バーの色を格納するバッファ
double          ColorLineColors[];

//--- 色を格納する7 要素の配列
color colors[]
={clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod};
//--- 線のスタイルを格納する配列
ENUM_LINE_STYLE styles[]
={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOTDOT};
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列と指標バッファを関連付ける
   SetIndexBuffer(0,ColorLineBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorLineColors,INDICATOR_COLOR_INDEX);
//--- 擬似乱数生成器を初期化
   MathSrand(GetTickCount());
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- 線のスタイル、色、及び幅を変更するティックを計算する
   ticks++;
//--- 充分なティックの数が蓄積されている場合
   if(ticks>=N)
     {
      //--- 線のプロパティを変更する
      ChangeLineAppearance();
      //--- 線分の色を変更する
      ChangeColors(colors,5);
      //--- ティックカウンタをゼロにリセットする
      ticks=0;
     }

//--- 指標安倍を計算するブロック
   for(int i=0;i<rates_total;i++)
     {
      //--- 指標値をバッファに書く
      ColorLineBuffer[i]=close[i];
      //--- このバーのカラーインデックスをランダムに設定する
      int color_index=i%(5*Length);
      color_index=color_index/Length;
      //--- このバーでは color_index がインデックスの色を使用する
      ColorLineColors[i]=color_index;
     }

//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//|  線分の色を変更する                                              |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- 色の数
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- それぞれのカラーインデックに新しい色をランダムに定義する
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- 乱数を取得
       int number=MathRand();
       //--- 整数除算の余りから col[] 配列のインデックスを取得
       int i=number%size;
       //--- 各インデックスの色をPLOT_LINE_COLOR として設定する
       PlotIndexSetInteger(0,                          //  グラフィックスタイルの番号
                           PLOT_LINE_COLOR,            //  プロパティの識別子
                           plot_color_ind,             //  色を書き込む色インデックス
                           cols[i]);                   //  新しい色
       //--- 色を書く
       comm=comm+StringFormat("LineColorIndex[%d]=%s
\r\n",plot_color_ind,ColorToString(cols[i],true));
       ChartSetString(0,CHART_COMMENT,comm);
      }
//---
  }
//+------------------------------------------------------------------+
//|  指標に表示された線の外観を変更する                                |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- 線のプロパティに関する情報を形成するための文字列
   string comm="";
//--- 線の幅を変更するブロック
   int number=MathRand();
//--- 整数除算の余りの幅を取得
   int width=number%5; // 幅は 0～4 に設定される
//--- 色を PLOT_LINE_WIDTH プロパティに設定
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- 線の幅を書く
   comm=comm+" Width="+IntegerToString(width);

//--- 線のスタイルを変更するブロック
   number=MathRand();
//--- 除数は、スタイルの配列の大きさに等しい
   int size=ArraySize(styles);
//--- 新しいスタイルを選択するためのインデックスを整数除算の余りから取得
   int style_index=number%size;
//--- 色をPLOT_LINE_COLOR プロパティとして設定
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- 線のスタイルを書く
   comm=EnumToString(styles[style_index])+", "+comm;
//--- コメントを使用して、チャート上の情報を表示する
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_SECTION

DRAW_COLOR_SECTION スタイルは DRAW_SECTION スタイルの色付きバージョンですが、セクションをそれぞれ異なる色で描画することが可能です。DRAW_COLOR_SECTION スタイルは、タイトルに COLOR がついている全てのカラースタイルと同じく、特別に設けられた色の配列のカラーインデックス(番号)を格納する追加の特別な指標バッファを持っています。従って、それぞれのセクションの色は、セクションの終わりのバーのカラーインデックスのインデックスを指定することによって定義することが出来ます。

セクションの幅、色とスタイルは DRAW_SECTION スタイルのようにコンパイラディレクティブまたは 動的に PlotIndexSetInteger() 関数を使用して指定することが出来ます。プロットのプロパティの動的な変更は、指標の外観が現在の状況に応じて変わるようにして「盛り上げる」ことが出来ます。

セクションは指標バッファの空でない値からあと 1 つの空でない値まで描かれ、空の値は無視されます。「空」値を指定するには PLOT_EMPTY_VALUE プロパティで値を設定します。例えば、指標をゼロ以外の値のセクションのシーケンスとして描画する必要がある場合、ゼロを空値として設定する必要があります。

```
//--- 0(空)値は描画されない
   PlotIndexSetDouble(index_of_plot_DRAW_COLOR_SECTION,PLOT_EMPTY_VALUE,0);
```

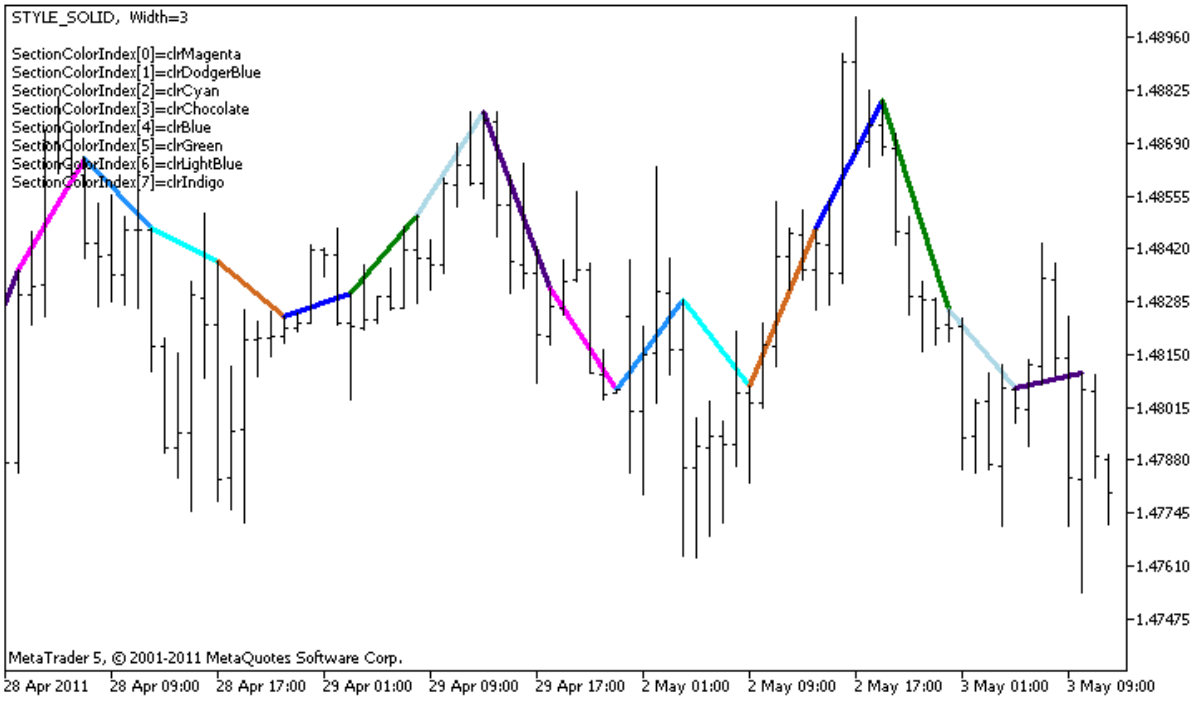
バッファの値は常に明示的に入力し、描画されないべき要素ではバッファに空の値を設定します。

DRAW_COLOR_SECTION のプロットに必要なバッファの数は 2 です。

- 1 つ目のバッファは線を描くために使用される指標値を格納します。
- もう 1 つは、矢印の描画に使用されるカラーインデックスを格納します(空値以外が必要)。

色は #property indicator_color1 コンパイラディレクティブでコンマで区切って指定することが出来ます。色の数は 64 を超えることは出来ません。

高値を用いて、5バーに相当する色つきのセクションを描画する指標の例です。色、幅とスタイルは Nティックごとにランダムに変更されます。



© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_SECTION を持つplot1 では、8 色は #property コンパイラディレクティブを使用して指定されることにご注意下さい。後に OnCalculate() 関数で、colors[] 配列に格納された色からランダムに設定されます。

N パラメータは手動設定の可能性を持って(指標の「プロパティ」ウィンドウの「パラメータ」タブ)指標 外部パラメータに設定されています。

```
//+------------------------------------------------------------------+
//|                                            DRAW_COLOR_SECTION.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "An indicator to demonstrate DRAW_COLOR_SECTION"
#property description "It draws colored sections with the length equal to the
specified number of bars"
#property description "The color, width and style of sections are changed randomly"
#property description "after every N ticks"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- ColorSection をプロットする
#property indicator_label1  "ColorSection"
#property indicator_type1   DRAW_COLOR_SECTION
//--- セクションを塗るために(特殊な配列に格納されている)8 色を定義する
#property indicator_color1
clrRed,clrGold,clrMediumBlue,clrLime,clrMagenta,clrBrown,clrTan,clrMediumVioletRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 入力パラメータ
input int       N=5;                      // 変化をもたらすティックの数
input int       bars_in_section=5;        // バーのセクションの長さ
//--- セクションの端を計算する補助変数
int             divider;
int             color_sections;
//--- プロットに使用されるバッファ
double          ColorSectionBuffer[];
//--- バーの色を格納するバッファ
double          ColorSectionColors[];
//--- 色を格納する14 要素の配列
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPu
rple
  };
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 線のスタイルを格納する配列
ENUM_LINE_STYLE styles[]
={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOTDOT};
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 指標バッファマッピング
   SetIndexBuffer(0,ColorSectionBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorSectionColors,INDICATOR_COLOR_INDEX);
//--- 0(空)値は描画されない
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---- セクションを塗る色の数
   int color_sections=8;   //  #property indicator_color1コメントを参照
//--- 指標パラメータをチェックする
   if(bars_in_section<=0)
     {
      PrintFormat("Invalid section length=%d",bars_in_section);
      return(INIT_PARAMETERS_INCORRECT);
     }
   else divider=color_sections*bars_in_section;
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- 線のスタイル、色、及び幅を変更するティックを計算する
   ticks++;
//--- 充分なティックの数が蓄積されている場合
   if(ticks>=N)
     {
      //--- 線のプロパティを変更する
      ChangeLineAppearance();
      //--- セクションをプロットする色を変更する
      ChangeColors(colors,color_sections);
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- ティックカウンタをゼロにリセットする
      ticks=0;
     }

//--- 指標値の算出が開始されるバーの番号
   int start=0;
//--- 指標が以前に計算されている場合には、開始点を前のバーに設定する
   if(prev_calculated>0) start=prev_calculated-1;
//--- 指標値の計算
   for(int i=start;i<rates_total;i++)
     {
      //--- section_lengthで割り切れるバー番号は セクションの終わりを意味する
      if(i%bars_in_section==0)
        {
         //--- この足の高値でセクションの終わりを設定する
         ColorSectionBuffer[i]=high[i];
         //--- バーの数を sceti on_l engthnumber_of_col ors で割った際の余り
         int rest=i%divider;
         // 0～number_of_colors-1 である色の番号を取得する
         int color_indext=rest/bars_in_section;
         ColorSectionColors[i]=color_indext;
        }
      //---除算の剰余は、バーに等しい場合
      else
        {
         //--- 何もない場合、バーを無視して 0 に設定
         else ColorSectionBuffer[i]=0;
        }
     }
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//|  線分の色を変更する                                              |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- 色の数
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- それぞれのカラーインデックスに新しい色をランダムに定義する
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- 乱数を取得
      int number=MathRand();
      //--- 整数除算の余りから col[] 配列のインデックスを取得
      int i=number%size;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 各インデックスの色をPLOT_LINE_COLOR として設定する
      PlotIndexSetInteger(0,                        //   グラフィックスタイルの番号
                          PLOT_LINE_COLOR,          //   プロパティの識別子
                          plot_color_ind,           //   色を書き込む色インデックス
                          cols[i]);                 //   新しい色
      //--- 色を書く
      comm=comm+StringFormat("SectionColorIndex[%d]=%s
\r\n",plot_color_ind,ColorToString(cols[i],true));
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//|  指標に表示された線の外観を変更する                                |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- 線のプロパティに関する情報を形成するための文字列
   string comm="";
//--- 線の幅を変更するブロック
   int number=MathRand();
//--- 整数除算の余りの幅を取得
   int width=number%5; // 幅は 0～4 に設定される
//--- 色を PLOT_LINE_WIDTH プロパティに設定
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- 線の幅を書く
   comm=comm+" Width="+IntegerToString(width);

//--- 線のスタイルを変更するブロック
   number=MathRand();
//--- 除数は、スタイルの配列の大きさに等しい
   int size=ArraySize(styles);
//--- 新しいスタイルを選択するためのインデックスを整数除算の余りから取得
   int style_index=number%size;
//--- 色をPLOT_LINE_COLOR プロパティとして設定
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- 線のスタイルを書く
   comm=EnumToString(styles[style_index])+", "+comm;
//--- コメントを使用して、チャート上の情報を表示する
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_HISTOGRAM

DRAW_COLOR_HISTOGRAM スタイルはゼロから指定された値まで色で塗られた列としてのヒストグラムを描画します。値は指標バッファから取られます。各列は、色の所定のセットから独自の色を持つことが出来ます。

ヒストグラムの幅、色とスタイルは DRAW_HISTOGRAM スタイルのように コンパイラディレクティブまたは動的に PlotIndexSetInteger() 関数を使用して指定することが出来ます。プロットプロパティの動的な変更は、現在の状況に基づいてのヒストグラムの外観の変更を可能にします。

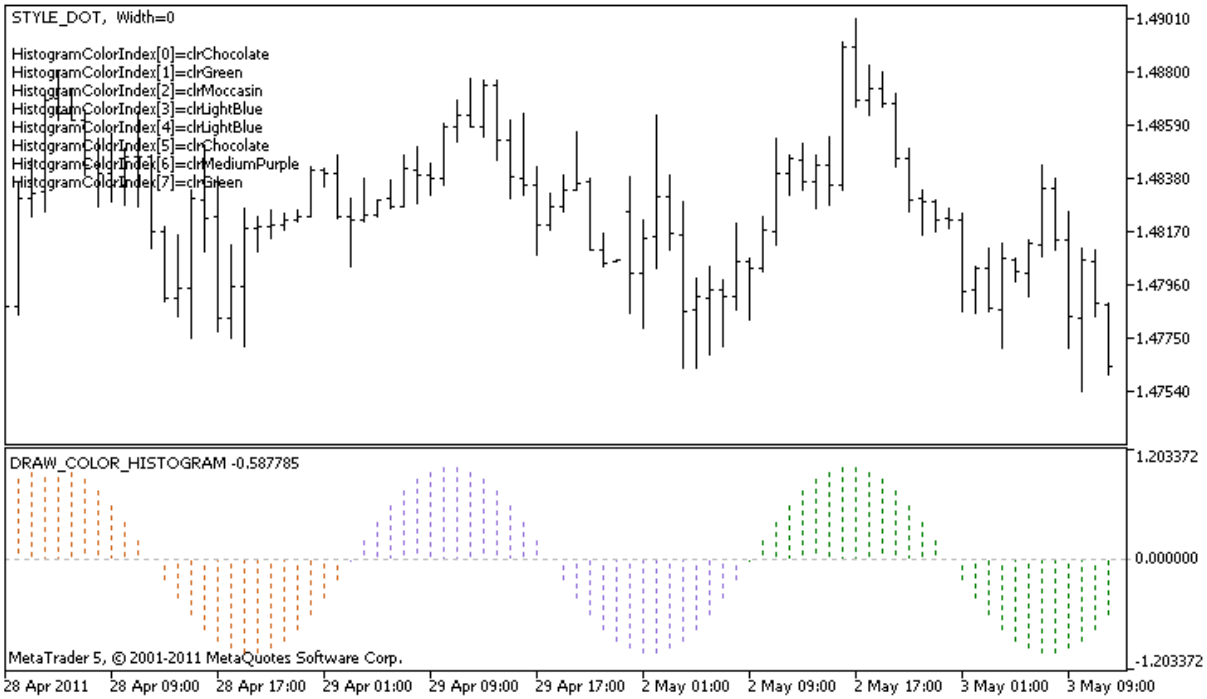
ゼロレベルからの列が各足の上に描画されているので、DRAW_COLOR_HISTOGRAM は、別のチャートウィンドウで使用される必要があります。ほとんどの場合、この種類のプロットは、オーサムオシレーター または マーケットファシリテーションインデックス のようなオシレーター型の指標を作成するために使用されます。空の表示不可能な値には、ゼロ値を指定する必要があります。

DRAW_COLOR_HISTOGRAM のプロットに必要なバッファの数は 2 です。

- 1 つ目のバッファは、各足の垂直セグメントのゼロ以外の値を格納し、セグメントの2 番目の端部は常に指標のゼロラインの上にあります。
- もう 1 つは、矢印の描画に使用されるカラーインデックスを格納します（空値以外が必要）。

色は #property indicator_color1 コンパイラディレクティブでコンマで区切って指定することが出来ます。色の数は 64 を超えることは出来ません。

MathSin() 関数に基づいて指定された色の正弦波を描く指標の例です。ヒストグラムの列の色、幅とスタイルは全て N ティックごとにランダムに変更されます。bars パラメータは正弦波の期間を指定し、バーの指定された数の後で、正弦波のサイクルが繰り返すことになります。



DRAW_COLOR_HISTOGRAM スタイルを持つplot1 では、5つの色が #propertyコンパイラディレクティブによって指定された後、OnCalculate() 関数で、colors[] 配列に格納された 14の色からランダムに設定されることにご注意ください。N パラメータは手動設定の可能性を持って（指標の「プロパティ」ウィンドウの「パラメータ」タブ）指標外部パラメータに設定されています。

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                          DRAW_COLOR_HISTOGRAM.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "An indicator to demonstrate DRAW_COLOR_HISTOGRAM"
#property description "It draws a sinusoid as a histogram in a separate window"
#property description "The color and width of columns are changed randomly"
#property description "after every N ticks"
#property description "The bars parameter sets the number of bars to repeat the
sinusoid"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- 入力パラメータ
input int      bars=30;          // バーでの正弦波の期間
input int      N=5;              // ヒストグラムを変更するティック数
//--- Color_Histogram をプロットする
#property indicator_label1  "Color_Histogram"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
//--- セクションを塗るために(特殊な配列に格納されている)8 色を定義する
#property indicator_color1
clrRed,clrGreen,clrBlue,clrYellow,clrMagenta,clrCyan,clrMediumSeaGreen,clrGold
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 値バッファ
double         Color_HistogramBuffer[];
//--- カラーインデックスを格納するバッファ
double         Color_HistogramColors[];
//--- bars パラメータを掛けた時ラジアンで2PI 角を取得する要因
double         multiplier;
int            color_sections;
//--- 色を格納する14 要素の配列
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPu
rple
  };
//--- 線のスタイルを格納する配列
ENUM_LINE_STYLE styles[]
={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOTDOT};
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 指標バッファマッピング
   SetIndexBuffer(0,Color_HistogramBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,Color_HistogramColors,INDICATOR_COLOR_INDEX);
//----正弦波を塗る色の数
   color_sections=8;   //  #property indicator_color1 コメントを参照
//--- 乗数を計算する
   if(bars>1)multiplier=2.*M_PI/bars;
   else
     {
      PrintFormat("Set the value of bars=%d greater than 1",bars);
      //--- 指標の早期終了
      return(INIT_PARAMETERS_INCORRECT);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- 線のスタイル、色、及び幅を変更するティックを計算する
   ticks++;
//--- 充分なティックの数が蓄積されている場合
   if(ticks>=N)
     {
      //--- 線のプロパティを変更する
      ChangeLineAppearance();
      //--- ヒストグラムに使用される色を変更する
      ChangeColors(colors,color_sections);
      //--- ティックカウンタをゼロにリセットする
      ticks=0;
     }

//--- 指標値を計算する
   int start=0;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- すでにOnCalculate の前回の開始時に算出した場合
   if(prev_calculated>0) start=prev_calculated-1; // 最後から2 つ目のバーから計算する
//--- 指標バッファに値を記入する
   for(int i=start;i<rates_total;i++)
     {
      //--- 値
      Color_HistogramBuffer[i]=sin(i*multiplier);
      //--- 色
      int color_index=i%(bars*color_sections);
      color_index/=bars;
      Color_HistogramColors[i]=color_index;
     }
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| 線分の色を変更する                                               |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- 色の数
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- それぞれのカラーインデックに新しい色をランダムに定義する
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- 乱数を取得
      int number=MathRand();
      //--- 整数除算の余りから col[] 配列のインデックスを取得
      int i=number%size;
      //--- 各インデックスの色をPLOT_LINE_COLOR として設定する
      PlotIndexSetInteger(0,                    //   グラフィックスタイルの番号
                          PLOT_LINE_COLOR,      //   プロパティの識別子
                          plot_color_ind,       //   色を書き込む色インデックス
                          cols[i]);             //   新しい色
      //--- 色を書く
      comm=comm+StringFormat("HistogramColorIndex[%d]=%s
\r\n",plot_color_ind,ColorToString(cols[i],true));
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| 指標に表示された線の外観を変更する                               |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 線のプロパティに関する情報を形成するための文字列
   string comm="";
//--- 線の幅を変更するブロック
   int number=MathRand();
//--- 整数除算の余りの幅を取得
   int width=number%5; // 幅は 0～4 に設定される
//--- 色を PLOT_LINE_WIDTH プロパティに設定
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- 線の幅を書く
   comm=comm+" Width="+IntegerToString(width);

//--- 線のスタイルを変更するブロック
   number=MathRand();
//--- 除数は、スタイルの配列の大きさに等しい
   int size=ArraySize(styles);
//--- 新しいスタイルを選択するためのインデックスを整数除算の余りから取得
   int style_index=number%size;
//--- 色をPLOT_LINE_COLOR プロパティとして設定
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- 線のスタイルを書く
   comm=EnumToString(styles[style_index])+", "+comm;
//--- コメントを使用して、チャート上の情報を表示する
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_HISTOGRAM2

DRAW_COLOR_HISTOGRAM2 スタイルは、指定された色のヒストグラム（2 つの指標バッファの値を使用した垂直セグメント）を描画します。しかし、1 色を使用する DRAW_HISTOGRAM2 とは異なり、このスタイルでは、それぞれのヒストグラムの列は事前定義されたセットから独自の色を持つことが出来ます。セグメントの値は指標バッファから取られます。

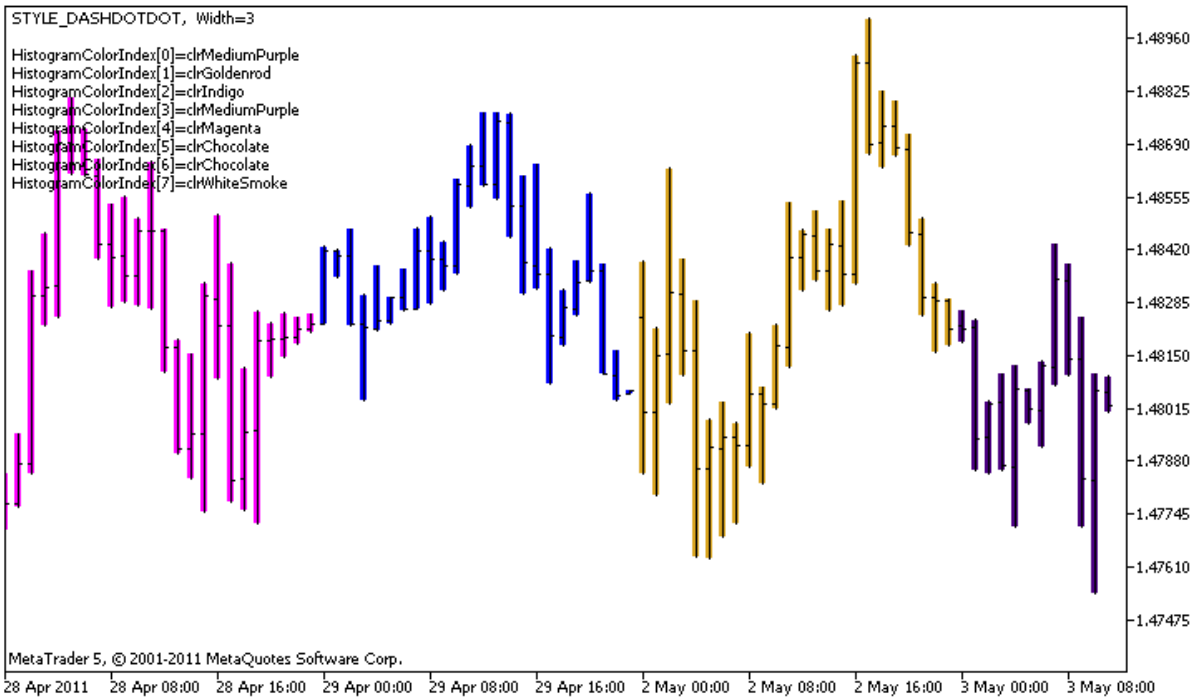
ヒストグラムの幅、スタイルと色は DRAW_HISTOGRAM2 スタイルのようにコンパイラディレクティブまたは 動的に PlotIndexSetInteger() 関数を使用して指定することが出来ます。プロットプロパティの動的な変更は、現在の状況に基づいてのヒストグラムの外観の変更を可能にします。

DRAW_COLOR_HISTOGRAM2 スタイルはチャートの別々のサブウィンドウとメインウィンドウで使用することが出来ます。空の値は描画されません。指標バッファ内の全ての値は明示的に設定する必要があります。バッファは空の値で初期化されません。

DRAW_COLOR_HISTOGRAM2 のプロットに必要なバッファの数は 3 です。

- その内の 2 つは、各足の垂直セグメントの上下の端を格納します。
- 3 つ目のバッファは、セグメントの描画に使用されるカラーインデックスを格納します（空値以外が必要）。

高値と安値の間で指定された色のヒストグラムを描画する指標の例です。各曜日のヒストグラムの線の色は異なります。日の色、幅とスタイルは N ティックごとにランダムに変更されます。



DRAW_COLOR_HISTOGRAM2 スタイルを持つ **plot1** では、5つの色が #property, コンパイラディレクティブによって指定された後、OnCalculate() 関数で、colors[] 配列に格納された 14の色からランダムに設定されることにご注意ください。

N パラメータは手動設定の可能性を持って（指標の「プロパティ」ウィンドウの「パラメータ」タブ）指標 外部パラメータに設定されています。

```
//+------------------------------------------------------------------+
//|                                          DRAW_COLOR_HISTOGRAM2.mq5 |
```

© 2000–2025, MetaQuotes Ltd.

```
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "An indicator to demonstrate DRAW_COLOR_HISTOGRAM2"
#property description "It draws a segment between Open and Close on each bar"
#property description "The color, width and style are changed randomly"
#property description "after every N ticks"

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   1
//--- ColorHistogram_2 をプロットする
#property indicator_label1  "ColorHistogram_2"
#property indicator_type1   DRAW_COLOR_HISTOGRAM2
//--- 曜日のヒストグラムに色を塗るための(特別な配列に格納された)5色を定義する
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1

//--- 入力パラメータ
input int       N=5;              // ヒストグラムを変更するティック数
int             color_sections;
//--- 値バッファ
double          ColorHistogram_2Buffer1[];
double          ColorHistogram_2Buffer2[];
//--- カラーインデックスを格納するバッファ
double          ColorHistogram_2Colors[];
//--- 色を格納する14 要素の配列
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPu
rple
  };
//--- 線のスタイルを格納する配列
ENUM_LINE_STYLE styles[]
={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOTDOT};
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 指標バッファマッピング
   SetIndexBuffer(0,ColorHistogram_2Buffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ColorHistogram_2Buffer2,INDICATOR_DATA);
```

© 2000–2025, MetaQuotes Ltd.

```
   SetIndexBuffer(2,ColorHistogram_2Colors,INDICATOR_COLOR_INDEX);
//--- 空の値を設定
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//----正弦波を塗る色の数
   color_sections=8;   //   #property indicator_color1 コメントを参照
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- 線のスタイル、色、及び幅を変更するティックを計算する
   ticks++;
//--- 充分なティックの数が蓄積されている場合
   if(ticks>=N)
     {
      //--- 線のプロパティを変更する
      ChangeLineAppearance();
      //--- ヒストグラムを描画する色を変更する
      ChangeColors(colors,color_sections);
      //--- ティックカウンタをゼロにリセットする
      ticks=0;
     }

//--- 指標値を計算する
   int start=0;
//--- 各足の始値で曜日を取得する
   MqlDateTime dt;
//--- すでにOnCalculate の前回の開始時に算出した場合
   if(prev_calculated>0) start=prev_calculated-1; // 最後から2 つ目のバーから計算する
//--- 指標バッファに値を記入する
   for(int i=start;i<rates_total;i++)
     {
      TimeToStruct(time[i],dt);
      //--- 値
      ColorHistogram_2Buffer1[i]=high[i];
      ColorHistogram_2Buffer2[i]=low[i];
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 曜日に応じて色のインデックスを設定する
      int day=dt.day_of_week;
      ColorHistogram_2Colors[i]=day;
     }
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| 線分の色を変更する                                               |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- 色の数
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- それぞれのカラーインデックに新しい色をランダムに定義する
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- 乱数を取得
      int number=MathRand();
      //--- 整数除算の余りから col[] 配列のインデックスを取得
      int i=number%size;
      //--- 各インデックスの色をPLOT_LINE_COLOR として設定する
      PlotIndexSetInteger(0,                    //  グラフィックスタイルの番号
                          PLOT_LINE_COLOR,      //  プロパティの識別子
                          plot_color_ind,       //  色を書き込む色インデックス
                          cols[i]);             //  新しい色
      //--- 色を書く
      comm=comm+StringFormat("HistogramColorIndex[%d]=%s
\r\n",plot_color_ind,ColorToString(cols[i],true));
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| 指標に表示された線の外観を変更する                               |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- 線のプロパティに関する情報を形成するための文字列
   string comm="";
//--- 線の幅を変更するブロック
   int number=MathRand();
//--- 整数除算の余りの幅を取得
   int width=number%5; // 幅は 0～4 に設定される
//--- 色を PLOT_LINE_WIDTH プロパティに設定
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 線の幅を書く
   comm=comm+" Width="+IntegerToString(width);

//--- 線のスタイルを変更するブロック
   number=MathRand();
//--- 除数は、スタイルの配列の大きさに等しい
   int size=ArraySize(styles);
//--- 新しいスタイルを選択するためのインデックスを整数除算の余りから取得
   int style_index=number%size;
//--- 色をPLOT_LINE_COLOR プロパティとして設定
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- 線のスタイルを書く
   comm=EnumToString(styles[style_index])+", "+comm;
//--- コメントを使用して、チャート上の情報を表示する
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_ARROW

DRAW_COLOR_ARROW スタイルは指標バッファの値に基づいて色付きの矢印（Wingdings フォントシンボルのセットを描画します。このスタイルでは DRAW_ARROW と異なり各シンボルのindicator_color1 プロパティで指定された色の所定のセットでの色の設定が可能です。

シンボルの幅と色は DRAW_ARROW スタイルの様にコンパイラディレクティブまたは 動的に PlotIndexSetInteger() 関数を使用して指定することが出来ます。プロットプロパティの動的な変更は、現在の状況に基づいての指標の外観の変更を可能にします。

シンボルコードは PLOT_ARROW プロパティを使用して設定されます。

```
//--- PLOT_ARROW の描画のためにWingdings フォントからのシンボルコードを定義する
   PlotIndexSetInteger(0,PLOT_ARROW,code);
```

PLOT_ARROW=159（丸）がデフォルトです。

矢印は、実際には、高さとアンカーポイントを持つシンボルであり、チャート上のいくつかの重要な情報（例えばバーの終値）をカバーすることが出来ます。従って、チャートの規模に依存しないピクセル単位での垂直シフトの指定が可能です。指標値はそのままですが、矢印が特定のピクセル数だけ下に移動します。

```
//--- 矢印の垂直シフトをピクセル単位で設定する
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
```

PLOT_ARROW_SHIFT の負の値は矢印の上向きのシフト、正の値は矢印の下向きのシフトを意味します。

DRAW_COLOR_ARROW スタイルはチャートの別々のサブウィンドウとメインウィンドウで使用することが出来ます。空の値は描画されず「データウィンドウ」に表示されません。指標バッファ内の全ての値は明示的に設定する必要があります。バッファはゼロで初期化されません。

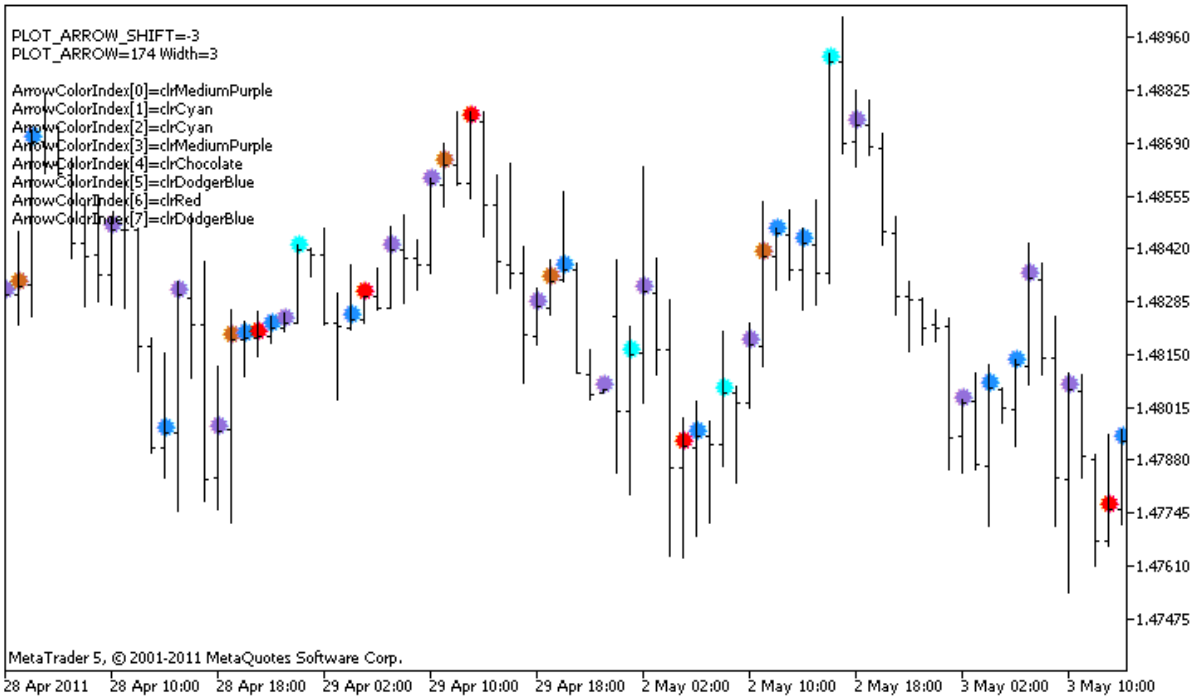
```
//--- 空の値を設定
   PlotIndexSetDouble(DRAW_COLOR_ARROW_plot_index,PLOT_EMPTY_VALUE,0);
```

DRAW_COLOR_ARROW のプロットに必要なバッファの数は 2 です。

- （PLOT_ARROW_SHIFT プロパティで指定されたピクセル単位でのシフトとともに）記号を描画するために使用される価格の値を格納するバッファ
- 矢印の描画に使用されるカラーインデックスを格納するバッファ（空値以外が必要）

前のバーの終値よりも高い終値で各足の矢印を描画する指標の例です。矢印の幅、シフトとシンボルコードは全て N ティックごとにランダムに変更されます。記号の色は、描画されたバーの数に依存します。

© 2000–2025, MetaQuotes Ltd.



この例では、DRAW_COLOR_ARROW スタイルを持つplot1 では、プロパティ、色とサイズは #property コンパイラディレクティブを使用して指定され、プロパティは OnCalculate() 関数でランダムに設定されます。N パラメータは手動設定の可能性を持って（指標の「プロパティ」ウィンドウの「パラメータ」タブ）指標外部パラメータに設定されています。

初めに8つの色が #property コンパイラディレクティブによって指定された後、OnCalculate() 関数で、colors[] 配列に格納された 14の色からランダムに設定されることにご注意ください。

```
//+------------------------------------------------------------------+
//|                                                DRAW_COLOR_ARROW.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "An indicator to demonstrate DRAW_COLOR_ARROW"
#property description "Draws different-color arrows set by Unicode characters, on a chart"
#property description "The color, size, shift and symbol code of the arrow are changed"
#property description " randomly every N ticks"
#property description "The code parameter sets the base value: code=159 (a circle)"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- ColorArrowt をプロットする
#property indicator_label1  "ColorArrow"
```

```
#property indicator_type1   DRAW_COLOR_ARROW
//--- ヒストグラムを塗るために(特殊な配列に格納されている)8 色を定義する
#property indicator_color1
clrRed,clrBlue,clrSeaGreen,clrGold,clrDarkOrange,clrMagenta,clrYellowGreen,clrChocol
ate
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1

//--- 入力パラメータ
input int       N=5;          // 変化をもたらすティックの数
input ushort    code=159;     // DRAW_ARROW で描画するシンボルコード
int             color_sections;
//--- プロットの指標バッファ
double          ColorArrowBuffer[];
//--- カラーインデックスを格納するバッファ
double          ColorArrowColors[];
//--- 色を格納する14 要素の配列
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPu
rple
  };
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 指標バッファマッピング
   SetIndexBuffer(0,ColorArrowBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorArrowColors,INDICATOR_COLOR_INDEX);
//--- PLOT_ARROW の描画のためにシンボルコードを定義する
   PlotIndexSetInteger(0,PLOT_ARROW,code);
//--- 矢印の垂直シフトをピクセル単位で設定する
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,5);
//--- 空の値の 0 を設定
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//----正弦波を塗る色の数
   color_sections=8;   //  #property indicator_color1 のコメントを参照
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- 矢印の色、大きさ、及びシフトコードを変更するティックを計算する
   ticks++;
//--- 充分なティックの数が蓄積されている場合
   if(ticks>=N)
     {
      //--- 矢印プロパティを変更する
      ChangeLineAppearance();
      //--- ヒストグラムを描画する色を変更する
      ChangeColors(colors,color_sections);
      //--- ティックカウンタをゼロにリセットする
      ticks=0;
     }

//--- 指標安倍を計算するブロック
   int start=1;
   if(prev_calculated>0) start=prev_calculated-1;
//--- 計算ループ
   for(int i=1;i<rates_total;i++)
     {
      //--- 現在の終値が以前の終値より高い場合、矢印を描画する
      if(close[i]>close[i-1])
         ColorArrowBuffer[i]=close[i];
      //--- さもなければ null を指定する
      else
         ColorArrowBuffer[i]=0;
      //--- 矢印の色
      int index=i%color_sections;
      ColorArrowColors[i]=index;
     }
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//|  線分の色を変更する                                              |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- 色の数
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";
```

© 2000-2025, MetaQuotes Ltd.

```
//--- それぞれのカラーインデックに新しい色をランダムに定義する
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- 乱数を取得
      int number=MathRand();
      //--- 整数除算の余りから col[] 配列のインデックスを取得
      int i=number%size;
      //--- 各インデックスの色をPLOT_LINE_COLOR として設定する
      PlotIndexSetInteger(0,                        //   グラフィックスタイルの番号
                          PLOT_LINE_COLOR,          //   プロパティの識別子
                          plot_color_ind,           //   色を書き込む色インデックス
                          cols[i]);                 //   新しい色
      //--- 色を書く
      comm=comm+StringFormat("ArrowColorIndex[%d]=%s
\r\n",plot_color_ind,ColorToString(cols[i],true));
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//|  指標に表示された線の外観を変更する                                |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- 線のプロパティに関する情報を形成するための文字列
   string comm="";
//--- 線の幅を変更するブロック
   int number=MathRand();
//--- 整数除算の余りの幅を取得
   int width=number%5; // 幅は 0～4 に設定される
//--- 色を PLOT_LINE_WIDTH プロパティに設定
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- 線の幅を書く
   comm=comm+" Width="+IntegerToString(width);

//--- 矢印のコードを変更するブロック(PLOT_ARROW)
   number=MathRand();
//--- 矢印の新しいコードを計算するために、整数の除算の剰余を取得(0～19)
   int code_add=number%20;
//--- 新しいシンボルコードをcode+code_add に設定
   PlotIndexSetInteger(0,PLOT_ARROW,code+code_add);
//--- PLOT_ARROW シンボルコードを書く
   comm="\r\n"+"PLOT_ARROW="+IntegerToString(code+code_add)+comm;

//--- 矢印の垂直シフトをピクセル単位で変更するブロック
   number=MathRand();
//--- 整数除算の余りとしてシフトを取得
   int shift=20-number%41;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 新しいシフトを設定
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
//--- PLOT_ARROW_SHIFT シフトを書く
   comm="\r\n"+"PLOT_ARROW_SHIFT="+IntegerToString(shift)+comm;

//--- コメントを使用して、チャート上の情報を表示する
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_ZIGZAG

DRAW_COLOR_ZIGZAG スタイルは 2 つの指標バッファの値に基づいて異なる色のセグメントを描画します。このスタイルはDRAW_ZIGZAG の色付きバージョンで、各セグメントの色を所定のセットから個々に指定することが出来ます。セグメントは、1 番目のバッファの値から2 番目の指標バッファの値にプロットされます。いずれのバッファも空の値のみを含むことは出来ません。この場合プロットがなされません。

線の幅、色とスタイルは DRAW_ZIGZAG スタイルのようにコンパイラディレクティブまたは 動的に PlotIndexSetInteger() 関数を使用して指定することが出来ます。プロットのプロパティの動的な変更は、指標の外観が現在の状況に応じて変わるようにして「盛り上げる」ことが出来ます。

セクションは指標バッファの空でない値からあと 1 つの空でない値まで描かれます。「空」値を指定するには PLOT_EMPTY_VALUE プロパティで値を設定します。

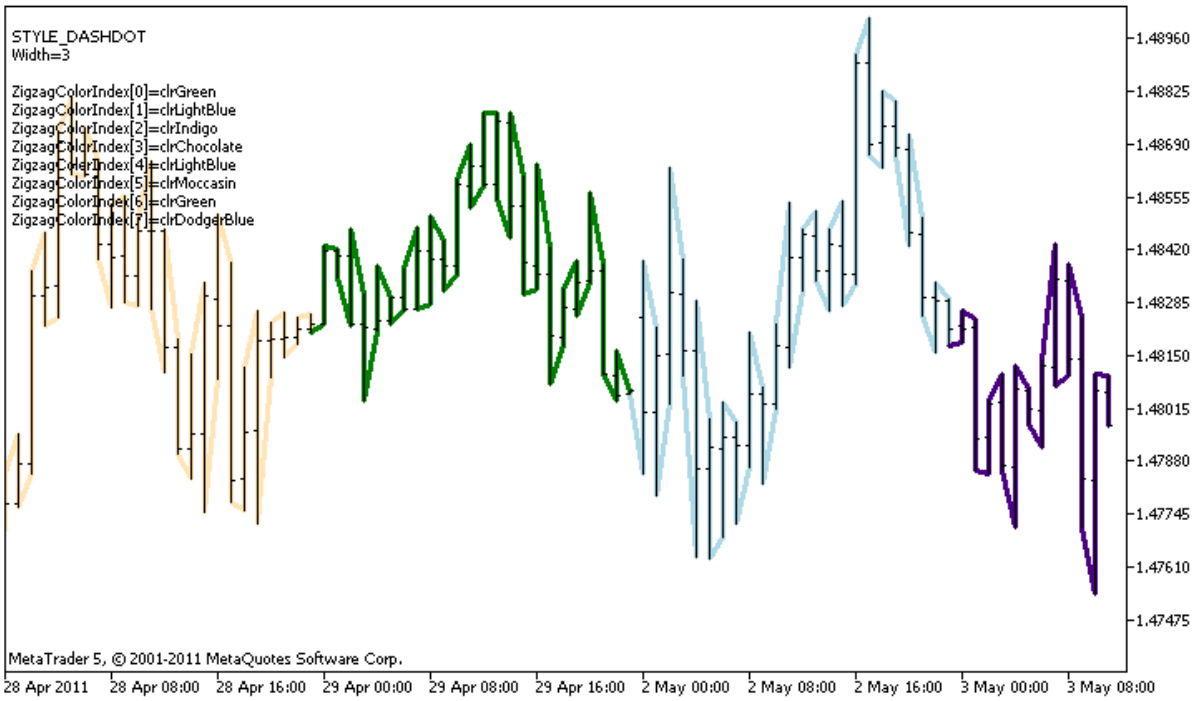
```
//--- 0(空)値は描画されない
   PlotIndexSetDouble(index_of_plot_DRAW_COLOR_ZIGZAG,PLOT_EMPTY_VALUE,0);
```

バッファの値は常に明示的に入力し、バーをスキップするにはバッファに空の値を設定します。

DRAW_COLOR_ZIGZAG のプロットに必要なバッファの数は 3 です。

- その内の 2 つは、ジグザグセクションの端を格納します。
- もう 1 つは、矢印の描画に使用されるカラーインデックスを格納します(空値以外が必要)。

高値と安値に基づいてのこぎりをプロットする指標の例。ジグザグ線の色、幅とスタイルは N ティックごとにランダムに変更されます。



DRAW_COLOR_ZIGZAG スタイルを持つ**plot1** では、8つの色が #property コンパイラディレクティブによって指定された後、OnCalculate() 関数で、colors[] 配列に格納された 14の色からランダムに設定されることにご注意ください。

© 2000–2025, MetaQuotes Ltd.

N パラメータは手動設定の可能性を持って（指標の「プロパティ」ウィンドウの「パラメータ」タブ）指標 外部パラメータに設定されています。

```
//+------------------------------------------------------------------+
//|                                                DRAW_COLOR_ZIGZAG.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "An indicator to demonstrate DRAW_COLOR_ZIGZAG"
#property description "It draws a broken line as a sequence of colored sections, the color depends on the number of the day of the week"
#property description "The color, width and style of segments are changed randomly"
#property description " every N ticks"

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   1
//--- Color_Zigzag をプロットする
#property indicator_label1  "Color_Zigzag"
#property indicator_type1   DRAW_COLOR_ZIGZAG
//--- セクションを塗るために(特殊な配列に格納されている)8 色を定義する
#property indicator_color1
clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrLime,clrOrange
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 入力パラメータ
input int       N=5;              // 変化をもたらすティックの数
int             color_sections;
//--- セグメントの端の値のバッファ
double          Color_ZigzagBuffer1[];
double          Color_ZigzagBuffer2[];
//--- セグメントの端のカラーインデックスを格納するバッファ
double          Color_ZigzagColors[];
//--- 色を格納する14 要素の配列
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurple
  };
//--- 線のスタイルを格納する配列
ENUM_LINE_STYLE styles[]
={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOTDOT};
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
int OnInit()
  {
//--- 指標バッファマッピング
   SetIndexBuffer(0,Color_ZigzagBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,Color_ZigzagBuffer2,INDICATOR_DATA);
   SetIndexBuffer(2,Color_ZigzagColors,INDICATOR_COLOR_INDEX);
//----ジグザグを塗る色の数
   color_sections=8;   //  #property indicator_color1 プロパティのコメントを参照
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- 線のスタイル、色、及び幅を変更するティックを計算する
   ticks++;
//--- 充分なティックの数が蓄積されている場合
   if(ticks>=N)
     {
      //--- 線のプロパティを変更する
      ChangeLineAppearance();
      //--- セクションをプロットする色を変更する
      ChangeColors(colors,color_sections);
      //--- ティックカウンタをゼロにリセットする
      ticks=0;
     }

//--- 各足の曜日を取得するのに時間の構造体が必要
   MqlDateTime dt;

//--- 計算開始位置
   int start=0;
//--- 指標が以前に計算されている場合には、計算開始点を最後から2 つ目のバーに設定する
   if(prev_calculated!=0) start=prev_calculated-1;
//--- 計算ループ
   for(int i=start;i<rates_total;i++)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- バーのオープン時間を書く
      TimeToStruct(time[i],dt);

      //--- バー番号が偶数の場合
      if(i%2==0)
        {
         //---  1 番目のバッファの高値と2 番目のバッファの安値を書く
         Color_ZigzagBuffer1[i]=high[i];
         Color_ZigzagBuffer2[i]=low[i];
         //--- セグメントの色
         Color_ZigzagColors[i]=dt.day_of_year%color_sections;
        }
      //--- バー番号が奇数
      else
        {
         //--- 逆の順序でバーに記入
         Color_ZigzagBuffer1[i]=low[i];
         Color_ZigzagBuffer2[i]=high[i];
         //--- セグメントの色
         Color_ZigzagColors[i]=dt.day_of_year%color_sections;
        }
     }
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| ジグザグセグメントの色を変更する                                 |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- 色の数
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- それぞれのカラーインデックスに新しい色をランダムに定義する
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- 乱数を取得
      int number=MathRand();
      //--- 整数除算の余りから col[] 配列のインデックスを取得
      int i=number%size;
      //--- 各インデックスの色をPLOT_LINE_COLOR として設定する
      PlotIndexSetInteger(0,                        //  グラフィックスタイルの番号
                          PLOT_LINE_COLOR,          //  プロパティの識別子
                          plot_color_ind,           //  色を書き込む色インデックス
                          cols[i]);                 //  新しい色
      //--- 色を書く
```

© 2000-2025, MetaQuotes Ltd.

```
      comm=comm+StringFormat("ZigzagColorIndex[%d]=%s
\r\n",plot_color_ind,ColorToString(cols[i],true));
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| ジグザグセグメントの外観を変更する                                |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Color_ZigZag のプロパティに関する情報を形成するための文字列
   string comm="";
//--- 線の幅を変更するブロック
   int number=MathRand();
//--- 整数除算の余りの幅を取得
   int width=number%5;    // 幅は 0～4 に設定される
//--- 色を PLOT_LINE_WIDTH プロパティに設定
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- 線の幅を書く
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- 線のスタイルを変更するブロック
   number=MathRand();
//--- 除数は、スタイルの配列の大きさに等しい
   int size=ArraySize(styles);
//--- 新しいスタイルを選択するためのインデックスを整数除算の余りから取得
   int style_index=number%size;
//--- 色をPLOT_LINE_COLOR プロパティとして設定
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- 線のスタイルを書く
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- コメントを使用して、チャート上の情報を表示する
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_BARS

DRAW_COLOR_BARS スタイルは、始値、高値、安値、終値の値を含む 4 つの指標バッファからバーを描画します。このスタイルは DRAW_BARS の高度なバージョンで、各足の色を所定のセットから個々に指定することが出来ます。これは、チャートの別々のサブウィンドウ内及びその他の金融商品のカスタム指標をバーとして作成するために使用されます。

バーの色はコンパイラディレクティブまたは動的に PlotIndexSetInteger() 関数で設定出来ます。プロットのプロパティの動的な変更は、指標の外観が現在の状況に応じて変わるようにして「盛り上げる」ことが出来ます。

指標は 4 つの指標バッファ全部に空以外の値があるバーにのみ描画されます。「空」値を指定するには PLOT_EMPTY_VALUE プロパティで値を設定します。

```
//--- 0(空)値は描画されない
   PlotIndexSetDouble(index_of_plot_DRAW_COLOR_BARS,PLOT_EMPTY_VALUE,0);
```

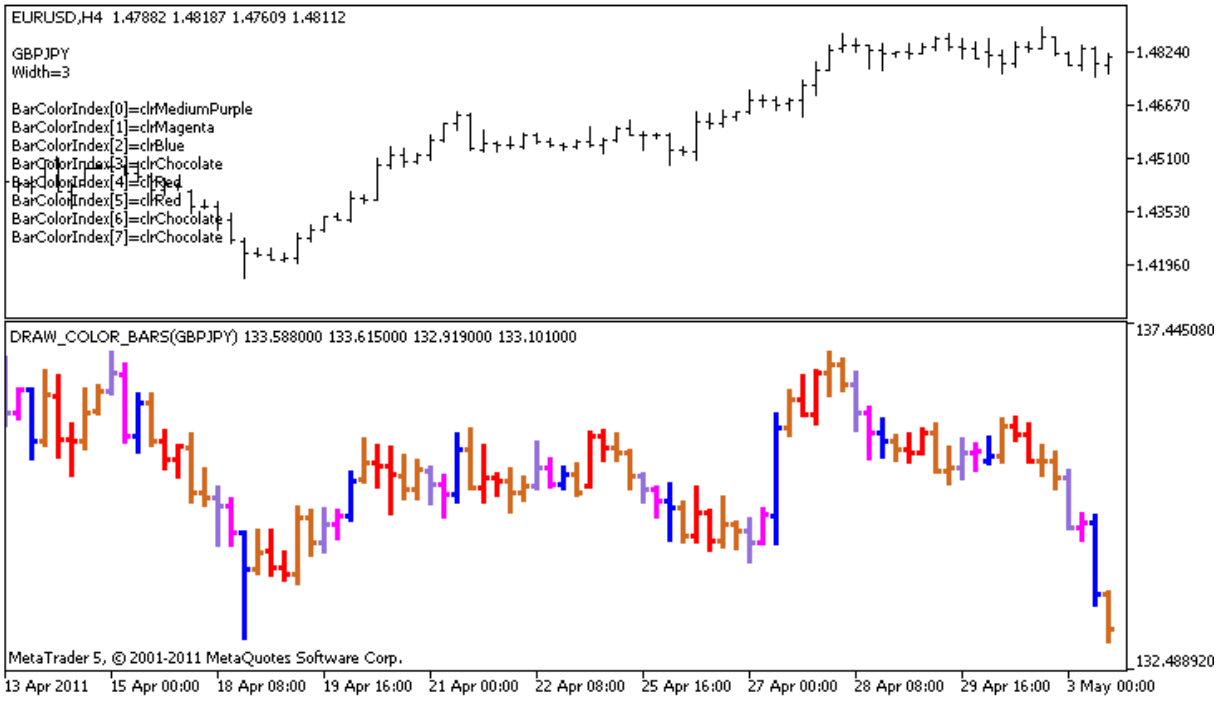
バッファの値は常に明示的に入力し、バーをスキップするにはバッファに空の値を設定します。

DRAW_COLOR_BARS のプロットに必要なバッファの数は5です。

- 始値、高値、安値、終値を格納する 4 つのバッファ
- バーを描画するのに使用される色のインデックスを格納するための 1 つのバッファ(描画されるバーのためのみに設定する)

プロットバッファは、始値、高値、安値、終値、カラーバッファの順番であるべきです。いずれのバッファも null のみを含むことは出来ません。この場合プロットがなされません。

別のウィンドウで選択された金融商品のバーを描画する指標の例です。バーの色は N ティックごとにランダムに変わります。パラメータは手動設定の可能性を持って(指標の「プロパティ」ウィンドウの「パラメータ」タブ)指標外部パラメータに設定されています。



© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_BARS スタイルを持つ plot1 では、8つの色が #property コンパイラディレクティブによって指定された後、OnCalculate() 関数で、colors[] 配列に格納された 14の色からランダムに設定されることにご注意ください。

```
//+------------------------------------------------------------------+
//|                                                DRAW_COLOR_BARS.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "An indicator to demonstrate DRAW_COLOR_BARS"
#property description "It draws different-color bars of a selected symbol in a separate window"
#property description "The color and width of bars, as well as the symbol are changed randomly"
#property description "every N ticks"

#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots   1
//--- ColorBars をプロットする
#property indicator_label1  "ColorBars"
#property indicator_type1   DRAW_COLOR_BARS
//--- バーを塗るために(特殊な配列に格納されている)8 色を定義する
#property indicator_color1
clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrLime,clrOrange
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 入力パラメータ
input int      N=5;              // 種類を変更するティック数
input int      bars=500;         // 表示するバーの数
input bool     messages=false;   // エキスパートアドバイザー操作ログにメッセージを表示する
//--- 指標バッファ
double         ColorBarsBuffer1[];
double         ColorBarsBuffer2[];
double         ColorBarsBuffer3[];
double         ColorBarsBuffer4[];
double         ColorBarsColors[];
//--- 銘柄名
string symbol;
int    bars_colors;
//--- 色を格納する14 要素の配列
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrMagenta,clrCyan,clrMediumPurple
```

© 2000-2025, MetaQuotes Ltd.

```
  };
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 指標バッファマッピング
   SetIndexBuffer(0,ColorBarsBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ColorBarsBuffer2,INDICATOR_DATA);
   SetIndexBuffer(2,ColorBarsBuffer3,INDICATOR_DATA);
   SetIndexBuffer(3,ColorBarsBuffer4,INDICATOR_DATA);
   SetIndexBuffer(4,ColorBarsColors,INDICATOR_COLOR_INDEX);
//---- バーを塗る色の数
   bars_colors=8;    // #property indicator_color1 プロパティのコメントを参照
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- バーのスタイル、色、及び幅を変更するティックを数える
   ticks++;
//--- 充分なティックの数が蓄積されている場合
   if(ticks>=N)
     {
      //---「気配値表示」ウィンドウで新しいシンボルを選ぶ
      symbol=GetRandomSymbolName();
      //--- 線のプロパティを変更する
      ChangeLineAppearance();
      //--- ローソク足を塗る色を変更する
      ChangeColors(colors,bars_colors);
      int tries=0;
      //--- シンボルからの価格でバッファを埋めることを五回試みる
      while(!CopyFromSymbolToBuffers(symbol,rates_total,bars_colors) && tries<5)
        {
         //--- CopyFromSymbolToBuffers() 関数呼び出しのカウンタ
         tries++;
```

© 2000-2025, MetaQuotes Ltd.

```
         }
       //--- ティックカウンタをゼロにリセットする
       ticks=0;
      }
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//|  指標バッファに価格を記入する                                    |
//+------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,int total,int bar_colors)
  {
//--- rates[] 配列で、始値、高値、安値、終値を複製する
   MqlRates rates[];
//--- 試みのカウンタ
   int attempts=0;
//--- 複製された量
   int copied=0;
//--- 希望する記号の時系列を取得するための25の試みをする
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) attempts=%
d",__FUNCTION__,name,attempts);
     }
//--- 充分な数のバーの複製が失敗した場合
   if(copied!=bars)
     {
      //--- メッセージ文字列を作る
      string comm=StringFormat("For the symbol %s, managed to receive only %d bars
of %d requested ones",
                               name,
                               copied,
                               bars
                               );
      //--- メインチャートウィンドウでコメントを表示する
      Comment(comm);
      //--- メッセージを表示する
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- シンボルの表示を設定する
      PlotIndexSetString(0,PLOT_LABEL,name+" Open;"+name+" High;"+name+"
Low;"+name+" Close");
      IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_BARS("+name+")");
     }
```

© 2000–2025, MetaQuotes Ltd.

```
//--- バッファを空の値で初期化する
   ArrayInitialize(ColorBarsBuffer1,0.0);
   ArrayInitialize(ColorBarsBuffer2,0.0);
   ArrayInitialize(ColorBarsBuffer3,0.0);
   ArrayInitialize(ColorBarsBuffer4,0.0);

//--- 価格をバッファに複製する
   for(int i=0;i<copied;i++)
     {
      //--- バッファの適切なインデックスを計算する
      int buffer_index=total-copied+i;
      //--- 価格をバッファに書く
      ColorBarsBuffer1[buffer_index]=rates[i].open;
      ColorBarsBuffer2[buffer_index]=rates[i].high;
      ColorBarsBuffer3[buffer_index]=rates[i].low;
      ColorBarsBuffer4[buffer_index]=rates[i].close;
      //---
      ColorBarsColors[buffer_index]=i%bar_colors;
     }
   return(true);
  }
//+------------------------------------------------------------------+
//| 「気配値表示」からランダムにシンボルを返す                                |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//---「気配値表示」ウィンドウに表示されるシンボルの数
   int symbols=SymbolsTotal(true);
//--- リスト内のシンボルの位置 - 0～シンボル数の乱数
   int number=MathRand()%symbols;
//--- 指定された位置のシンボルの名称を返す
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//| ジグザグセグメントの色を変更する                                       |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- 色の数
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- それぞれのカラーインデックに新しい色をランダムに定義する
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- 乱数を取得
      int number=MathRand();
      //--- 整数除算の余りから col[] 配列のインデックスを取得
```

© 2000-2025, MetaQuotes Ltd.

```
      int i=number%size;
      //--- 各インデックスの色をPLOT_LINE_COLOR として設定する
      PlotIndexSetInteger(0,                          //   グラフィックスタイルの番号
                          PLOT_LINE_COLOR,            //   プロパティの識別子
                          plot_color_ind,             //   色を書き込む色インデックス
                          cols[i]);                   //   新しい色
      //--- 色を書く
      comm=comm+StringFormat("BarColorIndex[%d]=%s
\r\n",plot_color_ind,ColorToString(cols[i],true));
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//|  バーの外観を変更する                                            |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- バーのプロパティに関する情報を形成するための文字列
   string comm="";

//--- バーの幅を変更するブロック
   int number=MathRand();
//--- 整数除算の余りの幅を取得
   int width=number%5;    // 幅は 0～4 に設定される
//--- 色を PLOT_LINE_WIDTH プロパティに設定
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- 線の幅を書く
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- 銘柄名を書く
   comm="\r\n"+symbol+comm;

//--- コメントを使用して、チャート上の情報を表示する
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_CANDLES

DRAW_COLOR_CANDLES スタイルは、DRAW_CANDLES の様に始値、高値、安値、終値の値を含む 4 つの指標バッファからローソク足を描画します。更に、指定されたセットから各ローソク足の色を指定することが出来ます。この目的のために、このスタイルには各足の色のインデックスを格納する特殊なカラーバッファがあります。これは、チャートの別々のサブウィンドウ内及びその他の金融商品のカスタム指標をローソク足として作成するために使用されます。

ローソク足の色の数はコンパイラディレクティブまたは動的に PlotIndexSetInteger() 関数で設定出来ます。プロットのプロパティの動的な変更は、指標の外観が現在の状況に応じて変わるようにして「盛り上げる」ことが出来ます。

指標は 4 つの価格バッファ全部に空以外の値があるバーにのみ描画されます。「空」値を指定するには PLOT_EMPTY_VALUE プロパティで値を設定します。

```
//--- 0(空)値は描画されない
   PlotIndexSetDouble(index_of_plot_DRAW_COLOR_CANDLES,PLOT_EMPTY_VALUE,0);
```

バッファの値は常に明示的に入力し、バーをスキップするにはバッファに空の値を設定します。

DRAW_COLOR_CANDLES のプロットに必要なバッファの数は5です。

- 始値、高値、安値、終値を格納する 4 つのバッファ
- ローソク足を描画するのに使用される色のインデックスを格納するための 1 つのバッファ(描画されるバーのためのみに設定する)

プロットバッファは、始値、高値、安値、終値、カラーバッファの順番であるべきです。いずれのバッファも空の値のみを含むことは出来ません。この場合プロットがなされません。

別のウィンドウで選択された金融商品のローソク足を描画する指標の例です。ローソク足の色はN ティックごとにランダムに変わります。パラメータは手動設定の可能性を持って(指標の「プロパティ」ウィンドウの「パラメータ」タブ)指標外部パラメータに設定されています。



© 2000-2025, MetaQuotes Ltd.

plot1では、色が #property コンパイラディレクティブによって指定された後、OnCalculate() 関数で事前に準備されたリストからランダムに設定されます。

```
//+------------------------------------------------------------------+
//|                                            DRAW_COLOR_CANDLES.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "An indicator to demonstrate DRAW_COLOR_CANDLES."
#property description "It draws candlesticks of a selected symbol in a separate window"
#property description " "
#property description "The color and width of candlesticks, as well as the symbol are changed"
#property description "randomly every N ticks"

#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots   1
//---- ColorCandles をプロットする
#property indicator_label1  "ColorCandles"
#property indicator_type1   DRAW_COLOR_CANDLES
//--- ローソク足を塗るために(特殊な配列に格納されている)8 色を定義する
#property indicator_color1
clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrLime,clrOrange
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1

//--- 入力パラメータ
input int      N=5;              // 種類を変更するティック数
input int      bars=500;         // 表示するローソク足の数
input bool     messages=false;   // エキスパートアドバイザー操作ログにメッセージを表示する
//--- 指標バッファ
double         ColorCandlesBuffer1[];
double         ColorCandlesBuffer2[];
double         ColorCandlesBuffer3[];
double         ColorCandlesBuffer4[];
double         ColorCandlesColors[];
int            candles_colors;
//--- 銘柄名
string symbol;
//--- 色を格納する14 要素の配列
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
```

© 2000-2025, MetaQuotes Ltd.

```
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrMagenta,clrCyan,clrMediumPurpl
e
  };
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- バーが小さい場合は、前もって完成する
   if(bars<50)
     {
      Comment("Please specify a larger number of bars! The operation of the
indicator has been terminated");
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- 指標バッファマッピング
   SetIndexBuffer(0,ColorCandlesBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ColorCandlesBuffer2,INDICATOR_DATA);
   SetIndexBuffer(2,ColorCandlesBuffer3,INDICATOR_DATA);
   SetIndexBuffer(3,ColorCandlesBuffer4,INDICATOR_DATA);
   SetIndexBuffer(4,ColorCandlesColors,INDICATOR_COLOR_INDEX);
//--- 空の値
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- バーが描画される銘柄名
   symbol=_Symbol;
//--- シンボルの表示を設定
   PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;"+symbol+" High;"+symbol+"
Low;"+symbol+" Close");
   IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_CANDLES("+symbol+")");
//---- ローソク足を塗る色の数
   candles_colors=8;     //    プロパティのコメントを参照
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//|  カスタム指標の反復関数                                          |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=INT_MAX-100;
```

© 2000–2025, MetaQuotes Ltd.

```
//--- スタイルと色を変更するティックを数える
   ticks++;
//--- 充分なティックの数が蓄積されている場合
   if(ticks>=N)
     {
      //---「気配値表示」ウィンドウで新しいシンボルを選ぶ
      symbol=GetRandomSymbolName();
      //--- 形を変更する
      ChangeLineAppearance();
      //--- ローソク足を塗る色を変更する
      ChangeColors(colors,candles_colors);

      int tries=0;
      //--- シンボルからの価格で plot1 バッファを埋めることを五回試みる
      while(!CopyFromSymbolToBuffers(symbol,rates_total,0,
            ColorCandlesBuffer1,ColorCandlesBuffer2,ColorCandlesBuffer3,
            ColorCandlesBuffer4,ColorCandlesColors,candles_colors)
            && tries<5)
        {
         //--- CopyFromSymbolToBuffers() 関数呼び出しのカウンタ
         tries++;
        }
      //--- ティックカウンタをゼロにリセットする
      ticks=0;
     }
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//|  指定されたローソク足に書き込む                                  |
//+------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,
                             int total,
                             int plot_index,
                             double &buff1[],
                             double &buff2[],
                             double &buff3[],
                             double &buff4[],
                             double &col_buffer[],
                             int    cndl_colors
                            )
  {
//--- rates[] 配列で、始値、高値、安値、終値を複製する
   MqlRates rates[];
//--- 試みのカウンタ
   int attempts=0;
//--- 複製された量
   int copied=0;
//--- 希望する記号の時系列を取得するための25の試みをする
```

© 2000-2025, MetaQuotes Ltd.

```
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) attempts=%
d",__FUNCTION__,name,attempts);
     }
//--- 充分な数のバーの複製が失敗した場合
   if(copied!=bars)
     {
      //--- メッセージ文字列を作る
      string comm=StringFormat("For the symbol %s, managed to receive only %d bars
of %d requested ones",
                               name,
                               copied,
                               bars
                               );
      //--- メインチャートウィンドウでコメントを表示する
      Comment(comm);
      //--- メッセージを表示する
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- シンボルの表示を設定する
      PlotIndexSetString(plot_index,PLOT_LABEL,name+" Open;"+name+" High;"+name+"
Low;"+name+" Close");
      IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_CANDLES("+symbol+")");
     }
//--- バッファを空の値で初期化する
   ArrayInitialize(buff1,0.0);
   ArrayInitialize(buff2,0.0);
   ArrayInitialize(buff3,0.0);
   ArrayInitialize(buff4,0.0);
//--- ティックのたびに価格をバッファに複製する
   for(int i=0;i<copied;i++)
     {
      //--- バッファの適切なインデックスを計算する
      int buffer_index=total-copied+i;
      //--- 価格をバッファに書く
      buff1[buffer_index]=rates[i].open;
      buff2[buffer_index]=rates[i].high;
      buff3[buffer_index]=rates[i].low;
      buff4[buffer_index]=rates[i].close;
      //--- Set the candlestick color
      int color_index=i%cndl_colors;
      col_buffer[buffer_index]=color_index;
     }
```

© 2000–2025, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| 「気配値表示」からランダムにシンボルを返す                       |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- 「気配値表示」ウィンドウに表示されるシンボルの数
   int symbols=SymbolsTotal(true);
//--- リスト内のシンボルの位置 - 0～シンボル数の乱数
   int number=MathRand()%symbols;
//--- 指定された位置のシンボルの名称を返す
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//| ローソク足セグメントの色を変更する                               |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- 色の数
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- それぞれのカラーインデックスに新しい色をランダムに定義する
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- 乱数を取得
      int number=MathRand();
      //--- 整数除算の余りから col[] 配列のインデックスを取得
      int i=number%size;
      //--- 各インデックスの色をPLOT_LINE_COLOR として設定する
      PlotIndexSetInteger(0,                    //  グラフィックスタイルの番号
                          PLOT_LINE_COLOR,      //  プロパティの識別子
                          plot_color_ind,       //  色を書き込む色インデックス
                          cols[i]);             //  新しい色
      //--- 色を書く
      comm=comm+StringFormat("CandleColorIndex[%d]=%s
\r\n",plot_color_ind,ColorToString(cols[i],true));
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| ローソク足の外観を変更する                                       |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- ローソク足のプロパティに関する情報を形成するための文字列
```

© 2000-2025, MetaQuotes Ltd.

```
   string comm="";
//--- 銘柄名を書く
   comm="\r\n"+symbol+comm;
//--- コメントを使用して、チャート上の情報を表示する
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

指標プロパティと対応した関数との関係

カスタム指標にはたくさんのプロパティがあります。そのうちのいくつかは必須で、常に記述の先頭に配置されます。下記が必須プロパティです。

- 指標プロットに使用されるウィンドウの表示 – indicator_separate_window または indicator_chart_window
- 指標バッファの数 – indicator_buffers
- 指標のプロット数 – indicator_plots

プリプロセッサ ディレクティブとカスタム指標作成に意図された関数を通じて設定出来るプロパティもあります。これらのプロパティと対応する機能を以下の表に記載されています。

指標サブウィンドウのプロパティに関する指令	IndicatorSet...() 型の関数	調整されたサブウィンドウプロパティの説明
indicator_height	IndicatorSetInteger (INDICATOR_INDICATOR_HEIGHT, nHeight)	サブウィンドウの高さの固定値
indicator_minimum	IndicatorSetDouble (INDICATOR_MINIMUM, dMaxValue)	縦軸の最小値
indicator_maximum	IndicatorSetDouble (INDICATOR_MAXIMUM, dMinValue)	縦軸の最大値
indicator_levelN	IndicatorSetDouble (INDICATOR_LEVELVALUE, N-1, nLevelValue)	N レベルの縦軸の値
プリプロセッサディレクティブなし	IndicatorSetString (INDICATOR_LEVELTEXT, N-1, sLevelName)	表示されたレベルの名称
indicator_levelcolor	IndicatorSetInteger (INDICATOR_LEVELCOLOR, N-1, nLevelColor)	N レベルの色
indicator_levelwidth	IndicatorSetInteger (INDICATOR_LEVELWIDTH, N-1, nLevelWidth)	N レベルの線の幅
indicator_levelstyle	IndicatorSetInteger (INDICATOR_LEVELSTYLE, N-1, nLevelStyle)	N レベルの線のスタイル
プロットプロパティに関する指令	**PlotIndexSet...() 型の関数**	**調整されたプロットプロパティの説明**
indicator_labelN	PlotIndexSetString(N-1,PLOT_LABEL,sLabel)	N 番目のプロットの短縮名（データウィンドウ、及び、マウスカーソルのホーバー時にポップアップツールヒントで表示）

© 2000–2025, MetaQuotes Ltd.

指標サブウィンドウのプロパティに関する指令	IndicatorSet...() 型の関数	調整されたサブウィンドウプロパティの説明
indicator_colorN	PlotIndexSetInteger(N-1, PLOT_LINE_COLOR, nColor)	N番目のプロットの線の色
indicator_styleN	PlotIndexSetInteger(N-1, PLOT_LINE_STYLE, nType)	N番目のプロットの線のスタイル
indicator_typeN	PlotIndexSetInteger(N-1, PLOT_DRAW_TYPE, nType)	N番目のプロットの線の種類
indicator_widthN	PlotIndexSetInteger(N-1, PLOT_LINE_WIDTH, nWidth)	N番目のプロットの線の幅
指標の共通プロパティ	**IndicatorSet...() 型の関数**	**説明**
プリプロセッサディレクティブなし	IndicatorSetString (INDICATOR_SHORTNAME, sShortName)	端末でCtrl+Iを押すことによって表示される指標リストでの指標の短縮名を設定します。
プリプロセッサディレクティブなし	IndicatorSetInteger (INDICATOR_DIGITS, nDigits)	小数の桁数として、指標値の表示に必要な精度を設定します。
プリプロセッサディレクティブなし	IndicatorSetInteger (INDICATOR_LEVELS, nLevels)	指標ウィンドウのレベルの数を設定します。
indicator_applied_price	このプロパティはプリプロセッサディレクティブのみで設定可能で関数はありません。	指標計算に使用されるデフォルトの価格型。OnCalculate() の1番目の種類が使用された時のみ、必要に応じて指定されます。 プロパティ値は指標プロパティダイアログの「パラメータ」タブの「適用価格」で設定します。

プリプロセッサではレベルやプロットの番号付けが1から始まる一方、関数で同じプロパティを使用する場合番号付けが0から始まることには留意されるべきです。

対応する関数のないディレクティブはいくつかあります。

ディレクティブ	説明
indicator_chart_window	メインウィンドウでの指標の表示
indicator_separate_window	別のサブウィンドウでの指標の表示
indicator_buffers	必要な指標バッファの数
indicator_plots	指標でのプロットの数

© 2000-2025, MetaQuotes Ltd.

SetIndexBuffer

この関数は、指定された指標バッファとdouble 型の1次元動的配列を関連付けます。

```
bool  SetIndexBuffer(
   int                   index,          // バッファインデックス
   double                buffer[],       // 配列
   ENUM_INDEXBUFFER_TYPE data_type       // 格納されるもの
   );
```

パラメータ

index

[in] 指標バッファの数番号付けは0から始まります。数は #property indicator_buffers で宣言された数より少なければなりません。

buffer[]

[in] カスタム指標プログラムで宣言された配列

data_type

[in] 指標配列に格納されたデータの型。デフォルトでは INDICATOR_DATA（カスタム指標値）。値は INDICATOR_COLOR_INDEX でも可能で、その場合、このバッファは前の指標バッファのカラーインデックスを格納します。#property indicator_colorN 行では 64 までの色の指定が可能です。INDICATOR_CALCULATIONS 値はバッファが指標の中間計算に使用されて、描画の意図外であることを示します。

戻り値

成功の場合 true、それ以外の場合は false

注意事項

結合後、buffer[] 動的配列は、結合された配列が時系列式のインデックスを持っていても、通常の配列のように索引付けされます。指標配列の要素アクセスの順序を変えたい場合は、SetIndexBuffer() 関数で配列が結合された後でArraySetAsSeries() 関数を使用します。 指標バッファをして使用されている動的配列のサイズは SetIndexBuffer() 関数では変更出来ないのでご注意下さい。指標バッファサイズ変更の操作は、全て端末の実行サブシステムによって実行されます。

例:

```
//+------------------------------------------------------------------+
//|                                                TestCopyBuffer1.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- MA をプロットする
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_label1  "MA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 入力パラメータ
input bool                AsSeries=true;
input int                 period=15;
input ENUM_MA_METHOD      smootMode=MODE_EMA;
input ENUM_APPLIED_PRICE  price=PRICE_CLOSE;
input int                 shift=0;
//--- 指標バッファ
double                    MABuffer[];
int                       ma_handle;
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 指標バッファマッピング
   if(AsSeries) ArraySetAsSeries(MABuffer,true);
   Print("Indicator buffer is timeseries = ",ArrayGetAsSeries(MABuffer));
   SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
   Print("Indicator buffer after SetIndexBuffer() is timeseries = ",
         ArrayGetAsSeries(MABuffer));

//--- 指標バッファの要素アクセスの順番を変更する
   ArraySetAsSeries(MABuffer,AsSeries);

   IndicatorSetString(INDICATOR_SHORTNAME,"MA("+period+")"+AsSeries);
//---
   ma_handle=iMA(Symbol(),0,period,shift,smootMode,price);
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- 移動平均値を MABuffer バッファに複製する
```

© 2000-2025, MetaQuotes Ltd.

```
   int copied=CopyBuffer(ma_handle,0,0,rates_total,MABuffer);

   Print("MABuffer[0] = ",MABuffer[0]);// AsSeries の値による
                                       // ものすごく古い値を受け取る
                                       // または現在の未完成のバー

//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

参照

カスタム指標プロパティ、時系列と指標へのアクセス

© 2000-2025, MetaQuotes Ltd.

IndicatorSetDouble

この関数は、対応する指標プロパティの値を設定します。指標プロパティは double 型でなければなりません。この関数には 2 つのバージョンがあります。

プロパティ識別子との呼び出し

```
bool  IndicatorSetDouble(
   int     prop_id,          // 識別子
   double  prop_value        // 設定する値
   );
```

プロパティ識別子と修飾子との呼び出し

```
bool  IndicatorSetDouble(
   int     prop_id,          // 識別子
   int     prop_modifier,    // 修飾子
   double  prop_value        // 設定する値
  )
```

パラメータ

prop_id

[in] 指標プロパティの識別子。値は ENUM_CUSTOMIND_PROPERTY_DOUBLE 列挙のいずれかです。

prop_modifier

[in] 指定されたプロパティの修飾子。修飾子はレベルプロパティのみで必要です。レベルの番号付けは0から始まります。なので、レベル2のプロパティを設定するには1を指定する必要があります（コンパイラディレクティブ使用時より 1 つ下）。

prop_value

[in] プロパティ値

戻り値

実行成功の場合は true、それ以外の場合は false

注意事項

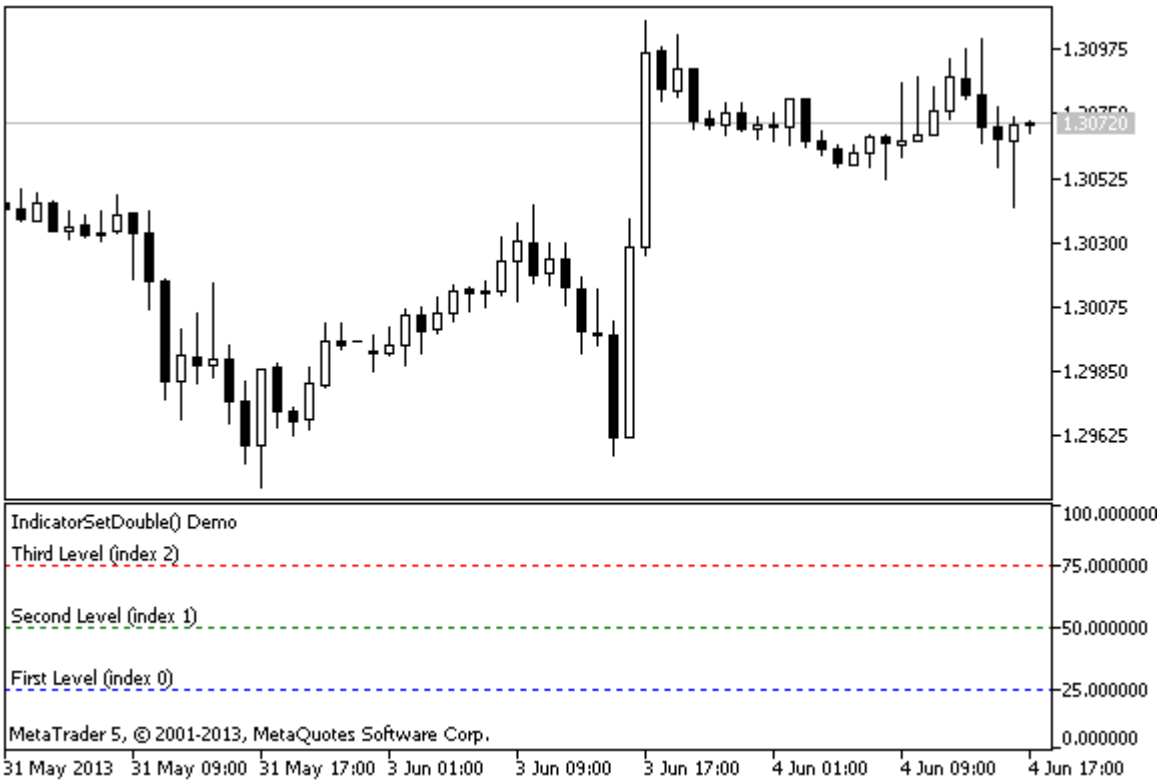
#property ディレクティブの使用時にはプロパティ（修飾子）の番号付けは1から始まる一方、関数では番号付けが0から始まります。レベル番号が正しく設定されていない場合、指標の表示目的のものと異なる場合があります。

例えば、別々のサブウィンドウでの指標の最初のレベル値は、次の 2 つの方法で設定出来ます。

- property indicator_level1 50 – レベル番号は1で指定
- IndicatorSetDouble(INDICATOR_LEVELVALUE, 0, 50) – レベル1を指定するのに 0 を使用

例： 指標ウィンドウの最大値と最小値及び水平線が配置されているレベル値を逆転する指標

© 2000–2025, MetaQuotes Ltd.




```
#property indicator_separate_window
//--- 指標ウィンドウの最大値と最小値を設定
#property indicator_minimum  0
#property indicator_maximum  100
//--- 3 つの水平レベルを別々の指標ウィンドウに表示
#property indicator_level1 25
#property indicator_level2 50
#property indicator_level3 75
//--- 水平レベルの厚さを設定
#property indicator_levelwidth 1
//--- 水平レベルのスタイルを設定
#property indicator_levelstyle STYLE_DOT
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 水平レベルの記述を設定
   IndicatorSetString(INDICATOR_LEVELTEXT,0,"First Level (index 0)");
   IndicatorSetString(INDICATOR_LEVELTEXT,1,"Second Level (index 1)");
   IndicatorSetString(INDICATOR_LEVELTEXT,2,"Third Level (index 2)");
//--- 指標の短縮名を設定
   IndicatorSetString(INDICATOR_SHORTNAME,"IndicatorSetDouble() Demo");
//--- それぞれのレベルの色を設定
   IndicatorSetInteger(INDICATOR_LEVELCOLOR,0,clrBlue);
   IndicatorSetInteger(INDICATOR_LEVELCOLOR,1,clrGreen);
   IndicatorSetInteger(INDICATOR_LEVELCOLOR,2,clrRed);
```

© 2000-2025, MetaQuotes Ltd.

```
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int tick_counter=0;
   static double level1=25,level2=50,level3=75;
   static double max=100,min=0, shift=100;
//--- ティックを計算
   tick_counter++;
//--- 10ティックごとにレベルを逆転する
   if(tick_counter%10==0)
     {
      //--- レベル値の符号を反転する
      level1=-level1;
      level2=-level2;
      level3=-level3;
      //--- 最大値と最小値の符号を反転する
      max-=shift;
      min-=shift;
      //--- シフト値を反転
      shift=-shift;
      //--- 新しいレベル値を設定
      IndicatorSetDouble(INDICATOR_LEVELVALUE,0,level1);
      IndicatorSetDouble(INDICATOR_LEVELVALUE,1,level2);
      IndicatorSetDouble(INDICATOR_LEVELVALUE,2,level3);
      //--- 指標ウィンドウの新しい最大値と最小値を設定
      Print("Set up max = ",max,",   min = ",min);
      IndicatorSetDouble(INDICATOR_MAXIMUM,max);
      IndicatorSetDouble(INDICATOR_MINIMUM,min);
     }
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
```

参照

© 2000-2025, MetaQuotes Ltd.

指標スタイルの例、指標プロパティと関数との関係、描画スタイル

© 2000–2025, MetaQuotes Ltd.

IndicatorSetInteger

この関数は、対応する指標プロパティの値を設定します。指標プロパティは int または 型でなければなりません。この関数には 2 つのバージョンがあります。

プロパティ識別子との呼び出し

```
bool  IndicatorSetInteger(
   int  prop_id,           // 識別子
   int  prop_value         // 設定する値
   );
```

プロパティ識別子と修飾子との呼び出し

```
bool  IndicatorSetInteger(
   int  prop_id,           // 識別子
   int  prop_modifier,     // 修飾子
   int  prop_value         // 設定する値
  )
```

パラメータ

prop_id

[in] 指標プロパティの識別子。値は ENUM_CUSTOMIND_PROPERTY_INTEGER 列挙のいずれかです。

prop_modifier

[in] 指定されたプロパティの修飾子。修飾子はレベルプロパティのみで必要です。

prop_value

[in] プロパティ値

戻り値

実行成功の場合は true、それ以外の場合は false

注意事項

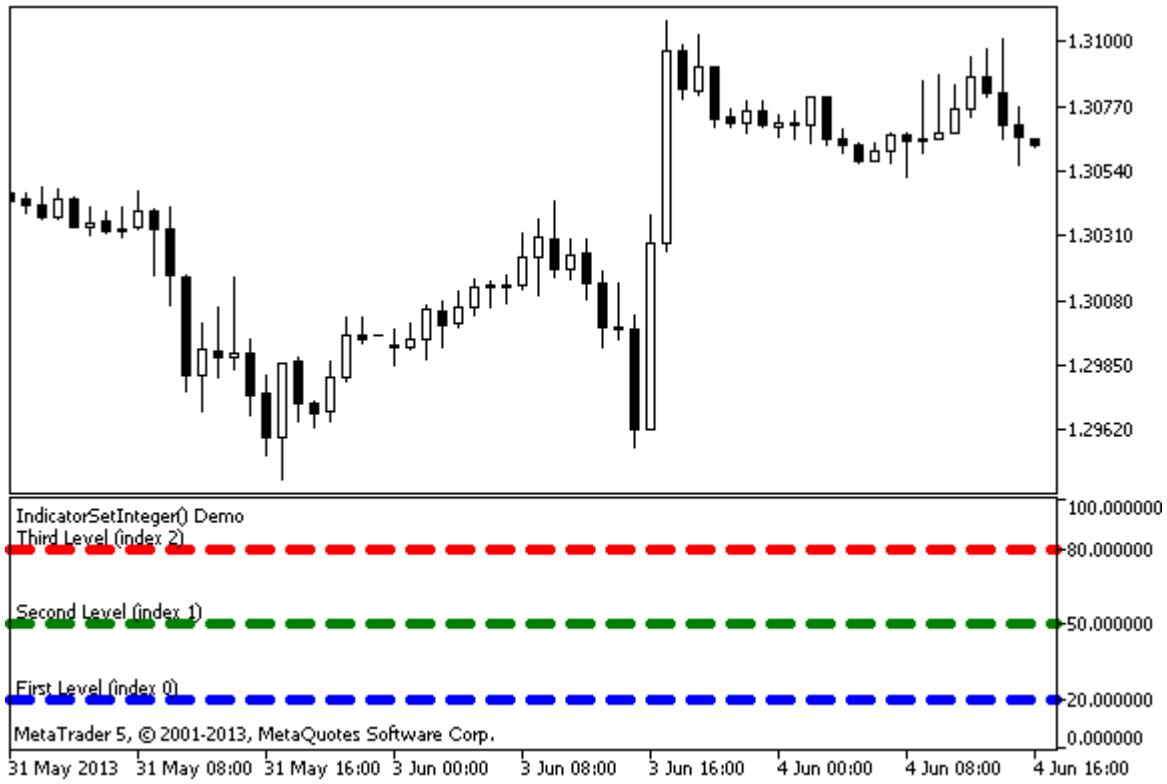
#property ディレクティブの使用時にはプロパティ（修飾子）の番号付けは1から始まる一方、関数では番号付けが0から始まります。レベル番号が正しく設定されていない場合、指標の表示目的のものと異なる場合があります。

例えば、最初の水平ラインの厚さを設定するにはゼロインデックスを使用します。

- IndicatorSetInteger(INDICATOR_LEVELWIDTH, 0, 5) – 最初のレベルの厚さを設定するにはインデックス 0 を使用する

例： 指標水平線の色、スタイルと太さを設定する指標

© 2000–2025, MetaQuotes Ltd.




```
#property indicator_separate_window
#property indicator_minimum 0
#property indicator_maximum 100
//--- 3 つの水平レベルを別々の指標ウィンドウに表示
#property indicator_level1 20
#property indicator_level2 50
#property indicator_level3 80
//--- 水平レベルの厚さを設定
#property indicator_levelwidth 5
//--- 水平レベルの色を設定
#property indicator_levelcolor clrAliceBlue
//--- 水平レベルのスタイルを設定
#property indicator_levelstyle STYLE_DOT
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 水平レベルの記述を設定
   IndicatorSetString(INDICATOR_LEVELTEXT,0,"First Level (index 0)");
   IndicatorSetString(INDICATOR_LEVELTEXT,1,"Second Level (index 1)");
   IndicatorSetString(INDICATOR_LEVELTEXT,2,"Third Level (index 2)");
//--- 指標の短縮名を設定
   IndicatorSetString(INDICATOR_SHORTNAME,"IndicatorSetInteger() Demo");
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| カスタム指標の反復関数                                                   |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int tick_counter=0;
//--- ティックを計算
   tick_counter++;
//--- また、ティックカウンタに応じて、水平レベルの色を計算
   ChangeLevelColor(0,tick_counter,3,6,10); // 最後の3つのパラメータは色を切り替える
   ChangeLevelColor(1,tick_counter,3,6,8);
   ChangeLevelColor(2,tick_counter,4,7,9);
//--- 水平レベルのスタイルを変更
   ChangeLevelStyle(0,tick_counter);
   ChangeLevelStyle(1,tick_counter+5);
   ChangeLevelStyle(2,tick_counter+15);
//--- 幅として5によるティック数の整数除算の余りを取得
   int width=tick_counter%5;
//--- 全ての水平レベルと反復処理して厚さを設定する
   for(int l=0;l<3;l++)
      IndicatorSetInteger(INDICATOR_LEVELWIDTH,l,width+1);
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| 別の指標ウィンドウにおける水平線の色を設定する                                   |
//+------------------------------------------------------------------+
void ChangeLevelColor(int level,       // 水平線の数
                      int tick_number,// 除算の剰余を取得する数
                      int f_trigger,  // 色切り替えの最初の除数
                      int s_trigger,  // 色切り替えの2 番目の除数
                      int t_trigger)  // 色切り替えの3 番目の除数
  {
   static color colors[3]={clrRed,clrBlue,clrGreen};
//--- colors[] 配列の色のインデックス
   int index=-1;
//--- 水平線を描画するのに colors[] 配列の色の数を計算
   if(tick_number%f_trigger==0)
      index=0;    // tick_number が f_trigger で割り切れる場合
   if(tick_number%s_trigger==0)
      index=1;    // tick_number が s_trigger で割り切れる場合
```

© 2000-2025, MetaQuotes Ltd.

```
   if(tick_number%t_trigger==0)
      index=2;    // tick_number が t_trigger で割り切れる場合
//--- 色が定義されたら設定する
   if(index!=-1)
      IndicatorSetInteger(INDICATOR_LEVELCOLOR,level,colors[index]);
//---
  }
//+------------------------------------------------------------------+
//| 別の指標ウィンドウにおける水平線のスタイルを設定する                |
//+------------------------------------------------------------------+
void ChangeLevelStyle(int level,      // 水平線の数
                      int tick_number// 除算の剰余を取得する数
                     )
  {
//--- スタイルを格納する配列
   static ENUM_LINE_STYLE styles[5]=
     {STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOTDOT};
//--- styles[] 配列のスタイルのインデックス
   int index=-1;
//--- styles[] 配列の数を計算して水平線のスタイルを設定
   if(tick_number%50==0)
      index=5;    // tick_number が 50 で割り切れる場合スタイルは STYLE_DASHDOTDOT
   if(tick_number%40==0)
      index=4;    // ... スタイルは STYLE_DASHDOT
   if(tick_number%30==0)
      index=3;    // ... STYLE_DOT
   if(tick_number%20==0)
      index=2;    // ... STYLE_DASH
   if(tick_number%10==0)
      index=1;    // ... STYLE_SOLID
//--- 定義されたスタイルを設定する
   if(index!=-1)
      IndicatorSetInteger(INDICATOR_LEVELSTYLE,level,styles[index]);
  }
```

参照

カスタム指標プロパティ、プログラムのプロパティ(#property)、描画スタイル

© 2000-2025, MetaQuotes Ltd.

IndicatorSetString

この関数は、対応する指標プロパティの値を設定します。指標プロパティは string 型でなければなりません。この関数には 2 つのバージョンがあります。

プロパティ識別子との呼び出し

```
bool  IndicatorSetString(
   int     prop_id,           // 識別子
   string  prop_value         // 設定する値
   );
```

プロパティ識別子と修飾子との呼び出し

```
bool  IndicatorSetString(
   int     prop_id,           // 識別子
   int     prop_modifier,     // 修飾子
   string  prop_value         // 設定する値
  )
```

パラメータ

prop_id

[in] 指標プロパティの識別子。値は ENUM_CUSTOMIND_PROPERTY_STRING 列挙のいずれかです。

prop_modifier

[in] 指定されたプロパティの修飾子。修飾子はレベルプロパティのみで必要です。

prop_value

[in] プロパティ値

戻り値

実行成功の場合は true、それ以外の場合は false

注意事項

#property ディレクティブの使用時にはプロパティ（修飾子）の番号付けは1から始まる一方、関数では番号付けが0から始まります。レベル番号が正しく設定されていない場合、指標の表示目的のものと異なる場合があります。

例えば、最初の水平ラインの記述を設定するにはゼロインデックスを使用します。

- IndicatorSetString(INDICATOR_LEVELTEXT, 0, "First Level") – 最初のレベルの記述を設定するにはインデックス 0 を使用する

例：指標水平線にラベルを設定する指標

© 2000–2025, MetaQuotes Ltd.




```
#property indicator_separate_window
#property indicator_minimum 0
#property indicator_maximum 100
//--- 3 つの水平レベルを別々の指標ウィンドウに表示
#property indicator_level1 30
#property indicator_level2 50
#property indicator_level3 70
//--- 水平レベルの色を設定
#property indicator_levelcolor clrRed
//--- 水平レベルのスタイルを設定
#property indicator_levelstyle STYLE_SOLID
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 水平レベルの記述を設定
   IndicatorSetString(INDICATOR_LEVELTEXT,0,"First Level (index 0)");
   IndicatorSetString(INDICATOR_LEVELTEXT,1,"Second Level (index 1)");
   IndicatorSetString(INDICATOR_LEVELTEXT,2,"Third Level (index 2)");
//--- 指標の短縮名を設定
   IndicatorSetString(INDICATOR_SHORTNAME,"IndicatorSetString() Demo");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
```

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---

//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
```

参照

カスタム指標プロパティ、プログラムのプロパティ（#property）

© 2000–2025, MetaQuotes Ltd.

PlotIndexSetDouble

この関数は、対応する指示線の対応するプロパティの値を設定します。指標プロパティは double 型でなければなりません。

```
bool  PlotIndexSetDouble(
   int     plot_index,     // プロットスタイルインデックス
   int     prop_id,        // プロパティ識別子
   double  prop_value      // 設定する値
   );
```

パラメータ

plot_index

[in] グラフィックプロットのインデックス

prop_id

[in] 値は ENUM_PLOT_PROPERTY_DOUBLE 列挙のいずれかの値

prop_value

[in] プロパティ値

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

PlotIndexSetInteger

この関数は、対応する指示線の対応するプロパティの値を設定します。指標プロパティは int、char、bool または color 型でなければなりません。この関数には 2 つのバージョンがあります。

プロパティの識別子を示す呼び出し

```
bool  PlotIndexSetInteger(
   int  plot_index,         // プロットスタイルインデックス
   int  prop_id,            // プロパティ識別子
   int  prop_value          // 設定する値
   );
```

プロパティの識別子と修飾子を示す呼び出し

```
bool  PlotIndexSetInteger(
   int  plot_index,         // プロットスタイルインデックス
   int  prop_id,            // プロパティ識別子
   int  prop_modifier,      // プロパティ修飾子
   int  prop_value          // 設定する値
  )
```

パラメータ

plot_index

[in] グラフィックプロットのインデックス

prop_id

[in] 値は ENUM_PLOT_PROPERTY_INTEGER 列挙のいずれかです。

prop_modifier

[in] 指定されたプロパティの修飾子。修飾子は色インデックスプロパティのみで必要です。

prop_value

[in] プロパティ値

戻り値

成功の場合は true、それ以外の場合は false

例：3 色の線を描画する指標色は 5 ティックごとに変わります。

© 2000–2025, MetaQuotes Ltd.




```
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//---- ColorLine をプロットする
#property indicator_label1  "ColorLine"
#property indicator_type1   DRAW_COLOR_LINE
#property indicator_color1  clrRed,clrGreen,clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  3
//--- 指標バッファ
double          ColorLineBuffer[];
double          ColorBuffer[];
int             MA_handle;
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- 指標バッファマッピング
   SetIndexBuffer(0,ColorLineBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorBuffer,INDICATOR_COLOR_INDEX);
//--- MA ハンドルの取得
   MA_handle=iMA(Symbol(),0,10,0,MODE_EMA,PRICE_CLOSE);
//---
  }
//+------------------------------------------------------------------+
//|  色インデックスを取得                                            |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
int getIndexOfColor(int i)
  {
   int j=i%300;
   if(j<100) return(0);// 1 番目のインデックス
   if(j<200) return(1);// 2 番目のインデックス
   return(2); // 3 番目のインデックス
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---
   static int ticks=0,modified=0;
   int limit;
//--- 初めての計算、またはバーの数が変わった
   if(prev_calculated==0)
     {
      //--- MA の値を ColorLineBuffer 指標バッファに複製する
      int copied=CopyBuffer(MA_handle,0,0,rates_total,ColorLineBuffer);
      if(copied<=0) return(0);// 複製に失敗。廃棄する。
      //--- バー全部で線の色を設定
      for(int i=0;i<rates_total;i++)
         ColorBuffer[i]=getIndexOfColor(i);
     }
   else
     {
      //--- MA の値を ColorLineBuffer 指標バッファに複製する
      int copied=CopyBuffer(MA_handle,0,0,rates_total,ColorLineBuffer);
      if(copied<=0) return(0);

      ticks++;// ティックを数える
      if(ticks>=5)//基本配色を変更する時
        {
         ticks=0; // カウンタをリセット
         modified++; // 色変換のカウンタ
         if(modified>=3)modified=0;// カウンタをリセット
         ResetLastError();
         switch(modified)
           {
```

© 2000-2025, MetaQuotes Ltd.

```
            case 0:// 1 番目の基本配色
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrRed);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrBlue);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrGreen);
               Print("Color scheme "+modified);
               break;
            case 1:// 2 番目の基本配色
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrYellow);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrPink);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrLightSlateGray);
               Print("Color scheme "+modified);
               break;
            default:// 3 番目の基本配色
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrLightGoldenrod);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrOrchid);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrLimeGreen);
               Print("Color scheme "+modified);
           }
        }
      else
        {
         //--- 開始位置を設定
         limit=prev_calculated-1;
         //--- バー全部で線の色を設定
         for(int i=limit;i<rates_total;i++)
            ColorBuffer[i]=getIndexOfColor(i);
        }
     }
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

PlotIndexSetString

この関数は、対応する指示線の対応するプロパティの値を設定します。指標プロパティは string 型でなければなりません。

```
bool  PlotIndexSetString(
   int      plot_index,      // プロットスタイルインデックス
   int      prop_id,         // プロパティ識別子
   string   prop_value       // 設定する値
   );
```

パラメータ

plot_index

[in] グラフィックプロットのインデックス

prop_id

[in] 値は ENUM_PLOT_PROPERTY_STRING 列挙のいずれかの値

prop_value

[in] プロパティ値

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

PlotIndexGetInteger

この関数は、対応する指示線の対応するプロパティの値を設定します。指標プロパティは int、color、bool または char 型でなければなりません。この関数には 2 つのバージョンがあります。

プロパティの識別子を示す呼び出し

```
int  PlotIndexGetInteger(
   int  plot_index,          // プロットスタイルインデックス
   int  prop_id,             // プロパティ識別子
   );
```

プロパティの識別子と修飾子を示す呼び出し

```
int  PlotIndexGetInteger(
   int  plot_index,          // プロットインデックス
   int  prop_id,             // プロパティ識別子
   int  prop_modifier        // プロパティ修飾子
  )
```

パラメータ

plot_index

[in] グラフィックプロットのインデックス

prop_id

[in] 値は ENUM_PLOT_PROPERTY_INTEGER 列挙のいずれかです。

prop_modifier

[in] 指定されたプロパティの修飾子。修飾子は色インデックスプロパティのみで必要です。

注意事項

この関数は、適切な指標線の描画の設定を抽出するように設計されています。この関数は線の描画プロパティを別の線に複製する PlotIndexSetInteger 関数と連結して機能します。

例：曜日に応じてローソク足を塗る指標。曜日ごとの色はプログラムで設定されます。

© 2000–2025, MetaQuotes Ltd.




```
#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots   1
//---- Col or Candl es をプロットする
#property indicator_label1  "ColorCandles"
#property indicator_type1   DRAW_COLOR_CANDLES
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 指標バッファ
double          OpenBuffer[];
double          HighBuffer[];
double          LowBuffer[];
double          CloseBuffer[];
double          ColorCandlesColors[];
color           ColorOfDay[6]={CLR_NONE,clrMediumSlateBlue,
                               clrDarkGoldenrod,clrForestGreen,clrBlueViolet,clrRed};
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- 指標バッファマッピング
   SetIndexBuffer(0,OpenBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,HighBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,LowBuffer,INDICATOR_DATA);
   SetIndexBuffer(3,CloseBuffer,INDICATOR_DATA);
   SetIndexBuffer(4,ColorCandlesColors,INDICATOR_COLOR_INDEX);
//--- 色バッファの色の数を設定
```

© 2000–2025, MetaQuotes Ltd.

```
   PlotIndexSetInteger(0,PLOT_COLOR_INDEXES,6);
//--- 色バッファの色を設定
   for(int i=1;i<6;i++)
      PlotIndexSetInteger(0,PLOT_LINE_COLOR,i,ColorOfDay[i]);
//--- 精度を設定
   IndicatorSetInteger(INDICATOR_DIGITS,_Digits);
   printf("We have %u colors of days",PlotIndexGetInteger(0,PLOT_COLOR_INDEXES));
//---
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---
   int i;
   MqlDateTime t;
//----
   if(prev_calculated==0) i=0;
   else i=prev_calculated-1;
//----
   while(i<rates_total)
     {
      OpenBuffer[i]=open[i];
      HighBuffer[i]=high[i];
      LowBuffer[i]=low[i];
      CloseBuffer[i]=close[i];
      //--- それぞれのローソク足の色を設定
      TimeToStruct(time[i],t);
      ColorCandlesColors[i]=t.day_of_week;
      //---
      i++;
     }
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

オブジェクト関数

これは、任意の指定されたチャートに関連するグラフィックオブジェクトを操作するために意図された関数群です。

グラフィックオブジェクトのプロパティを定義する関数、またチャートでオブジェクトを作成して移動する ObjectCreate() 及び ObjectMove() は実際にチャートにコマンドを送るのに使用されます。これらの関数が正常に実行された場合、コマンドがチャートイベントの共用キューに追加されます。グラフィックオブジェクトのプロパティの視覚変更は、チャートイベントのキューを処理する際に実装されています。

従って、これらの関数を呼び出した後、チャートの視覚更新が即時行われると期待してはいけません。一般的には、チャートは、新たな相場の到着やチャートウィンドウのサイズ変更などの変更イベントの後に、端末によって自動更新されます。強制的にグラフィックオブジェクトを更新するには ChartRedraw() 関数を使用します。

関数	アクション
ObjectCreate	指定されたチャートで指定された型のオブジェクトを作成します。
ObjectName	指定されたチャートまたはチャートサブウィンドウの指定された名称のオブジェクトを返します。
ObjectDelete	指定された名称のオブジェクトを指定されたチャートまたはチャートサブウィンドウから削除します。
ObjectsDeleteAll	指定された型のオブジェクトを全て指定されたチャートまたはチャートサブウィンドウから削除します。
ObjectFind	名称に指定された ID を持つオブジェクトを探します。
ObjectGetTimeByValue	指定されたオブジェクトの価格値の時間値を返します。
ObjectGetValueByTime	指定された時刻でのオブジェクトの価格値を返します。
ObjectMove	指定されたオブジェクトのアンカーポイントの座標を変更します。
ObjectsTotal	指定されたチャートまたはチャートサブウィンドウのオブジェクトの数を返します。
ObjectGetDouble	オブジェクトプロパティの double 値を返します。
ObjectGetInteger	オブジェクトプロパティの整数値を返します。
ObjectGetString	オブジェクトプロパティの string 値を返します。
ObjectSetDouble	オブジェクトプロパティの値を設定します。
ObjectSetInteger	オブジェクトプロパティの値を設定します。
ObjectSetString	オブジェクトプロパティの値を設定します。
TextSetFont	描画メソッドを使用してテキストを表示するためのフォントを設定します。デフォルトとしてArial20 が使用されます。
TextOut	テキストをグラフィックリソース作成に意図されたカスタム配列（バッファ）に転送する。
TextGetSize	現在のフォント設定での文字列の幅と高さを返します。

© 2000–2025, MetaQuotes Ltd.

全てのグラフィカルオブジェクトは、サブウィンドウを含んだチャート内では，一意の名称を持つ必要があります．グラフィックオブジェクトの名称の変更は、古い名称のオブジェクトの削除イベントと新しい名称のオブジェクトの作成イベントとの 2 つのイベントを生成します。

オブジェクト作成後またはオブジェクトプロパティの変更後には、強制的にチャートとチャート内の全ての可視なオブジェクトを描画するようクライアント端末に指示するChartRedraw() 関数を呼ぶことが推薦されています。

© 2000–2025, MetaQuotes Ltd.

ObjectCreate

この関数は、指定された名称、型、及び指定されたチャートサブウィンドウでの初期座標を使用してオブジェクトを作成します。作成時には 30 までの座標を指定することが出来ます。

```
bool  ObjectCreate(
   long          chart_id,       // チャート識別子
   string        name,           // オブジェクト名
   ENUM_OBJECT   type,           // オブジェクトの型
   sub_window    nwin,           // ウィンドウインデックス
   datetime      time1,          // 1 番目のアンカーポイントの時間
   double        price1,         // 1 番目のアンカーポイントの価格
   ...
   datetime      timeN=0,        // N 番目のアンカーポイントの時間
   double        priceN=0,       // N 番目のアンカーポイントの価格
   ...
   datetime      time30=0,       // 30番目のアンカーポイントの時間
   double        price30=0       // 30番目のアンカーポイントの価格
   );
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] オブジェクト名名称は、サブウィンドウを含めたチャート内で一意でなければなりません。

型

[in] オブジェクトの型値は ENUM_OBJECT 列挙のいずれかです。

sub_window

[in] チャートサブウィンドウの番号。（0 はメインチャートウィンドウ）指定されたサブウィンドウは存在しなければなりません。さもないと、この関数は false を返します。

time1

[in] 1 番目のアンカーの時間座標

price1

[in] 1 番目のアンカーの価格座標

timeN=0

[in] N 番目のアンカーの時間座標

priceN=0

[in] N 番目のアンカーの価格座標

time30=0

[in] 30番目のアンカーの時間座標

price30=0

[in] 30番目のアンカーの価格座標

© 2000–2025, MetaQuotes Ltd.

戻り値

この関数は、指定されたチャートのキューにコマンドが正常に追加された場合はtrueを返し、それ以外の場合はfalseを返します。オブジェクトが既に作成されている場合は、その座標の変更を試みます。

注意事項

ObjectCreate()では常に非同期呼び出しが使用されます。これが、この関数がチャートキューにコマンドを追加した結果だけを返す理由です。この場合、trueはコマンドが正常にエンキューされたことを意味しますのみで、その実行結果は不明です。

コマンドの実行結果を確認するにはObjectFind()関数またはオブジェクトプロパティをレクエストするObjectGetXXXなどのその他の関数を使用できます。ただし、そのような関数はそのチャートのキューの最後に追加され、(同期呼び出しのために)実行結果を待つため、時間がかかる可能性があります。チャート上で多数のオブジェクトを操作する場合は、この特徴を考慮する必要があります。

オブジェクト名は63字を超えられません。

チャートに指標を含むサブウィンドウが存在する場合、チャートサブウィンドウの番号付けは1から始まります。チャートのメインウィンドウは常に0インデックスを持ちます。

大多数(最大30)のアンカーポイントは、将来の使用のために実装されています。同時に、グラフィックオブジェクトのためのアンカーポイントの30での上限は、関数を呼び出す時に使用出来るパラメータ数の制限(64以下)によって決定されています。

オブジェクトの名称が変更された場合、2つのイベントが同時に形成されます。これらのイベントはエキスパートアドバイザーまたは指標のOnChartEvent()関数によって処理されます。

- 古い名称のオブジェクトの削除イベント
- 新しい名称のオブジェクトの作成イベント

それぞれのオブジェクトの型の作成時には、特定の数のアンカーポイントのを指定する必要があります。

ID	説明	アンカーポイント
OBJ_VLINE	垂直線。	1つのアンカーポイント実際には時間座標のみが使用されます。
OBJ_HLINE	水平線。	1つのアンカーポイント実際には価格座標のみが使用されます。
OBJ_TREND	傾向線。	2つのアンカーポイント
OBJ_TRENDBYANGLE	角度による傾向線。	2つのアンカーポイント
OBJ_CYCLES	サイクルライン。	2つのアンカーポイント
OBJ_ARROWED_LINE	矢印。	2つのアンカーポイント
OBJ_CHANNEL	等距離チャンネル。	3つのアンカーポイント
OBJ_STDDEVCHANNEL	標準偏差チャンネル。	2つのアンカーポイント
OBJ_REGRESSION	線形回帰チャンネル。	2つのアンカーポイント
OBJ_PITCHFORK	アンドリューズピッチフォーク。	3つのアンカーポイント

© 2000-2025, MetaQuotes Ltd.

ID	説明	アンカーポイント
OBJ_GANNLINE	ギャンライン。	2 つのアンカーポイント
OBJ_GANNFAN	ギャンファン（扇）。	2 つのアンカーポイント
OBJ_GANNGRID	ギャングリッド。	2 つのアンカーポイント
OBJ_FIBO	フィボナッチリトレースメント。	2 つのアンカーポイント
OBJ_FIBOTIMES	フィボナッチタイムゾーン。	2 つのアンカーポイント
OBJ_FIBOFAN	フィボナッチファン（扇）。	2 つのアンカーポイント
OBJ_FIBOARC	フィボナッチアーク（円弧）。	2 つのアンカーポイント
OBJ_FIBOCHANNEL	フィボナッチチャンネル。	3 つのアンカーポイント
OBJ_EXPANSION	フィボナッチ拡張。	3 つのアンカーポイント
OBJ_ELLIOTWAVE5	エリオット動機波。	5 つのアンカーポイント
OBJ_ELLIOTWAVE3	エリオット修正波。	3 つのアンカーポイント
OBJ_RECTANGLE	長方形。	2 つのアンカーポイント
OBJ_TRIANGLE	三角形。	3 つのアンカーポイント
OBJ_ELLIPSE	楕円形。	3 つのアンカーポイント
OBJ_ARROW_THUMB_UP	サムズアップ。	1 つのアンカーポイント
OBJ_ARROW_THUMB_DOWN	サムズダウン。	1 つのアンカーポイント
OBJ_ARROW_UP	上矢印。	1 つのアンカーポイント
OBJ_ARROW_DOWN	下矢印。	1 つのアンカーポイント
OBJ_ARROW_STOP	ストップサイン。	1 つのアンカーポイント
OBJ_ARROW_CHECK	チェックサイン。	1 つのアンカーポイント
OBJ_ARROW_LEFT_PRICE	左プライスラベル。	1 つのアンカーポイント
OBJ_ARROW_RIGHT_PRICE	右プライスラベル。	1 つのアンカーポイント
OBJ_ARROW_BUY	買いサイン。	1 つのアンカーポイント
OBJ_ARROW_SELL	売りサイン。	1 つのアンカーポイント
OBJ_ARROW	矢印。	1 つのアンカーポイント
OBJ_TEXT	テキスト。	1 つのアンカーポイント
OBJ_LABEL	ラベル。	位置は OBJPROP_XDISTANCE 及び OBJPROP_YDISTANCE プロパティで設定されます。
OBJ_BUTTON	ボタン。	位置は OBJPROP_XDISTANCE 及び OBJPROP_YDISTANCE プロパティで設定されます。

© 2000–2025, MetaQuotes Ltd.

ID	説明	アンカーポイント
OBJ_CHART	チャート。	位置は OBJPROP_XDISTANCE 及び OBJPROP_YDISTANCE プロパティで設定されます。
OBJ_BITMAP	ビットマップ。	1 つのアンカーポイント
OBJ_BITMAP_LABEL	ビットマップラベル。	位置は OBJPROP_XDISTANCE 及び OBJPROP_YDISTANCE プロパティで設定されます。
OBJ_EDIT	編集。	位置は OBJPROP_XDISTANCE 及び OBJPROP_YDISTANCE プロパティで設定されます。
OBJ_EVENT	経済カレンダーのイベントに対応する「イベント」オブジェクト。	1 つのアンカーポイント 実際には時間座標のみが使用されます。
OBJ_RECTANGLE_LABEL	カスタムグラフィカルインタフェースを作成及び設計するための「四角形ラベル」オブジェクト。	位置は OBJPROP_XDISTANCE 及び OBJPROP_YDISTANCE プロパティで設定されます。

© 2000–2025, MetaQuotes Ltd.

ObjectName

この関数は、指定されたチャートの指定されたサブウィンドウで、指定された型を持つオブジェクトの名称を返します。

```
string  ObjectName(
   long  chart_id,            // チャート識別子
   int   pos,                 // オブジェクトリスト内での番号
   int   sub_window=-1,       // ウィンドウ番号
   int   type=-1              // オブジェクトの型
   );
```

パラメータ

chart_id

[in] チャート識別子（0は現在のチャート）

pos

[in] サブウィンドウの数と種類によって指定されたフィルタに応じたオブジェクトの序数

sub_window=-1

[in] チャートサブウィンドウの番号。（0はメインウィンドウ、-1はメインウィンドウを含む全てのサブウィンドウ）

type=-1

[in] オブジェクトの型。値はENUM_OBJECT列挙のいずれかです。-1は全ての型を意味します。

戻り値

成功の場合はオブジェクト名

注意事項

この関数は同期呼び出しを使用します。つまり、この関数が呼び出される前にこのチャートのエンキューされたすべてのコマンドの実行が待機されるため、この関数は時間がかかることがあります。チャート上で多数のオブジェクトを操作する場合は、この特徴を考慮する必要があります。

オブジェクトの名称が変更された場合、2つのイベントが同時に形成されます。これらのイベントはエキスパートアドバイザーまたは指標のOnChartEvent()関数によって処理されます。

- 古い名称のオブジェクトの削除イベント
- 新しい名称のオブジェクトの作成イベント

© 2000-2025, MetaQuotes Ltd.

ObjectDelete

この関数は指定された名称のオブジェクトを指定されたチャートから削除します。

```
bool  ObjectDelete(
   long     chart_id,      // チャート識別子
   string   name           // オブジェクト名
   );
```

パラメータ

chart_id

[in] チャート識別子（0は現在のチャート）

name

[in] 削除されるオブジェクト名

戻り値

この関数は、指定されたチャートのキューにコマンドが正常に追加された場合はtrueを返し、それ以外の場合はfalseを返します。

注意事項

ObjectDelete()では常に非同期呼び出しが使用されます。これが、この関数がチャートキューにコマンドを追加した結果だけを返す理由です。この場合、trueはコマンドが正常にエンキューされたことを意味しますのみで、その実行結果は不明です。

コマンドの実行結果を確認するにはObjectFind()関数またはオブジェクトプロパティをレクエストするObjectGetXXXなどのその他の関数を使用できます。ただし、そのような関数はそのチャートのキューの最後に追加され、（同期呼び出しのために）実行結果を待つため、時間がかかる可能性があります。チャート上で多数のオブジェクトを操作する場合は、この特徴を考慮する必要があります。

オブジェクトの名称が変更された場合、2つのイベントが同時に形成されます。これらのイベントはエキスパートアドバイザーまたは指標のOnChartEvent()関数によって処理されます。

- 古い名称のオブジェクトの削除イベント
- 新しい名称のオブジェクトの作成イベント

© 2000–2025, MetaQuotes Ltd.

ObjectsDeleteAll

指定された型のオブジェクトを全て指定されたチャートまたはチャートサブウィンドウから削除します。

```
int  ObjectsDeleteAll(
   long  chart_id,            // チャート識別子
   int   sub_window=-1,       // ウィンドウ番号
   int   type=-1              // オブジェクトの型
   );
```

オブジェクト名のプレフィックスを使用して、指定された型のオブジェクトを全て削除します。

```
int  ObjectsDeleteAll(
   long           chart_id,   // チャート識別子
   const string     prefix,   // オブジェクト名のプレフィックス
   int        sub_window=-1,  // ウィンドウ番号
   int       object_type=-1   // オブジェクトの型
   );
```

パラメータ

chart_id

[in] チャート識別子（0は現在のチャート）

prefix

[in] オブジェクト名のプレフィックス。名称がこの文字のセットから始まる全てのオブジェクトがチャートから削除されます。プレフィックスは 'name' または 'name*' として指定でき、両バージョンの機能は同じです。空の文字列がプレフィックスとして指定された場合、全ての名称のオブジェクトが削除されます。

sub_window=-1

[in] チャートサブウィンドウの番号（0はメインウィンドウ、-1はメインウィンドウを含む全てのサブウィンドウ）

type=-1

[in] オブジェクトの型。値は ENUM_OBJECT 列挙のいずれかです。-1は全ての型を意味します。

戻り値

削除されたオブジェクトの数。エラーの詳細を読むには GetLastError() を呼びます。

注意事項

この関数は同期呼び出しを使用します。つまり、この関数が呼び出される前にこのチャートのエンキューされたすべてのコマンドの実行が待機されるため、この関数は時間がかかることがあります。チャート上で多数のオブジェクトを操作する場合は、この特徴を考慮する必要があります。

© 2000-2025, MetaQuotes Ltd.

ObjectFind

この関数は指定された ID を持つチャートで指定された名称のオブジェクトを探します。

```
int  ObjectFind(
   long     chart_id,      // チャート識別子
   string   name           // オブジェクト名
   );
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] 探されるオブジェクトの名称

戻り値

成功した場合、オブジェクトが見つかったサブウィンドウの番号（0 はメインウィンドウ）。オブジェクトが見つからなかった場合、負の数。エラーの詳細を読むには GetLastError() を呼びます。

注意事項

この関数は同期呼び出しを使用します。つまり、この関数が呼び出される前にこのチャートのエンキューされたすべてのコマンドの実行が待機されるため、この関数は時間がかかることがあります。チャート上で多数のオブジェクトを操作する場合は、この特徴を考慮する必要があります。

オブジェクトの名称が変更された場合、2 つのイベントが同時に形成されます。これらのイベントはエキスパートアドバイザーまたは指標の OnChartEvent() 関数によって処理されます。

- 古い名称のオブジェクトの削除イベント
- 新しい名称のオブジェクトの作成イベント

© 2000–2025, MetaQuotes Ltd.

ObjectGetTimeByValue

この関数は、指定されたオブジェクトの指定された価格値に対応する時間値を返します。

```
datetime  ObjectGetTimeByValue(
   long     chart_id,      // チャート識別子
   string   name,          // オブジェクト名
   double   value,         // 価格
   int      line_id        // 行番号
   );
```

パラメータ

chart_id

[in] チャート識別子（0は現在のチャート）

name

[in] オブジェクト名

value

[in] 価格値

line_id

[in] ラインID

戻り値

指定されたオブジェクトの指定された価格値に対応する時間値

注意事項

この関数は同期呼び出しを使用します。つまり、この関数が呼び出される前にこのチャートのエンキューされたすべてのコマンドの実行が待機されるため、この関数は時間がかかることがあります。チャート上で多数のオブジェクトを操作する場合は、この特徴を考慮する必要があります。

オブジェクト1つの価格座標で複数の値を持つことが出来るため、行番号を指定する必要があります。この関数は、次のオブジェクトにのみ適用されます。

- トレンドライン（OBJ_TREND）
- アングルトレンド（OBJ_TRENDBYANGLE）
- ギャンライン（OBJ_GANNLINE）
- 等距離チャンネル（OBJ_CHANNEL）－2ライン
- 線形回帰チャンネル（OBJ_REGRESSION）－3ライン
- 標準偏差チャンネル（OBJ_STDDEVCHANNEL）－3ライン
- 矢印線（OBJ_ARROWED_LINE）

参照

オブジェクト型

© 2000-2025, MetaQuotes Ltd.

ObjectGetValueByTime

この関数は、指定されたオブジェクトの指定された時間値に対応する価格値を返します。

```
double  ObjectGetValueByTime(
   long      chart_id,     // チャート識別子
   string    name,         // オブジェクト名
   datetime  time,         // 時間
   int       line_id       // 行番号
   );
```

パラメータ

chart_id

[in] チャート識別子（0は現在のチャート）

name

[in] オブジェクト名

time

[in] 時刻

line_id

[in] 線の識別子

戻り値

指定されたオブジェクトの指定された時間値に対応する価格値

注意事項

この関数は同期呼び出しを使用します。つまり、この関数が呼び出される前にこのチャートのエンキューされたすべてのコマンドの実行が待機されるため、この関数は時間がかかることがあります。チャート上で多数のオブジェクトを操作する場合は、この特徴を考慮する必要があります。

オブジェクト1つの価格座標で複数の値を持つことが出来るため、行番号を指定する必要があります。この関数は、次のオブジェクトにのみ適用されます。

- トレンドライン（OBJ_TREND）
- アングルトレンド（OBJ_TRENDBYANGLE）
- ギャンライン（OBJ_GANNLINE）
- 等距離チャンネル（OBJ_CHANNEL）－2ライン
- 線形回帰チャンネル（OBJ_REGRESSION）－3ライン
- 標準偏差チャンネル（OBJ_STDDEVCHANNEL）－3ライン
- 矢印線（OBJ_ARROWED_LINE）

参照

オブジェクト型

© 2000–2025, MetaQuotes Ltd.

ObjectMove

この関数は、指定されたアンカーポイントまたはオブジェクトの座標を変更します。

```
bool  ObjectMove(
   long      chart_id,         // チャート識別子
   string    name,             // オブジェクト名
   int       point_index,      // アンカーポイント番号
   datetime  time,             // 時間
   double    price             // 価格
   );
```

パラメータ

chart_id

[in] チャート識別子（0は現在のチャート）

name

[in] オブジェクト名

point_index

[in] アンカーポイントのインデックス。アンカーポイントの数はオブジェクト型によって違います。

time

[in] 選択されたアンカーポイントの時間座標

価格

[in] 選択されたアンカーポイントの価格座標

戻り値

この関数は、指定されたチャートのキューにコマンドが正常に追加された場合はtrueを返し、それ以外の場合はfalseを返します。

注意事項

ObjectMove()では常に非同期呼び出しが使用されます。これが、この関数がチャートキューにコマンドを追加した結果だけを返す理由です。この場合、trueはコマンドが正常にエンキューされたことを意味しますのみで、その実行結果は不明です。

コマンドの実行結果を確認するにはオブジェクトプロパティをレクエストするObjectGetXXXなどのその他の関数を使用できます。ただし、そのような関数はそのチャートのキューの最後に追加され、（同期呼び出しのために）実行結果を待つため、時間がかかる可能性があります。チャート上で多数のオブジェクトを操作する場合は、この特徴を考慮する必要があります。

© 2000–2025, MetaQuotes Ltd.

ObjectsTotal

この関数は、指定されたチャートの指定されたサブウィンドウで、指定された型を持つオブジェクトの数を返します。

```
int  ObjectsTotal(
   long  chart_id,            // チャート識別子
   int   sub_window=-1,       // ウィンドウ番号
   int   type=-1              // オブジェクトの型
   );
```

パラメータ

chart_id

[in] チャート識別子（0は現在のチャート）

sub_window=-1

[in] チャートサブウィンドウの番号。（0はメインウィンドウ、-1はメインウィンドウを含む全てのサブウィンドウ）

type=-1

[in] オブジェクトの型。値はENUM_OBJECT列挙のいずれかです。-1は全ての型を意味します。

戻り値

オブジェクトの数。

注意事項

この関数は同期呼び出しを使用します。つまり、この関数が呼び出される前にこのチャートのエンキューされたすべてのコマンドの実行が待機されるため、この関数は時間がかかることがあります。チャート上で多数のオブジェクトを操作する場合は、この特徴を考慮する必要があります。

© 2000-2025, MetaQuotes Ltd.

ObjectSetDouble

この関数は、対応するオブジェクトプロパティの値を設定します。オブジェクトプロパティは double 型でなければなりません。この関数には 2 つのバージョンがあります。

修飾語なしでのプロパティ値の設定

```
bool  ObjectSetDouble(
   long                          chart_id,          // チャート識別子
   string                        name,              // オブジェクト名
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // プロパティ
   double                        prop_value         // 値
   );
```

修飾子を使用してのプロパティ値の設定

```
bool  ObjectSetDouble(
   long                          chart_id,          // チャート識別子
   string                        name,              // オブジェクト名
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // プロパティ
   int                           prop_modifier,     // 修飾子
   double                        prop_value         // 値
   );
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] オブジェクト名

prop_id

[in] オブジェクトプロパティ識別子。値は ENUM_OBJECT_PROPERTY_DOUBLE 列挙のいずれかです。

prop_modifier

[in] 指定されたプロパティの修飾子。修飾子はフィボナッチツールのレベル数及びグラフィックオブジェクトのアンドリューズピッチフォークを示します。レベルの番号付けは0から始まります。

prop_value

[in] プロパティ値

戻り値

グラフィックオブジェクトのプロパティ変更コマンドが正常にチャートに送信された場合にのみ true、その他の場合は false。エラーの詳細を読むには GetLastError() を呼びます。

注意事項

この関数は非同期呼び出しを使用します。つまり、この関数は指定されたチャートのキューに追加されたコマンドの実行を待機せずに、代わりに、すぐに制御を返します。

© 2000–2025, MetaQuotes Ltd.

コマンドの実行結果を確認するには指定されたオブジェクトプロパティをレクエストする関数を使用できますが、そのような関数はそのチャートのキューの最後に追加され、実行結果を待つため時間がかかることにご注意ください。チャート上で多数のオブジェクトを操作する場合は、この特徴を考慮する必要があります。

フィボナッチオブジェクトを作成し新しいレベルを追加する例

```
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 補助配列
   double high[],low[],price1,price2;
   datetime time[],time1,time2;
//--- 始値のコピー - 最新の100バーで充分
   int copied=CopyHigh(Symbol(),0,0,100,high);
   if(copied<=0)
     {
      Print("Failed to copy the values of the High price series");
      return;
     }
//--- 終値のコピー - 最新の100バーで充分
   copied=CopyLow(Symbol(),0,0,100,low);
   if(copied<=0)
     {
      Print("Failed to copy the values of the Low price series");
      return;
     }
//--- 最後の100バーの始値を複製する
   copied=CopyTime(Symbol(),0,0,100,time);
   if(copied<=0)
     {
      Print("Failed to copy the values of the price series of Time");
      return;
     }
//--- 複製されたデータへのアクセスを時系列のように(反対に)整理する
   ArraySetAsSeries(high,true);
   ArraySetAsSeries(low,true);
   ArraySetAsSeries(time,true);

//--- フィボナッチオブジェクトの初めのアンカーポイントの座標
   price1=high[70];
   time1=time[70];
//--- フィボナッチオブジェクトの2 番目のアンカーポイントの座標
   price2=low[50];
   time2=time[50];

//--- フィボナッチオブジェクトを作成する
```

© 2000-2025, MetaQuotes Ltd.

```
   bool created=ObjectCreate(0,"Fibo",OBJ_FIBO,0,time1,price1,time2,price2);
   if(created) // オブジェクト作成が成功した場合
     {
      //--- フィボナッチレベルの色を設定する
      ObjectSetInteger(0,"Fibo",OBJPROP_LEVELCOLOR,Blue);
      //--- フィボナッチレベルの数を見つける
      int levels=ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS);
      Print("Fibo levels before = ",levels);
      //---操作ログに出力 => number of level:value level_description
      for(int i=0;i<levels;i++)
        {
         Print(i,": ",ObjectGetDouble(0,"Fibo",OBJPROP_LEVELVALUE,i),
               "  ",ObjectGetString(0,"Fibo",OBJPROP_LEVELTEXT,i));
        }
      //--- 単位当たりのレベルの数の増加を試みる
      bool modified=ObjectSetInteger(0,"Fibo",OBJPROP_LEVELS,levels+1);
      if(!modified) // レベル数の変更に失敗
        {
         Print("Failed to change the number of levels of Fibo, error
",GetLastError());
        }
      //--- 情報のみ
      Print("Fibo levels after = ",ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS));
      //--- 新しく作成したレベルの値を設定する
      bool added=ObjectSetDouble(0,"Fibo",OBJPROP_LEVELVALUE,levels,133);
      if(added) // レベルに値を設定することができた
        {
         Print("Successfully set one more Fibo level");
         //--- レベル説明を忘れずに設定
         ObjectSetString(0,"Fibo",OBJPROP_LEVELTEXT,levels,"my level");
         ChartRedraw(0);
         //--- フィボナッチオブジェクトの実際のレベル数を取得する
         levels=ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS);
         Print("Fibo levels after adding = ",levels);
         //--- 確認のためにもう一度全レベルを出力する
         for(int i=0;i<levels;i++)
           {
            Print(i,":",ObjectGetDouble(0,"Fibo",OBJPROP_LEVELVALUE,i),
                  "  ",ObjectGetString(0,"Fibo",OBJPROP_LEVELTEXT,i));
           }
        }
      else // フィボナッチオブジェクトのレベル数を増加しようとすると失敗する
        {
         Print("Failed to set one more Fibo level. Error ",GetLastError());
        }
     }
  }
```

参照

© 2000-2025, MetaQuotes Ltd.

オブジェクト の型 、オブジェクト プロパティ

© 2000–2025, MetaQuotes Ltd.

ObjectSetInteger

この関数は、対応するオブジェクトプロパティの値を設定します。オブジェクトプロパティは datetime、int、color、bool または char 型でなければなりません。この関数には 2 つのバージョンがあります。

修飾語なしでのプロパティ値の設定

```
bool  ObjectSetInteger(
   long                              chart_id,          // チャート識別子
   string                            name,              // オブジェクト名
   ENUM_OBJECT_PROPERTY_INTEGER      prop_id,           // プロパティ
   long                              prop_value         // 値
   );
```

修飾子を使用してのプロパティ値の設定

```
bool  ObjectSetInteger(
   long                              chart_id,          // チャート識別子
   string                            name,              // オブジェクト名
   ENUM_OBJECT_PROPERTY_INTEGER      prop_id,           // プロパティ
   int                               prop_modifier,     // 修飾子
   long                              prop_value         // 値
   );
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] オブジェクト名

prop_id

[in] オブジェクトプロパティ識別子。値は ENUM_OBJECT_PROPERTY_INTEGER 列挙のいずれかです。

prop_modifier

[in] 指定されたプロパティの修飾子。修飾子はフィボナッチツールのレベル数及びグラフィックオブジェクトのアンドリューズピッチフォークを示します。レベルの番号付けは0から始まります。

prop_value

[in] プロパティ値

戻り値

グラフィックオブジェクトのプロパティ変更コマンドが正常にチャートに送信された場合にのみ true、その他の場合は false。エラーの詳細を読むには GetLastError() を呼びます。

注意事項

この関数は非同期呼び出しを使用します。つまり、この関数は指定されたチャートのキューに追加されたコマンドの実行を待機せずに、代わりに、すぐに制御を返します。

© 2000–2025, MetaQuotes Ltd.

コマンドの実行結果を確認するには指定されたオブジェクトプロパティをレクエストする関数を使用できますが、そのような関数はそのチャートのキューの最後に追加され、実行結果を待つため時間がかかることにご注意ください。チャート上で多数のオブジェクトを操作する場合は、この特徴を考慮する必要があります。

ウェブカラーの表を作成する例

```
//+------------------------------------------------------------------+
//|                                                Table of Web Colors|
//|                        Copyright 2011, MetaQuotes Software Corp |
//|                                             https://www.metaquotes.net |
//+------------------------------------------------------------------+
#define X_SIZE 140       // 編集オブジェクトの幅
#define Y_SIZE 33        // 編集オブジェクトの幅の高さ
//+------------------------------------------------------------------+
//| ウェブカラーの配列                                                |
//+------------------------------------------------------------------+
color ExtClr[140]=
  {
   clrAliceBlue
,clrAntiqueWhite
,clrAqua,clrAquamarine,clrAzure,clrBeige,clrBisque,clrBlack,clrBlanchedAlmond,
   clrBlue
,clrBlueViolet
,clrBrown
,clrBurlyWood,clrCadetBlue,clrChartreuse,clrChocolate,clrCoral,clrCornflowerBlue,
   clrCornsilk
,clrCrimson
,clrCyan
,clrDarkBlue,clrDarkCyan,clrDarkGoldenrod,clrDarkGray,clrDarkGreen,clrDarkKhaki,
   clrDarkMagenta
,clrDarkOliveGreen
,clrDarkOrange,clrDarkOrchid,clrDarkRed,clrDarkSalmon,clrDarkSeaGreen,
   clrDarkSlateBlue
,clrDarkSlateGray
,clrDarkTurquoise,clrDarkViolet,clrDeepPink,clrDeepSkyBlue,clrDimGray,
   clrDodgerBlue
,clrFireBrick
,clrFloralWhite,clrForestGreen,clrFuchsia,clrGainsboro,clrGhostWhite,clrGold,
   clrGoldenrod
,clrGray
,clrGreen
,clrGreenYellow,clrHoneydew,clrHotPink,clrIndianRed,clrIndigo,clrIvory,clrKhaki,
   clrLavender
,clrLavenderBlush
,clrLawnGreen,clrLemonChiffon,clrLightBlue,clrLightCoral,clrLightCyan,
   clrLightGoldenrod
,clrLightGreen
,clrLightGray,clrLightPink,clrLightSalmon,clrLightSeaGreen,clrLightSkyBlue,
   clrLightSlateGray
,clrLightSteelBlue
,clrLightYellow,clrLime,clrLimeGreen,clrLinen,clrMagenta,clrMaroon,
   clrMediumAquamarine
,clrMediumBlue,clrMediumOrchid,clrMediumPurple,clrMediumSeaGreen,clrMediumSlateBlue,
```

© 2000–2025, MetaQuotes Ltd.

```
   clrMediumSpringGreen
,clrMediumTurquoise
,clrMediumVioletRed,clrMidnightBlue,clrMintCream,clrMistyRose,clrMoccasin,
   clrNavajoWhite
,clrNavy
,clrOldLace,clrOlive,clrOliveDrab,clrOrange,clrOrangeRed,clrOrchid,clrPaleGoldenrod,
   clrPaleGreen
,clrPaleTurquoise
,clrPaleVioletRed,clrPapayaWhip,clrPeachPuff,clrPeru,clrPink,clrPlum,clrPowderBlue,
   clrPurple
,clrRed
,clrRosyBrown
,clrRoyalBlue,clrSaddleBrown,clrSalmon,clrSandyBrown,clrSeaGreen,clrSeashell,
   clrSienna
,clrSilver
,clrSkyBlue
,clrSlateBlue,clrSlateGray,clrSnow,clrSpringGreen,clrSteelBlue,clrTan,clrTeal,
   clrThistle
,clrTomato
,clrTurquoise,clrViolet,clrWheat,clrWhite,clrWhiteSmoke,clrYellow,clrYellowGreen
  };
//+------------------------------------------------------------------+
//|  編集オブジェクトを作成と初期化                                  |
//+------------------------------------------------------------------+
void CreateColorBox(int x,int y,color c)
  {
//--- 新しい編集オブジェクトの名称を作成
   string name="ColorBox_"+(string)x+"_"+(string)y;
//--- 新しい編集オブジェクトを作成
   if(!ObjectCreate(0,name,OBJ_EDIT,0,0,0))
     {
      Print("Cannot create: '",name,"'");
      return;
     }
//--- 座礁、幅及び高さを設定する
   ObjectSetInteger(0,name,OBJPROP_XDISTANCE,x*X_SIZE);
   ObjectSetInteger(0,name,OBJPROP_YDISTANCE,y*Y_SIZE);
   ObjectSetInteger(0,name,OBJPROP_XSIZE,X_SIZE);
   ObjectSetInteger(0,name,OBJPROP_YSIZE,Y_SIZE);
//--- テキストの色を設定する
   if(clrBlack==c) ObjectSetInteger(0,name,OBJPROP_COLOR,clrWhite);
   else            ObjectSetInteger(0,name,OBJPROP_COLOR,clrBlack);
//--- 背景色を設定する
   ObjectSetInteger(0,name,OBJPROP_BGCOLOR,c);
//--- テキストを設定する
   ObjectSetString(0,name,OBJPROP_TEXT,(string)c);
  }
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                              |
//+------------------------------------------------------------------+
void OnStart()
```

© 2000–2025, MetaQuotes Ltd.

```
  {
//--- 色のついた編集オブジェクトの 7x20 の表を作る
   for(uint i=0;i<140;i++)
      CreateColorBox(i%7,i/7,ExtClr[i]);
  }
```

参照

オブジェクトの型、オブジェクトプロパティ

© 2000-2025, MetaQuotes Ltd.

ObjectSetString

この関数は、対応するオブジェクトプロパティの値を設定します。オブジェクトプロパティは string 型でなければなりません。この関数には 2 つのバージョンがあります。

修飾語なしでのプロパティ値の設定

```
bool  ObjectSetString(
   long                          chart_id,          // チャート識別子
   string                        name,              // オブジェクト名
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // プロパティ
   string                        prop_value         // 値
   );
```

修飾子を使用してのプロパティ値の設定

```
bool  ObjectSetString(
   long                          chart_id,          // チャート識別子
   string                        name,              // オブジェクト名
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // プロパティ
   int                           prop_modifier,     // 修飾子
   string                        prop_value         // 値
   );
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] オブジェクト名

prop_id

[in] オブジェクトプロパティ識別子。値は ENUM_OBJECT_PROPERTY_STRING 列挙のいずれかです。

prop_modifier

[in] 指定されたプロパティの修飾子。修飾子はフィボナッチツールのレベル数及びグラフィックオブジェクトのアンドリューズピッチフォークを示します。レベルの番号付けは0から始まります。

prop_value

[in] プロパティ値

戻り値

グラフィックオブジェクトのプロパティ変更コマンドが正常にチャートに送信された場合にのみ true、その他の場合は false。エラーの詳細を読むには GetLastError() を呼びます。

注意事項

この関数は非同期呼び出しを使用します。つまり、この関数は指定されたチャートのキューに追加されたコマンドの実行を待機せずに、代わりに、すぐに制御を返します。

© 2000–2025, MetaQuotes Ltd.

コマンドの実行結果を確認するには指定されたオブジェクトプロパティをレクエストする関数を使用できますが、そのような関数はそのチャートのキューの最後に追加され、実行結果を待つため時間がかかることにご注意ください。チャート上で多数のオブジェクトを操作する場合は、この特徴を考慮する必要があります。

オブジェクトの名称が変更された場合、2つのイベントが同時に形成されます。これらのイベントはエキスパートアドバイザーまたは指標の OnChartEvent() 関数によって処理されます。

- 古い名称のオブジェクトの削除イベント
- 新しい名称のオブジェクトの作成イベント

© 2000–2025, MetaQuotes Ltd.

ObjectGetDouble

この関数は、対応するオブジェクトプロパティの値を返します。オブジェクトプロパティは double 型でなければなりません。この関数には 2 つのバージョンがあります。

1. すぐにプロパティ値を返します。

```
double  ObjectGetDouble(
   long                          chart_id,          // チャート識別子
   string                        name,              // オブジェクト名
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // プロパティ識別子
   int                           prop_modifier=0    // 必要な場合、プロパティ修飾子
   );
```

2. 関数実行の成功に応じて true または false を返します。成功した場合、プロパティ値は最後のパラメータで参照によって受け渡された変数に配置されます。

```
bool  ObjectGetDouble(
   long                          chart_id,          // チャート識別子
   string                        name,              // オブジェクト名
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // プロパティ識別子
   int                           prop_modifier,     // プロパティ修飾子
   double&                       double_var         // プロパティ値を受け取る
   );
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] オブジェクト名

prop_id

[in] オブジェクトプロパティ識別子。値は ENUM_OBJECT_PROPERTY_DOUBLE 列挙のいずれかです。

prop_modifier

[in] 指定されたプロパティの修飾子。1 番目のバージョン例では、デフォルト修飾子の値が 0 に等しいです。ほとんどのプロパティは修飾子を必要としません。修飾子はフィボナッチツールのレベル数及びグラフィックオブジェクトのアンドリューズピッチフォークを示します。レベルの番号付けは0から始まります。

double_var

[out] リクエストされたプロパティの値を受け取ったdouble 型の変数

戻り値

最初の呼び出しバージョン例では、double 型の値

2 番目の呼び出しバージョン例では、指定されたプロパティが維持され、その値がdouble_var 変数に置かれている場合には true, それ以外の場合は false。エラーの詳細を読むには GetLastError() を呼びます。

注意事項

© 2000–2025, MetaQuotes Ltd.

この関数は同期呼び出しを使用します。つまり、この関数が呼び出される前にこのチャートのエンキューされたすべてのコマンドの実行が待機されるため、この関数は時間がかかることがあります。チャート上で多数のオブジェクトを操作する場合は、この特徴を考慮する必要があります。

© 2000–2025, MetaQuotes Ltd.

ObjectGetInteger

この関数は、対応するオブジェクトプロパティの値を返します。オブジェクトプロパティは datetime、int、color、bool または char 型でなければなりません。この関数には 2 つのバージョンがあります。

1. すぐにプロパティ値を返します。

```
long  ObjectGetInteger(
   long                          chart_id,          // チャート識別子
   string                        name,              // オブジェクト名
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,           // プロパティ識別子
   int                           prop_modifier=0    // 必要な場合、プロパティ修飾子
   );
```

2. 関数実行の成功に応じて true または false を返します。成功した場合、プロパティ値は最後のパラメータで参照によって受け渡された変数に配置されます。

```
bool  ObjectGetInteger(
   long                          chart_id,          // チャート識別子
   string                        name,              // オブジェクト名
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,           // プロパティ識別子
   int                           prop_modifier,     // プロパティ修飾子
   long&                         long_var           // プロパティ値を受け取る
   );
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] オブジェクト名

prop_id

[in] オブジェクトプロパティ識別子。値は ENUM_OBJECT_PROPERTY_INTEGER 列挙のいずれかです。

prop_modifier

[in] 指定されたプロパティの修飾子。1 番目のバージョン例では、デフォルト修飾子の値が 0 に等しいです。ほとんどのプロパティは修飾子を必要としません。修飾子はフィボナッチツールのレベル数及びグラフィックオブジェクトのアンドリューズピッチフォークを示します。レベルの番号付けは0から始まります。

long_var

[out] リクエストされたプロパティの値を受け取る long 型の変数

戻り値

最初の呼び出しバージョン例では long 型の値

2 番目の呼び出しバージョン例では、指定されたプロパティが維持され、その値がlong_var 変数に置かれている場合には true，それ以外の場合は falseエラーの詳細を読むには GetLastError() を呼びます。

注意事項

© 2000–2025, MetaQuotes Ltd.

この関数は同期呼び出しを使用します。つまり、この関数が呼び出される前にこのチャートのエンキューされたすべてのコマンドの実行が待機されるため、この関数は時間がかかることがあります。チャート上で多数のオブジェクトを操作する場合は、この特徴を考慮する必要があります。

© 2000–2025, MetaQuotes Ltd.

ObjectGetString

この関数は、対応するオブジェクトプロパティの値を返します。オブジェクトプロパティは string 型でなければなりません。この関数には 2 つのバージョンがあります。

1. すぐにプロパティ値を返します。

```
string  ObjectGetString(
   long                          chart_id,          // チャート識別子
   string                        name,              // オブジェクト名
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // プロパティ識別子
   int                           prop_modifier=0    // 必要な場合、プロパティ修飾子
   );
```

2. 関数実行の成功に応じて true または false を返します。成功した場合、プロパティ値は最後のパラメータで参照によって受け渡された変数に配置されます。

```
bool  ObjectGetString(
   long                          chart_id,          // チャート識別子
   string                        name,              // オブジェクト名
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // プロパティ識別子
   int                           prop_modifier,     // プロパティ修飾子
   string&                       string_var         // プロパティ値を受け取る
   );
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] オブジェクト名

prop_id

[in] オブジェクトプロパティ識別子。値は ENUM_OBJECT_PROPERTY_STRING 列挙のいずれかです。

prop_modifier

[in] 指定されたプロパティの修飾子。1 番目のバージョン例では、デフォルト修飾子の値が 0 に等しいです。ほとんどのプロパティは修飾子を必要としません。修飾子はフィボナッチツールのレベル数及びグラフィックオブジェクトのアンドリューズピッチフォークを示します。レベルの番号付けは0から始まります。

string_var

[out] リクエストされたプロパティ値を受け取る string 型の変数

戻り値

最初の呼び出しバージョン例では string 型の値

2 番目の呼び出しバージョン例では、指定されたプロパティが維持され、その値が string_var 変数に置かれている場合には true，それ以外の場合は falseエラーの詳細を読むには GetLastError() を呼びます。

注意事項

© 2000–2025, MetaQuotes Ltd.

この関数は同期呼び出しを使用します。つまり、この関数が呼び出される前にこのチャートのエンキューされたすべてのコマンドの実行が待機されるため、この関数は時間がかかることがあります。チャート上で多数のオブジェクトを操作する場合は、この特徴を考慮する必要があります。

オブジェクトの名称が変更された場合、2つのイベントが同時に形成されます。これらのイベントはエキスパートアドバイザーまたは指標の OnChartEvent() 関数によって処理されます。

- 古い名称のオブジェクトの削除イベント
- 新しい名称のオブジェクトの作成イベント

© 2000–2025, MetaQuotes Ltd.

TextSetFont

この関数は、描画メソッドを使用してテキストを表示するためのフォントを設定します。デフォルトとしてArial120(12 pt)が使用されます。

```
bool  TextSetFont(
   const string  name,              // フォント名またはディスクでのフォントへのパス
   int           size,              // フォントサイズ
   uint          flags,             // フラグの組み合わせ
   int           orientation=0      // テキストの傾斜角度
   );
```

パラメータ

name

[in] システム内フォント名、またはディスク上のフォントやフォントファイルのパスを含むリソースの名称。

size

[in] 正負の値で設定出来るフォントサイズ。正の値の場合は、表示されるテキストのサイズは、オペレーティングシステムのフォントサイズの設定に依存しません。負の値の場合、少数第一の桁で設定され、テキストのサイズは、オペレーティングシステムの設定(「標準規模」や「大規模」)に依存します。モードの違いについての詳細は、下記の注意事項を参照してください。

flags

[in] フォントスタイルを記述するフラグの組み合わせ

orientation

[in] 0.1度の測定単位での、X軸に対するテキストの水平方向の傾き。例えば *orientation=450* は45度に等しい傾きを表しています。

戻り値

現在のフォントが正常にインストールされている場合は true それ以外の場合は false を返します。可能なエラーコードは下記です。

- ERR_INVALID_PARAMETER(4003) - *name* が NULL または ""(空の文字列)です。
- ERR_INTERNAL_ERROR(4001) – オペレーティングシステムエラー(例えば、存在しないフォントを作成する試み)

注意事項

「::」がフォント名に使用された場合、フォントは EX5 リソースからダウンロードされます。*name* フォント名が拡張子と指定された場合ファイルからダウンロードされ、パスが「¥」または「/」から始まる場合、ファイルがMQL5 ディレクトリと相対して検索されます。それ以外の場合は TextSetFont() 関数を呼び出した EX5 ファイルのパスを基準に検索されます。

フォントサイズは、正または負の値を使用して設定されます。この事実は、テキストサイズのオペレーティングシステムの設定(サイズスケール)への依存性を定義します。

- サイズが正の数で指定された場合、このサイズは、使用可能なフォントから選ばれたシンボルのグリフの高さに一致するデバイスの物理的な測定単位(ピクセル)に変換されます。これは、TextOut() 関数で表示されたテキストとOBJ_LABEL ("Label") で表示されたテキストがチャートで一緒に使用される場合には推奨されていません。
- サイズが負の数で指定された場合、この数は論理ポイントの10倍となります(例えば -350 は 35 論理ポイント)。 取得された値は、デバイス(ピクセル)の物理的な測定単位に変換され、利用可能なフォ

© 2000–2025, MetaQuotes Ltd.

ントから選んだシンボルの高さの絶対値に相当します。画面のテキストのサイズを OBJ_LABEL オブジェクトの物と同じようにするには、オブジェクトプロパティで指定されたフォントサイズを -10 で掛けます。

フラグは、フォント幅を指定するフラグのいずれかの形式のフラグの組み合わせとして使用することが出来ます。フラグ名は下記に見られます。

フォントスタイルを指定するフラグ

フラグ	説明
FONT_ITALIC	イタリック
FONT_UNDERLINE	アンダーライン
FONT_STRIKEOUT	ストライクアウト

フォントの幅を指定するフラグ

フラグ
FW_DONTCARE
FW_THIN
FW_EXTRALIGHT
FW_ULTRALIGHT
FW_LIGHT
FW_NORMAL
FW_REGULAR
FW_MEDIUM
FW_SEMIBOLD
FW_DEMIBOLD
FW_BOLD
FW_EXTRABOLD
FW_ULTRABOLD
FW_HEAVY
FW_BLACK

参照

リソース、ResourceCreate()、ResourceSave()、TextOut()

© 2000-2025, MetaQuotes Ltd.

TextOut

この関数は、カスタム配列（バッファ）にテキストを表示し、その操作の結果を返します。配列はグラフィックリソースの作成に意図されています。

```
bool  TextOut(
   const string        text,          // 表示されたテキスト
   int                 x,             // X 座標
   int                 y,             // Y 座標
   uint                anchor,        // アンカーの種類
   uint                &data[],       // 出力バッファ
   uint                width,         // ピクセル単位でのバッファ幅
   uint                height,        // ピクセル単位でのバッファの高さ
   uint                color,         // テキストの色
   ENUM_COLOR_FORMAT   color_format   // 出力する色のフォーマット
   );
```

パラメータ

text

[in] バッファに書かれる表示テキスト。一行のテキストのみが表示されます。

x

[in] 表示テキストのアンカーポイントの X 座標

y

[in] 表示テキストのアンカーポイントの Y 座標

anchor

[in] 表示テキストのアンカーポイントの位置の事前定義された 9 つの方法のうちの 1 つでの値。この値は、水平方向と垂直方向のテキスト整列の 2 つのフラグの組み合わせで設定されます。フラグ名は下の「注意事項」にリストされています。

data[]

[in] テキストが表示されるバッファ。バッファはグラフィックリソースの作成に意図されています。

幅

[in] ピクセル単位でのバッファ幅

height

[in] ピクセル単位でのバッファの高さ

color

[in] テキストの色

color_format

[in] 色のフォーマットは ENUM_COLOR_FORMAT 列挙値から設定されます。

戻り値

成功の場合は true、それ以外の場合は false。

注意事項

© 2000–2025, MetaQuotes Ltd.

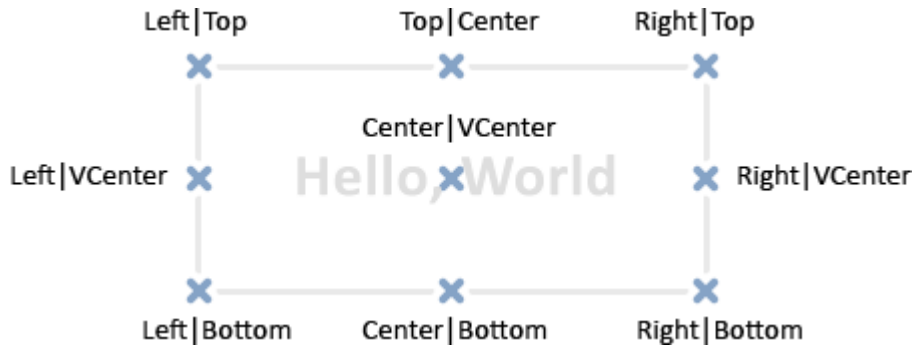
anchor で指定されたアンカーポイントは、水平方向と垂直方向のテキスト整列の 2 つのフラグの組み合わせで指定されます。水平方向テキスト整列のフラグ

- TA_LEFT – アンカーポイントが境界ボックスの左側に位置します。
- TA_CENTER – アンカーポイントが境界ボックスの中心に位置します。
- TA_RIGHT – アンカーポイントが境界ボックスの右側に位置します。

垂直方向テキスト整列のフラグ

- TA_TOP – アンカーポイントが境界ボックスの上側に位置します。
- TA_VCENTER – アンカーポイントが境界ボックスの中心に位置します。
- TA_BOTTOM – アンカーポイントが境界ボックスの下側に位置します。

フラグや指定されたアンカーポイントの可能な組み合わせは画像に示されています。



例:

```
//--- 描画に使用されるキャンバスの幅と高さ
#define IMG_WIDTH  200
#define IMG_HEIGHT 200
//--- スクリプトの起動時にパラメータのウィンドウを表示する
#property script_show_inputs
//--- 色のフォーマットの設定を有効にする
input ENUM_COLOR_FORMAT clr_format=COLOR_FORMAT_XRGB_NOALPHA;
//--- 描画配列(バッファ)
uint ExtImg[IMG_WIDTH*IMG_HEIGHT];
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 描画する OBJ_BITMAP_LABEL オブジェクトを作成する
   ObjectCreate(0,"CLOCK",OBJ_BITMAP_LABEL,0,0,0);
//--- CLOCK オブジェクトに書くためにグラフィックリソースの名称を指定する
   ObjectSetString(0,"CLOCK",OBJPROP_BMPFILE,"::IMG");

//--- 補助変数
   double a;              // 矢印のコーナー
   uint   nm=2700;        // 分のコーナー
   uint   nh=2700*12;     // 時間のコーナー
   uint   w,h;            // テキスト文字列のサイズを受け取る変数
```

© 2000–2025, MetaQuotes Ltd.

```
   int     x,y;           // テキスト文字列のアンカーポイントの現在の座標を算出するための変数

//--- スクリプトが停止するまで、無限ループで時計の針を回転させる
   while(!IsStopped())
     {
      //--- 時計描画のバッファ配列をクリアする
      ArrayFill(ExtImg,0,IMG_WIDTH*IMG_HEIGHT,0);
      //--- 時計面の描画数字のフォントを設定する
      TextSetFont("Arial",-200,FW_EXTRABOLD,0);
      //--- 時計面を描画する
      for(int i=1;i<=12;i++)
        {
         //--- 時計面の現在の時間サイズを受け取る
         TextGetSize(string(i),w,h);
         //--- 時計面の現在の時間サイズを受け取る
         a=-((i*300)%3600*M_PI)/1800.0;
         x=IMG_WIDTH/2-int(sin(a)*80+0.5+w/2);
         y=IMG_HEIGHT/2-int(cos(a)*80+0.5+h/2);
         //--- 時計面の時間を ExtImg[] バッファに出力する
         TextOut(string(i),x,y,TA_LEFT|
TA_TOP,ExtImg,IMG_WIDTH,IMG_HEIGHT,0xFFFFFFFF,clr_format);
        }
      //--- 分針を描画するためのフォントを指定する
      TextSetFont("Arial",-200,FW_EXTRABOLD,-int(nm%3600));
      //--- 分針のサイズを受け取る
      TextGetSize("----->",w,h);
      //--- 時計面の現在の分針のサイズを受け取る
      a=-(nm%3600*M_PI)/1800.0;
      x=IMG_WIDTH/2-int(sin(a)*h/2+0.5);
      y=IMG_HEIGHT/2-int(cos(a)*h/2+0.5);
      //--- 時計面の分針を ExtImg[] バッファに出力する
      TextOut("----->",x,y,TA_LEFT|
TA_TOP,ExtImg,IMG_WIDTH,IMG_HEIGHT,0xFFFFFFFF,clr_format);

      //--- 時針を描画するためのフォントを設定する
      TextSetFont("Arial",-200,FW_EXTRABOLD,-int(nh/12%3600));
      TextGetSize("==>",w,h);
      //--- 時計面での時針の座標を計算する
      a=-(nh/12%3600*M_PI)/1800.0;
      x=IMG_WIDTH/2-int(sin(a)*h/2+0.5);
      y=IMG_HEIGHT/2-int(cos(a)*h/2+0.5);
      //--- 時計面の時針を ExtImg[] バッファに出力する
の種類      TextOut("==>",x,y,TA_LEFT|
TA_TOP,ExtImg,IMG_WIDTH,IMG_HEIGHT,0xFFFFFFFF,clr_format);

      //--- グラフィックリソースを更新する
      ResourceCreate("::IMG",ExtImg,IMG_WIDTH,IMG_HEIGHT,0,0,IMG_WIDTH,clr_format);
      //--- 強制的なチャート更新
      ChartRedraw();

      //--- 時分のカウンタを増加する
```

© 2000-2025, MetaQuotes Ltd.

```
       nm+=60;
       nh+=60;
       //--- フレーム間に短い休止を維持する
       Sleep(10);
      }
//--- スクリプトの操作が終了した際に CLOCK オブジェクトを削除する
   ObjectDelete(0,"CLOCK");
//---
  }
```

参照

リソース、ResourceCreate()、ResourceSave()、TextGetSize()、TextSetFont()

© 2000-2025, MetaQuotes Ltd.

TextGetSize

この関数は現在のフォント設定での行の幅と高さを返します。

```
bool  TextGetSize(
   const string       text,          // テキスト文字列
   uint&               width,        // ピクセル単位でのバッファ幅
   uint&               height        // ピクセル単位でのバッファの高さ
   );
```

パラメータ

text

[in] 長さと幅が取得される文字列

幅

[out] 幅を受け取る入力パラメータ

height

[out] 高さを受け取る入力パラメータ

戻り値

成功の場合は true、それ以外の場合は false。可能なエラーコードは下記です。

- ERR_INTERNAL_ERROR(4001) – operating system error（オペレーティングシステムエラー）

参照

リソース、ResourceCreate()、ResourceSave()、TextSetFont()、TextOut()

© 2000–2025, MetaQuotes Ltd.

テクニカル指標関数

iMA、iAC、iMACD、iIchimoku のような全ての関数は、クライアント端末のグローバルキャッシュに対応するテクニカル指標のコピーを作成します。このようなパラメータを持つ指標の複製がすでに存在する場合は、新しいコピーは作成されず、既存のコピーの参照カウンタが増加します。

これらの関数は、指標の適切なコピーのハンドルを返します。更に、このハンドルを使用して、対応する指標によって計算されたデータを受信することが出来ます。対応するバッファのデータが(テクニカル指標は、指標に応じて 1～5 の内部バッファ内に計算されたデータを含みます)は CopyBuffer() 関数を使用して MQL5 プログラムに複製することが出来ます。

指標値の計算には時間が必要で作成直後に指標データを参照することは出来ないので OnInit() にハンドルを作成する方が良いでしょう。iCustom() 関数は対応するカスタム指標を作成し、正常に作成された場合にはそのハンドルを返します。カスタム指標は指標バッファを 512 まで持つことができ、その内容はハンドルを使用して CopyBuffer() 関数で取得することが出来ます。

テクニカル指標の作成には一般的にIndicatorCreate() 関数が使用されます。この関数は下記の入力パラメータを受け入れます。

- 銘柄名
- 時間軸
- 作成する指標の種類
- 指標の入力パラメータの数
- 必要な入力パラメータを全て含む MqlParam 型の配列

IndicatorRelease() 関数に指標ハンドルを渡すことによって、利用されていない指標のコンピュータメモリを解放することが出来ます。

注意事項:1 つの MQL5 プログラム内で指標関数を同じパラメータで繰り返し呼び出すことは、参照カウンタの複数の増加につながらず、カウンタは 1 回のみ増加します。しかし OnInit() 関数またはクラスコンストラクタで指標ハンドルを取得し、他の関数でもこれらのハンドルを使用することをお勧めします。参照カウンタは MQL5 プログラムが初期化解除された際には減少します。

全ての指標関数には、少なくとも銘柄と期間の 2 つのパラメータがあります。シンボルの NULL 値は現在のシンボルを意味し、期間の 0 値は現在の時間軸を意味します。

関数	返される指標ハンドル
iAC	ACオシレーター。
iAD	蓄積/配信(Accumulation/Distribution)。
iADX	平均方向性指数(Average Directional Index)。
iADXWilder	ウェルズ·ワイルダーの平均方向性指数(Average Directional Index by Welles Wilder)
iAlligator	アリゲーター。
iAMA	適応型移動平均(Adaptive Moving Average)。
iAO	オーサムオシレーター。
iATR	ATR(Average True Range)。

© 2000-2025, MetaQuotes Ltd.

関数	返される指標ハンドル
iBearsPower	ベアパワー（Bears Power）。
iBands	ボリンジャーバンド®（Bollinger Bands®）。
iBullsPower	ブルパワー（Bulls Power）。
iCCI	コモディティチャンネルインデックス（Commodity Channel Index）。
iChaikin	チャイキンオシレーター（Chaikin Oscillator）。
iCustom	カスタム指標。
iDEMA	2重指数移動平均（Double Exponential Moving Average）。
iDeMarker	デマーカー（DeMarker）。
iEnvelopes	エンベロープ（Envelopes）。
iForce	勢力指数（Force Index）。
iFractals	フラクタル。
iFrAMA	フラクタル適応型移動平均（Fractal Adaptive Moving Average）。
iGator	ゲーターオシレーター。
iIchimoku	一目均衡表（Ichimoku Kinko Hyo）。
iBWMFI	ビル・ウィリアムズのマーケットファシリテーションインデックス
iMomentum	モメンタム（Momentum）。
iMFI	マネーフローインデックス（Money Flow Index）。
iMA	移動平均（Moving Average）。
iOsMA	移動平均オシレーター（Moving Average of Oscillator）（MACDヒストグラム）。
iMACD	移動平均収束拡散法（Moving Averages Convergence-Divergence）
iOBV	オンバランスボリューム（On Balance Volume）。
iSAR	パラボリック停止・リバースシステム（Parabolic Stop And Reverse System）
iRSI	相対力指数（Relative Strength Index）。
iRVI	相対活力指数（Relative Vigor Index）。
iStdDev	標準偏差（Standard Deviation）。
iStochastic	ストキャスティックス（Stochastic Oscillator）。
iTEMA	3重指数移動平均（Triple Exponential Moving Average）。
iTriX	3　重指数移動平均オシレーター（Triple Exponential Moving Averages Oscillator）。
iWPR	ウィリアムパーセントレンジ（Williams' Percent Range）。
iVIDyA	可変インデックス動的平均（Variable Index Dynamic Average）。

© 2000–2025, MetaQuotes Ltd.

関数	返される指標ハンドル
iVolumes	ボリューム。

© 2000–2025, MetaQuotes Ltd.

iAC

この関数はクライアント端末のグローバルキャッシュに Accelerator Oscillator（ AC オシレーター）を作成してハンドルを返します。バッファは 1 つです。

```
int  iAC(
   string           symbol,     // 銘柄名
   ENUM_TIMEFRAMES  period      // 期間
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって 、コンピュータメモリを利用されていない指標から解放することが出来ます。

例：

```
//+------------------------------------------------------------------+
//|                                                     Demo_iAC.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iAC technical indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the
'type' parameter (function type)."

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- i AC のプロット
#property indicator_label1  "iAC"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen, clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
```

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAC,               // iAC を使用する
   Call_IndicatorCreate    // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iAC;          // 関数の種類
input string                symbol=" ";             // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // 時間軸
//--- 指標バッファ
double iACBuffer[];
double iACColors[];
//--- iAC 指標ハンドルを格納する変数
int    handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- ACオシレーター指標に値の数を保存
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,iACBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iACColors,INDICATOR_COLOR_INDEX);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iAC)
      handle=iAC(name,period);
   else
      handle=IndicatorCreate(name,period,IND_AC);
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("Failed to create handle of the iAC indicator for the symbol %s/%
s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- ACオシレーターが計算された銘柄/時間軸を表示
   short_name=StringFormat("iAC(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iAC 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
//--- これが指標計算の初めであるか iAC 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- もし iACBuffer 配列が銘柄/期間で iAC 指標の値の数より大きい場合、全体のコピーはしない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- これは初回の計算ではなく、
       //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
       values_to_copy=(rates_total-prev_calculated)+1;
      }
//--- ACオシレーター指標の値で iACBuffer と iACColors 配列を記入する
//--- FillArraysFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArraysFromBuffer(iACBuffer,iACColors,handle,values_to_copy)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- ACオシレーター指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iAC 指標から指標バッファを記入する                                |
//+------------------------------------------------------------------+
bool FillArraysFromBuffer(double &values[],         // ACオシレーター値の指標バッファ
                          double &color_indexes[], // 色のバッファ(色インデックスを格納)
                          int ind_handle,          // iAC 指標ハンドル
                          int amount               // 複製された値の数
                         )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で iACBuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iAC indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 色インデックスを複製する
   if(CopyBuffer(ind_handle,1,0,amount,color_indexes)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy color values from the iAC indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指標初期化解除関数                                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000–2025, MetaQuotes Ltd.

iAD

この関数は Accumulation（蓄積）/ Distribution（分配）指標ハンドルを返します。バッファは 1 つです。

```
int  iAD(
   string               symbol,              // 銘柄名
   ENUM_TIMEFRAMES      period,              // 期間
   ENUM_APPLIED_VOLUME  applied_volume       // 計算に使用するボリュームの種類
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

applied_volume

[in] 使用されるボリューム（ENUM_APPLIED_VOLUME 値のいずれか）

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって、コンピュータメモリを利用されていない指標から解放することが出来ます。

例：

```
//+------------------------------------------------------------------+
//|                                                     Demo_iAD.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iAD technical indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the
'type' parameter (function type)."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- i AD をプロットする
#property indicator_label1  "iAD"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
```

© 2000–2025, MetaQuotes Ltd.

```
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAD,                // ｉAD を使用する
   Call_IndicatorCreate     // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iAD;           // 型の関数
input ENUM_APPLIED_VOLUME   volumes;                 // 使用されるボリューム
input string                symbol=" ";              // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;   // 時間軸
//--- 指標バッファ
double          iADBuffer[];
//--- ｉAD 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- A/D 指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,iADBuffer,INDICATOR_DATA);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iAD)
      handle=iAD(name,period,volumes);
   else
     {
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[1];
```

© 2000-2025, MetaQuotes Ltd.

```
      pars[0].type=TYPE_INT;
      pars[0].integer_value=volumes;
      handle=IndicatorCreate(name,period,IND_AD,1,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iAD indicator for the symbol %s/%
s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- A/D が計算された銘柄/時間軸を表示
   short_name=StringFormat("iAD(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iAD 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
//--- これが指標計算の初めであるか iAD 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
```

© 2000-2025, MetaQuotes Ltd.

```
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- もし iADBuffer 配列が銘柄/期間で iAD 指標の値の数より大きい場合、全体のコピーはしない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- これは初回の計算ではなく、
      //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iADBuffer 配列をA/D の値で記入
//--- FillArraysFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArrayFromBuffer(iADBuffer,handle,values_to_copy)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- A/D 指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iAD 指標から指標バッファを記入する                                |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],   // A/D 指標値の指標バッファ
                         int ind_handle,     // iAD 指標ハンドル
                         int amount          // 複製された値の数
                        )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で iADBuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iAD indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
```

© 2000-2025, MetaQuotes Ltd.

```
    return(true);
   }
//+------------------------------------------------------------------+
//|  指標初期化解除関数                                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
   {
    if(handle!=INVALID_HANDLE)
       IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
    Comment("");
   }
```

© 2000–2025, MetaQuotes Ltd.

iADX

この関数は ADX（Average Directional Movement ）指標ハンドルを返します。

```
int  iADX(
   string            symbol,          // 銘柄名
   ENUM_TIMEFRAMES   period,          // 期間
   int               adx_period       // 平均期間
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

adx_period

[in] インデックス計算の期間

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって 、コンピュータメモリを利用されていない指標から解放することが出来ます。

注意事項

バッファ番号は 0 - MAIN_LINE、1 - PLUSDI_LINE、2 - MINUSDI_LINE です。

例：

```
//+------------------------------------------------------------------+
//|                                                   Demo_iADX.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iADX technical indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the
'type' parameter (function type)."

#property indicator_separate_window
#property indicator_buffers 3
#property indicator_plots   3
//--- ADX をプロットする
#property indicator_label1  "ADX"
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- DI_plus をプロットする
#property indicator_label2  "DI_plus"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrYellowGreen
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- DI_minus をプロットする
#property indicator_label3  "DI_minus"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrWheat
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iADX,              // use iADX を使用する
   Call_IndicatorCreate    // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iADX;          // 関数の種類
input int                   adx_period=14;           // 計算期間
input string                symbol=" ";              // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;   // 時間軸
//--- 指標バッファ
double          ADXBuffer[];
double          DI_plusBuffer[];
double          DI_minusBuffer[];
//--- iADX 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- ADX 指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,ADXBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,DI_plusBuffer,INDICATOR_DATA);
```

© 2000-2025, MetaQuotes Ltd.

```
   SetIndexBuffer(2,DI_minusBuffer,INDICATOR_DATA);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iADX)
      handle=iADX(name,period,adx_period);
   else
     {
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=adx_period;
      handle=IndicatorCreate(name,period,IND_ADX,1,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iADX indicator for the symbol %s/%
s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- ADX が計算された銘柄/時間軸を表示
   short_name=StringFormat("iADX(%s/%s period=%
d)",name,EnumToString(period),adx_period);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iADX 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
//--- これが指標計算の初めであるか iADX 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- もし iADXBuffer 配列が銘柄/期間で iADX 指標の値の数より大きい場合、全体のコピーはしない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- これは初回の計算ではなく、
      //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- 配列をADX 指標の値で記入
//--- FillArraysFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArraysFromBuffers(ADXBuffer,DI_plusBuffer,DI_minusBuffer,handle,values_to
_copy)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- Average Directional Movement Index 指標の値の数を覚える<55/66/79% >
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  i ADX 指標から指標バッファを記入する                                |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &adx_values[],      // ADX 指標値の指標バッファ
                           double &DIplus_values[],   // DI + 指標バッファ
                           double &DIminus_values[],  // DI - 指標バッファ
                           int ind_handle,            // i ADX 指標ハンドル
                           int amount                 // 複製された値の数
                          )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で i ADXBuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,0,amount,adx_values)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iADX indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }

//--- インデックス 1 を持つ指標バッファの値で DI_plusBuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,1,0,amount,DIplus_values)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iADX indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }

//--- インデックス 2 を持つ指標バッファの値で DI_minusBuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,2,0,amount,DIminus_values)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iADX indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  指標初期化解除関数                                                |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
    if(handle!=INVALID_HANDLE)
       IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
    Comment("");
   }
```

© 2000-2025, MetaQuotes Ltd.

iADXWilder

この関数は Average Directional Movement Index by Welles Wilder のハンドルを返します。

```
int  iADXWilder(
   string            symbol,          // 銘柄名
   ENUM_TIMEFRAMES   period,          // 期間
   int               adx_period       // 平均期間
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

adx_period

[in] インデックス計算の期間

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって 、コンピュータメモリを利用されていない指標から解放することが出来ます。

注意事項

バッファ番号は 0 - MAIN_LINE、1 - PLUSDI_LINE、2 - MINUSDI_LINE です。

例：

```
//+------------------------------------------------------------------+
//|                                                   iADXWilder.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iADXWilder technical indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the
'type' parameter (function type)."

#property indicator_separate_window
#property indicator_buffers 3
#property indicator_plots   3
//--- ADX をプロットする
#property indicator_label1  "ADX"
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- DI_plus をプロットする
#property indicator_label2  "DI_plus"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrYellowGreen
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- DI_minus をプロットする
#property indicator_label3  "DI_minus"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrWheat
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iADXWilder,         // iADXWilder を使用する
   Call_IndicatorCreate     // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iADXWilder;   // 関数の種類
input int                   adx_period=14;          // 計算期間
input string                symbol=" ";             // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // 時間軸
//--- 指標バッファ
double          ADXBuffer[];
double          DI_plusBuffer[];
double          DI_minusBuffer[];
//--- iADXWilder 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- Average Directional Movement Index by Welles Wilder 指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,ADXBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,DI_plusBuffer,INDICATOR_DATA);
```

© 2000-2025, MetaQuotes Ltd.

```
   SetIndexBuffer(2,DI_minusBuffer,INDICATOR_DATA);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iADXWilder)
      handle=iADXWilder(name,period,adx_period);
   else
     {
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=adx_period;
      handle=IndicatorCreate(name,period,IND_ADXW,1,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iADXWilder indicator for the
symbol %s/%s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- Average Directional Movement Index by Welles Wilder 指標が計算された銘柄/時間軸を表
示
   short_name=StringFormat("iADXWilder(%s/%s period=%
d)",name,EnumToString(period),adx_period);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- ｉADXWｌder 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
//--- これが指標計算の初めであるかｉADXWｌder 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- もしｉADXWｌder Buffer 配列が銘柄/期間でｉADXWｌder 指標の値の数より大きい場合、全体
のコピーはしない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- これは初回の計算ではなく、
      //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- 配列をAverage Directional Movement Index by Welles Wilder 指標の値で記入<81/80/82%
>
//--- FillArraysFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArraysFromBuffers(ADXBuffer,DI_plusBuffer,DI_minusBuffer,handle,values_to
_copy)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- Average Directional Movement Index 指標の値の数を覚える<55/66/79% >
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
```

© 2000-2025, MetaQuotes Ltd.

```
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iADXWilder 指標から指標バッファを記入する                          |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &adx_values[],      // ADX 指標値の指標バッファ
                           double &DIplus_values[],   // DI+ 指標バッファ
                           double &DIminus_values[],  // DI- 指標バッファ
                           int ind_handle,            // iADXWilder 指標ハンドル
                           int amount                 // 複製された値の数
                          )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で iADXBuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,0,amount,adx_values)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iADXWilder indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }

//--- インデックス 1 を持つ指標バッファの値で DI_plusBuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,1,0,amount,DIplus_values)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iADXWilder indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }

//--- インデックス 2 を持つ指標バッファの値で DI_minusBuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,2,0,amount,DIminus_values)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iADXWilder indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指標初期化解除関数                                               |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iAlligator

この関数は Alligator（アリゲーター）指標のハンドルを返します。

```
int  iAlligator(
   string               symbol,           // 銘柄名
   ENUM_TIMEFRAMES      period,           // 期間
   int                  jaw_period,       // 顎の計算期間
   int                  jaw_shift,        // 顎の水平シフト
   int                  teeth_period,     // 歯の計算期間
   int                  teeth_shift,      // 歯の水平シフト
   int                  lips_period,      // 口の計算期間
   int                  lips_shift,       // 口の水平シフト
   ENUM_MA_METHOD       ma_method,        // 平滑化の種類
   ENUM_APPLIED_PRICE   applied_price     // 価格の種類かハンドル
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

jaw_period

[in] 青線の平均期間（アリゲーターの顎）

jaw_shift

[in] 価格チャートに相対した青線のシフト

teeth_period

[in] 赤線の平均期間（アリゲーターの歯）

teeth_shift

[in] 価格チャートに相対した赤線のシフト

lips_period

[in] 緑線の平均期間（アリゲーターの口）

lips_shift

[in] 価格チャートに相対した緑線のシフト

ma_method

[in] 平均化の方法 ENUM_MA_METHOD 値のいずれか

applied_price

[in] 使用される価格。ENUM_APPLIED_PRICE 価格定数のいずれか、または別の指標ハンドル。

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって、コンピュータメモリを利用されていない指標から解放することが出来ます。

© 2000–2025, MetaQuotes Ltd.

注意事項

バッファ番号は 0 – GATORJAW_LINE、1 – GATORTEETH_LINE、2 – GATORLIPS_LINE です。

例:

```
//+------------------------------------------------------------------+
//|                                                Demo_iAlligator.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iAlligator technical indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the
'type' parameter (function type)."
#property description "All the other parameters are similar to the standard
Alligator."

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   3
//--- 顎をプロットする
#property indicator_label1  "Jaws"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 歯をプロットする
#property indicator_label2  "Teeth"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- 口をプロットする
#property indicator_label3  "Lips"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrLime
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
enum Creation
  {
```

© 2000–2025, MetaQuotes Ltd.

```
   Call_iAlligator,         // iAlligator を使用する
   Call_IndicatorCreate     // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iAlligator;   // 関数の種類
input string                symbol=" ";             // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // 時間軸
input int                   jaw_period=13;          // 顎線の期間
input int                   jaw_shift=8;            // 顎線のシフト
input int                   teeth_period=8;         // 歯線の期間
input int                   teeth_shift=5;          // 歯線のシフト
input int                   lips_period=5;          // 口線の期間
input int                   lips_shift=3;           // 口線のシフト
input ENUM_MA_METHOD        MA_method=MODE_SMMA;    // アリゲーター線の平均化の方法
input ENUM_APPLIED_PRICE    applied_price=PRICE_MEDIAN;// アリゲーター計算に使用される価格の種類
//--- 指標バッファ
double          JawsBuffer[];
double          TeethBuffer[];
double          LipsBuffer[];
//--- iAlligator 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- アリゲーター指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,JawsBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,TeethBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,LipsBuffer,INDICATOR_DATA);
//--- それぞれの線のシフトを設定
   PlotIndexSetInteger(0,PLOT_SHIFT,jaw_shift);
   PlotIndexSetInteger(1,PLOT_SHIFT,teeth_shift);
   PlotIndexSetInteger(2,PLOT_SHIFT,lips_shift);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iAlligator)
      handle=iAlligator(name,period,jaw_period,jaw_shift,teeth_period,
                        teeth_shift,lips_period,lips_shift,MA_method,applied_price);
   else
     {
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[8];
     //--- アリゲーター線の期間とシフト
      pars[0].type=TYPE_INT;
      pars[0].integer_value=jaw_period;
      pars[1].type=TYPE_INT;
      pars[1].integer_value=jaw_shift;
      pars[2].type=TYPE_INT;
      pars[2].integer_value=teeth_period;
      pars[3].type=TYPE_INT;
      pars[3].integer_value=teeth_shift;
      pars[4].type=TYPE_INT;
      pars[4].integer_value=lips_period;
      pars[5].type=TYPE_INT;
      pars[5].integer_value=lips_shift;
//--- 平滑化の種類
      pars[6].type=TYPE_INT;
      pars[6].integer_value=MA_method;
//--- 価格の種類
      pars[7].type=TYPE_INT;
      pars[7].integer_value=applied_price;
//--- ハンドルを作成
      handle=IndicatorCreate(name,period,IND_ALLIGATOR,8,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iAlligator indicator for the
symbol %s/%s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- アリゲーターが計算された銘柄/時間軸を表示
   short_name=StringFormat("iAlligator(%s/%s, %d,%d,%d,%d,%d,%
d)",name,EnumToString(period),
```

© 2000–2025, MetaQuotes Ltd.

```
                              jaw_period,jaw_shift,teeth_period,teeth_shift,lips_period
,lips_shift);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iAlligator 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
//--- これが指標計算の初めであるか iAlligator 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- もし JawsBuffer 配列が銘柄/期間で iAlligator 指標の値の数より大きい場合、全体のコピーはしない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- これは初回の計算ではなく、
      //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- 配列をアリゲーター指標の値で記入
//--- FillArraysFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!FillArraysFromBuffers(JawsBuffer,jaw_shift,TeethBuffer,teeth_shift,LipsBuffer
,lips_shift,handle,values_to_copy)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- アリゲーター指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iAlligator 指標から指標バッファを記入する                        |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &jaws_buffer[],  // 顎線の指標バッファ
                           int j_shift,            // 顎線のシフト
                           double &teeth_buffer[], // 歯線の指標バッファ
                           int t_shift,            // 歯線のシフト
                           double &lips_buffer[],  // 口線の指標バッファe
                           int l_shift,            // 口線のシフト
                           int ind_handle,         // iAlligator 指標ハンドル
                           int amount              // 複製された値の数
                          )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で JawsBuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,-j_shift,amount,jaws_buffer)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iAlligator indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }

//--- インデックス0 を持つ指標バッファの値で TeethBuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,1,-t_shift,amount,teeth_buffer)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iAlligator indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- インデックス 2 を持つ指標バッファの値で LipsBuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,2,-l_shift,amount,lips_buffer)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iAlligator indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指標初期化解除関数                                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000–2025, MetaQuotes Ltd.

iAMA

この関数は Adaptive Moving Average（適応型移動平均）指標のハンドルを返します。バッファは 1 つです。

```
int  iAMA(
   string               symbol,              // 銘柄名
   ENUM_TIMEFRAMES      period,              // 期間
   int                  ama_period,          // AMA 平均期間
   int                  fast_ma_period,      // 高速 MA 期間
   int                  slow_ma_period,      // 低速 MA 期間
   int                  ama_shift,           // 指標の水平シフト
   ENUM_APPLIED_PRICE   applied_price        // 価格の種類かハンドル
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

ama_period

[in] 効率係数が計算された計算期間

fast_ma_period

[in] 急速な市場のための平滑化係数計算の高速期間

slow_ma_period

[in] トレンド不在下での平滑化係数計算の低速期間

ama_shift

[in] 価格チャートに相対した指標のシフト

applied_price

[in] 使用される価格。ENUM_APPLIED_PRICE 価格定数のいずれか、または別の指標ハンドル。

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって、コンピュータメモリを利用されていない指標から解放することが出来ます。

例：

```
//+------------------------------------------------------------------+
//|                                                    Demo_iAMA.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

© 2000–2025, MetaQuotes Ltd.

```
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iAMA technical indicator."
#property description "A symbol and timeframe used for calculation of the indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the 'type' parameter (function type)."
#property description "All the other parameters are similar to the standard AMA."

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iAMA をプロットする
#property indicator_label1  "iAMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAMA,              // iAMA を使用する
   Call_IndicatorCreate    // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iAMA;          // 関数の種類
input string                symbol=" ";              // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;   // 時間軸
input int                   ama_period=15;           // 計算期間
input int                   fast_ma_period=2;        // 高速 MA 期間
input int                   slow_ma_period=30;       // 低速 MA 期間
input int                   ama_shift=0;             // 水平シフト
input ENUM_APPLIED_PRICE    applied_price;           // 価格の種類
//--- 指標バッファ
double          iAMABuffer[];
//--- iAMA 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- 適応型移動平均指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- 指標バッファマッピング
   SetIndexBuffer(0,iAMABuffer,INDICATOR_DATA);
//--- シフトを設定
   PlotIndexSetInteger(0,PLOT_SHIFT,ama_shift);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iAMA)
      handle=iAMA(name,period,ama_period,fast_ma_period,slow_ma_period,ama_shift,app
lied_price);
   else
     {
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[5];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ama_period;
      pars[1].type=TYPE_INT;
      pars[1].integer_value=fast_ma_period;
      pars[2].type=TYPE_INT;
      pars[2].integer_value=slow_ma_period;
      pars[3].type=TYPE_INT;
      pars[3].integer_value=ama_shift;
      //--- 価格の種類
      pars[4].type=TYPE_INT;
      pars[4].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_AMA,5,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iAMA indicator for the symbol %s/%
s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 適応型移動平均 が計算された銘柄/時間軸を表示
   short_name=StringFormat("iAMA(%s/%s,%d,%d,%
d,d)",name,EnumToString(period),ama_period,fast_ma_period,slow_ma_period,ama_shift);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iAMA 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
//--- これが指標計算の初めであるか iAMA 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- もし iAMABuffer 配列が銘柄/期間で iAMA 指標の値の数より大きい場合、全体のコピーはしない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- これは初回の計算ではなく、
      //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- 配列を 適応型移動平均指標の値で記入
//--- FillArraysFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!FillArrayFromBuffer(iAMABuffer,ama_shift,handle,values_to_copy)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- 適応型移動平均指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iAMA 指標から指標バッファを記入する                                |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &ama_buffer[],  // AMA 線の指標バッファ
                         int a_shift,           // AMA 線のシフト
                         int ind_handle,        // iAMA 指標ハンドル
                         int amount             // 複製された値の数
                        )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で iAMABuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,-a_shift,amount,ama_buffer)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iAMA indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指標初期化解除関数                                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iAO

この関数は Awesome Oscillator（オーサムオシレータ）指標のハンドルを返します。バッファは 1 つです。

```
int  iAO(
   string           symbol,     // 銘柄名
   ENUM_TIMEFRAMES  period      // 期間
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって 、コンピュータメモリを利用されていない指標から解放することが出来ます。

例：

```
//+------------------------------------------------------------------+
//|                                                     Demo_iAO.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iAO technical indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the
'type' parameter (function type)."

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- i AO プロット
#property indicator_label1  "iAO"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen,clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
```

© 2000–2025, MetaQuotes Ltd.

```
enum Creation
  {
   Call_iAO,                // iAO を使用する
   Call_IndicatorCreate     // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iAO;          // 関数の種類
input string                symbol=" ";             // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // 時間軸
//--- 指標バッファ
double          iAOBuffer[];
double          iAOColors[];
//--- iAO 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- Awesome Oscillator 指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,iAOBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iAOColors,INDICATOR_COLOR_INDEX);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iAO)
      handle=iAO(name,period);
   else
      handle=IndicatorCreate(name,period,IND_AO);
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("Failed to create handle of the iAO indicator for the symbol %s/%
s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- オーサムオシレーターが計算された銘柄/時間軸を表示r
   short_name=StringFormat("iAO(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- ｉAO 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
//--- これが指標計算の初めであるか ｉAO 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- もし ｉAOBuffer 配列が銘柄/期間で ｉAO 指標の値の数より大きい場合、全体のコピーはしない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- これは初回の計算ではなく、
       //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
       values_to_copy=(rates_total-prev_calculated)+1;
      }
//--- オーサムオシレーター指標の値で iAOBuffer と iAOColors 配列を記入する
//--- FillArraysFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArraysFromBuffer(iAOBuffer,iAOColors,handle,values_to_copy)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- オーサムオシレーター 指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iAO 指標から指標バッファを記入する                                |
//+------------------------------------------------------------------+
bool FillArraysFromBuffer(double &values[],         // オーサムオシレーター値の指標バッファ
                          double &color_indexes[], // 色のバッファ(色インデックスを格納)
                          int ind_handle,          // iAO 指標ハンドル
                          int amount               // 複製された値の数
                         )
  {
//--- エラーコードをリセットする
   ResetLastError();
//---インデックス0 を持つ指標バッファの値で iAOBuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
      {
       //--- 複製が失敗したら、エラーコードを出す
       PrintFormat("Failed to copy data from the iAO indicator, error code %
d",GetLastError());
       //--- ゼロ結果で終了。 指標は計算されていないと見なされる
       return(false);
      }
//--- 色インデックスを複製する
   if(CopyBuffer(ind_handle,1,0,amount,color_indexes)<0)
      {
       //--- 複製が失敗したら、エラーコードを出す
       PrintFormat("Failed to copy color values from the iAO indicator, error code %
d",GetLastError());
       //--- ゼロ結果で終了。 指標は計算されていないと見なされる
       return(false);
      }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指標初期化解除関数                                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iATR

この関数は ATR(Average True Range) 指標ハンドルを返します。バッファは 1 つです。

```
int  iATR(
   string           symbol,         // 銘柄名
   ENUM_TIMEFRAMES  period,         // 期間
   int              ma_period       // 平均期間
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名 (NULL は現在のシンボル)

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

ma_period

[in] 指標計算の平均期間の値

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって、コンピュータメモリを利用されていない指標から解放することが出来ます。

例:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iATR.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iATR technical indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the
'type' parameter (function type)."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- i ATR をプロットする
#property indicator_label1  "iATR"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
```

© 2000–2025, MetaQuotes Ltd.

```
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iATR,// ｉATR を使用する
   Call_IndicatorCreate     // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input int                  atr_period=14;          // 計算期間
input Creation             type=Call_iATR;         // 関数の種類
input string               symbol=" ";             // シンボル
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // 時間軸
//--- 指標バッファ
double          iATRBuffer[];
//--- ｉAC 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- ATR 指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,iATRBuffer,INDICATOR_DATA);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iATR)
      handle=iATR(name,period,atr_period);
   else
     {
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[1];
```

© 2000-2025, MetaQuotes Ltd.

```
      pars[0].type=TYPE_INT;
      pars[0].integer_value=atr_period;
      handle=IndicatorCreate(name,period,IND_ATR,1,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iATR indicator for the symbol %s/%
s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- ATR 指標が計算された銘柄/時間軸を表示
   short_name=StringFormat("iATR(%s/%s, period=%
d)",name,EnumToString(period),atr_period);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iATR 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
//--- これが指標計算の初めであるか iATR 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
```

© 2000-2025, MetaQuotes Ltd.

```
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //---もし iATRBuffer 配列が銘柄/期間で iATR 指標の値の数より大きい場合、全体のコピーはしない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- これは初回の計算ではなく、
      //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iATRBuffer 配列をATR 指標の値で記入
//--- FillArrayFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArrayFromBuffer(iATRBuffer,handle,values_to_copy)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- ATR 指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iATR 指標から指標バッファを記入する                                |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // ATR 値の指標バッファ
                         int ind_handle,    // iATR 指標ハンドル
                         int amount         // 複製された値の数
                        )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で iATRBuffer 配列の一部を記入するx
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iATR indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
```

© 2000-2025, MetaQuotes Ltd.

```
    return(true);
   }
//+------------------------------------------------------------------+
//|  指標初期化解除関数                                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
   {
    if(handle!=INVALID_HANDLE)
       IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
    Comment("");
   }
```

© 2000-2025, MetaQuotes Ltd.

iBearsPower

この関数は Bears Power（ベアパワー）指標のハンドルを返します。バッファは 1 つです。

```
int  iBearsPower(
   string               symbol,           // 銘柄名
   ENUM_TIMEFRAMES      period,           // 期間
   int                  ma_period,        // 平均期間
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

ma_period

[in] 指標計算の平均期間の値

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって、コンピュータメモリを利用されていない指標から解放することが出来ます。

例：

```
//+------------------------------------------------------------------+
//|                                                Demo_iBearsPower.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iBearsPower technical indicator."
#property description "A symbol and timeframe used for calculation of the indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the 'type' parameter (function type)."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iBearsPower プロット
#property indicator_label1  "iBearsPower"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
```

© 2000–2025, MetaQuotes Ltd.

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBearsPower,        // iBearsPower を使用する
   Call_IndicatorCreate     // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iBearsPower;  // 関数の種類
input int                   ma_period=13;           // 移動平均の期間
input string                symbol=" ";             // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // 時間軸
//--- 指標バッファ
double          iBearsPowerBuffer[];
//--- iBearsPower 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- ベアパワー指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,iBearsPowerBuffer,INDICATOR_DATA);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iBearsPower)
      handle=iBearsPower(name,period,ma_period);
   else
     {
      //--- 構造体を指標のパラメータで記入
```

© 2000-2025, MetaQuotes Ltd.

```
      MqlParam pars[1];
      //--- ma 期間
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_BEARS,1,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iBearsPower indicator for the
symbol %s/%s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- ベアパワー指標が計算された銘柄/時間軸を表示
   short_name=StringFormat("iBearsPower(%s/%s, period=%
d)",name,EnumToString(period),ma_period);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iBearsPower 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- これが指標計算の初めであるか ｉBearsPower 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- もし ｉBearsPowerBuffer 配列が銘柄/期間で ｉBearsPower 指標の値の数より大きい場合、全
体のコピーはしない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- これは初回の計算ではなく、
      //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- ｉBearsPowerBuffer 配列をベアパワー指標の値で記入
//--- FillArrayFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArrayFromBuffer(iBearsPowerBuffer,handle,values_to_copy)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- ベアパワー指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| ｉBearsPower 指標から指標バッファを記入する                      |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // ベアパワー値の指標バッファ
                         int ind_handle,    // ｉBearsPower 指標ハンドル
                         int amount         // 複製された値の数
                        )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で ｉBearsPowerBuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iBearsPower indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指標初期化解除関数                                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000–2025, MetaQuotes Ltd.

iBands

この関数は Bollinger Bands®（ボリンジャーバンド®）指標ハンドルを返します。

```
int  iBands(
   string               symbol,           // 銘柄名
   ENUM_TIMEFRAMES      period,           // 期間
   int                  bands_period,     // 平均線の計算の期間
   int                  bands_shift,      // 指標の水平シフト
   double               deviation,        // 標準偏差の数
   ENUM_APPLIED_PRICE   applied_price     // 価格の種類かハンドル
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

bands_period

[in] 指標の正中線の平均期間

bands_shift

[in] 価格チャートに相対した指標のシフト

偏差

[in] 正中線からの逸脱

applied_price

[in] 使用される価格。ENUM_APPLIED_PRICE 価格定数のいずれか、または別の指標ハンドル。

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって、コンピュータメモリを利用されていない指標から解放することが出来ます。

注意事項

バッファ番号は 0 - BASE_LINE、1 - UPPER_BAND、2 - LOWER_BAND です。

例：

```
//+------------------------------------------------------------------+
//|                                                   Demo_iBands.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
```

© 2000–2025, MetaQuotes Ltd.

```
#property description "of indicator buffers for the iBands technical indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the
'type' parameter (function type)."

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   3
//--- 上のプロット
#property indicator_label1  "Upper"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrMediumSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 下のプロット
#property indicator_label2  "Lower"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrMediumSeaGreen
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- 真ん中のプロット
#property indicator_label3  "Middle"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrMediumSeaGreen
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBands,             // iBands を使用する
   Call_IndicatorCreate     // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iBands;           // 関数の種類
input int                   bands_period=20;            // 移動平均の期間
input int                   bands_shift=0;              // シフト
input double                deviation=2.0;              // 標準偏差の数
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // 価格の種類
input string                symbol=" ";                 // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;      // 時間軸
//--- 指標バッファ
double          UpperBuffer[];
double          LowerBuffer[];
double          MiddleBuffer[];
//--- iBands 指標ハンドルを格納する変数
```

© 2000-2025, MetaQuotes Ltd.

```
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- ボリンジャーバンド指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,UpperBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,LowerBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,MiddleBuffer,INDICATOR_DATA);
//--- それぞれの線のシフトを設定
   PlotIndexSetInteger(0,PLOT_SHIFT,bands_shift);
   PlotIndexSetInteger(1,PLOT_SHIFT,bands_shift);
   PlotIndexSetInteger(2,PLOT_SHIFT,bands_shift);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iBands)
      handle=iBands(name,period,bands_period,bands_shift,deviation,applied_price);
   else
     {
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[4];
      //--- ma の期間
      pars[0].type=TYPE_INT;
      pars[0].integer_value=bands_period;
      //--- シフト
      pars[1].type=TYPE_INT;
      pars[1].integer_value=bands_shift;
      //--- 標準偏差の数
      pars[2].type=TYPE_DOUBLE;
      pars[2].double_value=deviation;
      //--- 価格の種類
      pars[3].type=TYPE_INT;
```

© 2000-2025, MetaQuotes Ltd.

```
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_BANDS,4,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iBands indicator for the symbol %
s/%s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- ボリンジャーバンド指標が計算された銘柄/時間軸を表示
   short_name=StringFormat("iBands(%s/%s, %d,%d,%G,%s)",name,EnumToString(period),
                           bands_period,bands_shift,deviation,EnumToString(applied_p
rice));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iBands 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
//--- これが指標計算の初めであるか iBands 指標の値の数が変更した changed
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
```

© 2000-2025, MetaQuotes Ltd.

```
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- もし iBands 配列が銘柄/期間で iAC 指標の値の数より大きい場合、全体のコピーはしない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- これは初回の計算ではなく、
      //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- 配列をボリンジャーバンド指標の値で記入
//--- FillArraysFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArraysFromBuffers(MiddleBuffer,UpperBuffer,LowerBuffer,bands_shift,handle
,values_to_copy)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- ボリンジャーバンド指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iBands 指標から指標バッファを記入する                                |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &base_values[],     // ボリンジャーバンドの中線の指標バッファ
                           double &upper_values[],    // 上縁の指標バッファ
                           double &lower_values[],    // 下縁の指標バッファ
                           int shift,                 // シフト
                           int ind_handle,            // iBands 指標ハンドル
                           int amount                 // 複製された値の数
                          )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で MiddleBuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,-shift,amount,base_values)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iBands indicator, error code %
d",GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }

//--- インデックス 1 を持つ指標バッファの値で Upper Buffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,1,-shift,amount,upper_values)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iBands indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }

//--- インデックス 2 を持つ指標バッファの値で Lower Buffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,2,-shift,amount,lower_values)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iBands indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指標初期化解除関数                                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iBullsPower

この関数は Bulls Power（ブルパワー）指標ハンドルを返します。バッファは 1 つです。

```
int  iBullsPower(
   string               symbol,            // 銘柄名
   ENUM_TIMEFRAMES      period,            // 期間
   int                  ma_period,         // 平均期間
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

ma_period

[in] 指標計算の平均期間

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって 、コンピュータメモリを利用されていない指標から解放することが出来ます。

例:

```
//+------------------------------------------------------------------+
//|                                              Demo_iBullsPower.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iBullsPower technical indicator."
#property description "A symbol and timeframe used for calculation of the indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the 'type' parameter (function type)."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- ブルパワープロット
#property indicator_label1  "iBullsPower"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBullsPower,        // iBullsPower を使用する
   Call_IndicatorCreate     // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iBullsPower;  // 関数の種類
input int                   ma_period=13;           // 移動平均の期間
input string                symbol=" ";             // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // 時間軸
//--- 指標バッファ
double          iBullsPowerBuffer[];
//--- iBullsPower 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- ブルパワー指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,iBullsPowerBuffer,INDICATOR_DATA);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iBullsPower)
      handle=iBullsPower(name,period,ma_period);
   else
     {
      //--- 構造体を指標のパラメータで記入
```

© 2000-2025, MetaQuotes Ltd.

```
      MqlParam pars[1];
      //--- ma の期間
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_BULLS,1,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iBullsPower indicator for the
symbol %s/%s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- ブルパワー指標が計算された銘柄/時間軸を表示
   short_name=StringFormat("iBullsPower(%s/%s, period=%
d)",name,EnumToString(period),ma_period);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iBullsPower 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- これが指標計算の初めであるか iBullsPower 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- もし iBullsPowerBuffer 配列が銘柄/期間で iBullsPower 指標の値の数より大きい場合、全
体のコピーはしない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- これは初回の計算ではなく、
      //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iBullsPowerBuffer 配列をブルパワー指標の値で記入
//--- FillArrayFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArrayFromBuffer(iBullsPowerBuffer,handle,values_to_copy)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- ブルパワー指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iBullsPower 指標から指標バッファを記入する                       |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // ブルパワー値の指標バッファ
                         int ind_handle,    // iBullsPower 指標ハンドル
                         int amount         // 複製された値の数
                        )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で iBullsPowerBuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iBullsPower indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指標初期化解除関数                                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
//+------------------------------------------------------------------+
```

© 2000–2025, MetaQuotes Ltd.

iCCI

この関数は CCI（Commodity Channel Index、商品チャンネル指数）指標ハンドルを返します。バッファは 1 つです。

```
int  iCCI(
   string               symbol,            // 銘柄名
   ENUM_TIMEFRAMES      period,            // 期間
   int                  ma_period,         // 平均期間
   ENUM_APPLIED_PRICE   applied_price      // 価格の種類かハンドル
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

ma_period

[in] 指標計算の平均期間

applied_price

[in] 使用される価格。ENUM_APPLIED_PRICE 価格定数のいずれか、または別の指標ハンドル。

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって、コンピュータメモリを利用されていない指標から解放することが出来ます。

例：

```
//+------------------------------------------------------------------+
//|                                                   Demo_iCCI.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iCCI technical indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the
'type' parameter (function type)."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
```

© 2000–2025, MetaQuotes Ltd.

```
//--- ｉCCI プロット
#property indicator_label1  "iCCI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 指標ウィンドウの水平レベル
#property indicator_level1  -100.0
#property indicator_level2  100.0
//+------------------------------------------------------------------+
//| 作成を処理する方法の列挙                                         |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iCCI,              // ｉCCI を使用する
   Call_IndicatorCreate    // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iCCI;               // 関数の種類
input int                   ma_period=14;                 // 移動平均の期間
input ENUM_APPLIED_PRICE    applied_price=PRICE_TYPICAL;  // 価格の種類
input string                symbol=" ";                   // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // 時間軸
//--- 指標バッファ
double          iCCIBuffer[];
//--- ｉCCI 指標ハンドルを格納する変数r
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- CCI 指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,iCCIBuffer,INDICATOR_DATA);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
```

© 2000-2025, MetaQuotes Ltd.

```
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iCCI)
      handle=iCCI(name,period,ma_period,applied_price);
   else
     {
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[2];
      //--- 移動平均の期間
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- 価格の種類
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_CCI,2,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iCCI indicator for the symbol %s/%
s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- CCI  指標が計算された銘柄/時間軸を表示
   short_name=StringFormat("iCCI(%s/%s, %d, %s)",name,EnumToString(period),
                           ma_period,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- iCCI 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
//--- これが指標計算の初めであるかiCCI 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- もしiCCIBuffer 配列が銘柄/期間でiCCI 指標の値の数より大きい場合、全体のコピーはしない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- これは初回の計算ではなく、
      //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iCCIBuffer 配列を CCI 指標の値で記入
//--- FillArrayFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArrayFromBuffer(iCCIBuffer,handle,values_to_copy)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- CCI 指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iCCI 指標から指標バッファを記入する                                |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],   // CCI 値の指標バッファ
                         int ind_handle,     // iCCI 指標ハンドル
                         int amount          // 複製された値の数
                        )
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で i CCI Buffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iCCI indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指標初期化解除関数                                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000–2025, MetaQuotes Ltd.

iChaikin

この関数は Chaikin Oscillator（チャイキンオシレーター）指標のハンドルを返します。バッファは 1 つです。

```
int  iChaikin(
   string               symbol,              // 銘柄名
   ENUM_TIMEFRAMES      period,              // 期間
   int                  fast_ma_period,      // 高速期間
   int                  slow_ma_period,      // 低速期間
   ENUM_MA_METHOD       ma_method,           // 平滑化の種類
   ENUM_APPLIED_VOLUME  applied_volume       // ボリュームの種類
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

fast_ma_period

[in] 計算の高速平均期間

slow_ma_period

[in] 計算の低速平均期間

ma_method

[in] 平滑化の種類（ENUM_MA_METHOD 定数のいずれか）

applied_volume

[in] 使用されるボリューム（ENUM_APPLIED_VOLUME 定数のいずれか）

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって、コンピュータメモリを利用されていない指標から解放することが出来ます。

例：

```
//+------------------------------------------------------------------+
//|                                                 Demo_iChaikin.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iChaikin technical indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the
'type' parameter (function type)."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iChaikin プロット
#property indicator_label1  "iChaikin"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                          |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iChaikin,           // iChaikinを使用する
   Call_IndicatorCreate     // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iChaikin;           // 関数の種類
input int                   fast_ma_period=3;             // 高速 MA 期間
input int                   slow_ma_period=10;            // 低速 MA 期間
input ENUM_MA_METHOD        ma_method=MODE_EMA;           // 平滑化の種類
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;   // ボリュームの種類
input string                symbol=" ";                   // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // 時間軸
//--- 指標バッファ
double          iChaikinBuffer[];
//--- iChaikin 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- チャイキンオシレーター 指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                      |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,iChaikinBuffer,INDICATOR_DATA);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
```

© 2000-2025, MetaQuotes Ltd.

```
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iChaikin)
      handle=iChaikin(name,period,fast_ma_period,slow_ma_period,ma_method,applied_vo
lume);
   else
     {
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[4];
      //--- 高速 MA 期間
      pars[0].type=TYPE_INT;
      pars[0].integer_value=fast_ma_period;
      //--- 低速 MA 期間
      pars[1].type=TYPE_INT;
      pars[1].integer_value=slow_ma_period;
      //--- 平滑化の種類
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- ボリュームの種類
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_CHAIKIN,4,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iChaikin indicator for the symbol
%s/%s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- チャイキンオシレーター 指標が計算された銘柄/時間軸を表示
   short_name=StringFormat("iChaikin(%s/%s, %d, %d, %s, %
s)",name,EnumToString(period),
                           fast_ma_period,slow_ma_period,
                           EnumToString(ma_method),EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
```

© 2000-2025, MetaQuotes Ltd.

```
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iChaikin 指標から複製された値の数r
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
//--- これが指標計算の初めであるか iChaikin 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- もし iChaikinBuffer 配列が銘柄/期間でi Chaikin 指標の値の数より大きい場合、全体のコピーはしない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- これは初回の計算ではなく、
      //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iChaikinBuffer 配列を チャイキンオシレーター 指標の値で記入
//--- FillArrayFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArrayFromBuffer(iChaikinBuffer,handle,values_to_copy)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
```

© 2000-2025, MetaQuotes Ltd.

```
                                short_name,
                                values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- チャイキンオシレーター 指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iChaikin 指標から指標バッファを記入する                          |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // チャイキンオシレーター 値の指標バッファ
                         int ind_handle,    // iChaikin 指標ハンドル
                         int amount         // 複製された値の数
                        )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で iChaikinBuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iChaikin indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指標初期化解除関数                                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iCustom

この関数は指定されたカスタム指標ハンドルを返します。

```
int  iCustom(
   string            symbol,      // 銘柄名
   ENUM_TIMEFRAMES   period,      // 期間
   string            name         // フォルダ/カスタム指標名
   ...                            // 指標入力パラメータのリスト
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

name

[in] カスタム指標名の前にバックスラッシュ(¥)が示されている場合、EXL指標ファイルはMQL5ルートフォルダーに対して相対的に検索されます。例えば、iCustom(Symbol(), Period(), "\FirstIndicator"...)が呼び出された場合、指標はMQL5¥FirstIndicator.ex5として読み込まれます。このパスでファイルが見つからない場合、エラー4802 (ERR_INDICATOR_CANNOT_CREATE)が返されます。

パスが「¥」で始まらない場合、指標は次のように検索およびダウンロードされます。

- まず、指標EX5ファイルが、呼び出しプログラムのEX5ファイルが置かれているフォルダーで検索されます。例えば、CrossMA.EX5 EAがMQL5¥Experts¥MyExpertsにあって、iCustom呼び出し(Symbol(), Period(), "SecondIndicator"...)が含まれるとすると、指標はMQL5¥Experts¥MyExperts¥SecondIndicator.ex5で検索されます。
- 同じディレクトリで指標がが見つからない場合、MQL5¥Indicators指標ルートディレクトリに早退した検索が実行されます。つまり、MQL5¥Indicators¥SecondIndicator.ex5ファイルの検索が実行されます。それでも指標が見つからない場合、関数はINVALID_HANDLEを返してエラー4802 (ERR_INDICATOR_CANNOT_CREATE)が発生します。

指標へのパスがサブディレクトリ(例: MyIndicators¥ThirdIndicator)に設定されている場合、検索は最初に呼び出し側プログラムフォルダーMQL5¥Experts¥MyExperts¥MyIndicators¥ThirdIndicator.ex5で実行されます(EAはMQL5¥Experts¥MyExpertsにある)。失敗した場合、MQL5¥Indicators¥MyIndicators¥ThirdIndicator.ex5ファイルの検索が実行されます。呼び出しでは、パスの区切り文字として必ず二重逆スラッシュ「¥¥」を使用してください(例: iCustom(Symbol(), Period(), "MyIndicators\\ThirdIndicator"...))

...

[in] コンマで区切られたカスタム指標の入力パラメータパラメータの種類と順序が一致している必要があります。パラメータの指定がない場合は初期値が使用されます。

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって、コンピュータメモリを利用されていない指標から解放することが出来ます。

注意事項

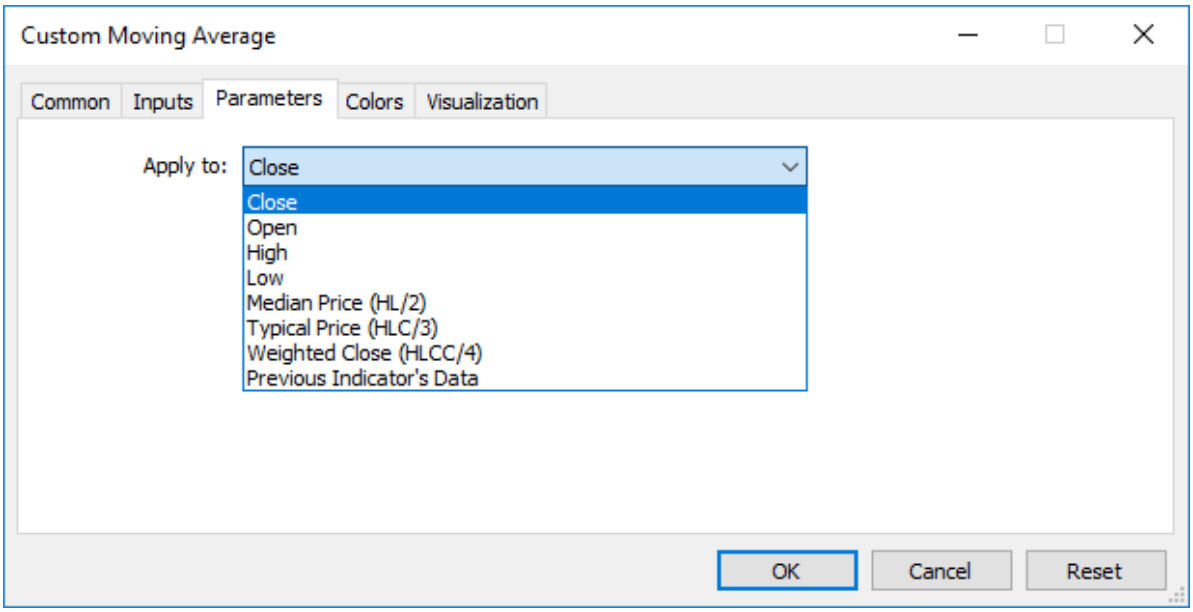
© 2000-2025, MetaQuotes Ltd.

カスタム指標はコンパイルされて（EX5 拡張子）クライアント端末の MQL5/Indicators ディレクトリかそのサブディレクトリに置かれなければなりません。

テストが必要な指標は、パラメータが文字列定数で設定されていれば、iCustom() 関数の呼び出しから自動的に定義されています。他の全ての場合（IndicatorCreate() 関数の使用や指標名を設定するパラメータでの非定数文字列の使用）#property tester_indicator プロパティが必要です。

```
#property tester_indicator "indicator_name.ex5"
```

指標が最初の呼び出し形式を使用する場合、カスタム指標を使い始める時に、その「パラメータ」タブで、計算のためのデータを指定することが出来ます。「適用価格」パラメータが明確に選択されていない場合、デフォルト計算は終値に基づきます。



MQL5 プログラムからカスタム指標が呼ばれる場合、Applied_Price パラメータまたは別の指標ハンドルは、カスタム指標の他の入力パラメータ全ての後で受け渡しされるべきです。

参照

プログラムプロパティ、時系列と指標へのアクセス、IndicatorCreate()、IndicatorRelease()

例:

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- Label 1 をプロットする
#property indicator_label1  "Label1"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 入力パラメータ
input int MA_Period=21;
input int MA_Shift=0;
input ENUM_MA_METHOD MA_Method=MODE_SMA;
//--- 指標バッファ
```

© 2000–2025, MetaQuotes Ltd.

```
double         Label1Buffer[];
//--- Custom Moving Average.mq5 カスタム指標ハンドル
int MA_handle;
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 指標バッファマッピング
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
   ResetLastError();
   MA_handle=iCustom(NULL,0,"Examples\\Custom Moving Average",
                     MA_Period,
                     MA_Shift,
                     MA_Method,
                     PRICE_CLOSE // 終値の使用
                     );
   Print("MA_handle = ",MA_handle,"  error = ",GetLastError());
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- 指標 Custom Moving Average の値を指標バッファに複製する
   int copy=CopyBuffer(MA_handle,0,0,rates_total,Label1Buffer);
   Print("copy = ",copy,"    rates_total = ",rates_total);
//--- I 試みが失敗したら報告する
   if(copy<=0)
      Print("An attempt to get the values if Custom Moving Average has failed");
//--- 次の呼び出しのために prev_calculated の値を返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

iDEMA

この関数は Double Exponential Moving Average（2 重指数移動平均）指標のハンドルを返します。バッファは 1 つです。

```
int  iDEMA(
   string               symbol,            // 銘柄名
   ENUM_TIMEFRAMES      period,            // 期間
   int                  ma_period,         // 平均期間
   int                  ma_shift,          // 水平シフト
   ENUM_APPLIED_PRICE   applied_price      // 価格の種類かハンドル
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

ma_period

[in] 計算の平均期間（バーの数）

ma_shift

[in] 価格チャートに相対した指標のシフト

applied_price

[in] 使用される価格。ENUM_APPLIED_PRICE 価格定数のいずれか、または別の指標ハンドル。

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって、コンピュータメモリを利用されていない指標から解放することが出来ます。

例:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iDEMA.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iDEMA technical indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the
'type' parameter (function type)."
```

© 2000–2025, MetaQuotes Ltd.

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iDEMA プロット
#property indicator_label1  "iDEMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iDEMA,              // iDEMA を使用する
   Call_IndicatorCreate     // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iDEMA;            // 関数の種類
input int                   ma_period=14;               // 移動平均の期間
input int                   ma_shift=0;                 // シフト
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // 価格の種類
input string                symbol=" ";                 // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;      // 時間軸
//--- 指標バッファ
double          iDEMABuffer[];
//--- iDEMA 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- 2 重指数移動平均指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,iDEMABuffer,INDICATOR_DATA);
   //--- シフトを設定
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
```

© 2000-2025, MetaQuotes Ltd.

```
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iDEMA)
      handle=iDEMA(name,period,ma_period,ma_shift,applied_price);
   else
     {
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[3];
      //--- 移動平均の期間
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- シフト
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- 価格の種類
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_DEMA,3,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iDEMA indicator for the symbol %s/
%s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- 2 重指数移動平均指標が計算された銘柄/時間軸を表示
   short_name=StringFormat("iDEMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iDEMA 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
//--- これが指標計算の初めであるか iDEMA 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合(価格履歴で何かが変更された)
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- もし iDEMABuffer 配列が銘柄/期間で iDEMA 指標の値の数より大きい場合、全体のコピーはしない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- これは初回の計算ではなく、
      //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iDEMABuffer 配列を oubl e Exponential Moving Average 指標の値で記入
//--- FillArrayFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArrayFromBuffer(iDEMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- 2 重指数移動平均指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  i DEMA 指標から指標バッファを記入する                                |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // 2 重指数移動平均 値の指標バッファ
                         int shift,         // シフト
                         int ind_handle,    // i DEMA 指標ハンドル
                         int amount         // 複製された値の数
                        )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で i DEMABuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iDEMA indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  指標初期化解除関数                                                |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iDeMarker

この関数は DeMarker（デマーカー）指標のハンドルを返します。バッファは 1 つです。

```
int  iDeMarker(
   string           symbol,        // 銘柄名
   ENUM_TIMEFRAMES  period,        // 期間
   int              ma_period      // 平均期間
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

ma_period

[in] 計算の平均期間（バーの数）

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって、コンピュータメモリを利用されていない指標から解放することが出来ます。

例：

```
//+------------------------------------------------------------------+
//|                                                Demo_iDeMarker.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iDeMarker technical indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the
'type' parameter (function type)."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iDeMarker プロット
#property indicator_label1  "iDeMarker"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
```

© 2000–2025, MetaQuotes Ltd.

```
#property indicator_width1  1
//--- 指標ウィンドウの水平レベル
#property indicator_level1  0.3
#property indicator_level2  0.7
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iDeMarker,          // iDeMarker を使用する
   Call_IndicatorCreate     // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iDeMarker;        // 関数の種類
input int                   ma_period=14;               // 移動平均の期間
input string                symbol=" ";                 // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;      // 時間軸
//--- 指標バッファ
double          iDeMarkerBuffer[];
//--- iDeMarker 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- DeMarker 指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,iDeMarkerBuffer,INDICATOR_DATA);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iDeMarker)
      handle=iDeMarker(name,period,ma_period);
   else
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- 構造体を指標のパラメータで記入
       MqlParam pars[1];
       //--- 移動平均の期間
       pars[0].type=TYPE_INT;
       pars[0].integer_value=ma_period;
       handle=IndicatorCreate(name,period,IND_DEMARKER,1,pars);
      }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
      {
       //--- 失敗した事実とエラーコードを出力する
       PrintFormat("Failed to create handle of the iDeMarker indicator for the symbol
%s/%s, error code %d",
                   name,
                   EnumToString(period),
                   GetLastError());
       //--- 指標が早期に中止された
       return(INIT_FAILED);
      }
//--- DeMarker 指標が計算された銘柄/時間軸を表示
   short_name=StringFormat("iDeMarker(%s/%s, period=%
d)",name,EnumToString(period),ma_period);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iDeMarker 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
      {
       PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
      return(0);
     }
//--- これが指標計算の初めであるか iDeMarker 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合(価格履歴で何かが変更された)
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- もし iDeMarkerBuffer 配列が銘柄/期間で iDeMarker 指標の値の数より大きい場合、全体のコピーはしない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- これは初回の計算ではなく、
      //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iDeMarkerBuffer 配列を DeMarker 指標の値で記入
//--- FillArrayFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArrayFromBuffer(iDeMarkerBuffer,handle,values_to_copy)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- DeMarker 指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iDeMarker 指標から指標バッファを記入する                         |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],   // DeMarker 値の指標バッファ
                         int ind_handle,     // iDeMarker 指標ハンドル
                         int amount          // 複製された値の数
                        )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で iDeMarkerBuffer 配列の一部を記入するx
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("Failed to copy data from the iDeMarker indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指標初期化解除関数                                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000–2025, MetaQuotes Ltd.

iEnvelopes

この関数は Envelopes（エンベロープ）指標のハンドルを返します。

```
int  iEnvelopes(
   string               symbol,           // 銘柄名
   ENUM_TIMEFRAMES      period,           // 期間
   int                  ma_period,        // 平均線計算の期間
   int                  ma_shift,         // 指標の水平シフト
   ENUM_MA_METHOD       ma_method,        // 平滑化の種類
   ENUM_APPLIED_PRICE   applied_price,    // 価格の種類かハンドル
   double               deviation         // 百分率での正中線から境界線のズレ
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

ma_period

[in] 正中線の平均期間

ma_shift

[in] 価格チャートに相対した指標のシフト

ma_method

[in] 平滑化の種類ENUM_MA_METHOD のいずれかの値

applied_price

[in] 使用される価格。ENUM_APPLIED_PRICE 価格定数のいずれか、または別の指標ハンドル。

偏差

[in] 百分率での正中線からのズレ

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって 、コンピュータメモリを利用されていない指標から解放することが出来ます。

注意事項

バッファ番号は 0 - UPPER_LINE, 1 - LOWER_LINE です。

例：

```
//+------------------------------------------------------------------+
//|                                                Demo_iEnvelopes.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iEnvelopes technical indicator."
#property description "A symbol and timeframe used for calculation of the indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the 'type' parameter (function type)."

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//--- 上のプロット
#property indicator_label1  "Upper"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 下のプロット
#property indicator_label2  "Lower"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iEnvelopes,         // iEnvelopes を使用する
   Call_IndicatorCreate     // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation             type=Call_iEnvelopes;       // 関数の種類
input int                  ma_period=14;               // 移動平均の期間
input int                  ma_shift=0;                 // シフト
input ENUM_MA_METHOD       ma_method=MODE_SMA;         // 平滑化の種類
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // 価格の種類
input double               deviation=0.1;              // 堺の移動平均からの偏差
input string               symbol=" ";                 // シンボル
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;      // 時間軸
//--- 指標バッファ
double          UpperBuffer[];
double          LowerBuffer[];
//--- iEnvelopes 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
```

© 2000-2025, MetaQuotes Ltd.

```
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- Envelopes 指標に値の数を保存
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,UpperBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,LowerBuffer,INDICATOR_DATA);
//--- それぞれの線のシフトを設定
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
   PlotIndexSetInteger(1,PLOT_SHIFT,ma_shift);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iEnvelopes)
      handle=iEnvelopes(name,period,ma_period,ma_shift,ma_method,applied_price,deviation);
   else
     {
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[5];
      //--- 移動平均の期間
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- シフト
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- 平滑化の種類
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- 価格の種類
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      //--- 価格の種類
      pars[4].type=TYPE_DOUBLE;
```

© 2000-2025, MetaQuotes Ltd.

```
      pars[4].double_value=deviation;
      handle=IndicatorCreate(name,period,IND_ENVELOPES,5,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iEnvelopes indicator for the
symbol %s/%s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- Envelopes 指標が計算された銘柄/時間軸を表示
   short_name=StringFormat("iEnvelopes(%s/%s, %d, %d, %s,%s, %
G)",name,EnumToString(period),
   ma_period,ma_shift,EnumToString(ma_method),EnumToString(applied_price),deviation)
;
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iEnvelopes 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
//--- これが指標計算の初めであるか iEnvelopes 指標の値の数が変更した
```

© 2000-2025, MetaQuotes Ltd.

```
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- もし UpperBuffer 配列が銘柄/期間で iEnvelopes 指標の値の数より大きい場合、全体のコピー
はしない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- これは初回の計算ではなく、
      //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- エンベロープ指標の値で UpperBuffer 及び LowerBuffer 配列を記入する
//--- FillArrayFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArraysFromBuffers(UpperBuffer,LowerBuffer,ma_shift,handle,values_to_copy)
) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- エンベロープ指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iEnvelopes 指標から指標バッファを記入する                        |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &upper_values[],     // 上縁の指標バッファ
                           double &lower_values[],     // 下縁の指標バッファ
                           int shift,                  // シフト
                           int ind_handle,             // iEnvelopes 指標ハンドル
                           int amount                  // 複製された値の数
                          )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で UpperBuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,-shift,amount,upper_values)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("Failed to copy data from the iEnvelopes indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- インデックス 1 を持つ指標バッファの値で Lower Buffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,1,-shift,amount,lower_values)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iEnvelopes indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指標初期化解除関数                                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iForce

この関数は Force Index（勢力指数）指標のハンドルを返します。バッファは 1 つです。

```
int  iForce(
   string               symbol,           // 銘柄名
   ENUM_TIMEFRAMES      period,           // 期間
   int                  ma_period,        // 平均期間
   ENUM_MA_METHOD       ma_method,        // 平滑化の種類
   ENUM_APPLIED_VOLUME  applied_volume    // 計算に使用するボリュームの種類
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

ma_period

[in] 指標計算の平均期間

ma_method

[in] 平滑化の種類ENUM_MA_METHOD のいずれかの値

applied_volume

[in] 使用されるボリューム（ENUM_APPLIED_VOLUME のいずれかの値）

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって、コンピュータメモリを利用されていない指標から解放することが出来ます。

例：

```
//+------------------------------------------------------------------+
//|                                                  Demo_iForce.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iForce technical indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the
'type' parameter (function type)."
```

© 2000–2025, MetaQuotes Ltd.

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iForce の描画
#property indicator_label1  "iForce"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| 作成を処理する方法の列挙                                         |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iForce,             // iForce を使用する
   Call_IndicatorCreate     // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iForce;             // 関数の種類
input int                   ma_period=13;                 // 平均期間
input ENUM_MA_METHOD        ma_method=MODE_SMA;           // 平滑化の種類
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;   // ボリュームの種類
input string                symbol=" ";                   // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // 時間軸
//--- 指標バッファ
double          iForceBuffer[];
//--- iForce 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- フォース指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,iForceBuffer,INDICATOR_DATA);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iForce)
      handle=iForce(name,period,ma_period,ma_method,applied_volume);
   else
     {
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[3];
      //--- 移動平均の期間
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- 平滑化の種類
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_method;
      //--- ボリュームの種類
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_volume;
      //--- 価格の種類
      handle=IndicatorCreate(name,period,IND_FORCE,3,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iForce indicator for the symbol %
s/%s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- フォース指標が計算された銘柄/時間軸を表示
   short_name=StringFormat("iForce(%s/%s, %d, %s, %s)",name,EnumToString(period),
                           ma_period,EnumToString(ma_method),EnumToString(applied_vo
lume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//|  カスタム指標の反復関数                                          |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iForce 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
//--- これが指標計算の初めであるか iForce 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- もし iForceBuffer 配列が銘柄/期間で iForce 指標の値の数より大きい場合、全体のコピーはし
ない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- これは初回の計算ではなく、
      //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iForceBuffer 配列をフォース指標の値で記入
//--- FillArrayFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArrayFromBuffer(iForceBuffer,handle,values_to_copy)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- フォース指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  iForce 指標から指標バッファを記入する                           |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],   // フォース指標バッファ values
                         int ind_handle,     // iForce 指標ハンドル
                         int amount          // 複製された値の数
                        )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で iForceBuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iForce indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  指標初期化解除関数                                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iFractals

この関数は Fractals（フラクタル）指標ハンドルを返します。

```
int  iFractals(
   string           symbol,     // 銘柄名
   ENUM_TIMEFRAMES  period      // 期間
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって、コンピュータメモリを利用されていない指標から解放することが出来ます。

注意事項

バッファ番号は 0 – UPPER_LINE, 1 – LOWER_LINE です。

例：

```
//+------------------------------------------------------------------+
//|                                                Demo_iFractals.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iFractals technical indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the
'type' parameter (function type)."

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//--- Fractal Up プロット
#property indicator_label1  "FractalUp"
```

© 2000–2025, MetaQuotes Ltd.

```
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrBlue
//--- Fractal Down プロット
#property indicator_label2  "FractalDown"
#property indicator_type2   DRAW_ARROW
#property indicator_color2  clrRed
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iFractals,          // iFractals を使用する
   Call_IndicatorCreate     // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iFractals;          // 関数の種類
input string                symbol=" ";                   // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // 時間軸
//--- 指標バッファ
double          FractalUpBuffer[];
double          FractalDownBuffer[];
//--- iFractals 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- フラクタル指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,FractalUpBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,FractalDownBuffer,INDICATOR_DATA);
//--- PLOT_ARROWプロパティのコードを Wingdings 文字セットからの記号を使用して設定
   PlotIndexSetInteger(0,PLOT_ARROW,217); // 上矢印
   PlotIndexSetInteger(1,PLOT_ARROW,218); // 下矢印
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
```

© 2000-2025, MetaQuotes Ltd.

```
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iFractals)
      handle=iFractals(name,period);
   else
      handle=IndicatorCreate(name,period,IND_FRACTALS);
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iFractals indicator for the symbol %s/%s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- フラクタル指標が計算された銘柄/時間軸を表示
   short_name=StringFormat("iFractals(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iFractals 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastError());
      return(0);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- これが指標計算の初めであるか iFractals 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- もし FractalUpBuffer 配列が銘柄/期間で iFractals指標の値の数より大きい場合、全体のコピーはしない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- これは初回の計算ではなく、
      //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- フラクタル指標の値で FractalUpBuffer 及び FractalDownBuffer 配列を記入する
//--- FillArrayFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArraysFromBuffers(FractalUpBuffer,FractalDownBuffer,handle,values_to_copy
)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- フラクタル指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iFractals 指標から指標バッファを記入する                         |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &up_arrows[],        // 上矢印の指標バッファ
                           double &down_arrows[],      // 上矢印の指標バッファ
                           int ind_handle,             // iFractals 指標ハンドル
                           int amount                  // 複製された値の数
                          )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で FractalUpBuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,0,amount,up_arrows)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("Failed to copy data from the iFractals indicator to the
FractalUpBuffer array, error code %d",
                  GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- インデックス 1 を持つ指標バッファの値で Fractal DownBuffer 配列の一部を記入する1
   if(CopyBuffer(ind_handle,1,0,amount,down_arrows)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iFractals indicator to the
FractalDownBuffer array, error code %d",
                  GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指標初期化解除関数                                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iFrAMA

この関数は Fractal Adaptive Moving Average（フラクタル適応型移動平均）指標のハンドルを返します。バッファは 1 つです。

```
int  iFrAMA(
   string               symbol,            // 銘柄名
   ENUM_TIMEFRAMES      period,            // 期間
   int                  ma_period,         // 平均期間
   int                  ma_shift,          // チャートの水平シフト
   ENUM_APPLIED_PRICE   applied_price      // 価格の種類かハンドル
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

ma_period

[in] 指標計算の平均期間（バーの数）

ma_shift

[in] 価格チャートでの指標のシフト

applied_price

[in] 使用される価格。ENUM_APPLIED_PRICE 価格定数のいずれか、または別の指標ハンドル。

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって、コンピュータメモリを利用されていない指標から解放することが出来ます。

例：

```
//+------------------------------------------------------------------+
//|                                                  Demo_iFrAMA.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iFrAMA technical indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the
'type' parameter (function type)."
```

© 2000–2025, MetaQuotes Ltd.

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iFrAMA の描画
#property indicator_label1  "iFrAMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iFrAMA,             // iFrAMA を使用する
   Call_IndicatorCreate     // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iFrAMA;           // 関数の種類
input int                   ma_period=14;               // 平均期間
input int                   ma_shift=0;                 // シフト
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE;  // 価格の種類
input string                symbol=" ";                 // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;      // 時間軸
//--- 指標バッファ
double          iFrAMABuffer[];
//--- iFrAMA 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- フラクタル適応型移動平均指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,iFrAMABuffer,INDICATOR_DATA);
//--- シフトを設定
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
```

© 2000-2025, MetaQuotes Ltd.

```
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iFrAMA)
      handle=iFrAMA(name,period,ma_period,ma_shift,applied_price);
   else
     {
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[3];
      //--- 移動平均の期間
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- シフト
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- 価格の種類
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_price;
      //--- 価格の種類
      handle=IndicatorCreate(name,period,IND_FRAMA,3,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iFrAMA indicator for the symbol %
s/%s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- iFrAMA 指標が計算された銘柄/時間軸を表示
   short_name=StringFormat("iFrAMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
```

© 2000-2025, MetaQuotes Ltd.

```
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iFrAMA 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
//--- これが指標計算の初めであるか iFrAMA 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- もし iFrAMABuffer 配列が銘柄/期間で iFrAMA 指標の値の数より大きい場合、全体のコピーはし
ない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- これは初回の計算ではなく、
      //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iFrAMABuffer 配列を フラクタル適応型移動平均指標の値で記入
//--- FillArrayFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArrayFromBuffer(iFrAMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- フラクタル適応型移動平均指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
```

© 2000-2025, MetaQuotes Ltd.

```
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iFrAMA 指標から指標バッファを記入する                                |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // フラクタル適応型移動平均 値の指標バッファ
                         int shift,         // シフト
                         int ind_handle,    // iFrAMA 指標ハンドル
                         int amount         // 複製された値の数
                        )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で iFrAMABuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iFrAMA indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指標初期化解除関数                                                |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iGator

この関数は Gator（ゲーター）指標のハンドルを返します。 オシレーターは、アリゲーターの青線と赤線の差（上ヒストグラム）と、赤線と緑線の差（下ヒストグラム）を示します。

```
int  iGator(
   string               symbol,            // 銘柄名
   ENUM_TIMEFRAMES      period,            // 期間
   int                  jaw_period,        // 顎の計算期間
   int                  jaw_shift,         // 顎の水平シフト
   int                  teeth_period,      // 歯の計算期間
   int                  teeth_shift,       // 歯の水平シフト
   int                  lips_period,       // 口の計算期間
   int                  lips_shift,        // 口の水平シフト
   ENUM_MA_METHOD       ma_method,         // 平滑化の種類
   ENUM_APPLIED_PRICE   applied_price      // 価格の種類かハンドル
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

jaw_period

[in] 青線の平均期間（アリゲーターの顎）

jaw_shift

[in] 価格チャートに相対した青線のシフト（指標ヒストグラムの視覚的なズレとは直接関係しません）。

teeth_period

[in] 赤線の平均期間（アリゲーターの歯）

teeth_shift

[in] 価格チャートに相対した赤線のシフト（指標ヒストグラムの視覚的なズレとは直接関係しません）。

lips_period

[in] 緑線の平均期間（アリゲーターの口）

lips_shift

[in] 価格チャートに相対した緑線のシフト（指標ヒストグラムの視覚的なズレとは直接関係しません）。

ma_method

[in] 平滑化の種類ENUM_MA_METHOD のいずれかの値

applied_price

[in] 使用される価格。ENUM_APPLIED_PRICE 価格定数のいずれか、または別の指標ハンドル。

© 2000–2025, MetaQuotes Ltd.

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって、コンピュータメモリを利用されていない指標から解放することが出来ます。

注意事項

バッファ番号は 0 - UPPER_HISTOGRAM, 1 - 上ヒストグラムの色バッファ 2 - LOWER_HISTOGRAM, 3 - 下ヒストグラムの色バッファです。

例:

```
//+------------------------------------------------------------------+
//|                                                  Demo_iGator.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iGator technical indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the
'type' parameter (function type)."
#property description "All other parameters are as in a standard Gator Oscillator."

#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   2
//--- GatorUp の描画
#property indicator_label1  "GatorUp"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen, clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- GatorDown の描画
#property indicator_label2  "GatorDown"
#property indicator_type2   DRAW_COLOR_HISTOGRAM
#property indicator_color2  clrGreen, clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iGator,             // iGator を使用する
   Call_IndicatorCreate     // IndicatorCreateを使用する
```

© 2000-2025, MetaQuotes Ltd.

```
  };
//--- 入力パラメータ
input Creation              type=Call_iGator;       // 関数の種類
input string                symbol=" ";             // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // 時間軸
input int                   jaw_period=13;          // 顎線の期間
input int                   jaw_shift=8;            // 顎線のシフト
input int                   teeth_period=8;         // 歯線の期間
input int                   teeth_shift=5;          // 歯線のシフト
input int                   lips_period=5;          // 口線の期間
input int                   lips_shift=3;           // 口線のシフト
input ENUM_MA_METHOD        MA_method=MODE_SMMA;    // アリゲーター線の平均化の方法
input ENUM_APPLIED_PRICE    applied_price=PRICE_MEDIAN;// アリゲーター計算に使用される価格の種類
//--- 指標バッファ
double          GatorUpBuffer[];
double          GatorUpColors[];
double          GatorDownBuffer[];
double          GatorDownColors[];
//--- iGator 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- 上下ヒストグラムのシフト値
int shift;
//--- ゲーターオシレーター指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,GatorUpBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,GatorUpColors,INDICATOR_COLOR_INDEX);
   SetIndexBuffer(2,GatorDownBuffer,INDICATOR_DATA);
   SetIndexBuffer(3,GatorDownColors,INDICATOR_COLOR_INDEX);
/*
  パラメータで指定された全てのシフトは、ゲーターオシレーターが描画されているアリゲーター指標を参照します。
  これが、ゲーター指標自体は動数にアリゲーター線を動かし、
  ゲーターオシレーター値の計算に使用される理由です。
*/
//--- 顎線と歯線との差に等しい上下のヒストグラムのシフトを計算してみる
   shift=MathMin(jaw_shift,teeth_shift);
   PlotIndexSetInteger(0,PLOT_SHIFT,shift);
//--- 指標の 2 つのヒストグラムに同じシフトを使用されるのが iGator 指標の実装の仕方
   PlotIndexSetInteger(1,PLOT_SHIFT,shift);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iGator)
      handle=iGator(name,period,jaw_period,jaw_shift,teeth_period,teeth_shift,
                    lips_period,lips_shift,MA_method,applied_price);
   else
     {
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[8];
      //--- アリゲーター線の期間とシフト
      pars[0].type=TYPE_INT;
      pars[0].integer_value=jaw_period;
      pars[1].type=TYPE_INT;
      pars[1].integer_value=jaw_shift;
      pars[2].type=TYPE_INT;
      pars[2].integer_value=teeth_period;
      pars[3].type=TYPE_INT;
      pars[3].integer_value=teeth_shift;
      pars[4].type=TYPE_INT;
      pars[4].integer_value=lips_period;
      pars[5].type=TYPE_INT;
      pars[5].integer_value=lips_shift;
      //--- 平滑化の種類
      pars[6].type=TYPE_INT;
      pars[6].integer_value=MA_method;
      //--- 価格の種類
      pars[7].type=TYPE_INT;
      pars[7].integer_value=applied_price;
      //--- ハンドルを作成
      handle=IndicatorCreate(name,period,IND_GATOR,8,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iGator indicator for the symbol %
s/%s, error code %d",
                  name,
```

© 2000-2025, MetaQuotes Ltd.

```
                    EnumToString(period),
                    GetLastError());
       //--- 指標が早期に中止された
       return(INIT_FAILED);
      }
//--- ゲーターオシレーターが計算された銘柄/時間軸を表示
   short_name=StringFormat("iGator(%s/%s, %d, %d ,%d, %d, %d, %
d)",name,EnumToString(period),
                           jaw_period,jaw_shift,teeth_period,teeth_shift,lips_period
,lips_shift);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iGator 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
//--- これが指標計算の初めであるか iGator 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- もし GatorUpBuffer 配列が銘柄/期間で iGator 指標の値の数より大きい場合、全体のコピーはし
ない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- これは初回の計算ではなく、
       //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
       values_to_copy=(rates_total-prev_calculated)+1;
      }
//--- 配列をゲーターオシレーター指標の値で記入
//--- FillArraysFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArraysFromBuffers(GatorUpBuffer,GatorUpColors,GatorDownBuffer,GatorDownCo
lors,
       shift,handle,values_to_copy)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- ゲーターオシレーター指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iGator 指標から指標バッファを記入する                                |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &ups_buffer[],          // 上ヒストグラムの指標バッファ
                           double &up_color_buffer[],     // 上ヒストグラムの価格インデックスの
指標バッファ
                           double &downs_buffer[],        // 下ヒストグラムの指標バッファ
                           double &downs_color_buffer[], // 下ヒストグラムの価格インデックスの
指標バッファ
                           int u_shift,                   // 上下ヒストグラムのシフト
                           int ind_handle,                // iGator 指標ハンドル
                           int amount                     // 複製された値の数
                          )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で GatorUpBuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,-u_shift,amount,ups_buffer)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iGator indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }

//--- インデックス 1 を持つ指標バッファの値で GatorUpColors 配列の一部を記入する
```

© 2000-2025, MetaQuotes Ltd.

```
   if(CopyBuffer(ind_handle,1,-u_shift,amount,up_color_buffer)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iGator indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }

//--- インデックス 2 を持つ指標バッファの値で GatorDownBuffer 配列の一部を記入する GatorDownColors
   if(CopyBuffer(ind_handle,2,-u_shift,amount,downs_buffer)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iGator indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }

//--- インデックス3 を持つ指標バッファの値でGatorDownColors 配列の一部を記入する
   if(CopyBuffer(ind_handle,3,-u_shift,amount,downs_color_buffer)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iGator indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指標初期化解除関数                                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iIchimoku

この関数は一目均衡表指標ハンドルを返します。

```
int  iIchimoku(
   string           symbol,           // 銘柄名
   ENUM_TIMEFRAMES  period,           // 期間
   int              tenkan_sen,       // 転換線の期間
   int              kijun_sen,        // 基準線の期間
   int              senkou_span_b     // 先行スパンBの期間
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

tenkan_sen

[in] 転換線の平均期間

kijun_sen

[in] 基準線の平均期間

senkou_span_b

[in] 先行スパンBの平均期間

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって、コンピュータメモリを利用されていない指標から解放することが出来ます。

注意事項

バッファ番号は 0 - TENKANSEN_LINE、1 - KIJUNSEN_LINE、2 - SENKOUSPANA_LINE、3 - SENKOUSPANB_LINE、4 - CHIKOUSPAN_LINE です。

例：

```
//+------------------------------------------------------------------+
//|                                                Demo_iIchimoku.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iIchimoku technical indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the 'type' parameter (function type)."
#property description "All other parameters just like in the standard Ichimoku Kinko Hyo."

#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   4
//--- Tenkan_sen プロット
#property indicator_label1  "Tenkan_sen"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- Kijun_sen プロット
#property indicator_label2  "Kijun_sen"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrBlue
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- Senkou_Span プロット
#property indicator_label3  "Senkou Span A;Senkou Span B" // 2 つのフィールドはデータウィンドウに表示される
#property indicator_type3   DRAW_FILLING
#property indicator_color3  clrSandyBrown, clrThistle
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//--- Chikou_Span プロット
#property indicator_label4  "Chinkou_Span"
#property indicator_type4   DRAW_LINE
#property indicator_color4  clrLime
#property indicator_style4  STYLE_SOLID
#property indicator_width4  1
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iIchimoku,          // iIchimoku を使用する
   Call_IndicatorCreate     // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iIchimoku;        // 関数の種類
input int                   tenkan_sen=9;               // 転換線の期間
input int                   kijun_sen=26;               // 基準線の期間
input int                   senkou_span_b=52;           // 先行スパンBの期間
input string                symbol=" ";                 // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;      // 時間軸
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 指標バッファ
double          Tenkan_sen_Buffer[];
double          Kijun_sen_Buffer[];
double          Senkou_Span_A_Buffer[];
double          Senkou_Span_B_Buffer[];
double          Chinkou_Span_Buffer[];
//--- 一目指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- 一目均衡表指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,Tenkan_sen_Buffer,INDICATOR_DATA);
   SetIndexBuffer(1,Kijun_sen_Buffer,INDICATOR_DATA);
   SetIndexBuffer(2,Senkou_Span_A_Buffer,INDICATOR_DATA);
   SetIndexBuffer(3,Senkou_Span_B_Buffer,INDICATOR_DATA);
   SetIndexBuffer(4,Chinkou_Span_Buffer,INDICATOR_DATA);
//--- 基準線バーの先行スパンチャンネルのシフトを将来に向けて設定する
   PlotIndexSetInteger(2,PLOT_SHIFT,kijun_sen);
//--- 遅行データスパンはすでに一目のシフトに保存されているので
//--- 遅行スパン線のシフトは設定する必要がない
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iIchimoku)
      handle=iIchimoku(name,period,tenkan_sen,kijun_sen,senkou_span_b);
   else
     {
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[3];
      //--- アリゲーター線の期間とシフト
      pars[0].type=TYPE_INT;
```

© 2000-2025, MetaQuotes Ltd.

```
      pars[0].integer_value=tenkan_sen;
      pars[1].type=TYPE_INT;
      pars[1].integer_value=kijun_sen;
      pars[2].type=TYPE_INT;
      pars[2].integer_value=senkou_span_b;
      //--- ハンドルを作成
      handle=IndicatorCreate(name,period,IND_ICHIMOKU,3,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iIchimoku indicator for the symbol
%s/%s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- 一目均衡表指標が計算された銘柄/時間軸を表示
   short_name=StringFormat("iIchimoku(%s/%s, %d, %d ,%d)",name,EnumToString(period),
                           tenkan_sen,kijun_sen,senkou_span_b);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- 一目均衡表指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
```

© 2000–2025, MetaQuotes Ltd.

```
      return(0);
     }
//--- これが指標計算の初めであるか一目均衡表指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- Tenkan_sen_Buffer 配列が銘柄/期間で一目指標の値の数より大きい場合、全体のコピーはしない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- これは初回の計算ではなく、
      //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- 配列を一目均衡表指標の値で記入
//--- FillArraysFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArraysFromBuffers(Tenkan_sen_Buffer,Kijun_sen_Buffer,Senkou_Span_A_Buffer
,Senkou_Span_B_Buffer,Chinkou_Span_Buffer,
      kijun_sen,handle,values_to_copy)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- 一目均衡表指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//|  一目指標から指標バッファを記入する                                |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &tenkan_sen_buffer[],     // 転換線の指標バッファ
                           double &kijun_sen_buffer[],      // 基準線の指標バッファ
                           double &senkou_span_A_buffer[],  // 選考スパンA線の指標バッファ
                           double &senkou_span_B_buffer[],  // 選考スパンB線の指標バッファ
                           double &chinkou_span_buffer[],   // 遅行スパン線の指標バッファ
                           int senkou_span_shift,           // 選考スパン線を将来に向けてシフト
                           int ind_handle,                  // 一目指標ハンドル
                           int amount                       // 複製された値の数
                          )
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で Tenkan_sen_Buffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,0,amount,tenkan_sen_buffer)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("1.Failed to copy data from the iIchimoku indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }

//--- インデックス 1 を持つ指標バッファの値で Kijun_sen_Buffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,1,0,amount,kijun_sen_buffer)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("2.Failed to copy data from the iIchimoku indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }

//--- インデックス 2 を持つ指標バッファの値で Chinkou_Span_Buffer 配列の一部を記入する
//--- senkou_span_shift>0 の場合、線はsenkou_span_shift バーの数だけ将来に向けてシフトされる
   if(CopyBuffer(ind_handle,2,-senkou_span_shift,amount,senkou_span_A_buffer)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("3.Failed to copy data from the iIchimoku indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }

//--- インデックス3 を持つ指標バッファの値で Senkou_Span_A_Buffer 配列の一部を記入する
//--- senkou_span_shift>0 の場合、線はsenkou_span_shift バーの数だけ将来に向けてシフトされる
   if(CopyBuffer(ind_handle,3,-senkou_span_shift,amount,senkou_span_B_buffer)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("4.Failed to copy data from the iIchimoku indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }

//--- インデックス0 を持つ指標バッファの値で Senkou_Span_B_Buffer 配列の一部を記入する
//--- 遅行スパンデータは既に一目シフトに保存されているので、遅行スパンの複製時に
//--- 遅行スパン線のシフトは設定する必要がない
   if(CopyBuffer(ind_handle,4,0,amount,chinkou_span_buffer)<0)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("5.Failed to copy data from the iIchimoku indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指標初期化解除関数                                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000–2025, MetaQuotes Ltd.

iBWMFI

この関数は Market Facilitation Index 指標ハンドルを返します。バッファは 1 つです。

```
int  iBWMFI(
   string                symbol,              // 銘柄名
   ENUM_TIMEFRAMES       period,              // 期間
   ENUM_APPLIED_VOLUME   applied_volume       // 計算に使用するボリュームの種類
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

applied_volume

[in] 使用されるボリューム（ENUM_APPLIED_VOLUME 定数のいずれか）

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって 、コンピュータメモリを利用されていない指標から解放することが出来ます。

例：

```
//+------------------------------------------------------------------+
//|                                                Demo_iBWMFI.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iBWMFI technical indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the
'type' parameter (function type)."

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- i BWMFI プロット
#property indicator_label1  "iBWMFI"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrLime,clrSaddleBrown,clrBlue,clrPink
#property indicator_style1  STYLE_SOLID
```

© 2000–2025, MetaQuotes Ltd.

```
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBWMFI,             // use iBWMFI を使用する
   Call_IndicatorCreate     // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iBWMFI;           // 関数の種類
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;// ボリュームの種類
input string                symbol=" ";                 // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;      // 時間軸
//--- 指標バッファ
double          iBWMFIBuffer[];
double          iBWMFIColors[];
//--- iBWMFI 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- BW MFI （ビルウィリアムズのマーケットファシリテーションインデックス）指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,iBWMFIBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iBWMFIColors,INDICATOR_COLOR_INDEX);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iBWMFI)
      handle=iBWMFI(name,period,applied_volume);
   else
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[1];
      //--- ボリュームの種類
      pars[0].type=TYPE_INT;
      pars[0].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_BWMFI,1,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iBWMFI indicator for the symbol %
s/%s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- BW MFI （ビルウィリアムズのマーケットファシリテーションインデックス）指標が計算された銘柄/時間軸を表示
   short_name=StringFormat("iBWMFI(%s/%s, %s)",name,EnumToString(period),
                           EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iBWMFI 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- これが指標計算の初めであるか iBWMFI 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合(価格履歴で何かが変更された)
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
      {
       //--- もしiBWMFIBuffer 配列が銘柄/期間で iBWMFI 指標の値の数より大きい場合、全体のコピーはしない
       //--- 他の場合、指標バッファサイズより少ない量をコピーをする
       if(calculated>rates_total) values_to_copy=rates_total;
       else                       values_to_copy=calculated;
      }
   else
      {
       //--- これは初回の計算ではなく、
       //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
       values_to_copy=(rates_total-prev_calculated)+1;
      }
//--- 配列を BW MFI (ビルウィリアムズのマーケットファシリテーションインデックス)指標の値で記入
//--- FillArraysFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArraysFromBuffers(iBWMFIBuffer,iBWMFIColors,handle,values_to_copy))
return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- BW MFI (ビルウィリアムズのマーケットファシリテーションインデックス)指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iBWMFI 指標から指標バッファを記入する                                |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &values[],     // ヒストグラム値の指標バッファ
                           double &colors[],     // ヒストグラムの色の指標バッファ colors
                           int ind_handle,       // iBWMFI 指標ハンドル
                           int amount            // 複製された値の数
                          )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で iBWMFIBuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
      {
       //--- 複製が失敗したら、エラーコードを出す
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("Failed to copy data from the iBWMFI indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- インデックス 1 を持つ指標バッファの値で iBWMFI Colors 配列の一部を記入する
   if(CopyBuffer(ind_handle,1,0,amount,colors)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iBWMFI indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指標初期化解除関数                                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iMomentum

この関数は Momentum（モメンタム）指標のハンドルを返します。バッファは 1 つです。

```
int  iMomentum(
   string                symbol,              // 銘柄名
   ENUM_TIMEFRAMES       period,              // 期間
   int                   mom_period,          // 平均期間
   ENUM_APPLIED_PRICE    applied_price        // 価格の種類かハンドル
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

mom_period

[in] 価格変化計算の平均期間（バーの数

applied_price

[in] 使用される価格。ENUM_APPLIED_PRICE 価格定数のいずれか、または別の指標ハンドル。

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって、コンピュータメモリを利用されていない指標から解放することが出来ます。

例：

```
//+------------------------------------------------------------------+
//|                                                Demo_iMomentum.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iMomentum technical indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the
'type' parameter (function type)."
#property description "All the other parameters are similar to the standard
Momentum."

#property indicator_separate_window
#property indicator_buffers 1
```

© 2000–2025, MetaQuotes Ltd.

```
#property indicator_plots   1
//--- モメンタムをプロット
#property indicator_label1  "iMomentum"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDodgerBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iMomentum,          // iMomentum を使用する
   Call_IndicatorCreate     // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iMomentum;        // 関数の種類
input int                   mom_period=14;              // モメンタムの期間
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // 価格の種類
input string                symbol=" ";                 // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;      // 時間軸
//--- 指標バッファ
double          iMomentumBuffer[];
//--- iMomentum 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- モメンタム指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,iMomentumBuffer,INDICATOR_DATA);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削する
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 指標ハンドルを作成する
   if(type==Call_iMomentum)
      handle=iMomentum(name,period,mom_period,applied_price);
   else
     {
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[2];
      //--- 期間
      pars[0].type=TYPE_INT;
      pars[0].integer_value=mom_period;
      //--- 価格の種類
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_MOMENTUM,2,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iMomentum indicator for the symbol %s/%s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- モメンタム指標が計算された銘柄/時間軸を表示
   short_name=StringFormat("iMomentum(%s/%s, %d, %s)",name,EnumToString(period),
                           mom_period, EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iMomentum 指標から複製された値の数
```

© 2000-2025, MetaQuotes Ltd.

```
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
//--- これが指標計算の初めであるか iMomentum 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- iMomentumBuffer 配列が銘柄/期間で iMomentum 指標の値の数より大きい場合、全体のコピー
はしない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- これは初回の計算ではなく、
      //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iMomentumBuffer 配列をモメンタム指標の値で記入
//--- FillArrayFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArrayFromBuffer(iMomentumBuffer,handle,values_to_copy)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- モメンタム指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iMomentum 指標から指標バッファを記入する                                  |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],   // モメンタム値の指標バッファ
                         int ind_handle,     // iMomentum 指標ハンドル
                         int amount          // 複製された値の数
                        )
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で iMomentumBuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iMomentum indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指標初期化解除関数                                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iMFI

この関数は MFI(Money Flow Index) 指標ハンドルを返します。

```
int  iMFI(
   string               symbol,              // 銘柄名
   ENUM_TIMEFRAMES      period,              // 期間
   int                  ma_period,           // 平均期間
   ENUM_APPLIED_VOLUME  applied_volume       // 計算に使用するボリュームの種類
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名 (NULL は現在のシンボル)

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

ma_period

[in] 計算の平均期間 (バーの数)

applied_volume

[in] 使用されるボリューム (ENUM_APPLIED_VOLUME 値のいずれか)

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって、コンピュータメモリを利用されていない指標から解放することが出来ます。

例:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iMFI.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iMFI technical indicator."
#property description "A symbol and timeframe used for calculation of the indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the 'type' parameter (function type)."
#property description "All the other parameters are similar to the standard Money Flow Index."

#property indicator_separate_window
#property indicator_buffers 1
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_plots   1
//--- ｉMFI プロット
#property indicator_label1  "iMFI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDodgerBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 指標ウィンドウの水平レベル
#property indicator_level1  20
#property indicator_level2  80
//+------------------------------------------------------------------+
//| 作成を処理する方法の列挙                                         |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iMFI,              // ｉMFI を使用する
   Call_IndicatorCreate    // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iMFI;              // 関数の種類
input int                   ma_period=14;                // 期間
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;  // ボリュームの種類
input string                symbol=" ";                  // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;       // 時間軸
//--- 指標バッファ
double          iMFIBuffer[];
//--- ｉMFI 指標ハンドルを格納する変数r
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- MFI 指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,iMFIBuffer,INDICATOR_DATA);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削する
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iMFI)
      handle=iMFI(name,period,ma_period,applied_volume);
   else
     {
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[2];
      //--- 期間
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- ボリュームの種類
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_MFI,2,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iMFI indicator for the symbol %s/%
s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- MFI 指標が計算された銘柄/時間軸を表示
   short_name=StringFormat("iMFI(%s/%s, %d, %s)",name,EnumToString(period),
                           ma_period, EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
```

© 2000–2025, MetaQuotes Ltd.

```
                const int &spread[])
  {
//--- iMFI 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
//--- これが指標計算の初めであるかiMFI 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- iMFIBuffer 配列が銘柄/期間でiMFI 指標の値の数より大きい場合、全体のコピーはしない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- これは初回の計算ではなく、
      //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iMFIBuffer 配列をMFI 指標の値で記入
//--- FillArrayFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArrayFromBuffer(iMFIBuffer,handle,values_to_copy)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- MFI 指標に値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iMFI 指標から指標バッファを記入する                                |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],   // MFI 値の指標バッファ
                         int ind_handle,     // iMFI 指標ハンドル
                         int amount          // 複製された値の数
```

© 2000-2025, MetaQuotes Ltd.

```
                              )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で iMFIBuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iMFI indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  指標初期化解除関数                                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000–2025, MetaQuotes Ltd.

iMA

この関数は Moving Average（移動平均線）指標のハンドルを返します。バッファは 1 つです。

```
int  iMA(
   string                symbol,              // 銘柄名
   ENUM_TIMEFRAMES       period,              // 期間
   int                   ma_period,           // 平均期間
   int                   ma_shift,            // 水平シフト
   ENUM_MA_METHOD        ma_method,           // 平滑化の種類
   ENUM_APPLIED_PRICE    applied_price        // 価格の種類かハンドル
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

ma_period

[in] 移動平均計算の期間

ma_shift

[in] 価格チャートに相対した指標のシフト

ma_method

[in] 平滑化の種類ENUM_MA_METHOD 値のいずれか

applied_price

[in] 使用される価格。ENUM_APPLIED_PRICE 価格定数のいずれか、または別の指標ハンドル。

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって 、コンピュータメモリを利用されていない指標から解放することが出来ます。

例：

```
//+------------------------------------------------------------------+
//|                                                     Demo_iMA.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iMA technical indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
```

© 2000–2025, MetaQuotes Ltd.

```
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the
'type' parameter (function type)."
#property description "All other parameters like in the standard Moving Average."

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iMA プロット
#property indicator_label1  "iMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iMA,                // iMA を使用
   Call_IndicatorCreate     // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iMA;                // 関数の種類
input int                   ma_period=10;                 // MA 期間
input int                   ma_shift=0;                   // シフト
input ENUM_MA_METHOD        ma_method=MODE_SMA;           // 平滑化の種類
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE;    // 価格の種類
input string                symbol=" ";                   // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // 時間軸
//--- 指標バッファ
double          iMABuffer[];
//--- iMA 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- 移動平均線指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,iMABuffer,INDICATOR_DATA);
//--- シフトを設定
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
```

© 2000–2025, MetaQuotes Ltd.

```
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iMA)
      handle=iMA(name,period,ma_period,ma_shift,ma_method,applied_price);
   else
     {
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[4];
      //--- 期間
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- シフト
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- 平滑化の種類
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- 価格の種類
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_MA,4,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iMA indicator for the symbol %s/%
s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- 移動平均線が計算された銘柄/時間軸を表示
   short_name=StringFormat("iMA(%s/%s, %d, %d, %s, %s)",name,EnumToString(period),
                           ma_period,
ma_shift,EnumToString(ma_method),EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iMA 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
//--- これが指標計算の初めであるか iMA 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- iMABuffer 配列が銘柄/期間で iMA 指標の値の数より大きい場合、全体のコピーはしない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- これは初回の計算ではなく、
      //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iMABuffer 配列を適応型移動平均線指標の値で記入する
//--- FillArrayFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArrayFromBuffer(iMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
```

© 2000-2025, MetaQuotes Ltd.

```
                                short_name,
                                values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- 移動平均線指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| 移動平均線指標から指標バッファを記入する                                  |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],   // 移動平均線値の指標バッファー
                         int shift,          // シフト
                         int ind_handle,     // iMA 指標ハンドル
                         int amount          // 複製された値の数
                        )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で iMABuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iMA indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指標初期化解除関数                                                  |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iOsMA

この関数は Moving Average of Oscillator（移動平均オシレーター）指標のハンドルを返します。OsMA オシレーターは MACD とシグナル線の差を表示します。バッファは 1 つです。

```
int  iOsMA(
   string               symbol,              // 銘柄名
   ENUM_TIMEFRAMES      period,              // 期間
   int                  fast_ema_period,     // 高速移動平均期間
   int                  slow_ema_period,     // 低速移動平均期間
   int                  signal_period,       // 差の平均期間
   ENUM_APPLIED_PRICE   applied_price        // 価格の種類かハンドル
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

fast_ema_period

[in] 高速移動平均計算の期間

slow_ema_period

[in] 低速移動平均計算の期間

signal_period

[in] シグナル線計算の平均期間

applied_price

[in] 使用される価格。ENUM_APPLIED_PRICE 価格定数のいずれか、または別の指標ハンドル。

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって、コンピュータメモリを利用されていない指標から解放することが出来ます。

注意事項

システムによってはこのオシレーターは MACD ヒストグラムとして知られています。

例：

```
//+------------------------------------------------------------------+
//|                                                   Demo_iOsMA.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

© 2000–2025, MetaQuotes Ltd.

```
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iOsMA technical indicator."
#property description "A symbol and timeframe used for calculation of the indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the 'type' parameter (function type)."
#property description "All the other parameters are similar to the standard Moving Average of Oscillator."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iOsMA プロット
#property indicator_label1  "iOsMA"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iOsMA,              // iOsMA を使用する
   Call_IndicatorCreate     // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iOsMA;            // 関数の種類
input int                   fast_ema_period=12;         // 高速 MA 期間
input int                   slow_ema_period=26;         // 低速 MA 期間
input int                   signal_period=9;            // 差の平均期間
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // 価格の種類
input string                symbol=" ";                 // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;      // 時間軸
//--- 指標バッファ
double          iOsMABuffer[];
//--- iAMA 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- 移動平均線指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,iOsMABuffer,INDICATOR_DATA);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iOsMA)
      handle=iOsMA(name,period,fast_ema_period,slow_ema_period,signal_period,applied
_price);
   else
     {
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[4];
      //--- 高速 MA 期間
      pars[0].type=TYPE_INT;
      pars[0].integer_value=fast_ema_period;
      //--- 低速 MA 期間
      pars[1].type=TYPE_INT;
      pars[1].integer_value=slow_ema_period;
      //--- 高速移動平均と低速移動平均の差の平均期間
      pars[2].type=TYPE_INT;
      pars[2].integer_value=signal_period;
      //--- 価格の種類
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_OSMA,4,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create a handle of iOsMA for the pair %s/%s, error code
is %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- 移動平均オシレーター指標が計算された銘柄/時間軸を表示
```

© 2000-2025, MetaQuotes Ltd.

```
   short_name=StringFormat("iOsMA(%s/%s,%d,%d,%d,%s)",name,EnumToString(period),
                           fast_ema_period,slow_ema_period,signal_period,EnumToStrin
g(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iOsMA 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
//--- これが指標計算の初めであるか iOsMA 指標の値の数が変更した changed
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- iOsMABuffer 配列が銘柄/期間で iOsMA 指標の値の数より大きい場合、全体のコピーはしない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- これは初回の計算ではなく、
      //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- 配列をi OsMA 指標の値で記入
//--- FillArrayFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!FillArrayFromBuffer(iOsMABuffer,handle,values_to_copy)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- 移動平均オシレーター指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iOsMA 指標から指標バッファを記入する                             |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &ama_buffer[],  // OsMA 値の指標バッファ
                         int ind_handle,        // iOsMA 指標ハンドル
                         int amount             // 複製された値の数
                        )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で iOsMABuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,0,amount,ama_buffer)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iOsMA indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指標初期化解除関数                                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iMACD

この関数は MACD(Moving Averages Convergence/Divergence)指標ハンドルを返します。OsMA が MACD ヒストグラムと呼ばれるシステムでは、この指標は 2 本の線として表示されます。クライアント端末では MACD はヒストグラムのように見えます。

```
int  iMACD(
   string               symbol,              // 銘柄名
   ENUM_TIMEFRAMES      period,              // 期間
   int                  fast_ema_period,     // 高速平均計算の期間
   int                  slow_ema_period,     // 低速平均計算の期間
   int                  signal_period,       // 差の平均計算の期間
   ENUM_APPLIED_PRICE   applied_price        // 価格の種類かハンドル
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名(NULL は現在のシンボル)

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

fast_ema_period

[in] 高速移動平均計算の期間

slow_ema_period

[in] 低速移動平均計算の期間

signal_period

[in] シグナル線の計算期間

applied_price

[in] 使用される価格。ENUM_APPLIED_PRICE 価格定数のいずれか、または別の指標ハンドル。

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって、コンピュータメモリを利用されていない指標から解放することが出来ます。

注意事項

バッファ番号は 0 - MAIN_LINE, 1 - SIGNAL_LINE です。

例:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iMACD.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
```

© 2000-2025, MetaQuotes Ltd.

```
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iMACD technical indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the
'type' parameter (function type)."
#property description "All other parameters like in the standard MACD."

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   2
//--- MACD プロット
#property indicator_label1  "MACD"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- シグナルプロット
#property indicator_label2  "Signal"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_DOT
#property indicator_width2  1
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iMACD,              // iMACD を使用する
   Call_IndicatorCreate     // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iMACD;            // 関数の種類
input int                   fast_ema_period=12;         // 高速 MA 期間
input int                   slow_ema_period=26;         // 低速 MA 期間
input int                   signal_period=9;            // 差の平均期間
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // 価格の種類
input string                symbol=" ";                 // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;      // 時間軸
//--- 指標バッファ
double          MACDBuffer[];
double          SignalBuffer[];
//--- iMACD 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
```

© 2000-2025, MetaQuotes Ltd.

```
string short_name;
//--- MACD 指標に値の数を保存
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,MACDBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iMACD)
      handle=iMACD(name,period,fast_ema_period,slow_ema_period,signal_period,applied
_price);
   else
     {
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[4];
      //--- 高速 MA 期間
      pars[0].type=TYPE_INT;
      pars[0].integer_value=fast_ema_period;
      //--- 低速 MA 期間
      pars[1].type=TYPE_INT;
      pars[1].integer_value=slow_ema_period;
      //--- 高速移動平均と低速移動平均の差の平均期間
      pars[2].type=TYPE_INT;
      pars[2].integer_value=signal_period;
      //--- 価格の種類
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_MACD,4,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("Failed to create handle of the iMACD indicator for the symbol %s/
%s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- MACD が計算された銘柄/時間軸を表示
   short_name=StringFormat("iMACD(%s/%s,%d,%d,%d,%s)",name,EnumToString(period),
                           fast_ema_period,slow_ema_period,signal_period,EnumToStrin
g(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iMACD 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
//--- これが指標計算の初めであるか iMACD 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- MACDBuffer 配列が銘柄/期間で MACD 指標の値の数より大きい場合、全体のコピーはしない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
    else
      {
       //--- これは初回の計算ではなく、
       //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
       values_to_copy=(rates_total-prev_calculated)+1;
      }
//--- 配列を iMACD 指標の値で記入
//--- FillArraysFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArraysFromBuffers(MACDBuffer,SignalBuffer,handle,values_to_copy))
return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- MACD 指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iMACD 指標から指標バッファを記入する                                |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &macd_buffer[],     // MACD 値の指標バッファ
                           double &signal_buffer[],   // MACD シグナル線の指標バッファ
                           int ind_handle,            // iMACD 指標ハンドル
                           int amount                 // 複製された値の数
                          )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で iMACDBuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,0,amount,macd_buffer)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iMACD indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }

//--- インデックス 1 を持つ指標バッファの値で Signal Buffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,1,0,amount,signal_buffer)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("Failed to copy data from the iMACD indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指標初期化解除関数                                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000–2025, MetaQuotes Ltd.

iOBV

この関数は On Balance Volume（オンバランスボリューム）指標のハンドルを返します。バッファは 1 つです。

```
int  iOBV(
   string                symbol,             // 銘柄名
   ENUM_TIMEFRAMES       period,             // 期間
   ENUM_APPLIED_VOLUME   applied_volume      // 計算に使用するボリュームの種類
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

applied_volume

[in] 使用されるボリューム（ENUM_APPLIED_VOLUME 値のいずれか）

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって、コンピュータメモリを利用されていない指標から解放することが出来ます。

例：

```
//+------------------------------------------------------------------+
//|                                                   Demo_iOBV.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iOBV technical indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the
'type' parameter (function type)."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---  iOBV
#property indicator_label1  "iOBV"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
```

© 2000–2025, MetaQuotes Ltd.

```
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iOBV ,              // ｉOBV を使用する
   Call_IndicatorCreate     // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iOBV;                // 関数の種類
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;    // ボリュームの種類
input string                symbol=" ";                    // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;         // 時間軸
//--- 指標バッファ
double          iOBVBuffer[];
//--- ｉOBV 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- オンバランスボリューム指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,iOBVBuffer,INDICATOR_DATA);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削する
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iOBV)
      handle=iOBV(name,period,applied_volume);
   else
     {
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[1];
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- ボリュームの種類
      pars[0].type=TYPE_INT;
      pars[0].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_OBV,1,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iOBV indicator for the symbol %s/%
s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- オンバランスボリューム指標が計算された銘柄/時間軸を表示
   short_name=StringFormat("iOBV(%s/%s, %s)",name,EnumToString(period),
                           EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iOBV 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
//--- これが指標計算の初めであるか iOBV 指標の値の数が変更した
```

© 2000-2025, MetaQuotes Ltd.

```
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- iOBVBuffer 配列が銘柄/期間で iOBV 指標の値の数より大きい場合、全体のコピーはしない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- これは初回の計算ではなく、
      //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- 配列を iOBV 指標の値で記入
//--- FillArrayFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArrayFromBuffer(iOBVBuffer,handle,values_to_copy)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- オンバランスボリューム指標値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iOBV 指標から指標バッファを記入する                                |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &obv_buffer[],  // OBV 値の指標バッファ
                         int ind_handle,        // iOBV 指標ハンドル
                         int amount             // 複製された値の数
                        )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で iOBVBuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,0,amount,obv_buffer)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iOBV indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指標初期化解除関数                                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000–2025, MetaQuotes Ltd.

iSAR

この関数は Parabolic Stop and Reverse system（パラボリック SAR）の指標ハンドルを返します。バッファは 1 つです。

```
int  iSAR(
   string            symbol,      // 銘柄名
   ENUM_TIMEFRAMES   period,      // 期間
   double            step,        // 価格増分ステップ(加速因子)
   double            maximum      // ステップの最大値
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

step

[in] 価格増分ステップ（通常 0.02）

maximum

[in] ステップの最大値（通常 0.2）

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって、コンピュータメモリを利用されていない指標から解放することが出来ます。

例：

```
//+------------------------------------------------------------------+
//|                                                    Demo_iSAR.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iSAR technical indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the
'type' parameter (function type)."
#property description "All the other parameters are similar to the standard
Parabolic Stop and Reverse."

#property indicator_chart_window
```

© 2000–2025, MetaQuotes Ltd.

```
#property indicator_buffers 1
#property indicator_plots   1
//--- ｉSAR の描画
#property indicator_label1  "iSAR"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iSAR,              // ｉSAR を使用する
   Call_IndicatorCreate    // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iSAR;              // 関数の種類
input double                step=0.02;                   // ストップトレイリングの加速因子
input double                maximum=0.2;                 // ステップの最大値
input string                symbol=" ";                  // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;       // 時間軸
//--- 指標バッファ
double          iSARBuffer[];
//--- ｉSAR 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- パラボリックSAR 指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,iSARBuffer,INDICATOR_DATA);
//--- シャーとに表示する PLOT_ARROW プロパティのコードを Wingdings 文字セットからの記号を使用して設定
   PlotIndexSetInteger(0,PLOT_ARROW,159);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削する
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iSAR)
      handle=iSAR(name,period,step,maximum);
   else
     {
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[2];
      //--- ステップ値
      pars[0].type=TYPE_DOUBLE;
      pars[0].double_value=step;
      //--- 計算に使用するステップ値の制限
      pars[1].type=TYPE_DOUBLE;
      pars[1].double_value=maximum;
      handle=IndicatorCreate(name,period,IND_SAR,2,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iSAR indicator for the symbol %s/%
s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- パラボリック SAR 指標が計算された銘柄/時間軸を表示する
   short_name=StringFormat("iSAR(%s/%s, %G, %G)",name,EnumToString(period),
                           step,maximum);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const int &spread[])
  {
//--- iSAR 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
//--- これが指標計算の初めであるか iSAR 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- iSARBuffer 配列が銘柄/期間でi SAR 指標の値の数より大きい場合、全体のコピーはしない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- これは初回の計算ではなく、
      //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- 配列をi SAR 指標の値で記入
//--- FillArrayFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArrayFromBuffer(iSARBuffer,handle,values_to_copy)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- パラボリックSAR 指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iSAR 指標から指標バッファを記入する                                |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &sar_buffer[],   // パラボリック SAR 値の指標バッファ
                         int ind_handle,         // iSAR 指標ハンドル
                         int amount              // 複製された値の数
```

© 2000-2025, MetaQuotes Ltd.

```
                              )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で iSARBuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,0,amount,sar_buffer)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iSAR indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  指標初期化解除関数                                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000–2025, MetaQuotes Ltd.

iRSI

この関数は Relative Strength Index（相対力指数）指標のハンドルを返します。バッファは 1 つです。

```
int  iRSI(
   string               symbol,            // 銘柄名
   ENUM_TIMEFRAMES      period,            // 期間
   int                  ma_period,         // 平均期間
   ENUM_APPLIED_PRICE   applied_price      // 価格の種類かハンドル
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

ma_period

[in] RSI 計算の平均期間

applied_price

[in] 使用される価格。ENUM_APPLIED_PRICE 価格定数のいずれか、または別の指標ハンドル。

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって、コンピュータメモリを利用されていない指標から解放することが出来ます。

例：

```
//+------------------------------------------------------------------+
//|                                                    Demo_iRSI.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iRSI technical indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the
'type' parameter (function type)."
#property description "All the other parameters are similar to the standard Relative
Strength Index."

#property indicator_separate_window
#property indicator_buffers 1
```

© 2000–2025, MetaQuotes Ltd.

```
#property indicator_plots   1
//--- ｉRSI の描画
#property indicator_label1  "iRSI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDodgerBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 指標ウィンドウでの値の表示の制限
#property indicator_maximum 100
#property indicator_minimum 0
//--- 指標ウィンドウの水平レベル
#property indicator_level1  70.0
#property indicator_level2  30.0
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iRSI,              // ｉRSI を使用する
   Call_IndicatorCreate    // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iRSI;              // 関数の種類
input int                   ma_period=14;                // 平均期間
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE;   // 価格の種類
input string                symbol=" ";                  // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;       // 時間軸
//--- 指標バッファ
double          iRSIBuffer[];
//--- ｉRSI 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- 相対力指数指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,iRSIBuffer,INDICATOR_DATA);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
```

© 2000-2025, MetaQuotes Ltd.

```
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iRSI)
      handle=iRSI(name,period,ma_period,applied_price);
   else
     {
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[2];
      //--- 移動平均の期間
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- 計算に使用するステップ値の制限
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_RSI,2,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iRSI indicator for the symbol %s/%
s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- 相対力指数指標が計算された銘柄/時間軸を表示
   short_name=StringFormat("iRSI(%s/%s, %d, %d)",name,EnumToString(period),
                           ma_period,applied_price);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//|  カスタム指標の反復関数                                          |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iRSI 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
//--- これが指標計算の初めであるかiRSI 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- iRSIBuffer 配列が銘柄/期間でiRSI 指標の値の数より大きい場合、全体のコピーはしない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- これは初回の計算ではなく、
      //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- 配列をiRSI 指標の値で記入
//--- FillArrayFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArrayFromBuffer(iRSIBuffer,handle,values_to_copy)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- 相対力指数指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iRSI 指標から指標バッファを記入する                                |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool FillArrayFromBuffer(double &rsi_buffer[],   // 相対力指数値の指標バッファ
                         int ind_handle,         // iRSI 指標ハンドル
                         int amount              // 複製された値の数
                        )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で iRSI Buffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,0,amount,rsi_buffer)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iRSI indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指標初期化解除関数                                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000–2025, MetaQuotes Ltd.

iRVI

この関数は Relative Vigor Index（相対活力指数）指標ハンドルを返します。

```
int  iRVI(
   string           symbol,         // 銘柄名
   ENUM_TIMEFRAMES  period,         // 期間
   int              ma_period       // 平均期間
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

ma_period

[in] RVI 計算の平均期間

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって 、コンピュータメモリを利用されていない指標から解放することが出来ます。

注意事項

バッファ番号は 0 - MAIN_LINE, 1 - SIGNAL_LINE です。

例：

```
//+------------------------------------------------------------------+
//|                                                    Demo_iRVI.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iRVI technical indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the
'type' parameter (function type)."
#property description "All the other parameters are similar to the standard Relative
Vigor Index."

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   2
```

© 2000-2025, MetaQuotes Ltd.

```
//--- RVI プロット
#property indicator_label1  "RVI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- シグナルプロット
#property indicator_label2  "Signal"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//+------------------------------------------------------------------+
//| 作成を処理する方法の列挙                                         |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iRVI,              // iRVI を使用する
   Call_IndicatorCreate    // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iRVI;              // 関数の種類
input int                   ma_period=10;                // 計算期間
input string                symbol=" ";                  // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;       // 時間軸
//--- 指標バッファ
double          RVIBuffer[];
double          SignalBuffer[];
//--- iRVI 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- 相対活力指数指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,RVIBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
```

© 2000-2025, MetaQuotes Ltd.

```
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iRVI)
      handle=iRVI(name,period,ma_period);
   else
     {
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[1];
      //--- 計算期間
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_RVI,1,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iRVI indicator for the symbol %s/%
s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- 相対活力指数指標が計算された銘柄/時間軸を表示
   short_name=StringFormat("iRVI(%s/%s, %d, %
d)",name,EnumToString(period),ma_period);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const int &spread[])
  {
//--- ｉRVI　指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
//--- これが指標計算の初めであるかｉRVI　指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- RVI Buffer 配列が銘柄/期間でｉRVI　指標の値の数より大きい場合、全体のコピーはしない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- これは初回の計算ではなく、
      //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- 配列をｉRVI　指標の値で記入
//--- FillArrayFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArrayFromBuffer(RVIBuffer,SignalBuffer,handle,values_to_copy)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- 相対活力指数指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//|　ｉRVI　指標から指標バッファを記入する                                |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &rvi_buffer[],     // 相対活力指数値の指標バッファ
                         double &signal_buffer[],  // シグナル線の指標バッファ
                         int ind_handle,           // ｉRVI　指標ハンドル
```

© 2000-2025, MetaQuotes Ltd.

```
                              int amount                 // 複製された値の数
                             )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で iRVI Buffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,0,amount,rvi_buffer)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iRVI indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- インデックス 1 を持つ指標バッファの値で Signal Buffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,1,0,amount,signal_buffer)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iRVI indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指標初期化解除関数                                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iStdDev

この関数は Standard Deviation（標準偏差）指標のハンドルを返します。バッファは 1 つです。

```
int  iStdDev(
   string               symbol,            // 銘柄名
   ENUM_TIMEFRAMES      period,            // 期間
   int                  ma_period,         // 平均期間
   int                  ma_shift,          // 水平シフト
   ENUM_MA_METHOD       ma_method,         // 平滑化の種類
   ENUM_APPLIED_PRICE   applied_price      // 価格の種類かハンドル
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

ma_period

[in] 指標計算の平均期間

ma_shift

[in] 価格チャートに相対した指標のシフト

ma_method

[in] 平均化の方法ENUM_MA_METHOD 値のいずれか

applied_price

[in] 使用される価格。ENUM_APPLIED_PRICE 価格定数のいずれか、または別の指標ハンドル。

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって 、コンピュータメモリを利用されていない指標から解放することが出来ます。

例：

```
//+------------------------------------------------------------------+
//|                                                 Demo_iStdDev.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iStdDev technical indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
```

© 2000–2025, MetaQuotes Ltd.

```
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the
'type' parameter (function type)."
#property description "All the other parameters are similar to the normal Standard
Deviation."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iStdDev プロット
#property indicator_label1  "iStdDev"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrMediumSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iStdDev,            // iStdDev を使用
   Call_IndicatorCreate     // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iStdDev;          // 関数の種類
input int                   ma_period=20;               // 平均期間
input int                   ma_shift=0;                 // シフト
input ENUM_MA_METHOD        ma_method=MODE_SMA;         // 平滑化の種類
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // 価格の種類
input string                symbol=" ";                 // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;      // 時間軸
//--- 指標バッファ
double          iStdDevBuffer[];
//--- iStdDev 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- 標準偏差指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,iStdDevBuffer,INDICATOR_DATA);
//--- シフトを設定
```

© 2000–2025, MetaQuotes Ltd.

```
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iStdDev)
      handle=iStdDev(name,period,ma_period,ma_shift,ma_method,applied_price);
   else
     {
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[4];
      //--- 期間
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- シフト
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- 平滑化の種類
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- 価格の種類
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_STDDEV,4,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iStdDev indicator for the symbol %
s/%s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- 標準偏差指標が計算された銘柄/時間軸を表示
   short_name=StringFormat("iStdDev(%s/%s, %d, %d, %s, %
s)",name,EnumToString(period),
```

© 2000-2025, MetaQuotes Ltd.

```
                              ma_period,ma_shift,EnumToString(ma_method),EnumToString(a
pplied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iStdDev 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
//--- これが指標計算の初めであるか iStdDev 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- iStdDevBuffer 配列が銘柄/期間で iStdDev 指標の値の数より大きい場合、全体のコピーはしな
い
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- これは初回の計算ではなく、
      //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- 配列を標準偏差指標の値で記入
//--- FillArrayFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!FillArrayFromBuffer(iStdDevBuffer,ma_shift,handle,values_to_copy)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- 標準偏差指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iStdDev 指標から指標バッファを記入する                           |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &std_buffer[],  // 標準偏差線の指標バッファ
                         int std_shift,         // 標準偏差線のシフト
                         int ind_handle,        // iStdDev 指標ハンドル
                         int amount             // 複製された値の数
                        )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で iStdDevBuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,-std_shift,amount,std_buffer)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iStdDev indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指標初期化解除関数                                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iStochastic

この関数は Stochastic Oscillator（ストキャスティックス）指標のハンドルを返します。

```
int  iStochastic(
   string            symbol,          // 銘柄名
   ENUM_TIMEFRAMES   period,          // 期間
   int               Kperiod,         // K期間（計算に使用されるバーの数）
   int               Dperiod,         // D期間（初めの平滑化の期間）
   int               slowing,         // 最終の平滑化
   ENUM_MA_METHOD    ma_method,       // 平滑化の種類
   ENUM_STO_PRICE    price_field      // 確率論的計算方法
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

Kperiod

[in] %K 線計算の平均期間（バーの数）

Dperiod

[in] %D 線計算の平均期間（バーの数）

slowing

[in] スロー値

ma_method

[in] 平均化の方法ENUM_MA_METHOD 値のいずれか

price_field

[in] 計算に使用される価格選択のパラメータENUM_STO_PRICE 値のいずれか

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって 、コンピュータメモリを利用されていない指標から解放することが出来ます。

注意事項

バッファ番号は 0 - MAIN_LINE, 1 - SIGNAL_LINE です。

例：

```
//+------------------------------------------------------------------+
//|                                              Demo_iStochastic.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iStochastic technical
indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the
'type' parameter (function type)."
#property description "All the other parameters are similar to the standard
Stochastic Oscillator."

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   2
//--- ストキャスティックスプロット
#property indicator_label1  "Stochastic"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- シグナルプロット
#property indicator_label2  "Signal"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- 指標値を制限する
#property indicator_minimum 0
#property indicator_maximum 100
//--- 指標ウィンドウの水平レベル
#property indicator_level1  -100.0
#property indicator_level2  100.0
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iStochastic,       // use iStochastic を使用する
   Call_IndicatorCreate    // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iStochastic;     // 関数の種類
input int                   Kperiod=5;                 // K 期間(計算に使用されるバーの数)
input int                   Dperiod=3;                 // D 期間(主要な平滑化の期間)
input int                   slowing=3;                 // 最終平滑化期間
input ENUM_MA_METHOD        ma_method=MODE_SMA;        // 平滑化の種類
```

© 2000-2025, MetaQuotes Ltd.

```
input ENUM_STO_PRICE       price_field=STO_LOWHIGH;    // 確率の計算方法
input string               symbol=" ";                 // シンボル
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;      // 時間軸
//--- 指標バッファ
double          StochasticBuffer[];
double          SignalBuffer[];
//--- iStochastic 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- ストキャスティックス—指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| カスタム指標を初期化する関数                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,StochasticBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iStochastic)
      handle=iStochastic(name,period,Kperiod,Dperiod,slowing,ma_method,price_field);
   else
     {
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[5];
      //--- 計算の K 期間
      pars[0].type=TYPE_INT;
      pars[0].integer_value=Kperiod;
      //--- 主要な平滑化の D 期間
      pars[1].type=TYPE_INT;
      pars[1].integer_value=Dperiod;
      //--- 最後の平滑化の K 期間
      pars[2].type=TYPE_INT;
      pars[2].integer_value=slowing;
```

© 2000–2025, MetaQuotes Ltd.

```
      //--- 平滑化の種類
      pars[3].type=TYPE_INT;
      pars[3].integer_value=ma_method;
      //--- 確率の計算方法
      pars[4].type=TYPE_INT;
      pars[4].integer_value=price_field;
      handle=IndicatorCreate(name,period,IND_STOCHASTIC,5,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iStochastic indicator for the
symbol %s/%s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- ストキャスティックスーが計算された銘柄/時間軸を表示
   short_name=StringFormat("iStochastic(%s/%s, %d, %d, %d, %s, %
s)",name,EnumToString(period),
                           Kperiod,Dperiod,slowing,EnumToString(ma_method),EnumToStr
ing(price_field));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iStochastic 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
//--- これが指標計算の初めであるか iStochastic 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- StochasticBuffer 配列が銘柄/期間で iStochastic 指標の値の数より大きい場合、全体のコ
ピーはしない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- これは初回の計算ではなく、
      //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- f配列を iStochastic 指標の値で記入
//--- FillArraysFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArraysFromBuffers(StochasticBuffer,SignalBuffer,handle,values_to_copy))
return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- ストキャスティックス一指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iStochastic 指標から指標バッファを記入する                       |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &main_buffer[],    // ストキャスティックス一値の指標バッファ
                           double &signal_buffer[],  // シグナル線の指標バッファ
                           int ind_handle,           // iStochastic 指標ハンドル
                           int amount                // 複製された値の数
                          )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で StochasticBuffer 配列の一部を記入する
```

© 2000-2025, MetaQuotes Ltd.

```
   if(CopyBuffer(ind_handle,MAIN_LINE,0,amount,main_buffer)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iStochastic indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- インデックス 1 を持つ指標バッファの値で Signal Buffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,SIGNAL_LINE,0,amount,signal_buffer)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iStochastic indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指標初期化解除関数                                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iTEMA

この関数は Triple Exponential Moving Average 指標ハンドルを返します。バッファは 1 つです。

```
int  iTEMA(
   string               symbol,           // 銘柄名
   ENUM_TIMEFRAMES      period,           // 期間
   int                  ma_period,        // 平均期間
   int                  ma_shift,         // 指標の水平シフト
   ENUM_APPLIED_PRICE   applied_price     // 価格の種類かハンドル
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

ma_period

[in] 計算の平均期間（バーの数）

ma_shift

[in] 価格チャートに相対した指標のシフト

applied_price

[in] 使用される価格。ENUM_APPLIED_PRICE 価格定数のいずれか、または別の指標ハンドル。

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって、コンピュータメモリを利用されていない指標から解放することが出来ます。

例：

```
//+------------------------------------------------------------------+
//|                                                  Demo_iTEMA.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iTEMA technical indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the
'type' parameter (function type)."
#property description "All the other parameters are similar to the standard Triple
Exponential Moving Average."
```

© 2000–2025, MetaQuotes Ltd.

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- ｉTEMA プロット
#property indicator_label1  "iTEMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iTEMA,              // ｉTEMA を使用する
   Call_IndicatorCreate     // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iTEMA;            // 関数の種類
input int                   ma_period=14;               // 平均期間
input int                   ma_shift=0;                 // シフト
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // 価格の種類
input string                symbol=" ";                 // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;      // 時間軸
//--- 指標バッファ
double          iTEMABuffer[];
//--- ｉTEMA 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- Triple Exponential Moving Average 指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,iTEMABuffer,INDICATOR_DATA);
//--- シフトを設定
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
```

© 2000-2025, MetaQuotes Ltd.

```
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iTEMA)
      handle=iTEMA(name,period,ma_period,ma_shift,applied_price);
   else
     {
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[3];
      //--- 期間
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- シフト
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- 価格の種類
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_TEMA,3,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iTEMA indicator for the symbol %s/
%s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- Triple Exponential Moving Average 指標が計算された銘柄/時間軸を表示
   short_name=StringFormat("iTEMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iTEMA 指標から複製された値の数r
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
//--- これが指標計算の初めであるか iTEMA 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- iTEMABuffer 配列が銘柄/期間で iTEMA 指標の値の数より大きい場合、全体のコピーはしない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- これは初回の計算ではなく、
      //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- 配列をTriple Exponential Moving Average 指標の値で記入
//--- FillArrayFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArrayFromBuffer(iTEMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- Triple Exponential Moving Average 指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  i TEMA  指標から指標バッファを記入する                          |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &tema_buffer[], // Triple Exponential Moving Average
値の指標バッファ
                         int t_shift,           // 線のシフト
                         int ind_handle,        // i TEMA  指標ハンドル
                         int amount             // 複製された値の数
                        )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0  を持つ指標バッファの値で i TEMABuffer  配列の一部を記入する
   if(CopyBuffer(ind_handle,0,-t_shift,amount,tema_buffer)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iTEMA indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  指標初期化解除関数                                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iTriX

この関数はTriple Exponential Moving Averages Oscillator 指標ハンドルを返します。バッファは 1 つです。

```
int  iTriX(
   string               symbol,            // 銘柄名
   ENUM_TIMEFRAMES      period,            // 期間
   int                  ma_period,         // 平均期間
   ENUM_APPLIED_PRICE   applied_price      // 価格の種類かハンドル
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

ma_period

[in] 計算の平均期間（バーの数）

applied_price

[in] 使用される価格。ENUM_APPLIED_PRICE 価格定数のいずれか、または別の指標ハンドル。

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって、コンピュータメモリを利用されていない指標から解放することが出来ます。

例：

```
//+------------------------------------------------------------------+
//|                                                   Demo_iTriX.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iTriX technical indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the
'type' parameter (function type)."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iTriX プロット
```

© 2000–2025, MetaQuotes Ltd.

```
#property indicator_label1  "iTriX"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iTriX,              // iTriX を使用する
   Call_IndicatorCreate     // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iTriX;            // 関数の種類
input int                   ma_period=14;               // 期間
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // 価格の種類
input string                symbol=" ";                 // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;      // 時間軸
//--- 指標バッファ
double          iTriXBuffer[];
//--- iTriX 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- Triple Exponential Moving Averages Oscillator 指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,iTriXBuffer,INDICATOR_DATA);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iTriX)
```

© 2000-2025, MetaQuotes Ltd.

```
      handle=iTriX(name,period,ma_period,applied_price);
   else
     {
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[2];
      //--- 期間
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- 価格の種類
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_TRIX,2,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iTriX indicator for the symbol %s/
%s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- Triple Exponential Moving Averages Oscillator 指標が計算された銘柄/時間軸を表示
   short_name=StringFormat("iTriX(%s/%s, %d, %s)",name,EnumToString(period),
                           ma_period,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iTriX 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
```

© 2000-2025, MetaQuotes Ltd.

```
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
//--- これが指標計算の初めであるか iTriX 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- iTriXBuffer 配列が銘柄/期間でi TriX 指標の値の数より大きい場合、全体のコピーはしない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- これは初回の計算ではなく、
      //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- Triple Exponential Moving Averages Oscillator 指標の値で記入
//--- FillArrayFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArrayFromBuffer(iTriXBuffer,handle,values_to_copy)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- Triple Exponential Moving Averages Oscillator 指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iTriX 指標から指標バッファを記入する                                |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &trix_buffer[], // Triple Exponential Moving
Averages Oscillator 値の指標バッファ
                         int ind_handle,        // iTriX 指標ハンドル
                         int amount             // 複製された値の数
                        )
  {
//--- エラーコードをリセットする
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- インデックス0 を持つ指標バッファの値で iTriXBuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,0,amount,trix_buffer)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iTriX indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指標初期化解除関数                                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iWPR

この関数は Larry Williams' Percent Range（ラリーウィリアムパーセントレンジ）指標のハンドルを返します。バッファは 1 つです。

```
int  iWPR(
   string           symbol,          // 銘柄名
   ENUM_TIMEFRAMES  period,          // 期間
   int              calc_period      // 平均期間
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

calc_period

[in] 指標計算の平均期間（バーの数）

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって、コンピュータメモリを利用されていない指標から解放することが出来ます。

例：

```
//+------------------------------------------------------------------+
//|                                                    Demo_iWPR.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iWPR technical indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the
'type' parameter (function type)."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- i WPR プロット
#property indicator_label1  "iWPR"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrCyan
```

© 2000–2025, MetaQuotes Ltd.

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 指標値を制限する
#property indicator_minimum -100
#property indicator_maximum 0
//--- 指標ウィンドウの水平レベル
#property indicator_level1  -20.0
#property indicator_level2  -80.0
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iWPR,              // iWPRを使用する
   Call_IndicatorCreate    // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iWPR;           // 関数の種類
input int                   calc_period=14;           // 期間
input string                symbol=" ";               // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;    // 時間軸
//--- 指標バッファ
double          iWPRBuffer[];
//--- iWPR 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- ラリーウィリアムパーセントレンジ 指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,iWPRBuffer,INDICATOR_DATA);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削する
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 指標ハンドルを作成する
   if(type==Call_iWPR)
      handle=iWPR(name,period,calc_period);
   else
     {
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[1];
      //--- 期間
      pars[0].type=TYPE_INT;
      pars[0].integer_value=calc_period;
      handle=IndicatorCreate(name,period,IND_WPR,1,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iWPR indicator for the symbol %s/%
s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- ウィリアムパーセントレンジ 指標が計算された銘柄/時間軸を表示
   short_name=StringFormat("iWPR(%s/%s, %d)",name,EnumToString(period),calc_period);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iWPR 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
       return(0);
      }
//--- これが指標計算の初めであるか iWPR 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
      {
       //--- iiWPRBuffer 配列が銘柄/期間で iWPR 指標の値の数より大きい場合、全体のコピーはしない
       //--- 他の場合、指標バッファサイズより少ない量をコピーをする
       if(calculated>rates_total) values_to_copy=rates_total;
       else                       values_to_copy=calculated;
      }
   else
      {
       //--- これは初回の計算ではなく、
       //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
       values_to_copy=(rates_total-prev_calculated)+1;
      }
//---配列をウィリアムパーセントレンジ 指標の値で記入
//--- FillArrayFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArrayFromBuffer(iWPRBuffer,handle,values_to_copy)) return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- ラリーウィリアムパーセントレンジ 指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iWPR 指標から指標バッファを記入する                                |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &wpr_buffer[],  // ウィリアムパーセントレンジ 値の指標バッファ
                         int ind_handle,        // iWPR 指標ハンドル
                         int amount             // 複製された値の数
                        )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で iWPRBuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,0,amount,wpr_buffer)<0)
      {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iWPR indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指標初期化解除関数                                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000–2025, MetaQuotes Ltd.

iVIDyA

この関数は Variable Index Dynamic Average 指標ハンドルを返します。バッファは 1 つです。

```
int  iVIDyA(
   string               symbol,           // 銘柄名
   ENUM_TIMEFRAMES      period,           // 期間
   int                  cmo_period,       // Chandeモメンタムの期間 Momentum
   int                  ema_period,       // EMA 平滑期間
   int                  ma_shift,         // 価格チャートでの水平シフト
   ENUM_APPLIED_PRICE   applied_price     // 価格の種類かハンドル
   );
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

cmo_period

[in] Chande モメンタムオシレーター計算の期間（バーの数）

ema_period

[in] 平滑化係数計算の EMA 期間（バーの数）

ma_shift

[in] 価格チャートに相対した指標のシフト

applied_price

[in] 使用される価格。ENUM_APPLIED_PRICE 価格定数のいずれか、または別の指標ハンドル。

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって、コンピュータメモリを利用されていない指標から解放することが出来ます。

例：

```
//+------------------------------------------------------------------+
//|                                                  Demo_iVIDyA.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iVIDyA technical indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
```

© 2000–2025, MetaQuotes Ltd.

```
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the
'type' parameter (function type)."
#property description "All other parameters like in the standard Variable Index
Dynamic Average."

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iVIDyA プロット
#property indicator_label1  "iVIDyA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iVIDyA,             // iVIDyA を使用する
   Call_IndicatorCreate     // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iVIDyA;           // 関数の種類
input int                   cmo_period=15;              // Chande モメンタム期間
input int                   ema_period=12;              // 平滑化係数の期間
input int                   ma_shift=0;                 // シフト
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // 価格の種類
input string                symbol=" ";                 // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;      // 時間軸
//--- 指標バッファ
double          iVIDyABuffer[];
//--- iVIDyA 指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- Variable Index Dynamic Average 指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,iVIDyABuffer,INDICATOR_DATA);
//--- シフトを設定
```

© 2000-2025, MetaQuotes Ltd.

```
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iVIDyA)
      handle=iVIDyA(name,period,cmo_period,ema_period,ma_shift,applied_price);
   else
     {
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[4];
      //--- Chande モメンタム期間
      pars[0].type=TYPE_INT;
      pars[0].integer_value=cmo_period;
      //--- 平滑化係数の期間
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ema_period;
      //--- シフト
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_shift;
      //--- 価格の種類
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_VIDYA,4,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iVIDyA indicator for the symbol %
s/%s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- Variable Index Dynamic Average 指標が計算された銘柄/時間軸を表示
   short_name=StringFormat("iVIDyA(%s/%s, %d, %d, %d, %
s)",name,EnumToString(period),
```

© 2000-2025, MetaQuotes Ltd.

```
                              cmo_period,ema_period,ma_shift,EnumToString(applied_price
));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iVIDyA 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
     }
//--- これが指標計算の初めであるか iVIDyA 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
     {
      //--- iVIDyABuffer 配列が銘柄/期間でi VIDyA 指標の値の数より大きい場合、全体のコピーはしない
      //--- 他の場合、指標バッファサイズより少ない量をコピーをする
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- これは初回の計算ではなく、
      //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- 配列を Variable Index Dynamic Average 指標の値で記入
//--- FillArrayFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArrayFromBuffer(iVIDyABuffer,ma_shift,handle,values_to_copy)) return(0);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- Variable Index Dynamic Average 指標の値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iVIDyA 指標から指標バッファを記入する                                |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &vidya_buffer[],// Variable Index Dynamic Average 値の指標バッファ
                         int v_shift,           // 線のシフト
                         int ind_handle,        // iVIDyA 指標ハンドル
                         int amount             // 複製された値の数
                        )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で iVIDyABuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,-v_shift,amount,vidya_buffer)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iVIDyA indicator, error code %d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指標初期化解除関数                                                |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iVolumes

この関数は Volumes（ボリューム）指標のハンドルを返します。バッファは 1 つです。

```
int  iVolumes(
   string               symbol,             // 銘柄名
   ENUM_TIMEFRAMES      period,             // 期間
   ENUM_APPLIED_VOLUME  applied_volume      // 計算に使用するボリュームの種類
  )
```

パラメータ

symbol

[in] データが指標計算に使用される有価証券の銘柄名（NULL は現在のシンボル）

period

[in] 期間の値は ENUM_TIMEFRAMES 列挙の値の 1 つで、0 は現在の時間軸の意味です。

applied_volume

[in] 使用されるボリューム（ENUM_APPLIED_VOLUME 値のいずれか）

戻り値

指定されたテクニカル指標ハンドル。失敗の場合 INVALID_HANDLE。IndicatorRelease() 関数に指標ハンドルを渡すことによって、コンピュータメモリを利用されていない指標から解放することが出来ます。

例：

```
//+------------------------------------------------------------------+
//|                                                 Demo_iVolumes.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The indicator demonstrates how to obtain data"
#property description "of indicator buffers for the iVolumes technical indicator."
#property description "A symbol and timeframe used for calculation of the
indicator,"
#property description "are set by the symbol and period parameters."
#property description "The method of creation of the handle is set through the
'type' parameter (function type)."

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- iVolumes プロット
#property indicator_label1  "iVolumes"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen, clrRed
#property indicator_style1  STYLE_SOLID
```

© 2000–2025, MetaQuotes Ltd.

```
#property indicator_width1  1
//+------------------------------------------------------------------+
//|  作成を処理する方法の列挙                                        |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iVolumes,           // iVolumesを使用する
   Call_IndicatorCreate     // IndicatorCreateを使用する
  };
//--- 入力パラメータ
input Creation              type=Call_iVolumes;           // 関数の種類
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;   // ボリュームの種類
input string                symbol=" ";                   // シンボル
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // 時間軸
//--- 指標バッファ
double          iVolumesBuffer[];
double          iVolumesColors[];
//--- iVolumes  指標ハンドルを格納する変数
int     handle;
//--- 格納に使用される変数
string name=symbol;
//--- チャートでの指標名
string short_name;
//--- ボリューム指標に値の数を保存
int     bars_calculated=0;
//+------------------------------------------------------------------+
//|  カスタム指標を初期化する関数                                    |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 配列の指標バッファへの割り当て
   SetIndexBuffer(0,iVolumesBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iVolumesColors,INDICATOR_COLOR_INDEX);
//--- 指標が描画するシンボルを決める
   name=symbol;
//--- 左右のスペースを削る
   StringTrimRight(name);
   StringTrimLeft(name);
//---「name」文字列の長さがゼロになった場合
   if(StringLen(name)==0)
     {
      //--- 指標が接続されているチャートのシンボルを取る
      name=_Symbol;
     }
//--- 指標ハンドルを作成する
   if(type==Call_iVolumes)
      handle=iVolumes(name,period,applied_volume);
   else
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 構造体を指標のパラメータで記入
      MqlParam pars[1];
      //--- 価格の種類
      pars[0].type=TYPE_INT;
      pars[0].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_VOLUMES,1,pars);
     }
//--- ハンドルが作成されなかった場合
   if(handle==INVALID_HANDLE)
     {
      //--- 失敗した事実とエラーコードを出力する
      PrintFormat("Failed to create handle of the iVolumes indicator for the symbol
%s/%s, error code %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 指標が早期に中止された
      return(INIT_FAILED);
     }
//--- ボリューム指標が計算された銘柄/時間軸を表示
   short_name=StringFormat("iVolumes(%s/%s, %
s)",name,EnumToString(period),EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 通常の指標の初期化
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| カスタム指標の反復関数                                           |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iVolumes 指標から複製された値の数
   int values_to_copy;
//--- 指標で計算された値の数を決める
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %
d",calculated,GetLastError());
      return(0);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- これが指標計算の初めであるか iVolumes 指標の値の数が変更した
//---または、2 つ以上のバーの指標の計算が必要である場合（価格履歴で何かが変更された）
   if(prev_calculated==0 || calculated!=bars_calculated ||
rates_total>prev_calculated+1)
      {
       //--- iVolumesBuffer 配列が銘柄/期間で iVolumes 指標の値の数より大きい場合、全体のコピーはし
ない
       //--- 他の場合、指標バッファサイズより少ない量をコピーをする
       if(calculated>rates_total) values_to_copy=rates_total;
       else                       values_to_copy=calculated;
      }
   else
      {
       //--- これは初回の計算ではなく、
       //--- 前回の OnCalculate() から、一以上のバーが加えられてない。
       values_to_copy=(rates_total-prev_calculated)+1;
      }
//--- 配列を iVolumes 指標の値で記入
//--- FillArraysFromBuffer が false を返した場合、情報の準備が終わっていないので操作を終了する
   if(!FillArraysFromBuffers(iVolumesBuffer,iVolumesColors,handle,values_to_copy))
return(0);
//--- メッセージを形成する
   string comm=StringFormat("%s ==>  Updated value in the indicator %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- チャートにサービスメッセージを表示する
   Comment(comm);
//--- ボリューム指標値の数を覚える
   bars_calculated=calculated;
//--- prev_calculated 値を次の関数呼び出しのために返す
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iVolumes 指標から指標バッファを記入する                          |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &volume_buffer[],   // ボリューム値の指標バッファ
                           double &color_buffer[],    // 色の指標バッファ
                           int ind_handle,            // ボリューム指標ハンドル
                           int amount                 // 複製された値の数
                          )
  {
//--- エラーコードをリセットする
   ResetLastError();
//--- インデックス0 を持つ指標バッファの値で iVolumesBuffer 配列の一部を記入する
   if(CopyBuffer(ind_handle,0,0,amount,volume_buffer)<0)
      {
       //--- 複製が失敗したら、エラーコードを出す
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("Failed to copy data from the iVolumes indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- インデックス 1 を持つ指標バッファの値で iVolumesColors 配列の一部を記入する
   if(CopyBuffer(ind_handle,1,0,amount,color_buffer)<0)
     {
      //--- 複製が失敗したら、エラーコードを出す
      PrintFormat("Failed to copy data from the iVolumes indicator, error code %
d",GetLastError());
      //--- ゼロ結果で終了。 指標は計算されていないと見なされる
      return(false);
     }
//--- 全てが成功
   return(true);
  }
//+------------------------------------------------------------------+
//| 指標初期化解除関数                                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 指標の削除後チャートをクリアする
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

最適化の結果の操作

これらは、ストラテジーテスターで最適化の結果のカスタム処理を整理するための関数です。テストエージェントでの最適化中のみでなく、エキスパートアドバイザーとスクリプトでの局所的な呼び出しも可能です。

ストラテジーテスターでエキスパートアドバイザーを実行する際には、基本データ型または（文字列、クラスオブジェクト及び動的配列オブジェクトを含まない）基本構造体 のデータ配列を作成することが出来ます。このデータは FrameAdd() 関数で「フレーム」と呼ばれる特別な構造体に保存されます。エキスパートアドバイザーの最適化中には、それぞれのエージェントが端末に一連のフレームを送信します。全ての受信されたフレームは、terminal_directory/MQL5/Files/Tester フォルダでエキスパートアドバイザーとして *.MQD に書かれます。これらは、エージェントから受信された順序で書かれています。テストエージェントからクライアント端末におけるフレームの受信は TesterPass イベントを作成します。

フレームは、コンピュータメモリ内及び指定された名称のファイルに格納することが出来ます。MQL5 言語ではフレーム数に制限はありません。

MQL5クラウドネットワークのメモリとディスク容量の制限

次の制限は、MQL5クラウドネットワークで実行される最適化に適用されます: エキスパートアドバイザー(EA)は、4GBを超える情報をディスクに書き込んだり、4GBを超えるRAMを使用したりしてはなりません 。上限を超えると、ネットワークエージェントは計算を正しく完了できず、結果を受け取ることができなくなります。ただし、計算に費やした時間はすべて料金が発生します。

各最適化パスから情報を取得する必要がある場合は、ディスクに書き込まずにフレームを送信します。MQL5クラウドネットワークでの計算中にEAでのファイル操作の使用を回避するには、次のチェックを使用できます。

```
int handle=INVALID_HANDLE;
bool file_operations_allowed=true;
if(MQLInfoInteger(MQL_OPTIMIZATION) || MQLInfoInteger(MQL_FORWARD))
   file_operations_allowed=false;

if(file_operations_allowed)
  {
   ...
   handle=FileOpen(...);
   ...
  }
```

関数	アクション
FrameFirst	フレーム読み込みのポインタを先頭に移動し、以前に設定したフィルタをリセットします。
FrameFilter	フレーム読み込みフィルタを設定して、ポインタを先頭に移動します。
FrameNext	フレームを読んでポインタを次に移動します。
FrameInputs	フレーム形成に使用される入力パラメータを受け取ります。
FrameAdd	データとフレームを追加します。

© 2000–2025, MetaQuotes Ltd.

関数	アクション
ParameterGetRange	ストラテジーテスターでエキスパートアドバイザーを最適化する際に、値範囲のデータを受け取り入力変数のステップを変更します。
ParameterSetRange	ストラテジーテスターでエキスパートアドバイザーを最適化する際に、入力変数（値、チェンジステップ、最初と最終の値）の使い方を指定します。

参照

テストの統計、MQL5 プログラム実行のプロパティ

© 2000–2025, MetaQuotes Ltd.

FrameFirst

フレーム読み込みのポインタを先頭に移動し、以前に設定したフィルタをリセットします。

```
bool  FrameFirst();
```

戻り値

成功の場合は true、それ以外の場合は false 。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

© 2000–2025, MetaQuotes Ltd.

FrameFilter

フレーム読み込みフィルタを設定して、ポインタを先頭に移動します。

```
bool  FrameFilter(
   const string  name,          // Public 名称/レベル
   long          id             // パブリック ID
   );
```

戻り値

成功の場合は true、それ以外の場合は false 。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

注意事項

最初のパラメータとして空の文字列が渡された場合、フィルタは数値パラメータでのみ機能し、指定された ID を持ったフレームのみが表示されます。2 番目のパラメータが ULONG_MAX の場合、テキストフィルタのみが機能します。

FrameFilter("", ULONG_MAX) の呼び出しは FrameFirst() と同一、つまりフィルタを使用しません。

© 2000–2025, MetaQuotes Ltd.

FrameNext

フレームを読んでポインタを次に移動します。この関数には 2 つのバージョンがあります。

1. 1 つの数値を受け取る

```
bool  FrameNext(
   ulong&   pass,       // フレームが追加された最適化でのパスの番号
   string&  name,       // Public 名称・レベル
   long&    id,         // パブリック ID
   double&  value       // 値
   );
```

2. フレームの全てのデータを受け取る

```
bool  FrameNext(
   ulong&   pass,       // フレームが追加された最適化でのパスの番号
   string&  name,       // Public 名称・レベル
   long&    id,         // パブリック ID
   double&  value,      // 値
   void&    data[]      // 任意型の配列
   );
```

パラメータ

pass

[out] ストラテジーテスターでの最適化のパスの数

name

[out] 識別子の名称

id

[out] 識別子の値

value

[out] 単一の数値

data

[out] 任意型の配列

戻り値

成功の場合は true、それ以外の場合は false 。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

注意事項

2 番目のバージョンの呼び出しでは受信されたデータを *data[]* 配列で正しく処理する必要があります。

© 2000-2025, MetaQuotes Ltd.

FrameInputs

指定されたパス番号を持つフレームを形成する入力パラメータを受け取ります。

```
bool  FrameInputs(
   ulong    pass,                // 最適化でのパスの数
   string&  parameters[],        // 「parameterN=valueN」形式を持つ文字列の配列
   uint&    parameters_count     // パラメータ数
   );
```

パラメータ

pass

[in] ストラテジーテスターでの最適化のパスの数

parameters

[out] 名称ととパラメータ値の記述のための文字列の配列

parameters_count

[out] *parameters[]* 配列の要素数

戻り値

成功の場合は true、それ以外の場合は false 。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

注意事項

parameters[] 配列の *parameters_count* で文字列の数を取得することによって、全てのレコードを通過するループを整理することが出来ます。これにより、エキスパートアドバイザーの指定されたパス数の入力パラメータの値を見つけられます。

© 2000-2025, MetaQuotes Ltd.

FrameAdd

データとフレームを追加します。この関数には 2 つのバージョンがあります。

1. ファイルからのデータの追加

```
bool  FrameAdd(
   const string  name,        // Public 名称・レベル
   long          id,          // パブリック ID
   double        value,       // 値
   const string  filename     // データファイル名
   );
```

2. 任意型の配列からのデータの追加

```
bool  FrameAdd(
   const string  name,        // Public 名称・レベル
   long          id,          // パブリック ID
   double        value,       // 値
   const void&   data[]       // 任意型の配列
   );
```

パラメータ

name

[in] Public フレームラベル。FrameFilter() 関数のフィルタ内で利用出来ます。

id

[in] public フレーム識別子。FrameFilter() 関数のフィルタ内で利用出来ます。

value

[in] フレームに書き込む数値。OnTester() 関数でのように単一パスの結果を送信するために使用されます。

filename

[in] フレームに追加するデータを含むファイルの名称。このファイルは MQL5/Files に位置しなければなりません。

data

[in] フレームに書き込む任意の型の配列。参照で引き渡されます。

戻り値

成功の場合は true、それ以外の場合は false 。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

© 2000–2025, MetaQuotes Ltd.

ParameterGetRange

ストラテジーテスターでエキスパートアドバイザーを最適化する際に、値範囲のデータを受け取り入力変数のステップを変更します。この関数には 2 つのバージョンがあります。

1. 整数型入力パラメータのデータを受け取る

```
bool  ParameterGetRange(
   const string  name,           // パラメータ(入力変数)名
   bool&         enable,         // パラメータ最適化が有効
   long&         value,          // パラメータ値
   long&         start,          // 初期値
   long&         step,           // ステップ変更
   long&         stop            // 最終値
   );
```

2. real 型入力パラメータのデータを受け取る

```
bool  ParameterGetRange(
   const string  name,           // パラメータ(入力変数)名
   bool&         enable,         // パラメータ最適化が有効
   double&       value,          // パラメータ値
   double&       start,          // 初期値
   double&       step,           // ステップ変更
   double&       stop            // 最終値
   );
```

パラメータ

name

[in] 入力変数 ID。これらの変数は、アプリケーションの外部パラメータです。チャート起動中または単一のテスト中での指定が可能です。

enable

[out] このパラメータがストラテジーテスターで最適化の際に値を列挙するために使用することが出来るというフラグ

value

[out] パラメータ値

start

[out] 最適化中の初期のパラメータ値

step

[out] 値の列挙中のパラメータ変更ステップ

stop

[out] 最適化中の最終パラメータ値

戻り値

© 2000–2025, MetaQuotes Ltd.

成功の場合は true、それ以外の場合は false 。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

注意事項

この関数は OnTesterInit()、OnTesterPass() 及び OnTesterDeinit() ハンドラーのみから呼び出されます。これは、エキスパートアドバイザーの入力パラメータ値を受信するために導入され、変動はストラテジーテスターで最適化中に範囲されています。

OnTesterInit()で呼び出された場合、取得されたデータは、任意の入力変数列挙のためのルールの ParameterSetRange() 関数を使用しての再定義に使用することが出来ます。そのため、新たな Start、Stop 及び Step 値を設定することができ、ストラテジーテスターの設定にかかわらず、入力パラメータを完全に最適化から除外することも出来ます。なので、エキスパートアドバイザーの重要なパラメータの値に応じて最適化からいくつかのパラメータを除外することによって最適化の際に入力パラメータのエリアを管理することが出来ます。

例:

```
#property description "Expert Advisor for ParameterGetRange() function
demonstration."
#property description "Should be launched in the optimization mode of the Strategy
Tester"
//--- 入力パラメータ
input int                  Input1=1;
input double               Input2=2.0;
input bool                 Input3=false;
input ENUM_DAY_OF_WEEK     Input4=SUNDAY;

//+------------------------------------------------------------------+
//| エキスパート初期化に使用される関数                                |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- エキスパートアドバイザーはストラテジーテスター内のみで動作するように設計されている
   if(!MQL5InfoInteger(MQL5_OPTIMIZATION))
     {
      MessageBox("Should be launched in the optimization mode of the Strategy
Tester!");
      //--- 事前にエキスパートアドバイザー操作を終了してチャートから削除する
      return(INIT_FAILED);
     }
//--- 初期化が正常に完了した
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| TesterInit 関数                                                  |
//+------------------------------------------------------------------+
void OnTesterInit()
  {
//--- long 型入力パラメータの例
   string name="Input1";
   bool enable;
```

© 2000-2025, MetaQuotes Ltd.

```
   long par1,par1_start,par1_step,par1_stop;
   ParameterGetRange(name,enable,par1,par1_start,par1_step,par1_stop);
   Print("First parameter");
   PrintFormat("%s=%d  enable=%s  from %d to %d with step=%d",
               name,par1,(string)enable,par1_start,par1_stop,par1_step);
//--- double 型入力パラメータの例
   name="Input2";
   double par2,par2_start,par2_step,par2_stop;
   ParameterGetRange(name,enable,par2,par2_start,par2_step,par2_stop);
   Print("Second parameter");
   PrintFormat("%s=%G  enable=%s  from %G to %G with step=%G",
パブリック                 name,par2,(string)enable,par2_start,par2_stop,par2_step);

//--- bool 型入力パラメータの例
   name="Input3";
   long par3,par3_start,par3_step,par3_stop;
   ParameterGetRange(name,enable,par3,par3_start,par3_step,par3_stop);
   Print("Third parameter");
   PrintFormat("%s=%s  enable=%s  from %s to %s",
               name,(string)par3,(string)enable,
               (string)par3_start,(string)par3_stop);
//--- 列挙型入力パラメータの例
   name="Input4";
   long par4,par4_start,par4_step,par4_stop;
   ParameterGetRange(name,enable,par4,par4_start,par4_step,par4_stop);
   Print("Fourth parameter");
   PrintFormat("%s=%s  enable=%s  from %s to %s",
               name,EnumToString((ENUM_DAY_OF_WEEK)par4),(string)enable,
               EnumToString((ENUM_DAY_OF_WEEK)par4_start),
               EnumToString((ENUM_DAY_OF_WEEK)par4_stop));
  }
//+------------------------------------------------------------------+
//| TesterDeinit 関数                                                |
//+------------------------------------------------------------------+
void OnTesterDeinit()
  {
//--- このメッセージは最適化が完了した後に表示されます
   Print(__FUNCTION__," Optimization completed");
  }
```

© 2000–2025, MetaQuotes Ltd.

ParameterSetRange

ストラテジーテスターでエキスパートアドバイザーを最適化する際に、入力変数値、チェンジステップ、最初と最終の値）の使い方を指定します。この関数には 2 つのバージョンがあります。

1. 整数型入力パラメータの値を指定する

```
bool  ParameterSetRange(
   const string  name,           // パラメータ(入力変数)名
   bool          enable,         // パラメータ最適化が有効
   long          value,          // パラメータ値
   long          start,          // 初期値
   long          step,           // ステップ変更
   long          stop            // 最終値
   );
```

2. real 型入力パラメータの値を指定する

```
bool  ParameterSetRange(
   const string  name,           // パラメータ(入力変数)名
   bool          enable,         // パラメータ最適化が有効
   double        value,          // パラメータ値
   double        start,          // 初期値
   double        step,           // ステップ変更
   double        stop            // 最終値
   );
```

パラメータ

name

[in] input または sinput 変数 ID。これらの変数は、アプリケーションの外部パラメータです。このパラメータの値はプログラム起動時に指定することが出来ます。

enable

[in] ストラテジーテスターで最適化の際に値を列挙するために使用ためにはこのパラメータを有効にします。

value

[in] パラメータ値

start

[in] 最適化中の初期のパラメータ値

step

[in] 値の列挙中のパラメータ変更ステップ

stop

[in] 最適化中の最終パラメータ値

戻り値

© 2000-2025, MetaQuotes Ltd.

成功の場合は true、それ以外の場合は false 。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

注意事項

この関数はストラテジーテスターから最適化を軌道する際に OnTesterInit() ハンドラからのみ呼ばれます。これは、パラメータの範囲及び変更のステップを指定するために設計されています。ストラテジーテスターの設定にかかわらず、パラメータは完全に最適化から除外することも出来ます。また、最適化プロセスにおけるsinput 修飾子で宣言された変数を使用も可能にします。

ParameterSetRange() 関数を使用すると、最適化から必要な入力パラメータを含むまたは除き、必要な範囲及び変更ステップを設定することにより、その主要なパラメータの値に応じてストラテジーテスターでエキスパートアドバイザーの最適化を管理することが出来ます。

© 2000–2025, MetaQuotes Ltd.

イベント関数

このグループには、カスタムイベントとタイマーイベントを操作するための関数が含まれています。このグループの他に、定義済みイベントを処理する特別関数が存在します。

関数	アクション
EventSetMillisecondTimer	現在のチャートで使用される、期間が1秒未満の高分解能タイマーのイベントジェネレータを起動します。
EventSetTimer	現在のチャートで使用される、指定された時間軸のタイマーイベントジェネレータを開始します。
EventKillTimer	現在のチャートのタイマーでイベントが発生するのを停止します。
EventChartCustom	指定されたチャートのカスタムイベントを生成します。

参照

チャートイベントの種類

© 2000–2025, MetaQuotes Ltd.

EventSetMillisecondTimer

この関数はクライアント端末にこのエキスパートアドバイザーまたは指標のためのタイマー イベントは1秒未満の間隔で生成されるべきだと示します。

```
bool  EventSetMillisecondTimer(
   int  milliseconds      // ミリ秒数
   );
```

パラメータ

milliseconds

[in] タイマーイベントの頻度を定義するミリ秒数

戻り値

実行成功の場合は true、それ以外の場合は falseエラーコードを受け取るには GetLastError() 関数が呼ばれるべきです。

注意事項

この機能は、高分解能タイマーが必要な場合のために設計されています。言い換えると、タイマーイベントは1秒に1回よりも頻繁に受信する必要があります。1秒以上の時間軸を持つ従来のタイマーが充分である場合 EventSetTimer() が使用されます。

一般的に、タイマー期間が減少する際にはタイマーイベントのハンドラがより頻繁に呼び出されるようにテスト時間が増加します。リアルタイムモードで操作している場合、タイマーイベントは、ハードウェアの制約のために10～16ミリ秒に1回以下で生成されます。

通常、この関数は OnInit() 関数またはクラスコンストラクタで呼ばれるべきです。タイマーからのイベントを処理するには、エキスパートアドバイザーまたは指標は OnTimer() 関数を持つべきです。

それぞれのエキスパートアドバイザーと指標は独自のタイマーを操作しそのタイマーのみからイベントを受信します。MQL5 プログラムが停止した場合、作成されて EventKillTimer() 関数で無効にされていないタイマーは強制的に破壊されます。

1 つのプログラムでは 1 つのタイマーのみが起動出来ます。各MQL5アプリケーションとチャートは、新たに到着したイベントが配置される独自のイベントキューを持っています。キューにすでにタイマーイベントがある、またはこのイベントが処理段階にある場合、新しいタイマーイベントは MQL5 アプリケーションキューには追加されません。

© 2000–2025, MetaQuotes Ltd.

EventSetTimer

この関数はクライアント端末にこのエキスパートアドバイザーまたは指標のためのタイマーイベントは指定された時間軸で生成するべきだと示します。

```
bool  EventSetTimer(
   int  seconds      // 秒数
   );
```

パラメータ

seconds

[in] タイマーイベントの発生頻度を決定する秒数

戻り値

成功の場合は true、それ以外の場合は false。エラーコードを受け取るには GetLastError() 関数が呼ばれるべきです。

注意事項

通常、この関数は OnInit() 関数またはクラスコンストラクタで呼ばれるべきです。タイマーからくるイベントを処理するために，エキスパートアドバイザーはOnTimer() 関数を持つべきです。

エキスパートアドバイザーならびに指標は、全て独自のタイマーで動作しそこからのみイベントを受けとります。作成されたタイマーは EventKillTimer() 関数で無効にされていない限り MQL5 プログラムの操作の終了時に強制的に破壊されます。

プログラムでは複数のタイマーを実行することは出来ません。各MQL5アプリケーションとチャートは、新たに到着したイベントが配置されるイベントの独自のキューを持っています。タイマーイベントがキューに存在するまたは処理中の場合、新しいタイマーイベントは MQL5 プログラムのキューに加えられません。

© 2000–2025, MetaQuotes Ltd.

EventKillTimer

Timer イベントの発生を停止する必要があるクライアント端末を指定します。

```
void  EventKillTimer();
```

戻り値

なし

注意事項

この関数は、一般的に、EventSetTimer() 関数がOnInit() 関数から呼ばれた場合、OnDeinit() 関数から呼ばれなければなりません。EventSetTimer() 関数がクラスのコンストラクタで呼び出されている場合、この関数はデストラクタからも呼び出せます。

エキスパートアドバイザーならびに指標は、全て独自のタイマーで動作しそこからのみイベントを受けとります。作成されたタイマーは EventKillTimer() 関数で無効にされていない限り MQL5 プログラムの操作の終了時に強制的に破壊されます。

© 2000–2025, MetaQuotes Ltd.

EventChartCustom

この関数は指定されたチャートのカスタムイベントを生成します。

```
bool  EventChartCustom(
   long    chart_id,            // イベントを受け取るチャートの識別子
   ushort  custom_event_id,     // イベント識別子
   long    lparam,              // long 型のパラメータ
   double  dparam,              // double型のパラメータ
   string  sparam               // イベントの string パラメータ
   );
```

パラメータ

chart_id

[in] チャート識別子（0は現在のチャート）

custom_event_id

[in] ユーザイベントの ID。この識別子は自動的に CHARTEVENT_CUSTOM 値に足され、整数型に変換されます。

lparam

[in] OnChartEvent 関数に渡される long 型のイベントパラメータ

dparam

[in] OnChartEvent 関数に渡される double 型のイベントパラメータ

sparam

[in] OnChartEvent 関数に渡される string 型のイベントパラメータ（文字列が 63 文字より長い場合は切り捨てられます）

戻り値

カスタムイベントがイベントを受信するチャートのイベントキューに正常に配置された場合は true を返します。失敗の場合 false を返します。エラーコードを取得するには GetLastError() を使用します。

注意事項

エキスパートアドバイザや特定のチャートに付属する指標は OnChartEvent(int event_id, long& lparam, double& dparam, string& sparam) を使用してイベントを処理します。

OnChartEvent() 関数の入力パラメータはイベントの種類ごとにイベント処理に必要とされる一定の値を有しています。パラメータで受け渡しされるイベントと値は以下の表に記載されています。

イベント	id パラメータ値	lparam パラメータ値	dparam パラメータ値	sparam パラメータ値
キーストロークイベント	CHARTEVENT_KEYDOWN	押されたキーのコード	繰り返し回数（ユーザのキーの長押しがキーストロークの繰り返しとみなされた場合の繰り返し回数）	キーボードボタンの状態を記述するビットマスクの文字列値

© 2000–2025, MetaQuotes Ltd.

イベント	id パラメータ値	lparam パラメータ値	dparam パラメータ値	sparam パラメータ値
マウスイベント（チャートに CHART_EVENT_MOUSE_MOVE=true プロパティが設定されている場合）	CHARTEVENT_MOUSE_MOVE	X 座標	Y 座標	マウスボタンの状態を記述するビットマスクの文字列の値
グラフィックオブジェクトの作成イベント（CHART_EVENT_OBJECT_CREATE=true がチャートに設定された場合）	CHARTEVENT_OBJECT_CREATE	—	—	作成したグラフィックオブジェクトの名称
プロパティダイアログでのオブジェクトプロパティ変更イベント	CHARTEVENT_OBJECT_CHANGE	—	—	変更したグラフィックオブジェクトの名称
グラフィックオブジェクト削除イベント（チャートに CHART_EVENT_OBJECT_DELETE=true が設定された場合）	CHARTEVENT_OBJECT_DELETE	—	—	削除したグラフィックオブジェクトの名称
チャート上でのマウスクリックイベント	CHARTEVENT_CLICK	X 座標	Y 座標	—
チャートに属するグラフィックオブジェクトでのマウスクリックイベント	CHARTEVENT_OBJECT_CLICK	X 座標	Y 座標	イベントが発生したグラフィック·オブジェクトの名称
マウスを使用してグラフィックオブジェクトのドラッグイベント	CHARTEVENT_OBJECT_DRAG	—	—	移動したグラフィックオブジェクトの名称
LabelEdit グラフィックオブジェクトの入力ボックスでのテキスト編集完成のイベント	CHARTEVENT_OBJECT_ENDEDIT	—	—	テキスト編集が完成した LabelEdit グラフィックオブジェクトの名称

© 2000–2025, MetaQuotes Ltd.

イベント	id パラメータ値	lparam パラメータ値	dparam パラメータ値	sparam パラメータ値
チャートでの変更イベント	CHARTEVENT_CHART_CHANGE	—	—	—
N 番号を使用したユーザイベント ID	CHARTEVENT_CUSTOM+N	EventChartCustom() 関数で設定された値	EventChartCustom() 関数で設定された値	EventChartCustom() 関数で設定された値

例：

```
//+------------------------------------------------------------------+
//|                                              ButtonClickExpert.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                              https://www.MQL5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

string buttonID="Button";
string labelID="Info";
int broadcastEventID=5000;
//+------------------------------------------------------------------+
//| エキスパート初期化に使用される関数                                |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- カスタムイベントを送信するボタンを作成する
   ObjectCreate(0,buttonID,OBJ_BUTTON,0,100,100);
   ObjectSetInteger(0,buttonID,OBJPROP_COLOR,clrWhite);
   ObjectSetInteger(0,buttonID,OBJPROP_BGCOLOR,clrGray);
   ObjectSetInteger(0,buttonID,OBJPROP_XDISTANCE,100);
   ObjectSetInteger(0,buttonID,OBJPROP_YDISTANCE,100);
   ObjectSetInteger(0,buttonID,OBJPROP_XSIZE,200);
   ObjectSetInteger(0,buttonID,OBJPROP_YSIZE,50);
   ObjectSetString(0,buttonID,OBJPROP_FONT,"Arial");
   ObjectSetString(0,buttonID,OBJPROP_TEXT,"Button");
   ObjectSetInteger(0,buttonID,OBJPROP_FONTSIZE,10);
   ObjectSetInteger(0,buttonID,OBJPROP_SELECTABLE,0);

//--- 情報を表示するラベルを作成する
   ObjectCreate(0,labelID,OBJ_LABEL,0,100,100);
   ObjectSetInteger(0,labelID,OBJPROP_COLOR,clrRed);
   ObjectSetInteger(0,labelID,OBJPROP_XDISTANCE,100);
   ObjectSetInteger(0,labelID,OBJPROP_YDISTANCE,50);
   ObjectSetString(0,labelID,OBJPROP_FONT,"Trebuchet MS");
   ObjectSetString(0,labelID,OBJPROP_TEXT,"No information");
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(0,labelID,OBJPROP_FONTSIZE,20);
   ObjectSetInteger(0,labelID,OBJPROP_SELECTABLE,0);

//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| エキスパート初期化解除に使用される関数                                  |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   ObjectDelete(0,buttonID);
   ObjectDelete(0,labelID);
  }
//+------------------------------------------------------------------+
//| エキスパートティック関数                                           |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- マウスボタンを押してイベントをチェックする
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      string clickedChartObject=sparam;
      //--- buttonID の名称のオブジェクトをクリックした場合
      if(clickedChartObject==buttonID)
        {
         //--- ボタンの状態—押されたか放された
         bool selected=ObjectGetInteger(0,buttonID,OBJPROP_STATE);
         //--- デバッグメッセージをログする
         Print("Button pressed = ",selected);
         int customEventID; // 送信するカスタムイベントの数
         string message;    // イベントで送られるメッセージ
         //--- ボタンが押された場合
         if(selected)
           {
            message="Button pressed";
            customEventID=CHARTEVENT_CUSTOM+1;
           }
         else // ボタンが押されていない
```

© 2000–2025, MetaQuotes Ltd.

```
            {
             message="Button in not pressed";
             customEventID=CHARTEVENT_CUSTOM+999;
            }
          //--- カスタムイベントを「自分の」チャートに送る
          EventChartCustom(0,customEventID-CHARTEVENT_CUSTOM,0,0,message);
          ///--- メッセージを全てのチャートに送る
          BroadcastEvent(ChartID(),0,"Broadcast Message");
          //--- デバッグメッセージ
          Print("Sent an event with ID = ",customEventID);
         }
       ChartRedraw();// チャートオブジェクトの強制再描画
      }

//--- イベントがユーザイベントだとチェック
    if(id>CHARTEVENT_CUSTOM)
      {
       if(id==broadcastEventID)
         {
          Print("Got broadcast message from a chart with id = "+lparam);
         }
       else
         {
          //--- イベントのテキストメッセージを読む
          string info=sparam;
          Print("Handle the user event with the ID = ",id);
          //--- メッセージをラベルで表示する
          ObjectSetString(0,labelID,OBJPROP_TEXT,sparam);
          ChartRedraw();// チャートオブジェクトの強制再描画
         }
      }
  }
//+------------------------------------------------------------------+
//|  全ての開いているチャートにブロードキャストイベントを送信                                  |
//+------------------------------------------------------------------+
void BroadcastEvent(long lparam,double dparam,string sparam)
  {
   int eventID=broadcastEventID-CHARTEVENT_CUSTOM;
   long currChart=ChartFirst();
   int i=0;
   while(i<CHARTS_MAX)                  // 開いているチャートは確実に CHARTS_MAX 以内
     {
      EventChartCustom(currChart,eventID,lparam,dparam,sparam);
      currChart=ChartNext(currChart); // 前に新しいチャートを受け取った
      if(currChart==-1) break;         // チャートリストの最後に達した
      i++;// カウンタを増加させることを忘れない
     }
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

参照

クライアント端末イベント、イベント処理関数

© 2000–2025, MetaQuotes Ltd.

OpenCL 操作

OpenCLプログラムは OpenCL1.1以上をサポートするビデオカード上の計算を実行するために使用されます。現代のビデオカードには、着信されたデータストリームを同時に単純な数学的操作を実行することが出来る小さな専用プロセッサの数百人が含まれています。OpenCLの言語は、並列コンピューティングを整理し、特定クラスのタスクためのより大きな速度を提供します。

グラフィックカードによっては double 型の数の操作はデフォルトでは無効にされています。これはコンパイルエラー5105 につながります。double 型数値のサポートを有効にするには #pragma OPENCL EXTENSION cl_khr_fp64 : enable ディレクティブをお使いのプログラムに追加してください。グラフィックカードが double をサポートしない場合は、ディレクティブの使用はサポートを有効にしません。

OpenCL用のソースコードは別のCLファイルに書き込むことをお勧めします。これらのファイルは後にリソース変数を使用してMQL5プログラムに組み込むことができます。

OpenCL プログラムでのエラー処理

OpenCLプログラムで直近のエラーに関する情報を取得するには、エラーコードとテキスト記述を取得できるようにするCLGetInfoIntegerおよびCLGetInfoString関数を使用します。

OpenCLの直近のエラーコード: 直近のOpenCLエラーを取得するには、CLGetInfoIntegerを呼び出します。*handle*パラメータは無視されます(ゼロに設定できます)。エラーの説明: https://registry.khronos.org/OpenCL/specs/3.0-unified/html/OpenCL_API.html#CL_SUCCESS。

不明なエラーコードの場合、「unknown OpenCL error N」という文字列が返されます。N はエラーコードです。例:

```
//--- 直近のエラーコードを取得するとき、最初のhandleパラメータは無視される
intcode= (int)CLGetInfoInteger(0,CL_LAST_ERROR);
```

OpenCLエラーの文字での説明: 直近のOpenCLエラーを取得するには、CLGetInfoStringを呼び出します。エラーコードは、*handle*パラメータとして渡す必要があります。

エラーの説明: https://registry.khronos.org/OpenCL/specs/3.0-unified/html/OpenCL_API.html#CL_SUCCESS. エラーコードの代わりにCL_LAST_ERRORが渡された場合、関数は最後のエラーの説明を返します。例:

```
//--- 直近のOpenCLエラーのコードを取得する
int    code= (int)CLGetInfoInteger(0,CL_LAST_ERROR);
stringdesc;// エラーのテキスト記述を取得

//--- エラーコードを使用してエラーのテキスト記述を取得
if(!CLGetInfoString(code,CL_ERROR_DESCRIPTION,desc))
 desc="cannot get OpenCL error description,"+ (string)GetLastError();
Print(desc);

//--- 最初にコードを取得せずに最後のOpenCLエラー記述を取得するには、CL_LAST_ERROR を渡す
if(!CLGetInfoString(CL_LAST_ERROR,CL_ERROR_DESCRIPTION,desc))
 desc="cannot get OpenCL error description,"+ (string)GetLastError();
Print(desc);;
```

© 2000-2025, MetaQuotes Ltd.

これまでのところ、エラー記述として内部列挙の名前が示されています。そのデコードは https://registry.khronos.org/OpenCL/specs/3.0-unified/html/OpenCL_API.html#CL_SUCCESSで確認できます。たとえば、CL_INVALID_KERNEL_ARGS　値は、「一部のカーネル引数が設定されていないか無効な場合に、カーネルをキューに入れるときに返される」ことを意味します。

OpenCL でプログラムを実行するための関数

関数	アクション
CLHandleType	OpenCL ハンドルの型をENUM_OPENCL_HANDLE_TYPE 列挙の値として返します。
CLGetInfoInteger	OpenCL オブジェクトまたはデバイスから整数プロパティ値を返します。
CLContextCreate	OpenCL コンテキストを作成します。
CLContextFree	OpenCL コンテキストを削除します。
CLGetDeviceInfo	OpenCL ドライバからデバイスプロパティを受け取ります。
CLProgramCreate	OpenCLプログラムをソースコードから作成します。
CLProgramFree	OpenCLプログラムを削除します。
CLKernelCreate	OpenCL start 関数を作成します。
CLKernelFree	OpenCL start 関数を削除します。
CLSetKernelArg	OpenCL 関数のパラメータを設定します。
CLSetKernelArgMem	OpenCL バッファを OpenCL 関数のパラメータをして設定します。
CLSetKernelArgMemLocal	ローカルバッファをカーネル関数の引数として設定します。
CLBufferCreate	OpenCL バッファを作成します。
CLBufferFree	OpenCL バッファを削除します。
CLBufferWrite	配列を OpenCL バッファに書き込みます。
CLBufferRead	OpenCL バッファを配列に読み込みます。
CLExecute	OpenCLプログラムを実行します。
CLExecutionStatus	OpenCLプログラム実行ステータスを返します。

参照

OpenCL, リソース

© 2000–2025, MetaQuotes Ltd.

CLHandleType

OpenCL ハンドルの型をENUM_OPENCL_HANDLE_TYPE 列挙の値として返します。

```
ENUM_OPENCL_HANDLE_TYPE  CLHandleType(
   int  handle     // OpenCL オブジェクトのハンドル
   );
```

パラメータ

handle

[in] OpenCL オブジェクトのハンドル、すなわちコンテキスト、カーネルまたは OpenCL プログラム

戻り値

ENUM_OPENCL_HANDLE_TYPE 列挙での OpenCL ハンドル値

ENUM_OPENCL_HANDLE_TYPE

識別子	説明
OPENCL_INVALID	不正なハンドル。
OPENCL_CONTEXT	OpenCL コンテキストのハンドル
OPENCL_PROGRAM	OpenCL プログラムのハンドル
OPENCL_KERNEL	OpenCL カーネルのハンドル
OPENCL_BUFFER	OpenCL バッファのハンドル

© 2000–2025, MetaQuotes Ltd.

CLGetInfoInteger

OpenCL オブジェクトまたはデバイスの整数プロパティ値を返します。

```
long  CLGetInfoInteger(
   int  handle,                              // OpenCL オブジェクトのハンドルまたはOpenCL デバイスの
番号
   ENUM_OPENCL_PROPERTY_INTEGER  prop        // リクエストされたプロパティ
   );
```

パラメータ

handle

[in] OpenCL オブジェクトのハンドルまたはOpenCL デバイスの番号。OpenCL デバイス番号はゼロで始まります。

prop

[in] 値の取得に使用されるENUM_OPENCL_PROPERTY_INTEGER 列挙からのリクエストされたプロパティの型

戻り値

成功の場合はプロパティ値、エラーの場合は -1。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

ENUM_OPENCL_PROPERTY_INTEGER

識別子	説明	Type
CL_DEVICE_COUNT	OpenCL をサポートするデバイスの数 このプロパティは初めのパラメータの指定を必要としません。つまり *handle* パラメータにはゼロ値を渡すことが出来ます。	int
CL_DEVICE_TYPE	デバイスの種類	ENUM_CL_DEVICE_TYPE
CL_DEVICE_VENDOR_ID	一意のベンダー識別子	uint
CL_DEVICE_MAX_COMPUTE_UNITS	OpenCL のデバイスで並列計算されたタスクの数 1 つのワーキンググループは、1つの計算作業を解決します。最少値は 1 です。	uint
CL_DEVICE_MAX_CLOCK_FREQUENCY	MHz単位でデバイスの最高周波数設定	uint
CL_DEVICE_GLOBAL_MEM_SIZE	バイト単位でデバイスのグローバルメモリのサイズ	ulong
CL_DEVICE_LOCAL_MEM_SIZE	バイト単位での処理されたデータ(シーン)ローカルメモリのサイズ	uint

© 2000-2025, MetaQuotes Ltd.

識別子	説明	Type
CL_BUFFER_SIZE	バイト単位でのOpenCLバッファの実際のサイズ	ulong
CL_DEVICE_MAX_WORK_GROUP_SIZE	OpenCLデバイスで使用可能なローカルワーキンググループの総数	ulong
CL_KERNEL_WORK_GROUP_SIZE	OpenCLプログラムで使用可能なローカルワーキンググループの総数	ulong
CL_KERNEL_LOCAL_MEM_SIZE	グループ内のすべての並列タスクを解決するためにOpenCLプログラムが使用するローカルメモリのサイズ(バイト単位)。使用可能な最大値を受け取るにはCL_DEVICE_LOCAL_MEM_SIZEを使用します。	ulong
CL_KERNEL_PRIVATE_MEM_SIZE	OpenCLプログラムカーネルの各タスクで使用されるプライベートメモリの最小サイズ(バイト単位)	ulong
CL_LAST_ERROR	直近のOpenCLエラーの値	int

ENUM_CL_DEVICE_TYPE 列挙は OpenCL をサポートするデバイスの可能な種類を含みます。CLGetInfoInteger(handle_or_deviceN, CL_DEVICE_TYPE) を呼び出すことによって、デバイスの種類の番号や OpenCL オブジェクトのハンドルを受け取ることが出来ます。

ENUM_CL_DEVICE_TYPE

識別子	説明
CL_DEVICE_ACCELERATOR	OpenCL 専用のアクセラレータ(例えば、IBM CELL Blade)。これらのデバイスは PCIe のような周辺機器を使用してホストプロセッサと通信します。
CL_DEVICE_CPU	ホストプロセッサであるOpenCL デバイス。ホストプロセッサは、OpenCLの実装を実行する単一またはマルチコアCPUです。
CL_DEVICE_GPU	GPU であるOpenCL デバイス。
CL_DEVICE_DEFAULT	システムのデフォルト OpenCL デバイス。CL_DEVICE_TYPE_CUSTOM デバイスはデフォルトにはなりません。
CL_DEVICE_CUSTOM	OpenCLのCで書かれたプログラムをサポートしない専用アクセラレータ。

例:

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
   int cl_ctx;
//--- OpenCL コンテキストを初期化する
   if((cl_ctx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
     {
      Print("OpenCL not found");
      return;
     }
//--- OpenCL デバイスの一般的な情報を表示する
   Print("OpenCL type: ",EnumToString((ENUM_CL_DEVICE_TYPE)
CLGetInfoInteger(cl_ctx,CL_DEVICE_TYPE)));
   Print("OpenCL vendor ID: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_VENDOR_ID));
   Print("OpenCL units: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_COMPUTE_UNITS));
   Print("OpenCL freq: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_CLOCK_FREQUENCY),"
MHz");
   Print("OpenCL global mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_GLOBAL_MEM_SIZE),"
bytes");
   Print("OpenCL local mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_LOCAL_MEM_SIZE),"
bytes");
//--- free OpenCL context
   CLContextFree(cl_ctx);
  }
```

© 2000–2025, MetaQuotes Ltd.

CLGetInfoString

OpenCL オブジェクトまたはデバイスのプロパティの string 値を返します。

```
bool  CLGetInfoString(
   int  handle,                              // OpenCL オブジェクトハンドルまたはOpenCL デバイス番号
   ENUM_OPENCL_PROPERTY_STRING  prop,        // リクエストされたプロパティ
   string&  value                            // 参照された文字列
   );
```

パラメータ

handle

[in] OpenCL オブジェクトハンドルまたはOpenCL デバイス番号。OpenCL デバイス番号付けはゼロで始まります。

prop

[in] 値の取得に使用される ENUM_OPENCL_PROPERTY_STRING 列挙からのリクエストされたプロパティ

value

[out] プロパティ値を受け取る文字列

戻り値

成功の場合は true、それ以外の場合は false。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

ENUM_OPENCL_PROPERTY_STRING

識別子	説明
CL_PLATFORM_PROFILE	CL_PLATFORM_PROFILE – OpenCL プロファイル。プロファイル名は次のうちの 1 つです。 • FULL_PROFILE – 実装は、OpenCLをサポートします。(機能は OpenCL のサポートのための追加の拡張機能を必要とせずに、カーネル仕様の一部として定義されています。) • EMBEDDED_PROFILE – 実装は、サブリメントとしてOpenCL をサポートします。改正されたプロファイルは、各 OpenCL のバージョンのサブセットとして定義されています。
CL_PLATFORM_VERSION	OpenCL バージョン
CL_PLATFORM_VENDOR	プラットフォームベンダー名
CL_PLATFORM_EXTENSIONS	プラットフォームでサポートされている拡張機能のリスト。拡張名は、このプラットフォームに関連する全てのデバイスによってサポートされる必要があります。
CL_DEVICE_NAME	デバイス名
CL_DEVICE_VENDOR	ベンダー名

© 2000–2025, MetaQuotes Ltd.

識別子	説明
CL_DRIVER_VERSION	major_number.minor_number 形式での OpenCL ドライババージョン
CL_DEVICE_PROFILE	OpenCL デバイスプロファイル。プロファイル名は次のうちの 1 つです。 • FULL_PROFILE – 実装は、OpenCLをサポートします。(機能は OpenCL のサポートのための追加の拡張機能を必要とせずに、カーネル仕様の一部として定義されています。) • EMBEDDED_PROFILE – 実装は、サブリメントとしてOpenCL をサポートします。改正されたプロファイルは、各 OpenCL のバージョンのサブセットとして定義されています。
CL_DEVICE_VERSION	「OpenCL<space><major_version.minor_version><space><vendor-specific information>」 形式での OpenCL バージョン
CL_DEVICE_EXTENSIONS	デバイスがサポートしている拡張機能のリストリストは、ベンダーがサポートしている拡張機能ならびに1つ以上の承認された名称を含みます。 cl_khr_int64_base_atomics cl_khr_int64_extended_atomics cl_khr_fp16 cl_khr_gl_sharing cl_khr_gl_event cl_khr_d3d10_sharing cl_khr_dx9_media_sharing cl_khr_d3d11_sharing
CL_DEVICE_BUILT_IN_KERNELS	サポートされたカーネルは「;」で区切られます。
CL_DEVICE_OPENCL_C_VERSION	このデバイス用のコンパイラでサポートされている最大バージョン。バージョン形式 : 「OpenCL<space>C<space><major_version.minor_version><space><vendor-specific information> 」
CL_ERROR_DESCRIPTION	OpenCLエラーのテキスト記述

例 :

```
void OnStart()
  {
   int cl_ctx;
   string str;
//--- OpenCL コンテキストを初期化する
   if((cl_ctx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
     {
      Print("OpenCL not found");
      return;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- プラットフォームの情報を表示する
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_NAME,str))
      Print("OpenCL platform name: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_VENDOR,str))
      Print("OpenCL platform vendor: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_VERSION,str))
      Print("OpenCL platform ver: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_PROFILE,str))
      Print("OpenCL platform profile: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_EXTENSIONS,str))
      Print("OpenCL platform ext: ",str);
//--- デバイスの情報を表示する
   if(CLGetInfoString(cl_ctx,CL_DEVICE_NAME,str))
      Print("OpenCL device name: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_PROFILE,str))
      Print("OpenCL device profile: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_BUILT_IN_KERNELS,str))
      Print("OpenCL device kernels: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_EXTENSIONS,str))
      Print("OpenCL device ext: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_VENDOR,str))
      Print("OpenCL device vendor: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_VERSION,str))
      Print("OpenCL device ver: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_OPENCL_C_VERSION,str))
      Print("OpenCL open c ver: ",str);
//--- OpenCL デバイスの一般的な情報を表示する
   Print("OpenCL type: ",EnumToString((ENUM_CL_DEVICE_TYPE)
CLGetInfoInteger(cl_ctx,CL_DEVICE_TYPE)));
   Print("OpenCL vendor ID: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_VENDOR_ID));
   Print("OpenCL units: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_COMPUTE_UNITS));
   Print("OpenCL freq: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_CLOCK_FREQUENCY));
   Print("OpenCL global mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_GLOBAL_MEM_SIZE));
   Print("OpenCL local mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_LOCAL_MEM_SIZE));
//--- free OpenCL context
   CLContextFree(cl_ctx);
  }
```

© 2000-2025, MetaQuotes Ltd.

CLContextCreate

OpenCL コンテキストを作成してハンドルを返します。

```
int  CLContextCreate(
   int  device=CL_USE_ANY      // OpenCL デバイスのシリアル番号またはマクロ
   );
```

パラメータ

device

[in] システム内の OpenCL デバイスの序数特定な数の代わりに、次のいずれかの値の指定が可能です。

- CL_USE_ANY – OpenCL をサポートする使用可能なデバイス
- CL_USE_CPU_ONLY – CPU での OpenCLのエミュレーションのみが許可されています
- CL_USE_GPU_ONLY – OpenCL エミュレーションが禁止され、OpenCL をサポートする特定のデバイス（ビデオカード）のみが使用可能です
- CL_USE_GPU_DOUBLE_ONLY - double型をサポートするGPUのみが許可されます

戻り値

成功の場合は OpenCL コンテキスト、その他の場合は -1。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

© 2000–2025, MetaQuotes Ltd.

CLContextFree

OpenCL コンテキストを削除します。

```
void  CLContextFree(
   int  context     // OpenCL コンテンツのハンドル
   );
```

パラメータ

context

[in] OpenCL コンテンツのハンドル

戻り値

なし。内部エラーの場合 _LastError の値が変更されます。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

© 2000-2025, MetaQuotes Ltd.

CLGetDeviceInfo

この関数は、OpenCL ドライバからデバイスプロパティを受け取ります。

```
bool  CLGetDeviceInfo(
   int      handle,          // OpenCL デバイスハンドル
   int      property_id,     // リクエストされたプロパティ識別子
   uchar&   data[],          // データを受け取る配列
   uint&    size             // 配列要素のシフト（デフォルトでは0
   );
```

パラメータ

handle

[in] OpenCL デバイスインデックス、または CLContextCreate() 関数で作成されたOpenCL ハンドル。

property_id

[in] 受け取られる OpenCL デバイスプロパティ識別子。値は下記の表に記載されている所定のもの 1 つであることが出来ます。

data[]

[out] リクエストされたプロパティのデータを受け取る配列

size

[out] *data[]* 配列での受け取ったデータのサイズ

戻り値

成功の場合は true、それ以外の場合は false。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

注意事項

OpenCL バッファへの読み込みのための始まる要素の番号は AS_SERIESフラグを考慮して計算されます。

OpenCL デバイスプロパティに使用可能な識別子のリスト

プロパティの繊細は OpenCL の公式ウェブサイトにて。

識別子	値
CL_DEVICE_TYPE	0x1000
CL_DEVICE_VENDOR_ID	0x1001
CL_DEVICE_MAX_COMPUTE_UNITS	0x1002
CL_DEVICE_MAX_WORK_ITEM_DIMENSIONS	0x1003
CL_DEVICE_MAX_WORK_GROUP_SIZE	0x1004
CL_DEVICE_MAX_WORK_ITEM_SIZES	0x1005
CL_DEVICE_PREFERRED_VECTOR_WIDTH_CHAR	0x1006

© 2000–2025, MetaQuotes Ltd.

識別子	値
CL_DEVICE_PREFERRED_VECTOR_WIDTH_SHORT	0x1007
CL_DEVICE_PREFERRED_VECTOR_WIDTH_INT	0x1008
CL_DEVICE_PREFERRED_VECTOR_WIDTH_LONG	0x1009
CL_DEVICE_PREFERRED_VECTOR_WIDTH_FLOAT	0x100A
CL_DEVICE_PREFERRED_VECTOR_WIDTH_DOUBLE	0x100B
CL_DEVICE_MAX_CLOCK_FREQUENCY	0x100C
CL_DEVICE_ADDRESS_BITS	0x100D
CL_DEVICE_MAX_READ_IMAGE_ARGS	0x100E
CL_DEVICE_MAX_WRITE_IMAGE_ARGS	0x100F
CL_DEVICE_MAX_MEM_ALLOC_SIZE	0x1010
CL_DEVICE_IMAGE2D_MAX_WIDTH	0x1011
CL_DEVICE_IMAGE2D_MAX_HEIGHT	0x1012
CL_DEVICE_IMAGE3D_MAX_WIDTH	0x1013
CL_DEVICE_IMAGE3D_MAX_HEIGHT	0x1014
CL_DEVICE_IMAGE3D_MAX_DEPTH	0x1015
CL_DEVICE_IMAGE_SUPPORT	0x1016
CL_DEVICE_MAX_PARAMETER_SIZE	0x1017
CL_DEVICE_MAX_SAMPLERS	0x1018
CL_DEVICE_MEM_BASE_ADDR_ALIGN	0x1019
CL_DEVICE_MIN_DATA_TYPE_ALIGN_SIZE	0x101A
CL_DEVICE_SINGLE_FP_CONFIG	0x101B
CL_DEVICE_GLOBAL_MEM_CACHE_TYPE	0x101C
CL_DEVICE_GLOBAL_MEM_CACHELINE_SIZE	0x101D
CL_DEVICE_GLOBAL_MEM_CACHE_SIZE	0x101E
CL_DEVICE_GLOBAL_MEM_SIZE	0x101F
CL_DEVICE_MAX_CONSTANT_BUFFER_SIZE	0x1020
CL_DEVICE_MAX_CONSTANT_ARGS	0x1021
CL_DEVICE_LOCAL_MEM_TYPE	0x1022

© 2000–2025, MetaQuotes Ltd.

識別子	値
CL_DEVICE_LOCAL_MEM_SIZE	0x1023
CL_DEVICE_ERROR_CORRECTION_SUPPORT	0x1024
CL_DEVICE_PROFILING_TIMER_RESOLUTION	0x1025
CL_DEVICE_ENDIAN_LITTLE	0x1026
CL_DEVICE_AVAILABLE	0x1027
CL_DEVICE_COMPILER_AVAILABLE	0x1028
CL_DEVICE_EXECUTION_CAPABILITIES	0x1029
CL_DEVICE_QUEUE_PROPERTIES	0x102A
CL_DEVICE_NAME	0x102B
CL_DEVICE_VENDOR	0x102C
CL_DRIVER_VERSION	0x102D
CL_DEVICE_PROFILE	0x102E
CL_DEVICE_VERSION	0x102F
CL_DEVICE_EXTENSIONS	0x1030
CL_DEVICE_PLATFORM	0x1031
CL_DEVICE_DOUBLE_FP_CONFIG	0x1032
CL_DEVICE_PREFERRED_VECTOR_WIDTH_HALF	0x1034
CL_DEVICE_HOST_UNIFIED_MEMORY	0x1035
CL_DEVICE_NATIVE_VECTOR_WIDTH_CHAR	0x1036
CL_DEVICE_NATIVE_VECTOR_WIDTH_SHORT	0x1037
CL_DEVICE_NATIVE_VECTOR_WIDTH_INT	0x1038
CL_DEVICE_NATIVE_VECTOR_WIDTH_LONG	0x1039
CL_DEVICE_NATIVE_VECTOR_WIDTH_FLOAT	0x103A
CL_DEVICE_NATIVE_VECTOR_WIDTH_DOUBLE	0x103B
CL_DEVICE_NATIVE_VECTOR_WIDTH_HALF	0x103C
CL_DEVICE_OPENCL_C_VERSION	0x103D
CL_DEVICE_LINKER_AVAILABLE	0x103E
CL_DEVICE_BUILT_IN_KERNELS	0x103F
CL_DEVICE_IMAGE_MAX_BUFFER_SIZE	0x1040
CL_DEVICE_IMAGE_MAX_ARRAY_SIZE	0x1041

© 2000–2025, MetaQuotes Ltd.

識別子	値
CL_DEVICE_PARENT_DEVICE	0x1042
CL_DEVICE_PARTITION_MAX_SUB_DEVICES	0x1043
CL_DEVICE_PARTITION_PROPERTIES	0x1044
CL_DEVICE_PARTITION_AFFINITY_DOMAIN	0x1045
CL_DEVICE_PARTITION_TYPE	0x1046
CL_DEVICE_REFERENCE_COUNT	0x1047
CL_DEVICE_PREFERRED_INTEROP_USER_SYNC	0x1048
CL_DEVICE_PRINTF_BUFFER_SIZE	0x1049
CL_DEVICE_IMAGE_PITCH_ALIGNMENT	0x104A
CL_DEVICE_IMAGE_BASE_ADDRESS_ALIGNMENT	0x104B

例：

```
void OnStart()
  {
//---
   int dCount= CLGetInfoInteger(0,CL_DEVICE_COUNT);
   for(int i = 0; i<dCount; i++)
     {
      int clCtx=CLContextCreate(i);
      if(clCtx == -1)
         Print("ERROR in CLContextCreate");
      string device;
      CLGetInfoString(clCtx,CL_DEVICE_NAME,device);
      Print(i,": ",device);
      uchar data[1024];
      uint size;
      CLGetDeviceInfo(clCtx,CL_DEVICE_VENDOR,data,size);
      Print("size = ",size);
      string str=CharArrayToString(data);
      Print(str);
     }
  }
//--- エキスパート操作ログエントリーの例
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      2: Advanced Micro Devices, Inc.
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      size = 32
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      Tahiti
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      Intel(R) Corporation
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      size = 21
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      1:        Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz
```

© 2000-2025, MetaQuotes Ltd.

```
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      NVIDIA Corporation
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      size = 19
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      0: GeForce GTX 580
```

© 2000–2025, MetaQuotes Ltd.

CLProgramCreate

OpenCL プログラムをソースコードから作成します。

```
int  CLProgramCreate(
   int           context,     // OpenCL コンテキストのハンドル
   const string  source       // ソースコード
   );
```

オーバーロード関数は OpenCL プログラムを作成し、渡された文字列にコンパイラメッセージを書き込みます。

```
int  CLProgramCreate(
   int           context,     // OpenCL コンテキストのハンドル
   const string  source,      // ソースコード
   string       &build_log    // コンパイルログを受け取る文字列
   );
```

パラメータ

context

[in] OpenCL コンテンツのハンドル

source

[in] OpenCL プログラムのソースコードを含む文字列

&build_log

[in] OpenCL コンパイラメッセージを受け取る文字列

戻り値

成功の場合 OpenCL オブジェクトハンドル。エラーの場合は -1。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

注意事項

現時点では、次のエラーコードが使用されます。

- ERR_OPENCL_INVALID_HANDLE - 不正な OpenCL コンテキストハンドル
- ERR_INVALID_PARAMETER - 不正な文字列パラメータ
- ERR_NOT_ENOUGH_MEMORY - 操作完了に不充分なメモリ
- ERR_OPENCL_PROGRAM_CREATE - OpenCL 内部エラーまたはコンパイルエラー

グラフィックカードによっては double 型の数の操作はデフォルトでは無効にされています。これはコンパイルエラー5105 につながります。double 型数値のサポートを有効にするには #pragma OPENCL EXTENSION cl_khr_fp64 : enable ディレクティブをお使いのプログラムに追加してください。グラフィックカードが double をサポートしない場合は、ディレクティブの使用はサポートを有効にしません。

例:

```
//+------------------------------------------------------------------+
//| OpenCL カーネル                                                  |
//+------------------------------------------------------------------+
const string
```

© 2000-2025, MetaQuotes Ltd.

```
cl_src=
       //--- デフォルトでは double をサポートしない GPU がある
       //--- cl_khr_fp64 directive が double の操作を有効にするために使用される
       "#pragma OPENCL EXTENSION cl_khr_fp64 : enable      \r\n"
       //--- OpenCL カーネル関数
       "__kernel void Test_GPU(__global double *data,      \r\n"
       "                       const    int N,             \r\n"
       "                       const    int total_arrays)  \r\n"
       "  {                                                 \r\n"
       "   uint kernel_index=get_global_id(0);              \r\n"
       "   if (kernel_index>total_arrays) return;           \r\n"
       "   uint local_start_offset=kernel_index*N;          \r\n"
       "   for(int i=0; i<N; i++)                           \r\n"
       "     {                                              \r\n"
       "       data[i+local_start_offset] *= 2.0;           \r\n"
       "     }                                              \r\n"
       "  }                                                 \r\n";
//+------------------------------------------------------------------+
//| Test_CPU                                                         |
//+------------------------------------------------------------------+
bool Test_CPU(double &data[],const int N,const int id,const int total_arrays)
  {
//--- 配列サイズをチェックする
   if(ArraySize(data)==0) return(false);
//--- 配列インデックスをチェックする
   if(id>total_arrays) return(false);
//--- インデックス id の配列のローカルオフセットを計算する
   int local_start_offset=id*N;
//--- 要素を二倍にする
   for(int i=0; i<N; i++)
     {
      data[i+local_start_offset]*=2.0;
     }
   return true;
  }
//---
#define ARRAY_SIZE   100  // 配列サイズ
#define TOTAL_ARRAYS 5    // 全ての配列
//--- OpenCL ハンドル
int cl_ctx;  // OpenCL コンテキストハンドル
int cl_prg;  // OpenCL プログラムハンドル
int cl_krn;  // OpenCL カーネルハンドル
int cl_mem;  // OpenCL バッファハンドル
//---
double DataArray1[]; // CPU 計算のためのデータ配列
double DataArray2[]; // GPU 計算のためのデータ配列
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
int OnStart()
  {
//--- OpenCL オブジェクトを初期化する
//--- OpenCL コンテキストを作成する
   if((cl_ctx=CLContextCreate())==INVALID_HANDLE)
     {
      Print("OpenCL not found. Error=",GetLastError());
      return(1);
     }
//--- OpenCL プログラムを作成する
   if((cl_prg=CLProgramCreate(cl_ctx,cl_src))==INVALID_HANDLE)
     {
      CLContextFree(cl_ctx);
      Print("OpenCL program create failed. Error=",GetLastError());
      return(1);
     }
//--- OpenCL カーネルを作成する
   if((cl_krn=CLKernelCreate(cl_prg,"Test_GPU"))==INVALID_HANDLE)
     {
      CLProgramFree(cl_prg);
      CLContextFree(cl_ctx);
      Print("OpenCL kernel create failed. Error=",GetLastError());
      return(1);
     }
//--- OpenCL バッファを作成する
   if((cl_mem=CLBufferCreate(cl_ctx,ARRAY_SIZE*TOTAL_ARRAYS*sizeof
(double),CL_MEM_READ_WRITE))==INVALID_HANDLE)
     {
      CLKernelFree(cl_krn);
      CLProgramFree(cl_prg);
      CLContextFree(cl_ctx);
      Print("OpenCL buffer create failed. Error=",GetLastError());
      return(1);
     }
//--- OpenCL カーネルの定数パラメータを設定する
   CLSetKernelArgMem(cl_krn,0,cl_mem);
   CLSetKernelArg(cl_krn,1,ARRAY_SIZE);
   CLSetKernelArg(cl_krn,2,TOTAL_ARRAYS);
//--- データ配列を準備する
   ArrayResize(DataArray1,ARRAY_SIZE*TOTAL_ARRAYS);
   ArrayResize(DataArray2,ARRAY_SIZE*TOTAL_ARRAYS);
//--- データに配列を書き込む
   for(int j=0; j<TOTAL_ARRAYS; j++)
     {
      //--- j 番目の配列のローカル開始オフセットを計算する
      uint local_offset=j*ARRAY_SIZE;
       //--- インデックス j の配列を準備する
      for(int i=0; i<ARRAY_SIZE; i++)
        {
```

© 2000-2025, MetaQuotes Ltd.

```
         //--- MathCos(i+j) の結果を配列に書き込む
         DataArray1[i+local_offset]=MathCos(i+j);
         DataArray2[i+local_offset]=MathCos(i+j);
        }
     };
//--- CPU 計算をテストする
   for(int j=0; j<TOTAL_ARRAYS; j++)
     {
      //--- インデックス j の配列の計算
      Test_CPU(DataArray1,ARRAY_SIZE,j,TOTAL_ARRAYS);
     }
//--- CLExecute パラメータを準備する
   uint  offset[]={0};
//--- グローバルワークサイズ
   uint  work[]={TOTAL_ARRAYS};
//--- OpenCL バッファにデータを書く
   CLBufferWrite(cl_mem,DataArray2);
//--- OpenCL カーネルを実行する
   CLExecute(cl_krn,1,offset,work);
//--- OpenCL バッファからデータを読む
   CLBufferRead(cl_mem,DataArray2);
//--- エラーの合計
   double total_error=0;
//--- 結果を比較してエラーを計算する
   for(int j=0; j<TOTAL_ARRAYS; j++)
     {
      //--- j 番目の配列のローカルオフセットを計算する
      uint local_offset=j*ARRAY_SIZE;
      //--- 結果を比較する
      for(int i=0; i<ARRAY_SIZE; i++)
        {
         double v1=DataArray1[i+local_offset];
         double v2=DataArray2[i+local_offset];
         double delta=MathAbs(v2-v1);
         total_error+=delta;
         //--- 最初と最後の配列を表示する
         if((j==0) || (j==TOTAL_ARRAYS-1))
            PrintFormat("array %d of %d, element [%d]:  %f, %f, [error]=%
f",j+1,TOTAL_ARRAYS,i,v1,v2,delta);
        }
     }
   PrintFormat("Total error: %f",total_error);
//--- OpenCL オブジェクトを削除する
//--- OpenCL バッファを解放する
   CLBufferFree(cl_mem);
//--- OpenCL カーネルを解放する
   CLKernelFree(cl_krn);
//--- OpenCL プログラムを解放する
   CLProgramFree(cl_prg);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- OpenCL コンテキストを解放する
   CLContextFree(cl_ctx);
//---
   return(0);
  }
```

© 2000–2025, MetaQuotes Ltd.

。CLProgramFree

OpenCL プログラムを削除します。

```
void  CLProgramFree(
   int  program     // OpenCL オブジェクトのハンドル
   );
```

パラメータ

program

[in] OpenCL オブジェクトのハンドル

戻り値

なし。内部エラーの場合 _LastError の値が変更されます。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

© 2000–2025, MetaQuotes Ltd.

CLKernelCreate

OpenCL プログラムカーネルを作成してハンドルを返します。

```
int  CLKernelCreate(
   int           program,        // OpenCL オブジェクトのハンドル
   const string  kernel_name     // カーネル名
   );
```

パラメータ

program

[in] OpenCL プログラムオブジェクトのハンドル

kernel_name

[in] 実行が開始される OpenCL プログラムのカーネル関数の名称

戻り値

成功の場合 OpenCL オブジェクトハンドル。エラーの場合は -1。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

注意事項

現時点では、次のエラーコードが使用されます。

- ERR_OPENCL_INVALID_HANDLE - invalid handle to OpenCL program.
- ERR_INVALID_PARAMETER – 不正な文字列パラメータ
- ERR_OPENCL_TOO_LONG_KERNEL_NAME – カーネル名が 127 文字を超えます。
- ERR_OPENCL_KERNEL_CREATE – OpenCL オブジェクト作成時に内部エラーが発生しました。

© 2000–2025, MetaQuotes Ltd.

CLKernelFree

OpenCL start 関数を削除します。

```
void  CLKernelFree(
   int  kernel     // OpenCL プログラムカーネルのハンドル
   );
```

パラメータ

kernel_name

[in] カーネルオブジェクトのハンドル

戻り値

なし。内部エラーの場合 _LastError の値が変更されます。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

© 2000–2025, MetaQuotes Ltd.

CLSetKernelArg

OpenCL 関数のパラメータを設定します。

```
bool  CLSetKernelArg(
   int   kernel,         // OpenCL プログラムカーネルのハンドル
   uint  arg_index,      // OpenCL 関数の引数の番号
   void  arg_value       // ソースコード
   );
```

パラメータ

kernel

[in] OpenCL プログラムカーネルのハンドル

arg_index

[in] OpenCL 関数の引数の番号（番号付けはゼロから）

arg_value

[in] 関数の引数の値

戻り値

成功の場合は true、それ以外の場合は false。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

注意事項

現時点では、次のエラーコードが使用されます。

- ERR_INVALID_PARAMETER,
- ERR_OPENCL_INVALID_HANDLE - invalid handle to the OpenCL kernel.
- ERR_OPENCL_SET_KERNEL_PARAMETER - internal error of OpenCL.

© 2000–2025, MetaQuotes Ltd.

CLSetKernelArgMem

OpenCL バッファを OpenCL 関数のパラメータとして設定します。

```
bool  CLSetKernelArgMem(
   int   kernel,           // OpenCL プログラムカーネルのハンドル
   uint  arg_index,        // OpenCL 関数の引数の番号
   int   cl_mem_handle     // OpenCL バッファハンドル
   );
```

パラメータ

kernel

[in] OpenCL プログラムカーネルのハンドル

arg_index

[in] OpenCL 関数の引数の番号（番号付けはゼロから）

cl_mem_handle

[in] OpenCL バッファハンドル

戻り値

成功の場合は true、それ以外の場合は false。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

© 2000–2025, MetaQuotes Ltd.

CLSetKernelArgMemLocal

ローカルバッファをカーネル関数の引数として設定します。

```
bool  CLSetKernelArgMemLocal(
   int    kernel,           // OpenCLプログラムのカーネルへのハンドル
   uint   arg_index,        // OpenCL関数の引数の数
   ulong  local_mem_size    // バッファサイズ
   );
```

パラメータ

kernel

[in] OpenCL プログラムカーネルのハンドル

arg_index

[in] OpenCL 関数の引数の番号（番号付けはゼロから）

local_mem_size

[in] バイト単位でのバッファサイズ

戻り値

成功の場合は true、それ以外の場合は false。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

© 2000–2025, MetaQuotes Ltd.

CLBufferCreate

OpenCL バッファを作成してハンドルを返します。

```
int  CLBufferCreate(
   int   context,     // OpenCL コンテンツのハンドル
   uint  size,        // バッファサイズ
   uint  flags        // OpenCL バッファのプロパティを指定するフラグの組み合わせ
   );
```

パラメータ

context

[in] OpenCL コンテンツのハンドル

size

[in] バイト単位でのバッファサイズ

flags

[in] フラグの組み合わせで指定されるバッファプロパティは CL_MEM_READ_WRITE、CL_MEM_WRITE_ONLY、CL_MEM_READ_ONLY 及び CL_MEM_ALLOC_HOST_PTR です。

戻り値

成功の場合 OpenCL バッファハンドル。エラーの場合は -1。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

注意事項

現時点では、次のエラーコードが使用されます。

- ERR_OPENCL_INVALID_HANDLE - 不正な OpenCL コンテキストハンドル
- ERR_NOT_ENOUGH_MEMORY - 不充分なメモリ
- ERR_OPENCL_BUFFER_CREATE - バッファ作成時の内部エラー

© 2000-2025, MetaQuotes Ltd.

CLBufferFree

OpenCL バッファを削除します。

```
void  CLBufferFree(
   int   buffer     // OpenCL バッファハンドル
   );
```

パラメータ

buffer

[in] OpenCL バッファハンドル

戻り値

なし。内部エラーの場合 _LastError の値が変更されます。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

© 2000–2025, MetaQuotes Ltd.

CLBufferWrite

OpenCL バッファに書き込み、書かれた要素の数を返します。

```
uint  CLBufferWrite(
   int           buffer,                    // OpenCL バッファハンドル
   const void&   data[],                    // 値の配列
   uint          buffer_offset=0,           // OpenCL バッファ内のバイト単位のオフセット（デフォルトで
0)
   uint          data_offset=0,             // OpenCL バッファ内の要素数でのオフセット（デフォルトで
0)
   uint          data_count=WHOLE_ARRAY     // 書かれる配列の値の数（デフォルトでは配列全体）
   );
```

行列とベクトルを処理するためのバージョンもあります。

行列の値をバッファに書き込み、成功した場合は true を返します。

```
uint  CLBufferWrite(
   int           buffer,                    // OpenCL バッファハンドル
   uint          buffer_offset,             // OpenCL バッファ内のバイト単位のオフセット
   matrix<T>     &mat                       // バッファに書き込む値の行列
   );
```

ベクトルの値をバッファに書き込み、成功した場合は true を返します。

```
uint  CLBufferWrite(
   int           buffer,                    // OpenCL バッファハンドル
   uint          buffer_offset,             // OpenCL バッファ内のバイト単位のオフセット
   vector<T>     &vec                       // バッファに書き込む値のベクトル
   );
```

パラメータ

buffer

[in] OpenCL バッファハンドル

data[]

[in] OpenCL バッファに書かれる値の配列。参照で引き渡されます。

buffer_offset

[in] OpenCL バッファ内のバイト単位での読み込みが始まるオフセット。デフォルトでは書き込みはバッファの一番初めから始まります。

data_offset

[in] OpenCLのバッファに値を書き込むための最初の配列要素のインデックス。デフォルトでは配列の一番初めの値が使用されます。

data_count

[in] 書き込まれる値の数。デフォルトでは配列の値全部。

mat

[out] バッファにデータを書き込む行列は、matrix、matrixf、matrixc の 3 つのタイプのいずれかです。

© 2000–2025, MetaQuotes Ltd.

vec

[out] バッファにデータを書き込むベクトルは、vector、vectorf、vectorc の 3 つのタイプのいずれかです。

戻り値

書かれた要素の数。エラーの場合 0。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

行列またはベクトルが正常に処理された場合は true、それ以外の場合は false。

注意事項

1 次元配列では OpenCL バッファへの書き入れのための読み込みが始まる要素の番号は AS_SERIESフラグを考慮して計算されます。

2 次元以上の配列は1 次元として提示されます。この場合 *data_offset* は、1 次元目の要素の数ではなく、抜かされるべき要素の数です。

MatMul メソッドと OpenCL での並列計算を使用した行列乗算の例

```
#define M        3000       // 1 番目の行列の行数
#define K        2000       // 1 番目の行列の列数は 2 番目の行列の行数と同じ
#define N        3000       // 2 番目の行列の列数

//+------------------------------------------------------------------+
const string clSrc=
  "#define N     "+IntegerToString(N)+"                              \r\n"
  "#define K     "+IntegerToString(K)+"                              \r\n"
  "                                                                  \r\n"
  "__kernel void matricesMul( __global float *in1,                   \r\n"
  "                           __global float *in2,                   \r\n"
  "                           __global float *out  )                 \r\n"
  "{                                                                 \r\n"
  "  int m = get_global_id( 0 );                                     \r\n"
  "  int n = get_global_id( 1 );                                     \r\n"
  "  float sum = 0.0;                                                \r\n"
  "  for( int k = 0; k < K; k ++ )                                   \r\n"
  "     sum += in1[ m * K + k ] * in2[ k * N + n ];                  \r\n"
  "  out[ m * N + n ] = sum;                                         \r\n"
  "}                                                                 \r\n";
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- 擬似乱数生成器を初期化する
  MathSrand((int)TimeCurrent());
//--- 指定されたサイズの行列にランダムな値を書き込む
  matrixf mat1(M, K, MatrixRandom) ;     // 1 番目の行列
  matrixf mat2(K, N, MatrixRandom);      // 2 番目の行列
```

© 2000–2025, MetaQuotes Ltd.

```
//--- 単純な方法を使用して行列の積を計算する
  uint start=GetTickCount();
  matrixf matrix_naive=matrixf::Zeros(M, N);// 2 つの行列を乗算した結果がここに設定される
  for(int m=0; m<M; m++)
    for(int k=0; k<K; k++)
      for(int n=0; n<N; n++)
        matrix_naive[m][n]+=mat1[m][k]*mat2[k][n];
  uint time_naive=GetTickCount()-start;

//--- MatMul を使用して行列の積を計算する
  start=GetTickCount();
  matrixf matrix_matmul=mat1.MatMul(mat2);
  uint time_matmul=GetTickCount()-start;

//--- OpenCL で行列の積を計算する
  matrixf matrix_opencl=matrixf::Zeros(M, N);
  int cl_ctx;                // コンテキストハンドル
  if((cl_ctx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
   {
    Print("OpenCL not found, leaving");
    return;
   }
  int cl_prg;                // プログラムハンドル
  int cl_krn;                // カーネルハンドル
  int cl_mem_in1;            // 1 番目の(入力)バッファハンドル
  int cl_mem_in2;            // 2 番目の(入力)バッファハンドル
  int cl_mem_out;            // 3 番目の(出力)バッファハンドル
//--- プログラムとカーネルを作成する
  cl_prg = CLProgramCreate(cl_ctx, clSrc);
  cl_krn = CLKernelCreate(cl_prg, "matricesMul");
//--- 3 つの行列に対して 3 つのバッファすべてを作成する
  cl_mem_in1=CLBufferCreate(cl_ctx, M*K*sizeof(float), CL_MEM_READ_WRITE);
  cl_mem_in2=CLBufferCreate(cl_ctx, K*N*sizeof(float), CL_MEM_READ_WRITE);
//--- 3 番目の行列 - 出力
  cl_mem_out=CLBufferCreate(cl_ctx, M*N*sizeof(float), CL_MEM_READ_WRITE);
//--- カーネル引数を設定する
  CLSetKernelArgMem(cl_krn, 0, cl_mem_in1);
  CLSetKernelArgMem(cl_krn, 1, cl_mem_in2);
  CLSetKernelArgMem(cl_krn, 2, cl_mem_out);
//--- 行列をデバイスバッファに書く
  CLBufferWrite(cl_mem_in1, 0, mat1);
  CLBufferWrite(cl_mem_in2, 0, mat2);
  CLBufferWrite(cl_mem_out, 0, matrix_opencl);
//--- OpenCL コード実行時間の開始
  start=GetTickCount();
//--- タスクの作業領域のパラメータを設定し、OpenCL プログラムを実行する
  uint  offs[2] = {0, 0};
  uint works[2] = {M, N};
  start=GetTickCount();
```

© 2000-2025, MetaQuotes Ltd.

```
  bool ex=CLExecute(cl_krn, 2, offs, works);
//--- 結果を行列に計算する
  if(CLBufferRead(cl_mem_out, 0, matrix_opencl))
    PrintFormat("[%d x %d] matrix read: ", matrix_opencl.Rows(),
matrix_opencl.Cols());
   else
      Print("CLBufferRead(cl_mem_out, 0, matrix_opencl failed. Error
",GetLastError());
  uint time_opencl=GetTickCount()-start;
  Print("Compare calculation time using each method");
  PrintFormat("Naive product time = %d ms",time_naive);
  PrintFormat("MatMul product time = %d ms",time_matmul);
  PrintFormat("OpenCl product time = %d ms",time_opencl);
//--- すべての OpenCL コンテキストを解放する
  CLFreeAll(cl_ctx, cl_prg, cl_krn, cl_mem_in1, cl_mem_in2, cl_mem_out);

//--- 得られたすべての結果行列を相互に比較する
  Print("How many discrepancy errors are there between result matrices?");
  ulong errors=matrix_naive.Compare(matrix_matmul,(float)1e-12);
  Print("matrix_direct.Compare(matrix_matmul,1e-12)=",errors);
  errors=matrix_matmul.Compare(matrix_opencl,float(1e-12));
  Print("matrix_matmul.Compare(matrix_opencl,1e-12)=",errors);
/*
  結果:

   [3000 x 3000] matrix read:
  各方法での計算時間の比較
   Naive product time = 54750 ms
   MatMul product time = 4578 ms
   OpenCl product time = 922 ms
  結果行列間にあった不一致エラーの数
   matrix_direct.Compare(matrix_matmul,1e-12)=0
   matrix_matmul.Compare(matrix_opencl,1e-12)=0
*/
 }
//+------------------------------------------------------------------+
//|  行列にランダムな値を書き込む                                    |
//+------------------------------------------------------------------+
void MatrixRandom(matrixf& m)
 {
  for(ulong r=0; r<m.Rows(); r++)
   {
    for(ulong c=0; c<m.Cols(); c++)
     {
      m[r][c]=(float)((MathRand()-16383.5)/32767.);
     }
   }
 }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| すべての OpenCL コンテキストを解放する                                  |
//+------------------------------------------------------------------+
void CLFreeAll(int cl_ctx, int cl_prg, int cl_krn,
               int cl_mem_in1, int cl_mem_in2, int cl_mem_out)
 {
//--- OpenCL によって作成されたすべてのコンテキストを逆の順序で削除する
  CLBufferFree(cl_mem_in1);
  CLBufferFree(cl_mem_in2);
  CLBufferFree(cl_mem_out);
  CLKernelFree(cl_krn);
  CLProgramFree(cl_prg);
  CLContextFree(cl_ctx);
 }
```

© 2000–2025, MetaQuotes Ltd.

CLBufferRead

OpenCL バッファを配列に読み込んで読まれた要素の数を返します。

```
uint  CLBufferRead(
   int           buffer,                    // OpenCL バッファハンドル
   const void&   data[],                    // 値の配列
   uint          buffer_offset=0,           // OpenCL バッファ内のバイト単位のオフセット（デフォルトで
0)
   uint          data_offset=0,             // OpenCL バッファ内の要素数でのオフセット（デフォルトで
0)
   uint          data_count=WHOLE_ARRAY     // バッファから読まれる値の数（デフォルトではバッファ全体）
   );
```

行列とベクトルを処理するためのバージョンもあります。

OpenCLバッファの値を行列に読み込み、成功した場合は true を返します。

```
uint  CLBufferRead(
   int            buffer,                   // OpenCL バッファハンドル
   uint           buffer_offset,            // OpenCL バッファ内のバイト単位のオフセット
   const matrxi&  mat,                      // バッファから値を受け取る行列
   ulong          rows=-1,                  // 行列の行数
   ulong          cols=-1                   // 行列の列数
   );
```

OpenCLバッファの値をベクトルに読み込み、成功した場合は true を返します。

```
uint  CLBufferRead(
   int            buffer,                   // OpenCL バッファハンドル
   uint           buffer_offset,            // OpenCL バッファ内のバイト単位のオフセット
   const vector&  vec,                      // バッファから値を受け取るベクトル
   ulong          size-1,                   // ベクトルの長さ
   );
```

パラメータ

buffer

[in] OpenCL バッファハンドル

data[]

[in] OpenCL バッファから値を受け取る配列。参照で引き渡されます。

buffer_offset

[in] OpenCL バッファ内のバイト単位での読み込みが始まるオフセット。デフォルトでは読み込みはバッファの一番初めから始まります。

data_offset

[in] OpenCLのバッファの値を書き込むための最初の配列要素のインデックス。デフォルトでは、配列への読み込み値の書き込みはゼロインデックスから始まります。

data_count

© 2000–2025, MetaQuotes Ltd.

[in] 読まれる値の数。デフォルトでは OpenCL バファ全体が読まれます。

mat

[out] バッファにデータを書き込む行列は、matrix、matrixf、matrixc の 3 つのタイプのいずれかです。

vec

[out] バッファにデータを書き込むベクトルは、vector、vectorf、vectorc の 3 つのタイプのいずれかです。

rows=-1

[in] パラメータが指定されている場合、*cols*パラメータも指定する必要があります。新しい行列の次元が指定されていない場合は、現在の次元が使用されます。値が -1 の場合、行数は変わりません。

cols=-1

[in] パラメータが指定されていない場合、*rows*パラメータもスキップする必要があります。行列は次のルールに従います。両方のパラメータが指定されているか、何も指定されていない場合、エラーが発生します。両方のパラメータ(*rows* と *cols*)が指定されている場合、行列のサイズが変更されます。-1の場合、列数は変わりません。

size=-1

[in] パラメータが指定されていない場合、またはその値が -1 の場合、ベクトルの長さは変わりません。

戻り値

読まれた要素の数。エラーの場合 0。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

行列またはベクトルが正常に処理された場合は true、それ以外の場合は false。

注意事項

1 次元配列では OpenCL バッファへの書き入れが始まる要素の番号はAS_SERIES フラグを考慮して計算されます。

2 次元以上の配列は1 次元として提示されます。この場合 *data_offset* は、1 次元目の要素の数ではなく、抜かされるべき要素の数です。

式を使用した Pi の計算の例

$$\pi = \lim_{N\to\infty}\frac{1}{N}\sum_{k=0}^{N-1}\frac{4}{1+\left(\frac{2k+1}{2N}\right)^2} = 16\lim_{N\to\infty}N\sum_{k=0}^{N-1}\frac{1}{4N^2+(2k+1)^2}$$

```
#define  _num_steps        1000000000
#define  _divisor          40000
#define  _step             1.0 / _num_steps
#define  _intrnCnt         _num_steps / _divisor

//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
string D2S(double arg, int digits) { return DoubleToString(arg, digits); }
```

© 2000-2025, MetaQuotes Ltd.

```
string I2S(int arg)                { return IntegerToString(arg); }

//--- OpenCL プログラムコード
const string clSource=
  "#define _step "+D2S(_step, 12)+"                    \r\n"
  "#define _intrnCnt "+I2S(_intrnCnt)+"                \r\n"
  "                                                    \r\n"
  "__kernel void Pi( __global double *out )            \r\n"
  "{                                                   \r\n"
  "  int i = get_global_id( 0 );                       \r\n"
  "  double partsum = 0.0;                             \r\n"
  "  double x = 0.0;                                   \r\n"
  "  long from = i * _intrnCnt;                        \r\n"
  "  long to = from + _intrnCnt;                       \r\n"
  "  for( long j = from; j < to; j ++ )                \r\n"
  "  {                                                 \r\n"
  "     x = ( j + 0.5 ) * _step;                       \r\n"
  "     partsum += 4.0 / ( 1. + x * x );               \r\n"
  "  }                                                 \r\n"
  "  out[ i ] = partsum;                               \r\n"
  "}                                                   \r\n";

//+------------------------------------------------------------------+
//|  メモリを解放する補助ルーチン                                    |
//+------------------------------------------------------------------+
void CLFreeAll(const int clMem, const int clKrn, const int clPrg, const int clCtx)
 {
  CLBufferFree(clMem);
  CLKernelFree(clKrn);
  CLProgramFree(clPrg);
  CLContextFree(clCtx);
 }
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
int OnStart()
 {
  Print("Pi Calculation: step = "+D2S(_step, 12)+"; _intrnCnt = "+I2S(_intrnCnt));
//--- OpenCL コンテキストを準備する
  int clCtx;
  if((clCtx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
   {
    Print("OpenCL not found");
    return(-1);
   }
  int clPrg = CLProgramCreate(clCtx, clSource);
  int clKrn = CLKernelCreate(clPrg, "Pi");
  int clMem=CLBufferCreate(clCtx, _divisor*sizeof(double), CL_MEM_READ_WRITE);
  CLSetKernelArgMem(clKrn, 0, clMem);
```

© 2000–2025, MetaQuotes Ltd.

```
  const uint offs[1]  = {0};
  const uint works[1] = {_divisor};
//--- OpenCL プログラムを起動する
  ulong start=GetMicrosecondCount();
  if(!CLExecute(clKrn, 1, offs, works))
    {
     Print("CLExecute(clKrn, 1, offs, works) failed! Error ", GetLastError());
     CLFreeAll(clMem, clKrn, clPrg, clCtx);
     return(-1);
    }
//--- OpenCL デバイスから結果を取得する
  vector buffer(_divisor);
  if(!CLBufferRead(clMem, 0, buffer))
    {
     Print("CLBufferRead(clMem, 0, buffer) failed! Error ", GetLastError());
     CLFreeAll(clMem, clKrn, clPrg, clCtx);
     return(-1);
    }
//--- すべての値を合計して Pi を計算する
  double Pi=buffer.Sum()*_step;

  double time=(GetMicrosecondCount()-start)/1000.;
  Print("OpenCL: Pi calculated for "+D2S(time, 2)+" ms");
  Print("Pi = "+DoubleToString(Pi, 12));
//--- メモリを開放する
  CLFreeAll(clMem, clKrn, clPrg, clCtx);
//--- 成功
  return(0);
 }
  /*
  Pi Calculation: step = 0.000000001000; _intrnCnt = 25000
  OpenCL: GPU device 'Ellesmere' selected
  OpenCL: Pi calculated for 99.98 ms
  Pi = 3.141592653590
  */
```

© 2000–2025, MetaQuotes Ltd.

CLExecute

この関数は OpenCL プログラムを実行します。この関数には 3 つのバージョンがあります。

1. 単数のカーネルを使用してのカーネル関数起動

```
bool  CLExecute(
   int           kernel                  // OpenCL プログラムカーネルのハンドル
   );
```

2. 作業空間の記述を使用した複数のカーネル(OpenCL 関数)の起動

```
bool  CLExecute(
   int           kernel,                 // OpenCL プログラムカーネルのハンドル
   uint          work_dim,               // 作業空間の寸法
   const uint&   global_work_offset[],   // 作業空間の初期オフセット
   const uint&   global_work_size[]      // 作業の数
   );
```

3. 作業空間の記述とグループのローカル作業サブセットのサイズの指定とともの複数のカーネル(OpenCL関数)の起動

```
bool  CLExecute(
   int           kernel,                 // OpenCL プログラムカーネルのハンドル
   uint          work_dim,               // 作業空間の寸法
   const uint&   global_work_offset[],   // 作業空間の初期オフセット
   const uint&   global_work_size[],     // 作業の数
   const uint&   local_work_size[]       // ローカルグループの作業数
   );
```

パラメータ

kernel

[in] OpenCL カーネルのハンドル

work_dim

[in] 作業空間の寸法

global_work_offset[]

[in] 作業空間の初期オフセット

global_work_size[]

[in] 作業サブセットのサイズ

local_work_size[]

[in] グループのローカル作業サブセットのサイズ

戻り値

成功の場合は true、それ以外の場合は false。エラー情報を取得するには、GetLastError() 関数が呼ばれます。

注意事項

© 2000-2025, MetaQuotes Ltd.

下記の例でのパラメータの使用の仕方を見てみましょう。

- *work_dim* は作業を記述する *work_items[]* 配列の次元を指定します。*work_dim*=3 の場合、三次元の配列 *work_items[N1, N2, N3]* が使用されます。
- *global_work_size[]* は *work_items[]* 配列のサイズを設定する値を含みます。work_dim=3 の場合、*global_work_size[3]* 配列は例えば {40, 100, 320} です。その場合 *work_items[40, 100, 320]* です。従って、作業数の合計は 40 x 100 x 320 = 1 280 000 です。
- *local_work_size[]* は指定された OpenCL プログラムのカーネルで実行される作業のサブセットを設定します。そのサイズは *work_items[]* のサイズと等しく、除算での残りの部分を失うことなく、一般的なタスクの一部をより小さなサブセットに分割することが出来ます。実際には *local_work_size[]* 配列のサイズは*work_items[]* グローバル作業セットがより小さいサブセットに分けられるように選択されるべきです。この例では *work_items[40, 100, 320]* は除算のあまりなしで *local_items[10, 10, 10]* から得られるため *local_work_size[3]={10, 10, 10}* が可能です。

© 2000–2025, MetaQuotes Ltd.

CLExecutionStatus

OpenCLプログラム実行ステータスを返します。

```
int  CLExecutionStatus(
   int   kernel            // OpenCLプログラムのカーネルへのハンドル
   );
```

パラメータ

kernel

[in] OpenCL プログラムカーネルのハンドル

戻り値

OpenCLプログラム実行ステータスを返します。可能な値は下記の通りです。

- CL_COMPLETE=0 - プログラム完了
- CL_RUNNING=1 - 実行中
- CL_SUBMITTED=2 - 実行のために提出された
- CL_QUEUED=3 - キューに入れられた
- -1（マイナス1）- CLExecutionStatus()の実行中にエラーが発生した

© 2000-2025, MetaQuotes Ltd.

データベースの操作

データベースを操作するための関数は、一般的で使いやすいSQLiteエンジンを使用します。このエンジンの便利な機能は、データベース全体がユーザーPCのハードディスク上の単一のファイルにあることです。

これらの関数を使用すると、簡単なSQL要求を使用して、テーブルの作成、データの追加、変更の実行、サンプリングを簡単に行うことができます。

- 任意の形式からの取引履歴と相場の受け取り
- 最適化とテスト結果の保存
- 他の分析パッケージとのデータの準備および交換
- MQL5アプリケーションの設定とステータスの保存

クエリでは、統計関数と数学関数を使用できます。

データベースを使用するための関数を使用すると、最も頻繁に繰り返される大きなデータ配列処理操作をSQL要求に置き換えることができるため、多くの場合、複雑なループと比較をプログラミングする代わりにDatabaseExecute/DatabasePrepareは、呼び出すことができます。DatabaseReadBind関数を使用すると、既成の構造体でクエリ結果を簡単に取得できます。この関数では、1回の呼び出しですべてのレコードフィールドを一度に読み取ることができます。

読み取り、書き込み、変更を高速化するために、データベースはDATABASE_OPEN_MEMORYフラグを使用してRAMで開いたり作成したりできますが、このようなデータベースは特定のアプリケーションでのみ使用でき、共有されません。ハードディスク上にあるデータベースを使用する場合、バルクデータの挿入/変更は、DatabaseTransactionBegin/DatabaseTransactionCommit/DatabaseTransactionRollbackを使用してトランザクションにラップする必要があります。これにより、プロセスが何百回も加速されます。

関数の使用をはじめるにあたって、SQLite: MQL5 での SQL データベースのネイティブな処理稿をお読みください。

関数	アクション
DatabaseOpen	指定したファイルでデータベースを開くか作成します
DatabaseClose	データベースを閉じます
DatabaseImport	ファイルからテーブルにデータをインポートします。
DatabaseExport	テーブルまたはSQL要求の実行結果をCSVファイルにエクスポートします。ファイルはUTF-8エンコードで作成されます。
DatabasePrint	テーブルまたはSQL要求の実行結果をエクスパート操作ログに出力します。
DatabaseTableExists	データベース内の表の存在を確認します
DatabaseExecute	指定されたデータベースへのリクエストを実行します
DatabasePrepare	DatabaseRead()を使用して実行できるリクエストのハンドルを作成します
DatabaseReset	DatabasePrepare()の呼び出し後などに、リクエストをリセットします
DatabaseBind	リクエストにパラメータ値を設定します

© 2000-2025, MetaQuotes Ltd.

関数	アクション
DatabaseBindArray	パラメータ値として配列を設定します
DatabaseRead	リクエストの結果として次のエントリに移動します
DatabaseFinalize	DatabasePrepare()で作成されたリクエストを削除します
DatabaseTransactionBegin	トランザクションの実行を開始します
DatabaseTransactionCommit	トランザクションの実行を完了します
DatabaseTransactionRollback	トランザクションをロールバックします
DatabaseColumnsCount	リクエスト内のフィールドの数を取得します
DatabaseColumnName	インデックスでフィールド名を取得します
DatabaseColumnType	インデックスでフィールドタイプを取得します
DatabaseColumnSize	バイト単位のフィールドサイズを取得します
DatabaseColumnText	現在のレコードからフィールド値を文字列として取得します
DatabaseColumnInteger	現在のレコードからint型の値を取得します
DatabaseColumnLong	現在のレコードからlong型の値を取得します
DatabaseColumnDouble	現在のレコードからdouble型の値を取得します
DatabaseColumnBlob	現在のレコードからフィールド値を配列として取得します

統計関数:

- mode - mode
- median - median (50パーセンタイル)
- percentile_25 - 25パーセンタイル
- percentile_75
- percentile_90
- percentile_95
- percentile_99
- stddev or stddev_samp — 標本の標準偏差
- stddev_pop — 母集団の標準偏差
- variance or var_samp — 標本の分散
- var_pop — 母集団の分散

数学関数

- acos(X) – 逆余弦(ラジアン単位)
- acosh(X) – 双曲線逆余弦
- asin(X) – 逆正弦(ラジアン単位)
- asinh(X) – 双曲線逆正弦

© 2000–2025, MetaQuotes Ltd.

- atan(X) – 逆正接(ラジアン単位)
- atan2(X,Y) – X/Y率逆正接(ラジアン単位)
- atanh(X) – 双曲線逆正接
- ceil(X) – 整数に切り上げる
- ceiling(X) – 整数に切り上げる
- cos(X) – 余弦(ラジアン単位)
- cosh(X) – 双曲線余弦
- degrees(X) – ラジアンを角度に変換
- exp(X) – 累乗
- floor(X) – 整数に切り下げる
- ln(X) – 自然対数
- log(B,X) – 示された基数に対する対数
- log(X) – 十進対数
- log10(X) – 十進対数
- log2(X) – 二進対数
- mod(X,Y) – 除算の余り
- pi() – おおよその円周率
- pow(X,Y) – 示された基数に対する累乗
- power(X,Y) – 示された基数に対する累乗
- radians(X) – 角度をラジアンに変換
- sin(X) – サイン(ラジアン単位)
- sinh(X) – 双曲線正弦
- sqrt(X) – 平方根
- tan(X) – 正接(ラジアン単位)
- tanh(X) – 双曲線正接
- trunc(X) – 0に最も近い整数に切り捨てる(絶対値を切り捨て、元の符号を保つ)

例:

```
select
  count(*) as book_count,
  cast(avg(parent) as integer) as mean,
  cast(median(parent) as integer) as median,
  mode(parent) as mode,
  percentile_90(parent) as p90,
  percentile_95(parent) as p95,
  percentile_99(parent) as p99
from moz_bookmarks;
```

© 2000–2025, MetaQuotes Ltd.

DatabaseOpen

指定したファイルでデータベースを開くか作成します。

```
int  DatabaseOpen(
   string  filename,       // ファイル名
   uint    flags           // フラグの組み合わせ
   );
```

パラメータ

filename

[in]「MQL5¥Files」フォルダに相対するファイル名

flags

[in] ENUM_DATABASE_OPEN_FLAGS列挙体からのフラグの組み合わせ

戻り値

正常に実行されると、関数はデータベースハンドルを返します。データベースハンドルは、データベースへのアクセスに使用されます。その他の場合はINVALID_HANDLEを返します。エラーコードを受け取るにはGetLastError() 関数を使用します。以下の応答が可能です。

- ERR_INTERNAL_ERROR (4001) – 重大なランタイムエラー
- ERR_WRONG_INTERNAL_PARAMETER (4002) –「MQL5¥Files」フォルダへのアクセス中の内部エラー
- ERR_INVALID_PARAMETER (4003) – データベースファイルへのパスに空の文字列が含まれているか、フラグの互換性のない組み合わせが設定されています
- ERR_NOT_ENOUGH_MEMORY (4004) – 不充分なメモリ
- ERR_WRONG_FILENAME (5002) – 不正なデータベースファイル名
- ERR_TOO_LONG_FILENAME (5003) – データベースファイルへの絶対パスが最大長を超えています
- ERR_DATABASE_TOO_MANY_OBJECTS (5122) – データベースオブジェクトの最大許容数を超えています
- ERR_DATABASE_CONNECT (5123) – データベース接続エラー
- ERR_DATABASE_MISUSE (5621) – SQLiteライブラリの誤使用

注意事項

*filename*パラメータがNULLまたは空の文字列""の場合、一時ファイルがディスク上に作成されて、データベース接続を閉じた後、自動的に削除されます。

*filename*パラメータが「:memory:」の場合、データベースがメモリ内に作成され、接続が閉じられた後に自動的に削除されます。

*flags*パラメータにDATABASE_OPEN_READONLYもDATABASE_OPEN_READWRITEフラグもない場合、DATABASE_OPEN_READWRITEフラグが使用されます。

ファイル拡張子が指定されていない場合は「.sqlite」が使用されます。

ENUM_DATABASE_OPEN_FLAGS

ID	説明
DATABASE_OPEN_READONLY	読み取り専用

© 2000–2025, MetaQuotes Ltd.

ID	説明
DATABASE_OPEN_READWRITE	読み取りおよび書き込み用に開きます
DATABASE_OPEN_CREATE	必要に応じてディスク上にファイルを作成します
DATABASE_OPEN_MEMORY	RAMにデータベースを作成します
DATABASE_OPEN_COMMON	ファイルはすべての端末の共通フォルダーにあります

参照

DatabaseClose

© 2000–2025, MetaQuotes Ltd.

DatabaseClose

データベースを作成します。

```
void  DatabaseClose(
   int  database      // DatabaseOpenで受信したデータベースハンドル
   );
```

パラメータ

database

[in] DatabaseOpen()で受信したデータベースハンドル。

戻り値

なし

注意事項

DatabaseCloseを呼び出した後で、データベースへのすべてのリクエストのハンドルは自動的に削除されて無効になります。

ハンドルが無効な場合、関数はERR_DATABASE_INVALID_HANDLEエラーを設定します。エラーはGetLastError()を使用してチェックできます。

参照

DatabaseOpen、DatabasePrepare

© 2000-2025, MetaQuotes Ltd.

DatabaseImport

ファイルからテーブルにデータをインポートします。

```
long  DatabaseImport(
   int           database,          // DatabaseOpenで受信したデータベースハンドル
   const string  table,             // データを挿入するテーブルの名前
   const string  filename,          // データをインポートするファイルの名前
   uint          flags,             // フラグの組み合わせ
   const string  separator,         // データの区切り文字
   ulong         skip_rows,         // 最初にスキップする文字列の数
   const string  skip_comments      // コメントを定義する文字列
   );
```

パラメータ

database

[in] DatabaseOpen()で受信したデータベースハンドル。

table

[in] ファイルのデータを追加するテーブルの名前。

filename

[in] データを読み取るためのCSVファイルまたはZIPアーカイブ。名前にはサブディレクトリを含めることができ、MQL5¥Filesフォルダーに相対して設定されます。

flags

[in] ENUM_DATABASE_IMPORT_FLAGS列挙体からのフラグの組み合わせ。

separator

[in] CSVファイルでのデータの区切り文字

skip_rows

[in] ファイルからデータを読み取るときにスキップされる初期文字列の数。

skip_comments

[in] 文字列をコメントとして指定するための文字列。*skip_comments*文字が文字列の先頭で検出された場合、そのような文字列はコメントと見なされ、インポートされません。

戻り値

インポートされた文字列の数を返すか、エラーの場合は-1を返します。エラーコードを受け取るにはGetLastError()を使用します。以下の応答が可能です。

- ERR_INVALID_PARAMETER (4003) – テーブル名が指定されていない(空の文字列またはNULL)
- ERR_DATABASE_INTERNAL (5120) – 内部データベースエラー
- ERR_DATABASE_INVALID_HANDLE (5121) – 無効なデータベースハンドル

注意事項

*table*という名前のテーブルは存在しません。自動的に生成されます。作成されたテーブルの名前とフィールドタイプは、ファイルデータに基づいて自動的に定義されます。

© 2000–2025, MetaQuotes Ltd.

ENUM_DATABASE_IMPORT_FLAGS

ID	説明
DATABASE_IMPORT_HEADER	最初の行にはテーブルフィールドの名前が含まれます。
DATABASE_IMPORT_CRLF	CRLF（デフォルトは　LF）は文字列の区切りと見なされます。
DATABASE_IMPORT_APPEND	既存のテーブルの末尾にデータを追加します。
DATABASE_IMPORT_QUOTED_STRINGS	二重引用符で囲まれた文字列値。
DATABASE_IMPORT_COMMON_FOLDER	ファイルは全クライアント端末の共通フォルダ¥Terminal¥Common¥Fileに保存されます。

DatabaseExportの例のコードで作成された、ファイルからテーブルを読み取る例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
  string csv_filename;
//--- クライアント端末の共通フォルダからダウンロードするテキストファイルの名前を取得する
  string filenames[];
  if(FileSelectDialog("Select a CSV file to download a table", NULL,
                      "Text files (*.csv)|*.csv",
                      FSD_WRITE_FILE|FSD_COMMON_FOLDER, filenames, "data.csv")>0)
   {
    //--- 選択した各ファイルの名前を表示する
    if(ArraySize(filenames)==1)
      csv_filename=filenames[0];
    else
     {
      Print("Unknown error while selecting file. Error code ", GetLastError());
      return;
     }
   }
  else
   {
    Print("CSV file not selected");
    return;
   }
//--- データベースを作成または開く
  string db_filename="test.sqlite";
  int db=DatabaseOpen(db_filename, DATABASE_OPEN_READWRITE|DATABASE_OPEN_CREATE);
//--- TESTテーブルの存在をチェックする
  if(DatabaseTableExists(db, "TEST"))
```

© 2000–2025, MetaQuotes Ltd.

```
    {
     //--- TESTテーブルを削除する
     if(!DatabaseExecute(db, "DROP TABLE IF EXISTS TEST"))
      {
       Print("Failed to drop the TEST table with code ", GetLastError());
       DatabaseClose(db);
       return;
      }
    }
  //--- ファイルからTESTテーブルにエントリをインポートする
  long imported=DatabaseImport(db, "TEST", csv_filename, DATABASE_IMPORT_HEADER|
DATABASE_IMPORT_COMMON_FOLDER|DATABASE_IMPORT_APPEND, ";", 0, NULL);
  if(imported>0)
    {
     Print(imported," lines imported in table TEST");
     DatabasePrint(db,"SELECT * FROM TEST",DATABASE_PRINT_NO_INDEX);
    }
  else
     {
      Print("DatabaseImport() failed. Error ",GetLastError());
     }
//--- データベースファイルを閉じて、その件を通知する
  DatabaseClose(db);
  PrintFormat("Database: %s closed", db_filename);
 }
```

参照

DatabaseOpen、DatabasePrint

© 2000-2025, MetaQuotes Ltd.

DatabaseExport

テーブルまたはSQL要求の実行結果をCSVファイルにエクスポートします。ファイルはUTF-8エンコードで作成されます。

```
long  DatabaseExport(
   int           database,          // DatabaseOpenで受信したデータベースハンドル
   const string  table_or_sql,      // テーブル名またはSQL要求
   const string  filename,          // データエクスポートのためのCSVファイル名
   uint          flags,             // フラグの組み合わせ
   const string  separator          // CSVファイルでの区切り文字
   );
```

パラメータ

database

[in] DatabaseOpen()で受信したデータベースハンドル。

table_or_sql

[in] 結果が指定されたファイルにエクスポートされるテーブルの名前またはSQL要求のテキスト。

filename

[in] データエクスポートのファイル名。パスはMQL5¥Filesフォルダーを基準にして設定されます。

flags

[in] ENUM_DATABASE_EXPORT_FLAGS列挙体からのフラグの組み合わせ。

separator

[in] データの区切り文字。NULLを指定すると、区切り文字として「¥t」タブ文字が使用されます。空の文字列「""」は有効な区切り文字と見なされますが、取得されたCSVファイルはテーブルとして読み取ることができず、文字列のセットと見なされます。

戻り値

エクスポートされたエントリの数またはエラーの場合は負の値を返します。エラーコードを受け取るにはGetLastError()を使用します。以下の応答が可能です。

- ERR_INTERNAL_ERROR (4001) – 重大なランタイムエラー
- ERR_INVALID_PARAMETER (4003) – データベースファイルへのパスに空の文字列が含まれているか、フラグの互換性のない組み合わせが設定されています
- ERR_NOT_ENOUGH_MEMORY (4004) – 不充分なメモリ
- ERR_FUNCTION_NOT_ALLOWED(4014) – 指定されたパイプは許可されていません
- ERR_PROGRAM_STOPPED(4022) – 操作がキャンセルされました(MQLプログラムが停止しました)
- ERR_WRONG_FILENAME (5002) – 無効なファイル名
- ERR_TOO_LONG_FILENAME (5003) – ファイルへの絶対パスが最大長を超えています
- ERR_CANNOT_OPEN_FILE(5004) – 書き込み用にファイルを開けません
- ERR_FILE_WRITEERROR(5026) – ファイルに書き込みできません
- ERR_DATABASE_INTERNAL (5120) – 内部データベースエラー
- ERR_DATABASE_INVALID_HANDLE (5121) – 無効なデータベースハンドル
- ERR_DATABASE_QUERY_PREPARE(5125) – リクエスト生成エラー

© 2000–2025, MetaQuotes Ltd.

- ERR_DATABASE_QUERY_NOT_READONLY – 読み取り専用リクエストが許可されています

注意事項

要求結果がエクスポートされる場合、SQL要求は「SELECT」または「select」で始まる必要があります。つまり、SQL要求はデータベースのステータスを変更できません。そうしないと、DatabaseExport()はエラーで失敗します。

データベース文字列値には、変換文字(「¥r」または「¥r¥n」)、および*separator*パラメーターに設定された値区切り文字を含めることができます。この場合、「flags」パラメーターでDATABASE_EXPORT_QUOTED_STRINGSフラグを使用してください。このフラグが存在する場合、表示されるすべての文字列は二重引用符で囲まれます。文字列に含まれる二重引用符は、2つの二重引用符に置き換えられます。

ENUM_DATABASE_EXPORT_FLAGS

ID	説明
DATABASE_EXPORT_HEADER	最初の文字列のフィールド名を表示します。
DATABASE_EXPORT_INDEX	文字列インデックスを表示します。
DATABASE_EXPORT_NO_BOM	ファイルの先頭にBOMマークを挿入しません(BOMはデフォルトで挿入されます)。
DATABASE_EXPORT_CRLF	文字列の区切りにCRLFを使用します(デフォルトはLF)。
DATABASE_EXPORT_APPEND	既存のファイルの最後にデータを追加します(デフォルトでは、ファイルは上書きされます)。ファイルが存在しない場合、作成されます。
DATABASE_EXPORT_QUOTED_STRINGS	文字列値を二重引用符で囲まれて表示します。
DATABASE_EXPORT_COMMON_FOLDER	CSVファイルは全クライアント端末の共通フォルダ¥Terminal¥Common¥Fileに作成されます。

例:

```
input int InpRates=100;
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
  MqlRates rates[];
//--- バーを受け取る前に開始時間を覚えておく
  ulong start=GetMicrosecondCount();
//--- 最後の100バーをリクエストする
  if(CopyRates(Symbol(), PERIOD_H1, 1, InpRates, rates)<InpRates)
```

© 2000–2025, MetaQuotes Ltd.

```
    {
     Print("CopyRates() failed,, Error ", GetLastError());
     return;
    }
  else
    {
     //--- 受信されたバーの数と受信にかかった時間
     PrintFormat("%s: CopyRates received %d bars in %d ms ",
                 _Symbol, ArraySize(rates), (GetMicrosecondCount()-start)/1000);
    }
//--- データベースを保存するためのファイル名を設定する
  string filename=_Symbol+"_"+EnumToString(PERIOD_H1)
+"_"+TimeToString(TimeCurrent())+".sqlite";
  StringReplace(filename, ":", "-"); // ":" character is not allowed in file names
//--- 共通の端末フォルダーでデータベースを作成または開く
  int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE|DATABASE_OPEN_CREATE|
DATABASE_OPEN_COMMON);
  if(db==INVALID_HANDLE)
    {
     Print("Database: ", filename, " open failed with code ", GetLastError());
     return;
    }
  else
    Print("Database: ", filename, " opened successfully");

//--- RATESテーブルの存在をチェックする
  if(DatabaseTableExists(db, "RATES"))
    {
     //--- RATESテーブルを削除する
     if(!DatabaseExecute(db, "DROP TABLE IF EXISTS RATES"))
       {
        Print("Failed to drop the RATES table with code ", GetLastError());
        DatabaseClose(db);
        return;
       }
    }
//--- RATESテーブルを作成する
  if(!DatabaseExecute(db, "CREATE TABLE RATES("
                      "SYMBOL             CHAR(10),"
                      "TIME               INT NOT NULL,"
                      "OPEN               REAL,"
                      "HIGH               REAL,"
                      "LOW                REAL,"
                      "CLOSE              REAL,"
                      "TICK_VOLUME        INT,"
                      "SPREAD             INT,"
                      "REAL_VOLUME        INT);"))
    {
     Print("DB: ", filename, " create table RATES with code ", GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
    DatabaseClose(db);
    return;
   }
//--- RATESテーブルのすべてのフィールドのリストを表示する
  if(DatabasePrint(db, "PRAGMA TABLE_INFO(RATES)", 0)<0)
   {
    PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO(RATES)\") failed, error code=%d
at line %d", GetLastError(), __LINE__);
    DatabaseClose(db);
    return;
   }
//--- RATESテーブルにバーを追加するためにパラメータ化されたリクエストを作成する
  string sql="INSERT INTO RATES
(SYMBOL,TIME,OPEN,HIGH,LOW,CLOSE,TICK_VOLUME,SPREAD,REAL_VOLUME)"
             " VALUES (?1,?2,?3,?4,?5,?6,?7,?8,?9)"; // request parameters
  int request=DatabasePrepare(db, sql);
  if(request==INVALID_HANDLE)
   {
    PrintFormat("DatabasePrepare() failed with code=%d", GetLastError());
    Print("SQL request: ", sql);
    DatabaseClose(db);
    return;
   }
//--- 最初のリクエストパラメータの値を設定する
  DatabaseBind(request, 0, _Symbol);
//--- RATESテーブルにバーを追加する前に開始時間を覚えておく
  start=GetMicrosecondCount();
  DatabaseTransactionBegin(db);
  int total=ArraySize(rates);
  bool request_error=false;
  for(int i=0; i<total; i++)
   {
    //--- エントリを追加する前に残りのパラメータの値を設定する
    ResetLastError();
    if(!DatabaseBind(request, 1, rates[i].time))
     {
      PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
      PrintFormat("Bar #%d line=%d", i+1, __LINE__);
      request_error=true;
      break;
     }
    //--- 以前のDatabaseBind()呼び出しが成功した場合、次のパラメータを設定する
    if(!request_error && !DatabaseBind(request, 2, rates[i].open))
     {
      PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
      PrintFormat("Bar #%d line=%d", i+1, __LINE__);
      request_error=true;
      break;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
    if(!request_error && !DatabaseBind(request, 3, rates[i].high))
     {
      PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
      PrintFormat("Bar #%d line=%d", i+1, __LINE__);
      request_error=true;
      break;
     }
    if(!request_error && !DatabaseBind(request, 4, rates[i].low))
     {
      PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
      PrintFormat("Bar #%d line=%d", i+1, __LINE__);
      request_error=true;
      break;
     }
    if(!request_error && !DatabaseBind(request, 5, rates[i].close))
     {
      PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
      PrintFormat("Bar #%d line=%d", i+1, __LINE__);
      request_error=true;
      break;
     }
    if(!request_error && !DatabaseBind(request, 6, rates[i].tick_volume))
     {
      PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
      PrintFormat("Bar #%d line=%d", i+1, __LINE__);
      request_error=true;
      break;
     }
    if(!request_error && !DatabaseBind(request, 7, rates[i].spread))
     {
      PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
      PrintFormat("Bar #%d line=%d", i+1, __LINE__);
      request_error=true;
      break;
     }
    if(!request_error && !DatabaseBind(request, 8, rates[i].real_volume))
     {
      PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
      PrintFormat("Bar #%d line=%d", i+1, __LINE__);
      request_error=true;
      break;
     }

    //--- エントリを挿入するリクエストを実行し、エラーを確認する
    if(!request_error && !DatabaseRead(request) &&
(GetLastError()!=ERR_DATABASE_NO_MORE_DATA))
     {
      PrintFormat("DatabaseRead() failed with code=%d", GetLastError());
      DatabaseFinalize(request);
```

© 2000–2025, MetaQuotes Ltd.

```
      request_error=true;
      break;
     }
    //--- 次のパラメータ更新の前にリクエストをリセットする
    if(!request_error && !DatabaseReset(request))
     {
      PrintFormat("DatabaseReset() failed with code=%d", GetLastError());
      DatabaseFinalize(request);
      request_error=true;
      break;
     }
   } //--- すべてのバーの処理が完了した

//--- トランザクション状態
  if(request_error)
   {
    PrintFormat("Table RATES: failed to add %d bars ", ArraySize(rates));
    DatabaseTransactionRollback(db);
    DatabaseClose(db);
    return;
   }
  else
   {
    DatabaseTransactionCommit(db);
    PrintFormat("Table RATES: added %d bars in %d ms",
                ArraySize(rates), (GetMicrosecondCount()-start)/1000);
   }
//--- RATESテーブルをCSVファイルに保存する
  string csv_filename=Symbol()+".csv";
  long saved=DatabaseExport(db, "SELECT * FROM RATES", csv_filename,
DATABASE_EXPORT_HEADER|DATABASE_EXPORT_INDEX|DATABASE_EXPORT_COMMON_FOLDER, ";");
  if(saved>0)
    Print("Table RATES saved in ", Symbol(), ".csv");
  else
    Print("DatabaseExport() failed. Error ", GetLastError());
//--- データベースファイルを閉じて、その件を通知する
  DatabaseClose(db);
  PrintFormat("Database: %s created and closed", filename);
```

参照

DatabasePrint、DatabaseImport

© 2000-2025, MetaQuotes Ltd.

DatabasePrint

テーブルまたはSQL要求の実行結果をエクスパート操作ログに出力します。

```
long  DatabasePrint(
   int           database,          // DatabaseOpenで受信したデータベースハンドル
   const string  table_or_sql,      // テーブルまたはSQL要求
   uint          flags              // フラグの組み合わせ
   );
```

パラメータ

database

[in] DatabaseOpen()で受信したデータベースハンドル。

table_or_sql

[in] 結果がエキスパート操作ログに表示されるテーブルの名前またはSQL要求のテキスト。

フラグ

[in] 出力形式を設定するフラグの組み合わせ。フラグは下記の様に定義されます。

DATABASE_PRINT_NO_HEADER – テーブルの列名（フィールド名）を表示しない
DATABASE_PRINT_NO_INDEX – 文字列インデックスを表示しない
DATABASE_PRINT_NO_FRAME do– ヘッダーとデータを分離するフレームを表示しない
DATABASE_PRINT_STRINGS_RIGHT – 文字列を右に揃える

flags=0の場合、列と文字列が表示され、ヘッダーとデータはフレームで区切られ、文字列は左に揃えられます。

戻り値

エクスポートされた文字列の数を返すか、エラーの場合は-1を返します。エラーコードを受け取るにはGetLastError()を使用します。以下の応答が可能です。

- ERR_INTERNAL_ERROR (4001) – 重大なランタイムエラー
- ERR_NOT_ENOUGH_MEMORY (4004) – 不充分なメモリ
- ERR_DATABASE_INTERNAL (5120) – 内部データベースエラー
- ERR_DATABASE_INVALID_HANDLE (5121) – 無効なデータベースハンドル

注意事項

操作ログに要求結果が表示される場合、SQL要求は「SELECT」または「select」で始まる必要があります。つまり、SQL要求はデータベースのステータスを変更できません。そうしないと、DatabasePrint() はエラーで失敗します。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string filename="departments.sqlite";
//--- 共通の端末フォルダーでデータベースを作成または開きます
```

© 2000–2025, MetaQuotes Ltd.

```
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE |
DATABASE_OPEN_COMMON);
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }

//--- COMPANYテーブルを作成する
   if(!CreataTableCompany(db))
     {
      DatabaseClose(db);
      return;
     }
//--- DEPARTMENTテーブルを作成する
   if(!CreataTableDepartment(db))
     {
      DatabaseClose(db);
      return;
     }

//--- COMPANYおよびDEPARTMENTテーブルのすべてのフィールドのリストを表示する
   PrintFormat("Try to print request \"PRAGMA TABLE_INFO(COMPANY);PRAGMA
TABLE_INFO(DEPARTMENT)\"");
   if(DatabasePrint(db, "PRAGMA TABLE_INFO(COMPANY);PRAGMA TABLE_INFO(DEPARTMENT)",
0)<0)
     {
      PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO()\") failed, error code=%d",
GetLastError());
      DatabaseClose(db);
      return;
     }
//--- COMPANYテーブルをログに表示する
   PrintFormat("Try to print request \"SELECT * from COMPANY\"");
   if(DatabasePrint(db, "SELECT * from COMPANY", 0)<0)
     {
      Print("DatabasePrint failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
//--- COMPANYテーブルとDEPARTMENTテーブルを組み合わせるためのテキストをリクエストする
   string request="SELECT EMP_ID, NAME, DEPT FROM COMPANY LEFT OUTER JOIN DEPARTMENT
"
                  "ON COMPANY.ID = DEPARTMENT.EMP_ID";
//--- 組み合わせた結果のテーブルを表示する
   PrintFormat("Try to print request \"SELECT EMP_ID, NAME, DEPT FROM COMPANY LEFT
OUTER JOIN DEPARTMENT\"");
   if(DatabasePrint(db, request, 0)<0)
     {
      Print("DatabasePrint failed with code ", GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
      DatabaseClose(db);
      return;
     }
//--- データベースを閉じる
   DatabaseClose(db);
  }
/*
結論:
Try to print request "PRAGMA TABLE_INFO(COMPANY);PRAGMA TABLE_INFO(DEPARTMENT)"
#| cid name    type     notnull dflt_value pk
-+-----------------------------------------
1|   0 ID      INT            1             1
2|   1 NAME    TEXT           1             0
3|   2 AGE     INT            1             0
4|   3 ADDRESS CHAR(50)       0             0
5|   4 SALARY  REAL           0             0
#| cid name   type     notnull dflt_value pk
-+-----------------------------------------
1|   0 ID     INT            1             1
2|   1 DEPT   CHAR(50)       1             0
3|   2 EMP_ID INT            1             0
Try to print request "SELECT * from COMPANY"
#| ID NAME  AGE ADDRESS     SALARY
-+--------------------------------
1|  1 Paul   32 California 25000.0
2|  2 Allen  25 Texas      15000.0
3|  3 Teddy  23 Norway     20000.0
4|  4 Mark   25 Rich-Mond  65000.0
5|  5 David  27 Texas      85000.0
6|  6 Kim    22 South-Hall 45000.0
7|  7 James  24 Houston    10000.0
Try to print request "SELECT EMP_ID, NAME, DEPT FROM COMPANY LEFT OUTER JOIN
DEPARTMENT"
#| EMP_ID NAME  DEPT
-+-------------------------
1|      1 Paul  IT Billing
2|      2 Allen Engineering
3|        Teddy
4|        Mark
5|        David
6|        Kim
7|      7 James Finance
*/
//+------------------------------------------------------------------+
//| COMPANYテーブルを作成する                                        |
//+------------------------------------------------------------------+
bool CreateTableCompany(int database)
  {
//--- COMPANYテーブルが存在する場合は削除する
```

© 2000-2025, MetaQuotes Ltd.

```
   if(DatabaseTableExists(database, "COMPANY"))
     {
      //--- delete the table
      if(!DatabaseExecute(database, "DROP TABLE COMPANY"))
        {
         Print("Failed to drop table COMPANY with code ", GetLastError());
         return(false);
        }
     }
//--- COMPANYテーブルを作成する
   if(!DatabaseExecute(database, "CREATE TABLE COMPANY("
                       "ID INT PRIMARY KEY     NOT NULL,"
                       "NAME           TEXT    NOT NULL,"
                       "AGE            INT     NOT NULL,"
                       "ADDRESS        CHAR(50),"
                       "SALARY         REAL );"))
     {
      Print("DB: create table COMPANY failed with code ", GetLastError());
      return(false);
     }

//--- COMPANYテーブルにデータを入力する
   if(!DatabaseExecute(database, "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (1, 'Paul', 32, 'California', 25000.00); "
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (2,
'Allen', 25, 'Texas', 15000.00); "
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (3,
'Teddy', 23, 'Norway', 20000.00); "
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (4,
'Mark', 25, 'Rich-Mond', 65000.00); "
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (5,
'David', 27, 'Texas', 85000.0); "
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (6,
'Kim', 22, 'South-Hall', 45000.0); "
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (7,
'James', 24, 'Houston', 10000.00); "))
     {
      Print("COMPANY insert failed with code ", GetLastError());
      return(false);
     }
//--- 成功
   return(true);
  }
//+------------------------------------------------------------------+
//|  DEPARTMENTテーブルを作成する                                    |
//+------------------------------------------------------------------+
bool CreateTableDepartment(int database)
  {
//--- DEPARTMENTテーブルが存在する場合は削除する
   if(DatabaseTableExists(database, "DEPARTMENT"))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- delete the table
       if(!DatabaseExecute(database, "DROP TABLE DEPARTMENT"))
         {
          Print("Failed to drop table DEPARTMENT  with code ", GetLastError());
          return(false);
         }
      }
//--- DEPARTMENTテーブルを作成する
   if(!DatabaseExecute(database, "CREATE TABLE DEPARTMENT ("
                         "ID      INT PRIMARY KEY   NOT NULL,"
                         "DEPT    CHAR(50)          NOT NULL,"
                         "EMP_ID  INT               NOT NULL);"))
      {
       Print("DB: create table DEPARTMENT failed with code ", GetLastError());
       return(false);
      }

//--- DEPARTMENTテーブルにデータを入力する
   if(!DatabaseExecute(database, "INSERT INTO DEPARTMENT (ID,DEPT,EMP_ID) VALUES (1,
'IT Billing', 1); "
                         "INSERT INTO DEPARTMENT (ID,DEPT,EMP_ID) VALUES (2,
'Engineering', 2); "
                         "INSERT INTO DEPARTMENT (ID,DEPT,EMP_ID) VALUES (3,
'Finance', 7);"))
      {
       Print("DEPARTMENT insert failed with code ", GetLastError());
       return(false);
      }
//--- 成功
   return(true);
  }
//+------------------------------------------------------------------
```

参照

DatabaseExport、DatabaseImport

© 2000-2025, MetaQuotes Ltd.

DatabaseTableExists

データベース内の表の存在を確認します。

```
bool  DatabaseTableExists(
   int     database,      // DatabaseOpenで受信したデータベースハンドル
   string  table          // 表名
   );
```

パラメータ

database

[in] DatabaseOpen()で受信したデータベースハンドル。

table

[in] 表名。

戻り値

成功の場合は true、それ以外の場合は false 。エラーコードを受け取るには GetLastError() 関数を使用します。以下の応答が可能です。

- ERR_INVALID_PARAMETER (4003) – テーブル名が指定されていない(空の文字列またはNULL)
- ERR_WRONG_STRING_PARAMETER (5040) – 要求をUTF-8文字列に変換する際のエラー
- ERR_DATABASE_INTERNAL (5120) – 内部データベースエラー
- ERR_DATABASE_INVALID_HANDLE (5121) – 無効なデータベースハン
- ERR_DATABASE_EXECUTE (5124) – リクエスト実行エラー
- ERR_DATABASE_NO_MORE_DATA (5126) – 表が存在しない(エラーではなく、正常に完了する)

参照

DatabasePrepare、DatabaseFinalize

© 2000–2025, MetaQuotes Ltd.

DatabaseExecute

指定されたデータベースへのリクエストを実行します。

```
bool  DatabaseExecute(
   int      database,       // DatabaseOpenで受信したデータベースハンドル
   string   sql             // SQL要求
   );
```

パラメータ

database

[in] DatabaseOpen()で受信したデータベースハンドル。

sql

[in] SQL要求。

戻り値

成功の場合は true、それ以外の場合は false 。エラーコードを受け取るには GetLastError() 関数を使用します。以下の応答が可能です。

- ERR_INTERNAL_ERROR (4001) – 重大なランタイムエラー
- ERR_INVALID_PARAMETER (4003) – sqlパラメータに空の文字列が含まれています
- ERR_NOT_ENOUGH_MEMORY (4004) – 不充分なメモリ
- ERR_WRONG_STRING_PARAMETER (5040) – 要求をUTF-8文字列に変換する際のエラー
- ERR_DATABASE_INTERNAL (5120) – 内部データベースエラー
- ERR_DATABASE_INVALID_HANDLE (5121) – 無効なデータベースハンドル
- ERR_DATABASE_EXECUTE (5124) – リクエスト実行エラー

Example:

```
//--- symbol statistics
struct Symbol_Stats
  {
   string            name;             // symbol name
   int               trades;           // number of trades for the symbol
   double            gross_profit;     // total profit for the symbol
   double            gross_loss;       // total loss for the symbol
   double            total_commission; // total commission for the symbol
   double            total_swap;       // total swaps for the symbol
   double            total_profit;     // total profit excluding swaps and
commissions
   double            net_profit;       // net profit taking into account swaps and
commissions
   int               win_trades;       // number of profitable trades
   int               loss_trades;      // number of losing trades
   double            expected_payoff;  // expected payoff for the trade excluding
swaps and commissions
   double            win_percent;      // percentage of winning trades
   double            loss_percent;     // percentage of losing trades
```

© 2000–2025, MetaQuotes Ltd.

```
   double            average_profit;   // average profit
   double            average_loss;     // average loss
   double            profit_factor;    // profit factor
  };

//--- Magic Number statistics
struct Magic_Stats
  {
   long              magic;            // EA's Magic Number
   int               trades;           // number of trades for the symbol
   double            gross_profit;     // total profit for the symbol
   double            gross_loss;       // total loss for the symbol
   double            total_commission; // total commission for the symbol
   double            total_swap;       // total swaps for the symbol
   double            total_profit;     // total profit excluding swaps and
commissions
   double            net_profit;       // net profit taking into account swaps and
commissions
   int               win_trades;       // number of profitable trades
   int               loss_trades;      // number of losing trades
   double            expected_payoff;  // expected payoff for the trade excluding
swaps and commissions
   double            win_percent;      // percentage of winning trades
   double            loss_percent;     // percentage of losing trades
   double            average_profit;   // average profit
   double            average_loss;     // average loss
   double            profit_factor;    // profit factor
  };

//--- entry hour statistics
struct Hour_Stats
  {
   char              hour_in;          // market entry hour
   int               trades;           // number of trades in this entry hour
   double            volume;           // volume of trades in this entry hour
   double            gross_profit;     // total profit in this entry hour
   double            gross_loss;       // total loss in this entry hour
   double            net_profit;       // net profit taking into account swaps and
commissions
   int               win_trades;       // number of profitable trades
   int               loss_trades;      // number of losing trades
   double            expected_payoff;  // expected payoff for the trade excluding
swaps and commissions
   double            win_percent;      // percentage of winning trades
   double            loss_percent;     // percentage of losing trades
   double            average_profit;   // average profit
   double            average_loss;     // average loss
   double            profit_factor;    // profit factor
  };
```

© 2000–2025, MetaQuotes Ltd.

```
int ExtDealsTotal=0;;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create the file name
   string filename=IntegerToString(AccountInfoInteger(ACCOUNT_LOGIN))
+"_stats.sqlite";
//--- open/create the database in the common terminal folder
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE |
DATABASE_OPEN_COMMON);
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
//--- create the DEALS table
   if(!CreateTableDeals(db))
     {
      DatabaseClose(db);
      return;
     }
   PrintFormat("Deals in the trading history: %d ", ExtDealsTotal);

//--- get trading statistics per symbols
   int request=DatabasePrepare(db, "SELECT r.*,"
                               "   (case when r.trades != 0 then
(r.gross_profit+r.gross_loss)/r.trades else 0 end) as expected_payoff,"
                               "   (case when r.trades != 0 then
r.win_trades*100.0/r.trades else 0 end) as win_percent,"
                               "   (case when r.trades != 0 then
r.loss_trades*100.0/r.trades else 0 end) as loss_percent,"
                               "   r.gross_profit/r.win_trades as average_profit,"
                               "   r.gross_loss/r.loss_trades as average_loss,"
                               "   (case when r.gross_loss!=0.0 then r.gross_profit/
(-r.gross_loss) else 0 end) as profit_factor "
                               "FROM "
                               "   ("
                               "   SELECT SYMBOL,"
                               "   sum(case when entry =1 then 1 else 0 end) as
trades,"
                               "   sum(case when profit > 0 then profit else 0 end)
as gross_profit,"
                               "   sum(case when profit < 0 then profit else 0 end)
as gross_loss,"
                               "   sum(swap) as total_swap,"
                               "   sum(commission) as total_commission,"
                               "   sum(profit) as total_profit,"
                               "   sum(profit+swap+commission) as net_profit,"
```

© 2000–2025, MetaQuotes Ltd.

```
                                  "   sum(case when profit > 0 then 1 else 0 end) as
win_trades,"
                                  "   sum(case when profit < 0 then 1 else 0 end) as
loss_trades "
                                  "   FROM DEALS "
                                  "   WHERE SYMBOL <> '' and SYMBOL is not NULL "
                                  "   GROUP BY SYMBOL"
                                  "   ) as r");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   Symbol_Stats stats[], symbol_stats;
   ArrayResize(stats, ExtDealsTotal);
   int i=0;
//--- get records from request results
   for(; DatabaseReadBind(request, symbol_stats) ; i++)
     {
      stats[i].name=symbol_stats.name;
      stats[i].trades=symbol_stats.trades;
      stats[i].gross_profit=symbol_stats.gross_profit;
      stats[i].gross_loss=symbol_stats.gross_loss;
      stats[i].total_commission=symbol_stats.total_commission;
      stats[i].total_swap=symbol_stats.total_swap;
      stats[i].total_profit=symbol_stats.total_profit;
      stats[i].net_profit=symbol_stats.net_profit;
      stats[i].win_trades=symbol_stats.win_trades;
      stats[i].loss_trades=symbol_stats.loss_trades;
      stats[i].expected_payoff=symbol_stats.expected_payoff;
      stats[i].win_percent=symbol_stats.win_percent;
      stats[i].loss_percent=symbol_stats.loss_percent;
      stats[i].average_profit=symbol_stats.average_profit;
      stats[i].average_loss=symbol_stats.average_loss;
      stats[i].profit_factor=symbol_stats.profit_factor;
     }
   ArrayResize(stats, i);
   Print("Trade statistics by Symbol");
   ArrayPrint(stats);
   Print("");
//--- delete the request
   DatabaseFinalize(request);

//--- get trading statistics for Expert Advisors by Magic Numbers
   request=DatabasePrepare(db, "SELECT r.*,"
                               "   (case when r.trades != 0 then
(r.gross_profit+r.gross_loss)/r.trades else 0 end) as expected_payoff,"
```

© 2000–2025, MetaQuotes Ltd.

```
                              "   (case when r.trades != 0 then
r.win_trades*100.0/r.trades else 0 end) as win_percent,"
                              "   (case when r.trades != 0 then
r.loss_trades*100.0/r.trades else 0 end) as loss_percent,"
                              "   r.gross_profit/r.win_trades as average_profit,"
                              "   r.gross_loss/r.loss_trades as average_loss,"
                              "   (case when r.gross_loss!=0.0 then r.gross_profit/(-
r.gross_loss) else 0 end) as profit_factor "
                              "FROM "
                              "   ("
                              "   SELECT MAGIC,"
                              "   sum(case when entry =1 then 1 else 0 end) as trades,"
                              "   sum(case when profit > 0 then profit else 0 end) as
gross_profit,"
                              "   sum(case when profit < 0 then profit else 0 end) as
gross_loss,"
                              "   sum(swap) as total_swap,"
                              "   sum(commission) as total_commission,"
                              "   sum(profit) as total_profit,"
                              "   sum(profit+swap+commission) as net_profit,"
                              "   sum(case when profit > 0 then 1 else 0 end) as
win_trades,"
                              "   sum(case when profit < 0 then 1 else 0 end) as
loss_trades "
                              "   FROM DEALS "
                              "   WHERE SYMBOL <> '' and SYMBOL is not NULL "
                              "   GROUP BY MAGIC"
                              "   ) as r");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   Magic_Stats EA_stats[], magic_stats;
   ArrayResize(EA_stats, ExtDealsTotal);
   i=0;
//--- print
   for(; DatabaseReadBind(request, magic_stats) ; i++)
     {
      EA_stats[i].magic=magic_stats.magic;
      EA_stats[i].trades=magic_stats.trades;
      EA_stats[i].gross_profit=magic_stats.gross_profit;
      EA_stats[i].gross_loss=magic_stats.gross_loss;
      EA_stats[i].total_commission=magic_stats.total_commission;
      EA_stats[i].total_swap=magic_stats.total_swap;
      EA_stats[i].total_profit=magic_stats.total_profit;
      EA_stats[i].net_profit=magic_stats.net_profit;
      EA_stats[i].win_trades=magic_stats.win_trades;
      EA_stats[i].loss_trades=magic_stats.loss_trades;
```

© 2000–2025, MetaQuotes Ltd.

```
      EA_stats[i].expected_payoff=magic_stats.expected_payoff;
      EA_stats[i].win_percent=magic_stats.win_percent;
      EA_stats[i].loss_percent=magic_stats.loss_percent;
      EA_stats[i].average_profit=magic_stats.average_profit;
      EA_stats[i].average_loss=magic_stats.average_loss;
      EA_stats[i].profit_factor=magic_stats.profit_factor;
     }
   ArrayResize(EA_stats, i);
   Print("Trade statistics by Magic Number");
   ArrayPrint(EA_stats);
   Print("");
//--- delete the request
   DatabaseFinalize(request);

//--- make sure that hedging system for open position management is used on the
account
   if((ENUM_ACCOUNT_MARGIN_MODE)
AccountInfoInteger(ACCOUNT_MARGIN_MODE)!=ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)
     {
      //--- deals cannot be transformed to trades using a simple method through
transactions, therefore complete operation
      DatabaseClose(db);
      return;
     }

//--- now create the TRADES table based on the DEALS table
   if(!CreateTableTrades(db))
     {
      DatabaseClose(db);
      return;
     }
//--- fill in the TRADES table using an SQL query based on DEALS table data
   if(DatabaseTableExists(db, "DEALS"))
      //--- populate the TRADES table
      if(!DatabaseExecute(db, "INSERT INTO
TRADES(TIME_IN,HOUR_IN,TICKET,TYPE,VOLUME,SYMBOL,PRICE_IN,TIME_OUT,PRICE_OUT,COMMISS
ION,SWAP,PROFIT) "
                              "SELECT "
                              "   d1.time as time_in,"
                              "   d1.hour as hour_in,"
                              "   d1.position_id as ticket,"
                              "   d1.type as type,"
                              "   d1.volume as volume,"
                              "   d1.symbol as symbol,"
                              "   d1.price as price_in,"
                              "   d2.time as time_out,"
                              "   d2.price as price_out,"
                              "   d1.commission+d2.commission as commission,"
                              "   d2.swap as swap,"
                              "   d2.profit as profit "
```

© 2000–2025, MetaQuotes Ltd.

```
                              "FROM DEALS d1 "
                              "INNER JOIN DEALS d2 ON d1.position_id=d2.position_id "
                              "WHERE d1.entry=0 AND d2.entry=1      "))
        {
         Print("DB: fillng the table TRADES failed with code ", GetLastError());
         return;
        }

//--- get trading statistics by market entry hours
   request=DatabasePrepare(db, "SELECT r.*,"
                               "   (case when r.trades != 0 then
(r.gross_profit+r.gross_loss)/r.trades else 0 end) as expected_payoff,"
                               "   (case when r.trades != 0 then
r.win_trades*100.0/r.trades else 0 end) as win_percent,"
                               "   (case when r.trades != 0 then
r.loss_trades*100.0/r.trades else 0 end) as loss_percent,"
                               "   r.gross_profit/r.win_trades as average_profit,"
                               "   r.gross_loss/r.loss_trades as average_loss,"
                               "   (case when r.gross_loss!=0.0 then r.gross_profit/(-
r.gross_loss) else 0 end) as profit_factor "
                               "FROM "
                               "   ("
                               "   SELECT HOUR_IN,"
                               "   count() as trades,"
                               "   sum(volume) as volume,"
                               "   sum(case when profit > 0 then profit else 0 end) as
gross_profit,"
                               "   sum(case when profit < 0 then profit else 0 end) as
gross_loss,"
                               "   sum(profit) as net_profit,"
                               "   sum(case when profit > 0 then 1 else 0 end) as
win_trades,"
                               "   sum(case when profit < 0 then 1 else 0 end) as
loss_trades "
                               "   FROM TRADES "
                               "   WHERE SYMBOL <> '' and SYMBOL is not NULL "
                               "   GROUP BY HOUR_IN"
                               "   ) as r");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   Hour_Stats hours_stats[], h_stats;
   ArrayResize(hours_stats, ExtDealsTotal);
   i=0;
//--- print
   for(; DatabaseReadBind(request, h_stats) ; i++)
     {
```

© 2000–2025, MetaQuotes Ltd.

```
      hours_stats[i].hour_in=h_stats.hour_in;
      hours_stats[i].trades=h_stats.trades;
      hours_stats[i].volume=h_stats.volume;
      hours_stats[i].gross_profit=h_stats.gross_profit;
      hours_stats[i].gross_loss=h_stats.gross_loss;
      hours_stats[i].net_profit=h_stats.net_profit;
      hours_stats[i].win_trades=h_stats.win_trades;
      hours_stats[i].loss_trades=h_stats.loss_trades;
      hours_stats[i].expected_payoff=h_stats.expected_payoff;
      hours_stats[i].win_percent=h_stats.win_percent;
      hours_stats[i].loss_percent=h_stats.loss_percent;
      hours_stats[i].average_profit=h_stats.average_profit;
      hours_stats[i].average_loss=h_stats.average_loss;
      hours_stats[i].profit_factor=h_stats.profit_factor;
     }
   ArrayResize(hours_stats, i);
   Print("Trade statistics by entry hour");
   ArrayPrint(hours_stats);
   Print("");
//--- delete the request
   DatabaseFinalize(request);

//--- close database
   DatabaseClose(db);
   return;
  }
/*
Deals in the trading history: 2771
Trade statistics by Symbol
      [name] [trades] [gross_profit] [gross_loss] [total_commission] [total_swap]
[total_profit] [net_profit] [win_trades] [loss_trades] [expected_payoff]
[win_percent] [loss_percent] [average_profit] [average_loss] [profit_factor]
[0] "AUDUSD"      112      503.20000   -568.00000           -8.83000    -24.64000
  -64.80000    -98.27000           70            42           -0.57857      62.50000
     37.50000          7.18857      -13.52381         0.88592
[1] "EURCHF"      125      607.71000   -956.85000          -11.77000    -45.02000
 -349.14000   -405.93000           54            71           -2.79312      43.20000
     56.80000         11.25389      -13.47676         0.63512
[2] "EURJPY"      127     1078.49000  -1057.83000          -10.61000    -45.76000
   20.66000    -35.71000           64            63            0.16268      50.39370
     49.60630         16.85141      -16.79095         1.01953
[3] "EURUSD"      233     1685.60000  -1386.80000          -41.00000    -83.76000
  298.80000    174.04000          127           106            1.28240      54.50644
     45.49356         13.27244      -13.08302         1.21546
[4] "GBPCHF"      125     1881.37000  -1424.72000          -22.60000    -51.56000
  456.65000    382.49000           80            45            3.65320      64.00000
     36.00000         23.51712      -31.66044         1.32052
[5] "GBPJPY"      127     1943.43000  -1776.67000          -18.84000    -52.46000
  166.76000     95.46000           76            51            1.31307      59.84252
     40.15748         25.57145      -34.83667         1.09386
```

© 2000–2025, MetaQuotes Ltd.

```
[6] "GBPUSD"      121     1668.50000  -1438.20000           -7.96000    -49.93000
  230.30000    172.41000           77            44           1.90331      63.63636
     36.36364         21.66883      -32.68636         1.16013
[7] "USDCAD"       99      405.28000   -475.47000           -8.68000    -31.68000
  -70.19000   -110.55000           51            48           -0.70899      51.51515
     48.48485          7.94667       -9.90563         0.85238
[8] "USDCHF"      206     1588.32000  -1241.83000          -17.98000    -65.92000
  346.49000    262.59000          131            75           1.68199      63.59223
     36.40777         12.12458      -16.55773         1.27902
[9] "USDJPY"      107      464.73000   -730.64000           -35.12000    -34.24000
 -265.91000   -335.27000           50            57           -2.48514      46.72897
     53.27103          9.29460      -12.81825         0.63606


Trade statistics by Magic Number

    [magic] [trades] [gross_profit] [gross_loss] [total_commission] [total_swap]
[total_profit] [net_profit] [win_trades] [loss_trades] [expected_payoff]
[win_percent] [loss_percent] [average_profit] [average_loss] [profit_factor]
[0]     100      242     2584.80000  -2110.00000          -33.36000    -93.53000
 474.80000    347.91000          143            99           1.96198      59.09091
    40.90909         18.07552      -21.31313         1.22502
[1]     200      254     3021.92000  -2834.50000          -29.45000    -98.22000
 187.42000     59.75000          140           114           0.73787      55.11811
    44.88189         21.58514      -24.86404         1.06612
[2]     300      250     2489.08000  -2381.57000          -34.37000    -96.58000
 107.51000    -23.44000          134           116           0.43004      53.60000
    46.40000         18.57522      -20.53078         1.04514
[3]     400      224     1272.50000  -1283.00000          -24.43000    -64.80000
 -10.50000    -99.73000          131            93           -0.04687      58.48214
    41.51786          9.71374      -13.79570         0.99182
[4]     500      198     1141.23000  -1051.91000          -27.66000    -63.36000
  89.32000     -1.70000          116            82           0.45111      58.58586
    41.41414          9.83819      -12.82817         1.08491
[5]     600      214     1317.10000  -1396.03000          -34.12000    -68.48000
 -78.93000   -181.53000          116            98           -0.36883      54.20561
    45.79439         11.35431      -14.24520         0.94346


Trade statistics by entry hour

     [hour_in] [trades] [volume] [gross_profit] [gross_loss] [net_profit]
[win_trades] [loss_trades] [expected_payoff] [win_percent] [loss_percent]
[average_profit] [average_loss] [profit_factor]
[ 0]         0       50  5.00000      336.51000   -747.47000   -410.96000
21            29          -8.21920      42.00000       58.00000         16.02429
 -25.77483         0.45020
[ 1]         1       20  2.00000      102.56000    -57.20000     45.36000
12             8           2.26800      60.00000       40.00000          8.54667
  -7.15000         1.79301
[ 2]         2        6  0.60000       38.55000    -14.60000     23.95000
5              1           3.99167      83.33333       16.66667          7.71000
-14.60000         2.64041
[ 3]         3       38  3.80000      173.84000   -200.15000    -26.31000
22            16          -0.69237      57.89474       42.10526          7.90182
 -12.50938         0.86855
[ 4]         4       60  6.00000      361.44000   -389.40000    -27.96000
27            33          -0.46600      45.00000       55.00000         13.38667
 -11.80000         0.92820
```

© 2000–2025, MetaQuotes Ltd.

```
[ 5]         5       32  3.20000      157.43000   -179.89000    -22.46000
20            12          -0.70187       62.50000        37.50000          7.87150
 -14.99083          0.87515
[ 6]         6       18  1.80000       95.59000   -162.33000    -66.74000
11             7          -3.70778       61.11111        38.88889          8.69000
 -23.19000          0.58886
[ 7]         7       14  1.40000       38.48000   -134.30000    -95.82000
9              5          -6.84429       64.28571        35.71429          4.27556
-26.86000          0.28652
[ 8]         8       42  4.20000      368.48000   -322.30000     46.18000
24            18           1.09952       57.14286        42.85714         15.35333
 -17.90556          1.14328
[ 9]         9      118 11.80000     1121.62000   -875.21000    246.41000
72            46           2.08822       61.01695        38.98305         15.57806
 -19.02630          1.28154
[10]        10      206 20.60000     2280.59000  -2021.80000    258.79000
115           91           1.25626       55.82524        44.17476         19.83122
  -22.21758          1.12800
[11]        11      138 13.80000     1377.02000   -994.18000    382.84000
84            54           2.77420       60.86957        39.13043         16.39310
 -18.41074          1.38508
[12]        12      152 15.20000     1247.56000  -1463.80000   -216.24000
84            68          -1.42263       55.26316        44.73684         14.85190
 -21.52647          0.85227
[13]        13       64  6.40000      778.27000   -516.22000    262.05000
36            28           4.09453       56.25000        43.75000         21.61861
 -18.43643          1.50763
[14]        14       62  6.20000      536.93000   -427.47000    109.46000
38            24           1.76548       61.29032        38.70968         14.12974
 -17.81125          1.25606
[15]        15       50  5.00000      699.92000   -413.00000    286.92000
28            22           5.73840       56.00000        44.00000         24.99714
 -18.77273          1.69472
[16]        16       88  8.80000      778.55000   -514.00000    264.55000
51            37           3.00625       57.95455        42.04545         15.26569
 -13.89189          1.51469
[17]        17       76  7.60000      533.92000  -1019.46000   -485.54000
44            32          -6.38868       57.89474        42.10526         12.13455
 -31.85813          0.52373
[18]        18       52  5.20000      237.17000   -246.78000     -9.61000
24            28          -0.18481       46.15385        53.84615          9.88208
  -8.81357          0.96106
[19]        19       52  5.20000      407.67000   -150.36000    257.31000
30            22           4.94827       57.69231        42.30769         13.58900
  -6.83455          2.71129
[20]        20       18  1.80000       65.92000    -89.09000    -23.17000
9              9          -1.28722       50.00000        50.00000          7.32444
 -9.89889          0.73993
[21]        21       10  1.00000       41.86000    -32.38000      9.48000
7              3           0.94800       70.00000        30.00000          5.98000
-10.79333          1.29277
[22]        22       14  1.40000       45.55000    -83.72000    -38.17000
6              8          -2.72643       42.85714        57.14286          7.59167
-10.46500          0.54408
[23]        23        2  0.20000        1.20000     -1.90000     -0.70000
1              1          -0.35000       50.00000        50.00000          1.20000
 -1.90000          0.63158
```

© 2000–2025, MetaQuotes Ltd.

```
*/

//+------------------------------------------------------------------+
//| Creates the DEALS table                                          |
//+------------------------------------------------------------------+
bool CreateTableDeals(int database)
  {
//--- if the DEALS table already exists, delete it
   if(!DeleteTable(database, "DEALS"))
     {
      return(false);
     }
//--- check if the table exists
   if(!DatabaseTableExists(database, "DEALS"))
      //--- create the table
      if(!DatabaseExecute(database, "CREATE TABLE DEALS("
                          "ID          INT KEY NOT NULL,"
                          "ORDER_ID    INT     NOT NULL,"
                          "POSITION_ID INT     NOT NULL,"
                          "TIME        INT     NOT NULL,"
                          "TYPE        INT     NOT NULL,"
                          "ENTRY       INT     NOT NULL,"
                          "SYMBOL      CHAR(10),"
                          "VOLUME      REAL,"
                          "PRICE       REAL,"
                          "PROFIT      REAL,"
                          "SWAP        REAL,"
                          "COMMISSION  REAL,"
                          "MAGIC       INT,"
                          "HOUR        INT,"
                          "REASON      INT);"))
        {
         Print("DB: create the DEALS table  failed with code ", GetLastError());
         return(false);
        }
//---  request the entire trading history
   datetime from_date=0;
   datetime to_date=TimeCurrent();
//--- request the history of deals in the specified interval
   HistorySelect(from_date, to_date);
   ExtDealsTotal=HistoryDealsTotal();
//--- add deals to the table
   if(!InsertDeals(database))
      return(false);
//--- the table has been successfully created
   return(true);
  }
//+------------------------------------------------------------------+
//| Deletes a table with the specified name from the database        |
```

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool DeleteTable(int database, string table_name)
  {
   if(!DatabaseExecute(database, "DROP TABLE IF EXISTS "+table_name))
     {
      Print("Failed to drop the DEALS table with code ", GetLastError());
      return(false);
     }
//--- the table has been successfully deleted
   return(true);
  }
//+------------------------------------------------------------------+
//| Adds deals to the database table                                 |
//+------------------------------------------------------------------+
bool InsertDeals(int database)
  {
//--- Auxiliary variables
   ulong    deal_ticket;          // deal ticket
   long     order_ticket;         // the ticket of the order by which the deal was
executed
   long     position_ticket;      // ID of the position to which the deal belongs
   datetime time;                 // deal execution time
   long     type ;                // deal type
   long     entry ;               // deal direction
   string   symbol;               // the symbol fro which the deal was executed
   double   volume;               // operation volume
   double   price;                // price
   double   profit;               // financial result
   double   swap;                 // swap
   double   commission;           // commission
   long     magic;                // Magic number (Expert Advisor ID)
   long     reason;               // deal execution reason or source
   char     hour;                 // deal execution hour
   MqlDateTime time_strusture;
//--- go through all deals and add them to the database
   bool failed=false;
   int deals=HistoryDealsTotal();
// --- lock the database before executing transactions
   DatabaseTransactionBegin(database);
   for(int i=0; i<deals; i++)
     {
      deal_ticket=    HistoryDealGetTicket(i);
      order_ticket=   HistoryDealGetInteger(deal_ticket, DEAL_ORDER);
      position_ticket=HistoryDealGetInteger(deal_ticket, DEAL_POSITION_ID);
      time= (datetime)HistoryDealGetInteger(deal_ticket, DEAL_TIME);
      type=           HistoryDealGetInteger(deal_ticket, DEAL_TYPE);
      entry=          HistoryDealGetInteger(deal_ticket, DEAL_ENTRY);
      symbol=         HistoryDealGetString(deal_ticket, DEAL_SYMBOL);
      volume=         HistoryDealGetDouble(deal_ticket, DEAL_VOLUME);
```

© 2000–2025, MetaQuotes Ltd.

```
      price=          HistoryDealGetDouble(deal_ticket, DEAL_PRICE);
      profit=         HistoryDealGetDouble(deal_ticket, DEAL_PROFIT);
      swap=           HistoryDealGetDouble(deal_ticket, DEAL_SWAP);
      commission=     HistoryDealGetDouble(deal_ticket, DEAL_COMMISSION);
      magic=          HistoryDealGetInteger(deal_ticket, DEAL_MAGIC);
      reason=         HistoryDealGetInteger(deal_ticket, DEAL_REASON);
      TimeToStruct(time, time_strusture);
      hour= (char)time_strusture.hour;
      //--- add each deal to the table using the following request
      string request_text=StringFormat("INSERT INTO DEALS
(ID,ORDER_ID,POSITION_ID,TIME,TYPE,ENTRY,SYMBOL,VOLUME,PRICE,PROFIT,SWAP,COMMISSION,
MAGIC,REASON,HOUR)"
                                       "VALUES (%d, %d, %d, %d, %d, %d, '%s', %G, %
G, %G, %G, %G, %d, %d,%d)",
                                       deal_ticket, order_ticket, position_ticket,
time, type, entry, symbol, volume, price, profit, swap, commission, magic, reason,
hour);
      if(!DatabaseExecute(database, request_text))
        {
         PrintFormat("%s: failed to insert deal #%d with code %d", __FUNCTION__,
deal_ticket, GetLastError());
         PrintFormat("i=%d: deal #%d  %s", i, deal_ticket, symbol);
         failed=true;
         break;
        }
     }
//--- check for transaction execution errors
   if(failed)
     {
      //--- roll back all transactions and unlock the database
      DatabaseTransactionRollback(database);
      PrintFormat("%s: DatabaseExecute() failed with code ", __FUNCTION__,
GetLastError());
      return(false);
     }
//--- all transactions have been performed successfully - record changes and unlock
the database
   DatabaseTransactionCommit(database);
   return(true);
  }
//+------------------------------------------------------------------+
//| Creates the TRADES table                                         |
//+------------------------------------------------------------------+
bool CreateTableTrades(int database)
  {
//--- if the TRADES table already exists, delete it
   if(!DeleteTable(database, "TRADES"))
      return(false);
//--- check if the table exists
   if(!DatabaseTableExists(database, "TRADES"))
```

© 2000–2025, MetaQuotes Ltd.

```
      //--- create the table
      if(!DatabaseExecute(database, "CREATE TABLE TRADES("
                          "TIME_IN     INT     NOT NULL,"
                          "HOUR_IN     INT     NOT NULL,"
                          "TICKET      INT     NOT NULL,"
                          "TYPE        INT     NOT NULL,"
                          "VOLUME      REAL,"
                          "SYMBOL      CHAR(10),"
                          "PRICE_IN    REAL,"
                          "TIME_OUT    INT     NOT NULL,"
                          "PRICE_OUT   REAL,"
                          "COMMISSION  REAL,"
                          "SWAP        REAL,"
                          "PROFIT      REAL);"))
        {
         Print("DB: create the TRADES table failed with code ", GetLastError());
         return(false);
        }
//--- the table has been successfully created
   return(true);
  }
//+------------------------------------------------------------------+
```

参照

DatabasePrepare、DatabaseFinalize

© 2000–2025, MetaQuotes Ltd.

DatabasePrepare

DatabaseRead()を使用して実行できるリクエストのハンドルを作成します。

```
int  DatabasePrepare(
   int      database,       // DatabaseOpenで受信したデータベースハンドル
   string   sql,            // SQL要求
            ...             // リクエストパラメータ
   );
```

パラメータ

database

[in] DatabaseOpen()で受信したデータベースハンドル。

sql

[in] ?1,?2,..... という名前の自動的に置換されたパラメータを含む可能性のあるSQL要求

...

[in] 自動的に置換された要求パラメータ。

戻り値

成功の場合は関数はSQL要求のハンドルを返します。その他の場合はINVALID_HANDLEを返します。エラーコードを受け取るには GetLastError() 関数を使用します。以下の応答が可能です。

- ERR_INVALID_PARAMETER (4003) – データベースファイルへのパスに空の文字列が含まれているか、フラグの互換性のない組み合わせが設定されています
- ERR_NOT_ENOUGH_MEMORY (4004) – 不充分なメモリ
- ERR_WRONG_STRING_PARAMETER (5040) – 要求をUTF-8文字列に変換する際のエラー
- ERR_DATABASE_INVALID_HANDLE (5121) – 無効なデータベースハンドル
- ERR_DATABASE_TOO_MANY_OBJECTS (5122) – データベースオブジェクトの最大許容数を超えています
- ERR_DATABASE_PREPARE (5125) – リクエスト生成エラー

注意事項

DatabasePrepare()関数は、データベースへの要求を実行しません。その目的は、要求パラメータを検証し、検証結果に基づいてSQL要求を実行するためのハンドルを返すことです。要求自体は、DatabaseRead()の最初の呼び出し中に設定されます。

Example:

```
//--- Structure to store the deal
struct Deal
  {
   ulong             ticket;           // DEAL_TICKET
   long              order_ticket;     // DEAL_ORDER
   long              position_ticket;  // DEAL_POSITION_ID
   datetime          time;             // DEAL_TIME
   char              type;             // DEAL_TYPE
   char              entry;            // DEAL_ENTRY
   string            symbol;           // DEAL_SYMBOL
```

© 2000–2025, MetaQuotes Ltd.

```
   double            volume;           // DEAL_VOLUME
   double            price;            // DEAL_PRICE
   double            profit;           // DEAL_PROFIT
   double            swap;             // DEAL_SWAP
   double            commission;       // DEAL_COMMISSION
   long              magic;            // DEAL_MAGIC
   char              reason;           // DEAL_REASON
  };
//--- Structure to store the trade: the order of members corresponds to the position in the terminal
struct Trade
  {
   datetime          time_in;          // entry time
   ulong             ticket;           // position ID
   char              type;             // buy or sell
   double            volume;           // volume
   string            symbol;           // symbol
   double            price_in;         // entry price
   datetime          time_out;         // exit time
   double            price_out;        // exit price
   double            commission;       // entry and exit commission
   double            swap;             // swap
   double            profit;           // profit or loss
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create the file name
   string filename=IntegerToString(AccountInfoInteger(ACCOUNT_LOGIN))
+"_trades.sqlite";
//--- open/create the database in the common terminal folder
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE |
DATABASE_OPEN_COMMON);
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
//--- create the DEALS table
   if(!CreateTableDeals(db))
     {
      DatabaseClose(db);
      return;
     }
//---  request the entire trading history
   datetime from_date=0;
   datetime to_date=TimeCurrent();
```

© 2000–2025, MetaQuotes Ltd.

```
//--- request the history of deals in the specified interval
   HistorySelect(from_date, to_date);
   int deals_total=HistoryDealsTotal();
   PrintFormat("Deals in the trading history: %d ", deals_total);
//--- add deals to the table
   if(!InsertDeals(db))
      return;
//--- show the first 10 deals
   Deal deals[], deal;
   ArrayResize(deals, 10);
   int request=DatabasePrepare(db, "SELECT * FROM DEALS");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   int i;
   for(i=0; DatabaseReadBind(request, deal); i++)
     {
      if(i>=10)
         break;
      deals[i].ticket=deal.ticket;
      deals[i].order_ticket=deal.order_ticket;
      deals[i].position_ticket=deal.position_ticket;
      deals[i].time=deal.time;
      deals[i].type=deal.type;
      deals[i].entry=deal.entry;
      deals[i].symbol=deal.symbol;
      deals[i].volume=deal.volume;
      deals[i].price=deal.price;
      deals[i].profit=deal.profit;
      deals[i].swap=deal.swap;
      deals[i].commission=deal.commission;
      deals[i].magic=deal.magic;
      deals[i].reason=deal.reason;
     }
//--- print the deals
   if(i>0)
     {
      ArrayResize(deals, i);
      PrintFormat("The first %d deals:", i);
      ArrayPrint(deals);
     }

//--- delete request after use
   DatabaseFinalize(request);
```

© 2000–2025, MetaQuotes Ltd.

```
//--- make sure that hedging system for open position management is used on the
account
   if((ENUM_ACCOUNT_MARGIN_MODE)
AccountInfoInteger(ACCOUNT_MARGIN_MODE)!=ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)
     {
      //--- deals cannot be transformed to trades using a simple method through
transactions, therefore complete operation
      DatabaseClose(db);
      return;
     }

//--- now create the TRADES table based on the DEALS table
   if(!CreateTableTrades(db))
     {
      DatabaseClose(db);
      return;
     }
//--- fill in the TRADES table using an SQL query based on DEALS table data
   ulong start=GetMicrosecondCount();
   if(DatabaseTableExists(db, "DEALS"))
      //--- populate the TRADES table
      if(!DatabaseExecute(db, "INSERT INTO
TRADES(TIME_IN,TICKET,TYPE,VOLUME,SYMBOL,PRICE_IN,TIME_OUT,PRICE_OUT,COMMISSION,SWAP
,PROFIT) "
                          "SELECT "
                          "   d1.time as time_in,"
                          "   d1.position_id as ticket,"
                          "   d1.type as type,"
                          "   d1.volume as volume,"
                          "   d1.symbol as symbol,"
                          "   d1.price as price_in,"
                          "   d2.time as time_out,"
                          "   d2.price as price_out,"
                          "   d1.commission+d2.commission as commission,"
                          "   d2.swap as swap,"
                          "   d2.profit as profit "
                          "FROM DEALS d1 "
                          "INNER JOIN DEALS d2 ON d1.position_id=d2.position_id "
                          "WHERE d1.entry=0 AND d2.entry=1     "))
        {
         Print("DB: fillng the TRADES table failed with code ", GetLastError());
         return;
        }
   ulong transaction_time=GetMicrosecondCount()-start;

//--- show the first 10 deals
   Trade trades[], trade;
   ArrayResize(trades, 10);
   request=DatabasePrepare(db, "SELECT * FROM TRADES");
   if(request==INVALID_HANDLE)
```

© 2000–2025, MetaQuotes Ltd.

```
      {
       Print("DB: ", filename, " request failed with code ", GetLastError());
       DatabaseClose(db);
       return;
      }
   for(i=0; DatabaseReadBind(request, trade); i++)
     {
      if(i>=10)
         break;
      trades[i].time_in=trade.time_in;
      trades[i].ticket=trade.ticket;
      trades[i].type=trade.type;
      trades[i].volume=trade.volume;
      trades[i].symbol=trade.symbol;
      trades[i].price_in=trade.price_in;
      trades[i].time_out=trade.time_out;
      trades[i].price_out=trade.price_out;
      trades[i].commission=trade.commission;
      trades[i].swap=trade.swap;
      trades[i].profit=trade.profit;
     }
//--- print trades
   if(i>0)
     {
      ArrayResize(trades, i);
      PrintFormat("\r\nThe first %d trades:", i);
      ArrayPrint(trades);
      PrintFormat("Filling the TRADES table took %.2f
milliseconds",double(transaction_time)/1000);
     }
//--- delete request after use
   DatabaseFinalize(request);

//--- close the database
   DatabaseClose(db);
  }
/*
Results:
   Deals in the trading history: 2741
   The first 10 deals:
       [ticket] [order_ticket] [position_ticket]              [time] [type] [entry]
[symbol] [volume]    [price]    [profit] [swap] [commission] [magic] [reason]
   [0] 34429573              0                 0 2019.09.05 22:39:59      2       0
""         0.00000   0.00000 2000.00000 0.0000      0.00000       0        0
   [1] 34432127       51447238          51447238 2019.09.06 06:00:03      0       0
"USDCAD"   0.10000   1.32320    0.00000 0.0000     -0.16000     500        3
   [2] 34432128       51447239          51447239 2019.09.06 06:00:03      1       0
"USDCHF"   0.10000   0.98697    0.00000 0.0000     -0.16000     500        3
   [3] 34432450       51447565          51447565 2019.09.06 07:00:00      0       0
"EURUSD"   0.10000   1.10348    0.00000 0.0000     -0.18000     400        3
```

© 2000–2025, MetaQuotes Ltd.

```
   [4] 34432456       51447571          51447571 2019.09.06 07:00:00       1       0
"AUDUSD"  0.10000   0.68203    0.00000 0.0000      -0.11000      400         3
   [5] 34432879       51448053          51448053 2019.09.06 08:00:00       1       0
"USDCHF"  0.10000   0.98701    0.00000 0.0000      -0.16000      600         3
   [6] 34432888       51448064          51448064 2019.09.06 08:00:00       0       0
"USDJPY"  0.10000 106.96200    0.00000 0.0000      -0.16000      600         3
   [7] 34435147       51450470          51450470 2019.09.06 10:30:00       1       0
"EURUSD"  0.10000   1.10399    0.00000 0.0000      -0.18000      100         3
   [8] 34435152       51450476          51450476 2019.09.06 10:30:00       0       0
"GBPUSD"  0.10000   1.23038    0.00000 0.0000      -0.20000      100         3
   [9] 34435154       51450479          51450479 2019.09.06 10:30:00       1       0
"EURJPY"  0.10000 118.12000    0.00000 0.0000      -0.18000      200         3


   The first 10 trades:
                 [time_in] [ticket] [type] [volume] [symbol] [price_in]
[time_out] [price_out] [commission]   [swap]  [profit]
   [0] 2019.09.06 06:00:03 51447238      0  0.10000 "USDCAD"    1.32320 2019.09.06
18:00:00     1.31761     -0.32000  0.00000 -42.43000
   [1] 2019.09.06 06:00:03 51447239      1  0.10000 "USDCHF"    0.98697 2019.09.06
18:00:00     0.98641     -0.32000  0.00000   5.68000
   [2] 2019.09.06 07:00:00 51447565      0  0.10000 "EURUSD"    1.10348 2019.09.09
03:30:00     1.10217     -0.36000 -1.31000 -13.10000
   [3] 2019.09.06 07:00:00 51447571      1  0.10000 "AUDUSD"    0.68203 2019.09.09
03:30:00     0.68419     -0.22000  0.03000 -21.60000
   [4] 2019.09.06 08:00:00 51448053      1  0.10000 "USDCHF"    0.98701 2019.09.06
18:00:01     0.98640     -0.32000  0.00000   6.18000
   [5] 2019.09.06 08:00:00 51448064      0  0.10000 "USDJPY"  106.96200 2019.09.06
18:00:01   106.77000     -0.32000  0.00000 -17.98000
   [6] 2019.09.06 10:30:00 51450470      1  0.10000 "EURUSD"    1.10399 2019.09.06
14:30:00     1.10242     -0.36000  0.00000  15.70000
   [7] 2019.09.06 10:30:00 51450476      0  0.10000 "GBPUSD"    1.23038 2019.09.06
14:30:00     1.23040     -0.40000  0.00000   0.20000
   [8] 2019.09.06 10:30:00 51450479      1  0.10000 "EURJPY"  118.12000 2019.09.06
14:30:00   117.94100     -0.36000  0.00000  16.73000
   [9] 2019.09.06 10:30:00 51450480      0  0.10000 "GBPJPY"  131.65300 2019.09.06
14:30:01   131.62500     -0.40000  0.00000  -2.62000
   Filling the TRADES table took 12.51 milliseconds
*/
//+------------------------------------------------------------------+
//| Creates the DEALS table                                          |
//+------------------------------------------------------------------+
bool CreateTableDeals(int database)
  {
//--- if the DEALS table already exists, delete it
   if(!DeleteTable(database, "DEALS"))
     {
      return(false);
     }
//--- check if the table exists
   if(!DatabaseTableExists(database, "DEALS"))
      //--- create the table
      if(!DatabaseExecute(database, "CREATE TABLE DEALS("
```

© 2000–2025, MetaQuotes Ltd.

```
                              "ID          INT KEY NOT NULL,"
                              "ORDER_ID    INT     NOT NULL,"
                              "POSITION_ID INT     NOT NULL,"
                              "TIME        INT     NOT NULL,"
                              "TYPE        INT     NOT NULL,"
                              "ENTRY       INT     NOT NULL,"
                              "SYMBOL      CHAR(10),"
                              "VOLUME      REAL,"
                              "PRICE       REAL,"
                              "PROFIT      REAL,"
                              "SWAP        REAL,"
                              "COMMISSION  REAL,"
                              "MAGIC       INT,"
                              "REASON      INT );"))
        {
         Print("DB: create the DEALS table failed with code ", GetLastError());
         return(false);
        }
//--- the table has been successfully created
   return(true);
  }
//+------------------------------------------------------------------+
//| Deletes a table with the specified name from the database        |
//+------------------------------------------------------------------+
bool DeleteTable(int database, string table_name)
  {
   if(!DatabaseExecute(database, "DROP TABLE IF EXISTS "+table_name))
     {
      Print("Failed to drop the DEALS table  with code ", GetLastError());
      return(false);
     }
//--- the table has been successfully deleted
   return(true);
  }
//+------------------------------------------------------------------+
//| Adds deals to the database table                                 |
//+------------------------------------------------------------------+
bool InsertDeals(int database)
  {
//--- Auxiliary variables
   ulong    deal_ticket;         // deal ticket
   long     order_ticket;        // the ticket of the order by which the deal was
executed
   long     position_ticket;     // ID of the position to which the deal belongs
   datetime time;                // deal execution time
   long     type ;               // deal type
   long     entry ;              // deal direction
   string   symbol;              // the symbol fro which the deal was executed
   double   volume;              // operation volume
```

© 2000–2025, MetaQuotes Ltd.

```
   double   price;               // price
   double   profit;              // financial result
   double   swap;                // swap
   double   commission;          // commission
   long     magic;               // Magic number (Expert Advisor ID)
   long     reason;              // deal execution reason or source
//--- go through all deals and add them to the database
   bool failed=false;
   int deals=HistoryDealsTotal();
// --- lock the database before executing transactions
   DatabaseTransactionBegin(database);
   for(int i=0; i<deals; i++)
     {
      deal_ticket=    HistoryDealGetTicket(i);
      order_ticket=   HistoryDealGetInteger(deal_ticket, DEAL_ORDER);
      position_ticket=HistoryDealGetInteger(deal_ticket, DEAL_POSITION_ID);
      time= (datetime)HistoryDealGetInteger(deal_ticket, DEAL_TIME);
      type=           HistoryDealGetInteger(deal_ticket, DEAL_TYPE);
      entry=          HistoryDealGetInteger(deal_ticket, DEAL_ENTRY);
      symbol=         HistoryDealGetString(deal_ticket, DEAL_SYMBOL);
      volume=         HistoryDealGetDouble(deal_ticket, DEAL_VOLUME);
      price=          HistoryDealGetDouble(deal_ticket, DEAL_PRICE);
      profit=         HistoryDealGetDouble(deal_ticket, DEAL_PROFIT);
      swap=           HistoryDealGetDouble(deal_ticket, DEAL_SWAP);
      commission=     HistoryDealGetDouble(deal_ticket, DEAL_COMMISSION);
      magic=          HistoryDealGetInteger(deal_ticket, DEAL_MAGIC);
      reason=         HistoryDealGetInteger(deal_ticket, DEAL_REASON);
      //--- add each deal to the table using the following request
      string request_text=StringFormat("INSERT INTO DEALS
(ID,ORDER_ID,POSITION_ID,TIME,TYPE,ENTRY,SYMBOL,VOLUME,PRICE,PROFIT,SWAP,COMMISSION,
MAGIC,REASON)"
                                       "VALUES (%d, %d, %d, %d, %d, %d, '%s', %G, %
G, %G, %G, %G, %d, %d)",
                                       deal_ticket, order_ticket, position_ticket,
time, type, entry, symbol, volume, price, profit, swap, commission, magic, reason);
      if(!DatabaseExecute(database, request_text))
        {
         PrintFormat("%s: failed to insert deal #%d with code %d", __FUNCTION__,
deal_ticket, GetLastError());
         PrintFormat("i=%d: deal #%d  %s", i, deal_ticket, symbol);
         failed=true;
         break;
        }
     }
//--- check for transaction execution errors
   if(failed)
     {
      //--- roll back all transactions and unlock the database
      DatabaseTransactionRollback(database);
```

© 2000–2025, MetaQuotes Ltd.

```
      PrintFormat("%s: DatabaseExecute() failed with code %d", __FUNCTION__,
GetLastError());
      return(false);
     }
//--- all transactions have been performed successfully - record changes and unlock
the database
   DatabaseTransactionCommit(database);
   return(true);
  }
//+------------------------------------------------------------------+
//| Creates the TRADES table                                         |
//+------------------------------------------------------------------+
bool CreateTableTrades(int database)
  {
//--- if the TRADES table already exists, delete it
   if(!DeleteTable(database, "TRADES"))
      return(false);
//--- check if the table exists
   if(!DatabaseTableExists(database, "TRADES"))
      //--- create the table
      if(!DatabaseExecute(database, "CREATE TABLE TRADES("
                          "TIME_IN     INT     NOT NULL,"
                          "TICKET      INT     NOT NULL,"
                          "TYPE        INT     NOT NULL,"
                          "VOLUME      REAL,"
                          "SYMBOL      CHAR(10),"
                          "PRICE_IN    REAL,"
                          "TIME_OUT    INT     NOT NULL,"
                          "PRICE_OUT   REAL,"
                          "COMMISSION  REAL,"
                          "SWAP        REAL,"
                          "PROFIT      REAL);"))
        {
         Print("DB: create the TRADES table failed with code ", GetLastError());
         return(false);
        }
//--- the table has been successfully created
   return(true);
  }
//+------------------------------------------------------------------+
```

参照

DatabaseExecute、DatabaseFinalize

© 2000–2025, MetaQuotes Ltd.

DatabaseReset

DatabasePrepare()の呼び出し後などに、リクエストをリセットします。

```
int  DatabaseReset(
   int  request      // DatabasePrepareで受信したリクエストハンドル
   );
```

パラメータ

request

[in] DatabasePrepare()で取得されたリクエストのハンドル。

戻り値

成功の場合は true、それ以外の場合は false 。エラーコードを受け取るにはGetLastError()を使用します。以下の応答が可能です。

- ERR_DATABASE_INVALID_HANDLE (5121) – 無効なデータベースハンドル、
- ERR_DATABASE_ERROR(5601)で始まるSQLiteエラーコード

注意事項

DatabaseReset()関数は、異なるパラメータ値を使用したリクエストの複数の実行を目的としています。たとえば、INSERTコマンドを使用してテーブルにデータを一括で追加する場合、各エントリに対してフィールド値のカスタムセットを形成する必要があります。

DatabasePrepare()と異なり、DatabaseReset() 呼び出しは、SQLコマンドを含む文字列を新しいリクエストにコンパイルしないため、DatabaseReset()はDatabasePrepare()よりもはるかに高速に実行されます。

DatabaseReset()はDatabaseBind()関数及び/またはDatabaseRead()の実行後にリクエストパラメータ値を変更する必要がある場合はDatabaseBindArray()とともに使用されます。つまり、リクエストパラメータの新しい値を設定する前(DatabaseBind/DatabaseBindArray呼び出しのブロックの前)に、 DatabaseReset()を呼び出してリセットする必要があります。パラメータ化されたリクエスト自体はDatabasePrepare()を使用して作成する必要があります。

DatabasePrepare()同様、DatabaseReset()はデータベース要求を行いません。DatabaseRead()またはDatabaseReadBind()を呼び出すと、直接リクエストが実行されます。

リクエストのパラメータ値は、DatabaseBind()/DatabaseBindArray()の呼び出しによって設定された場合、つまりパラメータが値を保持している場合、DatabaseReset()の呼び出しによってリセットされません。したがって、単一のパラメーターの値のみを変更できます。DatabaseReset()を呼び出した後に、すべてのリクエストパラメータを新たに設定する必要はありません。

DatabaseFinalize()を使用して削除されたリクエストのハンドルはDatabaseReset()に渡すことはできません。さもないと、エラーが生じます。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---  データベースを作成または開く
```

© 2000–2025, MetaQuotes Ltd.

```
   string filename="symbols.sqlite";
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE);
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
   else
      Print("Database: ", filename, " opened successfully");
//--- SYMBOLSテーブルが存在する場合は削除する
   if(DatabaseTableExists(db, "SYMBOLS"))
     {
      //--- delete the table
      if(!DatabaseExecute(db, "DROP TABLE SYMBOLS"))
        {
         Print("Failed to drop table SYMBOLS with code ", GetLastError());
         DatabaseClose(db);
         return;
        }
     }
//--- SYMBOLSテーブルを作成する
   if(!DatabaseExecute(db, "CREATE TABLE SYMBOLS("
                       "NAME           TEXT    NOT NULL,"
                       "DESCRIPTION    TEXT            ,"
                       "PATH           TEXT            ,"
                       "SPREAD         INT             ,"
                       "POINT          REAL    NOT NULL,"
                       "DIGITS         INT     NOT NULL,"
                       "JSON           BLOB );"))
     {
      Print("DB: ", filename, " create table failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
//--- SYMBOLSテーブルのすべてのフィールドのリストを表示する
   if(DatabasePrint(db, "PRAGMA TABLE_INFO(SYMBOLS)", 0)<0)
     {
      PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO(SYMBOLS)\") failed, error code=
%d at line %d", GetLastError(), __LINE__);
      DatabaseClose(db);
      return;
     }

//--- SYMBOLSテーブルに銘柄を追加するためにパラメータ化されたリクエストを作成する
   string sql="INSERT INTO SYMBOLS (NAME,DESCRIPTION,PATH,SPREAD,POINT,DIGITS,JSON)"
              " VALUES (?1,?2,?3,?4,?5,?6,?7);"; // request parameters
   int request=DatabasePrepare(db, sql);
   if(request==INVALID_HANDLE)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("DatabasePrepare() failed with code=%d", GetLastError());
      Print("SQL request: ", sql);
      DatabaseClose(db);
      return;
     }

//--- すべての銘柄を1つずつ、SYMBOLSテーブルに追加する
   int symbols=SymbolsTotal(false);
   bool request_error=false;
   DatabaseTransactionBegin(db);
   for(int i=0; i<symbols; i++)
     {
      //--- 銘柄を追加する前にパラメータの値を設定する
      ResetLastError();
      string symbol=SymbolName(i, false);
      if(!DatabaseBind(request, 0, symbol))
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__,
GetLastError());
         request_error=true;
         break;
        }
      //--- 以前のDatabaseBind()呼び出しが成功した場合、次のパラメータを設定する
      if(!DatabaseBind(request, 1, SymbolInfoString(symbol, SYMBOL_DESCRIPTION)))
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__,
GetLastError());
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 2, SymbolInfoString(symbol, SYMBOL_PATH)))
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__,
GetLastError());
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 3, SymbolInfoInteger(symbol, SYMBOL_SPREAD)))
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__,
GetLastError());
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 4, SymbolInfoDouble(symbol, SYMBOL_POINT)))
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__,
GetLastError());
         request_error=true;
         break;
```

© 2000-2025, MetaQuotes Ltd.

```
         }
       if(!DatabaseBind(request, 5, SymbolInfoInteger(symbol, SYMBOL_DIGITS)))
         {
          PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__,
GetLastError());
          request_error=true;
          break;
         }
       if(!DatabaseBind(request, 6, GetSymBolAsJson(symbol)))
         {
          PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__,
GetLastError());
          request_error=true;
          break;
         }

       //--- エントリを挿入するリクエストを実行し、エラーを確認する
       if(!DatabaseRead(request)&&(GetLastError()!=ERR_DATABASE_NO_MORE_DATA))
         {
          PrintFormat("DatabaseRead() failed with code=%d", GetLastError());
          DatabaseFinalize(request);
          request_error=true;
          break;
         }
       else
          PrintFormat("%d: added %s", i+1, symbol);
       //--- 次のパラメータ更新の前にリクエストをリセットする
       if(!DatabaseReset(request))
         {
          PrintFormat("DatabaseReset() failed with code=%d", GetLastError());
          DatabaseFinalize(request);
          request_error=true;
          break;
         }
      } //--- すべての銘柄の処理が完了した

//--- トランザクション状態
   if(request_error)
     {
      PrintFormat("Table SYMBOLS: failed to add %d symbols", symbols);
      DatabaseTransactionRollback(db);
      DatabaseClose(db);
      return;
     }
   else
     {
      DatabaseTransactionCommit(db);
      PrintFormat("Table SYMBOLS: added %d symbols",symbols);
     }
```

© 2000–2025, MetaQuotes Ltd.

```
//--- SYMBOLSテーブルをCSVファイルに保存する
   string csv_filename="symbols.csv";
   if(DatabaseExport(db, "SELECT * FROM SYMBOLS", csv_filename,
                     DATABASE_EXPORT_HEADER|DATABASE_EXPORT_INDEX|
DATABASE_EXPORT_QUOTED_STRINGS, ";"))
      Print("Database: table SYMBOLS saved in ", csv_filename);
   else
      Print("Database: DatabaseExport(\"SELECT * FROM SYMBOLS\") failed with code",
GetLastError());

//--- データベースファイルを閉じて、その件を通知する
   DatabaseClose(db);
   PrintFormat("Database: %s created and closed", filename);
  }
//+------------------------------------------------------------------+
//|  銘柄の仕様をJSONとして返す                                      |
//+------------------------------------------------------------------+
string GetSymBolAsJson(string symbol)
  {
//--- インデント
   string indent1=Indent(1);
   string indent2=Indent(2);
   string indent3=Indent(3);
//---
   int digits=(int)SymbolInfoInteger(symbol, SYMBOL_DIGITS);
   string json="{"+
               "\n"+indent1+"\"ConfigSymbols\":["+
               "\n"+indent2+"{"+
               "\n"+indent3+"\"Symbol\":\""+symbol+"\","+
               "\n"+indent3+"\"Path\":\""+SymbolInfoString(symbol, SYMBOL_PATH)
+"\","+
               "\n"+indent3+"\"CurrencyBase\":\""+SymbolInfoString(symbol,
SYMBOL_CURRENCY_BASE)+"\","+
               "\n"+indent3+"\"CurrencyProfit\":\""+SymbolInfoString(symbol,
SYMBOL_CURRENCY_PROFIT)+"\","+
               "\n"+indent3+"\"CurrencyMargin\":\""+SymbolInfoString(symbol,
SYMBOL_CURRENCY_MARGIN)+"\","+
               "\n"+indent3+"\"ColorBackground\":\""+ColorToString((color)
SymbolInfoInteger(symbol, SYMBOL_BACKGROUND_COLOR))+"\","+
               "\n"+indent3+"\"Digits\":
\""+IntegerToString(SymbolInfoInteger(symbol, SYMBOL_DIGITS))+"\","+
               "\n"+indent3+"\"Point\":\""+DoubleToString(SymbolInfoDouble(symbol,
SYMBOL_POINT), digits)+"\","+
               "\n"+indent3+"\"TickBookDepth\":
\""+IntegerToString(SymbolInfoInteger(symbol, SYMBOL_TICKS_BOOKDEPTH))+"\","+
               "\n"+indent3+"\"ChartMode\":
\""+IntegerToString(SymbolInfoInteger(symbol, SYMBOL_CHART_MODE))+"\","+
               "\n"+indent3+"\"TradeMode\":
\""+IntegerToString(SymbolInfoInteger(symbol, SYMBOL_TRADE_EXEMODE))+"\","+
```

© 2000-2025, MetaQuotes Ltd.

```
                "\n"+indent3+"\"TradeCalcMode\":
\""+IntegerToString(SymbolInfoInteger(symbol, SYMBOL_TRADE_CALC_MODE))+"\","+
                "\n"+indent3+"\"OrderMode\":
\""+IntegerToString(SymbolInfoInteger(symbol, SYMBOL_ORDER_MODE))+"\","+
                "\n"+indent3+"\"CalculationMode\":
\""+IntegerToString(SymbolInfoInteger(symbol, SYMBOL_TRADE_CALC_MODE))+"\","+
                "\n"+indent3+"\"ExecutionMode\":
\""+IntegerToString(SymbolInfoInteger(symbol, SYMBOL_TRADE_EXEMODE))+"\","+
                "\n"+indent3+"\"ExpirationMode\":
\""+IntegerToString(SymbolInfoInteger(symbol, SYMBOL_EXPIRATION_MODE))+"\","+
                "\n"+indent3+"\"FillFlags\":
\""+IntegerToString(SymbolInfoInteger(symbol, SYMBOL_FILLING_MODE))+"\","+
                "\n"+indent3+"\"ExpirFlags\":
\""+IntegerToString(SymbolInfoInteger(symbol, SYMBOL_EXPIRATION_MODE))+"\","+
                "\n"+indent3+"\"Spread\":
\""+IntegerToString(SymbolInfoInteger(symbol, SYMBOL_SPREAD))+"\","+
                "\n"+indent3+"\"TickValue\":\""+StringFormat("%G",
(SymbolInfoDouble(symbol, SYMBOL_TRADE_TICK_VALUE)))+"\","+
                "\n"+indent3+"\"TickSize\":\""+StringFormat("%G",
(SymbolInfoDouble(symbol, SYMBOL_TRADE_TICK_SIZE)))+"\","+
                "\n"+indent3+"\"ContractSize\":\""+StringFormat("%G",
(SymbolInfoDouble(symbol, SYMBOL_TRADE_CONTRACT_SIZE)))+"\","+
                "\n"+indent3+"\"StopsLevel\":
\""+IntegerToString(SymbolInfoInteger(symbol, SYMBOL_TRADE_STOPS_LEVEL))+"\","+
                "\n"+indent3+"\"VolumeMin\":\""+StringFormat("%G",
(SymbolInfoDouble(symbol, SYMBOL_VOLUME_MIN)))+"\","+
                "\n"+indent3+"\"VolumeMax\":\""+StringFormat("%G",
(SymbolInfoDouble(symbol, SYMBOL_VOLUME_MAX)))+"\","+
                "\n"+indent3+"\"VolumeStep\":\""+StringFormat("%G",
(SymbolInfoDouble(symbol, SYMBOL_VOLUME_STEP)))+"\","+
                "\n"+indent3+"\"VolumeLimit\":\""+StringFormat("%G",
(SymbolInfoDouble(symbol, SYMBOL_VOLUME_STEP)))+"\","+
                "\n"+indent3+"\"SwapMode\":
\""+IntegerToString(SymbolInfoInteger(symbol, SYMBOL_SWAP_MODE))+"\","+
                "\n"+indent3+"\"SwapLong\":\""+StringFormat("%G",
(SymbolInfoDouble(symbol, SYMBOL_SWAP_LONG)))+"\","+
                "\n"+indent3+"\"SwapShort\":\""+StringFormat("%G",
(SymbolInfoDouble(symbol, SYMBOL_SWAP_SHORT)))+"\","+
                "\n"+indent3+"\"Swap3Day\":
\""+IntegerToString(SymbolInfoInteger(symbol, SYMBOL_SWAP_ROLLOVER3DAYS))+"\""+
                "\n"+indent2+"}"+
                "\n"+indent1+"]"+
                "\n}";
   return(json);
  }
//+------------------------------------------------------------------+
//|  スペースで作られたインデントを形成する                                  |
//+------------------------------------------------------------------+
string Indent(const int number, const int characters=3)
  {
   int length=number*characters;
   string indent=NULL;
   StringInit(indent, length, ' ');
```

© 2000-2025, MetaQuotes Ltd.

```
   return indent;
  }
/*
 結果:
  Database: symbols.sqlite opened successfully
  #| cid name        type notnull dflt_value pk
  -+----------------------------------------
  1|   0 NAME        TEXT       1             0
  2|   1 DESCRIPTION TEXT       0             0
  3|   2 PATH        TEXT       0             0
  4|   3 SPREAD      INT        0             0
  5|   4 POINT       REAL       1             0
  6|   5 DIGITS      INT        1             0
  7|   6 JSON        BLOB       0             0
  1: added EURUSD
  2: added GBPUSD
  3: added USDCHF
  ...
  82: added USDCOP
  83: added USDARS
  84: added USDCLP
  Table SYMBOLS: added 84 symbols
  Database: table SYMBOLS saved in symbols.csv
  Database: symbols.sqlite created and closed
*/
```

参照

DatabasePrepare、DatabaseBind、DatabaseBindArray、DatabaseFinalize

© 2000–2025, MetaQuotes Ltd.

DatabaseBind

リクエストにパラメータ値を設定します。

```
bool  DatabaseBind(
   int   request,       // DatabasePrepareで作成されたリクエストのハンドル
   int   index,         // リクエストのパラメータインデックス
   T     value          // 単純型パラメータの値
   );
```

パラメータ

request

[in] DatabasePrepare()で作成されたリクエストのハンドル。

index

[in] 値を設定する必要があるリクエストのパラメータインデックス(0から始まる)。

value

[in] 設定される値。拡張タイプ: bool、char、uchar、short、ushart、int、uint、color、datetime、long、ulong、float、double、string。

戻り値

成功の場合はtrue、それ以外の場合はfalse。エラーコードを受け取るにはGetLastError()関数を使用します。以下の応答が可能です。

- ERR_INVALID_PARAMETER (4003) – サポートされていないタイプ
- ERR_DATABASE_INVALID_HANDLE (5121) – 無効なデータベースハンドル
- ERR_DATABASE_NOT_READY (5128) – 現在、関数を使用してリクエストを行うことはできません。リクエストは実行中か、すでに完了しています。DatabaseReset()が呼び出されるべきです。

注意事項

この関数は、SQL要求に「?」または「?N」パラメータ化可能値が含まれている場合に使用されます。ここで、Nはパラメータインデックス(1から始まる)を意味します。同時に、DatabaseBind()のパラメータインデックスはゼロから開始します。

例は下記の通りです。

```
INSERT INTO table VALUES (?,?,?)
SELECT * FROM table WHERE id=?
```

この関数は、パラメータ化されたリクエストがDatabasePrepare()で作成された直後、またはリクエストがDatabaseReset()を使用してリセットされた直後に呼び出すことができます。

この関数をDatabaseReset()と一緒に使用して、異なるパラメータ値で必要な回数だけリクエストを実行します。

この関数は、単純型パラメータで動作するように設計されています。パラメータを配列に対して確認する必要がある場合は、DatabaseBindArray()関数を使用します。

例：

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   MqlTick ticks[];
//--- ティックを受け取る前に開始時間を覚えておく
   uint start=GetTickCount();
//--- 1日あたりのティック履歴をリクエストする
   ulong to=TimeCurrent()*1000;
   ulong from=to-PeriodSeconds(PERIOD_D1)*1000;
   if(CopyTicksRange(_Symbol, ticks, COPY_TICKS_ALL, from, to)==-1)
     {
      PrintFormat("%s: CopyTicksRange(%s - %s) failed, error=%d",
                  _Symbol, TimeToString(datetime(from/1000)),
TimeToString(datetime(to/1000)), _LastError);
      return;
     }
   else
     {
      //--- 受信されたティックの数と受信にかかった時間
      PrintFormat("%s: CopyTicksRange received %d ticks in %d ms (from %s to %s)",
                  _Symbol, ArraySize(ticks), GetTickCount()-start,
                  TimeToString(datetime(from/1000)),
TimeToString(datetime(to/1000)));
     }

//--- データベースを保存するためのファイル名を設定する
   string filename=_Symbol+" "+TimeToString(datetime(from/1000))+" -
"+TimeToString(datetime(to/1000))+".sqlite";
   StringReplace(filename, ":", "."); // ":" 文字はファイル名に使用できない
//--- 共通の端末フォルダーでデータベースを作成または開く
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE |
DATABASE_OPEN_COMMON);
   if(db==INVALID_HANDLE)
     {
      Print("Database: ", filename, " open failed with code ", GetLastError());
      return;
     }
   else
      Print("Database: ", filename, " opened successfully");

//--- TICKSテーブルを作成する
   if(!DatabaseExecute(db, "CREATE TABLE TICKS("
                       "SYMBOL             CHAR(10),"
                       "TIME               INT NOT NULL,"
                       "BID                REAL,"
                       "ASK                REAL,"
                       "LAST               REAL,"
```

© 2000-2025, MetaQuotes Ltd.

```
                           "VOLUME              INT,"
                           "TIME_MSC            INT,"
                           "VOLUME_REAL         REAL);"))
      {
       Print("DB: ", filename, " create table TICKS failed with code ",
GetLastError());
       DatabaseClose(db);
       return;
      }
//--- TICKSテーブルのすべてのフィールドのリストを表示する
   if(DatabasePrint(db, "PRAGMA TABLE_INFO(TICKS)", 0)<0)
      {
       PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO(TICKS)\") failed, error code=%d
at line %d", GetLastError(), __LINE__);
       DatabaseClose(db);
       return;
      }
//--- TICKSテーブルにティックを追加するためにパラメータ化されたリクエストを作成する
   string sql="INSERT INTO TICKS
(SYMBOL,TIME,BID,ASK,LAST,VOLUME,TIME_MSC,VOLUME_REAL)"
              " VALUES (?1,?2,?3,?4,?5,?6,?7,?8)"; // リクエストパラメータ
   int request=DatabasePrepare(db, sql);
   if(request==INVALID_HANDLE)
      {
       PrintFormat("DatabasePrepare() failed with code=%d", GetLastError());
       Print("SQL request: ", sql);
       DatabaseClose(db);
       return;
      }
//--- 最初のリクエストパラメータの値を設定する
   DatabaseBind(request, 0, _Symbol);
//--- TICKSテーブルにティックを追加する前に開始時間を覚えておく
   start=GetTickCount();
   DatabaseTransactionBegin(db);
   int total=ArraySize(ticks);
   bool request_error=false;
   for(int i=0; i<total; i++)
      {
       //--- エントリを追加する前に残りのパラメータの値を設定する
       ResetLastError();
       if(!DatabaseBind(request, 1, ticks[i].time))
         {
          PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
          PrintFormat("Tick #%d line=%d", i+1, __LINE__);
          request_error=true;
          break;
         }
       //--- 算のDatabaseBind()呼び出しが成功した場合、次のパラメータを設定する
       if(!request_error && !DatabaseBind(request, 2, ticks[i].bid))
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
          PrintFormat("Tick #%d line=%d", i+1, __LINE__);
          request_error=true;
          break;
         }
       if(!request_error && !DatabaseBind(request, 3, ticks[i].ask))
         {
          PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
          PrintFormat("Tick #%d line=%d", i+1, __LINE__);
          request_error=true;
          break;
         }
       if(!request_error && !DatabaseBind(request, 4, ticks[i].last))
         {
          PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
          PrintFormat("Tick #%d line=%d", i+1, __LINE__);
          request_error=true;
          break;
         }
       if(!request_error && !DatabaseBind(request, 5, ticks[i].volume))
         {
          PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
          PrintFormat("Tick #%d line=%d", i+1, __LINE__);
          request_error=true;
          break;
         }
       if(!request_error && !DatabaseBind(request, 6, ticks[i].time_msc))
         {
          PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
          PrintFormat("Tick #%d line=%d", i+1, __LINE__);
          request_error=true;
          break;
         }
       if(!request_error && !DatabaseBind(request, 7, ticks[i].volume_real))
         {
          PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
          PrintFormat("Tick #%d line=%d", i+1, __LINE__);
          request_error=true;
          break;
         }

       //--- エントリを挿入するリクエストを実行し、エラーを確認する
       if(!request_error && !DatabaseRead(request) &&
(GetLastError()!=ERR_DATABASE_NO_MORE_DATA))
         {
          PrintFormat("DatabaseRead() failed with code=%d", GetLastError());
          DatabaseFinalize(request);
          request_error=true;
```

© 2000–2025, MetaQuotes Ltd.

```
         break;
        }
      //--- 次のパラメータ更新の前にリクエストをリセットする
      if(!request_error && !DatabaseReset(request))
        {
         PrintFormat("DatabaseReset() failed with code=%d", GetLastError());
         DatabaseFinalize(request);
         request_error=true;
         break;
        }
     } //--- すべてのティックの処理が完了した

//--- トランザクション状態
   if(request_error)
     {
      PrintFormat("Table TICKS: failed to add %d ticks ", ArraySize(ticks));
      DatabaseTransactionRollback(db);
      DatabaseClose(db);
      return;
     }
   else
     {
      DatabaseTransactionCommit(db);
      PrintFormat("Table TICKS: added %d ticks in %d ms",
                  ArraySize(ticks), GetTickCount()-start);
     }

//--- データベースファイルを閉じて、その件を通知する
   DatabaseClose(db);
   PrintFormat("Database: %s created and closed", filename);
  }
/*
 結果:
  EURUSD: CopyTicksRange received 268061 ticks in 47 ms (from 2020.03.18 12:40 to
2020.03.19 12:40)
  Database: EURUSD 2020.03.18 12.40 - 2020.03.19 12.40.sqlite opened successfully
  #| cid name         type     notnull dflt_value pk
  -+----------------------------------------------
  1|   0 SYMBOL       CHAR(10)        0             0
  2|   1 TIME         INT             1             0
  3|   2 BID          REAL            0             0
  4|   3 ASK          REAL            0             0
  5|   4 LAST         REAL            0             0
  6|   5 VOLUME       INT             0             0
  7|   6 TIME_MSC     INT             0             0
  8|   7 VOLUME_REAL  REAL            0             0
  Table TICKS: added 268061 ticks in 797 ms
  Database: EURUSD 2020.03.18 12.40 - 2020.03.19 12.40.sqlite created and closed
  OnCalculateCorrelation=0.87 2020.03.19 13:00:  EURUSD vs GBPUSD  PERIOD_M30
```

© 2000-2025, MetaQuotes Ltd.

```
*/
```

参照

DatabasePrepare、DatabaseReset、DatabaseRead、DatabaseBindArray

© 2000–2025, MetaQuotes Ltd.

DatabaseBindArray

パラメータ値として配列を設定します。

```
bool  DatabaseBind(
   int   request,       // DatabasePrepareで作成されたリクエストのハンドル
   int   index,         // リクエストのパラメータインデックス
   T&    array[]        // 配列としてのパラメータ値
   );
```

パラメータ

request

[in] DatabasePrepare()で作成されたリクエストのハンドル。

index

[in] 値を設定する必要があるリクエストのパラメータインデックス(0から始まる)。

array[]

[in] リクエストパラメータ値として設定される配列

戻り値

成功の場合はtrue、それ以外の場合はfalse。エラーコードを受け取るにはGetLastError()関数を使用します。以下の応答が可能です。

- ERR_INVALID_PARAMETER (4003) – サポートされていないタイプ
- ERR_ARRAY_BAD_SIZE (4011) – 配列サイズ(バイト単位)がINT_MAXを超える
- ERR_DATABASE_INVALID_HANDLE (5121) – 無効なデータベースハンドル
- ERR_DATABASE_NOT_READY (5128) – 現在、この関数を使用してリクエストを行うことはできない(リクエストが実行されているかすでに完了しているため、DatabaseResetを呼び出す必要がある)

注意事項

この関数は、SQL要求に「?」または「?N」パラメータ化可能値が含まれている場合に使用されます。ここで、Nはパラメータインデックス(1から始まる)を意味します。同時に、DatabaseBindArray()のパラメータインデックスはゼロから開始します。

例は下記の通りです。

```
INSERT INTO table VALUES (?,?,?)
```

この関数は、パラメータ化されたリクエストがDatabasePrepare()で作成された直後、またはリクエストがDatabaseReset()を使用してリセットされた直後に呼び出すことができます。

この関数をDatabaseReset()と一緒に使用して、異なるパラメータ値で必要な回数だけリクエストを実行します。

例：

```
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- DAT拡張子を持つファイルを選択するためのダイアログを開く
   string selected_files[];
   if(!FileSelectDialog("Select files to download", NULL,
                        "Data files (*.dat)|*.dat|All files (*.*)|*.*",
                        FSD_ALLOW_MULTISELECT, selected_files, "tester.dat")>0)
     {
      Print("Files not selected. Exit");
      return;
     }
//--- ファイルのサイズを取得する
   ulong filesize[];
   int filehandle[];
   int files=ArraySize(selected_files);
   ArrayResize(filesize, files);
   ZeroMemory(filesize);
   ArrayResize(filehandle, files);
   double total_size=0;
   for(int i=0; i<files; i++)
     {
      filehandle[i]=FileOpen(selected_files[i], FILE_READ|FILE_BIN);
      if(filehandle[i]!=INVALID_HANDLE)
        {
         filesize[i]=FileSize(filehandle[i]);
         //PrintFormat("%d, %s handle=%d %d bytes", i, selected_files[i],
filehandle[i], filesize[i]);
         total_size+=(double)filesize[i];
        }
     }
//--- ファイルの一般的なサイズを確認する
   if(total_size==0)
     {
      PrintFormat("Total files size is 0. Exit");
      return;
     }

//--- 共通の端末フォルダーでデータベースを作成または開きます
   string filename="dat_files.sqlite";
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE);
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
   else
      Print("Database: ", filename, " opened successfully");
//--- FI LESテーブルが存在する場合は削除する
   if(DatabaseTableExists(db, "FILES"))
     {
      //--- delete the table
```

© 2000–2025, MetaQuotes Ltd.

```
      if(!DatabaseExecute(db, "DROP TABLE FILES"))
        {
         Print("Failed to drop table FILES with code ", GetLastError());
         DatabaseClose(db);
         return;
        }
     }
//--- FILESテーブルを作成する
   if(!DatabaseExecute(db, "CREATE TABLE FILES("
                           "NAME           TEXT NOT NULL,"
                           "SIZE           INT  NOT NULL,"
                           "PERCENT_SIZE   REAL NOT NULL,"
                           "DATA           BLOB NOT NULL);"))
     {
      Print("DB: failed to create table FILES with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
//--- FILESテーブルのすべてのフィールドのリストを表示する
   if(DatabasePrint(db, "PRAGMA TABLE_INFO(FILES)", 0)<0)
     {
      PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO(FILES)\") failed, error code=%d
at line %d", GetLastError(), __LINE__);
      DatabaseClose(db);
      return;
     }

//--- FILESテーブルにファイルを追加するためにパラメータ化されたリクエストを作成する
   string sql="INSERT INTO FILES (NAME,SIZE,PERCENT_SIZE,DATA)"
              " VALUES (?1,?2,?3,?4);"; // リクエストパラメータ
   int request=DatabasePrepare(db, sql);
   if(request==INVALID_HANDLE)
     {
      PrintFormat("DatabasePrepare() failed with code=%d", GetLastError());
      Print("SQL request: ", sql);
      DatabaseClose(db);
      return;
     }

//--- すべてのファイルを1つずつ、FILESテーブルに追加する
   bool request_error=false;
   DatabaseTransactionBegin(db);
   int count=0;
   uint size;
   for(int i=0; i<files; i++)
     {
      if(filehandle[i]!=INVALID_HANDLE)
        {
         char data[];
```

© 2000-2025, MetaQuotes Ltd.

```
         size=FileReadArray(filehandle[i], data);
         if(size==0)
           {
            PrintFormat("FileReadArray(%s) failed with code %d", selected_files[i],
GetLastError());
            continue;
           }

         count++;
         //--- ファイルをテーブルに追加する前にパラメータの値を設定する
         if(!DatabaseBind(request, 0, selected_files[i]))
           {
            PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__,
GetLastError());
            request_error=true;
            break;
           }
         if(!DatabaseBind(request, 1, size))
           {
            PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__,
GetLastError());
            request_error=true;
            break;
           }
         if(!DatabaseBind(request, 2, double(size)*100./total_size))
           {
            PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__,
GetLastError());
            request_error=true;
            break;
           }
         if(!DatabaseBindArray(request, 3, data))
           {
            PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__,
GetLastError());
            request_error=true;
            break;
           }
         //--- エントリを挿入するリクエストを実行し、エラーを確認する
         if(!DatabaseRead(request)&&(GetLastError()!=ERR_DATABASE_NO_MORE_DATA))
           {
            PrintFormat("DatabaseRead() failed with code=%d", GetLastError());
            DatabaseFinalize(request);
            request_error=true;
            break;
           }
         else
            PrintFormat("%d. %s: %d bytes", count, selected_files[i],size);
         //--- 次のパラメータ更新の前にリクエストをリセットする
         if(!DatabaseReset(request))
```

© 2000–2025, MetaQuotes Ltd.

```
            {
             PrintFormat("DatabaseReset() failed with code=%d", GetLastError());
             DatabaseFinalize(request);
             request_error=true;
             break;
            }
         }
      }
//--- トランザクション状態
   if(request_error)
      {
       PrintFormat("Table FILES: failed to add %d files", count);
       DatabaseTransactionRollback(db);
       DatabaseClose(db);
       return;
      }
   else
      {
       DatabaseTransactionCommit(db);
       PrintFormat("Table FILES: added %d files", count);
      }

//--- データベースファイルを閉じて、その件を通知する
   DatabaseClose(db);
   PrintFormat("Database: %s created and closed", filename);
  }
```

参照

DatabasePrepare、DatabaseReset、DatabaseRead、DatabaseBind

© 2000-2025, MetaQuotes Ltd.

DatabaseRead

リクエストの結果として次のエントリに移動します。

```
bool  DatabaseRead(
   int  request      // DatabasePrepareで受信したリクエストハンドル
   );
```

パラメータ

request

[in] DatabasePrepare()で受信したリクエストハンドル。

戻り値

成功の場合は true、それ以外の場合は false 。エラーコードを受け取るには GetLastError() 関数を使用します。以下の応答が可能です。

- ERR_INVALID_PARAMETER (4003) – テーブル名が指定されていない(空の文字列または NULL)
- ERR_WRONG_STRING_PARAMETER (5040) – 要求をUTF-8文字列に変換する際のエラー
- ERR_DATABASE_INTERNAL (5120) – 内部データベースエラー
- ERR_DATABASE_INVALID_HANDLE (5121) – 無効なデータベースハンドル
- ERR_DATABASE_EXECUTE (5124) – リクエスト実行エラー
- ERR_DATABASE_NO_MORE_DATA (5126) – テーブルが存在しない(エラーではなく、正常に完了する)

参照

DatabasePrepare, DatabaseReadBind

© 2000-2025, MetaQuotes Ltd.

DatabaseReadBind

次のレコードに移動し、そこから構造体にデータを読み込みます。

```
bool  DatabaseReadBind(
   int     request,            // DatabasePrepareで作成されたリクエストのハンドル
   void&   struct_object       // レコードを読み取るための構造体への参照
   );
```

パラメータ

request

[in] DatabasePrepare()で作成されたリクエストのハンドル。

struct_object

[out] 現在のレコードのデータが読み取られる構造体への参照。構造体は、メンバーとして数値型および/または文字列（配列は許可されません）のみを持つ必要があり、子孫にはできません。

戻り値

成功の場合は true、それ以外の場合は false 。エラーコードを受け取るには GetLastError() 関数を使用します。以下が可能です。

- ERR_INVALID_PARAMETER (4003) – テーブル名が指定されていない（空の文字列または NULL）
- ERR_WRONG_STRING_PARAMETER (5040) – 要求をUTF-8文字列に変換する際のエラー
- ERR_DATABASE_INTERNAL (5120) – 内部データベースエラー
- ERR_DATABASE_INVALID_HANDLE (5121) – 無効なデータベースハンドル
- ERR_DATABASE_EXECUTE (5124) – リクエスト実行エラー
- ERR_DATABASE_NO_MORE_DATA (5126) – テーブルが存在しない（エラーではなく、正常に完了する）

注意事項

*record*構造体のフィールド数はDatabaseColumnsCount()を越えられません。*struct_object*構造体のフィールド数がstruct_object構造体のフィールド数未満の場合、読み込みは部分的になります。残りのデータは、対応するDatabaseColumnText()、DatabaseColumnInteger()およびその他の関数を使用して明示的に取得できます。

例：

```
struct Person
  {
   int               id;
   string            name;
   int               age;
   string            address;
   double            salary;
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
```

© 2000–2025, MetaQuotes Ltd.

```
void OnStart()
  {
   int db;
   string filename="company.sqlite";
//--- open
   db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE |
DATABASE_OPEN_COMMON);
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
//--- if the table COMPANY exists then drop the table
   if(DatabaseTableExists(db, "COMPANY"))
     {
      //--- delete the table
      if(!DatabaseExecute(db, "DROP TABLE COMPANY"))
        {
         Print("Failed to drop table COMPANY with code ", GetLastError());
         DatabaseClose(db);
         return;
        }
     }
//--- create table
   if(!DatabaseExecute(db, "CREATE TABLE COMPANY("
                       "ID INT PRIMARY KEY     NOT NULL,"
                       "NAME           TEXT    NOT NULL,"
                       "AGE            INT     NOT NULL,"
                       "ADDRESS        CHAR(50),"
                       "SALARY         REAL );"))
     {
      Print("DB: ", filename, " create table failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }

//--- insert data
   if(!DatabaseExecute(db, "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES
(1, 'Paul', 32, 'California', 25000.00 ); "
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (2,
'Allen', 25, 'Texas', 15000.00 ); "
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (3,
'Teddy', 23, 'Norway', 20000.00 );"
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (4,
'Mark', 25, 'Rich-Mond ', 65000.00 );"))
     {
      Print("DB: ", filename, " insert failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
```

© 2000–2025, MetaQuotes Ltd.

```
//--- prepare the request
   int request=DatabasePrepare(db, "SELECT * FROM COMPANY WHERE SALARY>15000");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
//--- print records
   Person person;
   Print("Persons with salary > 15000:");
   for(int i=0; DatabaseReadBind(request, person); i++)
      Print(i, ":  ", person.id, " ", person.name, " ", person.age, " ",
person.address, " ", person.salary);
//--- delete request after use
   DatabaseFinalize(request);

   Print("Some statistics:");
//--- prepare new request about total salary
   request=DatabasePrepare(db, "SELECT SUM(SALARY) FROM COMPANY");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   while(DatabaseRead(request))
     {
      double total_salary;
      DatabaseColumnDouble(request, 0, total_salary);
      Print("Total salary=", total_salary);
     }
//--- delete request after use
   DatabaseFinalize(request);

//--- prepare new request about average salary
   request=DatabasePrepare(db, "SELECT AVG(SALARY) FROM COMPANY");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      ResetLastError();
      DatabaseClose(db);
      return;
     }
   while(DatabaseRead(request))
     {
      double aver_salary;
      DatabaseColumnDouble(request, 0, aver_salary);
```

© 2000–2025, MetaQuotes Ltd.

```
      Print("Average salary=", aver_salary);
     }
//--- delete request after use
   DatabaseFinalize(request);

//--- close database
   DatabaseClose(db);
  }
//+------------------------------------------------------------------+
/*
Output:
Persons with salary > 15000:
0:  1 Paul 32 California 25000.0
1:  3 Teddy 23 Norway 20000.0
2:  4 Mark 25 Rich-Mond  65000.0
Some statistics:
Total salary=125000.0
Average salary=31250.0
*/
```

参照

DatabasePrepare、DatabaseRead

© 2000–2025, MetaQuotes Ltd.

DatabaseFinalize

DatabasePrepare()で作成されたリクエストを削除します。

```
void  DatabaseFinalize(
   int  request      // DatabasePrepareで受信したリクエストハンドル
   );
```

パラメータ

database

[in] DatabasePrepare()で受信したリクエストハンドル。

戻り値

なし

注意事項

ハンドルが無効な場合、関数はERR_DATABASE_INVALID_HANDLEエラーを設定します。エラーはGetLastError()を使用してチェックできます。

参照

DatabasePrepare、DatabaseExecute

© 2000–2025, MetaQuotes Ltd.

DatabaseTransactionBegin

トランザクションの実行を開始します。

```
bool  DatabaseTransactionBegin(
   int  database      // DatabaseOpenで受信したデータベースハンドル
   );
```

パラメータ

database

[in] DatabaseOpen()で受信したデータベースハンドル。

成功の場合は true、それ以外の場合は false 。エラーコードを受け取るには GetLastError() 関数を使用します。以下の応答が可能です。

- ERR_INTERNAL_ERROR (4001) – 重大なランタイムエラー
- ERR_INVALID_PARAMETER (4003) – sqlパラメータに空の文字列が含まれています
- ERR_NOT_ENOUGH_MEMORY (4004) – 不充分なメモリ
- ERR_WRONG_STRING_PARAMETER (5040) – 要求をUTF-8文字列に変換する際のエラー
- ERR_DATABASE_INTERNAL (5120) – 内部データベースエラー
- ERR_DATABASE_INVALID_HANDLE (5121) – 無効なデータベースハンドル
- ERR_DATABASE_EXECUTE (5124) – リクエスト実行エラー

注意事項

DatabaseTransactionBegin()関数は、トランザクションの実行前に呼び出す必要があります。トランザクションは、DatabaseTransactionBegin()の呼び出しで始まり、DatabaseTransactionCommit()の呼び出しで終わる必要があります。

Example:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create the file name
   string filename=AccountInfoString(ACCOUNT_SERVER)
+"_"+IntegerToString(AccountInfoInteger(ACCOUNT_LOGIN))+".sqlite";
//--- open/create the database in the common terminal folder
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE |
DATABASE_OPEN_COMMON);
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
//--- if the DEALS table already exists, delete it
   if(!DeleteTable(db, "DEALS"))
     {
      DatabaseClose(db);
      return;
```

© 2000–2025, MetaQuotes Ltd.

```
     }
//--- create the DEALS table
   if(!CreateTableDeals(db))
     {
      DatabaseClose(db);
      return;
     }
//---  request the entire trading history
   datetime from_date=0;
   datetime to_date=TimeCurrent();
//--- request the history of deals in the specified interval
   HistorySelect(from_date, to_date);
   int deals_total=HistoryDealsTotal();
   PrintFormat("Deals in the trading history: %d ", deals_total);

//--- measure the transaction execution speed using
DatabaseTransactionBegin/DatabaseTransactionCommit
   ulong start=GetMicrosecondCount();
   bool fast_transactions=true;
   InsertDeals(db, fast_transactions);
   double fast_transactions_time=double(GetMicrosecondCount()-start)/1000;
   PrintFormat("Transations WITH
DatabaseTransactionBegin/DatabaseTransactionCommit: time=%.1f milliseconds",
fast_transactions_time);

//--- delete the DEALS table, and then create it again
   if(!DeleteTable(db, "DEALS"))
     {
      DatabaseClose(db);
      return;
     }
//--- create a new DEALS table
   if(!CreateTableDeals(db))
     {
      DatabaseClose(db);
      return;
     }

//--- test again, this time without using
DatabaseTransactionBegin/DatabaseTransactionCommit
   fast_transactions=false;
   start=GetMicrosecondCount();
   InsertDeals(db, fast_transactions);
   double slow_transactions_time=double(GetMicrosecondCount()-start)/1000;
   PrintFormat("Transations WITHOUT
DatabaseTransactionBegin/DatabaseTransactionCommit: time=%.1f milliseconds",
slow_transactions_time);
//--- report gain in time
   PrintFormat("Use of DatabaseTransactionBegin/DatabaseTransactionCommit provided
acceleration by %.1f times", double(slow_transactions_time)/fast_transactions_time);
```

© 2000–2025, MetaQuotes Ltd.

```
//--- close the database
   DatabaseClose(db);
  }
/*
Results:
   Deals in the trading history: 2737
   Transations WITH    DatabaseTransactionBegin/DatabaseTransactionCommit: time=48.5
 milliseconds
   Transations WITHOUT DatabaseTransactionBegin/DatabaseTransactionCommit:
time=25818.9 milliseconds
   Use of DatabaseTransactionBegin/DatabaseTransactionCommit provided acceleration
by 532.8 times
*/
//+------------------------------------------------------------------+
//| Deletes a table with the specified name from the database        |
//+------------------------------------------------------------------+
bool DeleteTable(int database, string table_name)
  {
   if(!DatabaseExecute(database, "DROP TABLE IF EXISTS "+table_name))
     {
      Print("Failed to drop table DEALS with code ", GetLastError());
      return(false);
     }
//--- the table has been successfully deleted
   return(true);
  }
//+------------------------------------------------------------------+
//| Creates the DEALS table                                          |
//+------------------------------------------------------------------+
bool CreateTableDeals(int database)
  {
//--- check if the table exists
   if(!DatabaseTableExists(database, "DEALS"))
      //--- create the table
      if(!DatabaseExecute(database, "CREATE TABLE DEALS("
                          "ID          INT KEY NOT NULL,"
                          "ORDER_ID    INT     NOT NULL,"
                          "POSITION_ID INT     NOT NULL,"
                          "TIME        INT     NOT NULL,"
                          "TYPE        INT     NOT NULL,"
                          "ENTRY       INT     NOT NULL,"
                          "SYMBOL      CHAR(10),"
                          "VOLUME      REAL,"
                          "PRICE       REAL,"
                          "PROFIT      REAL,"
                          "SWAP        REAL,"
                          "COMMISSION  REAL,"
                          "MAGIC       INT,"
                          "REASON      INT );"))
```

© 2000–2025, MetaQuotes Ltd.

```
         {
          Print("DB: create table failed with code ", GetLastError());
          return(false);
         }
//--- the table has been successfully created
   return(true);
  }
//+------------------------------------------------------------------+
//| Adds deals to the database table                                 |
//+------------------------------------------------------------------+
bool InsertDeals(int database, bool begintransaction=true)
  {
//--- Auxiliary variables
   ulong    deal_ticket;          // deal ticket
   long     order_ticket;         // the ticket of the order by which the deal was
executed
   long     position_ticket;      // ID of the position to which the deal belongs
   datetime time;                 // deal execution time
   long     type ;                // deal type
   long     entry ;               // deal direction
   string   symbol;               // the symbol fro which the deal was executed
   double   volume;               // operation volume
   double   price;                // price
   double   profit;               // financial result
   double   swap;                 // swap
   double   commission;           // commission
   long     magic;                // Magic number
   long     reason;               // deal execution reason or source
//--- go through all deals and add to the database
   bool failed=false;
   int deals=HistoryDealsTotal();
//--- if fast transaction performance method is used
   if(begintransaction)
     {
      // --- lock the database before executing transactions
      DatabaseTransactionBegin(database);
     }
   for(int i=0; i<deals; i++)
     {
      deal_ticket=    HistoryDealGetTicket(i);
      order_ticket=   HistoryDealGetInteger(deal_ticket, DEAL_ORDER);
      position_ticket=HistoryDealGetInteger(deal_ticket, DEAL_POSITION_ID);
      time= (datetime)HistoryDealGetInteger(deal_ticket, DEAL_TIME);
      type=           HistoryDealGetInteger(deal_ticket, DEAL_TYPE);
      entry=          HistoryDealGetInteger(deal_ticket, DEAL_ENTRY);
      symbol=         HistoryDealGetString(deal_ticket, DEAL_SYMBOL);
      volume=         HistoryDealGetDouble(deal_ticket, DEAL_VOLUME);
      price=          HistoryDealGetDouble(deal_ticket, DEAL_PRICE);
      profit=         HistoryDealGetDouble(deal_ticket, DEAL_PROFIT);
```

© 2000–2025, MetaQuotes Ltd.

```
      swap=           HistoryDealGetDouble(deal_ticket, DEAL_SWAP);
      commission=     HistoryDealGetDouble(deal_ticket, DEAL_COMMISSION);
      magic=          HistoryDealGetInteger(deal_ticket, DEAL_MAGIC);
      reason=         HistoryDealGetInteger(deal_ticket, DEAL_REASON);
      //--- add each deal using the following request
      string request_text=StringFormat("INSERT INTO DEALS
(ID,ORDER_ID,POSITION_ID,TIME,TYPE,ENTRY,SYMBOL,VOLUME,PRICE,PROFIT,SWAP,COMMISSION,
MAGIC,REASON)"
                                       "VALUES (%d, %d, %d, %d, %d, %d, '%s', %G, %
G, %G, %G, %G, %d, %d)",
                                       deal_ticket, order_ticket, position_ticket,
time, type, entry, symbol, volume, price, profit, swap, commission, magic, reason);
      if(!DatabaseExecute(database, request_text))
        {
         PrintFormat("%s: failed to insert deal #%dwith code %d", __FUNCTION__,
deal_ticket, GetLastError());
         PrintFormat("i=%d: deal #%d  %s", i, deal_ticket, symbol);
         failed=true;
         break;
        }
     }
//--- check for transaction execution errors
   if(failed)
     {
      //--- if fast transaction performance method is used
      if(begintransaction)
        {
         //--- roll back all transactions and unlock the database
         DatabaseTransactionRollback(database);
        }
      Print("%s: DatabaseExecute() failed with code ", __FUNCTION__,
GetLastError());
      return(false);
     }
//--- if fast transaction performance method is used
   if(begintransaction)
     {
      //--- all transactions have been performed successfully - record changes and
unlock the database
      DatabaseTransactionCommit(database);
     }
//--- successful completion
   return(true);
  }
//+------------------------------------------------------------------+
```

参照

DatabaseExecute、DatabasePrepare、DatabaseTransactionCommit、DatabaseTransactionRollback

© 2000–2025, MetaQuotes Ltd.

DatabaseTransactionCommit

トランザクションの実行を完了します。

```
bool  DatabaseTransactionCommit(
   int  database      // DatabaseOpenで受信したデータベースハンドル
   );
```

パラメータ

database

[in] DatabaseOpen()で受信したデータベースハンドル。

成功の場合は true、それ以外の場合は false 。エラーコードを受け取るには GetLastError() 関数を使用します。以下の応答が可能です。

- ERR_INTERNAL_ERROR (4001) – 重大なランタイムエラー
- ERR_INVALID_PARAMETER (4003) – sqlパラメータに空の文字列が含まれています
- ERR_NOT_ENOUGH_MEMORY (4004) – 不充分なメモリ
- ERR_WRONG_STRING_PARAMETER (5040) – 要求をUTF-8文字列に変換する際のエラー
- ERR_DATABASE_INTERNAL (5120) – 内部データベースエラー
- ERR_DATABASE_INVALID_HANDLE (5121) – 無効なデータベースハンドル
- ERR_DATABASE_EXECUTE (5124) – リクエスト実行エラー

注意事項

DatabaseTransactionCommit()関数はDatabaseBeginTransaction()関数を呼び出した後に実行されたすべてのトランザクションを完了します。すべてのトランザクションは、正常に完了するためにDatabaseTransactionBegin()の呼び出しで始まり、DatabaseTransactionCommit()の呼び出しで終わる必要があります。

参照

DatabaseExecute、DatabasePrepare、DatabaseTransactionBegin、DatabaseTransactionRollback

© 2000–2025, MetaQuotes Ltd.

DatabaseTransactionRollback

トランザクションをロールバックします。

```
bool  DatabaseTransactionRollback(
   int  database      // DatabaseOpenで受信したデータベースハンドル
   );
```

パラメータ

database

[in] DatabaseOpen()で受信したデータベースハンドル。

成功の場合は true、それ以外の場合は false 。エラーコードを受け取るには GetLastError() 関数を使用します。以下の応答が可能です。

- ERR_INTERNAL_ERROR (4001) – 重大なランタイムエラー
- ERR_INVALID_PARAMETER (4003) – sqlパラメータに空の文字列が含まれています
- ERR_NOT_ENOUGH_MEMORY (4004) – 不充分なメモリ
- ERR_WRONG_STRING_PARAMETER (5040) – 要求をUTF-8文字列に変換する際のエラー
- ERR_DATABASE_INTERNAL (5120) – 内部データベースエラー
- ERR_DATABASE_INVALID_HANDLE (5121) – 無効なデータベースハンドル
- ERR_DATABASE_EXECUTE (5124) – リクエスト実行エラー

注意事項

DatabaseTransactionRollback()の呼び出しはDatabaseTransactionBegin()関数を呼び出した後に実行されたすべてのトランザクションをキャンセルします。DatabaseTransactionRollback()関数は、トランザクションの実行中にエラーが発生した場合にデータベースの変更をロールバックするために必要です。

参照

DatabaseExecute、DatabasePrepare、DatabaseTransactionBegin、DatabaseTransactionCommit

© 2000–2025, MetaQuotes Ltd.

DatabaseColumnsCount

リクエスト内のフィールドの数を取得します。

```
int  DatabaseColumnsCount(
   int  request      // DatabasePrepareで受信したリクエストハンドル
   );
```

パラメータ

request

[in] DatabasePrepare()で受信したリクエストハンドル。

戻り値

フィールド数（エラーの場合は-1）。エラーコードを受け取るには GetLastError() 関数を使用します。以下の値が可能です。

- ERR_DATABASE_INVALID_HANDLE (5121) – 無効なリクエストハンドル

注意事項

DatabasePrepare()で作成されたリクエストのフィールド数を取得するのにDatabaseRead()関数を呼び出す必要はありません。残りのDatabaseColumnXXX()関数については、DatabaseRead()を事前に呼び出す必要があります。

参照

DatabasePrepare、DatabaseFinalize、DatabaseClose

© 2000–2025, MetaQuotes Ltd.

DatabaseColumnName

インデックスでフィールド名を取得します。

```
bool  DatabaseColumnName(
   int       request,      // DatabasePrepareで受信したリクエストハンドル
   int       column,       // リクエストのフィールドインデックス
   string&   name          // フィールド名を受け取るための変数への参照
   );
```

パラメータ

request

[in] DatabasePrepare()で受信したリクエストハンドル。

column

[in] リクエストのフィールドインデックス。フィールド番号はゼロから始まり、DatabaseColumnsCount() - 1を超えることはできません。

name

[out] フィールド名を書き込むための変数。

戻り値

成功の場合は true、それ以外の場合は false 。エラーコードを受け取るには GetLastError() 関数を使用します。以下の応答が可能です。

- ERR_DATABASE_INVALID_HANDLE (5121) - 無効なリクエストハンドル
- ERR_DATABASE_NO_MORE_DATA (5126) - 「column」インデックスがDatabaseColumnsCount() -1を超えています。

注意事項

値は、少なくとも1つのDatabaseRead()呼び出しが「要求」に対して事前に行われている場合にのみ取得できます。

参照

DatabasePrepare、DatabaseColumnsCount、DatabaseColumnType

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnType

インデックスでフィールドタイプを取得します。

```
ENUM_DATABASE_FIELD_TYPE  DatabaseColumnType(
   int  request,     // DatabasePrepareで受信したリクエストハンドル
   int  column       // リクエストのフィールドインデックス
   );
```

パラメータ

request

[in] DatabasePrepare()で受信したリクエストハンドル。

column

[in] リクエストのフィールドインデックス。フィールド番号はゼロから始まり、DatabaseColumnsCount() – 1を超えることはできません。

戻り値

ENUM_DATABASE_FIELD_TYPE列挙からのフィールドタイプを返します。エラーコードを受け取るにはGetLastError() 関数を使用します。以下の応答が可能です。

- ERR_DATABASE_INVALID_HANDLE (5121) – 無効なリクエストハンドル
- ERR_DATABASE_NO_MORE_DATA (5126) –「column」インデックスがDatabaseColumnsCount() –1を超えています。

注意事項

値は、少なくとも1つのDatabaseRead()呼び出しが「要求」に対して事前に行われている場合にのみ取得できます。

ENUM_DATABASE_FIELD_TYPE

ID	説明
DATABASE_FIELD_TYPE_INVALID	型の取得エラー。エラーコードは、int GetLastError()を使用して取得できます。
DATABASE_FIELD_TYPE_INTEGER	Integer型
DATABASE_FIELD_TYPE_FLOAT	Real型
DATABASE_FIELD_TYPE_TEXT	String型
DATABASE_FIELD_TYPE_BLOB	Binary型
DATABASE_FIELD_TYPE_NULL	特殊のNULL型

参照

DatabasePrepare、DatabaseColumnsCount、DatabaseColumnName

© 2000–2025, MetaQuotes Ltd.

DatabaseColumnSize

バイト単位のフィールドサイズを取得します。

```
int  DatabaseColumnSize(
   int  request,     // DatabasePrepareで受信したリクエストハンドル
   int  column       // リクエストのフィールドインデックス
   );
```

パラメータ

request

[in] DatabasePrepare()で受信したリクエストハンドル。

column

[in] リクエストのフィールドインデックス。フィールド番号はゼロから始まり、DatabaseColumnsCount() - 1を超えることはできません。

戻り値

成功した場合うあバイト単位のフィールドサイズが返され、そうでない場合は-1が返されます。エラーコードを受け取るには GetLastError() 関数を使用します。以下の応答が可能です。

- ERR_DATABASE_INVALID_HANDLE (5121) - 無効なリクエストハンドル
- ERR_DATABASE_NO_MORE_DATA (5126) - 「column」インデックスがDatabaseColumnsCount() -1を超えています。

注意事項

値は、少なくとも1つのDatabaseRead()呼び出しが「要求」に対して事前に行われている場合にのみ取得できます。

参照

DatabasePrepare、DatabaseColumnBlob、DatabaseColumnsCount、DatabaseColumnName、DatabaseColumnType

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnText

現在のレコードからフィールド値を文字列として取得します。

```
bool  DatabaseColumnText(
   int       request,      // DatabasePrepareで受信したリクエストハンドル
   int       column,       // リクエストのフィールドインデックス
   string&   value         // 値を受け取るための変数への参照
   );
```

パラメータ

request

[in] DatabasePrepare()で受信したリクエストハンドル。

column

[in] リクエストのフィールドインデックス。フィールド番号はゼロから始まり、DatabaseColumnsCount() - 1を超えることはできません。

value

[out] フィールド値を書き込むための変数への参照

戻り値

成功の場合は true、それ以外の場合は false 。エラーコードを受け取るには GetLastError() 関数を使用します。以下の応答が可能です。

- ERR_DATABASE_INVALID_HANDLE (5121) - 無効なリクエストハンドル
- ERR_DATABASE_NO_MORE_DATA (5126) - 「column」インデックスがDatabaseColumnsCount() -1を超えています。

注意事項

値は、少なくとも1つのDatabaseRead()呼び出しが「要求」に対して事前に行われている場合にのみ取得できます。

次のレコードから値を読み取るには、DatabaseRead()を事前に呼び出します。

参照

DatabasePrepare、DatabaseColumnsCount、DatabaseColumnType、DatabaseColumnName

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnInteger

現在のレコードからint型の値を取得します。

```
bool  DatabaseColumnInteger(
   int    request,      // DatabasePrepareで受信したリクエストハンドル
   int    column,       // リクエストのフィールドインデックス
   int&   value         // 値を受け取るための変数への参照
   );
```

パラメータ

request

[in] DatabasePrepare()で受信したリクエストハンドル。

column

[in] リクエストのフィールドインデックス。フィールド番号はゼロから始まり、DatabaseColumnsCount() - 1を超えることはできません。

value

[out] フィールド値を書き込むための変数への参照

戻り値

成功の場合は true、それ以外の場合は false 。エラーコードを受け取るには GetLastError() 関数を使用します。以下の応答が可能です。

- ERR_DATABASE_INVALID_HANDLE (5121) - 無効なリクエストハンドル
- ERR_DATABASE_NO_MORE_DATA (5126) - 「column」インデックスがDatabaseColumnsCount() -1を超えています。

注意事項

値は、少なくとも1つのDatabaseRead()呼び出しが「要求」に対して事前に行われている場合にのみ取得できます。

次のレコードから値を読み取るには、DatabaseRead()を事前に呼び出します。

参照

DatabasePrepare、DatabaseColumnsCount、DatabaseColumnType、DatabaseColumnName

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnLong

現在のレコードからlong型の値を取得します。

```
bool  DatabaseColumnLong(
   int     request,      // DatabasePrepareで受信したリクエストハンドル
   int     column,       // リクエストのフィールドインデックス
   long&   value         // 値を受け取るための変数への参照
   );
```

パラメータ

request

[in] DatabasePrepare()で受信したリクエストハンドル。

column

[in] リクエストのフィールドインデックス。フィールド番号はゼロから始まり、DatabaseColumnsCount() - 1を超えることはできません。

value

[out] フィールド値を書き込むための変数への参照

戻り値

成功の場合は true、それ以外の場合は false 。エラーコードを受け取るには GetLastError() 関数を使用します。以下の応答が可能です。

- ERR_DATABASE_INVALID_HANDLE (5121) - 無効なリクエストハンドル
- ERR_DATABASE_NO_MORE_DATA (5126) - 「column」インデックスがDatabaseColumnsCount() -1を超えています。

注意事項

値は、少なくとも1つのDatabaseRead()呼び出しが「要求」に対して事前に行われている場合にのみ取得できます。

次のレコードから値を読み取るには、DatabaseRead()を事前に呼び出します。

参照

DatabasePrepare、DatabaseColumnsCount、DatabaseColumnType、DatabaseColumnName

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnDouble

現在のレコードからdouble型の値を取得します。

```
bool  DatabaseColumnDouble(
   int       request,      // DatabasePrepareで受信したリクエストハンドル
   int       column,       // リクエストのフィールドインデックス
   double&   value         // 値を受け取るための変数への参照
   );
```

パラメータ

request

[in] DatabasePrepare()で受信したリクエストハンドル。

column

[in] リクエストのフィールドインデックス。フィールド番号はゼロから始まり、DatabaseColumnsCount() - 1を超えることはできません。

value

[out] フィールド値を書き込むための変数への参照

戻り値

成功の場合は true、それ以外の場合は false 。エラーコードを受け取るには GetLastError() 関数を使用します。以下の応答が可能です。

- ERR_DATABASE_INVALID_HANDLE (5121) - 無効なリクエストハンドル
- ERR_DATABASE_NO_MORE_DATA (5126) - 「column」インデックスがDatabaseColumnsCount() -1を超えています。

注意事項

値は、少なくとも1つのDatabaseRead()呼び出しが「要求」に対して事前に行われている場合にのみ取得できます。

次のレコードから値を読み取るには、DatabaseRead()を事前に呼び出します。

参照

DatabasePrepare、DatabaseColumnsCount、DatabaseColumnType、DatabaseColumnName

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnBlob

現在のレコードからフィールド値を配列として取得します

```
bool  DatabaseColumnBlob(
   int     request,      // DatabasePrepareで受信したリクエストハンドル
   int     column,       // リクエストのフィールドインデックス
   void&   data[]        // 値を受け取るための変数への参照
   );
```

パラメータ

request

[in] DatabasePrepare()で受信したリクエストハンドル。

column

[in] リクエストのフィールドインデックス。フィールド番号はゼロから始まり、DatabaseColumnsCount() - 1を超えることはできません。

data[]

[out] フィールド値を書き込むための配列への参照。

戻り値

成功の場合は true、それ以外の場合は false 。エラーコードを受け取るには GetLastError() 関数を使用します。以下の応答が可能です。

- ERR_DATABASE_INVALID_HANDLE (5121) - 無効なリクエストハンドル
- ERR_DATABASE_NO_MORE_DATA (5126) - 「column」インデックスがDatabaseColumnsCount() -1を超えています。

注意事項

値は、少なくとも1つのDatabaseRead()呼び出しが「要求」に対して事前に行われている場合にのみ取得できます。

次のレコードから値を読み取るには、DatabaseRead()を事前に呼び出します。

参照

DatabasePrepare、DatabaseColumnSize、DatabaseColumnsCount、DatabaseColumnType、DatabaseColumnName

© 2000-2025, MetaQuotes Ltd.

DirectXの操作

DirectX 11関数とシェーダーは、価格チャートで直接3D視覚化できるように設計されています。

3Dグラフィックスの作成には、必要な画像サイズを持ったグラフィックスコンテキスト(DXContextCreate)が必要です。また、頂点バッファとインデックスバッファを準備し(DXBufferCreate)、頂点シェーダーとピクセルシェーダーを作成する必要があります(DXShaderCreate)。これはグラフィックをカラーで表示するのに十分です。

次のレベルのグラフィックには、追加のレンダリングパラメータをシェーダーに渡すための入力(DXInputSet)が必要です。これにより、カメラと3Dオブジェクトの位置の設定、光源の説明、マウスとキーボードの制御の実装が可能になります。

したがって、組み込みのMQL5関数を使用すると、サードパーティのツールを必要とせずに、MetaTrader 5でアニメーション化された3Dチャートを直接作成できます。関数の動作には、ビデオカードがDX 11およびShader Model 5.0をサポートする必要があります。

ライブラリの使用をはじめるにあたって、METATRADER5でDIRECTXを使用して3Dグラフィックスを作成する方法稿をお読みください。

関数	アクション
DXContextCreate	指定されたサイズのフレームをレンダリングするためのグラフィックコンテキストを作成します
DXContextSetSize	DXContextCreate()で作成されたグラフィックコンテキストのフレームサイズを変更します
DXContextSetSize	DXContextCreate()で作成されたグラフィックコンテキストのフレームサイズを取得します
DXContextClearColors	レンダリングバッファのすべてのピクセルに指定された色を設定します
DXContextClearDepth	深度バッファをクリアします
DXContextGetColors	グラフィックコンテキストから指定されたサイズとオフセットの画像を取得します
DXContextGetDepth	レンダリングされたフレームの深度バッファを取得します
DXBufferCreate	データ配列に基づいて、指定された型のバッファを作成します
DXTextureCreate	渡された画像から切り取られた指定サイズの長方形から2Dテクスチャを作成します
DXInputCreate	シェーダー入力を作成します
DXInputSet	シェーダー入力を設定します
DXShaderCreate	指定された種類のシェーダーを作成します
DXShaderSetLayout	頂点シェーダーの頂点レイアウトを設定します
DXShaderInputsSet	シェーダー入力を設定します
DXShaderTexturesSet	シェーダーテクスチャを設定します

© 2000-2025, MetaQuotes Ltd.

関数	アクション
DXDraw	DXBufferSet()で設定された頂点バッファの頂点をレンダリングします
DXDrawIndexed	DXBufferSet()のインデックスバッファによって記述されたグラフィックプリミティブをレンダリングします
DXPrimiveTopologySet	DXDrawIndexed()を使用してレンダリングするプリミティブのタイプを設定します
DXBufferSet	現在のレンダリングにバッファを設定します
DXShaderSet	レンダリング用のシェーダーを設定します
DXHandleType	ハンドルの種類を返します
DXRelease	ハンドルを解放します

© 2000–2025, MetaQuotes Ltd.

DXContextCreate

指定されたサイズのフレームをレンダリングするためのグラフィックコンテキストを作成します。

```
int  DXContextCreate(
   uint  width,      // ピクセル単位での深度
   uint  height      // ピクセル単位での高さ
   );
```

パラメータ

width

[in] ピクセル単位でのフレーム幅

height

[in] ピクセル単位でのフレームの高さ

戻り値

作成されたコンテキストのハンドルまたはエラーの場合はINVALID_HANDLE。エラーコードを受信するには、GetLastError()関数が呼ばれるべきです。

注意事項

All graphical objects created using the DXBufferCreate、DXInputCreate、 DXShaderCreate、DXTextureCreate関数を使用して作成されたすべてのグラフィックスオブジェクトは作成されたグラフィックスコンテキスト内のみで使用できます。

フレームのサイズは後でDXContextSetSize()で変更できます。

使用されなくなった作成済みのハンドルは、DXRelease()関数によって明示的に解放する必要があります。

© 2000-2025, MetaQuotes Ltd.

DXContextSetSize

DXContextCreate()で作成されたグラフィックコンテキストのフレームサイズを変更します。

```
bool  DXContextSetSize(
   int     context,       // グラフィックコンテキストハンドル
   uint&   width,         // ピクセル単位での幅
   uint&   height         // ピクセル単位での高さ
   );
```

パラメータ

context

[in] DXContextCreate()で作成されたグラフィックコンテキストのハンドル

width

[in] ピクセル単位でのフレーム幅

height

[in] ピクセル単位でのフレームの高さ

戻り値

実行成功の場合はtrue、それ以外の場合はfalseを返します。エラーコードを受け取るにはGetLastError()関数が呼ばれるべきです。

注意事項

グラフィックコンテキストのフレームサイズは、フレームレンダリング間でのみ変更する必要があります。

© 2000-2025, MetaQuotes Ltd.

DXContextGetSize

DXContextCreate()で作成されたグラフィックコンテキストのフレームサイズを取得します。

```
bool  DXContextGetSize(
   int     context,       // グラフィックコンテキストハンドル
   uint&   width,         // ピクセル単位での幅
   uint&   height         // ピクセル単位での高さ
   );
```

パラメータ

context

[in] DXContextCreate()で作成されたグラフィックコンテキストのハンドル

width

[out] ピクセル単位でのフレーム幅

height

[out] ピクセル単位でのフレームの高さ

戻り値

実行成功の場合はtrue、それ以外の場合はfalseを返します。エラーコードを受け取るにはGetLastError()関数が呼ばれるべきです。

© 2000-2025, MetaQuotes Ltd.

DXContextClearColors

レンダリングバッファのすべてのピクセルに指定された色を設定します。

```
bool  DXContextClearColors(
   int              context,       // グラフィックスコンテキストハンドル
   const DXVector&  color          // 色
   );
```

パラメータ

context

[in] DXContextCreate()で作成されたグラフィックコンテキストのハンドル

color

[in] レンダリング色

戻り値

実行成功の場合はtrue、それ以外の場合はfalseを返します。エラーコードを受け取るにはGetLastError()関数が呼ばれるべきです。

注意事項

DXContextClearColors()関数は、次のフレームをレンダリングする前にカラーバッファをクリアするために使用できます。

© 2000-2025, MetaQuotes Ltd.

DXContextClearDepth

深度バッファをクリアします。

```
bool  DXContextClearDepth(
   int  context      // グラフィックスコンテキストハンドル
   );
```

パラメータ

context

[in] DXContextCreate()で作成されたグラフィックコンテキストのハンドル

戻り値

実行成功の場合はtrue、それ以外の場合はfalseを返します。エラーコードを受け取るにはGetLastError()関数が呼ばれるべきです。

注意事項

DXContextClearDepth()関数は、次のフレームをレンダリングする前に深度バッファーをクリアするために使用できます。

© 2000-2025, MetaQuotes Ltd.

DXContextGetColors

グラフィックコンテキストから指定されたサイズとオフセットの画像を取得します。

```
bool  DXContextGetColors(
   int    context,                      // グラフィックコンテキストハンドル
   uint&  image[],                      // 画像ピクセル配列
   int    image_width=WHOLE_ARRAY,      // ピクセル単位の画像の幅
   int    image_height=WHOLE_ARRAY,     // ピクセル単位の画像の高さ
   int    image_offset_x=0,             // Xオフセット
   int    image_offset_y=0              // Yオフセット
   );
```

パラメータ

context

[in] DXContextCreate()で作成されたグラフィックコンテキストのハンドル

image

[out] *image_width***image_height*ピクセルの配列（ARGB形式）。

image_width=WHOLE_ARRAY

[in] ピクセル単位での画像幅。

image_height=WHOLE_ARRAY

[in] ピクセル単位での画像の高さ。

image_offset_x=0

[in] Xオフセット。

image_offset_y=0

[in] Yオフセット。

戻り値

実行成功の場合はtrue、それ以外の場合はfalseを返します。エラーコードを受け取るにはGetLastError()関数が呼ばれるべきです。

© 2000–2025, MetaQuotes Ltd.

DXContextGetDepth

レンダリングされたフレームの深度バッファを取得します。

```
bool  DXContextGetDepth(
   int      context,       // グラフィックコンテキストハンドル
   float&   image[]        // 深度値配列
   );
```

パラメータ

context

[in] DXContextCreate()で作成されたグラフィックコンテキストのハンドル

image

[out] レンダリングされたフレーム深度バッファ値の配列

戻り値

実行成功の場合はtrue、それ以外の場合はfalseを返します。エラーコードを受け取るにはGetLastError()関数が呼ばれるべきです。

注意事項

返されるバッファには、相対的な単位(0.0から1.0)でDXContextGetColors()で取得できるレンダリングフレームの各ピクセルの深度が含まれます。

© 2000-2025, MetaQuotes Ltd.

DXBufferCreate

データ配列に基づいて、指定された型のバッファを作成します。

```
int  DXBufferCreate(
   int                  context,              // グラフィックスコンテキストハンドル
   ENUM_DX_BUFFER_TYPE  buffer_type,          // 作成されたバッファの型
   const void&          data[],               // バッファデータ
   uint                 start=0,              // 最初のインデックス
   uint                 count=WHOLE_ARRAY     // 要素数
   );
```

パラメータ

context

[in] DXContextCreate()で作成されたグラフィックコンテキストのハンドル

buffer_type

[in] ENUM_DX_BUFFER_TYPE列挙体からのバッファ型

data[]

[in] バッファ作成データ

start=0

[in] 配列の最初の要素のインデックス。配列の値はバッファの作成に使用されます。デフォルトでは、データは配列の先頭から取得されます。

count=WHOLE_ARRAY

[in] 値の数。デフォルトでは配列全体が使用されます(count=WHOLE_ARRAY)。

戻り値

作成されたバッファのハンドルまたはエラーの場合はINVALID_HANDLE。エラーコードを受信するには、GetLastError()関数が呼ばれるべきです。

注意事項

インデックスバッファについては、*data[]*配列は「uint」型であり、頂点バッファは、頂点を記述する構造の配列を受け取ります。

使用されなくなった作成済みのハンドルは、DXRelease()関数によって明示的に解放する必要があります。

ENUM_DX_BUFFER_TYPE

ID	値	説明
DX_BUFFER_VERTEX	1	頂点バッファ
DX_BUFFER_INDEX	2	インデックスバッファ

© 2000-2025, MetaQuotes Ltd.

DXTextureCreate

渡された画像から切り取られた指定サイズの長方形から2Dテクスチャを作成します。

```
int  DXTextureCreate(
   int              context,          // グラフィックスコンテキストハンドル
   ENUM_DX_FORMAT   format,           // ピクセル色の形式
   uint             width,            // ソース画像幅
   uint             height,           // ソース画像の高さ
   const void&      data[],           // ソース画像ピクセルの配列
   uint             data_x,           // テクスチャの作成に使用される長方形のX座標
   uint             data_y,           // テクスチャの作成に使用される長方形のY座標
   uint             data_width,       // テクスチャを作成するための長方形の幅
   uint             data_height       // テクスチャを作成するための長方形の高さ
   );
```

パラメータ

context

[in] DXContextCreate()で作成されたグラフィックコンテキストのハンドル

format

[in] ENUM_DX_FORMAT列挙値から設定されるピクセル色の形式

width

[in] テクスチャのベースとなる画像の幅

height

[in] テクスチャのベースとなる画像の高さ

data

[in] テクスチャのベースとなる画像のピクセル配列

data_x

[in] テクスチャの作成に使用される長方形のX座標

data_y

[in] テクスチャの作成に使用される長方形のY座標

data_width

[in] テクスチャの作成に使用される長方形の幅

data_height

[in] テクスチャの作成に使用される長方形の高さ

戻り値

テクスチャのハンドルまたはエラーの場合はINVALID_HANDLE。エラーコードを受信するには、GetLastError()関数が呼ばれるべきです。

注意事項

© 2000–2025, MetaQuotes Ltd.

使用されなくなった作成済みのハンドルは、DXRelease()関数によって明示的に解放する必要があります。

ENUM_DX_FORMAT

ID	値	DXGI_FORMATでの一致
DX_FORMAT_UNKNOWN	0	DXGI_FORMAT_UNKNOWN
DX_FORMAT_R32G32B32A32_TYPELESS	1	DXGI_FORMAT_R32G32B32A32_TYPELESS
DX_FORMAT_R32G32B32A32_FLOAT	2	DXGI_FORMAT_R32G32B32A32_FLOAT
DX_FORMAT_R32G32B32A32_UINT	3	DXGI_FORMAT_R32G32B32A32_UINT
DX_FORMAT_R32G32B32A32_SINT	4	DXGI_FORMAT_R32G32B32A32_SINT
DX_FORMAT_R32G32B32_TYPELESS	5	DXGI_FORMAT_R32G32B32_TYPELESS
DX_FORMAT_R32G32B32_FLOAT	6	DXGI_FORMAT_R32G32B32_FLOAT
DX_FORMAT_R32G32B32_UINT	7	DXGI_FORMAT_R32G32B32_UINT
DX_FORMAT_R32G32B32_SINT	8	DXGI_FORMAT_R32G32B32_SINT
DX_FORMAT_R16G16B16A16_TYPELESS	9	DXGI_FORMAT_R16G16B16A16_TYPELESS
DX_FORMAT_R16G16B16A16_FLOAT	10	DXGI_FORMAT_R16G16B16A16_FLOAT
DX_FORMAT_R16G16B16A16_UNORM	11	DXGI_FORMAT_R16G16B16A16_UNORM
DX_FORMAT_R16G16B16A16_UINT	12	DXGI_FORMAT_R16G16B16A16_UINT
DX_FORMAT_R16G16B16A16_SNORM	13	DXGI_FORMAT_R16G16B16A16_SNORM
DX_FORMAT_R16G16B16A16_SINT	14	DXGI_FORMAT_R16G16B16A16_SINT
DX_FORMAT_R32G32_TYPELESS	15	DXGI_FORMAT_R32G32_TYPELESS
DX_FORMAT_R32G32_FLOAT	16	DXGI_FORMAT_R32G32_FLOAT
DX_FORMAT_R32G32_UINT	17	DXGI_FORMAT_R32G32_UINT
DX_FORMAT_R32G32_SINT	18	DXGI_FORMAT_R32G32_SINT
DX_FORMAT_R32G8X24_TYPELESS	19	DXGI_FORMAT_R32G8X24_TYPELESS
DX_FORMAT_D32_FLOAT_S8X24_UINT	20	DXGI_FORMAT_D32_FLOAT_S8X24_UINT
DX_FORMAT_R32_FLOAT_X8X24_TYPELESS	21	DXGI_FORMAT_R32_FLOAT_X8X24_TYPELESS
DX_FORMAT_X32_TYPELESS_G8X24_UINT	22	DXGI_FORMAT_X32_TYPELESS_G8X24_UINT

© 2000–2025, MetaQuotes Ltd.

ID	値	DXGI_FORMATでの一致
DX_FORMAT_R10G10B10A2_TYPELESS	23	DXGI_FORMAT_R10G10B10A2_TYPELESS
DX_FORMAT_R10G10B10A2_UNORM	24	DXGI_FORMAT_R10G10B10A2_UNORM
DX_FORMAT_R10G10B10A2_UINT	25	DXGI_FORMAT_R10G10B10A2_UINT
DX_FORMAT_R11G11B10_FLOAT	26	DXGI_FORMAT_R11G11B10_FLOAT
DX_FORMAT_R8G8B8A8_TYPELESS	27	DXGI_FORMAT_R8G8B8A8_TYPELESS
DX_FORMAT_R8G8B8A8_UNORM	28	DXGI_FORMAT_R8G8B8A8_UNORM
DX_FORMAT_R8G8B8A8_UNORM_SRGB	29	DXGI_FORMAT_R8G8B8A8_UNORM_SRGB
DX_FORMAT_R8G8B8A8_UINT	30	DXGI_FORMAT_R8G8B8A8_UINT
DX_FORMAT_R8G8B8A8_SNORM	31	DXGI_FORMAT_R8G8B8A8_SNORM
DX_FORMAT_R8G8B8A8_SINT	32	DXGI_FORMAT_R8G8B8A8_SINT
DX_FORMAT_R16G16_TYPELESS	33	DXGI_FORMAT_R16G16_TYPELESS
DX_FORMAT_R16G16_FLOAT	34	DXGI_FORMAT_R16G16_FLOAT
DX_FORMAT_R16G16_UNORM	35	DXGI_FORMAT_R16G16_UNORM
DX_FORMAT_R16G16_UINT	36	DXGI_FORMAT_R16G16_UINT
DX_FORMAT_R16G16_SNORM	37	DXGI_FORMAT_R16G16_SNORM
DX_FORMAT_R16G16_SINT	38	DXGI_FORMAT_R16G16_SINT
DX_FORMAT_R32_TYPELESS	39	DXGI_FORMAT_R32_TYPELESS
DX_FORMAT_D32_FLOAT	40	DXGI_FORMAT_D32_FLOAT
DX_FORMAT_R32_FLOAT	41	DXGI_FORMAT_R32_FLOAT
DX_FORMAT_R32_UINT	42	DXGI_FORMAT_R32_UINT
DX_FORMAT_R32_SINT	43	DXGI_FORMAT_R32_SINT
DX_FORMAT_R24G8_TYPELESS	44	DXGI_FORMAT_R24G8_TYPELESS
DX_FORMAT_D24_UNORM_S8_UINT	45	DXGI_FORMAT_D24_UNORM_S8_UINT
DX_FORMAT_R24_UNORM_X8_TYPELESS	46	DXGI_FORMAT_R24_UNORM_X8_TYPELESS
DX_FORMAT_X24_TYPELESS_G8_UINT	47	DXGI_FORMAT_X24_TYPELESS_G8_UINT
DX_FORMAT_R8G8_TYPELESS	48	DXGI_FORMAT_R8G8_TYPELESS
DX_FORMAT_R8G8_UNORM	49	DXGI_FORMAT_R8G8_UNORM
DX_FORMAT_R8G8_UINT	50	DXGI_FORMAT_R8G8_UINT

© 2000–2025, MetaQuotes Ltd.

ID	値	DXGI_FORMATでの一致
DX_FORMAT_R8G8_SNORM	51	DXGI_FORMAT_R8G8_SNORM
DX_FORMAT_R8G8_SINT	52	DXGI_FORMAT_R8G8_SINT
DX_FORMAT_R16_TYPELESS	53	DXGI_FORMAT_R16_TYPELESS
DX_FORMAT_R16_FLOAT	54	DXGI_FORMAT_R16_FLOAT
DX_FORMAT_D16_UNORM	55	DXGI_FORMAT_D16_UNORM
DX_FORMAT_R16_UNORM	56	DXGI_FORMAT_R16_UNORM
DX_FORMAT_R16_UINT	57	DXGI_FORMAT_R16_UINT
DX_FORMAT_R16_SNORM	58	DXGI_FORMAT_R16_SNORM
DX_FORMAT_R16_SINT	59	DXGI_FORMAT_R16_SINT
DX_FORMAT_R8_TYPELESS	60	DXGI_FORMAT_R8_TYPELESS
DX_FORMAT_R8_UNORM	61	DXGI_FORMAT_R8_UNORM
DX_FORMAT_R8_UINT	62	DXGI_FORMAT_R8_UINT
DX_FORMAT_R8_SNORM	63	DXGI_FORMAT_R8_SNORM
DX_FORMAT_R8_SINT	64	DXGI_FORMAT_R8_SINT
DX_FORMAT_A8_UNORM	65	DXGI_FORMAT_A8_UNORM
DX_FORMAT_R1_UNORM	66	DXGI_FORMAT_R1_UNORM
DX_FORMAT_R9G9B9E5_SHAREDEXP	67	DXGI_FORMAT_R9G9B9E5_SHAREDEXP
DX_FORMAT_R8G8_B8G8_UNORM	68	DXGI_FORMAT_R8G8_B8G8_UNORM
DX_FORMAT_G8R8_G8B8_UNORM	69	DXGI_FORMAT_G8R8_G8B8_UNORM
DX_FORMAT_BC1_TYPELESS	70	DXGI_FORMAT_BC1_TYPELESS
DX_FORMAT_BC1_UNORM	71	DXGI_FORMAT_BC1_UNORM
DX_FORMAT_BC1_UNORM_SRGB	72	DXGI_FORMAT_BC1_UNORM_SRGB
DX_FORMAT_BC2_TYPELESS	73	DXGI_FORMAT_BC2_TYPELESS
DX_FORMAT_BC2_UNORM	74	DXGI_FORMAT_BC2_UNORM
DX_FORMAT_BC2_UNORM_SRGB	75	DXGI_FORMAT_BC2_UNORM_SRGB
DX_FORMAT_BC3_TYPELESS	76	DXGI_FORMAT_BC3_TYPELESS
DX_FORMAT_BC3_UNORM	77	DXGI_FORMAT_BC3_UNORM
DX_FORMAT_BC3_UNORM_SRGB	78	DXGI_FORMAT_BC3_UNORM_SRGB
DX_FORMAT_BC4_TYPELESS	79	DXGI_FORMAT_BC4_TYPELESS
DX_FORMAT_BC4_UNORM	80	DXGI_FORMAT_BC4_UNORM
DX_FORMAT_BC4_SNORM	81	DXGI_FORMAT_BC4_SNORM

© 2000–2025, MetaQuotes Ltd.

ID	値	DXGI_FORMATでの一致
DX_FORMAT_BC5_TYPELESS	82	DXGI_FORMAT_BC5_TYPELESS
DX_FORMAT_BC5_UNORM	83	DXGI_FORMAT_BC5_UNORM
DX_FORMAT_BC5_SNORM	84	DXGI_FORMAT_BC5_SNORM
DX_FORMAT_B5G6R5_UNORM	85	DXGI_FORMAT_B5G6R5_UNORM
DX_FORMAT_B5G5R5A1_UNORM	86	DXGI_FORMAT_B5G5R5A1_UNORM
DX_FORMAT_B8G8R8A8_UNORM	87	DXGI_FORMAT_B8G8R8A8_UNORM
DX_FORMAT_B8G8R8X8_UNORM	88	DXGI_FORMAT_B8G8R8X8_UNORM
DX_FORMAT_R10G10B10_XR_BIAS_A2_UNORM	89	DXGI_FORMAT_R10G10B10_XR_BIAS_A2_UNORM
DX_FORMAT_B8G8R8A8_TYPELESS	90	DXGI_FORMAT_B8G8R8A8_TYPELESS
DX_FORMAT_B8G8R8A8_UNORM_SRGB	91	DXGI_FORMAT_B8G8R8A8_UNORM_SRGB
DX_FORMAT_B8G8R8X8_TYPELESS	92	DXGI_FORMAT_B8G8R8X8_TYPELESS
DX_FORMAT_B8G8R8X8_UNORM_SRGB	93	DXGI_FORMAT_B8G8R8X8_UNORM_SRGB
DX_FORMAT_BC6H_TYPELESS	94	DXGI_FORMAT_BC6H_TYPELESS
DX_FORMAT_BC6H_UF16	95	DXGI_FORMAT_BC6H_UF16
DX_FORMAT_BC6H_SF16	96	DXGI_FORMAT_BC6H_SF16
DX_FORMAT_BC7_TYPELESS	97	DXGI_FORMAT_BC7_TYPELESS
DX_FORMAT_BC7_UNORM	98	DXGI_FORMAT_BC7_UNORM
DX_FORMAT_BC7_UNORM_SRGB	99	DXGI_FORMAT_BC7_UNORM_SRGB
DX_FORMAT_AYUV	100	DXGI_FORMAT_AYUV
DX_FORMAT_Y410	101	DXGI_FORMAT_Y410
DX_FORMAT_Y416	102	DXGI_FORMAT_Y416
DX_FORMAT_NV12	103	DXGI_FORMAT_NV12
DX_FORMAT_P010	104	DXGI_FORMAT_P010
DX_FORMAT_P016	105	DXGI_FORMAT_P016
DX_FORMAT_420_OPAQUE	106	DXGI_FORMAT_420_OPAQUE
DX_FORMAT_YUY2	107	DXGI_FORMAT_YUY2
DX_FORMAT_Y210	108	DXGI_FORMAT_Y210
DX_FORMAT_Y216	109	DXGI_FORMAT_Y216
DX_FORMAT_NV11	110	DXGI_FORMAT_NV11

© 2000–2025, MetaQuotes Ltd.

ID	値	DXGI_FORMATでの一致
DX_FORMAT_AI44	111	DXGI_FORMAT_AI44
DX_FORMAT_IA44	112	DXGI_FORMAT_IA44
DX_FORMAT_P8	113	DXGI_FORMAT_P8
DX_FORMAT_A8P8	114	DXGI_FORMAT_A8P8
DX_FORMAT_B4G4R4A4_UNORM	115	DXGI_FORMAT_B4G4R4A4_UNORM
DX_FORMAT_P208	130	DXGI_FORMAT_P208
DX_FORMAT_V208	131	DXGI_FORMAT_V208
DX_FORMAT_V408	132	DXGI_FORMAT_V408
DX_FORMAT_FORCE_UINT	0xffffffff	DXGI_FORMAT_FORCE_UINT

© 2000–2025, MetaQuotes Ltd.

DXInputCreate

シェーダー入力を作成します。

```
int  DXInputCreate(
   int   context,         // グラフィックスコンテキストハンドル
   uint  input_size       // バイト単位の入力サイズ
   );
```

パラメータ

context

[in] DXContextCreate()で作成されたグラフィックコンテキストのハンドル

input_size

[in] バイト単位でのパラメータ構造のサイズ

戻り値

シェーダー入力のハンドルまたはエラーの場合はINVALID_HANDLE。エラーコードを受信するには、GetLastError()関数が呼ばれるべきです。

使用されなくなった作成済みのハンドルは、DXRelease()関数によって明示的に解放する必要があります。

© 2000–2025, MetaQuotes Ltd.

DXInputSet

シェーダー入力を設定します。

```
bool  DXInputSet(
   int           input,       // グラフィックスコンテキストハンドル
   const void&   data         // 設定するデータ
   );
```

パラメータ

input

[in] DXInputCreate()で取得したシェーダーの入力ハンドル

data

[in] シェーダー入力を設定するためのデータ

戻り値

実行成功の場合は true、それ以外の場合は falseを返します。エラーコードを受け取るには GetLastError() 関数が呼ばれるべきです。

© 2000–2025, MetaQuotes Ltd.

DXShaderCreate

指定された種類のシェーダーを作成します。

```
int  DXShaderCreate(
   int                  context,          // グラフィックコンテキストハンドル
   ENUM_DX_SHADER_TYPE  shader_type,      // シェーダーの種類
   const string         source,           // シェーダーのソースコード
   const string         entry_point,      // エントリポイント
   string&              compile_error     // コンパイラメッセージを受け取る文字列
   );
```

パラメータ

context

[in] DXContextCreate()で作成されたグラフィックコンテキストのハンドル

shader_type

[out] ENUM_DX_SHADER_TYPE列挙体からの値。

source

[in] HLSL 5でのシェーダーのソースコード

entry_point

[in] エントリポイント – ソースコードでの関数名

compile_error

[in] コンパイルエラーを受け取る文字列

戻り値

シェーダーのハンドルまたはエラーの場合はINVALID_HANDLE。エラーコードを受信するには、GetLastError()関数が呼ばれるべきです。

注意事項

使用されなくなった作成済みのハンドルは、DXRelease()関数によって明示的に解放する必要があります。

ENUM_DX_SHADER_TYPE

ID	値	説明
DX_SHADER_VERTEX	0	頂点シェーダー
DX_SHADER_GEOMETRY	1	ジオメトリシェーダー
DX_SHADER_PIXEL	2	ピクセルシェーダー

© 2000–2025, MetaQuotes Ltd.

DXShaderSetLayout

頂点シェーダーの頂点レイアウトを設定します。

```
bool  DXShaderSetLayout(
   int                     shader,        // シェーダーハンドル
   const DXVertexLayout&   layout[]       //
   );
```

パラメータ

shader

[in] DXShaderCreate()で作成されたシェーダーのハンドル

layout[]

[in] 頂点フィールドの説明の配列です。説明はDXVertexLayout構造体で設定されています。

```
struct DXVertexLayout
  {
   string          semantic_name;        // シェーダー入力署名のこの要素に関連付けられたHLSLセマンティクス
   uint            semantic_index;       // 要素のセマンティックインデックス。セマンティックインデックスは、整数のインデックス番号でセマンティックを変更します。セマンティックインデックスは、同じセマンティックを持つ要素が複数ある場合にのみ必要です
   ENUM_DX_FORMAT format;                // 要素データのデータ型
  };
```

戻り値

実行成功の場合はtrue、それ以外の場合はfalseを返します。エラーコードを受け取るにはGetLastError()関数が呼ばれるべきです。

注意事項

レイアウトは、指定された頂点バッファー内の頂点の種類と一致する必要があります。また、頂点シェーダーコードのエントリポイントで使用される頂点入力の種類と一致する必要があります。

シェーダーの頂点バッファーはDXBufferSet()で設定されます。

DXVertexLayout構造体はD3D11_INPUT_ELEMENT_DESC MSDN構造体のバージョンです。

© 2000-2025, MetaQuotes Ltd.

DXShaderInputsSet

シェーダー入力を設定します。

```
bool  DXShaderInputsSet(
   int          shader,        // シェーダーハンドル
   const int&   inputs[]       // 入力ハンドルの配列
   );
```

パラメータ

shader

[in] DXShaderCreate()で作成されたシェーダーのハンドル

inputs[]

[in] DXInputCreate()で作成された入力ハンドルの配列

戻り値

実行成功の場合は true、それ以外の場合は falseを返します。エラーコードを受け取るには GetLastError() 関数が呼ばれるべきです。

注意事項

入力パラメータのサイズは、シェーダーコードで宣言されたcbufferオブジェクトの数と等しくなければなりません。

© 2000-2025, MetaQuotes Ltd.

DXShaderTexturesSet

シェーダーテクスチャを設定します。

```
bool  DXShaderTexturesSet(
   int           shader,          // シェーダーハンドル
   const  int&   textures[]       // 構造体ハンドルの配列
   );
```

パラメータ

shader

[in] DXShaderCreate()で作成されたシェーダーのハンドル

textures[]

[in] DXTextureCreate()を使用して作成されたテクスチャハンドルの配列

戻り値

実行成功の場合は true、それ以外の場合は falseを返します。エラーコードを受け取るには GetLastError() 関数が呼ばれるべきです。

注意事項

テクスチャ配列のサイズは、シェーダーコードで宣言されたTexture2Dオブジェクトの数と等しくする必要があります。

© 2000–2025, MetaQuotes Ltd.

DXDraw

DXBufferSet()DXBufferSet()で設定された頂点バッファの頂点をレンダリングします。

```
bool  DXDraw(
   int   context,              // グラフィックスコンテキストハンドル
   uint  start=0,              // 最初の頂点のインデックス
   uint  count=WHOLE_ARRAY     // 頂点数
   );
```

パラメータ

context

[in] DXContextCreate()で作成されたグラフィックコンテキストのハンドル

start=0

[in] レンダリング用の最初の頂点のインデックス

count=WHOLE_ARRAY

[in] レンダリングする頂点の数

戻り値

実行成功の場合は true、それ以外の場合は falseを返します。エラーコードを受け取るには GetLastError() 関数が呼ばれるべきです。

注意事項

シェーダーは、頂点のレンダリングに DXShaderSet() を使用して事前に設定する必要があります。

© 2000–2025, MetaQuotes Ltd.

DXDrawIndexed

DXBufferSet()のインデックスバッファによって記述されたグラフィックプリミティブをレンダリングします。

```
bool  DXDrawIndexed(
   int   context,                // グラフィックスコンテキストハンドル
   uint  start=0,                // 最初のプリミティブインデックス
   uint  count=WHOLE_ARRAY       // プリミティブ数
   );
```

パラメータ

context

[in] DXContextCreate()で作成されたグラフィックコンテキストのハンドル

start=0

[in] レンダリング用の最初のプリミティブインデックス

count=WHOLE_ARRAY

[in] レンダリングするプリミティブの数

戻り値

実行成功の場合はtrue、それ以外の場合はfalseを返します。エラーコードを受け取るにはGetLastError()関数が呼ばれるべきです。

注意事項

インデックスバッファによって記述されるプリミティブのタイプはDXPrimiveTopologySet()を使用して設定されます。

DXBufferSet()の頂点バッファは、プリミティブをレンダリングするために事前に設定する必要があります。

また、シェーダーはDXShaderSet()を使用して事前に設定する必要があります。

© 2000–2025, MetaQuotes Ltd.

DXPrimiveTopologySet

DXDrawIndexed()を使用してレンダリングするプリミティブのタイプを設定します。

```
bool  DXPrimiveTopologySet(
   int                          context,                // グラフィックスコンテキストハンドル
   ENUM_DX_PRIMITIVE_TOPOLOGY   primitive_topology      // プリミティブの種類
   );
```

パラメータ

context

[in] DXContextCreate()で作成されたグラフィックコンテキストのハンドル

primitive_topology

[in] ENUM_DX_PRIMITIVE_TOPOLOGY列挙体からの値

戻り値

実行成功の場合はtrue、それ以外の場合はfalseを返します。エラーコードを受け取るにはGetLastError()関数が呼ばれるべきです。

ENUM_DX_PRIMITIVE_TOPOLOGY

ID	値	D3D11_PRIMITIVE_TOPOLOGYでの一致
DX_PRIMITIVE_TOPOLOGY_POINTLIST	1	D3D11_PRIMITIVE_TOPOLOGY_POINTLIST
DX_PRIMITIVE_TOPOLOGY_LINELIST	2	D3D11_PRIMITIVE_TOPOLOGY_LINELIST
DX_PRIMITIVE_TOPOLOGY_LINESTRIP	3	D3D11_PRIMITIVE_TOPOLOGY_LINESTRIP
DX_PRIMITIVE_TOPOLOGY_TRIANGLELIST	4	D3D11_PRIMITIVE_TOPOLOGY_TRIANGLELIST
DX_PRIMITIVE_TOPOLOGY_TRIANGLESTRIP	5	D3D11_PRIMITIVE_TOPOLOGY_TRIANGLESTRIP
DX_PRIMITIVE_TOPOLOGY_LINELIST_ADJ	6	D3D11_PRIMITIVE_TOPOLOGY_LINELIST_ADJ
DX_PRIMITIVE_TOPOLOGY_LINESTRIP_ADJ	7	D3D11_PRIMITIVE_TOPOLOGY_LINESTRIP_ADJ
DX_PRIMITIVE_TOPOLOGY_TRIANGLELIST_ADJ	8	D3D11_PRIMITIVE_TOPOLOGY_TRIANGLELIST_ADJ
DX_PRIMITIVE_TOPOLOGY_TRIANGLESTRIP_ADJ	9	D3D11_PRIMITIVE_TOPOLOGY_TRIANGLESTRIP_ADJ

© 2000–2025, MetaQuotes Ltd.

DXBufferSet

現在のレンダリングにバッファを設定します。

```
bool  DXBufferSet(
   int   context,              // グラフィックスコンテキストハンドル
   int   buffer,               // 頂点またはインデックスバッファのハンドル
   uint  start=0,              // 最初のインデックス
   uint  count=WHOLE_ARRAY     // 要素数
   );
```

パラメータ

context

[in] DXContextCreate()で作成されたグラフィックコンテキストのハンドル

buffer

[in] DXBufferCreate()で作成された頂点またはインデックスバッファのハンドル

start=0

[in] バッファの最初の要素のインデックス。デフォルトでは、バッファの先頭からのデータが使用されます。

count=WHOLE_ARRAY

[in] 使用される値の数。デフォルトはすべてのバッファ値です。

戻り値

実行成功の場合はtrue、それ以外の場合はfalseを返します。エラーコードを受け取るにはGetLastError()関数が呼ばれるべきです。

注意事項

DXBufferSet()関数は、DXDraw()を使用して頂点またはインデックスバッファをレンダリングするために呼び出されるべきです。

© 2000-2025, MetaQuotes Ltd.

DXShaderSet

レンダリング用のシェーダーを設定します。

```
bool  DXShaderSet(
   int  context,       // グラフィックスコンテキストハンドル
   int  shader         // シェーダーハンドル
   );
```

パラメータ

context

[in] DXContextCreate()で作成されたグラフィックコンテキストのハンドル

shader

[in] DXShaderCreate()で作成されたシェーダーのハンドル

戻り値

実行成功の場合はtrue、それ以外の場合はfalseを返します。エラーコードを受け取るにはGetLastError()関数が呼ばれるべきです。

注意事項

レンダリングには同時に複数の種類のシェーダーを使用できます（頂点、ジオメトリ、ピクセル）。

© 2000-2025, MetaQuotes Ltd.

DXHandleType

ハンドルの種類を返します。

```
ENUM_DX_HANDLE_TYPE  DXHandleType(
   int  handle      // ハンドル
   );
```

パラメータ

handle

[in] ハンドル

戻り値

ENUM_DX_HANDLE_TYPE列挙体からの値

ENUM_DX_HANDLE_TYPE

ID	値	説明
DX_HANDLE_INVALID	0	無効なハンドル
DX_HANDLE_CONTEXT	1	グラフィックスコンテキストハンドル
DX_HANDLE_SHADER	2	シェーダーハンドル
DX_HANDLE_BUFFER	3	頂点またはインデックスバッファのハンドル
DX_HANDLE_INPUT	4	シェーダー入力のハンドル
DX_HANDLE_TEXTURE	5	テクスチャハンドル

© 2000–2025, MetaQuotes Ltd.

DXRelease

ハンドルを解放します。

```
bool  DXRelease(
   int  handle      // ハンドル
   );
```

パラメータ

context

[in] 解放されたハンドル

戻り値

実行成功の場合は true、それ以外の場合は falseを返します。エラーコードを受け取るには GetLastError() 関数が呼ばれるべきです。

注意事項

使用されなくなったすべての作成済みハンドルは、DXRelease()関数によって明示的に解放する必要があります。

© 2000-2025, MetaQuotes Ltd.

Pythonとの統合のためのMetaTraderモジュール

MQL5は金融市場での高性能取引アプリケーションの開発向けに設計されており、アルゴリズム取引で使用される他の特殊言語の中では他に類を見ないものです。MQL5プログラムの構文と速度は可能な限りC++に近いもので、OpenCLならびMS Visual Studioとの統合がサポートされています。また、統計、ファジィ論理、ALGLIBのライブラリも用意されています。MetaEditor開発環境はネイティブで.NETライブラリをサポートしており、「スマート」な関数インポート機能があるため、特別なラッパーを開発する必要がありません。サードパーティ製のC++ DLLも利用できます。 C++ソースコードファイル(CPPとH)はエディタから直接編集してDLLにコンパイルすることができます。これには、ユーザのPCにインストールされているMicrosoft Visual Studioを使用することができます。

Pythonは、スクリプトやアプリケーションを開発するための現代的な高級プログラミング言語で、機械学習、プロセス自動化、ならびにデータ分析および視覚化のための複数のライブラリが含まれます。

Python用MetaTraderパッケージは、MetaTrader 5ターミナルから直接プロセッサ間通信を介して交換データを便利に素早く取得するために設計されています。このようにして受信したデータは、統計計算や機械学習にも使用できます。

コマンドラインからのパッケージのインストール:

```
pip install MetaTrader5
```

コマンドラインからのパッケージの更新:

```
pip install --upgrade MetaTrader5
```

MetaTrader 5とPythonを統合するための関数

関数	アクション
initialize	MetaTrader 5ターミナルとの接続を確立します
login	指定されたパラメータを使用して取引口座に接続します
shutdown	MetaTrader 5ターミナルへの以前に確立された接続を閉じます
version	MetaTrader 5ターミナルバージョンを返します
last_error	最後に発生したエラーのデータを返します
account_info	現在の口座に関する情報を取得します
terminal_Info	接続されているMetaTrader 5ターミナルのステータスとパラメータを取得します
symbols_total	MetaTrader 5ターミナルのすべての金融商品の数を取得します
symbols_get	MetaTrader 5ターミナルのすべての金融商品を取得します
symbol_info	指定した金融商品のデータを取得します
symbol_info_tick	指定した金融商品の最後のティックを取得します
symbol_select	気配値表示ウィンドウで銘柄を選択したり、ウィンドウから銘柄を削除したりします。
market_book_add	MetaTrader5ターミナルを指定された銘柄の市場深度変更イベントにサブスクライブします

© 2000–2025, MetaQuotes Ltd.

関数	アクション
market_book_get	指定された銘柄のMarketDepthエントリを特徴とするBookInfoからタプルを返します
market_book_release	MetaTrader5ターミナルの指定された銘柄の板情報変更イベントへの購読をキャンセルします
copy_rates_from	指定された日以降のバーをMetaTrader 5ターミナルから取得します
copy_rates_from_pos	指定されたインデックス以降のバーをMetaTrader 5ターミナルから取得します
copyrates_range	指定された期間のバーをMetaTrader 5ターミナルから取得します
copy_ticks_from	指定された日以降のティックをMetaTrader 5ターミナルから取得します
copy_ticks_range	指定された期間のティックをMetaTrader 5ターミナルから取得します
orders_total	アクティブな注文の数を取得します
orders_get	アクティブな注文を取得します(銘柄またはチケットでフィルターする機能あり)
order_calc_margin	指定された取引操作を実行するための証拠金を口座通貨で返します
order_calc_profit	指定された取引操作の利益を口座通貨で返します
order_check	必要な取引操作を実行するために充分な資金があるかを確認します
order_send	取引操作を実行するリクエストを送信します
positions_total	ポジションの数を取得します
positions_get	ポジションを取得します(銘柄またはチケットでフィルターする機能あり)
history_orders_total	指定された期間内の取引履歴の注文数を取得します
history_orders_get	取引履歴から注文を取得します(チケットまたはポジションでフィルターする機能あり)
history_deals_total	指定された期間内の取引履歴の取引数を取得します
history_deals_get	取引履歴から取引を取得します(チケットまたはポジションでフィルターする機能あり)

PythonをMetaTrader 5に接続する例

1. 最新バージョンのPython 3.8をhttps://www.python.org/downloads/windowsからダウンロードします。
2. Pythonをインストールする際には、[Add Python 3.8 to PATH%]のチェックをオンにしてPythonスクリプトをコマンドラインから実行できるようにします。
3. MetaTrader 5モジュールをコマンドラインからインストールします。

```
pip install MetaTrader5
```

4. matplotlibパッケージとpandasパッケージを追加します。

```
pip install matplotlib
pip install pandas
```

© 2000–2025, MetaQuotes Ltd.

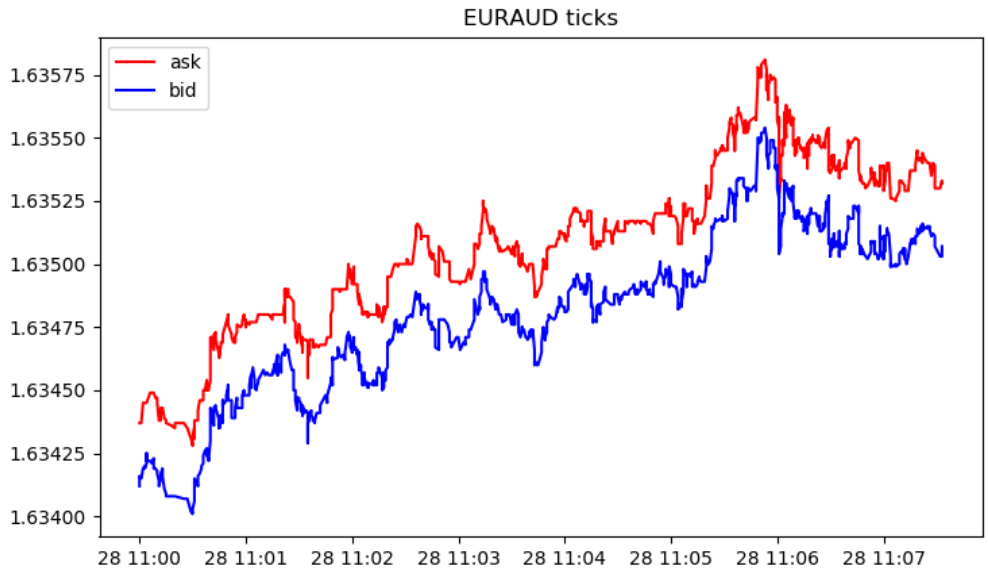
5. テスト スクリプト を起 動します。

```
from datetime import datetime
import matplotlib.pyplot as plt
import pandas as pd
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()
import MetaTrader5 as mt5

# MetaTrader 5に接続する
if not mt5.initialize():
    print("initialize() failed")
    mt5.shutdown()

# 接続状態とパラメータをリクエストする
print(mt5.terminal_info())
# MetaTrader 5バージョンについてのデータを取得する
print(mt5.version())

# EURAUDから1，000ティックをリクエストする
euraud_ticks = mt5.copy_ticks_from("EURAUD", datetime(2020,1,28,13), 1000,
mt5.COPY_TICKS_ALL)
# AUDUSDから2019.04.01 13:00 - 2019.04.02 13:00のティックをリクエストする
audusd_ticks = mt5.copy_ticks_range("AUDUSD", datetime(2020,1,27,13),
datetime(2020,1,28,13), mt5.COPY_TICKS_ALL)

# 数々の方法で異なる銘柄からバーを取得する
eurusd_rates = mt5.copy_rates_from("EURUSD", mt5.TIMEFRAME_M1,
datetime(2020,1,28,13), 1000)
eurgbp_rates = mt5.copy_rates_from_pos("EURGBP", mt5.TIMEFRAME_M1, 0, 1000)
eurcad_rates = mt5.copy_rates_range("EURCAD", mt5.TIMEFRAME_M1,
datetime(2020,1,27,13), datetime(2020,1,28,13))

# MetaTrader 5への接続をシャットダウンする
mt5.shutdown()

#データ
print('euraud_ticks(', len(euraud_ticks), ')')
for val in euraud_ticks[:10]: print(val)

print('audusd_ticks(', len(audusd_ticks), ')')
for val in audusd_ticks[:10]: print(val)

print('eurusd_rates(', len(eurusd_rates), ')')
for val in eurusd_rates[:10]: print(val)

print('eurgbp_rates(', len(eurgbp_rates), ')')
for val in eurgbp_rates[:10]: print(val)

print('eurcad_rates(', len(eurcad_rates), ')')
```

© 2000–2025, MetaQuotes Ltd.

```
for val in eurcad_rates[:10]: print(val)

#PLOT
# 取得したデータからDataFrameを作成する
ticks_frame = pd.DataFrame(euraud_ticks)
# 秒での時間をdatetime形式に変換する
ticks_frame['time']=pd.to_datetime(ticks_frame['time'], unit='s')
# チャートにティックを表示する
plt.plot(ticks_frame['time'], ticks_frame['ask'], 'r-', label='ask')
plt.plot(ticks_frame['time'], ticks_frame['bid'], 'b-', label='bid')

# 凡例を表示する
plt.legend(loc='upper left')

# ヘッダを追加する
plt.title('EURAUD ticks')

# チャートを表示する
plt.show()
```

6. データとチャートを取得します。




```
[2, 'MetaQuotes-Demo', '16167573']
[500, 2325, '19 Feb 2020']

euraud_ticks( 1000 )
(1580209200, 1.63412, 1.63437, 0., 0, 1580209200067, 130, 0.)
(1580209200, 1.63416, 1.63437, 0., 0, 1580209200785, 130, 0.)
(1580209201, 1.63415, 1.63437, 0., 0, 1580209201980, 130, 0.)
(1580209202, 1.63419, 1.63445, 0., 0, 1580209202192, 134, 0.)
(1580209203, 1.6342, 1.63445, 0., 0, 1580209203004, 130, 0.)
(1580209203, 1.63419, 1.63445, 0., 0, 1580209203487, 130, 0.)
(1580209203, 1.6342, 1.63445, 0., 0, 1580209203694, 130, 0.)
```

© 2000–2025, MetaQuotes Ltd.

```
(1580209203, 1.63419, 1.63445, 0., 0, 1580209203990, 130, 0.)
(1580209204, 1.63421, 1.63445, 0., 0, 1580209204194, 130, 0.)
(1580209204, 1.63425, 1.63445, 0., 0, 1580209204392, 130, 0.)
audusd_ticks( 40449 )
(1580122800, 0.67858, 0.67868, 0., 0, 1580122800244, 130, 0.)
(1580122800, 0.67858, 0.67867, 0., 0, 1580122800429, 4, 0.)
(1580122800, 0.67858, 0.67865, 0., 0, 1580122800817, 4, 0.)
(1580122801, 0.67858, 0.67866, 0., 0, 1580122801618, 4, 0.)
(1580122802, 0.67858, 0.67865, 0., 0, 1580122802928, 4, 0.)
(1580122809, 0.67855, 0.67865, 0., 0, 1580122809526, 130, 0.)
(1580122809, 0.67855, 0.67864, 0., 0, 1580122809699, 4, 0.)
(1580122813, 0.67855, 0.67863, 0., 0, 1580122813576, 4, 0.)
(1580122815, 0.67856, 0.67863, 0., 0, 1580122815190, 130, 0.)
(1580122815, 0.67855, 0.67863, 0., 0, 1580122815479, 130, 0.)
eurusd_rates( 1000 )
(1580149260, 1.10132, 1.10151, 1.10131, 1.10149, 44, 1, 0)
(1580149320, 1.10149, 1.10161, 1.10143, 1.10154, 42, 1, 0)
(1580149380, 1.10154, 1.10176, 1.10154, 1.10174, 40, 2, 0)
(1580149440, 1.10174, 1.10189, 1.10168, 1.10187, 47, 1, 0)
(1580149500, 1.10185, 1.10191, 1.1018, 1.10182, 53, 1, 0)
(1580149560, 1.10182, 1.10184, 1.10176, 1.10183, 25, 3, 0)
(1580149620, 1.10183, 1.10187, 1.10177, 1.10187, 49, 2, 0)
(1580149680, 1.10187, 1.1019, 1.1018, 1.10187, 53, 1, 0)
(1580149740, 1.10187, 1.10202, 1.10187, 1.10198, 28, 2, 0)
(1580149800, 1.10198, 1.10198, 1.10183, 1.10188, 39, 2, 0)
eurgbp_rates( 1000 )
(1582236360, 0.83767, 0.83767, 0.83764, 0.83765, 23, 9, 0)
(1582236420, 0.83765, 0.83765, 0.83764, 0.83765, 15, 8, 0)
(1582236480, 0.83765, 0.83766, 0.83762, 0.83765, 19, 7, 0)
(1582236540, 0.83765, 0.83768, 0.83758, 0.83763, 39, 6, 0)
(1582236600, 0.83763, 0.83768, 0.83763, 0.83767, 21, 6, 0)
(1582236660, 0.83767, 0.83775, 0.83765, 0.83769, 63, 5, 0)
(1582236720, 0.83769, 0.8377, 0.83758, 0.83764, 40, 7, 0)
(1582236780, 0.83766, 0.83769, 0.8376, 0.83766, 37, 6, 0)
(1582236840, 0.83766, 0.83772, 0.83763, 0.83772, 22, 6, 0)
(1582236900, 0.83772, 0.83773, 0.83768, 0.8377, 36, 5, 0)
eurcad_rates( 1441 )
(1580122800, 1.45321, 1.45329, 1.4526, 1.4528, 146, 15, 0)
(1580122860, 1.4528, 1.45315, 1.45274, 1.45301, 93, 15, 0)
(1580122920, 1.453, 1.45304, 1.45264, 1.45264, 82, 15, 0)
(1580122980, 1.45263, 1.45279, 1.45231, 1.45277, 109, 15, 0)
(1580123040, 1.45275, 1.4528, 1.45259, 1.45271, 53, 14, 0)
(1580123100, 1.45273, 1.45285, 1.45269, 1.4528, 62, 16, 0)
(1580123160, 1.4528, 1.45284, 1.45267, 1.45282, 64, 14, 0)
(1580123220, 1.45282, 1.45299, 1.45261, 1.45272, 48, 14, 0)
(1580123280, 1.45272, 1.45275, 1.45255, 1.45275, 74, 14, 0)
(1580123340, 1.45275, 1.4528, 1.4526, 1.4528, 94, 13, 0)
```

© 2000–2025, MetaQuotes Ltd.

© 2000–2025, MetaQuotes Ltd.

initialize

MetaTrader 5ターミナルとの接続を確立します。3つの呼び出しオプションがあります。

パラメータなしでの呼び出し: 接続するターミナルは自動的に検出されます。

```
initialize()
```

接続したいMetaTrader 5ターミナルへのパスを指定した呼び出し:

```
initialize(
   path                          // MetaTrader 5ターミナルのEXEファイルへのパス
   )
```

取引口座のパスとパラメータを指定した呼び出し:

```
initialize(
   path                          // MetaTrader 5ターミナルのEXEファイルへのパス file
   timeout=TIMEOUT,              // timeout
   login=LOGIN,                  // 口座番号
   password="PASSWORD",          // パスワード
   server="SERVER"               // ターミナルで指定されているサーバ名
   )
```

パラメータ

path

[in] metatrader.exeまたはmetatrader64.exeファイルへのパス。名前なしオプションパラメータ。最初にパラメータ名なしで示されます。パスが指定されていない場合、モジュールは自分自身で実行可能ファイルを見つけようとします。

timeout=TIMEOUT

[in] ミリ秒単位の接続タイムアウトです。名前付きオプションパラメータ。指定されていない場合、60000（60秒）の値が適用されます。指定された時間内に接続が確立されない場合、呼び出しは強制的に終了され、例外が生成されます。

login=LOGIN

[in] 取引口座番号。名前付きオプションパラメータ。指定されていない場合、最後の取引口座が使用されます。

password="PASSWORD"

[in] 取引口座のパスワード。名前付きオプションパラメータ。パスワードが設定されていない場合、ターミナルデータベースに保存されているパスワードが自動的に適用されます。

server="SERVER"

[in] 取引サーバ名。名前付きオプションパラメータ。サーバが設定されていない場合、最後に使用されたサーバが自動的に適用されます。

戻り値

MetaTrader 5ターミナルに接続できた場合はtrue、それ以外の場合はfalse

注意事項

必要に応じて、initialize()を呼び出す時にMetaTrader 5ターミナルを起動して接続を確立します。

© 2000-2025, MetaQuotes Ltd.

例：

```
import MetaTrader5 as mt5
# MetaTrader 5パッケージについてのデータを表示する
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# 指定された取引口座へのMetaTrader 5接続を確立する
if not mt5.initialize(login=25115284, server="MetaQuotes-Demo",password="4zatlbqx"):
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# 接続状態、サーバ名、取引口座に関するデータを表示する
print(mt5.terminal_info())
# MetaTrader 5バージョンについてのデータを表示する
print(mt5.version())

# MetaTrader 5ターミナルへの接続をシャットダウンする
mt5.shutdown()
```

参照

shutdown、terminal_info、version

© 2000–2025, MetaQuotes Ltd.

login

指定されたパラメータを使用して取引口座に接続します。

```
login(
   login,                    // 口座番号
   password="PASSWORD",      // パスワード
   server="SERVER",          // ターミナルで指定されたサーバ名
   timeout=TIMEOUT           // タイムアウト
   )
```

パラメータ

login

[in] 取引口座番号。名前なし必須パラメータ。

password

[in] 取引口座のパスワード。名前付きオプションパラメータ。パスワードが設定されていない場合、ターミナルデータベースに保存されているパスワードが自動的に適用されます。

server

[in] 取引サーバ名。名前付きオプションパラメータ。サーバが設定されていない場合、最後に使用されたサーバが自動的に適用されます。

timeout=TIMEOUT

[in] ミリ秒単位の接続タイムアウトです。名前付きオプションパラメータ。指定されていない場合、60000（60秒）の値が適用されます。指定された時間内に接続が確立されない場合、呼び出しは強制的に終了され、例外が生成されます。

戻り値

取引口座に接続できた場合はtrue、それ以外の場合はfalse

例：

```
import MetaTrader5 as mt5
# MetaTrader 5パッケージについてのデータを表示する
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5ターミナルとの接続を確立する
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# MetaTrader 5バージョンについてのデータを表示する
print(mt5.version())
# パスワードとサーバを指定して取引口座に接続する
account=17221085
authorized=mt5.login(account)  # ターミナルデータベースのパスワードは、接続データが保存されるように設定されている場合に適用される
if authorized:
    print("connected to account #{}".format(account))
```

© 2000–2025, MetaQuotes Ltd.

```
else:
    print("failed to connect at account #{}, error code: {}".format(account,
mt5.last_error()))

# 次に、パスワードを指定して別の取引口座に接続する
account=25115284
authorized=mt5.login(account, password="gqrtz0lbdm")
if authorized:
    # 取引口座データを「そのまま」表示する
    print(mt5.account_info())
    # 取引口座データをリスト形式で表示する
    print("Show account_info()._asdict():")
    account_info_dict = mt5.account_info()._asdict()
    for prop in account_info_dict:
        print("  {}={}".format(prop, account_info_dict[prop]))
else:
    print("failed to connect at account #{}, error code: {}".format(account,
mt5.last_error()))

# MetaTrader 5ターミナルへの接続をシャットダウンする
mt5.shutdown()


Result:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29
[500, 2367, '23 Mar 2020']

connected to account #17221085

connected to account #25115284
AccountInfo(login=25115284, trade_mode=0, leverage=100, limit_orders=200,
margin_so_mode=0, ...
account properties:
   login=25115284
   trade_mode=0
   leverage=100
   limit_orders=200
   margin_so_mode=0
   trade_allowed=True
   trade_expert=True
   margin_mode=2
   currency_digits=2
   fifo_close=False
   balance=99588.33
   credit=0.0
   profit=-45.23
   equity=99543.1
   margin=54.37
   margin_free=99488.73
```

© 2000–2025, MetaQuotes Ltd.

```
    margin_level=183084.6054809638
    margin_so_call=50.0
    margin_so_so=30.0
    margin_initial=0.0
    margin_maintenance=0.0
    assets=0.0
    liabilities=0.0
    commission_blocked=0.0
    name=James Smith
    server=MetaQuotes-Demo
    currency=USD
    company=MetaQuotes Software Corp.
```

参照

initialize、shutdown

© 2000–2025, MetaQuotes Ltd.

shutdown

MetaTrader 5ターミナルへの以前に確立された接続を閉じます。

```
shutdown()
```

戻り値

なし

例：

```
import MetaTrader5 as mt5
# MetaTrader 5パッケージについてのデータを表示する
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5ターミナルとの接続を確立する
if not mt5.initialize():
    print("initialize() failed")
    quit()

# 接続状態、サーバ名、取引口座に関するデータを表示する
print(mt5.terminal_info())
# MetaTrader 5バージョンについてのデータを表示する
print(mt5.version())

# MetaTrader 5ターミナルへの接続をシャットダウンする
mt5.shutdown()
```

参照

initialize、login_py、terminal_info、version

© 2000–2025, MetaQuotes Ltd.

version

MetaTrader 5ターミナルバージョンを返します。

```
version()
```

戻り値

MetaTrader 5のターミナルバージョン、ビルドおよびリリース日を返します。エラーの場合はNoneを返します。エラーに関する情報はlast_error()を使用して取得できます。

注意事項

version()関数は、ターミナルバージョン、ビルド、リリース日を3つの値のタプルとして返します。

種類	説明	値の例
integer	MetaTrader 5ターミナルバージョン	500
integer	ビルド	2007
string	ビルドリリース日	'25 Feb 2019'

例：

```
import MetaTrader5 as mt5
import pandas as pd
# MetaTrader 5パッケージについてのデータを表示する
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5ターミナルとの接続を確立する
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# MetaTrader 5バージョンについてのデータを表示する
print(mt5.version())

# 接続状態、サーバ名、取引口座に関するデータを「そのまま」表示する
print(mt5.terminal_info())
print()

# ディクショナリの形でプロパティを取得する
terminal_info_dict=mt5.terminal_info()._asdict()
# ディクショナリをDataFrameに変換して出力する
df=pd.DataFrame(list(terminal_info_dict.items()),columns=['property','value'])
print("terminal_info() as dataframe:")
print(df[:-1])

# MetaTrader 5ターミナルへの接続をシャットダウンする
```

© 2000–2025, MetaQuotes Ltd.

```
mt5.shutdown()


Result:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29
[500, 2367, '23 Mar 2020']
TerminalInfo(community_account=True, community_connection=True, connected=True,
dlls_allowed=False, trade_allowed=False, ...

terminal_info() as dataframe:
                  property                          value
0       community_account                           True
1    community_connection                           True
2               connected                           True
3            dlls_allowed                          False
4           trade_allowed                          False
5       tradeapi_disabled                          False
6           email_enabled                          False
7             ftp_enabled                          False
8   notifications_enabled                          False
9                    mqid                          False
10                  build                           2367
11                maxbars                           5000
12               codepage                           1251
13             ping_last                          77881
14      community_balance                        707.107
15         retransmission                              0
16                company     MetaQuotes Software Corp.
17                   name                   MetaTrader 5
18               language                        Russian
19                   path  E:\ProgramFiles\MetaTrader 5
20              data_path  E:\ProgramFiles\MetaTrader 5
```

参照

initialize、shutdown、terminal_info

© 2000–2025, MetaQuotes Ltd.

last_error

最後に発生したエラーのデータを返します。

```
last_error()
```

戻り値

最後のエラーコードと説明をタプルとして返します。

注意事項

last_error()を使用すると、MetaTrader 5ライブラリ関数の実行に失敗した場合にエラーコードを取得できます。これはGetLastError()に似ていますが、独自のエラーコードのみに適用されます。可能な値:

定数	値	説明
RES_S_OK	1	一般的な成功
RES_E_FAIL	-1	一般的な失敗
RES_E_INVALID_PARAMS	-2	無効な引数/パラメータ
RES_E_NO_MEMORY	-3	メモリ条件なし
RES_E_NOT_FOUND	-4	履歴なし
RES_E_INVALID_VERSION	-5	無効なバージョン
RES_E_AUTH_FAILED	-6	許可が失敗
RES_E_UNSUPPORTED	-7	サポートされていないメソッド
RES_E_AUTO_TRADING_DISABLED	-8	自動売買が無効
RES_E_INTERNAL_FAIL	-10000	内部IPCの一般的なエラー
RES_E_INTERNAL_FAIL_SEND	-10001	内部IPC送信の失敗
RES_E_INTERNAL_FAIL_RECEIVE	-10002	内部IPC受信の失敗
RES_E_INTERNAL_FAIL_INIT	-10003	内部IPC初期化の失敗
RES_E_INTERNAL_FAIL_CONNECT	-10003	内部IPCのipcなし
RES_E_INTERNAL_FAIL_TIMEOUT	-10005	内部タイムアウト

例:

```
import MetaTrader5 as mt5
# MetaTrader 5パッケージについてのデータを表示する
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5ターミナルとの接続を確立する
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()
```

© 2000–2025, MetaQuotes Ltd.

```
# MetaTrader 5ターミナルへの接続をシャットダウンする
mt5.shutdown()
```

参照

version、GetLastError

© 2000-2025, MetaQuotes Ltd.

account_info

現在の口座に関する情報を取得します。

```
account_info()
```

戻り値

名前付きタプル構造(namedtuple)の形式で情報を返します。エラーの場合はNoneを返します。エラーに関する情報はlast_error()を使用して取得できます。

注意事項

この関数では、AccountInfoInteger、AccountInfoDouble、AccountInfoStringを使用して取得できるすべてのデータを1回の呼び出しで返します。

例：

```
import MetaTrader5 as mt5
import pandas as pd
# MetaTrader 5パッケージについてのデータを表示する
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5ターミナルとの接続を確立する
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# パスワードとサーバを指定して取引口座に接続する
authorized=mt5.login(25115284, password="gqz0343lbdm")
if authorized:
    account_info=mt5.account_info()
    if account_info!=None:
        # display trading account data 'as is'
        print(account_info)
        # display trading account data in the form of a dictionary
        print("Show account_info()._asdict():")
        account_info_dict = mt5.account_info()._asdict()
        for prop in account_info_dict:
            print("  {}={}".format(prop, account_info_dict[prop]))
        print()

       # ディクショナリをDataFrameに変換して出力する
        df=pd.DataFrame
(list(account_info_dict.items()),columns=['property','value'])
        print("account_info() as dataframe:")
        print(df)
else:
    print("failed to connect to trade account 25115284 with password=gqz0343lbdm,
error code =",mt5.last_error())
```

© 2000–2025, MetaQuotes Ltd.

```
# MetaTrader 5ターミナルへの接続をシャットダウンする
mt5.shutdown()
```

結果:

```
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29
AccountInfo(login=25115284, trade_mode=0, leverage=100, limit_orders=200,
margin_so_mode=0, ....
Show account_info()._asdict():
  login=25115284
  trade_mode=0
  leverage=100
  limit_orders=200
  margin_so_mode=0
  trade_allowed=True
  trade_expert=True
  margin_mode=2
  currency_digits=2
  fifo_close=False
  balance=99511.4
  credit=0.0
  profit=41.82
  equity=99553.22
  margin=98.18
  margin_free=99455.04
  margin_level=101398.67590140559
  margin_so_call=50.0
  margin_so_so=30.0
  margin_initial=0.0
  margin_maintenance=0.0
  assets=0.0
  liabilities=0.0
  commission_blocked=0.0
  server=MetaQuotes-Demo
  currency=USD
  company=MetaQuotes Software Corp.

account_info() as dataframe
              property                     value
0                 login                  25115284
1            trade_mode                         0
2              leverage                       100
3          limit_orders                       200
4        margin_so_mode                         0
5         trade_allowed                      True
6          trade_expert                      True
7           margin_mode                         2
8      currency_digits                         2
```

© 2000–2025, MetaQuotes Ltd.

```
9           fifo_close                       False
10             balance                     99588.3
11              credit                           0
12              profit                      -45.13
13              equity                     99543.2
14              margin                       54.37
15         margin_free                     99488.8
16        margin_level                      183085
17      margin_so_call                          50
18        margin_so_so                          30
19      margin_initial                           0
20  margin_maintenance                           0
21              assets                           0
22         liabilities                           0
23  commission_blocked                           0
24                name                 James Smith
25              server             MetaQuotes-Demo
26            currency                         USD
27             company  MetaQuotes Software Corp.
```

参照

initialize、shutdown、login

© 2000–2025, MetaQuotes Ltd.

terminal_info

接続されているMetaTrader 5クライアントターミナルのステータスと設定を取得します。

```
terminal_info()
```

戻り値

名前付きタプル構造(namedtuple)の形式で情報を返します。エラーの場合はNoneを返します。エラーに関する情報はlast_error()を使用して取得できます。

注意事項

この関数では、>TerminalInfoInteger、TerminalInfoDouble、TerminalInfoDoubleを使用して取得できるすべてのデータを1回の呼び出しで返します。

例：

```
import MetaTrader5 as mt5
import pandas as pd
# MetaTrader 5パッケージについてのデータを表示する
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
# MetaTrader 5パッケージについてのデータを表示する
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5ターミナルとの接続を確立する
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# MetaTrader 5バージョンについてのデータを表示する
print(mt5.version())
# ターミナルの設定とステータスに関する情報を表示
terminal_info=mt5.terminal_info()
if terminal_info!=None:
    # ターミナルデータを「そのまま」表示する
    print(terminal_info)
    # データをリスト形式で表示する
    print("Show terminal_info()._asdict():")
    terminal_info_dict = mt5.terminal_info()._asdict()
    for prop in terminal_info_dict:
        print("  {}={}".format(prop, terminal_info_dict[prop]))
    print()
   # ディクショナリをDataFrameに変換して出力する
    df=pd.DataFrame(list(terminal_info_dict.items()),columns=['property','value'])
    print("terminal_info() as dataframe:")
    print(df)

# MetaTrader 5ターミナルへの接続をシャットダウンする
```

© 2000–2025, MetaQuotes Ltd.

```
mt5.shutdown()
```

結果:

```
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29
[500, 2366, '20 Mar 2020']
TerminalInfo(community_account=True, community_connection=True, connected=True,....
Show terminal_info()._asdict():
  community_account=True
  community_connection=True
  connected=True
  dlls_allowed=False
  trade_allowed=False
  tradeapi_disabled=False
  email_enabled=False
  ftp_enabled=False
  notifications_enabled=False
  mqid=False
  build=2366
  maxbars=5000
  codepage=1251
  ping_last=77850
  community_balance=707.10668201585
  retransmission=0.0
  company=MetaQuotes Software Corp.
  name=MetaTrader 5
  language=Russian
  path=E:\ProgramFiles\MetaTrader 5
  data_path=E:\ProgramFiles\MetaTrader 5
  commondata_path=C:\Users\Rosh\AppData\Roaming\MetaQuotes\Terminal\Common

terminal_info() as dataframe:
                 property                      value
0       community_account                       True
1    community_connection                       True
2               connected                       True
3            dlls_allowed                      False
4           trade_allowed                      False
5       tradeapi_disabled                      False
6           email_enabled                      False
7             ftp_enabled                      False
8   notifications_enabled                      False
9                    mqid                      False
10                  build                       2367
11                maxbars                       5000
12               codepage                       1251
13             ping_last                      80953
14      community_balance                    707.107
```

© 2000–2025, MetaQuotes Ltd.

```
15         retransmission                   0.063593
16                company  MetaQuotes Software Corp.
17                   name                MetaTrader 5
18               language                     Russian
```

参照

initialize、shutdown、version

© 2000–2025, MetaQuotes Ltd.

symbols_total

MetaTrader 5ターミナルのすべての金融商品の数を取得します。

```
symbols_total()
```

戻り値

整数値。

注意事項

この関数はSymbolsTotal()に似ています。ただし、カスタム銘柄及び気配値表示で無効にされた銘柄を含むすべての銘柄の数を返します。

例：

```
import MetaTrader5 as mt5
# MetaTrader 5パッケージについてのデータを表示する
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5ターミナルとの接続を確立する
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# 金融商品の数を取得する
symbols=mt5.symbols_total()
if symbols>0:
    print("Total symbols =",symbols)
else:
    print("symbols not found")

# MetaTrader 5ターミナルへの接続をシャットダウンする
mt5.shutdown()
```

参照

symbols_get、symbol_select、symbol_info

© 2000–2025, MetaQuotes Ltd.

symbols_get

MetaTrader 5ターミナルのすべての金融商品を取得します。

```
symbols_get(
   group="GROUP"       // 銘柄選択フィルター
)
```

group="GROUP"

[in] 必要な銘柄のグループを配置するためのフィルター。オプションパラメータ。グループが指定されている場合、関数は指定された基準を満たす銘柄のみを返します。

戻り値

タプル形式で銘柄を返します。エラーの場合はNoneを返します。エラーに関する情報はlast_error()を使用して取得できます。

注意事項

*group*パラメータを使用すると、銘柄を名前で並べ替えることができます。「*」は文字列の先頭と末尾で使用できます。

*group*パラメータは名前付きとしても名前なしとしても利用できます。両方のオプションは同じように機能します。名前付きオプション(group="GROUP")ではコードがより読みやすくなります。

*group*パラメータには、複数のコンマ区切りの条件を含めることができます。条件は「*」を使用してマスクとして設定できます。論理否定記号「！」は除外に使用できます。すべての条件は順番に適用されます。つまり、グループに含める条件を最初に指定してから、除外条件を指定する必要があります。例えば、「*group*="*, !EUR"」では、全銘柄を最初に選択してから、銘柄名に「EUR」が含まれる銘柄を後で除外します。

symbol_info()と異なり、symbols_get()関数は1回の呼び出しで、すべてのリクエストされた銘柄のデータを返します。

例：

```
import MetaTrader5 as mt5
# MetaTrader 5パッケージについてのデータを表示する
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5ターミナルとの接続を確立する
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# すべての銘柄を取得する
symbols=mt5.symbols_get()
print('Symbols: ', len(symbols))
count=0
# 初めの5銘柄を表示する
for s in symbols:
    count+=1
```

© 2000–2025, MetaQuotes Ltd.

```
    print("{}. {}".format(count,s.name))
    if count==5: break
print()

# 名前にRUを含む銘柄を取得する
ru_symbols=mt5.symbols_get("*RU*")
print('len(*RU*): ', len(ru_symbols))
for s in ru_symbols:
    print(s.name)
print()

# 名前にUSD、EUR、JPY、GBPを含まない銘柄を取得する
group_symbols=mt5.symbols_get(group="*,!*USD*,!*EUR*,!*JPY*,!*GBP*")
print('len(*,!*USD*,!*EUR*,!*JPY*,!*GBP*):', len(group_symbols))
for s in group_symbols:
    print(s.name,":",s)

# MetaTrader 5ターミナルへの接続をシャットダウンする
mt5.shutdown()
```

結果:

```
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29
Symbols:  84
1. EURUSD
2. GBPUSD
3. USDCHF
4. USDJPY
5. USDCNH

len(*RU*):  8
EURUSD
USDRUB
USDRUR
EURRUR
EURRUB
FORTS.RUB.M5
EURUSD_T20
EURUSD4

len(*,!*USD*,!*EUR*,!*JPY*,!*GBP*):  13
AUDCAD : SymbolInfo(custom=False, chart_mode=0, select=True, visible=True,
session_deals=0, session_buy_orders=0, session...
AUDCHF : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False,
session_deals=0, session_buy_orders=0, sessi...
AUDNZD : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False,
session_deals=0, session_buy_orders=0, sessi...
CADCHF : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False,
session_deals=0, session_buy_orders=0, sessi...
```

© 2000–2025, MetaQuotes Ltd.

```
NZDCAD : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False,
session_deals=0, session_buy_orders=0, sessi...
NZDCHF : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False,
session_deals=0, session_buy_orders=0, sessi...
NZDSGD : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False,
session_deals=0, session_buy_orders=0, sessi...
CADMXN : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False,
session_deals=0, session_buy_orders=0, sessi...
CHFMXN : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False,
session_deals=0, session_buy_orders=0, sessi...
NZDMXN : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False,
session_deals=0, session_buy_orders=0, sessi...
FORTS.RTS.M5 : SymbolInfo(custom=True, chart_mode=0, select=False, visible=False,
session_deals=0, session_buy_orders=0, ...
FORTS.RUB.M5 : SymbolInfo(custom=True, chart_mode=0, select=False, visible=False,
session_deals=0, session_buy_orders=0, ...
FOREX.CHF.M5 : SymbolInfo(custom=True, chart_mode=0, select=False, visible=False,
session_deals=0, session_buy_orders=0, ...
```

参照

symbols_total、symbol_select、symbol_info

© 2000–2025, MetaQuotes Ltd.

symbol_info

指定した金融商品のデータを取得します。

```
symbol_info(
   symbol       // 金融商品名
)
```

symbol

[in] 金融商品名。名前なし必須パラメータ。

戻り値

名前付きタプル構造(namedtuple)の形式で情報を返します。エラーの場合はNoneを返します。エラーに関する情報はlast_error()を使用して取得できます。

注意事項

この関数では、SymbolInfoInteger、SymbolInfoDouble、SymbolInfoStringを使用して取得できるすべてのデータを1回の呼び出しで返します。

例：

```
import MetaTrader5 as mt5
# MetaTrader 5パッケージについてのデータを表示する
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5ターミナルとの接続を確立する
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# 気配値表示でEURJPY銘柄の表示を有効にする試み
selected=mt5.symbol_select("EURJPY",True)
if not selected:
    print("Failed to select EURJPY")
    mt5.shutdown()
    quit()

# EURJPY銘柄プロパティを表示する
symbol_info=mt5.symbol_info("EURJPY")
if symbol_info!=None:
    # ターミナルデータを「そのまま」表示する
    print(symbol_info)
    print("EURJPY: spread =",symbol_info.spread,"  digits =",symbol_info.digits)
    # 銘柄のプロパティをリストとして表示する
    print("Show symbol_info(\"EURJPY\")._asdict():")
    symbol_info_dict = mt5.symbol_info("EURJPY")._asdict()
    for prop in symbol_info_dict:
        print("  {}={}".format(prop, symbol_info_dict[prop]))
```

© 2000-2025, MetaQuotes Ltd.

```
# MetaTrader 5ターミナルへの接続をシャットダウンする
mt5.shutdown()
```

結果:

```
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29
SymbolInfo(custom=False, chart_mode=0, select=True, visible=True, session_deals=0,
session_buy_orders=0, session_sell_orders=0, ...
EURJPY: spread = 17   digits = 3
Show symbol_info()._asdict():
  custom=False
  chart_mode=0
  select=True
  visible=True
  session_deals=0
  session_buy_orders=0
  session_sell_orders=0
  volume=0
  volumehigh=0
  volumelow=0
  time=1585069682
  digits=3
  spread=17
  spread_float=True
  ticks_bookdepth=10
  trade_calc_mode=0
  trade_mode=4
  start_time=0
  expiration_time=0
  trade_stops_level=0
  trade_freeze_level=0
  trade_exemode=1
  swap_mode=1
  swap_rollover3days=3
  margin_hedged_use_leg=False
  expiration_mode=7
  filling_mode=1
  order_mode=127
  order_gtc_mode=0
  option_mode=0
  option_right=0
  bid=120.024
  bidhigh=120.506
  bidlow=118.798
  ask=120.041
  askhigh=120.526
  asklow=118.828
```

© 2000–2025, MetaQuotes Ltd.

```
last=0.0
lasthigh=0.0
lastlow=0.0
volume_real=0.0
volumehigh_real=0.0
volumelow_real=0.0
option_strike=0.0
point=0.001
trade_tick_value=0.8977708350166538
trade_tick_value_profit=0.8977708350166538
trade_tick_value_loss=0.8978272580355541
trade_tick_size=0.001
trade_contract_size=100000.0
trade_accrued_interest=0.0
trade_face_value=0.0
trade_liquidity_rate=0.0
volume_min=0.01
volume_max=500.0
volume_step=0.01
volume_limit=0.0
swap_long=-0.2
swap_short=-1.2
margin_initial=0.0
margin_maintenance=0.0
session_volume=0.0
session_turnover=0.0
session_interest=0.0
session_buy_orders_volume=0.0
session_sell_orders_volume=0.0
session_open=0.0
session_close=0.0
session_aw=0.0
session_price_settlement=0.0
session_price_limit_min=0.0
session_price_limit_max=0.0
margin_hedged=100000.0
price_change=0.0
price_volatility=0.0
price_theoretical=0.0
price_greeks_delta=0.0
price_greeks_theta=0.0
price_greeks_gamma=0.0
price_greeks_vega=0.0
price_greeks_rho=0.0
price_greeks_omega=0.0
price_sensitivity=0.0
basis=
category=
currency_base=EUR
```

© 2000–2025, MetaQuotes Ltd.

```
currency_profit=JPY
currency_margin=EUR
bank=
description=Euro vs Japanese Yen
exchange=
formula=
isin=
name=EURJPY
page=http://www.google.com/finance?q=EURJPY
path=Forex\EURJPY
```

参照

account_info、terminal_info

© 2000–2025, MetaQuotes Ltd.

symbol_info_tick

指定した金融商品の最後のティックを取得します。

```
symbol_info_tick(
   symbol       // 金融商品名
)
```

symbol

[in] 金融商品名。名前なし必須パラメータ。

戻り値

タプル形式で情報を返します。エラーの場合はNoneを返します。エラーに関する情報はlast_error()を使用して取得できます。

注意事項

この関数はSymbolInfoTickに似ています。

例：

```
import MetaTrader5 as mt5
# MetaTrader 5パッケージについてのデータを表示する
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5ターミナルとの接続を確立する
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# 気配値表示でGBPUSDの表示を有効にする試み
selected=mt5.symbol_select("GBPUSD",True)
if not selected:
    print("Failed to select GBPUSD")
    mt5.shutdown()
    quit()

# 最後のGBPUSDティックを表示する
lasttick=mt5.symbol_info_tick("GBPUSD")
print(lasttick)
# ティックフィールド値をリスト形式で表示する
print("Show symbol_info_tick(\"GBPUSD\")._asdict():")
symbol_info_tick_dict = mt5.symbol_info_tick("GBPUSD")._asdict()
for prop in symbol_info_tick_dict:
    print("  {}={}".format(prop, symbol_info_tick_dict[prop]))

# MetaTrader 5ターミナルへの接続をシャットダウンする
mt5.shutdown()
```

© 2000–2025, MetaQuotes Ltd.

結果:

```
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29
Tick(time=1585070338, bid=1.17264, ask=1.17279, last=0.0, volume=0,
time_msc=1585070338728, flags=2, volume_real=0.0)
Show symbol_info_tick._asdict():
  time=1585070338
  bid=1.17264
  ask=1.17279
  last=0.0
  volume=0
  time_msc=1585070338728
  flags=2
  volume_real=0.0
```

参照

symbol_info、symbol_info

© 2000–2025, MetaQuotes Ltd.

symbol_select

気配値表示ウィンドウで銘柄を選択したり、ウィンドウから銘柄を削除したりします。

```
symbol_select(
   symbol,      // 金融商品名
   enable=None  // 有/無効化
)
```

symbol

[in] 金融商品名。名前なし必須パラメータ。

enable

[in] スイッチ。名前なしオプションパラメータ。「false」の場合、銘柄は「気配値表示」ウィンドウから削除されるべきです。その他の場合、銘柄は「気配値表示」ウィンドウで選択されるべきです。銘柄のチャートが開いている、または銘柄のポジションがある場合は、銘柄は削除することが出来ません。

戻り値

成功の場合はtrue、それ以外の場合はfalse。

注意事項

この関数はSymbolSelectに似ています。

例：

```
import MetaTrader5 as mt5
import pandas as pd
# MetaTrader 5パッケージについてのデータを表示する
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print()
# MetaTrader 5ターミナルとの接続を確立する
if not mt5.initialize(login=25115284, server="MetaQuotes-Demo",password="4zatlbqx"):
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# 気配値表示でEURCADの表示を有効にする試み
selected=mt5.symbol_select("EURCAD",True)
if not selected:
    print("Failed to select EURCAD, error code =",mt5.last_error())
else:
    symbol_info=mt5.symbol_info("EURCAD")
    print(symbol_info)
    print("EURCAD: currency_base =",symbol_info.currency_base,"  currency_profit
=",symbol_info.currency_profit,"  currency_margin =",symbol_info.currency_margin)
    print()

    # ディクショナリの形で銘柄のプロパティを取得する
    print("Show symbol_info()._asdict():")
    symbol_info_dict = symbol_info._asdict()
```

© 2000–2025, MetaQuotes Ltd.

```
    for prop in symbol_info_dict:
        print("  {}={}".format(prop, symbol_info_dict[prop]))
    print()

   # ディクショナリをDataFrameに変換して出力する
    df=pd.DataFrame(list(symbol_info_dict.items()),columns=['property','value'])
    print("symbol_info_dict() as dataframe:")
    print(df)

# MetaTrader 5ターミナルへの接続をシャットダウンする
mt5.shutdown()
```

結果:

```
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29
SymbolInfo(custom=False, chart_mode=0, select=True, visible=True, session_deals=0,
session_buy_orders=0, session_sell_orders=0, volume=0, volumehigh=0, ....
EURCAD: currency_base = EUR   currency_profit = CAD   currency_margin = EUR

Show symbol_info()._asdict():
  custom=False
  chart_mode=0
  select=True
  visible=True
  session_deals=0
  session_buy_orders=0
  session_sell_orders=0
  volume=0
  volumehigh=0
  volumelow=0
  time=1585217595
  digits=5
  spread=39
  spread_float=True
  ticks_bookdepth=10
  trade_calc_mode=0
  trade_mode=4
  start_time=0
  expiration_time=0
  trade_stops_level=0
  trade_freeze_level=0
  trade_exemode=1
  swap_mode=1
  swap_rollover3days=3
  margin_hedged_use_leg=False
  expiration_mode=7
  filling_mode=1
  order_mode=127
```

© 2000-2025, MetaQuotes Ltd.

```
order_gtc_mode=0
option_mode=0
option_right=0
bid=1.55192
bidhigh=1.55842
bidlow=1.5419800000000001
ask=1.5523099999999999
askhigh=1.55915
asklow=1.5436299999999998
last=0.0
lasthigh=0.0
lastlow=0.0
volume_real=0.0
volumehigh_real=0.0
volumelow_real=0.0
option_strike=0.0
point=1e-05
trade_tick_value=0.7043642408362214
trade_tick_value_profit=0.7043642408362214
trade_tick_value_loss=0.7044535553770941
trade_tick_size=1e-05
trade_contract_size=100000.0
trade_accrued_interest=0.0
trade_face_value=0.0
trade_liquidity_rate=0.0
volume_min=0.01
volume_max=500.0
volume_step=0.01
volume_limit=0.0
swap_long=-1.1
swap_short=-0.9
margin_initial=0.0
margin_maintenance=0.0
session_volume=0.0
session_turnover=0.0
session_interest=0.0
session_buy_orders_volume=0.0
session_sell_orders_volume=0.0
session_open=0.0
session_close=0.0
session_aw=0.0
session_price_settlement=0.0
session_price_limit_min=0.0
session_price_limit_max=0.0
margin_hedged=100000.0
price_change=0.0
price_volatility=0.0
price_theoretical=0.0
price_greeks_delta=0.0
```

© 2000–2025, MetaQuotes Ltd.

```
  price_greeks_theta=0.0
  price_greeks_gamma=0.0
  price_greeks_vega=0.0
  price_greeks_rho=0.0
  price_greeks_omega=0.0
  price_sensitivity=0.0
  basis=
  category=
  currency_base=EUR
  currency_profit=CAD
  currency_margin=EUR
  bank=
  description=Euro vs Canadian Dollar
  exchange=
  formula=
  isin=
  name=EURCAD
  page=http://www.google.com/finance?q=EURCAD
  path=Forex\EURCAD

symbol_info_dict() as dataframe:
         property                                      value
0          custom                                      False
1      chart_mode                                          0
2          select                                       True
3         visible                                       True
4   session_deals                                          0
..            ...                                        ...
91        formula
92           isin
93           name                                     EURCAD
94           page  http://www.google.com/finance?q=EURCAD
95           path                              Forex\EURCAD

[96 rows x 2 columns]
```

参照

symbol_info

© 2000–2025, MetaQuotes Ltd.

market_book_add

MetaTrader5ターミナルを指定された銘柄の市場深度変更イベントにサブスクライブします。

```
market_book_add(
   symbol       // 金融商品名
)
```

symbol

[in] 金融商品名。名前なし必須パラメータ。

戻り値

成功の場合はtrue、それ以外の場合はfalse。

注意事項

この関数はMarketBookAddに似ています。

参照

market_book_get、market_book_release、板情報構造体

© 2000–2025, MetaQuotes Ltd.

market_book_get

指定された銘柄のMarketDepthエントリを特徴とするBookInfoからタプルを返します。

```
market_book_get(
   symbol      // 金融商品名
)
```

symbol

[in] 金融商品名。名前なし必須パラメータ。

戻り値

注文タイプ、価格、およびロット単位のボリュームを特徴とするBookInfoエントリからMarketDepthコンテンツをタプルとして返します。BookInfoはMqlBookInfo構造体に似ています。

エラーの場合はNoneを返します。エラーに関する情報はlast_error()を使用して取得できます。

注意事項

Market Depth変更イベントの購読は、market_book_add()関数を使用して事前に実行する必要があります。

この関数はMarketBookGetに似ています。

例：

```
import MetaTrader5 as mt5
import time
# MetaTrader 5パッケージについてのデータを表示する
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print("")

# MetaTrader 5ターミナルとの接続を確立する
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
   # MetaTrader 5ターミナルへの接続をシャットダウンする
    mt5.shutdown()
    quit()

# EURUSDの板情報アップデートを購読する(Depth of Market)
if mt5.market_book_add('EURUSD'):
  # 10回の反復処理で板情報データを取得する
   for i in range(10):
        # 板情報コンテンツを取得する(Depth of Market)
        items = mt5.market_book_get('EURUSD')
        # 板情報全体を「現状のまま」単一の文字列で表示する
        print(items)
        # わかりやすくするために、各注文を個別に表示する
        if items:
```

© 2000-2025, MetaQuotes Ltd.

```
            for it in items:
                # 注文内容
                print(it._asdict())
        # 板情報データの次のリクエストの前に5秒間一時停止する
        time.sleep(5)
  # 板情報アップデートの購読をキャンセルする(Depth of Market)
   mt5.market_book_release('EURUSD')
else:
    print("mt5.market_book_add('EURUSD') failed, error code =",mt5.last_error())

# MetaTrader 5ターミナルへの接続をシャットダウンする
mt5.shutdown()
```

結果:

```
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.34

(BookInfo(type=1, price=1.20038, volume=250, volume_dbl=250.0), BookInfo(type=1,
price=1.20032, volume=100, volume...
{'type': 1, 'price': 1.20038, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20032, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.2003, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20028, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20026, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20025, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20023, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20017, 'volume': 250, 'volume_dbl': 250.0}
(BookInfo(type=1, price=1.2004299999999999, volume=250, volume_dbl=250.0),
BookInfo(type=1, price=1.20037, volume...
{'type': 1, 'price': 1.2004299999999999, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20037, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.20036, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20034, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20031, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20029, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20028, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20022, 'volume': 250, 'volume_dbl': 250.0}
(BookInfo(type=1, price=1.2004299999999999, volume=250, volume_dbl=250.0),
BookInfo(type=1, price=1.20037, volume...
{'type': 1, 'price': 1.2004299999999999, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20037, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.20036, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20034, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20031, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20029, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20028, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20022, 'volume': 250, 'volume_dbl': 250.0}
```

© 2000–2025, MetaQuotes Ltd.

```
(BookInfo(type=1, price=1.20036, volume=250, volume_dbl=250.0), BookInfo(type=1,
price=1.20029, volume=100, volume...
{'type': 1, 'price': 1.20036, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20029, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.20028, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20026, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20023, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20022, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20021, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20014, 'volume': 250, 'volume_dbl': 250.0}
(BookInfo(type=1, price=1.20035, volume=250, volume_dbl=250.0), BookInfo(type=1,
price=1.20029, volume=100, volume...
{'type': 1, 'price': 1.20035, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20029, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.20027, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20025, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20023, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20022, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20021, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20014, 'volume': 250, 'volume_dbl': 250.0}
(BookInfo(type=1, price=1.20037, volume=250, volume_dbl=250.0), BookInfo(type=1,
price=1.20031, volume=100, volume...
{'type': 1, 'price': 1.20037, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20031, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.2003, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20028, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20025, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20023, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20022, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20016, 'volume': 250, 'volume_dbl': 250.0}
```

参照

market_book_add、market_book_release、板情報構造体

© 2000–2025, MetaQuotes Ltd.

market_book_release

MetaTrader5ターミナルの指定された銘柄の板情報変更イベントへの購読をキャンセルします。

```
market_book_release(
   symbol      // 金融商品名
)
```

symbol

[in] 金融商品名。名前なし必須パラメータ。

戻り値

成功の場合はtrue、それ以外の場合はfalse。

注意事項

この関数はMarketBookReleaseに似ています。

参照

market_book_add、market_book_get、板情報構造体

© 2000–2025, MetaQuotes Ltd.

copy_rates_from

指定された日以降のバーをMetaTrader 5ターミナルから取得します。

```
copy_rates_from(
   symbol,       // 銘柄名
   timeframe,    // 時間枠
   date_from,    // 初めにバーが開いた日
   count         // バー数
   )
```

パラメータ

symbol

[in] 金融商品名(例: EURUSD)。名前なし必須パラメータ。

timeframe

[in] バーがリクエストされている時間枠で、TIMEFRAME列挙からの値で設定されます。名前なし必須パラメータ。

date_from

[in] リクエストされたサンプルの1番目のバーが開いた日で、「datetime」オブジェクトまたは1970年1月1日からの経過秒数として設定されます。名前なし必須パラメータ。

count

[in] 受信するバーの数名前なし必須パラメータ。

戻り値

バーを名前付きのtime、open、high、low、close、tick_volume、spread、real_volume列を持つNumPy配列として返します。エラーの場合はNoneを返します。エラーに関する情報はlast_error()を使用して取得できます。

注意事項

詳細はCopyRates()関数をご覧ください。

指定された日付とそれ以前のデータのみが戻されます。これは、が返されたバーの開始時刻は常に指定されたもの以下だということです。

MetaTrader 5ターミナルは、チャート上でユーザが利用できる履歴内でのみバーを提供します。ユーザが使用できるバーの数は「Max. bars in chart」パラメータで設定されます。

「datetime」オブジェクトを作成する際にPythonがローカルな時間帯を使用する一方、MetaTrader 5ではティックとバー開始時間はUTC時間帯(シフトなし)で保存されます。よって。時間を使用する関数を実行するためには「datetime」はUTC時間で作成されるべきです。MetaTrader 5ターミナルから受信したデータにはUTC時間があります。

TIMEFRAMEは可能なチャート期間値の列挙です。

ID	説明
TIMEFRAME_M1	1分

© 2000–2025, MetaQuotes Ltd.

ID	説 明
TIMEFRAME_M2	2分
TIMEFRAME_M3	3分
TIMEFRAME_M4	4分
TIMEFRAME_M5	5分
TIMEFRAME_M6	6分
TIMEFRAME_M10	10分
TIMEFRAME_M12	12分
TIMEFRAME_M12	15分
TIMEFRAME_M20	20分
TIMEFRAME_M30	30分
TIMEFRAME_H1	1 時間
TIMEFRAME_H2	2時間
TIMEFRAME_H3	3時間
TIMEFRAME_H4	4時間
TIMEFRAME_H6	6時間
TIMEFRAME_H8	8時間
TIMEFRAME_H12	12時間
TIMEFRAME_D1	1日
TIMEFRAME_W1	1週間
TIMEFRAME_MN1	1 ヶ月

例：

```
from datetime import datetime
import MetaTrader5 as mt5
# MetaTrader 5パッケージについてのデータを表示する
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# 表形式で得られたデータを表示するための「pandas」モジュールをインポートする
import pandas as pd
pd.set_option('display.max_columns', 500) # 表示される列の数
pd.set_option('display.width', 1500)      # 表示する表の最高幅
# 時間帯を使用するためのpytzモジュールをインポートする
import pytz
```

© 2000–2025, MetaQuotes Ltd.

```
# MetaTrader 5ターミナルとの接続を確立する
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# 時間帯をUTCに設定する
timezone = pytz.timezone("Etc/UTC")
# ローカル時間帯オフセットの実装を避けるために「datetime」オブジェクトをUTC時間帯で作成する
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
# 01.10.2020 UTC以降の10のEURUSD H4バーを取得する
rates = mt5.copy_rates_from("EURUSD", mt5.TIMEFRAME_H4, utc_from, 10)

# MetaTrader 5ターミナルへの接続をシャットダウンする
mt5.shutdown()
# 取得したデータの各要素を新しい行に表示する
print("Display obtained data 'as is'")
for rate in rates:
    print(rate)

# 取得したデータからDataFrameを作成する
rates_frame = pd.DataFrame(rates)
# 秒での時間をdatetime形式に変換する
rates_frame['time']=pd.to_datetime(rates_frame['time'], unit='s')

# データを表示する
print("\nDisplay dataframe with data")
print(rates_frame)
```

結果:

```
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Display obtained data 'as is'
(1578484800, 1.11382, 1.11385, 1.1111, 1.11199, 9354, 1, 0)
(1578499200, 1.11199, 1.11308, 1.11086, 1.11179, 10641, 1, 0)
(1578513600, 1.11178, 1.11178, 1.11016, 1.11053, 4806, 1, 0)
(1578528000, 1.11053, 1.11193, 1.11033, 1.11173, 3480, 1, 0)
(1578542400, 1.11173, 1.11189, 1.11126, 1.11182, 2236, 1, 0)
(1578556800, 1.11181, 1.11203, 1.10983, 1.10993, 7984, 1, 0)
(1578571200, 1.10994, 1.11173, 1.10965, 1.11148, 7406, 1, 0)
(1578585600, 1.11149, 1.11149, 1.10923, 1.11046, 7468, 1, 0)
(1578600000, 1.11046, 1.11097, 1.11033, 1.11051, 3450, 1, 0)
(1578614400, 1.11051, 1.11093, 1.11017, 1.11041, 2448, 1, 0)

Display dataframe with data
                 time     open     high      low    close  tick_volume  spread
real_volume
```

© 2000-2025, MetaQuotes Ltd.

```
0 2020-01-08 12:00:00  1.11382  1.11385  1.11110  1.11199         9354       1
      0
1 2020-01-08 16:00:00  1.11199  1.11308  1.11086  1.11179        10641       1
      0
2 2020-01-08 20:00:00  1.11178  1.11178  1.11016  1.11053         4806       1
      0
3 2020-01-09 00:00:00  1.11053  1.11193  1.11033  1.11173         3480       1
      0
4 2020-01-09 04:00:00  1.11173  1.11189  1.11126  1.11182         2236       1
      0
5 2020-01-09 08:00:00  1.11181  1.11203  1.10983  1.10993         7984       1
      0
6 2020-01-09 12:00:00  1.10994  1.11173  1.10965  1.11148         7406       1
      0
7 2020-01-09 16:00:00  1.11149  1.11149  1.10923  1.11046         7468       1
      0
8 2020-01-09 20:00:00  1.11046  1.11097  1.11033  1.11051         3450       1
      0
9 2020-01-10 00:00:00  1.11051  1.11093  1.11017  1.11041         2448       1
      0
```

参照

CopyRates、copy_rates_from_pos、copy_rates_range、copy_ticks_from、copy_ticks_range

© 2000–2025, MetaQuotes Ltd.

copy_rates_from_pos

指定されたインデックス以降のバーをMetaTrader 5ターミナルから取得します。

```
copy_rates_from_pos(
   symbol,       // 銘柄名
   timeframe,    // 時間枠
   start_pos,    // 初めのバーのインデックス
   count         // バー数
   )
```

パラメータ

symbol

[in] 金融商品名（例: EURUSD）。名前なし必須パラメータ。

timeframe

[in] バーがリクエストされている時間枠で、TIMEFRAME列挙からの値で設定されます。名前なし必須パラメータ。

start_pos

[in] データがリクエストされているバーの初めのインデックス。バーは現在から過去へ番号付けられるので、現在のバーは「ゼロバー」です。名前なし必須パラメータ。

count

[in] 受信するバーの数名前なし必須パラメータ。

戻り値

バーを名前付きのtime、open、high、low、close、tick_volume、spread、real_volume列を持つNumPy配列として返します。エラーの場合はNoneを返します。エラーに関する情報はlast_error()を使用して取得できます。

注意事項

詳細はCopyRates()関数をご覧ください。

MetaTrader 5ターミナルは、チャート上でユーザが利用できる履歴内でのみバーを提供します。ユーザが使用できるバーの数は「Max. bars in chart」パラメータで設定されます。

例：

```
from datetime import datetime
import MetaTrader5 as mt5
# MetaTrader 5パッケージについてのデータを表示する
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# 表形式で得られたデータを表示するための「pandas」モジュールをインポートする
import pandas as pd
pd.set_option('display.max_columns', 500) # 表示される列の数
pd.set_option('display.width', 1500)      # 表示する表の最高幅
```

© 2000–2025, MetaQuotes Ltd.

```
# MetaTrader 5ターミナルとの接続を確立する
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# 今日の10のGBPUSD D1バーを取得する
rates = mt5.copy_rates_from_pos("GBPUSD", mt5.TIMEFRAME_D1, 0, 10)

# MetaTrader 5ターミナルへの接続をシャットダウンする
mt5.shutdown()
# 取得したデータの各要素を新しい行に表示する
print("Display obtained data 'as is'")
for rate in rates:
    print(rate)

# 取得したデータからDataFrameを作成する
rates_frame = pd.DataFrame(rates)
# 秒での時間をdatetime形式に変換する
rates_frame['time']=pd.to_datetime(rates_frame['time'], unit='s')

# データを表示する
print("\nDisplay dataframe with data")
print(rates_frame)
```

結果:

```
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Display obtained data 'as is'
(1581552000, 1.29568, 1.30692, 1.29441, 1.30412, 68228, 0, 0)
(1581638400, 1.30385, 1.30631, 1.3001, 1.30471, 56498, 0, 0)
(1581897600, 1.30324, 1.30536, 1.29975, 1.30039, 49400, 0, 0)
(1581984000, 1.30039, 1.30486, 1.29705, 1.29952, 62288, 0, 0)
(1582070400, 1.29952, 1.3023, 1.29075, 1.29187, 57909, 0, 0)
(1582156800, 1.29186, 1.29281, 1.28489, 1.28792, 61033, 0, 0)
(1582243200, 1.28802, 1.29805, 1.28746, 1.29566, 66386, 0, 0)
(1582502400, 1.29426, 1.29547, 1.28865, 1.29283, 66933, 0, 0)
(1582588800, 1.2929, 1.30178, 1.29142, 1.30037, 80121, 0, 0)
(1582675200, 1.30036, 1.30078, 1.29136, 1.29374, 49286, 0, 0)

Display dataframe with data
        time     open     high      low    close  tick_volume  spread  real_volume
0 2020-02-13  1.29568  1.30692  1.29441  1.30412        68228       0            0
1 2020-02-14  1.30385  1.30631  1.30010  1.30471        56498       0            0
2 2020-02-17  1.30324  1.30536  1.29975  1.30039        49400       0            0
3 2020-02-18  1.30039  1.30486  1.29705  1.29952        62288       0            0
4 2020-02-19  1.29952  1.30230  1.29075  1.29187        57909       0            0
5 2020-02-20  1.29186  1.29281  1.28489  1.28792        61033       0            0
6 2020-02-21  1.28802  1.29805  1.28746  1.29566        66386       0            0
```

© 2000–2025, MetaQuotes Ltd.

7	2020-02-24	1.29426	1.29547	1.28865	1.29283	66933	0	0
8	2020-02-25	1.29290	1.30178	1.29142	1.30037	80121	0	0
9	2020-02-26	1.30036	1.30078	1.29136	1.29374	49286	0	0

参照

CopyRates、copy_rates_from、copy_rates_range、 copy_ticks_from、copy_ticks_range

© 2000–2025, MetaQuotes Ltd.

copy_rates_range

指定された期間のバーをMetaTrader 5ターミナルから取得します。

```
copy_rates_range(
   symbol,       // 銘柄名
   timeframe,    // 時間枠
   date_from,    // バーがリクエストされている最初の日
   date_to       // バーがリクエストされている最後の日
   )
```

パラメータ

symbol

[in] 金融商品名(例: EURUSD)。名前なし必須パラメータ。

timeframe

[in] バーがリクエストされている時間枠で、TIMEFRAME列挙からの値で設定されます。名前なし必須パラメータ。

date_from

[in] バーがリクエストされている最初の日。「datetime」オブジェクトまたは1970年1月1日からの経過秒数として設定されます。開始時間 >= date_fromのバーが返されます。名前なし必須パラメータ。

date_to

[in] バーがリクエストされている最後の日。「datetime」オブジェクトまたは1970年1月1日からの経過秒数として設定されます。開始時間 <= date_toのバーが返されます。名前なし必須パラメータ。

戻り値

バーを名前付きのtime、open、high、low、close、tick_volume、spread、real_volume列を持つNumPy配列として返します。エラーの場合はNoneを返します。エラーに関する情報はlast_error()を使用して取得できます。

注意事項

詳細はCopyRates()関数をご覧ください。

MetaTrader 5ターミナルは、チャート上でユーザが利用できる履歴内でのみバーを提供します。ユーザが使用できるバーの数は「Max. bars in chart」パラメータで設定されます。

「datetime」オブジェクトを作成する際にPythonがローカルな時間帯を使用する一方、MetaTrader 5ではティックとバー開始時間はUTC時間帯(シフトなし)で保存されます。よって。時間を使用する関数を実行するためには「datetime」はUTC時間で作成されるべきです。MetaTrader 5ターミナルから受信したデータにはUTC時間があります。

例:

```
from datetime import datetime
import MetaTrader5 as mt5
# MetaTrader 5パッケージについてのデータを表示する
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
```

© 2000-2025, MetaQuotes Ltd.

```
# テーブル形式で取得されたデータを表示するための「pandas」モジュールをインポートする
import pandas as pd
pd.set_option('display.max_columns', 500) # 表示する列の数
pd.set_option('display.width', 1500)      # 表示するテーブルの最大幅
# タイムゾーンを使用するためのpytzモジュールをインポートする
import pytz

# MetaTrader 5ターミナルとの接続を確立する
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# タイムゾーンをUTCに設定する
timezone = pytz.timezone("Etc/UTC")
# create 'datetime' objects in UTC time zone to avoid the implementation of a local
time zone offset
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
utc_to = datetime(2020, 1, 11, hour = 13, tzinfo=timezone)
# 2020.01.10 00:00-2020.01.11 13:00 UTCでUSDJPY M5からバーを取得する
rates = mt5.copy_rates_range("USDJPY", mt5.TIMEFRAME_M5, utc_from, utc_to)

# MetaTrader 5ターミナルへの接続をシャットダウンする
mt5.shutdown()

# 取得したデータの各要素を新しい行に表示する
print("Display obtained data 'as is'")
counter=0
for rate in rates:
    counter+=1
    if counter<=10:
        print(rate)

# 取得されたデータからDataFrameを作成する
rates_frame = pd.DataFrame(rates)
# 秒での時間を「datetime」形式に変換する
rates_frame['time']=pd.to_datetime(rates_frame['time'], unit='s')

# データを表示する
print("\nDisplay dataframe with data")
print(rates_frame.head(10))
```

結果:

```
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Display obtained data 'as is'
(1578614400, 109.513, 109.527, 109.505, 109.521, 43, 2, 0)
(1578614700, 109.521, 109.549, 109.518, 109.543, 215, 8, 0)
(1578615000, 109.543, 109.543, 109.466, 109.505, 98, 10, 0)
(1578615300, 109.504, 109.534, 109.502, 109.517, 155, 8, 0)
(1578615600, 109.517, 109.539, 109.513, 109.527, 71, 4, 0)
(1578615900, 109.526, 109.537, 109.484, 109.52, 106, 9, 0)
(1578616200, 109.52, 109.524, 109.508, 109.51, 205, 7, 0)
(1578616500, 109.51, 109.51, 109.491, 109.496, 44, 8, 0)
(1578616800, 109.496, 109.509, 109.487, 109.5, 85, 5, 0)
(1578617100, 109.5, 109.504, 109.487, 109.489, 82, 7, 0)
```

© 2000-2025, MetaQuotes Ltd.

```
Display dataframe with data
                 time     open     high      low    close  tick_volume  spread
real_volume
0 2020-01-10 00:00:00  109.513  109.527  109.505  109.521           43       2
       0
1 2020-01-10 00:05:00  109.521  109.549  109.518  109.543          215       8
       0
2 2020-01-10 00:10:00  109.543  109.543  109.466  109.505           98      10
       0
3 2020-01-10 00:15:00  109.504  109.534  109.502  109.517          155       8
       0
4 2020-01-10 00:20:00  109.517  109.539  109.513  109.527           71       4
       0
5 2020-01-10 00:25:00  109.526  109.537  109.484  109.520          106       9
       0
6 2020-01-10 00:30:00  109.520  109.524  109.508  109.510          205       7
       0
7 2020-01-10 00:35:00  109.510  109.510  109.491  109.496           44       8
       0
8 2020-01-10 00:40:00  109.496  109.509  109.487  109.500           85       5
       0
9 2020-01-10 00:45:00  109.500  109.504  109.487  109.489           82       7
       0
```

参照

CopyRates、copy_rates_from、copy_rates_range、copy_ticks_from、copy_ticks_range

© 2000–2025, MetaQuotes Ltd.

copy_ticks_from

指定された日以降のティックをMetaTrader 5ターミナルから取得します。

```
copy_ticks_from(
   symbol,        // 銘柄名
   date_from,     // ティックがリクエストされている最初の日
   count,         // リクエストされたティック数
   flags          // リクエストされたティックの種類を定義するためのフラグの組み合わせ
   )
```

パラメータ

symbol

[in] 金融商品名(例: EURUSD)。名前なし必須パラメータ。

date_from

[in] ティックがリクエストされている最初の日。「datetime」オブジェクトまたは1970年1月1日からの経過秒数として設定されます。名前なし必須パラメータ。

count

[in] 受信するティックの数。名前なし必須パラメータ。

フラグ

[in] リクエストされたティックのタイプを定義するフラグCOPY_TICKS_INFO – 買値と売値の変更でのティック、COPY_TICKS_TRADE –ラストと数量の変更でのティック、COPY_TICKS_ALL – 全てのティック。フラグ値はCOPY_TICKS列挙体で説明されています。名前なし必須パラメータ。

戻り値

ティックを名前付きのtime、bid、ask、last、flags列を持つNumPy配列として返します。「flags」値はTICK_FLAG列挙体からのフラグの組み合わせです。エラーの場合はNoneを返します。エラーに関する情報はlast_error()を使用して取得できます。

注意事項

詳細はCopyTicks関数をご覧ください。

「datetime」オブジェクトを作成する際にPythonがローカルな時間帯を使用する一方、MetaTrader 5ではティックとバー開始時間はUTC時間帯(シフトなし)で保存されます。よって。時間を使用する関数を実行するためには「datetime」はUTC時間で作成されるべきです。MetaTrader 5ターミナルから受信したデータにはUTC時間があります。

COPY_TICKSは、copy_ticks_from()およびcopy_ticks_range()関数を使用してリクエストできるティックの種類を定義します。

ID	説明
COPY_TICKS_ALL	すべてのティック
COPY_TICKS_INFO	買い気配値および/または売り気配値の変化を含むティック

© 2000–2025, MetaQuotes Ltd.

ID	説明
COPY_TICKS_TRADE	Lastおよび/またはVolume価格の変化を含むティック

TICK_FLAG はティックの可能なフラグを定義します。これらのフラグはcopy_ticks_from()およびcopy_ticks_range()関数で取得されるティックを説明するために使用されます。

ID	説明
TICK_FLAG_BID	売り気配値が変化した
TICK_FLAG_ASK	買い気配値が変化した
TICK_FLAG_LAST	Last価格が変化した
TICK_FLAG_VOLUME	出来高が変化した
TICK_FLAG_BUY	最後の買値が変化した
TICK_FLAG_SELL	最後の売値が変化した

例：

```
from datetime import datetime
import MetaTrader5 as mt5
# MetaTrader 5パッケージについてのデータを表示する
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# 表形式で得られたデータを表示するための「pandas」モジュールをインポートする
import pandas as pd
pd.set_option('display.max_columns', 500) # 表示される列の数
pd.set_option('display.width', 1500)      # 表示する表の最高幅
# 時間帯を使用するためのpytzモジュールをインポートする
import pytz

# MetaTrader 5ターミナルとの接続を確立する
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# タイムゾーンをUTCに設定する
timezone = pytz.timezone("Etc/UTC")
# ローカルタイムゾーンオフセットの実装を回避するために、UTCタイムゾーンで「datetime」オブジェクトを作成する
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
# 10.01.2019 UTCから開始して100 000 EURUSDティックをリクエストする
ticks = mt5.copy_ticks_from("EURUSD", utc_from, 100000, mt5.COPY_TICKS_ALL)
print("Ticks received:",len(ticks))

# MetaTrader 5ターミナルへの接続をシャットダウンする
mt5.shutdown()
```

© 2000-2025, MetaQuotes Ltd.

```
# 各ティックのデータを新しい行に表示する
print("Display obtained ticks 'as is'")
count = 0
for tick in ticks:
    count+=1
    print(tick)
    if count >= 10:
        break

# 取得したデータからDataFrameを作成する
ticks_frame = pd.DataFrame(ticks)
# 秒での時間をdatetime形式に変換する
ticks_frame['time']=pd.to_datetime(ticks_frame['time'], unit='s')

# データを表示する
print("\nDisplay dataframe with ticks")
print(ticks_frame.head(10))
```

結果:

```
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Ticks received: 100000
Display obtained ticks 'as is'
(1578614400, 1.11051, 1.11069, 0., 0, 1578614400987, 134, 0.)
(1578614402, 1.11049, 1.11067, 0., 0, 1578614402025, 134, 0.)
(1578614404, 1.1105, 1.11066, 0., 0, 1578614404057, 134, 0.)
(1578614404, 1.11049, 1.11067, 0., 0, 1578614404344, 134, 0.)
(1578614412, 1.11052, 1.11064, 0., 0, 1578614412106, 134, 0.)
(1578614418, 1.11039, 1.11051, 0., 0, 1578614418265, 134, 0.)
(1578614418, 1.1104, 1.1105, 0., 0, 1578614418905, 134, 0.)
(1578614419, 1.11039, 1.11051, 0., 0, 1578614419519, 134, 0.)
(1578614456, 1.11037, 1.11065, 0., 0, 1578614456011, 134, 0.)
(1578614456, 1.11039, 1.11051, 0., 0, 1578614456015, 134, 0.)

Display dataframe with ticks
                 time      bid      ask  last  volume       time_msc  flags
volume_real
0 2020-01-10 00:00:00  1.11051  1.11069   0.0       0  1578614400987    134
 0.0
1 2020-01-10 00:00:02  1.11049  1.11067   0.0       0  1578614402025    134
 0.0
2 2020-01-10 00:00:04  1.11050  1.11066   0.0       0  1578614404057    134
 0.0
3 2020-01-10 00:00:04  1.11049  1.11067   0.0       0  1578614404344    134
 0.0
4 2020-01-10 00:00:12  1.11052  1.11064   0.0       0  1578614412106    134
 0.0
5 2020-01-10 00:00:18  1.11039  1.11051   0.0       0  1578614418265    134
 0.0
6 2020-01-10 00:00:18  1.11040  1.11050   0.0       0  1578614418905    134
 0.0
```

© 2000–2025, MetaQuotes Ltd.

```
7 2020-01-10 00:00:19  1.11039  1.11051   0.0       0  1578614419519    134
 0.0
8 2020-01-10 00:00:56  1.11037  1.11065   0.0       0  1578614456011    134
 0.0
9 2020-01-10 00:00:56  1.11039  1.11051   0.0       0  1578614456015    134
 0.0
```

参照

CopyRates、copy_rates_from_pos、copy_rates_range、copy_ticks_from、copy_ticks_range

© 2000–2025, MetaQuotes Ltd.

copy_ticks_range

指定された期間のティックをMetaTrader 5ターミナルから取得します。

```
copy_ticks_range(
   symbol,        // 銘柄名
   date_from,     // ティックがリクエストされている最初の日
   date_to,       // ティックがリクエストされている最後の日
   flags          // リクエストされたティックの種類を定義するためのフラグの組み合わせ
   )
```

パラメータ

symbol

[in] 金融商品名（例: EURUSD）。名前なし必須パラメータ。

date_from

[in] ティックがリクエストされている最初の日。「datetime」オブジェクトまたは1970年1月1日からの経過秒数として設定されます。名前なし必須パラメータ。

date_to

[in] ティックがリクエストされている最後の日。「datetime」オブジェクトまたは1970年1月1日からの経過秒数として設定されます。名前なし必須パラメータ。

フラグ

[in] リクエストされたティックのタイプを定義するフラグCOPY_TICKS_INFO – 買値と売値の変更でのティック、COPY_TICKS_TRADE –ラストと数量の変更でのティック、COPY_TICKS_ALL – 全てのティック。フラグ値はCOPY_TICKS列挙体で説明されています。名前なし必須パラメータ。

戻り値

ティックを名前付きのtime、bid、ask、last、flags列を持つNumPy配列として返します。「flags」値はTICK_FLAG列挙体からのフラグの組み合わせです。エラーの場合はNoneを返します。エラーに関する情報はlast_error()を使用して取得できます。

注意事項

詳細はCopyTicks関数をご覧ください。

「datetime」オブジェクトを作成する際にPythonがローカルな時間帯を使用する一方、MetaTrader 5ではティックとバー開始時間はUTC時間帯（シフトなし）で保存されます。よって。時間を使用する関数を実行するためには「datetime」はUTC時間で作成されるべきです。MetaTrader 5から取得されたデータはUTC時間を持ちますが、Pythonは、出力する際に再度ローカル時間シフトを適用します。よって、取得されたデータは表示する前に修正されるべきです。

例：

```
from datetime import datetime
import MetaTrader5 as mt5
# MetaTrader 5パッケージについてのデータを表示する
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
```

© 2000–2025, MetaQuotes Ltd.

```
# テーブル形式で取得されたデータを表示するための「pandas」モジュールをインポートする
import pandas as pd
pd.set_option('display.max_columns', 500) # 表示する列の数
pd.set_option('display.width', 1500)      # 表示するテーブルの最大幅
# タイムゾーンを使用するためのpytzモジュールをインポートする
import pytz

# MetaTrader 5ターミナルとの接続を確立する
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# タイムゾーンをUTCに設定する
timezone = pytz.timezone("Etc/UTC")
# ローカルタイムゾーンオフセットの実装を回避するために、UTCタイムゾーンで「datetime」オブジェクトを作成する
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
utc_to = datetime(2020, 1, 11, tzinfo=timezone)
# 11.01.2020 - 11.01.2020からのAUDUSDティックをリクエストする
ticks = mt5.copy_ticks_range("AUDUSD", utc_from, utc_to, mt5.COPY_TICKS_ALL)
print("Ticks received:",len(ticks))

# MetaTrader 5ターミナルへの接続をシャットダウンする
mt5.shutdown()

# 各ティックのデータを新しい行に表示する
print("Display obtained ticks 'as is'")
count = 0
for tick in ticks:
    count+=1
    print(tick)
    if count >= 10:
        break

# 取得したデータからDataFrameを作成する
ticks_frame = pd.DataFrame(ticks)
# 秒での時間をdatetime形式に変換する
ticks_frame['time']=pd.to_datetime(ticks_frame['time'], unit='s')

# データを表示する
print("\nDisplay dataframe with ticks")
print(ticks_frame.head(10))
```

結果:

```
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Ticks received: 37008
Display obtained ticks 'as is'
(1578614400, 0.68577, 0.68594, 0., 0, 1578614400820, 134, 0.)
(1578614401, 0.68578, 0.68594, 0., 0, 1578614401128, 130, 0.)
(1578614401, 0.68575, 0.68594, 0., 0, 1578614401128, 130, 0.)
(1578614411, 0.68576, 0.68594, 0., 0, 1578614411388, 130, 0.)
(1578614411, 0.68575, 0.68594, 0., 0, 1578614411560, 130, 0.)
(1578614414, 0.68576, 0.68595, 0., 0, 1578614414973, 134, 0.)
(1578614430, 0.68576, 0.68594, 0., 0, 1578614430188, 4, 0.)
(1578614450, 0.68576, 0.68595, 0., 0, 1578614450408, 4, 0.)
```

© 2000-2025, MetaQuotes Ltd.

```
(1578614450, 0.68576, 0.68594, 0., 0, 1578614450519, 4, 0.)
(1578614456, 0.68575, 0.68594, 0., 0, 1578614456363, 130, 0.)

Display dataframe with ticks
                 time      bid      ask  last  volume       time_msc  flags
volume_real
0 2020-01-10 00:00:00  0.68577  0.68594   0.0       0  1578614400820    134
 0.0
1 2020-01-10 00:00:01  0.68578  0.68594   0.0       0  1578614401128    130
 0.0
2 2020-01-10 00:00:01  0.68575  0.68594   0.0       0  1578614401128    130
 0.0
3 2020-01-10 00:00:11  0.68576  0.68594   0.0       0  1578614411388    130
 0.0
4 2020-01-10 00:00:11  0.68575  0.68594   0.0       0  1578614411560    130
 0.0
5 2020-01-10 00:00:14  0.68576  0.68595   0.0       0  1578614414973    134
 0.0
6 2020-01-10 00:00:30  0.68576  0.68594   0.0       0  1578614430188      4
 0.0
7 2020-01-10 00:00:50  0.68576  0.68595   0.0       0  1578614450408      4
 0.0
8 2020-01-10 00:00:50  0.68576  0.68594   0.0       0  1578614450519      4
 0.0
9 2020-01-10 00:00:56  0.68575  0.68594   0.0       0  1578614456363    130
 0.0
```

参照

CopyRates、copy_rates_from_pos、copy_rates_range、copy_ticks_from、copy_ticks_range

© 2000–2025, MetaQuotes Ltd.

orders_total

アクティブな注文の数を取得します

```
orders_total()
```

戻り値

整数値。

注意事項

この関数はOrdersTotalに似ています。

例：

```
import MetaTrader5 as mt5
# MetaTrader 5パッケージについてのデータを表示する
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5ターミナルとの接続を確立する
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# アクティブ注文の存在を確認
orders=mt5.orders_total()
if orders>0:
    print("Total orders=",orders)
else:
    print("Orders not found")

# MetaTrader 5ターミナルへの接続をシャットダウンする
mt5.shutdown()
```

参照

orders_get、positions_total

© 2000–2025, MetaQuotes Ltd.

orders_get

アクティブな注文を取得します(銘柄またはチケットでフィルターする機能あり)。3つの呼び出しオプションがあります。

パラメータなしでの呼び出し:すべての銘柄のアクティブ注文を返します。

```
orders_get()
```

アクティブ注文を受け取る銘柄を指定した呼び出し:

```
orders_get(
   symbol="SYMBOL"       // 銘柄名
)
```

アクティブ注文を受け取る銘柄グループを指定した呼び出し:

```
orders_get(
   group="GROUP"         // 注文を銘柄で選択するためのフィルター
)
```

注文チケットを指定した呼び出し:

```
orders_get(
   ticket=TICKET         // チケット
)
```

symbol="SYMBOL"

[in] Symbol name. 名前付きオプションパラメータ。銘柄が指定されている場合は、*ticket*パラメータは無視されます。

group="GROUP"

[in] 必要な銘柄のグループを配置するためのフィルター。名前付きオプションパラメータ。グループが指定されている場合、関数は銘柄名の指定された基準を満たすアクティブ注文のみを返します。

ticket=TICKET

[in] 注文チケット(ORDER_TICKET)。名前の付いたオプションパラメータ。

戻り値

名前付きタプル構造(namedtuple)の形式で情報を返します。エラーの場合はNoneを返します。エラーに関する情報はlast_error()を使用して取得できます。

注意事項

この関数を使用すると、OrdersTotalおよびOrderSelect関数を呼び出すのに似た形で1回の呼び出しですべてのアクティブ注文を受け取ることができます。

*group*パラメータを使用すると、注文を銘柄で並べ替えることができます。「*」は文字列の先頭と末尾で使用できます。

*group*パラメータには、複数のコンマ区切りの条件を含めることができます。条件は「*」を使用してマスクとして設定できます。論理否定記号「!」は除外に使用できます。すべての条件は順番に適用されます。つまり、グループに含める条件を最初に指定してから、除外条件を指定する必要があります。例えば、「*group="*, !EUR"*」では、全銘柄の注文を最初に選択してから、銘柄名に「EUR」が含まれる注文を後で除外します。

© 2000-2025, MetaQuotes Ltd.

例：

```
import MetaTrader5 as mt5
import pandas as pd
pd.set_option('display.max_columns', 500) # 表示される列の数
pd.set_option('display.width', 1500)      # 表示する表の最高幅
# MetaTrader 5パッケージについてのデータを表示する
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print()
# MetaTrader 5ターミナルとの接続を確立する
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# GBPUSDのアクティブ注文のデータを表示する
orders=mt5.orders_get(symbol="GBPUSD")
if orders is None:
    print("No orders on GBPUSD, error code={}".format(mt5.last_error()))
else:
    print("Total orders on GBPUSD:",len(orders))
    # display all active orders
    for order in orders:
        print(order)
print()

# *GBP*
gbp_orders=mt5.orders_get(group="*GBP*")
if gbp_orders is None:
    print("No orders with group=\"*GBP*\", error code={}".format(mt5.last_error()))
else:
    print("orders_get(group=\"*GBP*\")={}".format(len(gbp_orders)))
    # pandas.DataFrameを使用してこれらの注文を表として表示する
    df=pd.DataFrame(list(gbp_orders),columns=gbp_orders[0]._asdict().keys())
    df.drop(['time_done', 'time_done_msc', 'position_id', 'position_by_id',
'reason', 'volume_initial', 'price_stoplimit'], axis=1, inplace=True)
    df['time_setup'] = pd.to_datetime(df['time_setup'], unit='s')
    print(df)

# MetaTrader 5ターミナルへの接続をシャットダウンする
mt5.shutdown()
```

結果:

```
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Total orders on GBPUSD: 2
TradeOrder(ticket=554733548, time_setup=1585153667, time_setup_msc=1585153667718,
time_done=0, time_done_msc=0, time_expiration=0, type=3, type_time=0, ...
```

© 2000-2025, MetaQuotes Ltd.

```
TradeOrder(ticket=554733621, time_setup=1585153671, time_setup_msc=1585153671419,
time_done=0, time_done_msc=0, time_expiration=0, type=2, type_time=0, ...


orders_get(group="*GBP*")=4
      ticket           time_setup  time_setup_msc  time_expiration  type  type_time
type_filling  state  magic  volume_current  price_open   sl   tp  price_current
symbol comment external_id
0  554733548 2020-03-25 16:27:47   1585153667718                0     3          0
           2      1      0             0.2     1.25379  0.0  0.0        1.16803
GBPUSD
1  554733621 2020-03-25 16:27:51   1585153671419                0     2          0
           2      1      0             0.2     1.14370  0.0  0.0        1.16815
GBPUSD
2  554746664 2020-03-25 16:38:14   1585154294401                0     3          0
           2      1      0             0.2     0.93851  0.0  0.0        0.92428
EURGBP
3  554746710 2020-03-25 16:38:17   1585154297022                0     2          0
           2      1      0             0.2     0.90527  0.0  0.0        0.92449
EURGBP
```

参照

orders_total、positions_get

© 2000–2025, MetaQuotes Ltd.

order_calc_margin

指定された取引操作を実行するための証拠金を口座通貨で返します。

```
order_calc_margin(
   action,      // 注文の種類(ORDER_TYPE_BUYまたはORDER_TYPE_SELL)
   symbol,      // 銘柄名
   volume,      // ボリューム
   price        // 始値
   )
```

パラメータ

action

[in] 注文の種類(ORDER_TYPE列挙からの値)名前なし必須パラメータ。

symbol

[in] 金融商品名。名前なし必須パラメータ。

volume

[in] 取引操作量。名前なし必須パラメータ。

price

[in] 始値。名前なし必須パラメータ。

戻り値

成功した場合は実際の値、それ以外の場合はなし。エラーコードは、last_error()を使用して取得できます。

注意事項

この関数を使用すると、現在の未決注文やポジションを考慮せずに、現在の口座および現在の市場環境で指定された注文の種類に必要な証拠金を見積もることができます。この関数はOrderCalcMarginに似ています。

ORDER_TYPE

ID	説明
ORDER_TYPE_BUY	成行買い注文
ORDER_TYPE_SELL	成行売り注文
ORDER_TYPE_BUY_LIMIT	買い指値注文
ORDER_TYPE_SELL_LIMIT	売り指値注文
ORDER_TYPE_BUY_STOP	買い逆指値注文
ORDER_TYPE_SELL_STOP	売り逆指値注文
ORDER_TYPE_BUY_STOP_LIMIT	注文価格に達すると、未決の買い指値注文がStopLimit価格で出されます

© 2000-2025, MetaQuotes Ltd.

ID	説明
ORDER_TYPE_SELL_STOP_LIMIT	注文価格に達すると、未決の売り指値注文がStopLimit価格で出されます
ORDER_TYPE_CLOSE_BY	反対のポジションでポジションを決済するための注文

例：

```
import MetaTrader5 as mt5
# MetaTrader 5パッケージにデータを表示する
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5ターミナルへの接続を確立する
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# 口座通貨を取得する
account_currency=mt5.account_info().currency
print("Account currency:",account_currency)

# 銘柄リストを整理する
symbols=("EURUSD","GBPUSD","USDJPY", "USDCHF","EURJPY","GBPJPY")
print("Symbols to check margin:", symbols)
action=mt5.ORDER_TYPE_BUY
lot=0.1
for symbol in symbols:
    symbol_info=mt5.symbol_info(symbol)
    if symbol_info is None:
        print(symbol,"not found, skipped")
        continue
    if not symbol_info.visible:
        print(symbol, "is not visible, trying to switch on")
        if not mt5.symbol_select(symbol,True):
            print("symbol_select({}) failed, skipped",symbol)
            continue
    ask=mt5.symbol_info_tick(symbol).ask
    margin=mt5.order_calc_margin(action,symbol,lot,ask)
    if margin != None:
        print("   {} buy {} lot margin: {}
{}".format(symbol,lot,margin,account_currency));
    else:
        print("order_calc_margin failed: , error code =", mt5.last_error())

# MetaTrader 5ターミナルへの接続をシャットダウンする
```

© 2000-2025, MetaQuotes Ltd.

```
mt5.shutdown()


Result:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Account currency: USD

Symbols to check margin: ('EURUSD', 'GBPUSD', 'USDJPY', 'USDCHF', 'EURJPY',
'GBPJPY')
   EURUSD buy 0.1 lot margin: 109.91 USD
   GBPUSD buy 0.1 lot margin: 122.73 USD
   USDJPY buy 0.1 lot margin: 100.0 USD
   USDCHF buy 0.1 lot margin: 100.0 USD
   EURJPY buy 0.1 lot margin: 109.91 USD
   GBPJPY buy 0.1 lot margin: 122.73 USD
```

参照

order_calc_profit、order_check

© 2000–2025, MetaQuotes Ltd.

order_calc_profit

指定された取引操作の利益を口座通貨で返します。

```
order_calc_profit(
   action,           // 注文の種類( ORDER_TYPE_ BUYまたはORDER_TYPE_ SELL)
   symbol,           // 銘柄名
   volume,           // ボリューム
   price_open,       // 始値
   price_close       // 終値
   );
```

パラメータ

action

[in] ORDER_TYPE列挙のいずれか(ORDER_TYPE_ BUYまたはORDER_TYPE_ SELL)の値を持つ注文の種類名前なし必須パラメータ。

symbol

[in] 金融商品名。名前なし必須パラメータ。

volume

[in] 取引操作量。名前なし必須パラメータ。

price_open

[in] 始値。名前なし必須パラメータ。

price_close

[in] 終値。名前なし必須パラメータ。

戻り値

成功した場合は実際の値、それ以外の場合はなし。エラーコードは、last_error()を使用して取得できます。

注意事項

この関数を使用すると、現在の口座および現在の取引環境での取引操作結果を推定できます。この関数はOrderCalcProfitに似ています。

例：

```
import MetaTrader5 as mt5
# MetaTrader 5パッケージにデータを表示する
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5ターミナルへの接続を確立する
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
```

© 2000-2025, MetaQuotes Ltd.

```
    quit()

# 口座通貨を取得する
account_currency=mt5.account_info().currency
print("Account currency:",account_currency)

# 銘柄リストを整理する
symbols = ("EURUSD","GBPUSD","USDJPY")
print("Symbols to check margin:", symbols)
# 売買による利益を見積もる(
lot=1.0
distance=300
、symbolsの銘柄):
    symbol_info=mt5.symbol_info(symbol)
    if symbol_info is None:
        print(symbol,"not found, skipped")
        continue
    if not symbol_info.visible:
        print(symbol, "is not visible, trying to switch on")
        if not mt5.symbol_select(symbol,True):
            print("symbol_select({}}) failed, skipped",symbol)
            continue
    point=mt5.symbol_info(symbol).point
    symbol_tick=mt5.symbol_info_tick(symbol)
    ask=symbol_tick.ask
    bid=symbol_tick.bid
    buy_profit=mt5.order_calc_profit
(mt5.ORDER_TYPE_BUY,symbol,lot,ask,ask+distance*point)
    if buy_profit!=None:
        print("   buy {} {} lot: profit on {} points => {}
{}".format(symbol,lot,distance,buy_profit,account_currency));
    else:
        print("order_calc_profit(ORDER_TYPE_BUY) failed, error code
=",mt5.last_error())
    sell_profit=mt5.order_calc_profit(mt5.ORDER_TYPE_SELL,symbol,lot,bid,bid-
distance*point)
    if sell_profit!=None:
        print("   sell {} {} lots: profit on {} points => {}
{}".format(symbol,lot,distance,sell_profit,account_currency));
    else:
        print("order_calc_profit(ORDER_TYPE_SELL) failed, error code
=",mt5.last_error())
    print()

# MetaTrader 5ターミナルへの接続をシャットダウンする
mt5.shutdown()


Result:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Account currency: USD
Symbols to check margin: ('EURUSD', 'GBPUSD', 'USDJPY')
   buy EURUSD 1.0 lot: profit on 300 points => 300.0 USD
   sell EURUSD 1.0 lot: profit on 300 points => 300.0 USD
```

© 2000-2025, MetaQuotes Ltd.

```
buy GBPUSD 1.0 lot: profit on 300 points => 300.0 USD
sell GBPUSD 1.0 lot: profit on 300 points => 300.0 USD

buy USDJPY 1.0 lot: profit on 300 points => 276.54 USD
sell USDJPY 1.0 lot: profit on 300 points => 278.09 USD
```

参照

order_calc_margin、order_check

© 2000–2025, MetaQuotes Ltd.

order_check

必要な取引操作を実行するために充分な資金があるかを確認します。確認の結果はMqlTradeCheckResult構造体として返されます。

```
order_check(
   request       // リクエスト構造体
   );
```

パラメータ

request

[in] 必要な取引アクションを記述するMqlTradeRequest型の構造体。名前なし必須パラメータ。リクエストの記入例と列挙内容を以下に示します。

戻り値

MqlTradeCheckResult構造体としての確認の結果。回答の*request*フィールドには、order_check()に渡される取引リクエストの構造体が含まれています。

注意事項

リクエストが正常に送信されても、**リクエストされた取引操作が正常に実行されるとは限りません**。order_check関数はOrderCheckに似ています。

TRADE_REQUEST_ACTIONS

ID	説明
TRADE_ACTION_DEAL	指定されたパラメータを使用してインスタント取引を注文します（成行注文を設定します）
TRADE_ACTION_PENDING	指定された条件（未決注文）で取引を実行するために取引注文を出します。
TRADE_ACTION_SLTP	ポジションの決済逆指値と決済指値を変更する
TRADE_ACTION_MODIFY	以前に行った取引注文のパラメータを変更する
TRADE_ACTION_REMOVE	以前に出した未決注文を削除する
TRADE_ACTION_CLOSE_BY	反対のポジションでポジションを決済する

ORDER_TYPE_FILLING

ID	説明
ORDER_FILLING_FOK	この実行ポリシーは、注文が指定された量でのみ満たすことが出来ることを意味します。金融製品の必要量が市場で現在利用できない場合、注文は実行されません。希望するボリュームは、いくつかの利用可能なオファーで構成できます。
ORDER_FILLING_IOC	注文で指定されたボリューム内で市場で利用可能な最大ボリュームで取引を実行するための合意。リクエストを完全に満たすことができない場合、利用可能なボリュームの注文が実行され、残りのボリュームはキャンセルされます。

© 2000–2025, MetaQuotes Ltd.

ID	説明
ORDER_FILLING_RETURN	このポリシーは、「市場」または「取引所」実行モードを持つ銘柄に対してのみ、成行注文(ORDER_TYPE_BUY及びORDER_TYPE_SELL)、指値注文及びストップリミット注文(ORDER_TYPE_BUY_LIMIT、ORDER_TYPE_SELL_LIMIT、ORDER_TYPE_BUY_STOP_LIMIT、ORDER_TYPE_SELL_STOP_LIMIT)の実行に使用されます。部分的に充填された場合、残りの数量を持つ市場や指値注文はキャンセルされず、さらに処理されます。ORDER_TYPE_BUY_STOP_LIMIT及びORDER_TYPE_SELL_STOP_LIMIT注文の発動のためには、ORDER_FILLING_RETURN実行型と対応する指値注文ORDER_TYPE_BUY_LIMIT／ ORDER_TYPE_SELL_LIMITが作成されます。

ORDER_TYPE_TIME

ID	説明
ORDER_TIME_GTC	注文は手動でキャンセルされるまでキューに残ります。
ORDER_TIME_DAY	注文は現在の取引日にのみアクティブです。
ORDER_TIME_SPECIFIED	注文は指定された日付までアクティブです。
ORDER_TIME_SPECIFIED_DAY	注文は、指定された日の23:59:59までアクティブです。この時間が取引セッションの外にあると思われる場合、有効期限は最も近い取引時間に処理されます。

例：

```
import MetaTrader5 as mt5
# MetaTrader 5パッケージにデータを表示する
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5ターミナルへの接続を確立する
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# 口座通貨を取得する
account_currency=mt5.account_info().currency
print("Account currency:",account_currency)

# リクエスト構造体を準備する
symbol="USDJPY"
symbol_info = mt5.symbol_info(symbol)
if symbol_info is None:
    print(symbol, "not found, can not call order_check()")
    mt5.shutdown()
    quit()

# 銘柄が「気配値表示」似ない場合は追加する
```

© 2000–2025, MetaQuotes Ltd.

```
if not symbol_info.visible:
    print(symbol, "is not visible, trying to switch on")
    if not mt5.symbol_select(symbol,True):
        print("symbol_select({}}) failed, exit",symbol)
        mt5.shutdown()
        quit()

# リクエストを準備する
point=mt5.symbol_info(symbol).point
request = {
    "action": mt5.TRADE_ACTION_DEAL,
    "symbol": symbol,
    "volume": 1.0,
    "type": mt5.ORDER_TYPE_BUY,
    "price": mt5.symbol_info_tick(symbol).ask,
    "sl": mt5.symbol_info_tick(symbol).ask-100*point,
    "tp": mt5.symbol_info_tick(symbol).ask+100*point,
    "deviation": 10,
    "magic": 234000,
    "comment": "python script",
    "type_time": mt5.ORDER_TIME_GTC,
    "type_filling": mt5.ORDER_FILLING_RETURN,
}

# チェックを実行し、結果を「そのまま」表示する
result = mt5.order_check(request)
print(result);
# 結果をディクショナリとしてリクエストし、要素ごとに表示する
result_dict=result._asdict()
for field in result_dict.keys():
    print("   {}={}".format(field,result_dict[field]))
    # if this is a trading request structure, display it element by element as well
    if field=="request":
        traderequest_dict=result_dict[field]._asdict()
        for tradereq_filed in traderequest_dict:
            print("       traderequest: {}
={}".format(tradereq_filed,traderequest_dict[tradereq_filed]))

# MetaTrader 5ターミナルへの接続をシャットダウンする
mt5.shutdown()


Result:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Account currency: USD
   retcode=0
   balance=101300.53
   equity=68319.53
   profit=-32981.0
   margin=51193.67
   margin_free=17125.86
   margin_level=133.45308121101692
```

© 2000–2025, MetaQuotes Ltd.

```
   comment=Done
   request=TradeRequest(action=1, magic=234000, order=0, symbol='USDJPY',
volume=1.0, ...
        traderequest: action=1
        traderequest: magic=234000
        traderequest: order=0
        traderequest: symbol=USDJPY
        traderequest: volume=1.0
        traderequest: price=108.081
        traderequest: stoplimit=0.0
        traderequest: sl=107.98100000000001
        traderequest: tp=108.181
        traderequest: deviation=10
        traderequest: type=0
        traderequest: type_filling=2
        traderequest: type_time=0
        traderequest: expiration=0
        traderequest: comment=python script
        traderequest: position=0
        traderequest: position_by=0
```

参照

order_send、OrderCheck、取引操作の種類、取引リクエスト構造体、取引リクエストのチェック結果の構造体、取引リクエスト結果の構造体

© 2000–2025, MetaQuotes Ltd.

order_send

取引操作を実行するリクエストをターミナルから取引サーバに送信します。この関数はOrderSendに似ています。

```
order_send(
   request       // リクエスト構造体
   );
```

パラメータ

request

[in] 必要な取引アクションを記述するMqlTradeRequest型の構造体。名前なし必須パラメータ。リクエストの記入例と列挙内容を以下に示します。

戻り値

MqlTradeResult構造体としての実行結果。回答の*request*フィールドには、order_send()に渡される取引リクエストの構造体が含まれています。エラーに関する情報はlast_error()を使用して取得できます。

MqlTradeRequest取引リクエスト構造体

フィールド	説明
action	取引操作の種類。値はTRADE_REQUEST_ACTIONS列挙体のうちの1つです。
magic	EAのID。取引注文の分析処理を調整できるようにします。各EAは、取引リクエストを送信するときに一意のIDを設定できます。
order	注文チケット。未決注文を変更するために必要です。
symbol	注文が行われた取引商品の名前。注文を変更する場合と決済する場合は不要。
volume	ロット単位でのリクエストされた取引量。取引の出来高は注文実行の種類によって異なります。
price	注文実行価格。「市場執行」(SYMBOL_TRADE_EXECUTION_MARKET)タイプの商品がTRADE_ACTION_DEAL型の成行注文の場合、価格は設定されません。
stoplimit	未決指値注文の価格は、価格が「price」の値に達したときに設定されます(この条件は必須です)。未決注文はその瞬間まで取引システムに渡されません。
sl	逆指値注文は、価格が好ましくない方向に動いたときに発動します。
tp	価格が好ましい方向に動いたときに指値注文が発動する価格。
deviation	リクエストされた価格からの最大許容偏差(ポイント単位)。
type	注文の種類。値はORDER_TYPE列挙体のうちの1つです。
type_filling	注文の種類。値はORDER_TYPE_FILLING値のうちの1つです。
type_time	注文有効期限の種類。値はORDER_TYPE_TIME値のうちの1つです。
expiration	未決注文の有効期限(TIME_SPECIFIED型注文)。
comment	注文コメント。

© 2000-2025, MetaQuotes Ltd.

フィールド	説明
position	ポジションチケット。明確に識別するために、ポジションを変更および決済するときに入力します。通常は、ポジションを開いた注文のチケットと同じです。
position_by	反対方向のポジションのチケット。反対のポジション(同じ銘柄で反対方向に開かれているポジション)でポジションを決済するときに使用されます。

注意事項

取引リクエストは、取引サーバ上で複数の検証段階を踏みます。まず、必要なすべてのリクエストフィールドの有効性が確認されます。エラーがない場合、サーバはさらに処理するために注文を受け入れます。取引操作の実行の詳細については、OrderSend関数の説明を参照してください。

例：

```
import time
import MetaTrader5 as mt5

# MetaTrader 5パッケージについてのデータを表示する
print("MetaTrader5 package author: ", mt5.__author__)
print("MetaTrader5 package version: ", mt5.__version__)

# MetaTrader 5ターミナルとの接続を確立する
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# 買いリクエスト構造体を準備する
symbol = "USDJPY"
symbol_info = mt5.symbol_info(symbol)
if symbol_info is None:
    print(symbol, "not found, can not call order_check()")
    mt5.shutdown()
    quit()

# 銘柄が「気配値表示」似ない場合は追加する
if not symbol_info.visible:
    print(symbol, "is not visible, trying to switch on")
    if not mt5.symbol_select(symbol,True):
        print("symbol_select({}) failed, exit",symbol)
        mt5.shutdown()
        quit()

lot = 0.1
point = mt5.symbol_info(symbol).point
price = mt5.symbol_info_tick(symbol).ask
deviation = 20
request = {
    "action": mt5.TRADE_ACTION_DEAL,
```

© 2000–2025, MetaQuotes Ltd.

```
    "symbol": symbol,
    "volume": lot,
    "type": mt5.ORDER_TYPE_BUY,
    "price": price,
    "sl": price - 100 * point,
    "tp": price + 100 * point,
    "deviation": deviation,
    "magic": 234000,
    "comment": "python script open",
    "type_time": mt5.ORDER_TIME_GTC,
    "type_filling": mt5.ORDER_FILLING_RETURN,
}

# 取引リクエストを送信する
result = mt5.order_send(request)
# 実行結果を確認する
print("1. order_send(): by {} {} lots at {} with deviation={}
points".format(symbol,lot,price,deviation));
if result.retcode != mt5.TRADE_RETCODE_DONE:
    print("2. order_send failed, retcode={}".format(result.retcode))
   # 結果をディクショナリとしてリクエストし、要素ごとに表示する
    result_dict=result._asdict()
    for field in result_dict.keys():
        print("   {}={}".format(field,result_dict[field]))
        # これが取引リクエスト構造体の場合は要素ごとに表示する
        if field=="request":
            traderequest_dict=result_dict[field]._asdict()
            for tradereq_filed in traderequest_dict:
                print("       traderequest: {}
={}".format(tradereq_filed,traderequest_dict[tradereq_filed]))
    print("shutdown() and quit")
    mt5.shutdown()
    quit()

print("2. order_send done, ", result)
print("   opened position with POSITION_TICKET={}".format(result.order))
print("   sleep 2 seconds before closing position #{}".format(result.order))
time.sleep(2)
# 決済リクエストを作成する
position_id=result.order
price=mt5.symbol_info_tick(symbol).bid
deviation=20
request={
    "action": mt5.TRADE_ACTION_DEAL,
    "symbol": symbol,
    "volume": lot,
    "type": mt5.ORDER_TYPE_SELL,
    "position": position_id,
    "price": price,
```

© 2000–2025, MetaQuotes Ltd.

```
    "deviation": deviation,
    "magic": 234000,
    "comment": "python script close",
    "type_time": mt5.ORDER_TIME_GTC,
    "type_filling": mt5.ORDER_FILLING_RETURN,
}
# 取引リクエストを送信する
result=mt5.order_send(request)
# 実行結果を確認する
print("3. close position #{}: sell {} {} lots at {} with deviation={}
points".format(position_id,symbol,lot,price,deviation));
if result.retcode != mt5.TRADE_RETCODE_DONE:
    print("4. order_send failed, retcode={}".format(result.retcode))
    print("   result",result)
else:
    print("4. position #{} closed, {}".format(position_id,result))
   # 結果をディクショナリとしてリクエストし、要素ごとに表示する
    result_dict=result._asdict()
    for field in result_dict.keys():
        print("   {}={}".format(field,result_dict[field]))
        # これが取引リクエスト構造体の場合は要素ごとに表示する
        if field=="request":
            traderequest_dict=result_dict[field]._asdict()
            for tradereq_filed in traderequest_dict:
                print("       traderequest: {}
={}".format(tradereq_filed,traderequest_dict[tradereq_filed]))

# MetaTrader 5ターミナルへの接続をシャットダウンする
mt5.shutdown()

Result
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29
1. order_send(): by USDJPY 0.1 lots at 108.023 with deviation=20 points
2. order_send done,  OrderSendResult(retcode=10009, deal=535084512, order=557416535,
volume=0.1, price=108.023, ...
   opened position with POSITION_TICKET=557416535
   sleep 2 seconds before closing position #557416535
3. close position #557416535: sell USDJPY 0.1 lots at 108.018 with deviation=20
points
4. position #557416535 closed, OrderSendResult(retcode=10009, deal=535084631,
order=557416654, volume=0.1, price=...
   retcode=10009
   deal=535084631
   order=557416654
   volume=0.1
   price=108.015
   bid=108.015
   ask=108.02
   comment=Request executed
```

© 2000–2025, MetaQuotes Ltd.

```
   request_id=55
   retcode_external=0
   request=TradeRequest(action=1, magic=234000, order=0, symbol='USDJPY',
volume=0.1, price=108.018, stoplimit=0.0, ...
       traderequest: action=1
       traderequest: magic=234000
       traderequest: order=0
       traderequest: symbol=USDJPY
       traderequest: volume=0.1
       traderequest: price=108.018
       traderequest: stoplimit=0.0
       traderequest: sl=0.0
       traderequest: tp=0.0
       traderequest: deviation=20
       traderequest: type=1
       traderequest: type_filling=2
       traderequest: type_time=0
       traderequest: expiration=0
       traderequest: comment=python script close
       traderequest: position=557416535
       traderequest: position_by=0
```

参照

order_check、OrderSend、取引操作の種類、取引リクエスト構造体、取引リクエストのチェック結果の構造体、取引リクエスト結果の構造体

© 2000–2025, MetaQuotes Ltd.

positions_total

ポジションの数を取得します。

```
positions_total()
```

戻り値

整数値。

注意事項

この関数はPositionsTotalに似ています。

例：

```
import MetaTrader5 as mt5
# MetaTrader 5パッケージについてのデータを表示する
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5ターミナルとの接続を確立する
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# ポジションの存在を確認する
positions_total=mt5.positions_total()
if positions_total>0:
    print("Total positions=",positions_total)
else:
    print("Positions not found")

# MetaTrader 5ターミナルへの接続をシャットダウンする
mt5.shutdown()
```

参照

positions_get、orders_total

© 2000–2025, MetaQuotes Ltd.

positions_get

ポジションを取得します(銘柄またはチケットでフィルターする機能あり)。3つの呼び出しオプションがあります。

パラメータなしでの呼び出し: すべてのポジションを返します。

```
positions_get()
```

ポジションを受け取る銘柄を指定した呼び出し:

```
positions_get(
   symbol="SYMBOL"       // 銘柄名
)
```

ポジションを受け取る銘柄グループを指定した呼び出し:

```
positions_get(
   group="GROUP"         // ポジションを銘柄で選択するためのフィルター
)
```

ポジションチケットを指定した呼び出し:

```
positions_get(
   ticket=TICKET         // チケット
)
```

パラメータ

symbol="SYMBOL"

[in] Symbol name. 名前付きオプションパラメータ。銘柄が指定されている場合は、*ticket*パラメータは無視されます。

group="GROUP"

[in] 必要な銘柄のグループを配置するためのフィルター。名前付きオプションパラメータ。グループが指定されている場合、関数は銘柄名の指定された基準を満たすポジションのみを返します。

ticket=TICKET

[in] ポジションチケット(POSITION_TICKET)。名前の付いたオプションパラメータ。

戻り値

名前付きタプル構造(namedtuple)の形式で情報を返します。エラーの場合はNoneを返します。エラーに関する情報はlast_error()を使用して取得できます。

注意事項

この関数を使用すると、PositionsTotalおよびPositionsSelect関数を呼び出すのに似た形で1回の呼び出しですべてのポジションを受け取ることができます。

*group*パラメータには、複数のコンマ区切りの条件を含めることができます。条件は「*」を使用してマスクとして設定できます。論理否定記号「!」は除外に使用できます。すべての条件は順番に適用されます。つまり、グループに含める条件を最初に指定してから、除外条件を指定する必要があります。例えば、「*group="*,!EUR"*」では、全銘柄のポジションを最初に選択してから、銘柄名に「EUR」が含まれるポジションを後で除外します。

例:

© 2000-2025, MetaQuotes Ltd.

```
import MetaTrader5 as mt5
import pandas as pd
pd.set_option('display.max_columns', 500) # 表示される列の数
pd.set_option('display.width', 1500)      # 表示する表の最高幅
# MetaTrader 5パッケージについてのデータを表示する
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print()
# MetaTrader 5ターミナルとの接続を確立する
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# USDCHFのポジションを取得する
positions=mt5.positions_get(symbol="USDCHF")
if positions==None:
    print("No positions on USDCHF, error code={}".format(mt5.last_error()))
elif len(positions)>0:
    print("Total positions on USDCHF =",len(positions))
    # すべてのポジションを表示する
    for position in positions:
        print(position)

# 名前に「*USD*」が含まれる銘柄のポジションリストを取得する
usd_positions=mt5.positions_get(group="*USD*")
if usd_positions==None:
    print("No positions with group=\"*USD*\", error
code={}".format(mt5.last_error()))
elif len(usd_positions)>0:
    print("positions_get(group=\"*USD*\")={}".format(len(usd_positions)))
    # pandas.DataFrameを使用してこれらのポジションを表として表示する
    df=pd.DataFrame(list(usd_positions),columns=usd_positions[0]._asdict().keys())
    df['time'] = pd.to_datetime(df['time'], unit='s')
    df.drop(['time_update', 'time_msc', 'time_update_msc', 'external_id'], axis=1,
inplace=True)
    print(df)

# MetaTrader 5ターミナルへの接続をシャットダウンする
mt5.shutdown()
```

結果:

```
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

positions_get(group="*USD*")=5
      ticket                time  type  magic  identifier  reason  volume
price_open       sl       tp  price_current  swap  profit  symbol comment
0  548297723 2020-03-18 15:00:55     1      0   548297723       3    0.01
1.09301  1.11490  1.06236        1.10104 -0.10   -8.03  EURUSD
```

© 2000-2025, MetaQuotes Ltd.

```
1  548655158 2020-03-18 20:31:26     0      0   548655158       3    0.01
1.08676  1.06107  1.12446        1.10099 -0.08   14.23  EURUSD
2  548663803 2020-03-18 20:40:04     0      0   548663803       3    0.01
1.08640  1.06351  1.11833        1.10099 -0.08   14.59  EURUSD
3  548847168 2020-03-19 01:10:05     0      0   548847168       3    0.01
1.09545  1.05524  1.15122        1.10099 -0.06    5.54  EURUSD
4  548847194 2020-03-19 01:10:07     0      0   548847194       3    0.02
1.09536  1.04478  1.16587        1.10099 -0.08   11.26  EURUSD
```

参照

positions_total、orders_get

© 2000–2025, MetaQuotes Ltd.

history_orders_total

指定された期間内の取引履歴の注文数を取得します。

```
history_orders_total(
   date_from,     // 注文がリクエストされている最初の日
   date_to        // 注文がリクエストされている最後の日
   )
```

パラメータ

date_from

[in] 注文がリクエストされている最初の日。「datetime」オブジェクトまたは1970年1月1日からの経過秒数として設定されます。名前なし必須パラメータ。

date_to

[in] 注文がリクエストされている最後の日。「datetime」オブジェクトまたは1970年1月1日からの経過秒数として設定されます。名前なし必須パラメータ。

戻り値

整数値。

注意事項

この関数はHistoryOrdersTotalに似ています。

例：

```
from datetime import datetime
import MetaTrader5 as mt5
# MetaTrader 5パッケージについてのデータを表示する
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5ターミナルとの接続を確立する
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# 履歴内の注文の数を取得する
from_date=datetime(2020,1,1)
to_date=datetime.now()
history_orders=mt5.history_orders_total(from_date, datetime.now())
if history_orders>0:
    print("Total history orders=",history_orders)
else:
    print("Orders not found in history")

# MetaTrader 5ターミナルへの接続をシャットダウンする
mt5.shutdown()
```

© 2000-2025, MetaQuotes Ltd.

参照

history_orders_get、history_deals_total

© 2000–2025, MetaQuotes Ltd.

history_orders_get

取引履歴から注文を取得します(チケットまたはポジションでフィルターする機能あり)。3つの呼び出しオプションがあります。

時間間隔を指定して呼び出します。指定された時間間隔内にあるすべての注文を返します。

```
history_orders_get(
   date_from,              // 注文がリクエストされている最初の日
   date_to,                // 注文がリクエストされている最後の日
   group="GROUP"           // 注文を銘柄で選択するためのフィルター
   )
```

注文チケットを指定した呼び出し:指定されたチケットを持つすべての注文を返します。

```
history_orders_get(
   ticket=TICKET           // 注文チケット
)
```

ポジションチケットを指定した呼び出し:ORDER_POSITION_IDプロパティに指定されたポジションチケットを持つすべての注文を返します。

```
history_orders_get(
   position=POSITION       // ポジションチケット
)
```

パラメータ

date_from

[in] 注文がリクエストされている最初の日。「datetime」オブジェクトまたは1970年1月1日からの経過秒数として設定されます。名前なし必須パラメータが最初に指定されます。

date_to

[in] 注文がリクエストされている最後の日。「datetime」オブジェクトまたは1970年1月1日からの経過秒数として設定されます。名前なし必須パラメータが2番目に指定されます。

group="GROUP"

[in] 必要な銘柄のグループを配置するためのフィルター。名前付きオプションパラメータ。グループが指定されている場合、関数は銘柄名の指定された基準を満たす注文のみを返します。

ticket=TICKET

[in] 受信される注文チケット。オプションパラメータ。指定されていない場合、フィルターは適用されません。

position=POSITION

[in] すべての注文が受信されるポジションのチケット(ORDER_POSITION_IDに格納)。オプションパラメータ。指定されていない場合、フィルターは適用されません。

戻り値

名前付きタプル構造(namedtuple)の形式で情報を返します。エラーの場合はNoneを返します。エラーに関する情報はlast_error()を使用して取得できます。

注意事項

© 2000-2025, MetaQuotes Ltd.

この関数を使用すると、指定した履歴期間内のすべての注文を、HistoryOrdersTotalおよびHistoryOrdersSelect関数を呼び出すのに似た形で受け取ることができます。

*group*パラメータには、複数のコンマ区切りの条件を含めることができます。条件は「*」を使用してマスクとして設定できます。論理否定記号「！」は除外に使用できます。すべての条件は順番に適用されます。つまり、グループに含める条件を最初に指定してから、除外条件を指定する必要があります。例えば、「*group*="*, !EUR"」では、全銘柄の取引を最初に選択してから、銘柄名に「EUR」が含まれる取引を後で除外します。

例：

```
from datetime import datetime
import MetaTrader5 as mt5
import pandas as pd
pd.set_option('display.max_columns', 500) # 表示される列の数
pd.set_option('display.width', 1500)      # 表示する表の最高幅
# MetaTrader 5パッケージについてのデータを表示する
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print()
# MetaTrader 5ターミナルとの接続を確立する
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# 履歴内の注文の数を取得する
from_date=datetime(2020,1,1)
to_date=datetime.now()
history_orders=mt5.history_orders_get(from_date, to_date, group="*GBP*")
if history_orders==None:
    print("No history orders with group=\"*GBP*\", error
code={}".format(mt5.last_error()))
elif len(history_orders)>0:
    print("history_orders_get({}, {}, group=\"*GBP*\")
={}".format(from_date,to_date,len(history_orders)))
print()

# ポジションチケットごとにすべての履歴注文を表示する
position_id=530218319
position_history_orders=mt5.history_orders_get(position=position_id)
if position_history_orders==None:
    print("No orders with position #{}".format(position_id))
    print("error code =",mt5.last_error())
elif len(position_history_orders)>0:
    print("Total history orders on position #{}:
{}".format(position_id,len(position_history_orders)))
   # 指定されたポジションチケットを持つすべての履歴注文を表示する
    for position_order in position_history_orders:
        print(position_order)
    print()
    # pandas.DataFrameを使用してこれらの注文を表として表示する
```

© 2000–2025, MetaQuotes Ltd.

```
    df=pd.DataFrame
(list(position_history_orders),columns=position_history_orders[0]._asdict().keys())
    df.drop(['time_expiration','type_time','state','position_by_id','reason','volume
_current','price_stoplimit','sl','tp'], axis=1, inplace=True)
    df['time_setup'] = pd.to_datetime(df['time_setup'], unit='s')
    df['time_done'] = pd.to_datetime(df['time_done'], unit='s')
    print(df)

# MetaTrader 5ターミナルへの接続をシャットダウンする
mt5.shutdown()
```

結果:

```
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

history_orders_get(2020-01-01 00:00:00, 2020-03-25 17:17:32.058795, group="*GBP*")
=14

Total history orders on position #530218319: 2
TradeOrder(ticket=530218319, time_setup=1582282114, time_setup_msc=1582282114681,
time_done=1582303777, time_done_msc=1582303777582, time_expiration=0, ...
TradeOrder(ticket=535548147, time_setup=1583176242, time_setup_msc=1583176242265,
time_done=1583176242, time_done_msc=1583176242265, time_expiration=0, ...

      ticket           time_setup  time_setup_msc           time_done  time_done_msc
 type  type_filling  magic  position_id  volume_initial  price_open  price_current
symbol comment external_id
0  530218319 2020-02-21 10:48:34   1582282114681 2020-02-21 16:49:37  1582303777582
    2             2      0    530218319            0.01     0.97898        0.97863
USDCHF
1  535548147 2020-03-02 19:10:42   1583176242265 2020-03-02 19:10:42  1583176242265
    1             0      0    530218319            0.01     0.95758        0.95758
USDCHF
```

参照

history_deals_total、history_deals_get

© 2000–2025, MetaQuotes Ltd.

history_deals_total

指定された期間内の取引履歴の取引数を取得します。

```
history_deals_total(
   date_from,     // 取引がリクエストされている最初の日
   date_to        // 取引がリクエストされている最後の日
   )
```

パラメータ

date_from

[in] 取引がリクエストされている最初の日。「datetime」オブジェクトまたは1970年1月1日からの経過秒数として設定されます。名前なし必須パラメータ。

date_to

[in] 取引がリクエストされている最後の日。「datetime」オブジェクトまたは1970年1月1日からの経過秒数として設定されます。名前なし必須パラメータ。

戻り値

整数値。

注意事項

この関数はHistoryDealsTotalに似ています。

例：

```
from datetime import datetime
import MetaTrader5 as mt5
# MetaTrader 5パッケージについてのデータを表示する
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5ターミナルとの接続を確立する
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# 履歴内の取引数を取得する
from_date=datetime(2020,1,1)
to_date=datetime.now()
deals=mt5.history_deals_total(from_date, to_date)
if deals>0:
    print("Total deals=",deals)
else:
    print("Deals not found in history")

# MetaTrader 5ターミナルへの接続をシャットダウンする
mt5.shutdown()
```

© 2000-2025, MetaQuotes Ltd.

参照

history_deals_get、history_orders_total

© 2000–2025, MetaQuotes Ltd.

history_deals_get

取引履歴から取引を取得します(チケットまたはポジションでフィルターする機能あり)。

時間間隔を指定して呼び出します。指定された時間間隔内にあるすべての取引を返します。

```
history_deals_get(
   date_from,            // 取引がリクエストされている最初の日
   date_to,              // 取引がリクエストされている最後の日
   group="GROUP"         // 銘柄の取引を選択するためのフィルター
   )
```

注文チケットを指定した呼び出し:DEAL_ORDERプロパティに指定された注文チケットを持つすべての取引を返します。

```
history_deals_get(
   ticket=TICKET         // 注文チケット
)
```

ポジションチケットを指定した呼び出し:DEAL_POSITION_IDプロパティに指定されたポジションチケットを持つすべての取引を返します。

```
history_deals_get(
   position=POSITION     // ポジションチケット
)
```

パラメータ

date_from

[in] 注文がリクエストされている最初の日。「datetime」オブジェクトまたは1970年1月1日からの経過秒数として設定されます。名前なし必須パラメータが最初に指定されます。

date_to

[in] 注文がリクエストされている最後の日。「datetime」オブジェクトまたは1970年1月1日からの経過秒数として設定されます。名前なし必須パラメータが2番目に指定されます。

group="GROUP"

[in] 必要な銘柄のグループを配置するためのフィルター。名前付きオプションパラメータ。グループが指定されている場合、関数は銘柄名の指定された基準を満たす取引のみを返します。

ticket=TICKET

[in] すべての取引が受信される注文のチケット(DEAL_ORDERに格納)。オプションパラメータ。指定されていない場合、フィルターは適用されません。

position=POSITION

[in] すべての取引が受信されるポジションのチケット(DEAL_POSITION_IDに格納)。オプションパラメータ。指定されていない場合、フィルターは適用されません。

戻り値

名前付きタプル構造(namedtuple)の形式で情報を返します。エラーの場合はNoneを返します。エラーに関する情報はlast_error()を使用して取得できます。

注意事項

© 2000-2025, MetaQuotes Ltd.

この関数を使用すると、指定した履歴期間内のすべての取引を、HistoryDealsTotalおよびHistoryDealSelect関数に似た呼び出しで受け取ることができます。

*group*パラメータを使用すると、取引を銘柄で並べ替えることができます。「*」は文字列の先頭と末尾で使用できます。

*group*パラメータには、複数のコンマ区切りの条件を含めることができます。条件は「*」を使用してマスクとして設定できます。論理否定記号「！」は除外に使用できます。すべての条件は順番に適用されます。つまり、グループに含める条件を最初に指定してから、除外条件を指定する必要があります。例えば、「*group*="*, !EUR"」では、全銘柄の取引を最初に選択してから、銘柄名に「EUR」が含まれる取引を後で除外します。

例：

```
import MetaTrader5 as mt5
from datetime import datetime
import pandas as pd
pd.set_option('display.max_columns', 500) # 表示される列の数
pd.set_option('display.width', 1500)      # 表示する表の最高幅
# MetaTrader 5パッケージについてのデータを表示する
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print()
# MetaTrader 5ターミナルとの接続を確立する
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# 履歴内の取引数を取得する
from_date=datetime(2020,1,1)
to_date=datetime.now()
# 指定された期間内に名前に「GBP」が含まれる銘柄の取引を取得する
deals=mt5.history_deals_get(from_date, to_date, group="*GBP*")
if deals==None:
    print("No deals with group=\"*USD*\", error code={}".format(mt5.last_error()))
elif len(deals)> 0:
    print("history_deals_get({}, {}, group=\"*GBP*\")
={}".format(from_date,to_date,len(deals)))

# 名前に「EUR」も「GBP」も含まれていない銘柄の取引を取得する
deals = mt5.history_deals_get(from_date, to_date, group="*,!*EUR*,!*GBP*")
if deals == None:
    print("No deals, error code={}".format(mt5.last_error()))
elif len(deals) > 0:
    print("history_deals_get(from_date, to_date, group=\"*,!*EUR*,!*GBP*\") =",
len(deals))
    # 取得したすべての取引を「そのまま」表示する
    for deal in deals:
        print("  ",deal)
    print()
    # pandas.DataFrameを使用してこれらの取引を表として表示する
```

© 2000–2025, MetaQuotes Ltd.

```
    df=pd.DataFrame(list(deals),columns=deals[0]._asdict().keys())
    df['time'] = pd.to_datetime(df['time'], unit='s')
    print(df)
print("")

# ポジション#530218319に関連したすべての取引を取得する
position_id=530218319
position_deals = mt5.history_deals_get(position=position_id)
if position_deals == None:
    print("No deals with position #{}".format(position_id))
    print("error code =", mt5.last_error())
elif len(position_deals) > 0:
    print("Deals with position id #{}: {}".format(position_id, len(position_deals)))
    # pandas.DataFrameを使用してこれらの取引を表として表示する
    df=pd.DataFrame(list(position_deals),columns=position_deals[0]._asdict().keys())
    df['time'] = pd.to_datetime(df['time'], unit='s')
    print(df)

# MetaTrader 5ターミナルへの接続をシャットダウンする
mt5.shutdown()
```

結果:

```
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

history_deals_get(from_date, to_date, group="*GBP*") = 14

history_deals_get(from_date, to_date, group="*,!*EUR*,!*GBP*") = 7
   TradeDeal(ticket=506966741, order=0, time=1582202125, time_msc=1582202125419,
type=2, entry=0, magic=0, position_id=0, reason=0, volume=0.0, pri ...
   TradeDeal(ticket=507962919, order=530218319, time=1582303777,
time_msc=1582303777582, type=0, entry=0, magic=0, position_id=530218319,
reason=0, ...
   TradeDeal(ticket=513149059, order=535548147, time=1583176242,
time_msc=1583176242265, type=1, entry=1, magic=0, position_id=530218319,
reason=0, ...
   TradeDeal(ticket=516943494, order=539349382, time=1583510003,
time_msc=1583510003895, type=1, entry=0, magic=0, position_id=539349382,
reason=0, ...
   TradeDeal(ticket=516943915, order=539349802, time=1583510025,
time_msc=1583510025054, type=0, entry=0, magic=0, position_id=539349802,
reason=0, ...
   TradeDeal(ticket=517139682, order=539557870, time=1583520201,
time_msc=1583520201227, type=0, entry=1, magic=0, position_id=539349382,
reason=0, ...
   TradeDeal(ticket=517139716, order=539557909, time=1583520202,
time_msc=1583520202971, type=1, entry=1, magic=0, position_id=539349802,
reason=0, ...

      ticket      order                time       time_msc  type  entry  magic
position_id  reason  volume    price  commission  swap     profit  fee  symbol
comment external_id
```

© 2000-2025, MetaQuotes Ltd.

```
0  506966741           0 2020-02-20 12:35:25  1582202125419     2      0      0
      0       0    0.00  0.00000         0.0   0.0  100000.00  0.0

1  507962919  530218319 2020-02-21 16:49:37  1582303777582     0      0      0
530218319       0    0.01  0.97898         0.0   0.0       0.00  0.0  USDCHF

2  513149059  535548147 2020-03-02 19:10:42  1583176242265     1      1      0
530218319       0    0.01  0.95758         0.0   0.0     -22.35  0.0  USDCHF

3  516943494  539349382 2020-03-06 15:53:23  1583510003895     1      0      0
539349382       0    0.10  0.93475         0.0   0.0       0.00  0.0  USDCHF

4  516943915  539349802 2020-03-06 15:53:45  1583510025054     0      0      0
539349802       0    0.10  0.66336         0.0   0.0       0.00  0.0  AUDUSD

5  517139682  539557870 2020-03-06 18:43:21  1583520201227     0      1      0
539349382       0    0.10  0.93751         0.0   0.0     -29.44  0.0  USDCHF

6  517139716  539557909 2020-03-06 18:43:22  1583520202971     1      1      0
539349802       0    0.10  0.66327         0.0   0.0      -0.90  0.0  AUDUSD


Deals with position id #530218319: 2
      ticket      order                time       time_msc  type  entry  magic
position_id  reason  volume    price  commission  swap  profit  fee  symbol comment
external_id
0  507962919  530218319 2020-02-21 16:49:37  1582303777582     0      0      0
530218319       0    0.01  0.97898         0.0   0.0    0.00  0.0  USDCHF

1  513149059  535548147 2020-03-02 19:10:42  1583176242265     1      1      0
530218319       0    0.01  0.95758         0.0   0.0  -22.35  0.0  USDCHF
```

参照

history_deals_total、history_orders_get

© 2000–2025, MetaQuotes Ltd.

機械学習におけるONNXモデル

ONNX (Open Neural Network Exchange)は、機械学習モデルのオープンソース形式です。このプロジェクトにはいくつかの大きな利点があります。

- ONNXは、Microsoft、Facebook、Amazonなどの大企業やその他のパートナーによってサポートされています。
- そのオープン形式により、異なる機械学習ツールキット間の形式変換が可能になります。また、Microsoftの ONNXMLTools では、モデルを ONNX形式に変換できます。
- MQL5は、渡されたパラメータの型がモデルと一致しない場合に、モデルの入力と出力に対して自動データ型変換を提供します。
- ONNXモデルは、さまざまな機械学習ツールを使用して作成できます。これらは現在、Caffe2、Microsoft Cognitive Toolkit、MXNet、PyTorch、OpenCVでサポートされています。他の一般的なフレームワークやライブラリのインターフェイスも利用できます。
- MQL5言語を使用すると、取引戦略でONNXモデルを実装できます。それをMetaTrader 5プラットフォームのすべての利点とともに使用して、金融市場で効率的な運用を実現できます。
- ライブ取引用にモデルを調整する前に、サードパーティツールを使用せずに、ストラテジーテスターの履歴データでモデルの動作をテストできます。

MQL5は、ONNXを操作するために次の関数を備えています。

関数	アクション
OnnxCreate	ONNXセッションを作成し、*.onnxファイルからモデルを読み込みます
OnnxCreateFromBuffer	ONNXセッションを作成し、データ配列からモデルを読み込みます
OnnxRelease	ONNXセッションを閉じます
OnnxRun	ONNXモデルを実行します
OnnxGetInputCount	ONNXモデルの入力数を取得します
OnnxGetOutputCount	ONNXモデルの出力数を取得します
OnnxGetInputName	インデックスによってモデルの入力の名前を取得します
OnnxGetOutputName	インデックスによってモデルの出力の名前を取得します
OnnxGetInputTypeInfo	モデルから入力型の説明を取得します
OnnxGetOutputTypeInfo	モデルから出力型の説明を取得します
OnnxSetInputShape	インデックスによってモデルの入力データの形状を設定します
OnnxSetOutputShape	インデックスによってモデルの出力データの形状を設定します

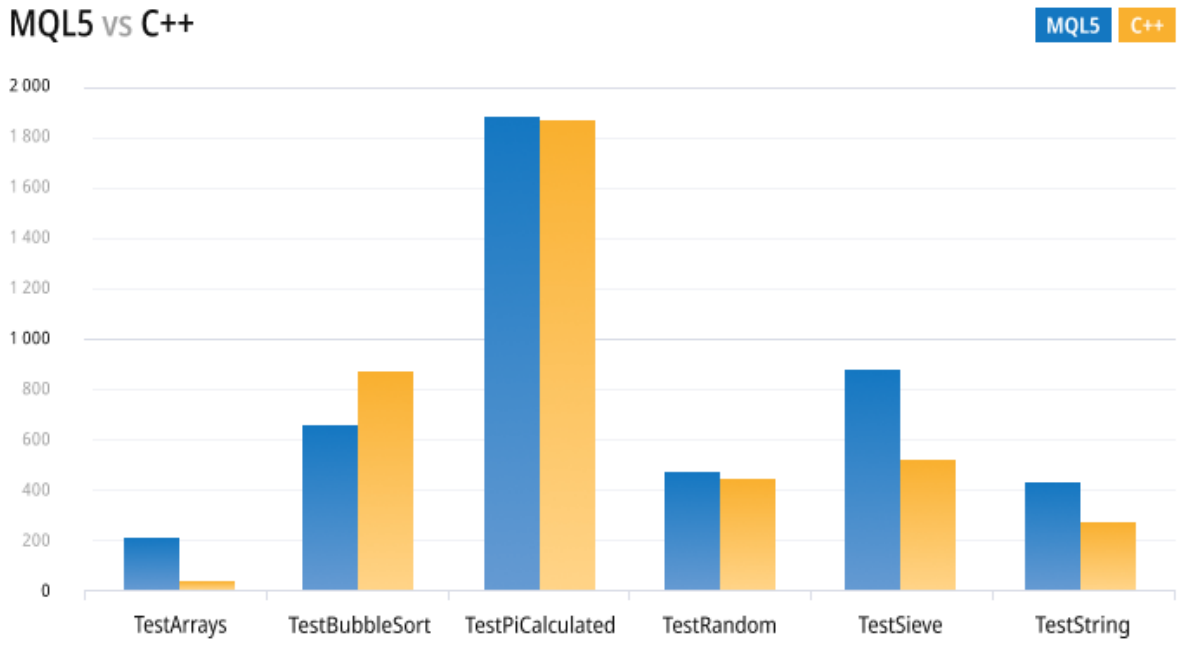
© 2000–2025, MetaQuotes Ltd.

MQL5のONNXサポート

ONNXは、機械学習モデルを表すオープンフォーマットビルドです。この標準は、開発者がさまざまなフレームワーク、ツール、ランタイム、コンパイラでモデルを使用できるようにするための演算子の共通セットと共通のファイル形式を定義します。

したがって、オープンなONNX形式を使用すると、異なるプラットフォームと機械学習ツールキット間で機械学習モデルを受信および転送することができます。ONNXはMQL5言語で実装されており、AI開発者はMetaTrader 5プラットフォームが提供する高性能実行環境で作成されたモデルを実行できるようになります。

MQL5の実行速度はC++アプリケーションの速度に匹敵します。これは、MQL5およびC++で標準テストを実行した結果によって証明されています。バーが低いほど、実行にかかる時間（ミリ秒単位）が短くなり、結果が良くなります。テストは、Windows 10（ビルド17763）x64、Xeon E5-2630 v4 @ 2.20GHz、メモリ: 65457 Mbで実施されました。



新しい非同期取引操作とネイティブONNXサポートは、これまで少数のプロのAI開発者や機関投資家のみが利用できたような新しい機会をトレーダーに提供します。MQL5のONNXサポートにより、トレーダーは好みの開発環境で金融市場取引のモデルを訓練し、低ネットワークコスト、オーダーブックの高速更新、非同期注文送信をもって取引できるようになります。

現在、ONNXはMicrosoft、Facebook、Amazonなどのパートナー企業によって開発および保守されており、このオープンプロジェクトのさらなる発展が保証されています。

© 2000–2025, MetaQuotes Ltd.

© 2000–2025, MetaQuotes Ltd.

フォーマット変換

ONNXはオープン形式であり、さまざまな機械学習ツールキットのモデルを使用できます。この形式は、Chainer、Caffee2、PyTorchなどの多くのフレームワークでサポートされています。

モデルをONNX形式に変換するための最も一般的なツールの1つは、MicrosoftのONNXMLToolsです。

ONNXMLToolsのインストールと使用手順は、GitHubリポジトリで入手できます。現在、次のツールキットがサポートされています。

- Keras（keras2onnxコンバーターのラッパー）
- Tensorflow（tf2onnxコンバーターのラッパー）
- scikit-learn（skl2onnxコンバーターのラッパー）
- Apple Core ML
- Spark ML（実験段階）
- LightGBM
- libscm
- XGBoost
- H2O
- CatBoost

ONNXMLToolsは簡単にインストールできます。インストールの詳細とモデル変換の例については、プロジェクトページ(https://github.com/onnx/onnxmltools#install)をご覧ください。

© 2000–2025, MetaQuotes Ltd.

ONNXモデルの実行時に入力値と出力値を自動変換します

MQL5の現在のONNXバージョンは、入出力値のテンソルのみをサポートしています。テンソルは、次のデータ型の要素を含むデータ配列です。

ONNX型	対応するMQL5型
ONNX_DATA_TYPE_BOOL	bool
ONNX_DATA_TYPE_FLOAT	float
ONNX_DATA_TYPE_UINT8	uchar
ONNX_DATA_TYPE_INT8	char
ONNX_DATA_TYPE_UINT16	ushort
ONNX_DATA_TYPE_INT16	short
ONNX_DATA_TYPE_INT32	int
ONNX_DATA_TYPE_INT64	long
ONNX_DATA_TYPE_FLOAT16	—
ONNX_DATA_TYPE_DOUBLE	double
ONNX_DATA_TYPE_UINT32	uint
ONNX_DATA_TYPE_UINT64	ulong
ONNX_DATA_TYPE_COMPLEX64	—
ONNX_DATA_TYPE_COMPLEX128	complex
ONNX_DATA_TYPE_BFLOAT16	—
ONNX_DATA_TYPE_STRING	—

配列、ベクトル、行列（これらをデータと呼びます）のみをONNXモデルに入出力値として供給できます。

パラメータ型がONNXモデルのパラメータ型と一致せず、OnnxRunがONNX_NO_CONVERSIONフラグなしで呼び出された場合を指定すると、自動データ変換が適用されます。自動変換は、ONNXモデルを実行する前に、ユーザデータが関連する変換を使用してONNXテンソルにコピーされることを意味します。

ONNXモデルが自動変換なしで実行される場合、モデルは追加のコピーなしでデータを使用して計算されます。

> 重要：自動変換はオーバーフロー（切り捨て）を制御しないため、ONNXモデルに入力されるデータとデータ型を注意深く監視する必要があります。

自動変換は次のONNX型をサポートしています。

- ONNX_DATA_TYPE_BOOL
- ONNX_DATA_TYPE_FLOAT
- ONNX_DATA_TYPE_UINT8

© 2000–2025, MetaQuotes Ltd.

- ONNX_DATA_TYPE_INT8
- ONNX_DATA_TYPE_UINT16
- ONNX_DATA_TYPE_INT16
- ONNX_DATA_TYPE_INT32
- ONNX_DATA_TYPE_INT64
- ONNX_DATA_TYPE_FLOAT16
- ONNX_DATA_TYPE_DOUBLE
- ONNX_DATA_TYPE_UINT32
- ONNX_DATA_TYPE_UINT64
- ONNX_DATA_TYPE_COMPLEX64
- ONNX_DATA_TYPE_COMPLEX128

次はサポートされていません。

- ONNX_DATA_TYPE_BFLOAT16
- ONNX_DATA_TYPE_STRING

テンソル型別の自動変換ルール

MQL5型がモデルでサポートされる型のリストに含まれていない場合、ONNXモデルを実行するとERR_ONNX_NOT_SUPPORTEDエラー（エラーコード5802）が返されます。

注：自動変換中、color型はuintとして処理され、datetimeはlongとして処理されます。

入力値の自動変換

ONNX型（テンソル項目型）	自動変換でサポートされるMQL5型
ONNX_DATA_TYPE_BOOL	bool、char、uchar、short、ushort、int、color、uint、datetime、long、folat、double、complex 変換中、データ要素は0との単純な比較によって確認されます
ONNX_DATA_TYPE_FLOAT16	float、double
ONNX_DATA_TYPE_FLOAT	char、uchar、short、ushort、int、color、uint、datetime、long、ulong、float、double
ONNX_DATA_TYPE_UINT8	ONNX_DATA_TYPE_FLOAT参照
ONNX_DATA_TYPE_INT8	ONNX_DATA_TYPE_FLOAT参照
ONNX_DATA_TYPE_UINT16	ONNX_DATA_TYPE_FLOAT参照
ONNX_DATA_TYPE_INT16	ONNX_DATA_TYPE_FLOAT参照
ONNX_DATA_TYPE_INT32	ONNX_DATA_TYPE_FLOAT参照

© 2000–2025, MetaQuotes Ltd.

ONNX型（テンソル項目型）	自動変換でサポートされるMQL5型
ONNX_DATA_TYPE_INT64	ONNX_DATA_TYPE_FLOAT参照
ONNX_DATA_TYPE_DOUBLE	ONNX_DATA_TYPE_FLOAT参照
ONNX_DATA_TYPE_UINT32	ONNX_DATA_TYPE_FLOAT参照
ONNX_DATA_TYPE_UINT64	ONNX_DATA_TYPE_FLOAT参照
ONNX_DATA_TYPE_COMPLEX64	complex
ONNX_DATA_TYPE_COMPLEX128	complex

出力値の自動変換

ONNX型（テンソル項目型）	自動変換でサポートされるMQL5型
ONNX_DATA_TYPE_BOOL	bool、char、uchar、short、ushort、int、color、uint、datetime、long、folat、double、complex テンソル要素がゼロの場合、データ要素は0に設定されます。それ以外の場合、値は1になります
ONNX_DATA_TYPE_FLOAT16	float、double
ONNX_DATA_TYPE_FLOAT	char、uchar、short、ushort、int、color、uint、datetime、long、ulong、float、double
ONNX_DATA_TYPE_UINT8	ONNX_DATA_TYPE_FLOAT参照
ONNX_DATA_TYPE_INT8	ONNX_DATA_TYPE_FLOAT参照
ONNX_DATA_TYPE_UINT16	ONNX_DATA_TYPE_FLOAT参照
ONNX_DATA_TYPE_INT16	ONNX_DATA_TYPE_FLOAT参照
ONNX_DATA_TYPE_INT32	ONNX_DATA_TYPE_FLOAT参照
ONNX_DATA_TYPE_INT64	ONNX_DATA_TYPE_FLOAT参照
ONNX_DATA_TYPE_DOUBLE	ONNX_DATA_TYPE_FLOAT参照
ONNX_DATA_TYPE_UINT32	ONNX_DATA_TYPE_FLOAT参照
ONNX_DATA_TYPE_UINT64	ONNX_DATA_TYPE_FLOAT参照
ONNX_DATA_TYPE_COMPLEX64	complex
ONNX_DATA_TYPE_COMPLEX128	complex

参照

型キャスト

© 2000–2025, MetaQuotes Ltd.

モデルの作成

ONNXフォーマットで準備が整ったモデルを取得するには、複数の方法が使用できます。人気のあるONNX Model Zooライブラリには、さまざまな種類のタスク用に事前訓練された複数のONNXモデルが含まれています。このコレクションの良いところは、各モデルのノートブックに訓練データセットへのリンクと、モデルアーキテクチャを説明する元の論文への参照が含まれていることです。

ほとんどの機械学習フレームワークはPythonを使用します。Python用のONNXランタイムをインストールするには、次のコマンドのいずれかを使用します。

```
pip install onnxruntime        # CPU build
pip install onnxruntime-gpu    # GPU build
```

PythonでONNXランタイムを呼び出すには、次のコマンドを使用します

```
import onnxruntime
session = onnxruntime.InferenceSession("path to model")
```

モデル入力と出力については、関連するモデルのドキュメントを確認してください。NetronやWinML Dashboardなどの視覚化ツールを使用してモデルを表示することもできます。ONNXランタイムでは、モデルのメタデータとその入力と出力をクエリすることもできます。

```
results = session.run(["output1", "output2"], {
                      "input1": indata1, "input2": indata2})
results = session.run([], {"input1": indata1, "input2": indata2})
```

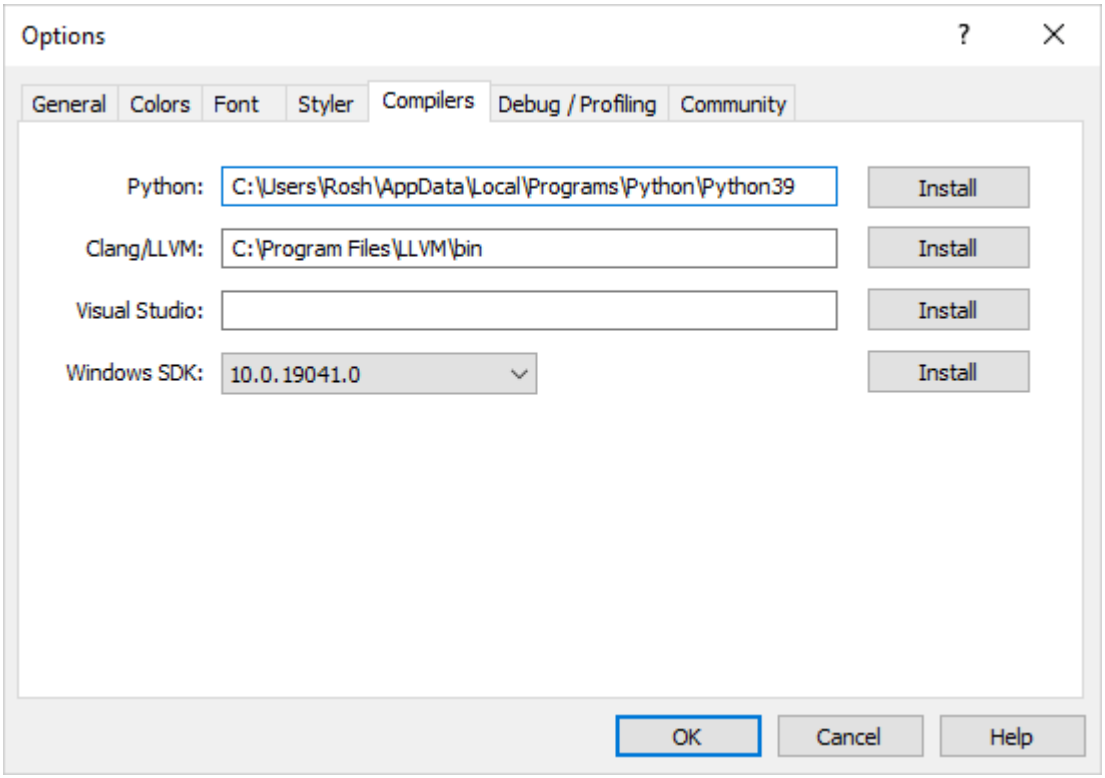
ONNXモデルは、MetaTrader 5ターミナルで直接作成することも、Pythonを使用してMetaEditorで作成することもできます。

MetaTrader 5でのPython

MetaTrader 5は、すぐに使えるPythonスクリプトのサポートを備えています。これらの操作を可能にするために、ターミナルの開発者はPython用のMetaTrader 5モジュールhttps://pypi.org/project/MetaTrader5/をが提供しています。

MetaEditor統合開発環境では、MQL5でアプリケーションを作成するだけでなく、エディタからPythonスクリプトを直接実行することもできます。これを行うには、MetaEditorの設定で実行可能ファイルへのパスを指定します。

© 2000–2025, MetaQuotes Ltd.



Pythonがコンピュータにインストールされていない場合は、[インストール]をクリックしてインストールファイルをダウンロードしてください。

MetaEdtiorでPythonスクリプトを作成することも、ターミナルのデータフォルダーにアップロードして、F7（コンパイル）キーを使用してすぐに実行することもできます。これにより、MetaTrader 5ターミナルが開かれ、現在のチャートでスクリプトが実行されます。Pythonコンソール（stdout、stderr）からのメッセージは、エラーセクションに表示されます。

MetaTrader 5でのモデルの操作

MQL5言語を使用すると、MetaTrader 5ターミナルでONNXモデルを直接実行できます。これは3つの手順で行われます。

1. Pythonなどのサードパーティプラットフォームでモデルを訓練します。
2. モデルをONNXに変換します。
3. ONNX関数を使用してONNXモデルをエキスパートアドバイザーに組み込み、MetaTrader 5 ターミナルで実行します。

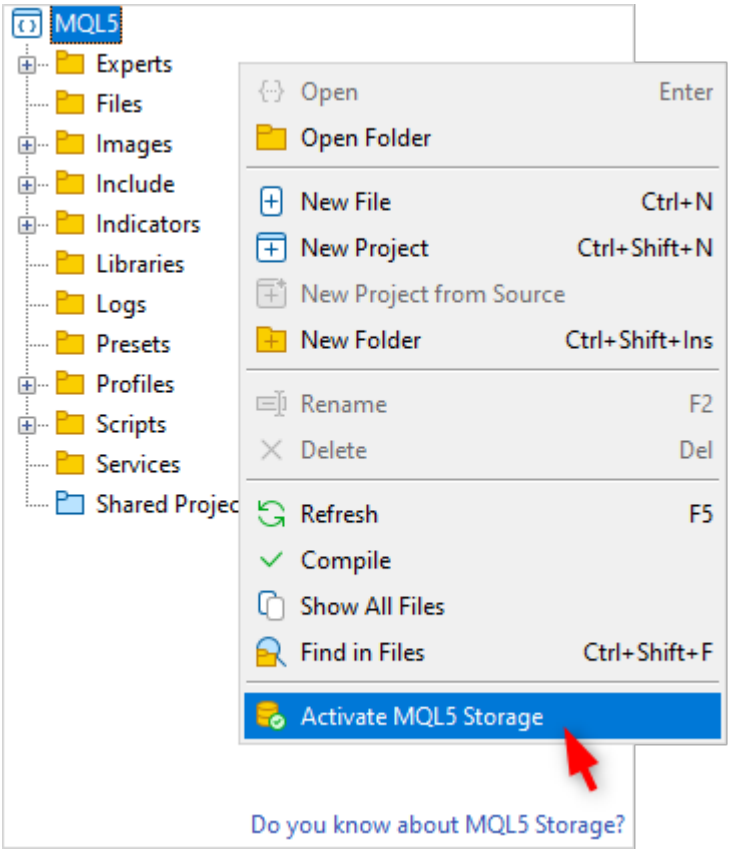
MQL5のPython統合により、Pythonスクリプトを実行してONNXモデルをMetaEditorに保存したり、MetaTrader 5のチャート上で直接実行したりできます。事前に作成されたPythonスクリプトを使用して、ターミナルで必要なだけ何度でもモデルを訓練できます。このライブラリには、ONNXモデルに入力できる価格データを取得するための既製の関数が含まれています。

- copy_rates_from – 指定された日付から始まるバーを取得する
- copy_rates_from_pos – 指定されたインデックスから始まるバーを取得する
- copy_ticks_range – 指定された期間のバーを取得する
- copy_ticks_from – 指定された日付から始まるティックを取得する
- copy_ticks_range – 指定された期間のティックを取得する

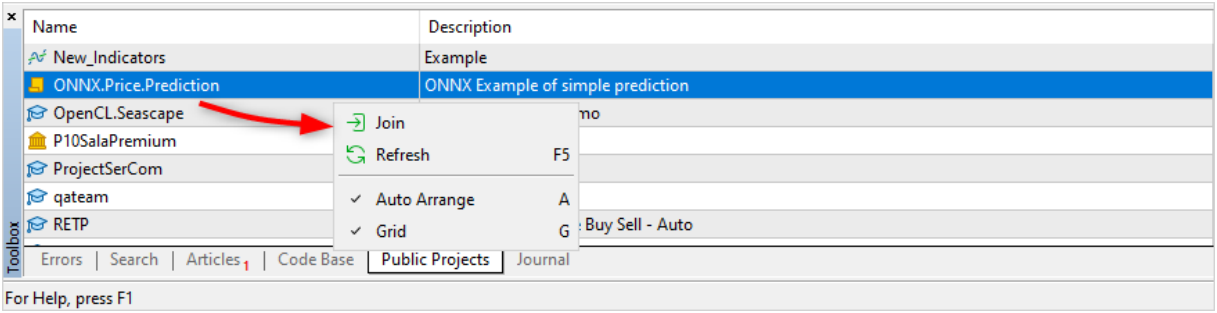
© 2000–2025, MetaQuotes Ltd.

モデルの例

完成したONNXモデルの例は、公開プロジェクトで入手できます。まず、MetaEditor設定でMQL5ログインを指定して（大文字と小文字を区別します）、ナビゲータで MQL5ストレージをアクティブにする必要があります。

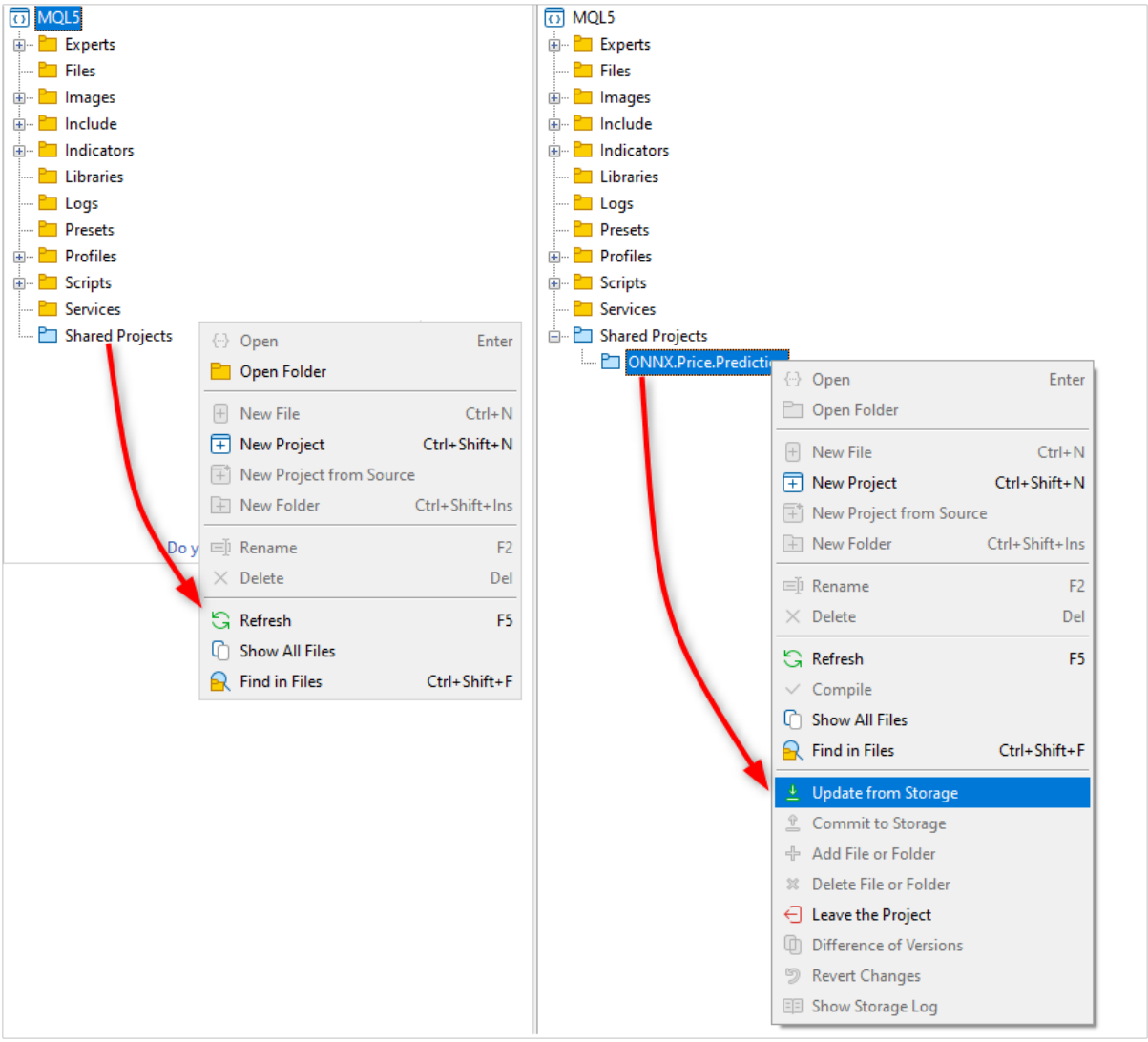


アクティブにした後、ONNX.Price.Predictionプロジェクトを見つけて、コンテキストメニューコマンドを使用して参加します。

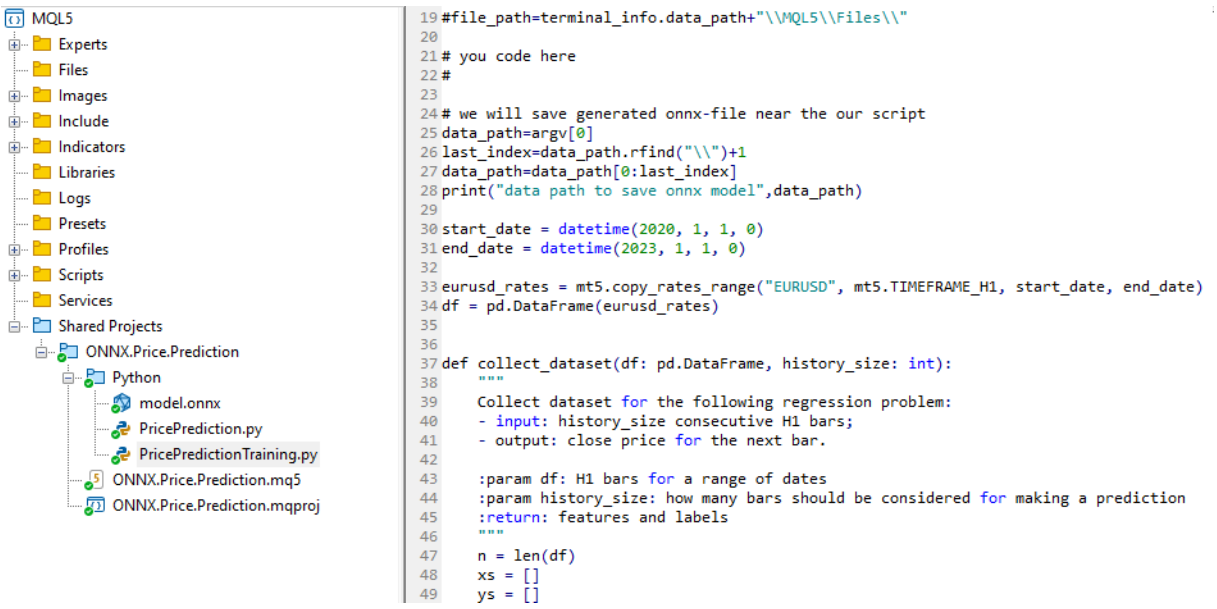


次に、MQL5ストレージからプロジェクトを更新します。

© 2000–2025, MetaQuotes Ltd.



プロジェクトには、ONNXモデル、2つのPython スクリプト、プロジェクト操作用のMQL5スクリプト、MQL5プロジェクトファイル(ONNX.Price.Prediction.mqproj)が含まれています。



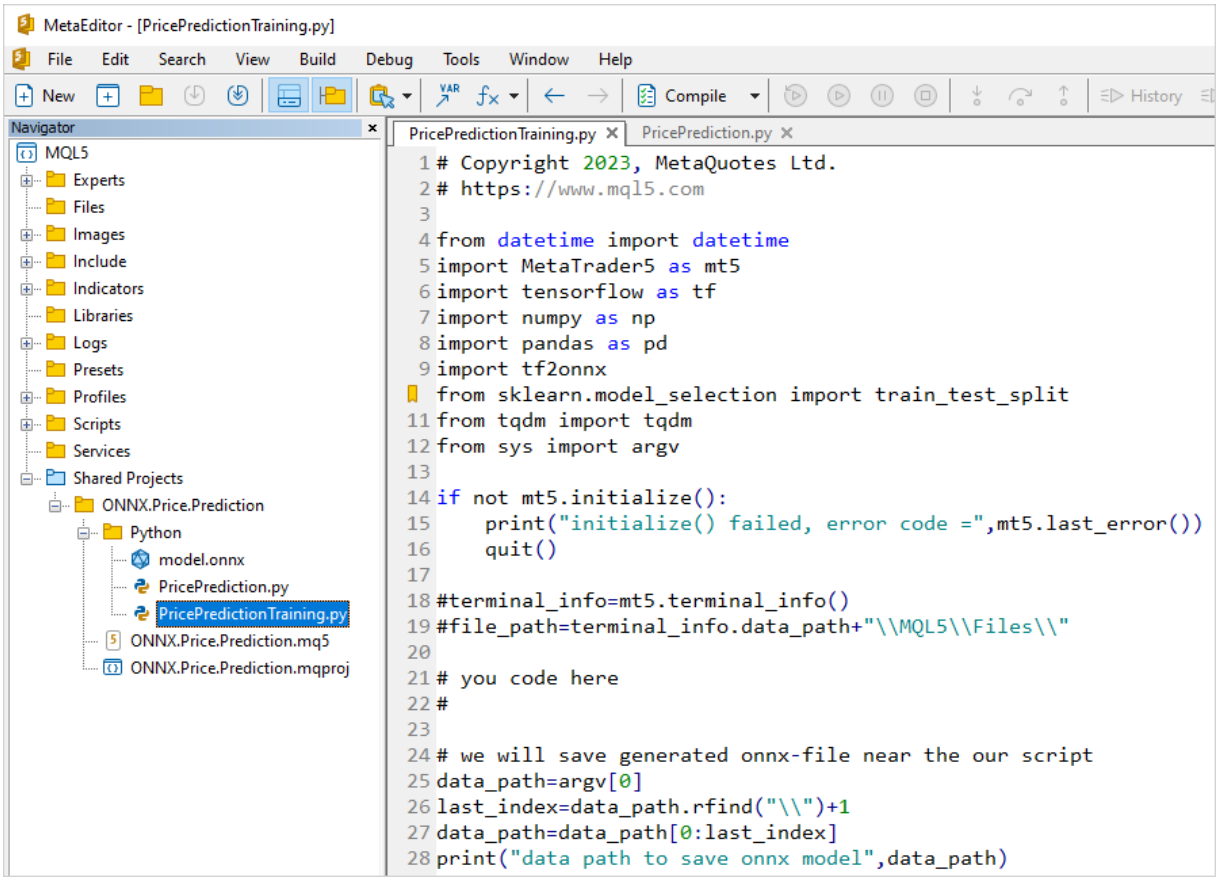

```
#file_path=terminal_info.data_path+"\\MQL5\\Files\\"

# you code here
#

# we will save generated onnx-file near the our script
data_path=argv[0]
last_index=data_path.rfind("\\")+1
data_path=data_path[0:last_index]
print("data path to save onnx model",data_path)

start_date = datetime(2020, 1, 1, 0)
end_date = datetime(2023, 1, 1, 0)

eurusd_rates = mt5.copy_rates_range("EURUSD", mt5.TIMEFRAME_H1, start_date, end_date)
df = pd.DataFrame(eurusd_rates)


def collect_dataset(df: pd.DataFrame, history_size: int):
    """
    Collect dataset for the following regression problem:
    - input: history_size consecutive H1 bars;
    - output: close price for the next bar.

    :param df: H1 bars for a range of dates
    :param history_size: how many bars should be considered for making a prediction
    :return: features and labels
    """
    n = len(df)
    xs = []
    ys = []
```

© 2000–2025, MetaQuotes Ltd.

プロジェクトに含まれているPricePredictionTraining.pyスクリプトを使用して、ONNXモデルを自分で作成できます。これを行うには、まずコマンドラインから必要なモジュールをインストールする必要があります。

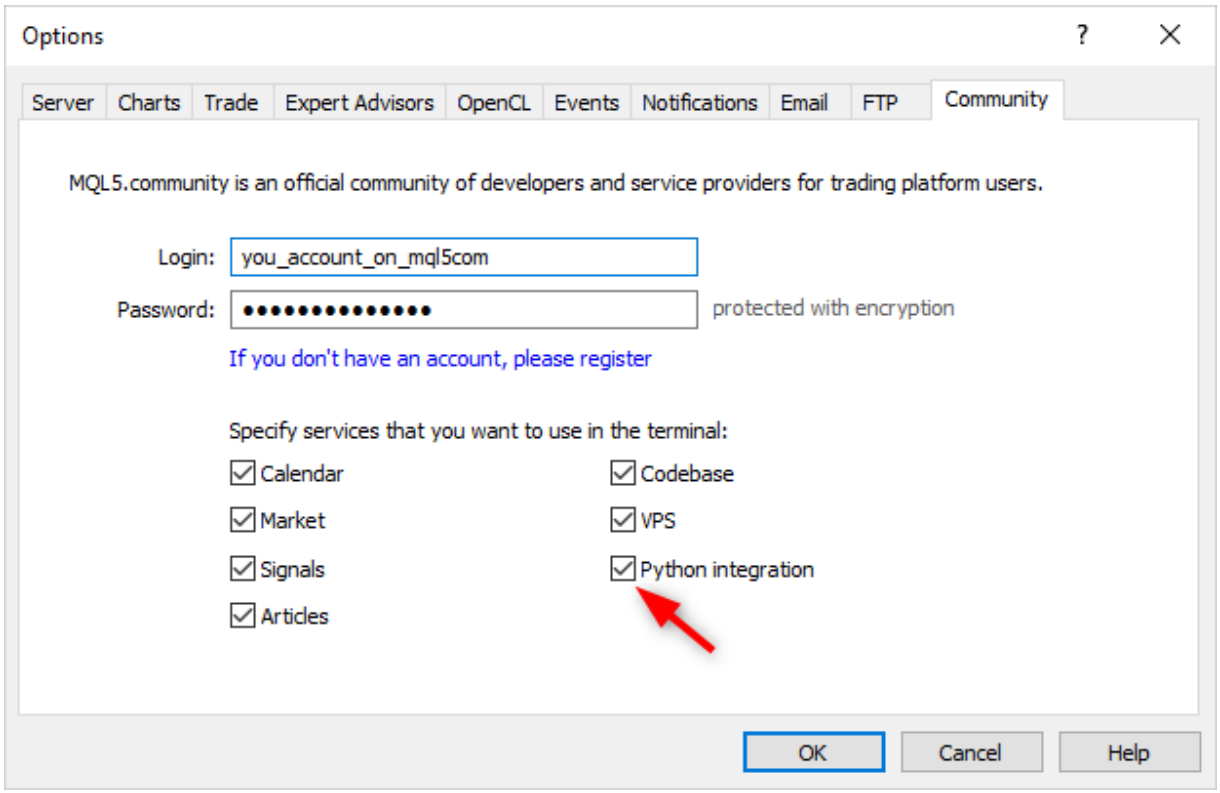
```
python.exe -m pip install --upgrade pip
python -m pip install --upgrade tensorflow
python -m pip install --upgrade pandas
python -m pip install --upgrade scikit-learn
python -m pip install --upgrade matplotlib
python -m pip install --upgrade tqdm
python -m pip install --upgrade metatrader5
python -m pip install --upgrade onnx==1.12
python -m pip install --upgrade tf2onnx
python -m pip install --upgrade numpy
python -m pip install onnxruntime
```

モジュールをインストールした後、MetaEditorでPricePredictionTraining.pyスクリプトを開き、[コンパイル]ボタンまたはF7キーを使用して実行します。



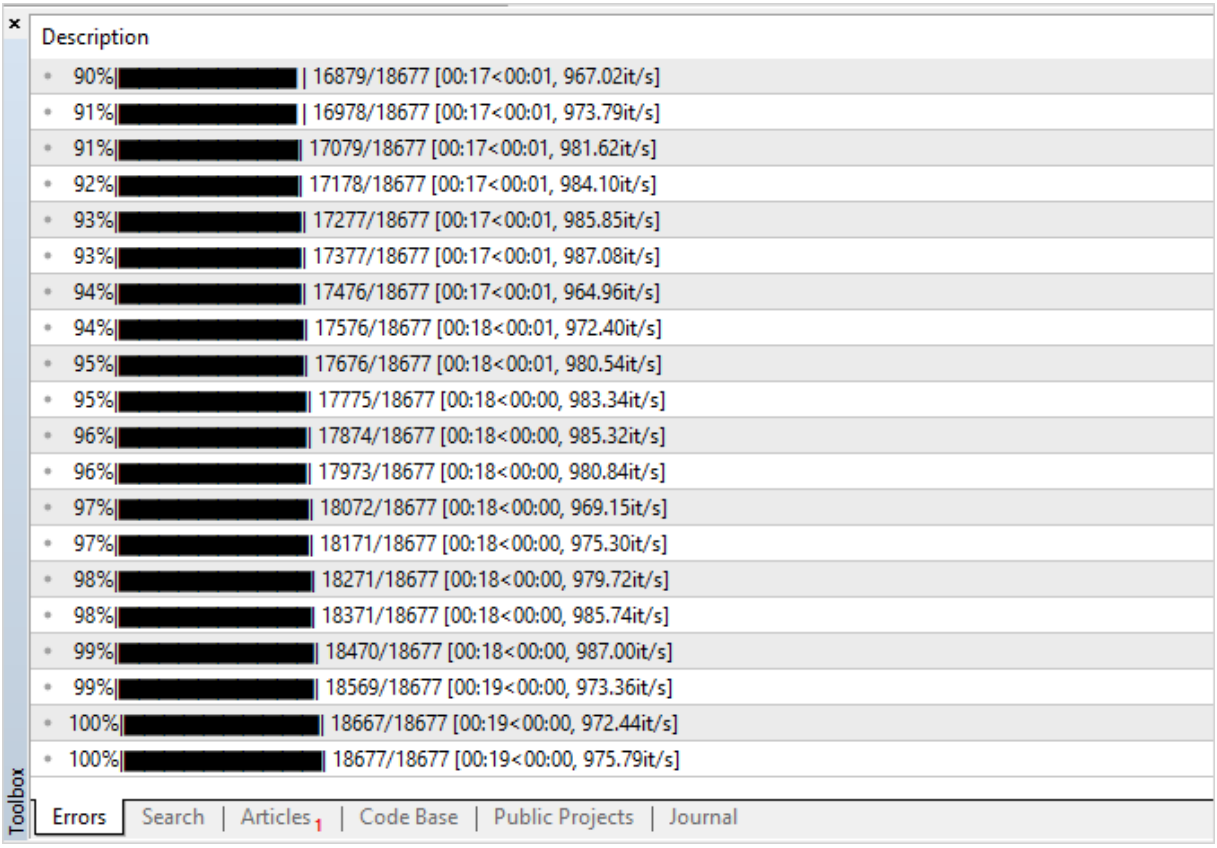
© 2000–2025, MetaQuotes Ltd.

Pythonスクリプトを実行する前に、MetaTrader 5ターミナルがEURUSD銘柄を使用できるサーバに接続されていることを確認してください。たとえば、MetaQuotes-Demoサーバに接続し、ターミナル設定で[Pythonとの統合]をオンにします。



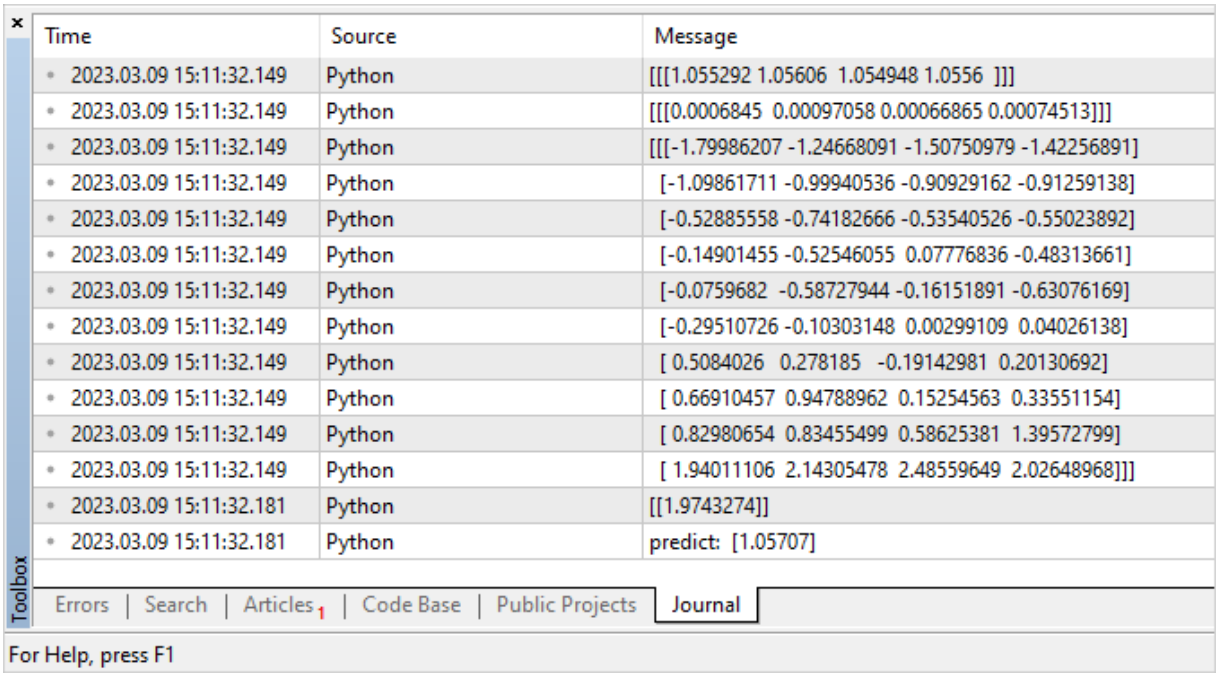
ネットワークの訓練中、MetaEditorはトレーニングが完了するまでPythonスクリプトからのメッセージを出力します。

© 2000-2025, MetaQuotes Ltd.



結果が100%になると、ONNXモデルの準備が完了し、<ターミナルデータディレクトリ>¥MQL5¥Shared Projects¥ONNX.Price.Prediction¥Pythonプロジェクトフォルダに保存されます。

F7を押して2番目のスクリプトPricePrediction.pyを実行すると、結果のモデルを確認できます。



© 2000–2025, MetaQuotes Ltd.

モデルの実行

MQL5でONNXモデルを実行するには、次の3つの手順を実行します。

1. OnnxCreate関数を使用して*.onnxファイルからモデルを読み込むか、OnnxCreateFromBufferを使用して配列からモデルを読み込みます。
2. OnnxSetInputShape関数とOnnxSetOutputShape関数を使用して入出力データの形状を指定します。
3. OnnxRun関数を使用してモデルを実行し、関連する入出力パラメータを渡します。
4. 必要に応じて、OnnxRelease関数を使用してモデル操作を終了できます。

ONNXモデルを作成するときは、https://github.com/microsoft/onnxruntime/blob/rel-1.14.0/docs/OperatorKernels.mdで説明されている既存の制限や規制を考慮する必要があります。

このような制限の例のいくつかを以下に示します。

操作	対応データ型
ReduceSum	tensor(double)、tensor(float)、tensor(int32)、tensor(int64)
Mul	tensor(bfloat16)、tensor(double)、tensor(float)、tensor(float16)、tensor(int32)、tensor(int64)、tensor(uint32)、tensor(uint64)

以下は、公開プロジェクトONNX.Price.PredictionのMQL5コード例です。

```
const long   ExtOutputShape[] = {1,1};     // モデルの出力形状
const long   ExtInputShape [] = {1,10,4}; // モデルの入力形状
#resource "Python/model.onnx" as uchar ExtModel[]// リソースとしてのモデル
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                      |
//+------------------------------------------------------------------+
int OnStart(void)
  {
   matrix rates;
//--- 10バーを取得する
   if(!rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,2,10))
      return(-1);
//--- 一連のOHLCベクトルを入力する
   matrix x_norm=rates.Transpose();
   vector m=x_norm.Mean(0);
   vector s=x_norm.Std(0);
   matrix mm(10,4);
   matrix ms(10,4);
//--- 正規化行列に入力する
   for(int i=0; i<10; i++)
     {
      mm.Row(m,i);
      ms.Row(s,i);
     }
//--- 入力データを正規化する
```

© 2000-2025, MetaQuotes Ltd.

```
   x_norm-=mm;
   x_norm/=ms;
//--- モデルを作成する
   long handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
//--- 入力データの形状を指定する
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("OnnxSetInputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- 出力データの形状を指定する
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("OnnxSetOutputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- 正規化された入力データをfloat型に変換する
   matrixf x_normf;
   x_normf.Assign(x_norm);
//--- ここでモデルの出力データ、つまり価格予測を取得する
   vectorf y_norm(1);
//--- モデルを実行する
   if(!OnnxRun(handle,ONNX_DEBUG_LOGS | ONNX_NO_CONVERSION,x_normf,y_norm))
     {
      Print("OnnxRun failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- モデルの出力値をログに出力する
   Print(y_norm);
//--- 逆変換を行って予測価格を取得する
   double y_pred=y_norm[0]*s[3]+m[3];
   Print("price predicted:",y_pred);
//--- 操作を完成する
   OnnxRelease(handle);
   return(0);
  }
```

スクリプト実行例：

```
ONNX: Creating and using per session threadpools since use_per_session_threads_ is
true
ONNX: Dynamic block base set to 0
ONNX: Initializing session.
ONNX: Adding default CPU execution provider.
ONNX: Total shared scalar initializer count: 0
ONNX: Total fused reshape node count: 0
```

© 2000-2025, MetaQuotes Ltd.

```
ONNX: Total shared scalar initializer count: 0
ONNX: Total fused reshape node count: 0
ONNX: Use DeviceBasedPartition as default
ONNX: Saving initialized tensors.
ONNX: Done saving initialized tensors
ONNX: Session successfully initialized.
[0.28188983]
predicted 1.0559258806393044
```

MetaTrader 5ターミナルは、計算に最適なエグゼキュータ（ONNXランタイム実行プロバイダ）を選択しました。この例では、モデルはCPU上で実行されました。

スクリプトを変更して、直前の10バーの値に基づいて行われた終値予測の成功率を計算してみましょう。

```
#resource "Python/model.onnx" as uchar ExtModel[]// リソースとしてのモデル

#define TESTS 10000  // テストデータセットの数
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
int OnStart()
  {
//--- モデルを作成する
   long session_handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
   if(session_handle==INVALID_HANDLE)
     {
      Print("Cannot create model. Error ",GetLastError());
      return(-1);
     }

//--- 入力テンソルサイズがモデルに定義されていないため、明示的に指定する
//--- 最初のインデックスはバッチサイズ、2番目のインデックスはシリーズサイズ、3番目のインデックスはシリーズ数（OHLC）
   const long input_shape[]={1,10,4};
   if(!OnnxSetInputShape(session_handle,0,input_shape))
     {
      Print("OnnxSetInputShape error ",GetLastError());
      return(-2);
     }

//--- 出力テンソルサイズがモデルに定義されていないため、明示的に指定する
//--- 最初のインデックスはバッチサイズであり、入力テンソルのバッチサイズと一致する必要がある
//--- 2番目のインデックスは予測価格の数（ここでは終値のみを予測）
   const long output_shape[]={1,1};
   if(!OnnxSetOutputShape(session_handle,0,output_shape))
     {
      Print("OnnxSetOutputShape error ",GetLastError());
      return(-3);
     }
//--- テストを実行する
   vector closes(TESTS);      // 検証価格を保存するためのベクトル
```

© 2000-2025, MetaQuotes Ltd.

```
   vector predicts(TESTS);     // 得られた予測を保存するためのベクトル
   vector prev_closes(TESTS); // 前の価格を保存するベクトル

   matrix rates;               // OHLCシリーズを取得するための行列
   matrix splitted[2];         // シリーズをテストと検証に分割するための2つの部分行列
   ulong  parts[]={10,1};      // 分割された部分行列のサイズ

//--- 前のバーから開始
   for(int i=1; i<=TESTS; i++)
     {
      //--- 11バーを取得する
      rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,i,11);
      //--- 行列をテストと検証に分割する
      rates.Vsplit(parts,splitted);
      //--- 検証行列から終値を取得する
      closes[i-1]=splitted[1][3][0];
      //--- last Close in the tested series
      prev_closes[i-1]=splitted[0][3][9];

      //--- 10バーのテスト行列をテストに提出する
      predicts[i-1]=PricePredictionTest(session_handle,splitted[0]);
      //--- ランタイムエラー
      if(predicts[i-1]<=0)
        {
         OnnxRelease(session_handle);
         return(-4);
        }
     }
//--- 操作を完成する
   OnnxRelease(session_handle);
//--- 価格変動が正しく予測されたかどうかを評価する
   int    right_directions=0;
   vector delta_predicts=prev_closes-predicts;
   vector delta_actuals=prev_closes-closes;

   for(int i=0; i<TESTS; i++)
      if((delta_predicts[i]>0 && delta_actuals[i]>0) || (delta_predicts[i]<0 &&
delta_actuals[i]<0))
         right_directions++;
   PrintFormat("right direction predictions = %.2f%%",
(right_directions*100.0)/double(TESTS));
//---
   return(0);
  }
//+------------------------------------------------------------------+
//|   データを準備してモデルを実行する                                |
//+------------------------------------------------------------------+
double PricePredictionTest(const long session_handle,matrix& rates)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   static matrixf input_data(10,4); // 変換された入力の行列
   static vectorf output_data(1);   // 結果を受け取るベクトル
   static matrix mm(10,4);          // 水平ベクトルMeanの行列
   static matrix ms(10,4);          // 水平ベクトルStdの行列

//--- 一連のOHLC垂直ベクトルをモデルに入力する必要がある
   matrix x_norm=rates.Transpose();
//--- 価格を正規化する
   vector m=x_norm.Mean(0);
   vector s=x_norm.Std(0);
   for(int i=0; i<10; i++)
     {
      mm.Row(m,i);
      ms.Row(s,i);
     }
   x_norm-=mm;
   x_norm/=ms;

//--- モデルを実行する
   input_data.Assign(x_norm);
   if(!OnnxRun(session_handle,ONNX_DEBUG_LOGS,input_data,output_data))
     {
      Print("OnnxRun error ",GetLastError());
      return(0);
     }
//--- 出力値から価格を非正規化する
   double y_pred=output_data[0]*s[3]+m[3];

   return(y_pred);
  }
```

スクリプトを実行します。予測精度は約51%です

```
ONNX: Creating and using per session threadpools since use_per_session_threads_ is
true
ONNX: Dynamic block base set to 0
ONNX: Initializing session.
ONNX: Adding default CPU execution provider.
ONNX: Total shared scalar initializer count: 0
ONNX: Total fused reshape node count: 0
ONNX: Total shared scalar initializer count: 0
ONNX: Total fused reshape node count: 0
ONNX: Use DeviceBasedPartition as default
ONNX: Saving initialized tensors.
ONNX: Done saving initialized tensors
ONNX: Session successfully initialized.
right direction predictions = 51.34 %
```

© 2000-2025, MetaQuotes Ltd.

© 2000–2025, MetaQuotes Ltd.

ストラテジーテスターでのモデルの検証

金融市場での運用のために作成されたモデルは、MetaTrader 5ターミナルストラテジーテスターで検証できます。これは最も速くて便利なオプションであり、市場環境や取引条件を追加でエミュレートする必要がなくなります。

モデルをテストするには、公開プロジェクトONNX.Price.Predictionのコードに基づいてエキスパートアドバイザーを作成しましょう。これにはいくつかの編集が必要になります。

モデルの作成をOnInit関数に移動します。onnxセッションはOnDeinitで終了します。OnTickハンドラへのメインモデル操作ブロックを見つけます。

また、実際の終値と予測を比較するために必要な、前の2つのバーの終値の取得を追加します。

エキスパートアドバイザーのコードは小さくて読みやすいです。

```
const long   ExtInputShape [] = {1,10,4}; // モデルの入力形状
const long   ExtOutputShape[] = {1,1};     // モデルの出力形状
#resource "Python/model.onnx" as uchar ExtModel[];// リソースとしてのモデル

long handle;          // モデルハンドル
ulong predictions=0; // 予測カウンタ
ulong confirmed=0;   // 成功した予測のカウンタ
//+------------------------------------------------------------------+
//| エキスパート初期化関数                                           |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 基本チェック
   if(_Symbol!="EURUSD")
     {
      Print("Symbol must be EURUSD, testing aborted");
      return(-1);
     }
   if(_Period!=PERIOD_H1)
     {
      Print("Timeframe must be H1, testing aborted");
      return(-1);
     }
//--- モデルを作成する
   handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
//--- 入力データの形状を指定する
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("OnnxSetInputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- 出力データの形状を指定する
```

© 2000-2025, MetaQuotes Ltd.

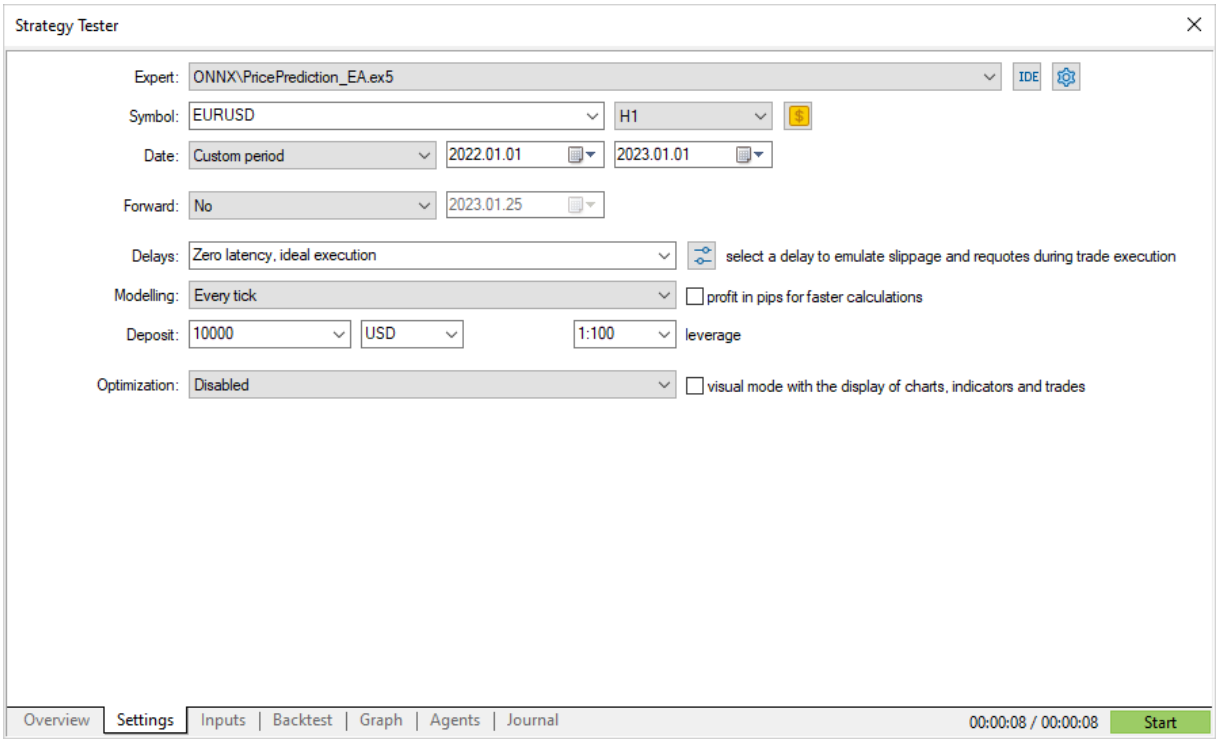
```
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("OnnxSetOutputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| エキスパート初期化解除関数                                       |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- モデル操作を完成する
   OnnxRelease(handle);
//--- 予測統計を計算して出力する
   PrintFormat("Successfull predictions = %.2f %
%",confirmed*100./double(predictions));
  }
//+------------------------------------------------------------------+
//| エキスパートティック関数                                         |
//+------------------------------------------------------------------+
void OnTick()
  {
   static datetime open_time=0;
   static double predict;
//--- 現在のバーの開始時間を確認する
   datetime time=iTime(_Symbol,_Period,0);
   if(time==0)
     {
      PrintFormat("Failed to get Time(0), error %d", GetLastError());
      return;
     }
//--- 開始時間が変更されていない場合は、次のOnTi ckを呼び出すまで終了する
   if(time==open_time)
      return;
//--- 完了した最後の2つのバーの終値を取得する
   double close[];
   int recieved=CopyClose(_Symbol,_Period,1,2,close);
   if(recieved!=2)
     {
      PrintFormat("CopyClose(2 bars) failed, error %d",GetLastError());
      return;
     }
   double delta_predict=predict-close[0]; // 予測される価格変動
   double delta_actual=close[1]-close[0]; // 実際の価格変動
   if((delta_predict>0 && delta_actual>0) || (delta_predict<0 && delta_actual<0))
      confirmed++;
```

© 2000–2025, MetaQuotes Ltd.

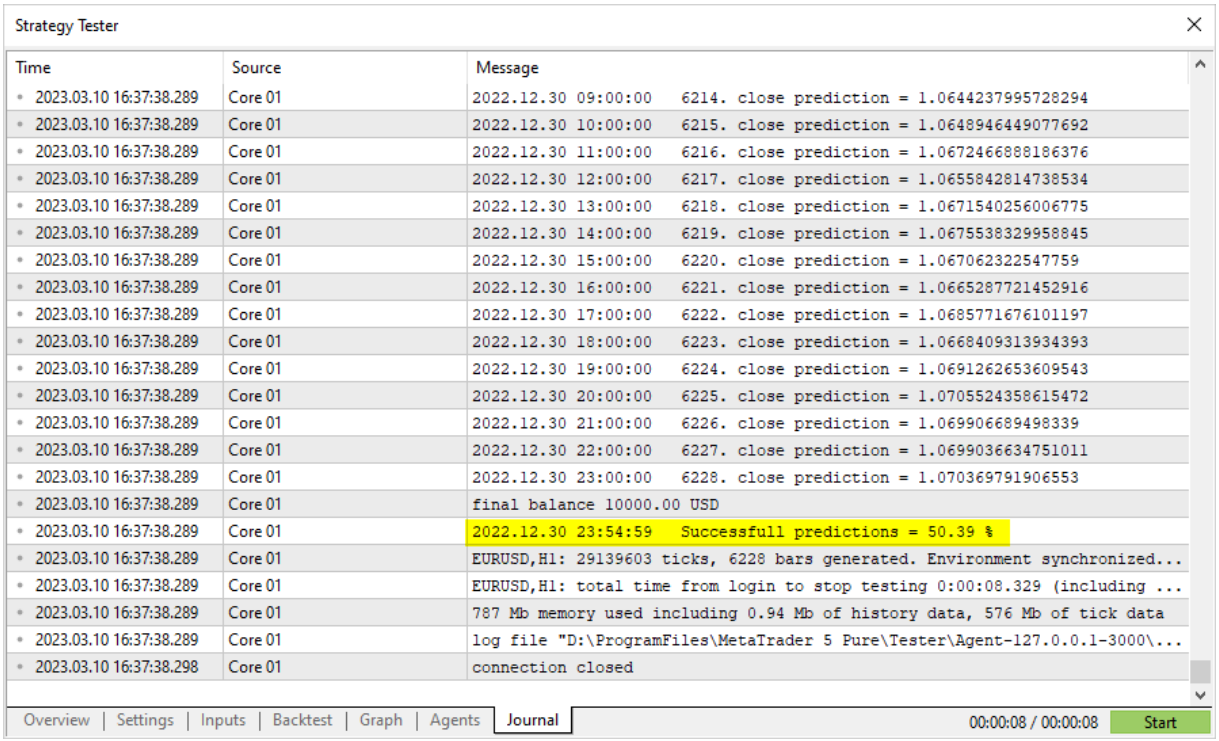
```
//--- 新しいバーの終値を計算して、次のバーの価格を検証する
   matrix rates;
//--- 10バーを取得する
   if(!rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,1,10))
      return;
//--- 一連のOHLCベクトルを入力する
   matrix x_norm=rates.Transpose();
   vector m=x_norm.Mean(0);
   vector s=x_norm.Std(0);
   matrix mm(10,4);
   matrix ms(10,4);
//--- 正規化行列に入力する
   for(int i=0; i<10; i++)
     {
      mm.Row(m,i);
      ms.Row(s,i);
     }
//--- 入力データを正規化する
   x_norm-=mm;
   x_norm/=ms;
//--- 正規化された入力データをfloat型に変換する
   matrixf x_normf;
   x_normf.Assign(x_norm);
//--- ここでモデルの出力データ、つまり価格予測を取得する
   vectorf y_norm(1);
//--- モデルを実行する
   if(!OnnxRun(handle,ONNX_DEBUG_LOGS | ONNX_NO_CONVERSION,x_normf,y_norm))
     {
      Print("OnnxRun failed, error ",GetLastError());
     }
//--- 逆変換を行って予測価格を取得し、それを新しいバーで検証する
   predict=y_norm[0]*s[3]+m[3];
   predictions++;  // 予測カウンタを増加する
   Print(predictions,". close prediction = ",predict);
//--- 次のティックを確認するためにバーの開始時間を保存する
   open_time=time;
  }
```

エキスパートアドバイザーをコンパイルし、2022年の期間でテストを実行します。EURUSDをH1時間枠で指定します。これは、モデルが訓練されたデータです。コードは新しいバーの出現を確認するため、ティックモデリングモードは無視できます。

© 2000-2025, MetaQuotes Ltd.



実行して、テスト操作ログで結果を確認します。2022年には予測の50%強が正しかったことが示されています。



予備的なモデルのテストで満足のいく結果が得られた場合は、このモデルに基づいて本格的な取引戦略の作成を開始できます。

© 2000–2025, MetaQuotes Ltd.

© 2000–2025, MetaQuotes Ltd.

OnnxCreate

ONNXセッションを作成し、*.onnxファイルからモデルを読み込みます。

```
long  OnnxCreate(
   string  filename,  // ファイルパス
   uint    flags      // モデル作成フラグ
   );
```

パラメータ

filename

[in] MQL5¥Files¥フォルダに相対する、モデルの*.onnxファイルへのパス

フラグ

[in] モデル作成モードを示す、ENUM_ONNX_FLAGSからのフラグ（ONNX_COMMON_FOLDER、ONNX_DEBUG_LOGS）

戻り値

作成されたセッションのハンドル。エラーが発生した場合はINVALID_HANDLE。エラーコードの取得にはGetLastError関数が呼ばれます。

注意事項

指定されたファイルがディスク上に見つからない場合、システムは、名前に「.onnx」拡張子を追加したファイルを開こうとします。

© 2000–2025, MetaQuotes Ltd.

OnnxCreateFromBuffer

ONNXセッションを作成し、データ配列からモデルを読み込みます。

```
long  OnnxCreateFromBuffer(
   const uchar&  buffer[],   // 配列の参照
   ulong         flags       // モデル作成フラグ
   );
```

パラメータ

buffer

[in] ONNXモデルデータを含む配列

フラグ

[in] モデル作成モードを示す、ENUM_ONNX_FLAGSからのフラグ（ONNX_COMMON_FOLDER、ONNX_DEBUG_LOGS）

戻り値

作成されたセッションのハンドル。エラーが発生した場合はINVALID_HANDLE。エラーコードの取得にはGetLastError関数が呼ばれます。

© 2000–2025, MetaQuotes Ltd.

OnnxRelease

ONNXセッションを閉じます。

```
bool  OnnxRelease(
   long    onnx_handle  // ONNXセッションハンドル
   );
```

パラメータ

onnx_handle

[in] OnnxCreateまたは OnnxCreateFromBufferで作成されたONNXセッションオブジェクトハンドル

戻り値

成功の場合はtrue、それ以外の場合はfalse。エラーコードの取得には GetLastError() 関数が呼ばれます。

© 2000–2025, MetaQuotes Ltd.

OnnxRun

ONNXモデルを実行します。

```
bool  OnnxRun(
   long     onnx_handle,  // ONNXセッションハンドル
   ulong    flags,        // 実行モードを説明するフラグ
   ...                    // モデルの入出力
   );
```

パラメータ

onnx_handle

[in] OnnxCreateまたは OnnxCreateFromBufferで作成されたONNXセッションオブジェクト ハンドル

フラグ

[in] 実行もウドを示す、ENUM_ONNX_FLAGS空のフラグ（ONNX_DEBUG_LOGS、ONNX_NO_CONVERSION）

...

[in] [out] モデルの入出力。

成功の場合はtrue、それ以外の場合はfalse。エラーコードの取得にはGetLastError関数が呼ばれます。

ENUM_ONNX_FLAGS

ID	説明
ONNX_DEBUG_LOGS	出力デバッグログ
ONNX_NO_CONVERSION	自動変換を無効にし、ユーザーデータをそのまま使用します
ONNX_COMMON_FOLDER	Common¥Filesフォルダからモデルファイルを読み込みます。値はFILE_COMMONフラグと同じです。

例：

```
const long                              ExtOutputShape[] = {1,1};     // モデルの出力形状
const long                              ExtInputShape [] = {1,10,4}; // モデルの入力形状
#resource "Python/model.onnx" as uchar ExtModel[]                    // リソースとしてのモデル
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
int OnStart(void)
  {
   matrix rates;
//--- 10バーを取得する
   if(!rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,2,10))
      return(-1);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 一連のOHLCベクトルを入力する
   matrix x_norm=rates.Transpose();
   vector m=x_norm.Mean(0);
   vector s=x_norm.Std(0);
   matrix mm(10,4);
   matrix ms(10,4);
//--- 正規化行列に入力する
   for(int i=0; i<10; i++)
     {
      mm.Row(m,i);
      ms.Row(s,i);
     }
//--- 入力データを正規化する
   x_norm-=mm;
   x_norm/=ms;
//--- モデルを作成する
   long handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
//--- 入力データの形状を指定する
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("OnnxSetInputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- 出力データの形状を指定する
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("OnnxSetOutputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- 正規化された入力データをfloat型に変換する
   matrixf x_normf;
   x_normf.Assign(x_norm);
//--- ここでモデルの出力データ、つまり価格予測を取得する
   vectorf y_norm(1);
//--- モデルを実行する
   if(!OnnxRun(handle,ONNX_DEBUG_LOGS | ONNX_NO_CONVERSION,x_normf,y_norm))
     {
      Print("OnnxRun failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- モデルの出力値をログに出力する
   Print(y_norm);
//--- 逆変換を行って予測価格を取得する
   double y_pred=y_norm[0]*s[3]+m[3];
   Print("price predicted:",y_pred);
//--- 操作完了
```

© 2000-2025, MetaQuotes Ltd.

```
   OnnxRelease(handle);
   return(0);
  };
```

参照

OnnxSetInputShape、OnnxSetOutputShape

© 2000–2025, MetaQuotes Ltd.

OnnxGetInputCount

ONNXモデルの入力数を取得します。

```
long  OnnxGetInputCount(
   long   onnx_handle  // ONNXセッションハンドル
   );
```

パラメータ

onnx_handle

[in] OnnxCreateまたはOnnxCreateFromBufferで作成されたONNXセッションオブジェクトハンドル

戻り値

成功した場合は入力パラメータの数、それ以外の場合は-1。エラーコードの取得にはGetLastError()関数が呼ばれます。

© 2000-2025, MetaQuotes Ltd.

OnnxGetOutputCount

ONNXモデルの出力数を取得します。

```
long  OnnxGetOutputCount(
   long    onnx_handle  // ONNXセッションハンドル
   );
```

パラメータ

onnx_handle

[in] OnnxCreateまたはOnnxCreateFromBufferで作成されたONNXセッションオブジェクトハンドル

戻り値

成功した場合は出力パラメータの数、それ以外の場合は-1。エラーコードの取得にはGetLastError()関数が呼ばれます。

© 2000-2025, MetaQuotes Ltd.

OnnxGetInputName

インデックスによってモデルの入力の名前を取得します。

```
string  OnnxGetInputName(
   long   onnx_handle,  // ONNXセッションハンドル
   long   index         // パラメータインデックス
   );
```

パラメータ

onnx_handle

[in] OnnxCreateまたは OnnxCreateFromBufferで作成されたONNXセッションオブジェクトハンドル

index

[in] 0から始まる入力パラメータのインデックス

戻り値

成功した場合は入力パラメータの名前、それ以外の場合はNULL。エラーコードの取得には GetLastError() 関数が呼ばれます。

© 2000-2025, MetaQuotes Ltd.

OnnxGetOutputName

インデックスによってモデルの出力の名前を取得します。

```
string  OnnxGetOutputName(
   long   onnx_handle,  // ONNXセッションハンドル
   long   index         // パラメータインデックス
   );
```

パラメータ

onnx_handle

[in] OnnxCreateまたは OnnxCreateFromBufferで作成されたONNXセッションオブジェクトハンドル

index

[in] 0から始まる出力パラメータのインデックス

戻り値

成功した場合は出力パラメータの名前、それ以外の場合はNULL。エラーコードの取得には GetLastError() 関数が呼ばれます。

© 2000-2025, MetaQuotes Ltd.

OnnxGetInputTypeInfo

モデルから入力型の説明を取得します。

```
bool  OnnxGetInputTypeInfo(
   long            onnx_handle,  // ONNXセッションハンドル
   long            index,        // パラメータインデックス
   OnnxTypeInfo&   typeinfo      // パラメータ型の記述
   );
```

パラメータ

onnx_handle

[in] OnnxCreateまたは OnnxCreateFromBufferで作成されたONNXセッションオブジェクトハンドル

index

[in] 0から始まる入力パラメータのインデックス

typeinfo

[out] 入力パラメータの型を記述するOnnxTypeInfo構造体

戻り値

成功の場合はtrue、それ以外の場合はfalse。エラーコードの取得には GetLastError() 関数が呼ばれます。

© 2000–2025, MetaQuotes Ltd.

OnnxGetOutputTypeInfo

モデルから出力型の説明を取得します。

```
bool  OnnxGetOutputTypeInfo(
   long            onnx_handle,  // ONNXセッションハンドル
   long            index,        // パラメータインデックス
   OnnxTypeInfo&   typeinfo      // パラメータ型の記述
   );
```

パラメータ

onnx_handle

[in] OnnxCreateまたは OnnxCreateFromBufferで作成されたONNXセッションオブジェクトハンドル

index

[in] 0から始まる出力パラメータのインデックス

typeinfo

[out] 出力パラメータの型を記述するOnnxTypeInfo構造体

戻り値

成功の場合はtrue、それ以外の場合はfalse。エラーコードの取得には GetLastError() 関数が呼ばれます。

© 2000–2025, MetaQuotes Ltd.

OnnxSetInputShape

インデックスによってモデルの入力データの形状を設定します。

```
bool  OnnxSetInputShape(
   long           onnx_handle,   // ONNXセッションハンドル
   long           input_index,   // 入力パラメータインデックス
   const ulong&   shape[]        // 入力データの形状を記述する配列
   );
```

パラメータ

onnx_handle

[in] OnnxCreateまたは OnnxCreateFromBufferで作成されたONNXセッションオブジェクトハンドル

input_index

[in] 0から始まる入力パラメータのインデックス

shape

[in] 入力データ形状を記述する配列

戻り値

成功した場合は入力パラメータの名前、それ以外の場合はNULL。エラーコードの取得には GetLastError() 関数が呼ばれます。

例：

```
//---- モデルの入出力データの形状を指定する
   const long  ExtOutputShape[] = {1,1};
   const long  ExtInputShape [] = {1,10,4};
//--- モデルを作成する
   long handle=OnnxCreateFromBuffer(model,ONNX_DEBUG_LOGS);
//--- 入力データの形状を指定する
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("failed, OnnxSetInputShape error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- 出力データの形状を指定する
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("failed, OnnxSetOutputShape error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
```

参照

OnnxSetOutputShape

© 2000-2025, MetaQuotes Ltd.

OnnxSetOutputShape

インデックスによってモデルの出力データの形状を設定します。

```
bool  OnnxSetOutputShape(
   long           onnx_handle,    // ONNXセッションハンドル
   long           output_index,   // 出力パラメータインデックス
   const ulong&   shape[]         // 出力データの形状を記述する配列
   );
```

パラメータ

onnx_handle

[in] OnnxCreateまたは OnnxCreateFromBufferで作成されたONNXセッションオブジェクトハンドル

output_index

[in] 0から始まる出力パラメータのインデックス

shape

[in] 出力データ形状を記述する配列

戻り値

成功した場合は入力パラメータの名前、それ以外の場合はNULL。エラーコードの取得には GetLastError() 関数が呼ばれます。

例：

```
//---- モデルの入出力データの形状を指定する
   const long  ExtOutputShape[] = {1,1};
   const long  ExtInputShape [] = {1,10,4};
//--- モデルを作成する
   long handle=OnnxCreateFromBuffer(model,ONNX_DEBUG_LOGS);
//--- 入力データの形状を指定する
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("failed, OnnxSetInputShape error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- 出力データの形状を指定する
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("failed, OnnxSetOutputShape error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
```

参照

OnnxSetInputShape

© 2000-2025, MetaQuotes Ltd.

データ構造体

ONNXモデルでの操作には次のデータ構造が使用されます。

OnnxTypeInfo

この構造体は、ONNXモデルの入力または出力パラメータの型を記述します

```
struct OnnxTypeInfo
 {
  ENUM_ONNX_TYPE         type;          // パラメータ型
  OnnxTensorTypeInfo     tensor;        // テンソルの説明
  OnnxMapTypeInfo        map;           // マップの説明
  OnnxSequenceTypeInfo   sequence;      // シーケンスの説明
 };
```

テンソル(ONNX_TYPE_TENSOR)のみを入力として使用できます。この場合、OnnxTypeInfo::tensorフィールドのみに値が入力され、他のフィールドmapとsequence)は定義されていません。

3つのOnnxTypeInfo型(ONNX_TYPE_TENSOR、ONNX_TYPE_MAP、ONNX_TYPE_SEQUENCE)のうち1つだけを入力として使用できます。対応する部分構造(OnnxTypeInfo::tensor、OnnxTypeInfo::map、OnnxTypeInfo::sequence)は、型に応じて埋められます。

OnnxTensorTypeInfo

構造体は、ONNXモデルの入力または出力パラメータのテンソルを記述します

```
struct OnnxTensorTypeInfo
 {
  const ENUM_ONNX_DATA_TYPE  data_type;      // テンソルのデータ型
  const long                 dimensions[];   // テンソルの要素数
 };
```

OnnxMapTypeInfo

この構造体は、ONNXモデルの出力パラメータで取得されるマップを記述します。

```
struct OnnxMapTypeInfo
 {
  const ENUM_ONNX_DATA_TYPE   key_type;      // キーの型
  const OnnxTypeInfo&         value_type;    // 値の型
 };
```

OnnxSequenceTypeInfo

この構造体は、ONNXモデルの出力パラメータで取得されるシークエンスを記述します。

```
struct OnnxSequenceTypeInfo
 {
```

© 2000-2025, MetaQuotes Ltd.

```
  const OnnxTypeInfo&          value_type;       // シークエンスのデータ型
};
```

ENUM_ONNX_TYPE

ENUM_ONNX_TYPE列挙はモデルパラメータの型を記述します

ID	説明
ONNX_TYPE_UNKNOWN	不明
ONNX_TYPE_TENSOR	テンソル
ONNX_TYPE_SEQUENCE	シークエンス
ONNX_TYPE_MAP	マップ
ONNX_TYPE_OPAQUE	抽象（不透明）
ONNX_TYPE_SPARSETENSOR	スパーステンソル

ENUM_ONNX_DATA_TYPE

ENUM_ONNX_DATA_TYPE列挙は使用されるデータの型を記述します

ID	説明
ONNX_DATA_TYPE_UNDEFINED	未定義
ONNX_DATA_TYPE_FLOAT	float
ONNX_DATA_TYPE_INT8	8ビットint
ONNX_DATA_TYPE_UINT16	16ビットuint
ONNX_DATA_TYPE_INT16	16ビットint
ONNX_DATA_TYPE_INT32	32ビットint
ONNX_DATA_TYPE_INT64	64ビットint
ONNX_DATA_TYPE_STRING	string
ONNX_DATA_TYPE_BOOL	bool
ONNX_DATA_TYPE_FLOAT16	16ビットfloat
ONNX_DATA_TYPE_DOUBLE	double
ONNX_DATA_TYPE_UINT32	32ビットuint
ONNX_DATA_TYPE_UINT64	64ビットuint
ONNX_DATA_TYPE_COMPLEX64	64ビット複素数

© 2000–2025, MetaQuotes Ltd.

ID	説明
ONNX_DATA_TYPE_COMPLEX128	128ビット複素数
ONNX_DATA_TYPE_BFLOAT16	16ビットbfloat(Brain Floating Point)

ENUM_ONNX_FLAGS

ENUM_ONNX_FLAGS列挙はモデルの実行モードを記述します

ID	説明
ONNX_DEBUG_LOGS	出力デバッグログ
ONNX_NO_CONVERSION	自動変換を無効にし、ユーザーデータをそのまま使用します
ONNX_COMMON_FOLDER	Common¥Filesフォルダからモデルファイルを読み込みます。値はFILE_COMMONフラグと同じです。

ONNXモデルを使用する場合の配列変換

機械学習タスクでは、必ずしも高い計算精度が必要なわけではありません。計算を高速化するために、一部のモデルではFloat16やFloat8などの低精度のデータ型を使用します。ユーザーが関連データをモデルに入力できるようにするために、MQL5は、標準のMQL5型を特別なFP16型およびFP8型に変換する4つの特別な関数を提供します。

関数	アクション
ArrayToFP16	float型またはdouble型の配列を、指定された形式のushort型の配列にコピーします
ArrayToFP8	float型またはdouble型の配列を、指定された形式のuchar型の配列にコピーします
ArrayFromFP16	ushort型の配列を、指定された形式のfloat型またはdouble型の配列にコピーします
ArrayFromFP8	uchar型の配列を、指定された形式のfloat型またはdouble型の配列にコピーします

これらの配列変換関数は、以下の列挙で指定された特別な形式を使用します。

ENUM_FLOAT16_FORMAT

ENUM_FLOAT16_FORMAT列挙では2つのFP16型形式が説明されています。

ID	説明
FLOAT_FP16	標準16ビット形式（ハーフとも呼ばれる）
FLOAT_BFP16	特別なbf16浮動小数点形式

© 2000-2025, MetaQuotes Ltd.

これらの形式にはそれぞれ利点と制限があります。FLOAT16は精度が高くなりますが、より多くのストレージと計算リソースを必要とします。一方、BFLOAT16はデータ処理のパフォーマンスと効率が高くなりますが、精度が劣る可能性があります。

ENUM_FLOAT8_FORMAT

ENUM_FLOAT8_FORMAT列挙では4つのFP8型形式が説明されています。

FP8（8ビット浮動小数点）は、浮動小数点数を表すために使用されるデータ型の1つです。FP8では、各数値は8データビットで表され、通常は符号、指数、仮数の3つの要素に分割されます。この形式は精度とストレージ効率のバランスが取れているため、メモリと計算効率を必要とするアプリケーションにとって魅力的です。

ID	説明
FLOAT_FP8_E4M3FN	8ビット浮動小数点数、指数部に4ビット、仮数部に3ビット。通常は係数として使用されます。
FLOAT_FP8_E4M3FNUZ	8ビット浮動小数点数、指数部に4ビット、仮数部に3ビット。NaNをサポートしますが、負のゼロとInfはサポートしません。通常は係数として使用されます。
FLOAT_FP8_E5M2FN	8ビット浮動小数点数、指数部に5ビット、仮数部に2ビット。NaNとInfをサポートします。通常は勾配に使用されます。
FLOAT_FP8_E5M2FNUZ	8ビット浮動小数点数、指数部に5ビット、仮数部に2ビット。NaNをサポートしますが、負のゼロとInfはサポートしません。勾配に使用されます。

FP8の主な利点の1つは、大規模なデータセットの処理効率です。FP8はコンパクトな数値表現を採用することにより、メモリ要件を削減し、計算を高速化します。これは、大規模なデータセットを処理することが多い機械学習および人工知能アプリケーションでは特に重要です。

© 2000–2025, MetaQuotes Ltd.

標準ライブラリ

このセクションには、MQL5　標準ライブラリの技術的な詳細と、その全ての主要コンポーネントの説明が含まれています。

MQL5 標準ライブラリは MQL5 で書かれており、エンドユーザのプログラム（指標、スクリプト、エキスパート）の作成の簡易化のために設計されています。ライブラリは、ほとんどの MQL5 内部機能への便利なアクセスを提供します。

MQL5 標準ライブラリは、端末の作業ディレクトリの Include フォルダ内に配置されています。

セクション	ディレクトリ
数学	Include\Math\
OpenCL	Include\OpenCL\
基本クラスCObject	Include\
データ収集	Include\Arrays\
ジェネリックデータコレクション	Include\Generic\
ファイル	Include\Files\
文字列	Include\Strings\
グラフィックオブジェクト	Include\Objects\
カスタムグラフィックス	Include\Canvas\
3Dグラフィックス	Include\Canvas\
価格チャート	Include\Charts\
科学的チャート	Include\Graphics\
指標	Include\Indicators\
トレードクラス	Include\Trade\
ストラテジーモジュール	Include\Expert\
パネルとダイアログ	Include\Controls\

© 2000–2025, MetaQuotes Ltd.

数学

数学のさまざまな分野での計算を実行するために、複数のライブラリが提供されています。

- 統計 – 確率論の様々な分布を扱う関数
- ファジィ論理 – マムダニ及び菅野ファジィ推論システムを実装するライブラリ
- ALGLIB – データ解析（クラスタリング、決定木、線形回帰、ニューラルネットワーク）、微分方程式の解法、フーリエ変換、数値積分、最適化問題、統計解析などが含まれる

© 2000–2025, MetaQuotes Ltd.

統計

統計ライブラリは、基本的な統計分布を扱う便利な方法を提供します。

このライブラリでは、配布ごとに5つの関数が用意されています。

1. 確率密度の計算 – MathProbabilityDensityX()関数
2. 確率の計算 – MathCumulativeDistributionX()関数
3. 分布分位数の計算 – MathQuantileX()関数
4. 指定された分布を持つ乱数の生成 – MathRandomX()関数
5. 分布の理論的モーメントの計算 – MathMomentsX()関数

ライブラリは、個々の確率変数の計算に加えて、配列の同じ計算を実行する関数のオーバーロードも提供します。

- 統計的特性
- 正規分布
- 対数正規分布
- ベータ分布
- 非心ベータ分布
- ガンマ分布
- カイ2乗分布
- 非心カイ2乗分布
- 指数分布
- F分布
- 非心F分布
- t分布
- 非心t分布
- ロジスティック分布
- コーシー分布
- 一様分布
- ワイブル分布
- 二項分布
- 負の二項分布
- 幾何分布
- 超幾何分布
- ポアソン分布
- サブ関数

例：

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                      NormalDistributionExample.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- 正規分布を計算する関数をインクルードする
#include <Math\Stat\Normal.mqh>
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 正規分布のパラメータを設定する
   double mu=5.0;
   double sigma=1.0;
   PrintFormat("Normal distribution with parameters mu=%G and sigma=%G, calculation examples:",mu,sigma);
//--- 間隔を設定する
   double x1=mu-sigma;
   double x2=mu+sigma;
//--- 確率計算のための変数
   double cdf1,cdf2,probability;
//--- エラーコードを表す変数
   int error_code1,error_code2;
//--- 分布関数の値を計算する
   cdf1=MathCumulativeDistributionNormal(x1,mu,sigma,error_code1);
   cdf2=MathCumulativeDistributionNormal(x2,mu,sigma,error_code2);
//--- エラーコードを確認する
   if(error_code1==ERR_OK && error_code2==ERR_OK)
     {
      //--- 範囲内の確率変数の確率を計算する
      probability=cdf2-cdf1;
      //--- 結果を出力する
      PrintFormat("1. Calculate probability of a random variable within the range of %.5f<x<%.5f",x1,x2);
      PrintFormat("  Answer: Probability = %5.8f",probability);
     }

//--- Find the value range of random variable x, corresponding to the 95% confidence level
   probability=0.95; //  信頼確率を設定する
//--- 区間の境界での確率を設定する
   double p1=(1.0-probability)*0.5;
   double p2=probability+(1.0-probability)*0.5;
//--- 区間の境界を計算する
   x1=MathQuantileNormal(p1,mu,sigma,error_code1);
   x2=MathQuantileNormal(p2,mu,sigma,error_code2);
//--- エラーコードを確認する
   if(error_code1==ERR_OK && error_code2==ERR_OK)
     {
      //--- 結果を出力する
      PrintFormat("2. For confidence interval = %.2f, find the range of random variable",probability);
      PrintFormat("  Answer: range is  %5.8f <= x <=%5.8f",x1,x2);
     }

   PrintFormat("3. Compute the first 4 calculated and theoretical moments of the distribution");
```

© 2000–2025, MetaQuotes Ltd.

```
//--- 乱数の配列を生成し、最初の4次のモーメントを計算し、理論値と比較する
   int data_count=1000000;  // 値の数を設定し、配列を準備する
   double data[];
   ArrayResize(data,data_count);
//--- 乱数値を生成して配列に格納する
   for(int i=0; i<data_count; i++)
     {
      data[i]=MathRandomNormal(mu,sigma,error_code1);
     }
//--- 計算に使われる初めの値のインデックスとデータ量を設定する
   int start=0;
   int count=data_count;
//--- 生成された値の最初の4次のモーメントを計算する
   double mean=MathMean(data,start,count);
   double variance=MathVariance(data,start,count);
   double skewness=MathSkewness(data,start,count);
   double kurtosis=MathKurtosis(data,start,count);
//--- 理論的モーメントを表す変数
   double normal_mean=0;
   double normal_variance=0;
   double normal_skewness=0;
   double normal_kurtosis=0;
//--- 計算されたモーメントの値を表示する
   PrintFormat("            Mean            Variance          Skewness
Kurtosis");
   PrintFormat("Calculated  %.10f    %.10f      %.10f
%.10f",mean,variance,skewness,kurtosis);
//--- モーメントの理論値を計算し、得られた値と比較する
   if(MathMomentsNormal(mu,sigma,normal_mean,normal_variance,normal_skewness,normal_
kurtosis,error_code1))
     {
      PrintFormat("Theoretical %.10f    %.10f      %.10f
%.10f",normal_mean,normal_variance,normal_skewness,normal_kurtosis);
      PrintFormat("Difference  %.10f    %.10f      %.10f       %.10f",mean-
normal_mean,variance-normal_variance,skewness-normal_skewness,kurtosis-
normal_kurtosis);
     }
  }
```

© 2000–2025, MetaQuotes Ltd.

統計的特性

この関数グループは、配列要素の統計的特性を計算します。

- 平均、
- 差異、
- 歪み、
- 尖度、
- 中央値、
- 二乗平均平方根、
- 標準偏差

関数	説明
MathMean	配列要素の平均（1次モーメント）を計算します
MathVariance	配列要素の差異（2次モーメント）を計算します
MathSkewness	配列要素の歪み（3次モーメント）を計算します
MathKurtosis	配列要素の尖度（4次モーメント）を計算します
MathMoments	配列要素の最初の4次のモーメント（平均、分散、歪み度、尖度）を計算します
MathMedian	配列要素の中央値を計算します
MathStandardDeviation	配列要素の標準偏差を計算します
MathAverageDeviation	配列要素の平均絶対偏差を計算します

© 2000–2025, MetaQuotes Ltd.

MathMean

配列要素の平均（1次モーメント）を計算します。Rのmean()の類似体です。

```
double  MathMean(
   const double&  array[]                 // データ配列
   );
```

パラメータ

array

[in] 平均が計算されるデータを持つ配列

start=0

[in] 計算のための初期インデックス

count=WHOLE_ARRAY

[in] 計算される要素数

戻り値

配列要素の平均値。エラーの場合は、NaN（not a number）を返します。

© 2000-2025, MetaQuotes Ltd.

MathVariance

配列要素の差異（2次モーメント）を計算します。Rのvar()の類似体です。

```
double  MathVariance(
   const double&  array[]                // データ配列
   );
```

パラメータ

array

[in] 計算されるデータを持つ配列

start=0

[in] 計算のための初期インデックス

count=WHOLE_ARRAY

[in] 計算される要素数

戻り値

配列要素の差異。エラーの場合は、NaN（not a number）を返します。

© 2000–2025, MetaQuotes Ltd.

MathSkewness

配列要素の歪み（3次モーメント）を計算します。R（e1071 ライブラリ）のskewness()の類似体です。

```
double  MathSkewness(
   const double&  array[]                // データ配列
   );
```

パラメータ

array

[in] 計算されるデータを持つ配列

start=0

[in] 計算のための初期インデックス

count=WHOLE_ARRAY

[in] 計算される要素数

戻り値

配列要素の歪み。エラーの場合は、NaN（not a number）を返します。

© 2000-2025, MetaQuotes Ltd.

MathKurtosis

配列要素の尖度（4次モーメント）を計算します。R（e1071 ライブラリ）のkurtosis()の類似体です。

```
double  MathKurtosis(
   const double&  array[]                // データ配列
   );
```

パラメータ

array

[in] 計算されるデータを持つ配列

start=0

[in] 計算のための初期インデックス

count=WHOLE_ARRAY

[in] 計算される要素数

戻り値

配列要素の尖度。エラーの場合は、NaN（not a number）を返します。

免責事項

尖度の計算は、正規分布周囲の過剰尖度（過剰尖度=尖度-3）を用いて行われます。すなわち、正規分布の過剰尖度はゼロです。

予想値を中心とした分布のピークが鋭い場合は正、ピークが平らな場合は負です。

© 2000-2025, MetaQuotes Ltd.

MathMoments

配列要素の最初の4次のモーメント（平均、分散、歪み度、尖度）を計算します。

```
double  MathMoments(
   const double&  array[],              // データ配列
   double&        mean,                 // 平均（1次モーメント）
   double&        variance,             // 分散（2次モーメント）
   double&        skewness,             // 歪み（3次モーメント）
   double&        kurtosis,             // 尖度（4次モーメント）
   const int      start=0,              // 初期インデックス
   const int      count=WHOLE_ARRAY     // 要素数
   );
```

パラメータ

array

[in] 計算されるデータを持つ配列

mean

[out] 平均（1次モーメント）を表す変数

variance

[out] 分散（2次モーメント）を表す変数

skewness

[out] 歪み（3次モーメント）を表す変数

kurtosis

[out] 尖度（4次モーメント）を表す変数

start=0

[in] 計算のための初期インデックス

count=WHOLE_ARRAY

[in] 計算される要素数

戻り値

モーメントが正しく計算された場合はtrue、そうでない場合はfalseを返します。

免責事項

尖度の計算は、正規分布周囲の過剰尖度（過剰尖度=尖度-3）を用いて行われます。すなわち、正規分布の過剰尖度はゼロです。

予想値を中心とした分布のピークが鋭い場合は正、ピークが平らな場合は負です。

© 2000-2025, MetaQuotes Ltd.

MathMedian

配列要素の中央値を計算します。Rのmedian()の類似体です。

```
double  MathMedian(
   const double&  array[]                 // データ配列
   );
```

パラメータ

array

[in] 計算されるデータを持つ配列

start=0

[in] 計算のための初期インデックス

count=WHOLE_ARRAY

[in] 計算される要素数

戻り値

配列要素の中央値。エラーの場合は、NaN(not a number)を返します。

© 2000-2025, MetaQuotes Ltd.

MathStandardDeviation

配列要素の標準偏差を計算します。Rのsd()の類似体です。

```
double  MathStandardDeviation(
   const double&  array[]                // データ配列
   );
```

パラメータ

array

[in] 計算されるデータを持つ配列

start=0

[in] 計算のための初期インデックス

count=WHOLE_ARRAY

[in] 計算される要素数

戻り値

配列要素の標準偏差を計算します。エラーの場合は、NaN(not a number)を返します。

© 2000-2025, MetaQuotes Ltd.

MathAverageDeviation

配列要素の平均絶対偏差を計算します。Rのaad()の類似体です。

```
double  MathAverageDeviation(
   const double&  array[]                // データ配列
   );
```

パラメータ

array

[in] 計算されるデータを持つ配列

start=0

[in] 計算のための初期インデックス

count=WHOLE_ARRAY

[in] 計算される要素数

戻り値

配列要素の平均絶対偏差。エラーの場合は、NaN(not a number)を返します。

© 2000-2025, MetaQuotes Ltd.

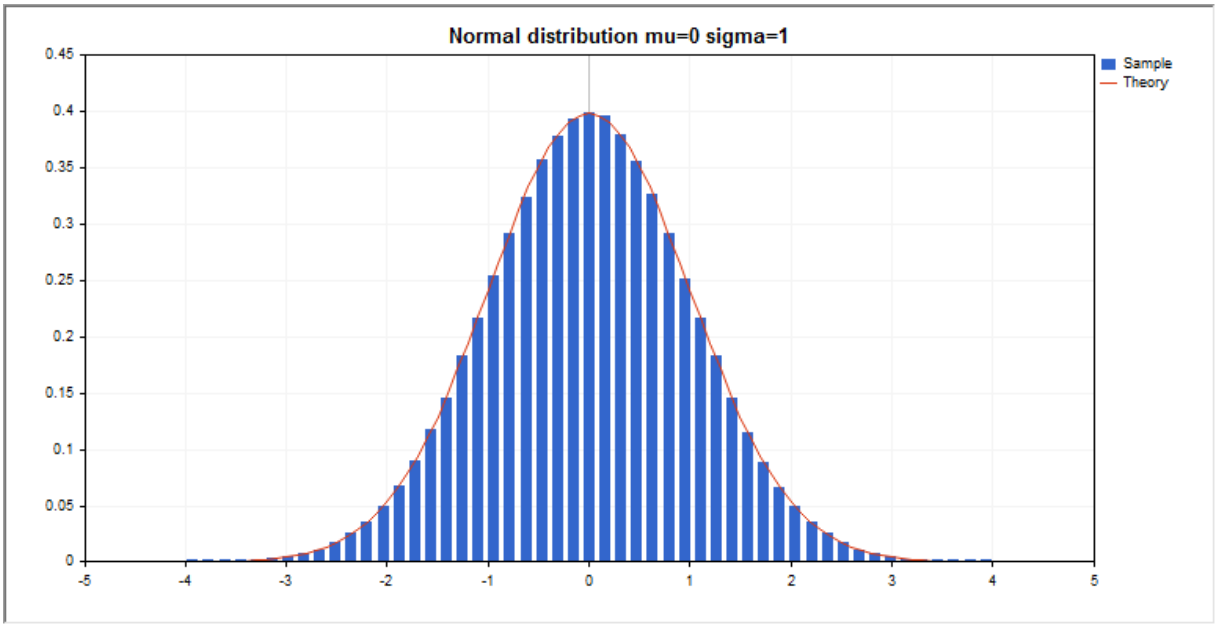
正規分布

このセクションには、正規分布を使用するための関数が含まれています。密度、確率、分位数を計算し、正規法則に従って分布した疑似乱数を生成することができます。正規分布は下記の式で定義されています。

$$f_{Normal}(x|\mu,\sigma)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

ここで

- x ー 確率変数の値
- μ ー 期待値
- σ ー 二乗平均平方根偏差



ライブラリは、個々の確率変数の計算に加えて、確率変数の配列を扱う能力も実装しています。

関数	説明
MathProbabilityDensityNormal	正規分布の確率密度関数を計算します
MathCumulativeDistributionNormal	正規分布関数の値を計算します
MathQuantileNormal	指定された確率に対する正規逆分布関数の値を計算します
MathRandomNormal	正規分布の法則に従って分布した擬似乱数変数/擬似乱数変数の配列を生成します
MathMomentsNormal	正規分布の最初の4次のモーメントの理論的数値を計算します

© 2000–2025, MetaQuotes Ltd.

例：

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Normal.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 入力パラメータ
input double mean_value=0;  // 期待値（平均）
input double std_dev=1;     // 二乗平均平方根偏差（標準偏差）
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 価格チャートを非表示にする
   ChartSetInteger(0,CHART_SHOW,false);
//--- 擬似乱数生成器を初期化する
   MathSrand(GetTickCount());
//--- 確率変数のサンプルを生成する
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // サンプルの値の数
   int ncells=51;          // ヒストグラムの間隔の数
   double x[];             // ヒストグラム間隔の中心
   double y[];             // 間隔内のサンプルからの値の数
   double data[];          // 確率変数値のサンプル
   double max,min;         // サンプルの最大値と最小値
//--- 正規分布からサンプルを取得する
   MathRandomNormal(mean_value,std_dev,n,data);
//--- ヒストグラムをプロットするためのデータを計算する
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 理論曲線をプロットするシーケンス境界とステップを取得する
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- [min、max]の間隔で理論的に計算されたデータを取得する
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNormal(x2,mean_value,std_dev,false,y2);
//--- 規模を設定する
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- チャートを出力する
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
```

© 2000-2025, MetaQuotes Ltd.

```
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Normal distribution mu=%G sigma=%
G",mean_value,std_dev));
   graphic.BackgroundMainSize(16);
//--- カーブをプロットする
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 今度は分布密度の理論曲線をプロットする
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
//--- カーブをプロットする
   graphic.CurvePlotAll();
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|   データセットの頻度を計算する                                   |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double
&frequency[],
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 間隔の中心を定義する
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 間隔内に入る頻度を記入する
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|   シーケンス生成の値を計算する                                   |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- 正規化の精度を得るためにシーケンスの絶対範囲を計算する
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- 最大値と最小値を指定された精度に正規化する
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 指定された精度に基づいてシーケンス生成ステップも設定される
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityNormal

確率変数xに対して、パラメータmu、sigmaを用いて正規分布の確率密度関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityNormal(
   const double  x,              // 確率変数の値
   const double  mu,             // 分布の平均パラメータ(期待値)
   const double  sigma,          // sigma parameter of the distribution (二乗平均平方根偏差)
   const bool    log_mode,       // 値の対数を計算する
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータmu、sigmaを用いて正規分布の確率密度関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityNormal(
   const double  x,              // 確率変数の値
   const double  mu,             // 分布の平均パラメータ(期待値)
   const double  sigma,          // sigma parameter of the distribution (二乗平均平方根偏差)
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対して、パラメータmu、sigmaを用いて正規分布の確率密度関数の値を計算します。エラーの場合は、falseを返します。Rのdnorm()の類似体です。

```
bool  MathProbabilityDensityNormal(
   const double&  x[],           // 確率変数の値を持つ配列
   const double   mu,            // 分布の平均パラメータ(期待値)
   const double   sigma,         // sigma parameter of the distribution (二乗平均平方根偏差)
   const bool     log_mode,      // 値の対数を計算する
   double&        result[]       // 確率密度関数の値の配列
   );
```

確率変数の配列x[]に対して、パラメータmu、sigmaを用いて正規分布の確率密度関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathProbabilityDensityNormal(
   const double&  x[],           // 確率変数の値を持つ配列
   const double   mu,            // 分布の平均パラメータ(期待値)
   const double   sigma,         // sigma parameter of the distribution (二乗平均平方根偏差)
   double&        result[]       // 確率密度関数の値の配列
   );
```

パラメータ

x

[in] 確率変数の値

© 2000-2025, MetaQuotes Ltd.

x[]

[in] 確率変数の値を持つ配列

mu

[in] 分布の平均パラメータ（期待値）

sigma

[in] sigma parameter of the distribution（二乗平均平方根偏差）

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率密度の自然対数を計算する。

error_code

[out] エラーコードを取得するための変数

result[]

[out] 確率密度関数の値を取得するための配列

© 2000–2025, MetaQuotes Ltd.

MathCumulativeDistributionNormal

確率変数xに対して、パラメータmu、sigmaを用いて正規分布の確率分布関数を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionNormal(
   const double  x,              // 確率変数の値
   const double  mu,             // 期待値
   const double  sigma,          // 二乗平均平方根偏差
   const bool    tail,           // tail計算のフラグ
   const bool    log_mode,       // 値の対数を計算する
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータmu、sigmaを用いて正規分布の確率分布関数を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionNormal(
   const double  x,              // 確率変数の値
   const double  mu,             // 期待値
   const double  sigma,          // 二乗平均平方根偏差
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータmu、sigmaを用いて正規分布の確率分布関数を計算します。エラーの場合は、falseを返します。Rのdnorm()の類似体です。

```
bool  MathCumulativeDistributionNormal(
   const double& x[],            // 確率変数の値を持つ配列
   const double  mu,             // 期待値
   const double  sigma,          // 二乗平均平方根偏差
   const bool    tail,           // tail計算のフラグ
   const bool    log_mode,       // 値の対数を計算する
   double&       result[]        // 確率関数の値の配列
   );
```

確率変数の配列x[]に対して、パラメータmu、sigmaを用いて正規分布の確率分布関数を計算します。エラーの場合は、falseを返します。

```
bool  MathCumulativeDistributionNormal(
   const double& x[],            // 確率変数の値を持つ配列
   const double  mu,             // 期待値
   const double  sigma,          // 二乗平均平方根偏差
   double&       result[]        // 確率関数の値の配列
   );
```

パラメータ

x

[in] 確率変数の値

x[]

© 2000-2025, MetaQuotes Ltd.

[in] 確率変数の値を持つ配列

mu

[in] 分布の平均パラメータ（期待値）

sigma

[in] sigma parameter of the distribution（二乗平均平方根偏差）

tail

[in] 計算のフラグ。tail=trueの場合、確率変数がx 以下である確率を計算する。

log_mode

[in] 値の対数を計算するためのフラグ。log_ mode=trueの場合、確率密度の自然対数を計算する。

error_code

[out] エラーコードを取得するための変数

result[]

[out] 確率関数の値を取得するための配列

© 2000–2025, MetaQuotes Ltd.

MathQuantileNormal

この関数は、指定されたprobabilityに対して、パラメータmu、sigmaを用いて正規逆分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileNormal(
   const double  probability,    // 確率変数の確率値
   const double  mu,             // 期待値
   const double  sigma,          // 二乗平均平方根偏差
   const bool    tail,           // tail計算のフラグ
   const bool    log_mode,       // 値の対数を計算する
   int&          error_code      // エラーコードを格納する変数
   );
```

この関数は、指定されたprobabilityに対して、パラメータmu、sigmaを用いて正規逆分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileNormal(
   const double  probability,    // 確率変数の確率値
   const double  mu,             // 期待値
   const double  sigma,          // 二乗平均平方根偏差
   int&          error_code      // エラーコードを格納する変数
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータmu、sigmaを用いて正規逆分布関数の値を計算します。エラーの場合は、falseを返します。Rのqnorm()の類似体です。

```
bool  MathQuantileNormal(
   const double& probability[],  // 確率変数の確率値を持つ配列
   const double  mu,             // 期待値
   const double  sigma,          // 二乗平均平方根偏差
   const bool    tail,           // tail計算のフラグ
   const bool    log_mode,       // 値の対数を計算する
   double&       result[]        // 分位数の値を持つ配列
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータmu、sigmaを用いて正規逆分布関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathQuantileNormal(
   const double& probability[],  // 確率変数の確率値を持つ配列
   const double  mu,             // 期待値
   const double  sigma,          // 二乗平均平方根偏差
   double&       result[]        // 分位数の値を持つ配列
   );
```

パラメータ

probability

[in] 確率変数の確率値

probability[]

© 2000–2025, MetaQuotes Ltd.

[in] 確率変数の確率値を持つ配列

mu

[in] 分布の平均パラメータ（期待値）

sigma

[in] sigma parameter of the distribution（二乗平均平方根偏差）

tail

[in] 計算のフラグ。falseの場合は1.0確率で計算する。

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率密度の自然対数を計算する。

error_code

[out] エラーコードを取得するための変数

result[]

[out] 分位数を取得するための配列

© 2000–2025, MetaQuotes Ltd.

MathRandomNormal

パラメータmu、sigmaを使って、正規分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、NaNを返します。

```
double  MathRandomNormal(
   const double  mu,              // 期待値
   const double  sigma,           // 二乗平均平方根偏差
   int&          error_code       // エラーコードを格納する変数
   );
```

パラメータmu、sigmaを使って、正規分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、falseを返します。Rのrnorm()の類似体です。

```
bool  MathRandomNormal(
   const double  mu,              // 期待値
   const double  sigma,           // 二乗平均平方根偏差
   const int     data_count,      // 必要なデータ量
   double&       result[]         // 擬似乱数変数を取得するための配列
   );
```

パラメータ

mu

[in] 分布の平均パラメータ（期待値）

sigma

[in] sigma parameter of the distribution（二乗平均平方根偏差）

data_count

[in] 取得される擬似乱数の数

error_code

[out] エラーコードを取得するための変数

result[]

[out] 擬似乱数の値を取得するための配列

© 2000-2025, MetaQuotes Ltd.

MathMomentsNormal

正規分布の最初の4次のモーメントの理論的数値を計算します。

```
double  MathMomentsNormal(
   const double  mu,              // 期待値
   const double  sigma,           // 二乗平均平方根偏差
   double&       mean,            // 平均を表す変数
   double&       variance,        // 分散を表す変数
   double&       skewness,        // 歪みを表す変数
   double&       kurtosis,        // 尖度を表す変数
   int&          error_code       // エラーコードを格納する変数
   );
```

パラメータ

mu

[in] 分布の平均パラメータ（期待値）

sigma

[in] sigma parameter of the distribution（二乗平均平方根偏差）

mean

[out] 平均値を取得するための変数

variance

[out] 分散を取得するための変数

skewness

[out] 歪みを取得するための変数

kurtosis

[out] 尖度を取得するための変数

error_code

[out] エラーコードを取得するための変数

戻り値

モーメントが正しく計算された場合はtrue、そうでない場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

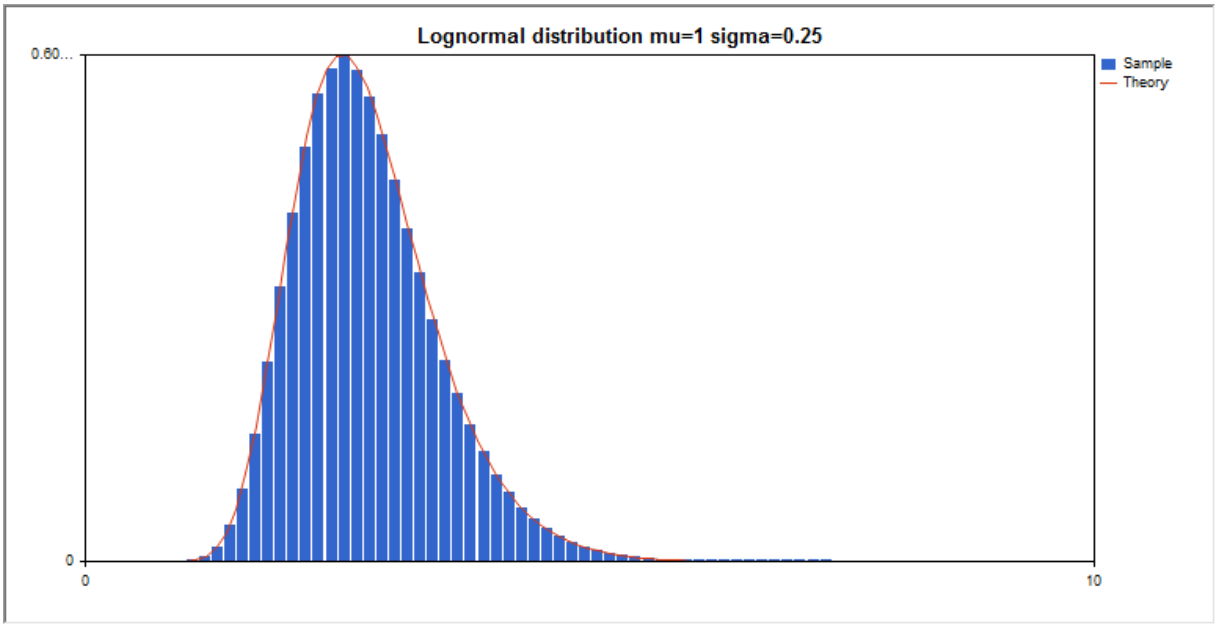
対数正規分布

このセクションには、対数正規分布を使用するための関数が含まれています。密度、確率、分位数を計算し、対数正規法則に従って分布した疑似乱数を生成することができます。対数正規分布は下記の式で定義されています。

$$f_{Lognormal}(x|\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln(x)-\mu)^2}{2\sigma^2}}$$

ここで

- x ー 確率変数の値
- μ ー 期待値の対数
- σ ー 二乗平均平方根偏差の対数



ライブラリは、個々の確率変数の計算に加えて、確率変数の配列を扱う能力も実装しています。

関数	説明
MathProbabilityDensityLognormal	対数正規分布の確率密度関数を計算します
MathCumulativeDistributionLognormal	対数正規分布関数の値を計算します
MathQuantileLognormal	指定された確率に対する対数正規逆分布関数の値を計算します
MathRandomLognormal	対数正規分布の法則に従って分布した擬似乱数変数/擬似乱数変数の配列を生成します

© 2000–2025, MetaQuotes Ltd.

関数	説明
MathMomentsLognormal	対数正規分布の最初の4次のモーメントの理論的数値を計算します

例：

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Lognormal.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 入力パラメータ
input double mean_value=1.0;  // 期待値の対数（対数平均）
input double std_dev=0.25;    // 二乗平均平方根偏差の対数（対数標準偏差）
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 価格チャートを非表示にする
   ChartSetInteger(0,CHART_SHOW,false);
//--- 擬似乱数生成器を初期化する
   MathSrand(GetTickCount());
//--- 確率変数のサンプルを生成する
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // サンプルの値の数
   int ncells=51;          // ヒストグラムの間隔の数
   double x[];             // ヒストグラム間隔の中心
   double y[];             // 間隔内のサンプルからの値の数
   double data[];          // 確率変数値のサンプル
   double max,min;         // サンプルの最大値と最小値
//--- 対数正規分布からサンプルを取得する
   MathRandomLognormal(mean_value,std_dev,n,data);
//--- ヒストグラムをプロットするためのデータを計算する
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 理論曲線をプロットするシーケンス境界とステップを取得する
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- [min、max]の間隔で理論的に計算されたデータを取得する
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityLognormal(x2,mean_value,std_dev,false,y2);
//--- 規模を設定する
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- チャートを出力する
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Lognormal distribution mu=%G sigma=%
G",mean_value,std_dev));
   graphic.BackgroundMainSize(16);
//--- Y軸の自動スケーリングを無効にする
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Max(theor_max);
   graphic.YAxis().Min(0);
//--- カーブをプロットする
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 今度は分布密度の理論曲線をプロットする
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- カーブをプロットする
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|   データセットの頻度を計算する                                   |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double
&frequency[],
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 間隔の中心を定義する
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 間隔内に入る頻度を記入する
   for(int i=0; i<size; i++)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  シーケンス生成の値を計算する                                    |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- 正規化の精度を得るためにシーケンスの絶対範囲を計算する
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- 最大値と最小値を指定された精度に正規化する
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 指定された精度に基づいてシーケンス生成ステップも設定される
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityLognormal

確率変数xに対して、パラメータmu、sigmaを用いて対数正規分布の確率密度関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityLognormal(
   const double  x,              // 確率変数の値
   const double  mu,             // 期待値の対数(対数平均)
   const double  sigma,          // 二乗平均平方根偏差の対数(対数標準偏差)
   const bool    log_mode,       // 値の対数を計算する。log_mode=trueの場合は、確率密度の自然対数が計算される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータmu、sigmaを用いて対数正規分布の確率密度関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityLognormal(
   const double  x,              // 確率変数の値
   const double  mu,             // 期待値の対数(対数平均)
   const double  sigma,          // 二乗平均平方根偏差の対数(対数標準偏差)
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対して、パラメータmu、sigmaを用いて対数正規分布の確率密度関数の値を計算します。エラーの場合は、NaNを返します。Rのdlnorm()の類似体です。

```
bool  MathProbabilityDensityLognormal(
   const double&  x[],           // 確率変数の値を持つ配列
   const double   mu,            // 期待値の対数(対数平均)
   const double   sigma,         // 二乗平均平方根偏差の対数(対数標準偏差)
   const bool     log_mode,      // 値の対数を計算する。log_mode=trueの場合は、確率密度の自然対数が計算される
   double&        result[]       // 確率密度関数の値の配列
   );
```

確率変数の配列x[]に対して、パラメータmu、sigmaを用いて対数正規分布の確率密度関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathProbabilityDensityLognormal(
   const double&  x[],           // 確率変数の値を持つ配列
   const double   mu,            // 期待値の対数(対数平均)
   const double   sigma,         // 二乗平均平方根偏差の対数(対数標準偏差)
   double&        result[]       // 確率密度関数の値の配列
   );
```

パラメータ

x

[in] 確率変数の値

x[]

© 2000-2025, MetaQuotes Ltd.

[in] 確率変数の値を持つ配列

mu

[in] 期待値の対数（対数平均）

sigma

[in] 二乗平均平方根偏差の対数（対数標準偏差）

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率密度の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率密度関数の値を取得するための配列

© 2000–2025, MetaQuotes Ltd.

MathCumulativeDistributionLognormal

確率変数xに対して、パラメータmu、sigmaを用いて対数正規分布の確率分布関数を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionLognormal(
   const double  x,              // 確率変数の値
   const double  mu,             // 期待値の対数(対数平均)
   const double  sigma,          // 二乗平均平方根偏差の対数(対数標準偏差)
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算する。log_mode=trueの場合は、確率の自然対数が返される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータmu、sigmaを用いて対数正規分布の確率分布関数を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionLognormal(
   const double  x,              // 確率変数の値
   const double  mu,             // 期待値の対数(対数平均)
   const double  sigma,          // 二乗平均平方根偏差の対数(対数標準偏差)
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対して、パラメータmu、sigmaを用いて対数正規分布の確率分布関数を計算します。エラーの場合は、falseを返します。Rのplnorm()の類似体です。

```
bool  MathCumulativeDistributionLognormal(
   const double&  x[],           // 確率変数の値を持つ配列
   const double   mu,            // 期待値の対数(対数平均)
   const double   sigma,         // 二乗平均平方根偏差の対数(対数標準偏差)
   const bool     tail,          // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool     log_mode,      // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される
   double&        result[]       // 確率関数の値の配列
   );
```

<t0>確率変数の配列x[]に対して、パラメータmu、sigmaを用いて対数正規分布の確率分布関数を計算します。エラーの場合は、falseを返します。

```
bool  MathCumulativeDistributionLognormal(
   const double&  x[],           // 確率変数の値を持つ配列
   const double   mu,            // 期待値の対数(対数平均)
   const double   sigma,         // 二乗平均平方根偏差の対数(対数標準偏差)
   double&        result[]       // 確率関数の値の配列
   );
```

パラメータ

© 2000-2025, MetaQuotes Ltd.

x

[in] 確率変数の値

x[]

[in] 確率変数の値を持つ配列

mu

[in] 期待値の対数（対数平均）

sigma

[in] 二乗平均平方根偏差の対数（対数標準偏差）

tail

[in] 計算のフラグ。true の場合、確率変数がx 以下である確率を計算する。

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率関数の値を取得するための配列

© 2000–2025, MetaQuotes Ltd.

MathQuantileLognormal

この関数は、指定されたprobabilityに対して、パラメータmu、sigmaを用いて対数正規逆分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileLognormal(
   const double  probability,     // 確率変数の発生確率値
   const double  mu,              // 期待値の対数(対数平均)
   const double  sigma,           // 二乗平均平方根偏差の対数(対数標準偏差)
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定されたprobabilityに対して、パラメータmu、sigmaを用いて対数正規逆分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileLognormal(
   const double  probability,     // 確率変数の発生確率値
   const double  mu,              // 期待値の対数(対数平均)
   const double  sigma,           // 二乗平均平方根偏差の対数(対数標準偏差)
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータmu、sigmaを用いて対数正規逆分布関数の値を計算します。エラーの場合は、falseを返します。Rのqlnorm()の類似体です。

```
bool  MathQuantileLognormal(
   const double&  probability[],  // 確率変数の確率値を持つ配列
   const double   mu,             // 期待値の対数(対数平均)
   const double   sigma,          // 二乗平均平方根偏差の対数(対数標準偏差)
   const bool     tail,           // 計算のフラグ。false の場合は1.0確率で計算する
   const bool     log_mode,       // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   double&        result[]        // 分位数の値を持つ配列
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータmu、sigmaを用いて対数正規逆分布関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathQuantileLognormal(
   const double&  probability[],  // 確率変数の確率値を持つ配列
   const double   mu,             // 期待値の対数(対数平均)
   const double   sigma,          // 二乗平均平方根偏差の対数(対数標準偏差)
   double&        result[]        // 分位数の値を持つ配列
   );
```

パラメータ

probability

© 2000-2025, MetaQuotes Ltd.

[in] 確率変数の発生確率値

probability[]

[in] 確率変数の確率値を持つ配列

mu

[in] 期待値の対数（対数平均）

sigma

[in] 二乗平均平方根偏差の対数（対数標準偏差）

tail

[in] 計算のフラグ。falseの場合は1.0確率で計算する

log_mode

[in] 計算のフラグ。log_mode=trueの場合、計算はExp(probability)に対して行われる

error_code

[out] エラーコードを格納する変数

result[]

[out] 分位数の値を持つ配列

© 2000-2025, MetaQuotes Ltd.

MathRandomLognormal

パラメータmu、sigmaを使って、対数正規分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、NaNを返します。

```
double  MathRandomLognormal(
   const double  mu,              // 期待値の対数(対数平均)
   const double  sigma,           // 二乗平均平方根偏差の対数(対数標準偏差)
   int&          error_code       // エラーコードを格納する変数
   );
```

パラメータmu、sigmaを使って、対数正規分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、falseを返します。Rのrlnorm()の類似体です。

```
double  MathRandomLognormal(
   const double  mu,              // 期待値の対数(対数平均)
   const double  sigma,           // 二乗平均平方根偏差の対数(対数標準偏差)
   const int     data_count,      // 必要なデータ量
   double&       result[]         // 擬似乱数変数の値の配列
   );
```

パラメータ

mu

[in] 期待値の対数(対数平均)

sigma

[in] 二乗平均平方根偏差の対数(対数標準偏差)

data_count

[out] 必要なデータ量

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率変数の値を持つ配列

© 2000-2025, MetaQuotes Ltd.

MathMomentsLognormal

対数正規分布の最初の4次のモーメントの理論的数値を計算します。モーメントの計算が成功した場合はtrue、そうでない場合はfalseを返します。

```
double  MathMomentsLognormal(
   const double  mu,              // 期待値の対数(対数平均)
   const double  sigma,           // 二乗平均平方根偏差の対数(対数標準偏差)
   double&       mean,            // 平均を表す変数
   double&       variance,        // 分散を表す変数
   double&       skewness,        // 歪みを表す変数
   double&       kurtosis,        // 尖度を表す変数
   int&          error_code       // エラーコードを格納する変数
   );
```

パラメータ

mu

[in] 期待値の対数(対数平均)

sigma

[in] 二乗平均平方根偏差の対数(対数標準偏差)

mean

[in] 平均を表す変数.

variance

[out] 分散を表す変数.

skewness

[out] 歪みを表す変数.

kurtosis

[out] 尖度を表す変数.

error code

[out] エラーコードを格納する変数

戻り値

モーメントが正しく計算された場合はtrue、そうでない場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

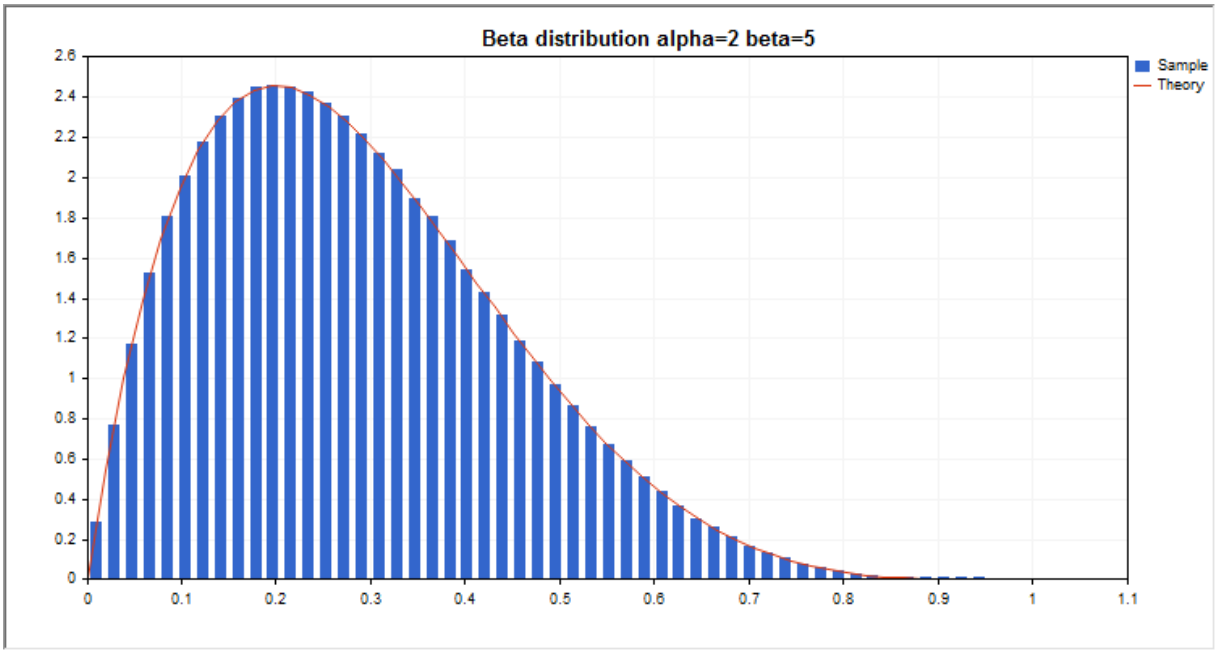
ベータ分布

このセクションには、ベータ分布を使用するための関数が含まれています。密度、確率、分位数を計算し、対応する法則に従って分布した疑似乱数を生成することができます。ベータ分布は下記の式で定義されています。

$$f_{Beta}(x \mid a,b) = \frac{1}{B(a,b)} x^{a-1}(1-x)^{b-1}$$

ここで

- x — 確率変数の値
- a — ベータ分布の1番目のパラメータ
- b — ベータ分布の2番目のパラメータ



ライブラリは、個々の確率変数の計算に加えて、確率変数の配列を扱う能力も実装しています。

関数	説明
MathProbabilityDensityBeta	ベータ分布の確率密度関数を計算します
MathCumulativeDistributionBeta	ベータ確率分布関数の値を計算します
MathQuantileBeta	指定された確率に対するベータ逆分布関数の値を計算します
MathRandomBeta	ベータ分布に従って分布した擬似乱数変数/擬似乱数変数の配列を生成します
MathMomentsBeta	ベータ分布の最初の4次のモーメントの理論的数値を計算します

© 2000–2025, MetaQuotes Ltd.

例：

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Beta.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 入力パラメータ
input double alpha=2;    // ベータ分布の1番目のパラメータ(形状1)
input double beta=5;     // ベータ分布の2番目のパラメータ（形状2）
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 価格チャートを非表示にする
   ChartSetInteger(0,CHART_SHOW,false);
//--- 擬似乱数生成器を初期化する
   MathSrand(GetTickCount());
//--- 確率変数のサンプルを生成する
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // サンプルの値の数
   int ncells=51;          // ヒストグラムの間隔の数
   double x[];             // ヒストグラム間隔の中心
   double y[];             // 間隔内のサンプルからの値の数
   double data[];          // 確率変数値のサンプル
   double max,min;         // サンプルの最大値と最小値
//--- ベータ分布からサンプルを取得す
   MathRandomBeta(alpha,beta,n,data);
//--- ヒストグラムをプロットするためのデータを計算する
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 理論曲線をプロットするシーケンス境界とステップを取得する
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- [min、max]の間隔で理論的に計算されたデータを取得する
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityBeta(x2,alpha,beta,false,y2);
//--- 規模を設定する
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- チャートを出力する
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
```

© 2000-2025, MetaQuotes Ltd.

```
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Beta distribution alpha=%G beta=%
G",alpha,beta));
   graphic.BackgroundMainSize(16);
//--- カーブをプロットする
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 今度は分布密度の理論曲線をプロットする
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- カーブをプロットする
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|   データセットの頻度を計算する                                   |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double
&frequency[],
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 間隔の中心を定義する
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 間隔内に入る頻度を記入する
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|   シーケンス生成の値を計算する                                   |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- 正規化の精度を得るためにシーケンスの絶対範囲を計算する
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- 最大値と最小値を指定された精度に正規化する
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 指定された精度に基づいてシーケンス生成ステップも設定される
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityBeta

確率変数xに対して、パラメータa、bを用いてベータ分布の確率密度関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityBeta(
   const double  x,              // 確率変数の値
   const double  a,              // ベータ分布の1番目のパラメータ(形式1)
   const double  b,              // ベータ分布の2番目のパラメータ(形式2)
   const bool    log_mode,       // 値の対数を計算する。log_mode=trueの場合は、確率密度の自然対数が計算される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータa、bを用いてベータ分布の確率密度関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityBeta(
   const double  x,              // 確率変数の値
   const double  a,              // ベータ分布の1番目のパラメータ(形式1)
   const double  b,              // ベータ分布の2番目のパラメータ(形式2)
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対して、パラメータa、bを用いてベータ分布の確率密度関数の値を計算します。エラーの場合は、falseを返します。Rのdbeta()の類似体です。

```
bool  MathProbabilityDensityBeta(
   const double& x[],            // 確率変数の値を持つ配列
   const double  a,              // ベータ分布の1番目のパラメータ(形式1)
   const double  b,              // ベータ分布の2番目のパラメータ(形式2)
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率密度の自然対数が計算される
   double&       result[]        // 確率密度関数の値の配列
   );
```

確率変数の配列x[]に対して、パラメータa、bを用いてベータ分布の確率密度関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathProbabilityDensityBeta(
   const double& x[],            // 確率変数の値を持つ配列
   const double  a,              // ベータ分布の1番目のパラメータ(形式1)
   const double  b,              // ベータ分布の2番目のパラメータ(形式2)
   double&       result[]        // 確率密度関数の値の配列
   );
```

パラメータ

x

[in] 確率変数の値

x[]

© 2000-2025, MetaQuotes Ltd.

[in] 確率変数の値を持つ配列

a

[in] ベータ分布の1番目のパラメータ（形状1）

b

[in] ベータ分布の2番目のパラメータ（形状2）

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率密度の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率密度関数の値の配列

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionBeta

確率変数xに対して、パラメータa、bを用いてベータ分布の確率分布関数を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionBeta(
   const double  x,              // 確率変数の値
   const double  a,              // ベータ分布の1番目のパラメータ(形式1)
   const double  b,              // ベータ分布の2番目のパラメータ(形式2)
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算する。log_mode=trueの場合は、確率の自然対数が返される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータa、bを用いてベータ分布の確率分布関数を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionBeta(
   const double  x,              // 確率変数の値
   const double  a,              // ベータ分布の1番目のパラメータ(形式1)
   const double  b,              // ベータ分布の2番目のパラメータ(形式2)
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数x []の配列に対して、パラメータa、bを用いてベータ分布の確率分布関数を計算します。エラーの場合は、falseを返します。Rのpbeta()の類似体です。

```
bool  MathCumulativeDistributionBeta(
   const double& x[],            // 確率変数の値を持つ配列
   const double  a,              // ベータ分布の1番目のパラメータ(形式1)
   const double  b,              // ベータ分布の2番目のパラメータ(形式2)
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される
   double&       result[]        // 確率関数の値の配列
   );
```

<t0>確率変数x []の配列に対して、パラメータa、bを用いてベータ分布の確率分布関数を計算します。エラーの場合は、falseを返します。

```
bool  MathCumulativeDistributionBeta(
   const double& x[],            // 確率変数の値を持つ配列
   const double  a,              // ベータ分布の1番目のパラメータ(形式1)
   const double  b,              // ベータ分布の2番目のパラメータ(形式2)
   double&       result[]        // 確率関数の値の配列
   );
```

パラメータ

© 2000-2025, MetaQuotes Ltd.

x

[in] 確率変数の値

x[]

[in] 確率変数の値を持つ配列

a

[in] ベータ分布の1番目のパラメータ（形状1）

b

[in] ベータ分布の2番目のパラメータ（形状2）

tail

[in] 計算のフラグ。true の場合、確率変数がx 以下である確率を計算する。

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率関数の値の配列

© 2000–2025, MetaQuotes Ltd.

MathQuantileBeta

この関数は、指定されたprobabilityに対して、パラメータa、bを用いてベータ逆分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileBeta(
   const double  probability,     // 確率変数の発生確率値
   const double  a,               // ベータ分布の1番目のパラメータ(形式1)
   const double  b,               // ベータ分布の2番目のパラメータ(形式2)
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定されたprobabilityに対して、パラメータa、bを用いてベータ逆分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileBeta(
   const double  probability,     // 確率変数の発生確率値
   const double  a,               // ベータ分布の1番目のパラメータ(形式1)
   const double  b,               // ベータ分布の2番目のパラメータ(形式2)
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定された確率値の配列*probability[]* に対して、パラメータa、bを用いてベータ逆分布関数の値を計算します。エラーの場合は、falseを返します。Rのqbeta()の類似体です。

```
double  MathQuantileBeta(
   const double&  probability[], // 確率変数の確率値を持つ配列
   const double  a,               // ベータ分布の1番目のパラメータ(形式1)
   const double  b,               // ベータ分布の2番目のパラメータ(形式2)
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   double&       result[]         // 分位数の値を持つ配列
   );
```

この関数は、指定された確率値の配列*probability[]* に対して、パラメータa、bを用いてベータ逆分布関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathQuantileBeta(
   const double& probability[],  // 確率変数の確率値を持つ配列
   const double  a,               // ベータ分布の1番目のパラメータ(形式1)
   const double  b,               // ベータ分布の2番目のパラメータ(形式2)
   double&       result[]         // 分位数の値を持つ配列
   );
```

パラメータ

probability

© 2000-2025, MetaQuotes Ltd.

[in] 確率変数の確率値

probability[]

[in] 確率変数の確率値を持つ配列

a

[in] ベータ分布の1番目のパラメータ（形式1）

b

[in] ベータ分布の2番目のパラメータ（形式2）

tail

[in] 計算のフラグ。lower_tail=falseの場合は1.0確率で計算する

log_mode

[in] 計算のフラグ。log_mode=trueの場合、計算はExp(probability)に対して行われる

error_code

[out] エラーコードを取得するための変数

result[]

[out] 分位数の値を持つ配列

© 2000-2025, MetaQuotes Ltd.

MathRandomBeta

パラメータa、bを使って、ベータ分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、NaNを返します。

```
double  MathRandomBeta(
   const double  a,              // ベータ分布の1番目のパラメータ(形式1)
   const double  b,              // ベータ分布の2番目のパラメータ(形式2)
   int&          error_code      // エラーコードを格納する変数
   );
```

パラメータa、bを使って、ベータ分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、falseを返します。Rのrbeta()の類似体です。

```
bool  MathRandomBeta(
   const double  a,               // ベータ分布の1番目のパラメータ(形式1)
   const double  b,               // ベータ分布の2番目のパラメータ(形式2)
   const int     data_count,      // 必要なデータ量
   double&       result[]         // 擬似乱数変数を取得するための配列
   );
```

パラメータ

a

[in] ベータ分布の1番目のパラメータ(形式1)

b

[in] ベータ分布の2番目のパラメータ(形式2)

data_count

[in] 取得される擬似乱数の数

error_code

[out] エラーコードを格納する変数

result[]

[out] 擬似乱数の値を取得するための配列

© 2000-2025, MetaQuotes Ltd.

MathMomentsBeta

ベータ分布の最初の4次のモーメントの理論的数値を計算します。

```
double  MathMomentsBeta(
   const double  a,              // ベータ分布の1番目のパラメータ(形式1)
   const double  b,              // ベータ分布の2番目のパラメータ(形式2)
   double&       mean,           // 平均を表す変数
   double&       variance,       // 分散を表す変数
   double&       skewness,       // 歪みを表す変数
   double&       kurtosis,       // 尖度を表す変数
   int&          error_code      // エラーコードを表す変数
   );
```

パラメータ

a

[in] ベータ分布の1番目のパラメータ(形式1)

b

[in] ベータ分布の2番目のパラメータ(形式2)

mean

[out] 平均値を取得するための変数

variance

[out] 分散を取得するための変数

skewness

[out] 歪みを取得するための変数

kurtosis

[out] 尖度を取得するための変数

error_code

[out] エラーコードを取得するための変数

戻り値

モーメントが正しく計算された場合はtrue、そうでない場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

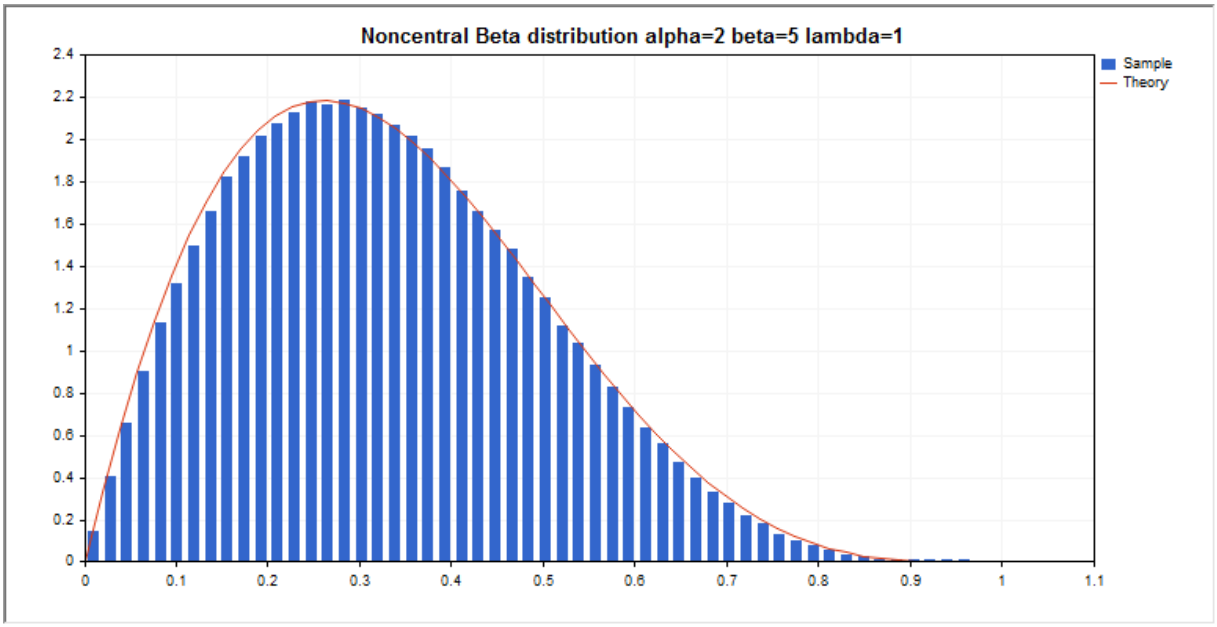
非心ベータ分布

このセクションには、非心ベータ分布を使用するための関数が含まれています。密度、確率、分位数を計算し、対応する法則に従って分布した疑似乱数を生成することができます。非心ベータ分布は下記の式で定義されています。

$$f_{NoncentralBeta}(x|a,b,\lambda)=\sum_{r=0}^{\infty}e^{-\frac{\lambda}{2}}\frac{\left(\frac{\lambda}{2}\right)^{r}}{r!}\frac{x^{a+r-1}(1-x)^{b-1}}{B(a+r,b)}$$

ここで

- x — 確率変数の値
- a — ベータ分布の1番目のパラメータ
- b — ベータ分布の2番目のパラメータ
- λ — 非心パラメータ



ライブラリは、個々の確率変数の計算に加えて、確率変数の配列を扱う能力も実装しています。

関数	説明
MathProbabilityDensityNoncentralBeta	非心ベータ分布の確率密度関数を計算します
MathCumulativeDistributionNoncentralBeta	非心ベータ確率分布関数の値を計算します
MathQuantileNoncentralBeta	指定された確率に対する非心ベータ逆分布関数の値を計算します

© 2000–2025, MetaQuotes Ltd.

関数	説明
MathRandomNoncentralBeta	非心ベータ分布に従って分布した擬似乱数変数/擬似乱数変数の配列を生成します
MathMomentsNoncentralBeta	非心ベータ分布の最初の4次のモーメントの理論的数値を計算します

例：

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\NoncentralBeta.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 入力パラメータ
input double a_par=2;     // ベータ分布の1番目のパラメータ(形状1)
input double b_par=5;     // ベータ分布の2番目のパラメータ（形状2）
input double l_par=1;     // 非心パラメータ（lambda）
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 価格チャートを非表示にする
   ChartSetInteger(0,CHART_SHOW,false);
//--- 擬似乱数生成器を初期化する
   MathSrand(GetTickCount());
//--- 確率変数のサンプルを生成する
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // サンプルの値の数
   int ncells=53;          // ヒストグラムの間隔の数
   double x[];             // ヒストグラム間隔の中心
   double y[];             // 間隔内のサンプルからの値の数
   double data[];          // 確率変数値のサンプル
   double max,min;         // サンプルの最大値と最小値
//--- 非心ベータ分布からサンプルを取得す
   MathRandomNoncentralBeta(a_par,b_par,l_par,n,data);
//--- ヒストグラムをプロットするためのデータを計算する
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 理論曲線をプロットするシーケンス境界とステップを取得する
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- [min、max]の間隔で理論的に計算されたデータを取得する
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNoncentralBeta(x2,a_par,b_par,l_par,false,y2);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 規模を設定する
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- チャートを出力する
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Noncentral Beta distribution alpha=%G beta=%
G lambda=%G",
                           a_par,b_par,l_par));
   graphic.BackgroundMainSize(16);
//--- カーブをプロットする
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 今度は分布密度の理論曲線をプロットする
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- カーブをプロットする
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  データセットの頻度を計算する                                    |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double
&frequency[],
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 間隔の中心を定義する
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 間隔内に入る頻度を記入する
   for(int i=0; i<size; i++)
```

© 2000–2025, MetaQuotes Ltd.

```
      {
       int ind=int((data[i]-minv)/width);
       if(ind>=cells) ind=cells-1;
       frequency[ind]++;
      }
   return (true);
  }
//+------------------------------------------------------------------+
//|   シーケンス生成の値を計算する                                   |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- 正規化の精度を得るためにシーケンスの絶対範囲を計算する
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- 最大値と最小値を指定された精度に正規化する
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 指定された精度に基づいてシーケンス生成ステップも設定される
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityNoncentralBeta

確率変数xに対して、パラメータa、b、lambdaを用いて非心ベータ分布の確率密度関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityNoncentralBeta(
   const double  x,              // 確率変数の値
   const double  a,              // ベータ分布の1番目のパラメータ(形式1)
   const double  b,              // ベータ分布の2番目のパラメータ(形式2)
   const double  lambda,         // 非心パラメータ
   const bool    log_mode,       // 値の対数を計算する。log_mode=trueの場合は、確率密度の自然対数が計算される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータa、b、lambdaを用いて非心ベータ分布の確率密度関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityNoncentralBeta(
   const double  x,              // 確率変数の値
   const double  a,              // ベータ分布の1番目のパラメータ(形式1)
   const double  b,              // ベータ分布の2番目のパラメータ(形式2)
   const double  lambda,         // 非心パラメータ
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対して、パラメータa、b、lambdaを用いて非心ベータ分布の確率密度関数の値を計算します。エラーの場合は、falseを返します。Rのdbeta()の類似体です。

```
bool  MathProbabilityDensityNoncentralBeta(
   const double& x[],            // 確率変数の値を持つ配列
   const double  a,              // ベータ分布の1番目のパラメータ(形式1)
   const double  b,              // ベータ分布の2番目のパラメータ(形式2)
   const double  lambda,         // 非心パラメータ
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率密度の自然対数が計算される
   double&       result[]        // 確率密度関数の値の配列
   );
```

<t0>確率変数の配列x[]に対して、パラメータa、b、lambdaを用いて非心ベータ分布の確率密度関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathProbabilityDensityNoncentralBeta(
   const double& x[],            // 確率変数の値を持つ配列
   const double  a,              // ベータ分布の1番目のパラメータ(形式1)
   const double  b,              // ベータ分布の2番目のパラメータ(形式2)
   const double  lambda,         // 非心パラメータ
   double&       result[]        // 確率密度関数の値の配列
   );
```

パラメータ

© 2000-2025, MetaQuotes Ltd.

x

[in] 確率変数の値

x[]

[in] 確率変数の値を持つ配列

a

[in] ベータ分布の1番目のパラメータ（形状1）

b

[in] ベータ分布の2番目のパラメータ（形状2）

lambda

[in] 非心パラメータ

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率密度の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率密度関数の値の配列

© 2000–2025, MetaQuotes Ltd.

MathCumulativeDistributionNoncentralBeta

確率変数xに対して、パラメータa、b、lambdaを用いて非心ベータ分布の確率分布関数を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionNoncentralBeta(
   const double  x,              // 確率変数の値
   const double  a,              // ベータ分布の1番目のパラメータ(形式1)
   const double  b,              // ベータ分布の2番目のパラメータ(形式2)
   const double  lambda,         // 非心パラメータ
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算する。log_mode=trueの場合は、確率の自然対数が返される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータa、b、lambdaを用いて非心ベータ分布の確率分布関数を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionNoncentralBeta(
   const double  x,              // 確率変数の値
   const double  a,              // ベータ分布の1番目のパラメータ(形式1)
   const double  b,              // ベータ分布の2番目のパラメータ(形式2)
   const double  lambda,         // 非心パラメータ
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数x []の配列に対して、パラメータa、b、lambdaを用いて非心ベータ分布の確率分布関数を計算します。エラーの場合は、falseを返します。Rのpbeta()の類似体です。

```
bool  MathCumulativeDistributionNoncentralBeta(
   const double& x[],            // 確率変数の値を持つ配列
   const double  a,              // ベータ分布の1番目のパラメータ(形式1)
   const double  b,              // ベータ分布の2番目のパラメータ(形式2)
   const double  lambda,         // 非心パラメータ
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される
   double&       result[]        // 確率関数の値の配列
   );
```

確率変数x []の配列に対して、パラメータa、b、lambdaを用いて非心ベータ分布の確率分布関数を計算します。エラーの場合は、falseを返します。

```
bool  MathCumulativeDistributionNoncentralBeta(
   const double& x[],            // 確率変数の値を持つ配列
   const double  a,              // ベータ分布の1番目のパラメータ(形式1)
   const double  b,              // ベータ分布の2番目のパラメータ(形式2)
   const double  lambda,         // 非心パラメータ
```

© 2000-2025, MetaQuotes Ltd.

```
   double&        result[]         // 確率関数の値の配列
   );
```

パラメータ

x

[in] 確率変数の値

x[]

[in] 確率変数の値を持つ配列

a

[in] ベータ分布の1番目のパラメータ（形状1）

b

[in] ベータ分布の2番目のパラメータ（形状2）

lambda

[in] 非心パラメータ

tail

[in] 計算のフラグ。true の場合、確率変数がx 以下である確率を計算する。

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率関数の値の配列

© 2000-2025, MetaQuotes Ltd.

MathQuantileNoncentralBeta

パラメータa、b、lambdaを用いて、確率変数probabilityの出現確率に対する非中心的ベータ分布の逆確率分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileNoncentralBeta(
   const double  probability,     // 確率変数の発生確率値
   const double  a,               // ベータ分布の1番目のパラメータ(形式1)
   const double  b,               // ベータ分布の2番目のパラメータ(形式2)
   const double  lambda,          // 非心パラメータ
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   int&          error_code       // エラーコードを格納する変数
   );
```

パラメータa、b、lambdaを用いて、確率変数probabilityの出現確率に対する非中心的ベータ分布の逆確率分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileNoncentralBeta(
   const double  probability,     // 確率変数の発生確率値
   const double  a,               // ベータ分布の1番目のパラメータ(形式1)
   const double  b,               // ベータ分布の2番目のパラメータ(形式2)
   const double  lambda,          // 非心パラメータ
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータa、b、lambdaを用いて非心ベータ逆分布関数の値を計算します。エラーの場合は、falseを返します。Rの qbeta()の類似体です。

```
double  MathQuantileNoncentralBeta(
   const double&  probability[],  // 確率変数の確率値を持つ配列
   const double  a,               // ベータ分布の1番目のパラメータ(形式1)
   const double  b,               // ベータ分布の2番目のパラメータ(形式2)
   const double  lambda,          // 非心パラメータ
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   double&       result[]         // 分位数の値を持つ配列
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータa、b、lambdaを用いて非心ベータ逆分布関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathQuantileNoncentralBeta(
   const double& probability[],  // 確率変数の確率値を持つ配列
   const double  a,              // ベータ分布の1番目のパラメータ(形式1)
   const double  b,              // ベータ分布の2番目のパラメータ(形式2)
   const double  lambda,         // 非心パラメータ
   double&       result[]        // 分位数の値を持つ配列
   );
```

© 2000-2025, MetaQuotes Ltd.

パラメータ

probability

[in] 確率変数の確率値

probability[]

[in] 確率変数の確率値を持つ配列

a

[in] ベータ分布の1番目のパラメータ（形式1）

b

[in] ベータ分布の2番目のパラメータ（形式2）

lambda

[in] 非心パラメータ

tail

[in] 計算のフラグ。falseの場合は1.0確率で計算する

log_mode

[in] 計算のフラグ。log_mode=trueの場合、計算はExp(probability)に対して行われる

error_code

[out] エラーコードを取得するための変数

result[]

[out] 分位数の値を持つ配列

© 2000–2025, MetaQuotes Ltd.

MathRandomNoncentralBeta

パラメータa、b、lambdaを使って、非心ベータ分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、NaNを返します。

```
double  MathRandomNoncentralBeta(
   const double  a,              // ベータ分布の1番目のパラメータ(形式1)
   const double  b,              // ベータ分布の2番目のパラメータ(形式2)
   const double  lambda,         // 非心パラメータ
   int&          error_code      // エラーコードを格納する変数
   );
```

パラメータa、b、lambdaを使って、非心ベータ分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、falseを返します。Rのrbeta()の類似体です。

```
bool  MathRandomNoncentralBeta(
   const double  a,               // ベータ分布の1番目のパラメータ(形式1)
   const double  b,               // ベータ分布の2番目のパラメータ(形式2)
   const double  lambda,          // 非心パラメータ
   const int     data_count,      // 必要なデータ量
   double&       result[]         // 擬似乱数変数を取得するための配列
   );
```

パラメータ

a

[in] ベータ分布の1番目のパラメータ(形式1)

b

[in] ベータ分布の2番目のパラメータ(形式2)

lambda

[in] 非心パラメータ

error_code

[out] エラーコードを格納する変数

data_count

[out] 必要なデータ量

result[]

[out] 擬似乱数の値を取得するための配列

© 2000-2025, MetaQuotes Ltd.

MathMomentsNoncentralBeta

パラメータa、bを使用して、非心ベータ分布の最初の4次のモーメントの理論的数値を計算します。

```
double  MathMomentsNoncentralBeta(
   const double  a,              // ベータ分布の1番目のパラメータ(形式1)
   const double  b,              // ベータ分布の2番目のパラメータ(形式2)
   const double  lambda,         // 非心パラメータ
   double&       mean,           // 平均を表す変数
   double&       variance,       // 分散を表す変数
   double&       skewness,       // 歪みを表す変数
   double&       kurtosis,       // 尖度を表す変数
   int&          error_code      // エラーコードを表す変数
   );
```

パラメータ

a

[in] ベータ分布の1番目のパラメータ(形式1)

b

[in] ベータ分布の2番目のパラメータ(形式2)

lambda

[in] 非心パラメータ

mean

[out] 平均値を取得するための変数

variance

[out] 分散を取得するための変数

skewness

[out] 歪みを取得するための変数

kurtosis

[out] 尖度を取得するための変数

error_code

[out] エラーコードを取得するための変数

戻り値

モーメントの計算が成功した場合はtrue、そうでない場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

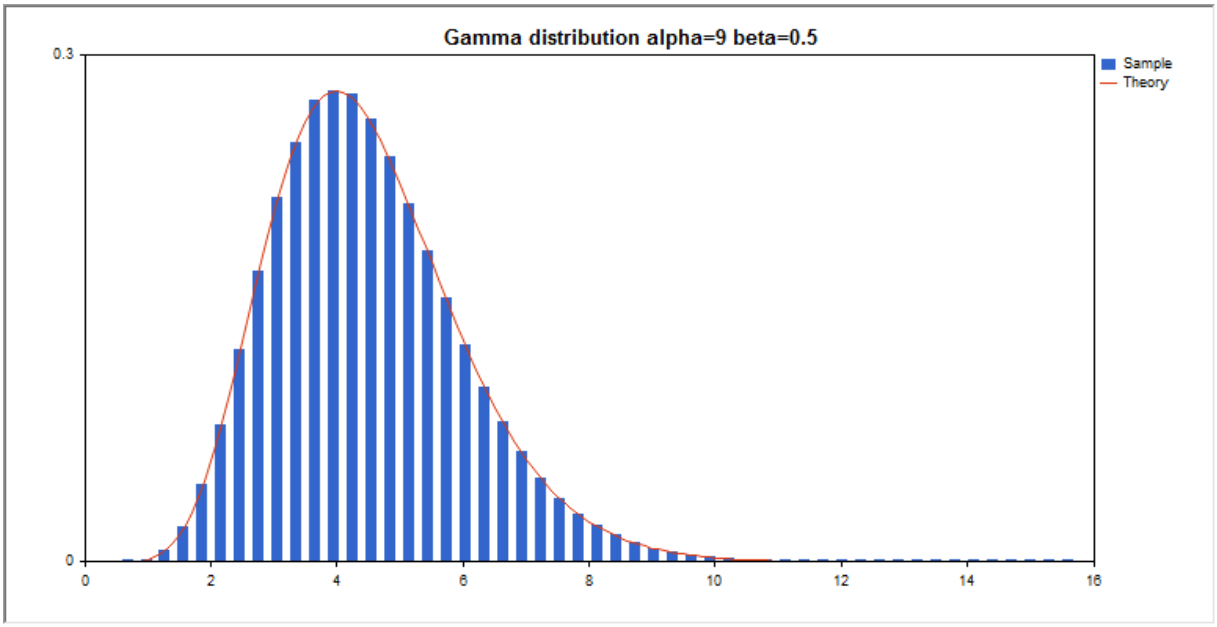
ガンマ分布

このセクションには、ガンマ分布を使用するための関数が含まれています。密度、確率、分位数を計算し、対応する法則に従って分布した疑似乱数を生成することができます。ガンマ分布は下記の式で定義されています。

$$f_{Gamma}(x|a,b)=\frac{1}{b^a\Gamma(a)}x^{a-1}e^{-\frac{x}{b}}$$

ここで

- x — 確率変数の値
- a — 分布の1番目のパラメータ
- b — 分布の2番目のパラメータ



ライブラリは、個々の確率変数の計算に加えて、確率変数の配列を扱う能力も実装しています。

関数	説明
MathProbabilityDensityGamma	ガンマ分布の確率密度関数を計算します
MathCumulativeDistributionGamma	ガンマ確率分布関数の値を計算します
MathQuantileGamma	指定された確率に対するガンマ逆分布関数の値を計算します
MathRandomGamma	ガンマ分布に従って分布した擬似乱数変数/擬似乱数変数の配列を生成します
MathMomentsGamma	ガンマ分布の最初の4次のモーメントの理論的数値を計算します

例：

© 2000–2025, MetaQuotes Ltd.

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Gamma.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 入力パラメータ
input double alpha=9;   // 分布の1番目のパラメータ(形状)
input double beta=0.5;  // 分布の2番目のパラメータ(スケール)
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 価格チャートを非表示にする
   ChartSetInteger(0,CHART_SHOW,false);
//--- 擬似乱数生成器を初期化する
   MathSrand(GetTickCount());
//--- 確率変数のサンプルを生成する
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // サンプルの値の数
   int ncells=51;          // ヒストグラムの間隔の数
   double x[];             // ヒストグラム間隔の中心
   double y[];             // 間隔内のサンプルからの値の数
   double data[];          // 確率変数値のサンプル
   double max,min;         // サンプルの最大値と最小値
//--- ガンマ分布からサンプルを取得する
   MathRandomGamma(alpha,beta,n,data);
//--- ヒストグラムをプロットするためのデータを計算する
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 理論曲線をプロットするシーケンス境界とステップを取得する
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- [min、max]の間隔で理論的に計算されたデータを取得する
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityGamma(x2,alpha,beta,false,y2);
//--- 規模を設定する
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- チャートを出力する
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
```

© 2000-2025, MetaQuotes Ltd.

```
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Gamma distribution alpha=%G beta=%
G",alpha,beta));
   graphic.BackgroundMainSize(16);
//--- Y軸の自動スケーリングを無効にする
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Max(NormalizeDouble(theor_max,1));
   graphic.YAxis().Min(0);
//--- カーブをプロットする
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 今度は分布密度の理論曲線をプロットする
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- カーブをプロットする
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|   データセットの頻度を計算する                                   |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double
&frequency[],
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 間隔の中心を定義する
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 間隔内に入る頻度を記入する
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|  シーケンス生成の値を計算する                                                |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- 正規化の精度を得るためにシーケンスの絶対範囲を計算する
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- 最大値と最小値を指定された精度に正規化する
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 指定された精度に基づいてシーケンス生成ステップも設定される
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityGamma

確率変数xに対して、パラメータa、bを用いてガンマ分布の確率密度関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityGamma(
   const double  x,              // 確率変数の値
   const double  a,              // 分布の1番目のパラメータ(形状)
   const double  b,              // 分布の2番目のパラメータ(スケール)
   const bool    log_mode,       // 値の対数を計算する。log_mode=trueの場合は、確率密度の自然対数が計算される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータa、bを用いてガンマ分布の確率密度関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityGamma(
   const double  x,              // 確率変数の値
   const double  a,              // 分布の1番目のパラメータ(形状)
   const double  b,              // 分布の2番目のパラメータ(スケール)
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対して、パラメータa、bを用いてガンマ分布の確率密度関数の値を計算します。エラーの場合は、falseを返します。Rのdgamma()の類似体です。

```
bool  MathProbabilityDensityGamma(
   const double& x[],            // 確率変数の値を持つ配列
   const double  a,              // 分布の1番目のパラメータ(形状)
   const double  b,              // 分布の2番目のパラメータ(スケール)
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率密度の自然対数が計算される
   double&       result[]        // 確率密度関数の値の配列
   );
```

確率変数の配列x[]に対して、パラメータa、bを用いてガンマ分布の確率密度関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathProbabilityDensityGamma(
   const double& x[],            // 確率変数の値を持つ配列
   const double  a,              // 分布の1番目のパラメータ(形状)
   const double  b,              // 分布の2番目のパラメータ(スケール)
   double&       result[]        // 確率密度関数の値の配列
   );
```

パラメータ

x

[in] 確率変数の値

x[]

© 2000-2025, MetaQuotes Ltd.

[in] 確率変数の値を持つ配列

a

[in] 分布の1番目のパラメータ(形状1)

b

[in] 分布の2番目のパラメータ(スケール)

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率密度の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率密度関数の値の配列

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionGamma

確率変数xに対して、パラメータa、bを用いてガンマ分布の確率分布関数を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionGamma(
   const double  x,              // 確率変数の値
   const double  a,              // 分布の1番目のパラメータ(形状)
   const double  b,              // 分布の2番目のパラメータ(スケール)
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算する。log_mode=trueの場合は、確率の自然対数が返される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータa、bを用いてガンマ分布の確率分布関数を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionGamma(
   const double  x,              // 確率変数の値
   const double  a,              // 分布の1番目のパラメータ(形状)
   const double  b,              // 分布の2番目のパラメータ(スケール)
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対してパラメータa、bを用いてガンマ分布の確率分布関数を計算します。エラーの場合は、falseを返します。Rのpgamma()の類似体です。

```
bool  MathCumulativeDistributionGamma(
   const double& x[],            // 確率変数の値を持つ配列
   const double  a,              // 分布の1番目のパラメータ(形状)
   const double  b,              // 分布の2番目のパラメータ(スケール)
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される
   double&       result[]        // 確率関数の値の配列
   );
```

確率変数の配列x[]に対してパラメータa、bを用いてガンマ分布の確率分布関数を計算します。エラーの場合は、falseを返します。

```
bool  MathCumulativeDistributionGamma(
   const double& x[],            // 確率変数の値を持つ配列
   const double  a,              // 分布の1番目のパラメータ(形状)
   const double  b,              // 分布の2番目のパラメータ(スケール)
   double&       result[]        // 確率関数の値の配列
   );
```

パラメータ

© 2000-2025, MetaQuotes Ltd.

x

[in] 確率変数の値

x[]

[in] 確率変数の値を持つ配列

a

[in] 分布の1番目のパラメータ（形状1）

b

[in] 分布の2番目のパラメータ（形状2）

tail

[in] 計算のフラグ。true の場合、確率変数がx 以下である確率を計算する。

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率関数の値の配列

© 2000–2025, MetaQuotes Ltd.

MathQuantileGamma

この関数は、指定されたprobabilityに対して、パラメータa、bを用いてガンマ逆分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileGamma(
   const double  probability,     // 確率変数の発生確率値
   const double  a,               // 分布の1番目のパラメータ(形状)
   const double  b,               // 分布の2番目のパラメータ(スケール)
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定されたprobabilityに対して、パラメータa、bを用いてガンマ逆分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileGamma(
   const double  probability,     // 確率変数の発生確率値
   const double  a,               // 分布の1番目のパラメータ(形状)
   const double  b,               // 分布の2番目のパラメータ(スケール)
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータa、bを用いてガンマ逆分布関数の値を計算します。エラーの場合は、falseを返します。Rのqgamma()の類似体です。

```
double  MathQuantileGamma(
   const double& probability[],   // 確率変数の確率値を持つ配列
   const double  a,               // 分布の1番目のパラメータ(形状)
   const double  b,               // 分布の2番目のパラメータ(スケール)
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   double&       result[]         // 分位数の値を持つ配列
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータa、bを用いてガンマ逆分布関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathQuantileGamma(
   const double& probability[],   // 確率変数の確率値を持つ配列
   const double  a,               // 分布の1番目のパラメータ(形状)
   const double  b,               // 分布の2番目のパラメータ(スケール)
   double&       result[]         // 分位数の値を持つ配列
   );
```

パラメータ

probability

© 2000-2025, MetaQuotes Ltd.

[in] 確率変数の確率値

probability[]

[in] 確率変数の確率値を持つ配列

a

[in] 分布の1番目のパラメータ(形状1)

b

[in] 分布の2番目のパラメータ(スケール)

tail

[in] 計算のフラグ。falseの場合は1.0確率で計算する

log_mode

[in] 計算のフラグ。log_mode=trueの場合、計算はExp(probability)に対して行われる

error_code

[out] エラーコードを取得するための変数

result[]

[out] 分位数の値を持つ配列

© 2000-2025, MetaQuotes Ltd.

MathRandomGamma

パラメータa、bを使って、ガンマ分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、NaNを返します。

```
double  MathRandomGamma(
   const double  a,              // 分布の1番目のパラメータ(形状)
   const double  b,              // 分布の2番目のパラメータ(スケール)
   int&          error_code      // エラーコードを格納する変数
   );
```

パラメータa、bを使って、ガンマ分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、falseを返します。Rのrgamma()の類似体です。

```
bool  MathRandomGamma(
   const double  a,               // 分布の1番目のパラメータ(形状)
   const double  b,               // 分布の2番目のパラメータ(スケール)
   const int     data_count,      // 必要なデータ量
   double&       result[]         // 擬似乱数変数の値の配列
   );
```

パラメータ

a

[in] 分布の1番目のパラメータ(形状1)

b

[in] 分布の2番目のパラメータ(スケール)

error_code

[out] エラーコードを格納する変数

data_count

[out] 必要なデータ量

result[]

[out] 擬似乱数の値を取得するための配列

© 2000-2025, MetaQuotes Ltd.

MathMomentsGamma

パラメータa、bを使用して、ガンマ分布の最初の4次のモーメントの理論的数値を計算します。

```
double  MathMomentsGamma(
   const double  a,              // 分布の1番目のパラメータ(形状)
   const double  b,              // 分布の2番目のパラメータ(スケール)
   double&       mean,           // 平均を表す変数
   double&       variance,       // 分散を表す変数
   double&       skewness,       // 歪みを表す変数
   double&       kurtosis,       // 尖度を表す変数
   int&          error_code      // エラーコードを表す変数
   );
```

パラメータ

a

[in] 分布の1番目のパラメータ(形状)

b

[in] 分布の2番目のパラメータ(スケール)

mean

[out] 平均値を取得するための変数

variance

[out] 分散を取得するための変数

skewness

[out] 歪みを取得するための変数

kurtosis

[out] 尖度を取得するための変数

error_code

[out] エラーコードを取得するための変数

戻り値

モーメントの計算が成功した場合はtrue、そうでない場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

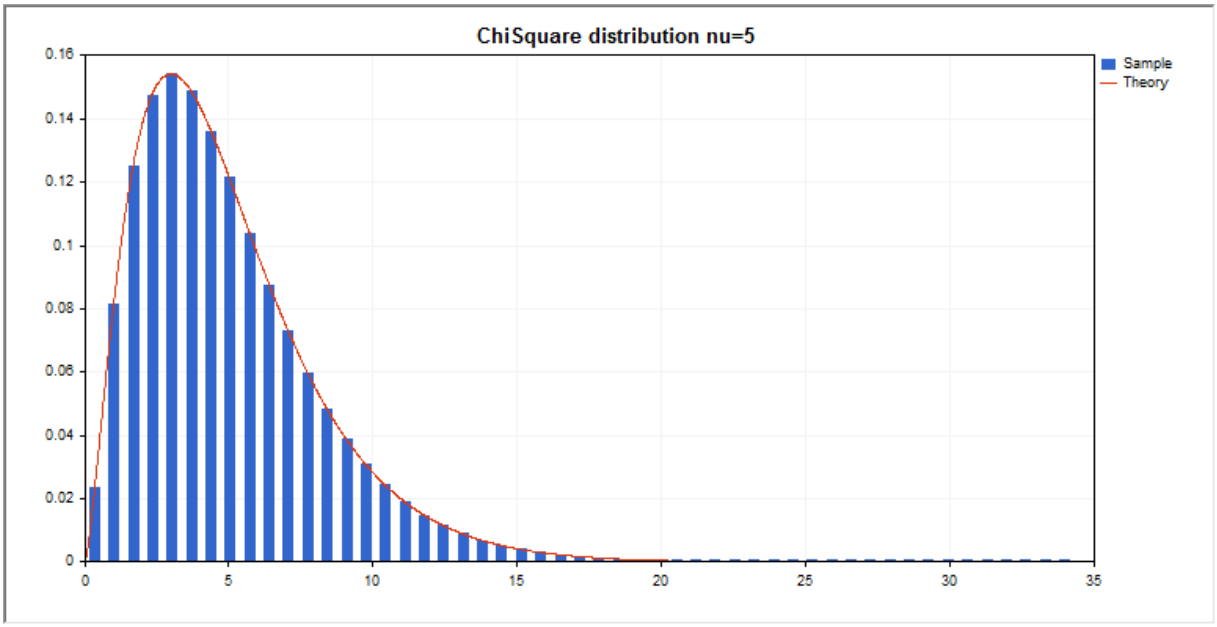
カイ2乗分布

このセクションには、カイ2乗分布を使用するための関数が含まれています。密度、確率、分位数を計算し、対応する法則に従って分布した疑似乱数を生成することができます。カイ2乗分布は下記の式で定義されています。

$$f_{Chi-Square}(x|\nu)=\frac{x^{\frac{(\nu-2)}{2}}e^{-\frac{x}{2}}}{2^{\frac{\nu}{2}}\Gamma\left(\frac{\nu}{2}\right)}$$

ここで

- x ― 確率変数の値
- ν ― 自由度の数



ライブラリは、個々の確率変数の計算に加えて、確率変数の配列を扱う能力も実装しています。

関数	説明
MathProbabilityDensityChiSquare	カイ2乗分布の確率密度関数を計算します
MathCumulativeDistributionChiSquare	カイ2乗確率分布関数の値を計算します
MathQuantileChiSquare	指定された確率に対するカイ2乗逆分布関数の値を計算します
MathRandomChiSquare	カイ2乗分布に従って分布した擬似乱数変数/擬似乱数変数の配列を生成します
MathMomentsChiSquare	カイ2乗分布の最初の4次のモーメントの理論的数値を計算します

© 2000–2025, MetaQuotes Ltd.

例：

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\ChiSquare.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 入力パラメータ
input double nu_par=5;     // 自由度の数
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 価格チャートを非表示にする
   ChartSetInteger(0,CHART_SHOW,false);
//--- 擬似乱数生成器を初期化する
   MathSrand(GetTickCount());
//--- 確率変数のサンプルを生成する
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // サンプルの値の数
   int ncells=51;          // ヒストグラムの間隔の数
   double x[];             // ヒストグラム間隔の中心
   double y[];             // 間隔内のサンプルからの値の数
   double data[];          // 確率変数値のサンプル
   double max,min;         // サンプルの最大値と最小値
//--- カイ2乗分布からサンプルを取得する
   MathRandomChiSquare(nu_par,n,data);
//--- ヒストグラムをプロットするためのデータを計算する
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 理論曲線をプロットするシーケンス境界とステップを取得する
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- [min、max]の間隔で理論的に計算されたデータを取得する
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityChiSquare(x2,nu_par,false,y2);
//--- 規模を設定する
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- チャートを出力する
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
```

© 2000-2025, MetaQuotes Ltd.

```
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("ChiSquare distribution nu=%G ",nu_par));
   graphic.BackgroundMainSize(16);
//--- カーブをプロットする
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 今度は分布密度の理論曲線をプロットする
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- カーブをプロットする
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  データセットの頻度を計算する                                    |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double
&frequency[],
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 間隔の中心を定義する
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 間隔内に入る頻度を記入する
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  シーケンス生成の値を計算する                                    |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 正規化の精度を得るためにシーケンスの絶対範囲を計算する
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- 最大値と最小値を指定された精度に正規化する
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 指定された精度に基づいてシーケンス生成ステップも設定される
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityChiSquare

確率変数xに対して、パラメータnuを用いてカイ2乗分布の確率密度関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityChiSquare(
   const double  x,              // 確率変数の値
   const double  nu,             // 分布のパラメータ(自由度の数)
   const bool    log_mode,       // 値の対数を計算する。log_mode=trueの場合は、確率密度の自然対数が計算される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータnuを用いてカイ2乗分布の確率密度関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityChiSquare(
   const double  x,              // 確率変数の値
   const double  nu,             // 分布のパラメータ(自由度の数)
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対して、パラメータnuを用いてカイ2乗分布の確率密度関数を計算します。エラーの場合は、falseを返します。Rのdchisq()の類似体です。

```
bool  MathProbabilityDensityChiSquare(
   const double& x[],            // 確率変数の値を持つ配列
   const double  nu,             // 分布のパラメータ(自由度の数)
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率密度の自然対数が計算される
   double&       result[]        // 確率密度関数の値の配列
   );
```

確率変数の配列x[]に対して、パラメータnuを用いてカイ2乗分布の確率密度関数を計算します。エラーの場合は、falseを返します。

```
bool  MathProbabilityDensityChiSquare(
   const double& x[],            // 確率変数の値を持つ配列
   const double  nu,             // 分布のパラメータ(自由度の数)
   double&       result[]        // 確率密度関数の値の配列
   );
```

パラメータ

x

[in] 確率変数の値

x[]

[in] 確率変数の値を持つ配列

nu

[in] 分布のパラメータ(自由度の数)

© 2000-2025, MetaQuotes Ltd.

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率密度の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率密度関数の値の配列

© 2000–2025, MetaQuotes Ltd.

MathCumulativeDistributionChiSquare

確率変数xに対して、パラメータnuを用いてカイ2乗分布の確率分布関数を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionChiSquare(
   const double  x,              // 確率変数の値
   const double  nu,             // 分布のパラメータ(自由度の数)
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算する。log_mode=trueの場合は、確率の自然対数が返される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータnuを用いてカイ2乗分布の確率分布関数を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionChiSquare(
   const double  x,              // 確率変数の値
   const double  nu,             // 分布のパラメータ(自由度の数)
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対してパラメータnuを用いてカイ2乗分布の確率分布関数を計算します。エラーの場合は、falseを返します。Rのpchisq()の類似体です。

```
bool  MathCumulativeDistributionChiSquare(
   const double& x[],            // 確率変数の値を持つ配列
   const double  nu,             // 分布のパラメータ(自由度の数)
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される
   double&       result[]        // 確率関数の値の配列
   );
```

確率変数の配列x[]に対してパラメータnuを用いてカイ2乗分布の確率分布関数を計算します。エラーの場合は、falseを返します。

```
bool  MathCumulativeDistributionChiSquare(
   const double& x[],            // 確率変数の値を持つ配列
   const double  nu,             // 分布のパラメータ(自由度の数)
   double&       result[]        // 確率関数の値の配列
   );
```

パラメータ

x

[in] 確率変数の値

x[]

© 2000-2025, MetaQuotes Ltd.

[in] 確率変数の値を持つ配列

nu

[in] 分布のパラメータ(自由度の数)

tail

[in] 計算のフラグ。true の場合、確率変数がx 以下である確率を計算する。

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率関数の値の配列

© 2000-2025, MetaQuotes Ltd.

MathQuantileChiSquare

この関数は、指定されたprobabilityに対して、カイ二乗逆分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileChiSquare(
   const double  probability,     // 確率変数の発生確率値
   const double  nu,              // 分布のパラメータ(自由度の数)
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定されたprobabilityに対して、カイ二乗逆分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileChiSquare(
   const double  probability,     // 確率変数の発生確率値
   const double  nu,              // 分布のパラメータ(自由度の数)
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定された確率値の配列probability[]に対して、カイ二乗逆分布関数の値を計算します。エラーの場合は、falseを返します。Rのqchisq()の類似体です。

```
double  MathQuantileChiSquare(
   const double& probability[],   // 確率変数の確率値を持つ配列
   const double  nu,              // 分布のパラメータ(自由度の数)
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   double&       result[]         // 分位数の値を持つ配列
   );
```

この関数は、指定された確率値の配列probability[]に対して、カイ二乗逆分布関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathQuantileChiSquare(
   const double& probability[],   // 確率変数の確率値を持つ配列
   const double  nu,              // 分布のパラメータ(自由度の数)
   double&       result[]         // 分位数の値を持つ配列
   );
```

パラメータ

probability

[in] 確率変数の確率値

probability[]

[in] 確率変数の確率値を持つ配列

nu

© 2000-2025, MetaQuotes Ltd.

[in] 分布のパラメータ（自由度の数）

tail

[in] 計算のフラグ。falseの場合は1.0確率で計算する

log_mode

[in] 計算のフラグ。log_mode=trueの場合、計算はExp(probability)に対して行われる

error_code

[out] エラーコードを取得するための変数

result[]

[out] 分位数の値を持つ配列

© 2000-2025, MetaQuotes Ltd.

MathRandomChiSquare

パラメータnuを使って、カイ二乗分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、NaNを返します。

```
double  MathRandomChiSquare(
   const double  nu,              // 分布のパラメータ(自由度の数)
   int&          error_code       // エラーコードを格納する変数
   );
```

パラメータnuを使って、カイ二乗分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、falseを返します。Rのrchisq()の類似体です。

```
bool  MathRandomChiSquare(
   const double  nu,               // 分布のパラメータ(自由度の数)
   const int     data_count,       // 必要なデータ量
   double&       result[]          // 擬似乱数変数の値の配列
   );
```

パラメータ

nu

[in] 分布のパラメータ(自由度の数)

error_code

[out] エラーコードを格納する変数

data_count

[out] 必要なデータ量

result[]

[out] 擬似乱数の値を取得するための配列

© 2000-2025, MetaQuotes Ltd.

MathMomentsChiSquare

パラメータnuを使用して、カイ2乗分布の最初の4次のモーメントの理論的数値を計算します。

```
double  MathMomentsChiSquare(
   const double  nu,              // 分布のパラメータ(自由度の数)
   double&       mean,            // 平均を表す変数
   double&       variance,        // 分散を表す変数
   double&       skewness,        // 歪みを表す変数
   double&       kurtosis,        // 尖度を表す変数
   int&          error_code       // エラーコードを表す変数
   );
```

パラメータ

nu

[in] 分布のパラメータ(自由度の数)

mean

[out] 平均値を取得するための変数

variance

[out] 分散を取得するための変数

skewness

[out] 歪みを取得するための変数

kurtosis

[out] 尖度を取得するための変数

error_code

[out] エラーコードを取得するための変数

戻り値

モーメントの計算が成功した場合はtrue、そうでない場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

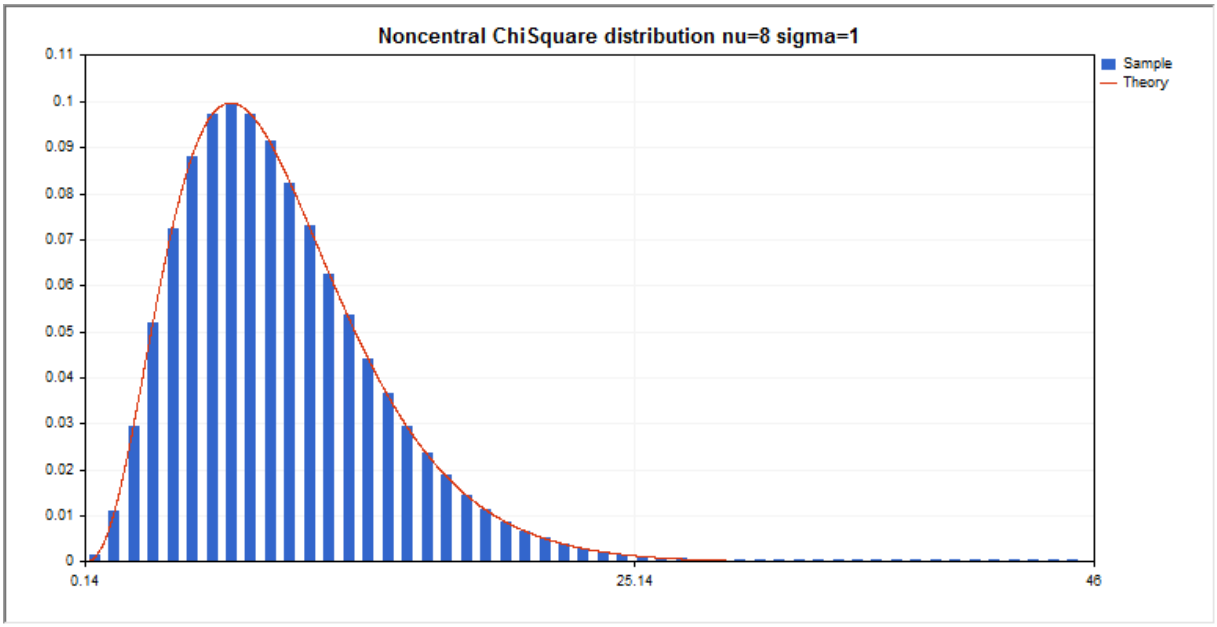
非心カイ2乗分布

このセクションには、非心カイ2乗分布を使用するための関数が含まれています。密度、確率、分位数を計算し、対応する法則に従って分布した疑似乱数を生成することができます。非心カイ2乗分布は下記の式で定義されています。

$$f_{NoncentralChiSquare}(x|\nu,\sigma)=\frac{1}{2^{\frac{\nu}{2}}\Gamma\left(\frac{1}{2}\right)}x^{\frac{\nu}{2}-1}e^{-\frac{(x+\sigma)}{2}}\sum_{r=0}^{\infty}\frac{(\lambda x)^r}{(2r)!}\frac{\Gamma\left(\frac{1}{2}+r\right)}{\Gamma\left(\frac{\nu}{2}+r\right)}$$

ここで

- x — 確率変数の値
- ν — 自由度の数
- σ — 非心パラメータ



ライブラリは、個々の確率変数の計算に加えて、確率変数の配列を扱う能力も実装しています。

関数	説明
MathProbabilityDensityNoncentralChiSquare	非心カイ2乗分布の確率密度関数を計算します
MathCumulativeDistributionNoncentralChiSquare	非心カイ2乗確率分布関数の値を計算します
MathQuantileNoncentralChiSquare	指定された確率に対する非心カイ2乗逆分布関数の値を計算します
MathRandomNoncentralChiSquare	非心カイ2乗分布に従って分布した擬似乱数変数/擬似乱数変数の配列を生成します

© 2000–2025, MetaQuotes Ltd.

関数	説明
MathMomentsNoncentralChiSquare	非心カイ2乗分布の最初の4次のモーメントの理論的数値を計算します

例：

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\NoncentralChiSquare.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 入力パラメータ
input double nu_par=8;     // 自由度の数
input double si_par=1;     // 非心パラメータ
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 価格チャートを非表示にする
   ChartSetInteger(0,CHART_SHOW,false);
//--- 擬似乱数生成器を初期化する
   MathSrand(GetTickCount());
//--- 確率変数のサンプルを生成する
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // サンプルの値の数
   int ncells=51;          // ヒストグラムの間隔の数
   double x[];             // ヒストグラム間隔の中心
   double y[];             // 間隔内のサンプルからの値の数
   double data[];          // 確率変数値のサンプル
   double max,min;         // サンプルの最大値と最小値
//--- 非新カイ2乗分布からサンプルを取得する
   MathRandomNoncentralChiSquare(nu_par,si_par,n,data);
//--- ヒストグラムをプロットするためのデータを計算する
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 理論曲線をプロットするシーケンス境界とステップを取得する
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- [min、max]の間隔で理論的に計算されたデータを取得する
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNoncentralChiSquare(x2,nu_par,si_par,false,y2);
//--- 規模を設定する
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- チャートを出力する
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Noncentral ChiSquare distribution nu=%G
sigma=%G",nu_par,si_par));
   graphic.BackgroundMainSize(16);
//--- Y軸の自動スケーリングを無効にする
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Max(NormalizeDouble(max,0));
   graphic.XAxis().Min(min);
//--- カーブをプロットする
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 今度は分布密度の理論曲線をプロットする
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- カーブをプロットする
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|   データセットの頻度を計算する                                   |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double
&frequency[],
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 間隔の中心を定義する
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 間隔内に入る頻度を記入する
   for(int i=0; i<size; i++)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  シーケンス生成の値を計算する                                    |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- 正規化の精度を得るためにシーケンスの絶対範囲を計算する
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- 最大値と最小値を指定された精度に正規化する
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 指定された精度に基づいてシーケンス生成ステップも設定される
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityNoncentralChiSquare

確率変数xに対して、パラメータnu、sigmaを用いて非心カイ2乗分布の確率密度関数を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityNoncentralChiSquare(
   const double  x,              // 確率変数の値
   const double  nu,             // 分布のパラメータ(自由度の数)
   const double  sigma,          // 非心パラメータ
   const bool    log_mode,       // 値の対数を計算する。log_mode=trueの場合は、確率密度の自然対数が計算される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータnu、sigmaを用いて非心カイ2乗分布の確率密度関数を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityNoncentralChiSquare(
   const double  x,              // 確率変数の値
   const double  nu,             // 分布のパラメータ(自由度の数)
   const double  sigma,          // 非心パラメータ
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対して、パラメータnu、sigmaを用いて非心カイ2乗分布の確率密度関数を計算します。エラーの場合は、falseを返します。Rのdchisq()の類似体です。

```
bool  MathProbabilityDensityNoncentralChiSquare(
   const double& x[],            // 確率変数の値を持つ配列
   const double  nu,             // 分布のパラメータ(自由度の数)
   const double  sigma,          // 非心パラメータ
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率密度の自然対数が計算される
   double&       result[]        // 確率密度関数の値の配列
   );
```

確率変数の配列x[]に対して、パラメータnu、sigmaを用いてカイ2乗分布の確率密度関数を計算します。エラーの場合は、falseを返します。

```
bool  MathProbabilityDensityNoncentralChiSquare(
   const double& x[],            // 確率変数の値を持つ配列
   const double  nu,             // 分布のパラメータ(自由度の数)
   const double  sigma,          // 非心パラメータ
   double&       result[]        // 確率密度関数の値の配列
   );
```

パラメータ

x

[in] 確率変数の値

x[]

© 2000-2025, MetaQuotes Ltd.

[in] 確率変数の値を持つ配列

nu

[in] 分布のパラメータ(自由度の数)

sigma

[in] 非心パラメータ.

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率密度の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率密度関数の値の配列

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionNoncentralChiSquare

確率変数xに対して、パラメータnu、sigmaを用いて非心カイ2乗分布の確率分布関数を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionNoncentralChiSquare(
   const double  x,              // 確率変数の値
   const double  nu,             // 分布のパラメータ(自由度の数)
   const double  sigma,          // 非心パラメータ
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算する。log_mode=trueの場合は、確率の自然対数が返される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータnu、sigmaを用いて非心カイ2乗分布の確率分布関数を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionNoncentralChiSquare(
   const double  x,              // 確率変数の値
   const double  nu,             // 分布のパラメータ(自由度の数)
   const double  sigma,          // 非心パラメータ
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータnu、sigmaを用いて非心カイ2乗分布の確率分布関数を計算します。エラーの場合は、falseを返します。Rのpchisq()の類似体です。

```
bool  MathCumulativeDistributionNoncentralChiSquare(
   const double& x[],            // 確率変数の値を持つ配列
   const double  nu,             // 分布のパラメータ(自由度の数)
   const double  sigma,          // 非心パラメータ
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される
   double&       result[]        // 確率関数の値の配列
   );
```

<t0>確率変数の配列x[]に対して、パラメータnu、sigmaを用いて非心カイ2乗分布の確率分布関数を計算します。エラーの場合は、falseを返します。

```
bool  MathCumulativeDistributionNoncentralChiSquare(
   const double& x[],            // 確率変数の値を持つ配列
   const double  nu,             // 分布のパラメータ(自由度の数)
   const double  sigma,          // 非心パラメータ
   double&       result[]        // 確率関数の値の配列
   );
```

パラメータ

© 2000–2025, MetaQuotes Ltd.

x

[in] 確率変数の値

x[]

[in] 確率変数の値を持つ配列

nu

[in] 分布のパラメータ（自由度の数）

sigma

[in] 非心パラメータ.

tail

[in] 計算のフラグ。true の場合、確率変数がx 以下である確率を計算する。

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率関数の値の配列

© 2000–2025, MetaQuotes Ltd.

MathQuantileNoncentralChiSquare

この関数は、指定されたprobabilityに対して、パラメータnu、sigmaを用いて非心カイ二乗逆分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileNoncentralChiSquare(
   const double  probability,     // 確率変数の発生確率値
   const double  nu,              // 分布のパラメータ(自由度の数)
   const double  sigma,           // 非心パラメータ
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定されたprobabilityに対して、パラメータnu、sigmaを用いて非心カイ二乗逆分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileNoncentralChiSquare(
   const double  probability,     // 確率変数の発生確率値
   const double  nu,              // 分布のパラメータ(自由度の数)
   const double  sigma,           // 非心パラメータ
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータnu、sigmaを用いて非心カイ二乗逆分布関数の値を計算します。エラーの場合は、falseを返します。Rのqchisq()の類似体です。

```
double  MathQuantileNoncentralChiSquare(
   const double&  probability[],  // 確率変数の確率値を持つ配列
   const double  nu,              // 分布のパラメータ(自由度の数)
   const double  sigma,           // 非心パラメータ
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   double&       result[]         // 分位数の値を持つ配列
   );
```

この関数は、指定された確率値の配列probability[]に対して、非心カイ二乗逆分布関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathQuantileNoncentralChiSquare(
   const double& probability[],  // 確率変数の確率値を持つ配列
   const double  nu,              // 分布のパラメータ(自由度の数)
   const double  sigma,           // 非心パラメータ
   double&       result[]         // 分位数の値を持つ配列
   );
```

パラメータ

probability

© 2000-2025, MetaQuotes Ltd.

[in] 確率変数の確率値

probability[]

[in] 確率変数の確率値を持つ配列

nu

[in] 分布のパラメータ(自由度の数)

sigma

[in] 非心パラメータ.

tail

[in] 計算のフラグ。falseの場合は1.0確率で計算する

log_mode

[in] 計算のフラグ。log_mode=trueの場合、計算はExp(probability)に対して行われる

error_code

[out] エラーコードを取得するための変数

result[]

[out] 分位数の値を持つ配列

© 2000-2025, MetaQuotes Ltd.

MathRandomNoncentralChiSquare

パラメータnu、sigmaを使って、非心カイ二乗分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、NaNを返します。

```
double  MathRandomNoncentralChiSquare(
   const double  nu,             // 分布のパラメータ(自由度の数)
   const double  sigma,          // 非心パラメータ
   int&          error_code      // エラーコードを格納する変数
   );
```

パラメータnu、sigmaを使って、非心カイ二乗分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、falseを返します。Rのrchisq()の類似体です。

```
bool  MathRandomNoncentralChiSquare(
   const double  nu,              // 分布のパラメータ(自由度の数)
   const double  sigma,           // 非心パラメータ
   const int     data_count,      // 必要なデータ量
   double&       result[]         // 擬似乱数変数の値の配列
   );
```

パラメータ

nu

[in] 分布のパラメータ(自由度の数)

sigma

[in] 非心パラメータ.

error_code

[out] エラーコードを格納する変数

data_count

[out] 必要なデータ量

result[]

[out] 擬似乱数の値を取得するための配列

© 2000-2025, MetaQuotes Ltd.

MathMomentsNoncentralChiSquare

パラメータnu、sigmaを使用して、非心カイ2乗分布の最初の4次のモーメントの理論的数値を計算します。

```
double  MathMomentsNoncentralChiSquare(
   const double  nu,              // 分布のパラメータ(自由度の数)
   const double  sigma,           // 非心パラメータ
   double&       mean,            // 平均を表す変数
   double&       variance,        // 分散を表す変数
   double&       skewness,        // 歪みを表す変数
   double&       kurtosis,        // 尖度を表す変数
   int&          error_code       // エラーコードを表す変数
   );
```

パラメータ

nu

[in] 分布のパラメータ(自由度の数)

sigma

[in] 非心パラメータ.

mean

[out] 平均値を取得するための変数

variance

[out] 分散を取得するための変数

skewness

[out] 歪みを取得するための変数

kurtosis

[out] 尖度を取得するための変数

error_code

[out] エラーコードを取得するための変数

戻り値

モーメントの計算が成功した場合はtrue、そうでない場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

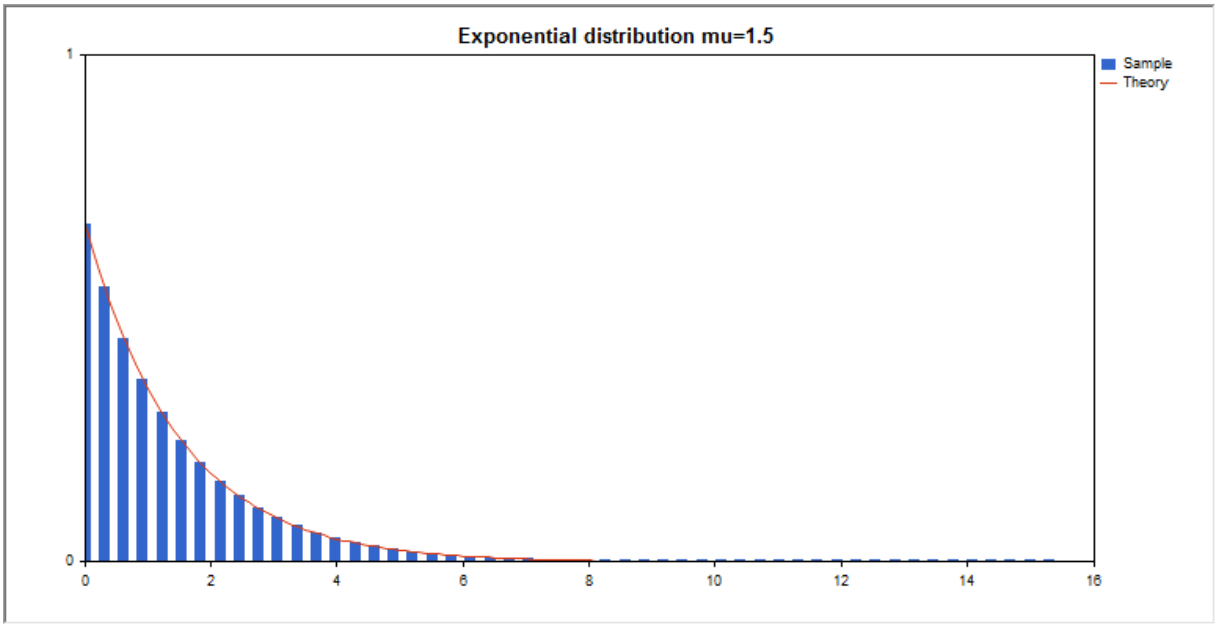
指数

このセクションには、指数分布を使用するための関数が含まれています。密度、確率、分位数を計算し、指数分布に従って分布した疑似乱数を生成することができます。指数分布は下記の式で定義されています。

$$f_{Exponential}(x|\mu)=\frac{1}{\mu}e^{-\frac{x}{\mu}}$$

ここで

- x ー 確率変数の値
- μ ー 期待値



ライブラリは、個々の確率変数の計算に加えて、確率変数の配列を扱う能力も実装しています。

関数	説明
MathProbabilityDensityExponential	指数分布の確率密度関数を計算します
MathCumulativeDistributionExponential	指数確率分布関数の値を計算します
MathQuantileExponential	指定された確率に対する指数逆分布関数の値を計算します
MathRandomExponential	指数分布に従って分布した擬似乱数変数/擬似乱数変数の配列を生成します
MathMomentsExponential	指数分布の最初の4次のモーメントの理論的数値を計算します

例：

```
#include <Graphics\Graphic.mqh>
```

© 2000-2025, MetaQuotes Ltd.

```
#include <Math\Stat\Exponential.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 入力パラメータ
input double mu_par=1.5;     // 自由度の数
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 価格チャートを非表示にする
   ChartSetInteger(0,CHART_SHOW,false);
//--- 擬似乱数生成器を初期化する
   MathSrand(GetTickCount());
//--- 確率変数のサンプルを生成する
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // サンプルの値の数
   int ncells=51;          // ヒストグラムの間隔の数
   double x[];             // ヒストグラム間隔の中心
   double y[];             // 間隔内のサンプルからの値の数
   double data[];          // 確率変数値のサンプル
   double max,min;         // サンプルの最大値と最小値
//--- 指数分布からサンプルを取得する
   MathRandomExponential(mu_par,n,data);
//--- ヒストグラムをプロットするためのデータを計算する
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 理論曲線をプロットするシーケンス境界とステップを取得する
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- [min、max]の間隔で理論的に計算されたデータを取得する
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityExponential(x2,mu_par,false,y2);
//--- 規模を設定する
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- チャートを出力する
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Exponential distribution mu=%G ",mu_par));
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.BackgroundMainSize(16);
//--- カーブをプロットする
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 今度は分布密度の理論曲線をプロットする
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- カーブをプロットする
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  データセットの頻度を計算する                                    |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double
&frequency[],
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 間隔の中心を定義する
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+i*width;
      frequency[i]=0;
     }
//--- 間隔内に入る頻度を記入する
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  シーケンス生成の値を計算する                                    |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- 正規化の精度を得るためにシーケンスの絶対範囲を計算する
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 最大値と最小値を指定された精度に正規化する
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 指定された精度に基づいてシーケンス生成ステップも設定される
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityExponential

確率変数xに対して、パラメータmuを用いて指数分布の確率密度関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityExponential(
   const double  x,              // 確率変数の値
   const double  mu,             // 分布のパラメータ(期待値)
   const bool    log_mode,       // 値の対数を計算する。log_mode=trueの場合は、確率密度の自然対数が計算される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータmuを用いて指数分布の確率密度関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityExponential(
   const double  x,              // 確率変数の値
   const double  mu,             // 分布のパラメータ(期待値)
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対して、パラメータmuを用いて確率分布の確率密度関数の値を計算します。エラーの場合は、falseを返します。Rのdexp()の類似体です。

```
bool  MathProbabilityDensityExponential(
   const double& x[],            // 確率変数の値を持つ配列
   const double  mu,             // 分布のパラメータ(期待値)
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率密度の自然対数が計算される
   double&       result[]        // 確率密度関数の値の配列
   );
```

確率変数の配列x[]に対して、パラメータmuを用いて確率分布の確率密度関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathProbabilityDensityExponential(
   const double& x[],            // 確率変数の値を持つ配列
   const double  mu,             // 分布のパラメータ(期待値)
   double&       result[]        // 確率密度関数の値の配列
   );
```

パラメータ

x

[in] 確率変数の値

x[]

[in] 確率変数の値を持つ配列

mu

[in] 分布のパラメータ(期待値)

© 2000-2025, MetaQuotes Ltd.

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率密度の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率密度関数の値の配列

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionExponential

確率変数xに対して、パラメータmuを用いて指数分布の確率分布関数を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionExponential(
   const double  x,              // 確率変数の値
   const double  mu,             // 分布のパラメータ(期待値)
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算する。log_mode=trueの場合は、確率の自然対数が返される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータmuを用いて指数分布の確率分布関数を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionExponential(
   const double  x,              // 確率変数の値
   const double  mu,             // 分布のパラメータ(期待値)
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対してパラメータmuを用いて指数分布の確率分布関数を計算します。エラーの場合は、falseを返します。Rのpexp()の類似体です。

```
bool  MathCumulativeDistributionExponential(
   const double& x[],            // 確率変数の値を持つ配列
   const double  mu,             // 分布のパラメータ(期待値)
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される
   double&       result[]        // 確率関数の値の配列
   );
```

確率変数の配列x[]に対してパラメータmuを用いて指数分布の確率分布関数を計算します。エラーの場合は、falseを返します。

```
bool  MathCumulativeDistributionExponential(
   const double& x[],            // 確率変数の値を持つ配列
   const double  mu,             // 分布のパラメータ(期待値)
   double&       result[]        // 確率関数の値の配列
   );
```

パラメータ

x

[in] 確率変数の値

x[]

© 2000-2025, MetaQuotes Ltd.

[in] 確率変数の値を持つ配列

mu

[in] 分布のパラメータ(期待値)

tail

[in] 計算のフラグ。true の場合、確率変数がx 以下である確率を計算する。

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率関数の値の配列

© 2000-2025, MetaQuotes Ltd.

MathQuantileExponential

この関数は、指定されたprobabilityに対して、パラメータmuを用いて指数逆分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileExponential(
   const double  probability,    // 確率変数の発生確率値
   const double  mu,             // 分布のパラメータ(期待値)
   const bool    tail,           // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,       // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   int&          error_code      // エラーコードを格納する変数
   );
```

この関数は、指定されたprobabilityに対して、パラメータmuを用いて指数逆分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileExponential(
   const double  probability,    // 確率変数の発生確率値
   const double  mu,             // 分布のパラメータ(期待値)
   int&          error_code      // エラーコードを格納する変数
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータmuを用いて指数逆分布関数の値を計算します。エラーの場合は、falseを返します。Rのqexp()の類似体です。

```
double  MathQuantileExponential(
   const double& probability[],  // 確率変数の確率値を持つ配列
   const double  mu,             // 分布のパラメータ(期待値)
   const bool    tail,           // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,       // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   double&       result[]        // 分位数の値を持つ配列
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータmuを用いて指数逆分布関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathQuantileExponential(
   const double& probability[],  // 確率変数の確率値を持つ配列
   const double  mu,             // 分布のパラメータ(期待値)
   double&       result[]        // 分位数の値を持つ配列
   );
```

パラメータ

probability

[in] 確率変数の確率値

probability[]

[in] 確率変数の確率値を持つ配列

mu

© 2000-2025, MetaQuotes Ltd.

[in] 分布のパラメータ（期待値）

tail

[in] 計算のフラグ。falseの場合は1.0確率で計算する

log_mode

[in] 計算のフラグ。log_mode=trueの場合、計算はExp(probability)に対して行われる

error_code

[out] エラーコードを取得するための変数

result[]

[out] 分位数の値を持つ配列

© 2000-2025, MetaQuotes Ltd.

MathRandomExponential

パラメータmuを使って、指数分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、NaNを返します。

```
double  MathRandomExponential(
   const double  mu,             // 分布のパラメータ(期待値)
   int&          error_code      // エラーコードを格納する変数
   );
```

パラメータmuを使って、指数分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、falseを返します。Rのrexp()の類似体です。

```
bool  MathRandomExponential(
   const double  mu,              // 分布のパラメータ(期待値)
   const int     data_count,      // 必要なデータ量
   double&       result[]         // 擬似乱数変数の値の配列
   );
```

パラメータ

mu

[in] 分布のパラメータ(期待値)

error_code

[out] エラーコードを格納する変数

data_count

[out] 必要なデータ量

result[]

[out] 擬似乱数の値を取得するための配列

© 2000-2025, MetaQuotes Ltd.

MathMomentsExponential

パラメータmuを使用して、指数分布の最初の4次のモーメントの理論的数値を計算します。

```
double  MathMomentsExponential(
   const double  mu,              // 分布のパラメータ(期待値)
   double&       mean,            // 平均を表す変数
   double&       variance,        // 分散を表す変数
   double&       skewness,        // 歪みを表す変数
   double&       kurtosis,        // 尖度を表す変数
   int&          error_code       // エラーコードを表す変数
   );
```

パラメータ

mu

[in] 分布のパラメータ(期待値)

mean

[out] 平均値を取得するための変数

variance

[out] 分散を取得するための変数

skewness

[out] 歪みを取得するための変数

kurtosis

[out] 尖度を取得するための変数

error_code

[out] エラーコードを取得するための変数

戻り値

モーメントの計算が成功した場合はtrue、そうでない場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

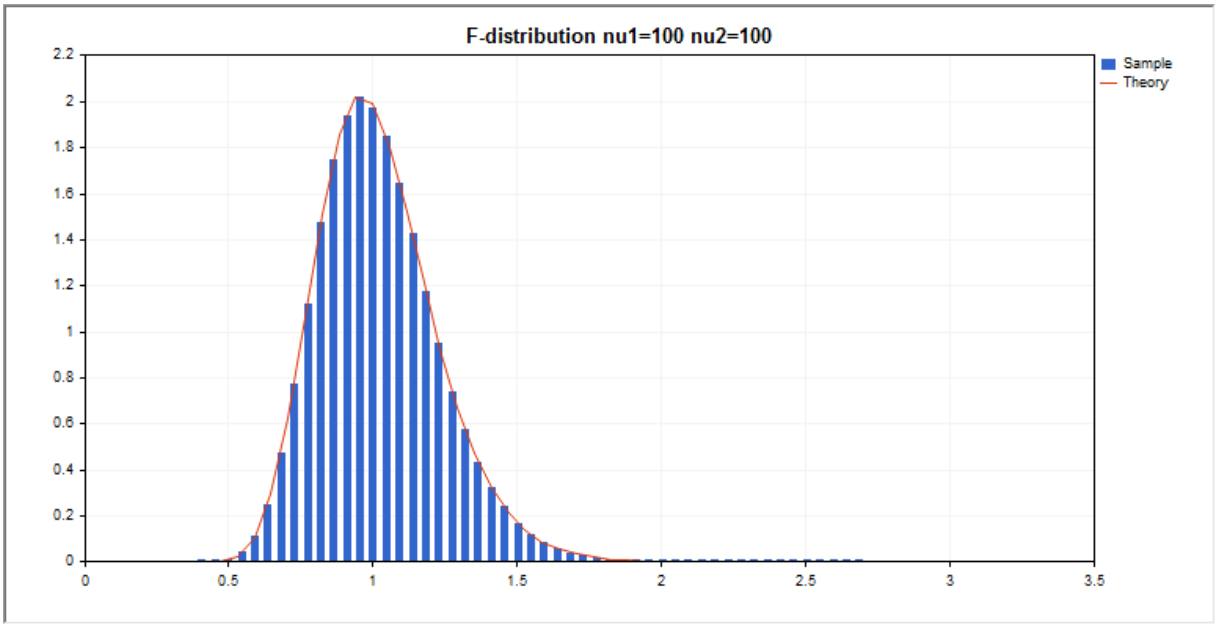
F分布

このセクションには、F分布を使用するための関数が含まれています。密度、確率、分位数を計算し、F分布に従って分布した疑似乱数を生成することができます。F分布は下記の式で定義されています。

$$f_F(x|\nu_1,\nu_2)=\frac{\Gamma\left(\frac{\nu_1+\nu_2}{2}\right)}{\Gamma\left(\frac{\nu_1}{2}\right)\Gamma\left(\frac{\nu_2}{2}\right)}\left(\frac{\nu_1}{\nu_2}\right)^{\frac{\nu_1}{2}}\frac{x^{\frac{\nu_1-2}{2}}}{\left(1+\left(\frac{\nu_1}{\nu_2}\right)x\right)^{\frac{\nu_1+\nu_2}{2}}}$$

ここで

- x — 確率変数の値
- ν1 — 分布の1番目のパラメータ(自由度の数)
- ν2 — 分布の2番目のパラメータ(自由度の数)</



ライブラリは、個々の確率変数の計算に加えて、確率変数の配列を扱う能力も実装しています。

関数	説明
MathProbabilityDensityF	F分布の確率密度関数を計算します
MathCumulativeDistributionF	F分布関数の値を計算します
MathQuantileF	指定された確率に対するF逆分布関数の値を計算します
MathRandomF	F分布に従って分布した擬似乱数変数/擬似乱数変数の配列を生成します
MathMomentsF	F分布の最初の4次のモーメントの理論的数値を計算します

例:

© 2000-2025, MetaQuotes Ltd.

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\F.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 入力パラメータ
input double nu_1=100;     // 1番目の自由度の数
input double nu_2=100;     // 2番目の自由度の数
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 価格チャートを非表示にする
   ChartSetInteger(0,CHART_SHOW,false);
//--- 擬似乱数生成器を初期化する
   MathSrand(GetTickCount());
//--- 確率変数のサンプルを生成する
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // サンプルの値の数
   int ncells=51;          // ヒストグラムの間隔の数
   double x[];             // ヒストグラム間隔の中心
   double y[];             // 間隔内のサンプルからの値の数
   double data[];          // 確率変数値のサンプル
   double max,min;         // サンプルの最大値と最小値
//--- フィッシャーのF分布からサンプルを取得する
   MathRandomF(nu_1,nu_2,n,data);
//--- ヒストグラムをプロットするためのデータを計算する
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 理論曲線をプロットするシーケンス境界とステップを取得する
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- [min、max]の間隔で理論的に計算されたデータを取得する
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityF(x2,nu_1,nu_2,false,y2);
//--- 規模を設定する
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- チャートを出力する
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
```

© 2000-2025, MetaQuotes Ltd.

```
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("F-distribution nu1=%G nu2=%G",nu_1,nu_2));
   graphic.BackgroundMainSize(16);
//--- カーブをプロットする
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(4);
//--- 今度は分布密度の理論曲線をプロットする
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- カーブをプロットする
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  データセットの頻度を計算する                                    |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double
&frequency[],
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 間隔の中心を定義する
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 間隔内に入る頻度を記入する
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  シーケンス生成の値を計算する                                    |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- 正規化の精度を得るためにシーケンスの絶対範囲を計算する
```

© 2000-2025, MetaQuotes Ltd.

```
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- 最大値と最小値を指定された精度に正規化する
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 指定された精度に基づいてシーケンス生成ステップも設定される
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityF

確率変数xに対して、パラメータnu1、nu2を用いてFisherのF分布の確率密度関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityF(
   const double  x,              // 確率変数の値
   const double  nu1,            // 分布の1番目のパラメータ(自由度の数)
   const double  nu2,            // 分布の2番目のパラメータ(自由度の数)
   const bool    log_mode,       // 値の対数を計算する。log_mode=trueの場合は、確率密度の自然対数が計算される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータnu1、nu2を用いてFisherのF分布の確率密度関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityF(
   const double  x,              // 確率変数の値
   const double  nu1,            // 分布の1番目のパラメータ(自由度の数)
   const double  nu2,            // 分布の2番目のパラメータ(自由度の数)
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対して、パラメータnu1、nu2を用いてFisherのF分布の確率密度関数の値を計算します。エラーの場合は、falseを返します。Rのdf()の類似体です。

```
bool  MathProbabilityDensityF(
   const double& x[],            // 確率変数の値を持つ配列
   const double  nu1,            // 分布の1番目のパラメータ(自由度の数)
   const double  nu2,            // 分布の2番目のパラメータ(自由度の数)
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率密度の自然対数が計算される
   double&       result[]        // 確率密度関数の値の配列
   );
```

確率変数の配列x[]に対して、パラメータnu1、nu2を用いてFisherのF分布の確率密度関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathProbabilityDensityF(
   const double& x[],            // 確率変数の値を持つ配列
   const double  nu1,            // 分布の1番目のパラメータ(自由度の数)
   const double  nu2,            // 分布の2番目のパラメータ(自由度の数)
   double&       result[]        // 確率密度関数の値の配列
   );
```

パラメータ

x

[in] 確率変数の値

x[]

© 2000-2025, MetaQuotes Ltd.

[in] 確率変数の値を持つ配列

nu1

[in] 分布の1番目のパラメータ(自由度の数)

nu2

[in] 分布の2番目のパラメータ(自由度の数)

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率密度の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率密度関数の値の配列

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionF

確率変数xに対して、パラメータnu1、nu2を用いてFisherのF分布の確率分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionF(
   const double  x,              // 確率変数の値
   const double  nu1,            // 分布の1番目のパラメータ(自由度の数)
   const double  nu2,            // 分布の2番目のパラメータ(自由度の数)
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算する。log_mode=trueの場合は、確率の自然対数が返される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータnu1、nu2を用いてFisherのF分布の確率分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionF(
   const double  x,              // 確率変数の値
   const double  nu1,            // 分布の1番目のパラメータ(自由度の数)
   const double  nu2,            // 分布の2番目のパラメータ(自由度の数)
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対して、パラメータnu1、nu2を用いてFisherのF分布の確率分布関数の値を計算します。エラーの場合は、falseを返します。Rのpf()の類似体です。

```
bool  MathCumulativeDistributionF(
   const double& x[],            // 確率変数の値を持つ配列
   const double  nu1,            // 分布の1番目のパラメータ(自由度の数)
   const double  nu2,            // 分布の2番目のパラメータ(自由度の数)
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算する。log_mode=trueの場合は、確率の自然対数が返される
   double&       result[]        // 確率関数の値の配列
   );
```

確率変数の配列x[]に対して、パラメータnu1、nu2を用いてFisherのF分布の確率分布関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathCumulativeDistributionF(
   const double& x[],            // 確率変数の値を持つ配列
   const double  nu1,            // 分布の1番目のパラメータ(自由度の数)
   const double  nu2,            // 分布の2番目のパラメータ(自由度の数)
   double&       result[]        // 確率関数の値の配列
   );
```

パラメータ

© 2000-2025, MetaQuotes Ltd.

x

[in] 確率変数の値

x[]

[in] 確率変数の値を持つ配列

nu1

[in] 分布の1番目のパラメータ（自由度の数）

nu2

[in] 分布の2番目のパラメータ（自由度の数）

tail

[in] 計算のフラグ。true の場合、確率変数がx 以下である確率を計算する。

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率関数の値の配列

© 2000–2025, MetaQuotes Ltd.

MathQuantileF

この関数は、指定されたprobabilityに対して、パラメータnu1、nu2を用いてFisherのF逆分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileF(
   const double  probability,     // 確率変数の発生確率値
   const double  nu1,             // 分布の1番目のパラメータ(自由度の数)
   const double  nu2,             // 分布の2番目のパラメータ(自由度の数)
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定されたprobabilityに対して、パラメータnu1、nu2を用いてFisherのF逆分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileF(
   const double  probability,     // 確率変数の発生確率値
   const double  nu1,             // 分布の1番目のパラメータ(自由度の数)
   const double  nu2,             // 分布の2番目のパラメータ(自由度の数)
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータnu1、nu2を用いてFisherのF逆分布関数の値を計算します。エラーの場合は、falseを返します。Rのqf()の類似体です。

```
double  MathQuantileF(
   const double& probability[],   // 確率変数の確率値を持つ配列
   const double  nu1,             // 分布の1番目のパラメータ(自由度の数)
   const double  nu2,             // 分布の2番目のパラメータ(自由度の数)
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   double&       result[]         // 分位数の値を持つ配列
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータnu1、nu2を用いてFisherのF逆分布関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathQuantileF(
   const double& probability[],   // 確率変数の確率値を持つ配列
   const double  nu1,             // 分布の1番目のパラメータ(自由度の数)
   const double  nu2,             // 分布の2番目のパラメータ(自由度の数)
   double&       result[]         // 分位数の値を持つ配列
   );
```

パラメータ

probability

© 2000-2025, MetaQuotes Ltd.

[in] 確率変数の確率値

probability[]

[in] 確率変数の確率値を持つ配列

nu1

[in] 分布の1番目のパラメータ(自由度の数)

nu2

[in] 分布の2番目のパラメータ(自由度の数)

tail

[in] 計算のフラグ。lower_tail=falseの場合は1.0確率で計算する

log_mode

[in] 計算のフラグ。log_mode=trueの場合、計算はExp(probability)に対して行われる

error_code

[out] エラーコードを取得するための変数

result[]

[out] 分位数の値を持つ配列

© 2000-2025, MetaQuotes Ltd.

MathRandomF

パラメータnu1、nu2を使って、FisherのF分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、NaNを返します。

```
double  MathRandomF(
   const double  nu1,              // 分布の1番目のパラメータ(自由度の数)
   const double  nu2,              // 分布の2番目のパラメータ(自由度の数)
   int&          error_code        // エラーコードを格納する変数
   );
```

パラメータnu1、nu2を使って、FisherのF分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、falseを返します。Rのrf()の類似体です。

```
bool  MathRandomF(
   const double  nu1,              // 分布の1番目のパラメータ(自由度の数)
   const double  nu2,              // 分布の2番目のパラメータ(自由度の数)
   const int     data_count,       // 必要なデータ量
   double&       result[]          // 擬似乱数変数の値の配列
   );
```

パラメータ

nu1

[in] 分布の1番目のパラメータ(自由度の数)

nu2

[in] 分布の2番目のパラメータ(自由度の数)

error_code

[out] エラーコードを格納する変数

data_count

[out] 必要なデータ量

result[]

[out] 擬似乱数の値を取得するための配列

© 2000-2025, MetaQuotes Ltd.

MathMomentsF

FisherのF分布の最初の4次のモーメントの理論的数値を計算します。

```
double  MathMomentsF(
   const double  nu1,            // 分布の1番目のパラメータ(自由度の数)
   const double  nu2,            // 分布の2番目のパラメータ(自由度の数)
   double&       mean,           // 平均を表す変数
   double&       variance,       // 分散を表す変数
   double&       skewness,       // 歪みを表す変数
   double&       kurtosis,       // 尖度を表す変数
   int&          error_code      // エラーコードを表す変数
   );
```

パラメータ

nu1

[in] 分布の1番目のパラメータ(自由度の数)

nu2

[in] 分布の2番目のパラメータ(自由度の数)

mean

[out] 平均値を取得するための変数

variance

[out] 分散を取得するための変数

skewness

[out] 歪みを取得するための変数

kurtosis

[out] 尖度を取得するための変数

error_code

[out] エラーコードを取得するための変数

戻り値

モーメントの計算が成功した場合はtrue、そうでない場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

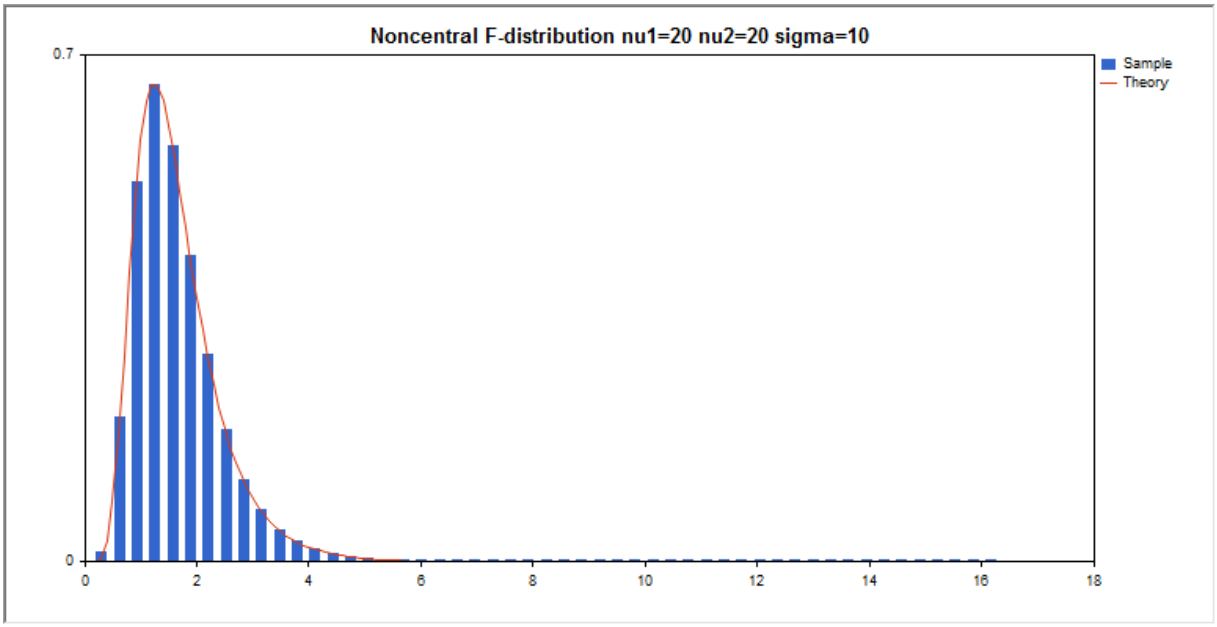
非心F分布

このセクションには、非心F分布を使用するための関数が含まれています。密度、確率、分位数を計算し、非心F分布に従って分布した疑似乱数を生成することができます。非心F分布は下記の式で定義されています。

$$f_{NoncentralF}(x|\nu_1,\nu_2,\sigma)=e^{-\frac{\sigma}{2}}\sum_{r=0}^{\infty}\frac{1}{r!}\left(\frac{\sigma}{2}\right)^r\frac{\Gamma\left(\frac{\nu_1+\nu_2}{2}+r\right)}{\Gamma\left(\frac{\nu_2}{2}+r\right)\Gamma\left(\frac{\nu_2}{2}\right)}\left(\frac{\nu_1}{\nu_2}\right)^{\frac{\nu_2}{2}+r}\frac{x^{\frac{\nu_2}{2}-1+r}}{\left(1+\frac{\nu_1}{\nu_2}x\right)^{\frac{\nu_1+\nu_2}{2}+r}}$$

ここで

- x — 確率変数の値
- ν1 — 分布の1番目のパラメータ(自由度の数)
- ν2 — 分布の2番目のパラメータ(自由度の数)</
- σ — 非心パラメータ



ライブラリは、個々の確率変数の計算に加えて、確率変数の配列を扱う能力も実装しています。

関数	説明
MathProbabilityDensityNoncentralF	非心F分布の確率密度関数を計算します
MathCumulativeDistributionNoncentralF	非心F分布関数の値を計算します
MathQuantileNoncentralF	指定された確率に対する非心F逆分布関数の値を計算します
MathRandomNoncentralF	非心F分布に従って分布した擬似乱数変数/擬似乱数変数の配列を生成します

© 2000–2025, MetaQuotes Ltd.

関数	説明
MathMomentsNoncentralF	非心F分布の最初の4次のモーメントの理論的数値を計算します

例：

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\NoncentralF.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 入力パラメータ
input double nu_1=20;     // 1番目の自由度の数
input double nu_2=20;     // 2番目の自由度の数
input double sig=10;      // 非心パラメータ
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 価格チャートを非表示にする
   ChartSetInteger(0,CHART_SHOW,false);
//--- 擬似乱数生成器を初期化する
   MathSrand(GetTickCount());
//--- 確率変数のサンプルを生成する
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // サンプルの値の数
   int ncells=51;          // ヒストグラムの間隔の数
   double x[];             // ヒストグラム間隔の中心
   double y[];             // 間隔内のサンプルからの値の数
   double data[];          // 確率変数値のサンプル
   double max,min;         // サンプルの最大値と最小値
//--- フィッシャーのF分布からサンプルを取得する
   MathRandomNoncentralF(nu_1,nu_2,sig,n,data);
//--- ヒストグラムをプロットするためのデータを計算する
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 理論曲線をプロットするシーケンス境界とステップを取得する
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- [min、max]の間隔で理論的に計算されたデータを取得する
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNoncentralF(x2,nu_1,nu_2,sig,false,y2);
//--- 規模を設定する
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
```

© 2000-2025, MetaQuotes Ltd.

```
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- チャートを出力する
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Noncentral F-distribution nu1=%G nu2=%G
sigma=%G",nu_1,nu_2,sig));
   graphic.BackgroundMainSize(16);
//--- カーブをプロットする
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 今度は分布密度の理論曲線をプロットする
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- カーブをプロットする
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|   データセットの頻度を計算する                                   |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double
&frequency[],
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 間隔の中心を定義する
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 間隔内に入る頻度を記入する
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
    return (true);
   }
//+------------------------------------------------------------------+
//|   シーケンス生成の値を計算する                                   |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
   {
//--- 正規化の精度を得るためにシーケンスの絶対範囲を計算する
    double range=MathAbs(maxv-minv);
    int degree=(int)MathRound(MathLog10(range));
//--- 最大値と最小値を指定された精度に正規化する
    maxv=NormalizeDouble(maxv,degree);
    minv=NormalizeDouble(minv,degree);
//--- 指定された精度に基づいてシーケンス生成ステップも設定される
    stepv=NormalizeDouble(MathPow(10,-degree),degree);
    if((maxv-minv)/stepv<10)
       stepv/=10.;
   }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityNoncentralF

確率変数xに対して、パラメータnu1、nu2、sigmaを用いてFisherの非心F分布の確率密度関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityNoncentralF(
   const double  x,              // 確率変数の値
   const double  nu1,            // 分布の1番目のパラメータ(自由度の数)
   const double  nu2,            // 分布の2番目のパラメータ(自由度の数)
   const double  sigma,          // 非心パラメータ
   const bool    log_mode,       // 値の対数を計算する。log_mode=trueの場合は、確率密度の自然対数が計算される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータnu1、nu2、sigmaを用いてFisherの非心F分布の確率密度関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityNoncentralF(
   const double  x,              // 確率変数の値
   const double  nu1,            // 分布の1番目のパラメータ(自由度の数)
   const double  nu2,            // 分布の2番目のパラメータ(自由度の数)
   const double  sigma,          // 非心パラメータ
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対して、パラメータnu1、nu2、sigmaを用いてFisherの非心F分布の確率密度関数の値を計算します。エラーの場合は、falseを返します。Rのdf()の類似体です。

```
bool  MathProbabilityDensityNoncentralF(
   const double& x[],            // 確率変数の値を持つ配列
   const double  nu1,            // 分布の1番目のパラメータ(自由度の数)
   const double  nu2,            // 分布の2番目のパラメータ(自由度の数)
   const double  sigma,          // 非心パラメータ
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率密度の自然対数が計算される
   double&       result[]        // 確率密度関数の値の配列
   );
```

確率変数の配列x[]に対して、パラメータnu1、nu2、sigmaを用いてFisherの非心F分布の確率密度関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathProbabilityDensityNoncentralF(
   const double& x[],            // 確率変数の値を持つ配列
   const double  nu1,            // 分布の1番目のパラメータ(自由度の数)
   const double  nu2,            // 分布の2番目のパラメータ(自由度の数)
   const double  sigma,          // 非心パラメータ
   double&       result[]        // 確率密度関数の値の配列
   );
```

パラメータ

© 2000-2025, MetaQuotes Ltd.

x

[in] 確率変数の値

x[]

[in] 確率変数の値を持つ配列

nu1

[in] 分布の1番目のパラメータ(自由度の数)

nu2

[in] 分布の2番目のパラメータ(自由度の数)

sigma

[in] 非心パラメータ.

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率密度の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率密度関数の値の配列

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionNoncentralF

確率変数xに対して、パラメータnu1、nu2、sigmaを用いてFisherの非心F分布の確率分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionNoncentralF(
   const double  x,              // 確率変数の値
   const double  nu1,            // 分布の1番目のパラメータ(自由度の数)
   const double  nu2,            // 分布の2番目のパラメータ(自由度の数)
   const double  sigma,          // 非心パラメータ
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータnu1、nu2、sigmaを用いてFisherの非心F分布の確率分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionNoncentralF(
   const double  x,              // 確率変数の値
   const double  nu1,            // 分布の1番目のパラメータ(自由度の数)
   const double  nu2,            // 分布の2番目のパラメータ(自由度の数)
   const double  sigma,          // 非心パラメータ
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対して、パラメータnu1、nu2、sigmaを用いてFisherの非心F分布の確率分布関数の値を計算します。エラーの場合は、falseを返します。Rのpf()の類似体です。

```
bool  MathCumulativeDistributionNoncentralF(
   const double& x[],            // 確率変数の値を持つ配列
   const double  nu1,            // 分布の1番目のパラメータ(自由度の数)
   const double  nu2,            // 分布の2番目のパラメータ(自由度の数)
   const double  sigma,          // 非心パラメータ
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される
   double&       result[]        // 確率関数の値の配列
   );
```

確率変数の配列x[]に対して、パラメータnu1、nu2、sigmaを用いてFisherの非心F分布の確率分布関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathCumulativeDistributionNoncentralF(
   const double& x[],            // 確率変数の値を持つ配列
   const double  nu1,            // 分布の1番目のパラメータ(自由度の数)
   const double  nu2,            // 分布の2番目のパラメータ(自由度の数)
   const double  sigma,          // 非心パラメータ
```

© 2000-2025, MetaQuotes Ltd.

```
   double&        result[]         // 確率関数の値の配列
   );
```

パラメータ

x

[in] 確率変数の値

x[]

[in] 確率変数の値を持つ配列

nu1

[in] 分布の1番目のパラメータ(自由度の数)

nu2

[in] 分布の2番目のパラメータ(自由度の数)

sigma

[in] 非心パラメータ.

tail

[in] 計算のフラグ。true の場合、確率変数がx 以下である確率を計算する。

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率関数の値の配列

© 2000-2025, MetaQuotes Ltd.

MathQuantileF

この関数は、指定されたprobabilityに対して、パラメータnu1、nu2、sigmaを用いてFisherの非心F逆分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileF(
   const double  probability,     // 確率変数の発生確率値
   const double  nu1,             // 分布の1番目のパラメータ(自由度の数)
   const double  nu2,             // 分布の2番目のパラメータ(自由度の数)
   const double  sigma,           // 非心パラメータ
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定されたprobabilityに対して、パラメータnu1、nu2、sigmaを用いてFisherの非心F逆分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileF(
   const double  probability,     // 確率変数の発生確率値
   const double  nu1,             // 分布の1番目のパラメータ(自由度の数)
   const double  nu2,             // 分布の2番目のパラメータ(自由度の数)
   const double  sigma,           // 非心パラメータ
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータnu1、nu2、sigmaを用いてFisherの非心F逆分布関数の値を計算します。エラーの場合は、falseを返します。Rのqf()の類似体です。

```
double  MathQuantileF(
   const double& probability[],   // 確率変数の確率値を持つ配列
   const double  nu1,             // 分布の1番目のパラメータ(自由度の数)
   const double  nu2,             // 分布の2番目のパラメータ(自由度の数)
   const double  sigma,           // 非心パラメータ
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   double&       result[]         // 分位数の値を持つ配列
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータnu1、nu2、sigmaを用いてFisherの非心F逆分布関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathQuantileF(
   const double& probability[],   // 確率変数の確率値を持つ配列
   const double  nu1,             // 分布の1番目のパラメータ(自由度の数)
   const double  nu2,             // 分布の2番目のパラメータ(自由度の数)
   double&       result[]         // 分位数の値を持つ配列
   );
```

© 2000-2025, MetaQuotes Ltd.

パラメータ

probability

[in] 確率変数の確率値

probability[]

[in] 確率変数の確率値を持つ配列

nu1

[in] 分布の1番目のパラメータ(自由度の数)

nu2

[in] 分布の2番目のパラメータ(自由度の数)

sigma

[in] 非心パラメータ.

tail

[in] 計算のフラグ。falseの場合は1.0確率で計算する

log_mode

[in] 計算のフラグ。log_mode=trueの場合、計算はExp(probability)に対して行われる

error_code

[out] エラーコードを取得するための変数

result[]

[out] 分位数の値を持つ配列

© 2000-2025, MetaQuotes Ltd.

MathRandomNoncentralF

パラメータnu1、nu2、sigmaを使って、Fisherの非心F分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、NaNを返します。

```
double  MathRandomNoncentralF(
   const double  nu1,            // 分布の1番目のパラメータ(自由度の数)
   const double  nu2,            // 分布の2番目のパラメータ(自由度の数)
   const double  sigma,          // 非心パラメータ
   int&          error_code      // エラーコードを格納する変数
   );
```

パラメータnu1、nu2、sigmaを使って、Fisherの非心F分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、falseを返します。Rのrf()の類似体です。

```
bool  MathRandomNoncentralF(
   const double  nu1,            // 分布の1番目のパラメータ(自由度の数)
   const double  nu2,            // 分布の2番目のパラメータ(自由度の数)
   const double  sigma,          // 非心パラメータ
   const int     data_count,     // 必要なデータ量
   double&       result[]        // 擬似乱数変数の値の配列
   );
```

パラメータ

nu1

[in] 分布の1番目のパラメータ(自由度の数)

nu2

[in] 分布の2番目のパラメータ(自由度の数)

sigma

[in] 非心パラメータ.

error_code

[out] エラーコードを格納する変数

data_count

[out] 必要なデータ量

result[]

[out] 擬似乱数の値を取得するための配列

© 2000-2025, MetaQuotes Ltd.

MathMomentsNoncentralF

パラメータnu1、nu2、sigmaを使用して、Fisherの非心F分布の最初の4次のモーメントの理論的数値を計算します。

```
double  MathMomentsNoncentralF(
   const double  nu1,            // 分布の1番目のパラメータ(自由度の数)
   const double  nu2,            // 分布の2番目のパラメータ(自由度の数)
   const double  sigma,          // 非心パラメータ
   double&       mean,           // 平均を表す変数
   double&       variance,       // 分散を表す変数
   double&       skewness,       // 歪みを表す変数
   double&       kurtosis,       // 尖度を表す変数
   int&          error_code      // エラーコードを表す変数
   );
```

パラメータ

nu1

[in] 分布の1番目のパラメータ(自由度の数)

nu2

[in] 分布の2番目のパラメータ(自由度の数)

sigma

[in] 非心パラメータ.

mean

[out] 平均値を取得するための変数

variance

[out] 分散を取得するための変数

skewness

[out] 歪みを取得するための変数

kurtosis

[out] 尖度を取得するための変数

error_code

[out] エラーコードを取得するための変数

戻り値

モーメントの計算が成功した場合はtrue、そうでない場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

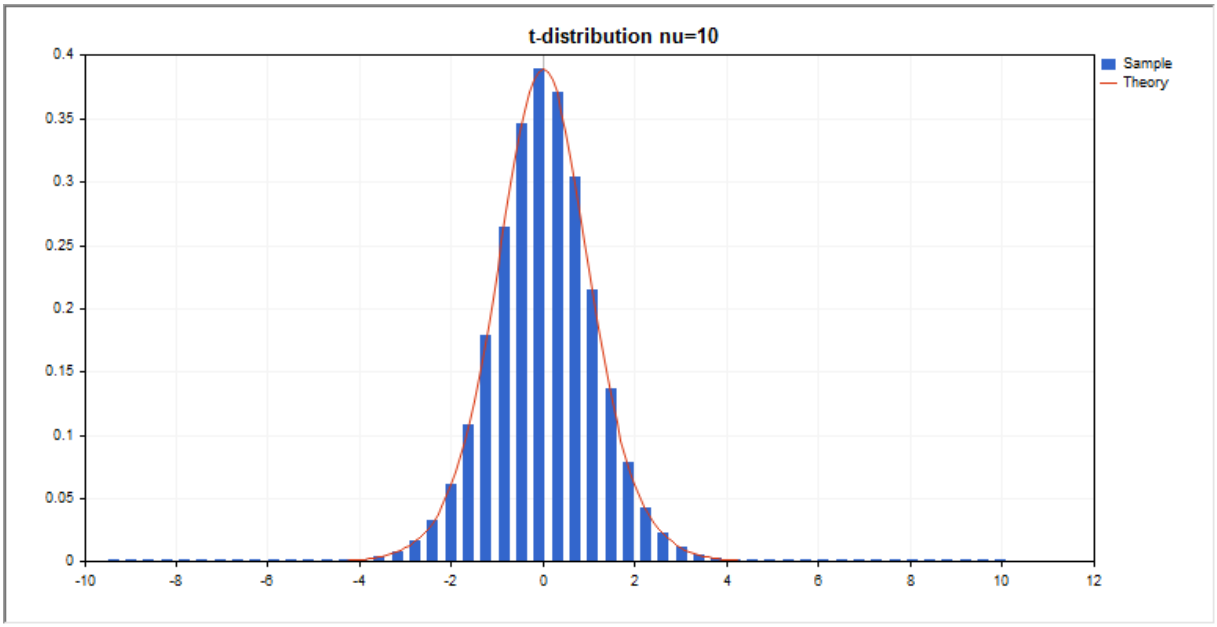
t分布

このセクションには、スチューデントのt分布を使用するための関数が含まれています。密度、確率、分位数を計算し、スチューデントのt分布法則に従って分布した疑似乱数を生成することができます。スチューデントのt分布は下記の式で定義されています。

$$f_T(x|\nu)=\frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)}\frac{1}{\sqrt{\pi\nu}}\frac{1}{\left(1+\frac{x^2}{\nu}\right)^{\frac{\nu+1}{2}}}$$

ここで

- x — 確率変数の値
- ν — 分布のパラメータ(自由度の数)



ライブラリは、個々の確率変数の計算に加えて、確率変数の配列を扱う能力も実装しています。

関数	説明
MathProbabilityDensityT	スチューデントのt分布の確率密度関数を計算します
MathCumulativeDistributionT	t分布関数の値を計算します
MathQuantileT	指定された確率に対するt逆分布関数の値を計算します
MathRandomT	t分布に従って分布した擬似乱数変数/擬似乱数変数の配列を生成します
MathMomentsT	スチューデントのt分布の最初の4次のモーメントの理論的数値を計算します

例:

© 2000–2025, MetaQuotes Ltd.

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\T.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 入力パラメータ
input double nu_par=10;    // 自由度の数
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 価格チャートを非表示にする
   ChartSetInteger(0,CHART_SHOW,false);
//--- 擬似乱数生成器を初期化する
   MathSrand(GetTickCount());
//--- 確率変数のサンプルを生成する
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // サンプルの値の数
   int ncells=51;          // ヒストグラムの間隔の数
   double x[];             // ヒストグラム間隔の中心
   double y[];             // 間隔内のサンプルからの値の数
   double data[];          // 確率変数値のサンプル
   double max,min;         // サンプルの最大値と最小値
//--- スチューデントのt 分布からサンプルを取得する
   MathRandomT(nu_par,n,data);
//--- ヒストグラムをプロットするためのデータを計算する
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 理論曲線をプロットするシーケンス境界とステップを取得する
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- [min、max]の間隔で理論的に計算されたデータを取得する
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityT(x2,nu_par,false,y2);
//--- 規模を設定する
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- チャートを出力する
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.BackgroundMain(StringFormat("t-distribution nu=%G",nu_par));
   graphic.BackgroundMainSize(16);
//--- カーブをプロットする
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 今度は分布密度の理論曲線をプロットする
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- カーブをプロットする
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|   データセットの頻度を計算する                                   |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double
&frequency[],
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 間隔の中心を定義する
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 間隔内に入る頻度を記入する
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|   シーケンス生成の値を計算する                                   |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- 正規化の精度を得るためにシーケンスの絶対範囲を計算する
   double range=MathAbs(maxv-minv);
```

© 2000-2025, MetaQuotes Ltd.

```
   int degree=(int)MathRound(MathLog10(range));
//--- 最大値と最小値を指定された精度に正規化する
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 指定された精度に基づいてシーケンス生成ステップも設定される
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityT

確率変数xに対して、パラメータnuを用いてスチューデントのt分布の確率密度関数を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityT(
   const double  x,              // 確率変数の値
   const double  nu,             // 分布のパラメータ(自由度の数)
   const bool    log_mode,       // 値の対数を計算する。log_mode=trueの場合は、確率密度の自然対数が計算される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータnuを用いてスチューデントのt分布の確率密度関数を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityT(
   const double  x,              // 確率変数の値
   const double  nu,             // 分布のパラメータ(自由度の数)
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対して、パラメータnuを用いてスチューデントのt分布の確率密度関数を計算します。エラーの場合は、falseを返します。Rのdt()の類似体です。

```
bool  MathProbabilityDensityT(
   const double& x[],            // 確率変数の値を持つ配列
   const double  nu,             // 分布のパラメータ(自由度の数)
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率密度の自然対数が計算される
   double&       result[]        // 確率密度関数の値の配列
   );
```

確率変数の配列x[]に対して、パラメータnuを用いてスチューデントのt分布の確率密度関数を計算します。エラーの場合は、falseを返します。

```
bool  MathProbabilityDensityT(
   const double& x[],            // 確率変数の値を持つ配列
   const double  nu,             // 分布のパラメータ(自由度の数)
   double&       result[]        // 確率密度関数の値の配列
   );
```

パラメータ

x

[in] 確率変数の値

x[]

[in] 確率変数の値を持つ配列

nu

[in] 分布のパラメータ(自由度の数)

© 2000-2025, MetaQuotes Ltd.

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率密度の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率密度関数の値の配列

© 2000–2025, MetaQuotes Ltd.

MathCumulativeDistributionT

確率変数xに対して、パラメータnuを用いてスチューデントのt分布の確率分布関数を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionT(
   const double  x,              // 確率変数の値
   const double  nu,             // 分布のパラメータ(自由度の数)
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータnuを用いてスチューデントのt分布の確率分布関数を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionT(
   const double  x,              // 確率変数の値
   const double  nu,             // 分布のパラメータ(自由度の数)
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対して、パラメータnuを用いてスチューデントのt分布の確率分布関数を計算します。エラーの場合は、falseを返します。Rのpt()の類似体です。

```
bool  MathCumulativeDistributionT(
   const double& x[],            // 確率変数の値を持つ配列
   const double  nu,             // 分布のパラメータ(自由度の数)
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される
   double&       result[]        // 確率関数の値の配列
   );
```

確率変数の配列x[]に対して、パラメータnuを用いてスチューデントのt分布の確率分布関数を計算します。エラーの場合は、falseを返します。

```
bool  MathCumulativeDistributionT(
   const double& x[],            // 確率変数の値を持つ配列
   const double  nu,             // 分布のパラメータ(自由度の数)
   double&       result[]        // 確率関数の値の配列
   );
```

パラメータ

x

[in] 確率変数の値

x[]

© 2000-2025, MetaQuotes Ltd.

[in] 確率変数の値を持つ配列

nu

[in] 分布のパラメータ（自由度の数）

tail

[in] 計算のフラグ。true の場合、確率変数がx 以下である確率を計算する。

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率関数の値の配列

© 2000–2025, MetaQuotes Ltd.

MathQuantileT

この関数は、指定されたprobabilityに対して、パラメータnuを用いてスチューデントのt逆分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileT(
   const double  probability,     // 確率変数の発生確率値
   const double  nu,              // 分布のパラメータ(自由度の数)
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定されたprobabilityに対して、パラメータnuを用いてスチューデントのt逆分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileT(
   const double  probability,     // 確率変数の発生確率値
   const double  nu,              // 分布のパラメータ(自由度の数)
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータnuを用いてスチューデントのt逆分布関数の値を計算します。エラーの場合は、falseを返します。Rのqt()の類似体です。

```
double  MathQuantileT(
   const double& probability[],   // 確率変数の確率値を持つ配列
   const double  nu,              // 分布のパラメータ(自由度の数)
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   double&       result[]         // 分位数の値を持つ配列
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータnuを用いてスチューデントのt逆分布関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathQuantileT(
   const double& probability[],   // 確率変数の確率値を持つ配列
   const double  nu,              // 分布のパラメータ(自由度の数)
   double&       result[]         // 分位数の値を持つ配列
   );
```

パラメータ

probability

[in] 確率変数の確率値

probability[]

[in] 確率変数の確率値を持つ配列

nu

© 2000-2025, MetaQuotes Ltd.

[in] 分布のパラメータ（自由度の数）

tail

[in] 計算のフラグ。falseの場合は1.0確率で計算する

log_mode

[in] 計算のフラグ。log_mode=trueの場合、計算はExp(probability)に対して行われる

error_code

[out] エラーコードを取得するための変数

result[]

[out] 分位数の値を持つ配列

© 2000-2025, MetaQuotes Ltd.

MathRandomT

パラメータnuを使って、スチューデントのt分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、NaNを返します。

```
double  MathRandomT(
   const double  nu,              // 分布のパラメータ(自由度の数)
   int&          error_code       // エラーコードを格納する変数
   );
```

パラメータnuを使って、スチューデントのt分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、falseを返します。Rのrt()の類似体です。

```
bool  MathRandomT(
   const double  nu,              // 分布のパラメータ(自由度の数)
   const int     data_count,      // 必要なデータ量
   double&       result[]         // 擬似乱数変数の値の配列
   );
```

パラメータ

nu

[in] 分布のパラメータ(自由度の数)

error_code

[out] エラーコードを格納する変数

data_count

[out] 必要なデータ量

result[]

[out] 擬似乱数の値を取得するための配列

© 2000-2025, MetaQuotes Ltd.

MathMomentsT

パラメータnuを使用して、スチューデントのt分布の最初の4次のモーメントの理論的数値を計算します。

```
double  MathMomentsT(
   const double  nu,              // 分布のパラメータ(自由度の数)
   double&       mean,            // 平均を表す変数
   double&       variance,        // 分散を表す変数
   double&       skewness,        // 歪みを表す変数
   double&       kurtosis,        // 尖度を表す変数
   int&          error_code       // エラーコードを表す変数
   );
```

パラメータ

nu

[in] 分布のパラメータ(自由度の数)

mean

[out] 平均値を取得するための変数

variance

[out] 分散を取得するための変数

skewness

[out] 歪みを取得するための変数

kurtosis

[out] 尖度を取得するための変数

error_code

[out] エラーコードを取得するための変数

戻り値

モーメントの計算が成功した場合はtrue、そうでない場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

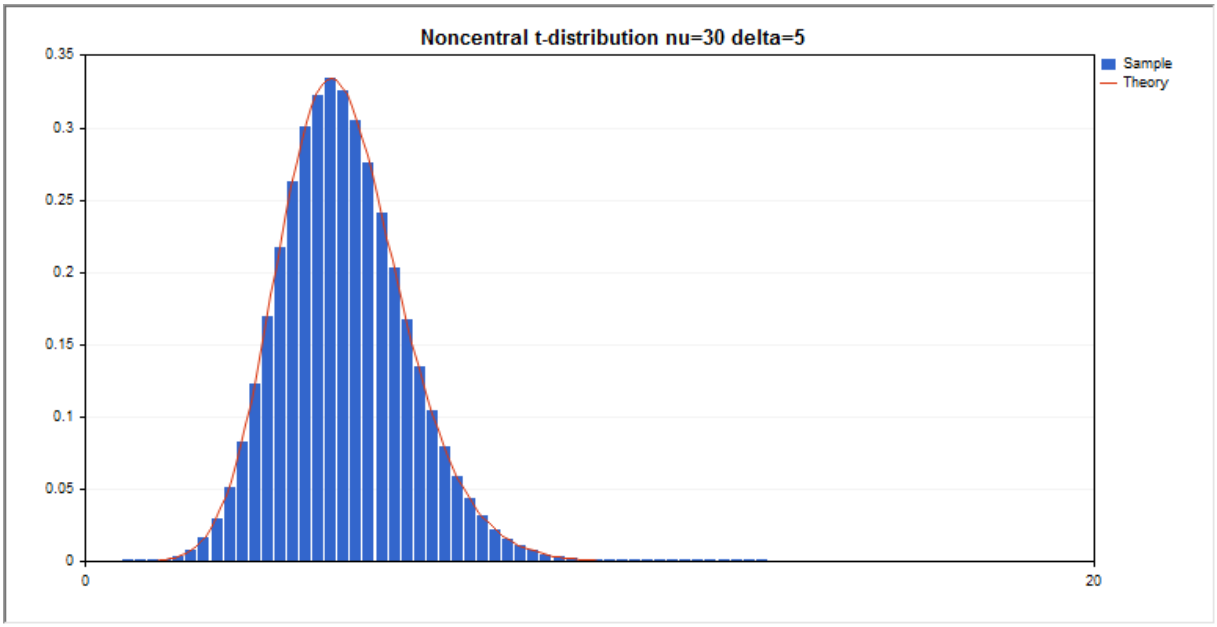
t分布

このセクションには、非心t分布を使用するための関数が含まれています。密度、確率、分位数を計算し、非心t分布に従って分布した疑似乱数を生成することができます。非心t分布は下記の式で定義されています。

$$f_{NoncentralT}(x|\nu,\delta)=\frac{\nu^{\frac{\nu}{2}}e^{-\frac{\sigma^2}{2}}}{\Gamma\left(\frac{\nu}{2}\right)\sqrt{\pi}\left(\nu+x^2\right)^{\frac{\nu+1}{2}}}\sum_{r=0}^{\infty}\frac{(x\delta)^r}{r!}\left(\frac{2}{\nu+x^2}\right)^{\frac{r}{2}}\Gamma\left(\frac{\nu+r+1}{2}\right)$$

ここで

- x — 確率変数の値
- ν — 分布のパラメータ(自由度の数)
- σ — 非心パラメータ



ライブラリは、個々の確率変数の計算に加えて、確率変数の配列を扱う能力も実装しています。

関数	説明
MathProbabilityDensityNoncentralT	非心t分布の確率密度関数を計算します
MathCumulativeDistributionNoncentralT	非心t分布関数の値を計算します
MathQuantileNoncentralT	指定された確率に対する非心t逆分布関数の値を計算します
MathRandomNoncentralT	非心t分布に従って分布した擬似乱数変数/擬似乱数変数の配列を生成します
MathMomentsNoncentralT	非心t分布の最初の4次のモーメントの理論的数値を計算します

© 2000–2025, MetaQuotes Ltd.

例：

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\NoncentralT.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 入力パラメータ
input double nu_par=30;       // 自由度の数
input double delta_par=5;     // 非心パラメータ
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 価格チャートを非表示にする
   ChartSetInteger(0,CHART_SHOW,false);
//--- 擬似乱数生成器を初期化する
   MathSrand(GetTickCount());
//--- 確率変数のサンプルを生成する
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // サンプルの値の数
   int ncells=51;          // ヒストグラムの間隔の数
   double x[];             // ヒストグラム間隔の中心
   double y[];             // 間隔内のサンプルからの値の数
   double data[];          // 確率変数値のサンプル
   double max,min;         // サンプルの最大値と最小値
//--- 非心t分布からサンプルを取得する
   MathRandomNoncentralT(nu_par,delta_par,n,data);
//--- ヒストグラムをプロットするためのデータを計算する
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 理論曲線をプロットするシーケンス境界とステップを取得する
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- [min、max]の間隔で理論的に計算されたデータを取得する
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNoncentralT(x2,nu_par,delta_par,false,y2);
//--- 規模を設定する
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- チャートを出力する
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
```

© 2000-2025, MetaQuotes Ltd.

```
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Noncentral t-distribution nu=%G delta=%
G",nu_par,delta_par));
   graphic.BackgroundMainSize(16);
//--- カーブをプロットする
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 今度は分布密度の理論曲線をプロットする
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- カーブをプロットする
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  データセットの頻度を計算する                                    |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double
&frequency[],
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 間隔の中心を定義する
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 間隔内に入る頻度を記入する
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  シーケンス生成の値を計算する                                    |
//+------------------------------------------------------------------+
```

© 2000–2025, MetaQuotes Ltd.

```
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- 正規化の精度を得るためにシーケンスの絶対範囲を計算する
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- 最大値と最小値を指定された精度に正規化する
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 指定された精度に基づいてシーケンス生成ステップも設定される
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityNoncentralT

確率変数xに対して、パラメータnu、deltaを用いて非心t分布の確率密度関数を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityNoncentralT(
   const double  x,              // 確率変数の値
   const double  nu,             // 分布のパラメータ(自由度の数)
   const double  delta,          // 非心パラメータ
   const bool    log_mode,       // 値の対数を計算する。log_mode=trueの場合は、確率密度の自然対数が計算される
   int&          error_code      // エラーコードを表す変数
   );
```

確率変数xに対して、パラメータnu、deltaを用いて非心t分布の確率密度関数を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityNoncentralT(
   const double  x,              // 確率変数の値
   const double  nu,             // 分布のパラメータ(自由度の数)
   const double  delta,          // 非心パラメータ
   int&          error_code      // エラーコードを表す変数
   );
```

確率変数の配列x[]に対して、パラメータnu、deltaを用いて非心t分布の確率密度関数を計算します。エラーの場合は、falseを返します。Rのdt()の類似体です。

```
bool  MathProbabilityDensityNoncentralT(
   const double& x[],            // 確率変数の値を持つ配列
   const double  nu,             // 分布のパラメータ(自由度の数)
   const double  delta,          // 非心パラメータ
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率密度の自然対数が計算される
   double&       result[]        // 確率密度関数の値の配列
   );
```

確率変数の配列x[]に対して、パラメータnu、deltaを用いて非心t分布の確率密度関数を計算します。エラーの場合は、falseを返します。

```
bool  MathProbabilityDensityNoncentralT(
   const double& x[],            // 確率変数の値を持つ配列
   const double  nu,             // 分布のパラメータ(自由度の数)
   const double  delta,          // 非心パラメータ
   double&       result[]        // 確率密度関数の値の配列
   );
```

パラメータ

x

[in] 確率変数の値

x[]

© 2000-2025, MetaQuotes Ltd.

[in] 確率変数の値を持つ配列

nu

[in] 分布のパラメータ(自由度の数)

delta

[in] 非心パラメータ.

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率密度の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率密度関数の値の配列

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionNoncentralT

確率変数xに対して、パラメータnu、deltaを用いて非心t分布の確率分布関数を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionNoncentralT(
   const double  x,              // 確率変数の値
   const double  nu,             // 分布のパラメータ(自由度の数)
   const double  delta,          // 非心パラメータ
   const bool    tail,           // 計算のフラグ。 true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータnu、deltaを用いて非心t分布の確率分布関数を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionNoncentralT(
   const double  x,              // 確率変数の値
   const double  nu,             // 分布のパラメータ(自由度の数)
   const double  delta,          // 非心パラメータ
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータnu、deltaを用いて非心t分布の確率分布関数を計算します。エラーの場合は、falseを返します。Rのpt()の類似体です。

```
bool  MathCumulativeDistributionNoncentralT(
   const double& x[],            // 確率変数の値を持つ配列
   const double  nu,             // 分布のパラメータ(自由度の数)
   const double  delta,          // 非心パラメータ
   const bool    tail,           // 計算のフラグ。 true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される
   double&       result[]        // 確率関数の値の配列
   );
```

<t0>確率変数の配列x[]に対して、パラメータnu、deltaを用いて非心t分布の確率分布関数を計算します。エラーの場合は、falseを返します。

```
bool  MathCumulativeDistributionNoncentralT(
   const double& x[],            // 確率変数の値を持つ配列
   const double  nu,             // 分布のパラメータ(自由度の数)
   const double  delta,          // 非心パラメータ
   double&       result[]        // 確率関数の値の配列
   );
```

パラメータ

© 2000–2025, MetaQuotes Ltd.

x

[in] 確率変数の値

x[]

[in] 確率変数の値を持つ配列

nu

[in] 分布のパラメータ（自由度の数）

delta

[in] 非心パラメータ.

tail

[in] 計算のフラグ。true の場合、確率変数がx 以下である確率を計算する。

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率関数の値の配列

© 2000–2025, MetaQuotes Ltd.

MathQuantileNoncentralT

この関数は、指定されたprobabilityに対して、パラメータnu、deltaを用いて非心t逆分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileNoncentralT(
   const double  probability,     // 確率変数の発生確率値
   const double  nu,              // 分布のパラメータ(自由度の数)
   const double  delta,           // 非心パラメータ
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定されたprobabilityに対して、パラメータnu、deltaを用いて非心t逆分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileNoncentralT(
   const double  probability,     // 確率変数の発生確率値
   const double  nu,              // 分布のパラメータ(自由度の数)
   const double  delta,           // 非心パラメータ
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータnu、deltaを用いて非心t逆分布関数の値を計算します。エラーの場合は、falseを返します。Rのqt()の類似体です。

```
double  MathQuantileNoncentralT(
   const double& probability[],   // 確率変数の確率値を持つ配列
   const double  nu,              // 分布のパラメータ(自由度の数)
   const double  delta,           // 非心パラメータ
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   double&       result[]         // 分位数の値を持つ配列
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータnu、deltaを用いて非心t逆分布関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathQuantileNoncentralT(
   const double& probability[],   // 確率変数の確率値を持つ配列
   const double  nu,              // 分布のパラメータ(自由度の数)
   const double  delta,           // 非心パラメータ
   double&       result[]         // 分位数の値を持つ配列
   );
```

パラメータ

probability

© 2000-2025, MetaQuotes Ltd.

[in] 確率変数の確率値

probability[]

[in] 確率変数の確率値を持つ配列

nu

[in] 分布のパラメータ(自由度の数)

delta

[in] 非心パラメータ.

tail

[in] 計算のフラグ。falseの場合は1.0確率で計算する

log_mode

[in] 計算のフラグ。log_mode=trueの場合、計算はExp(probability)に対して行われる

error_code

[out] エラーコードを取得するための変数

result[]

[out] 分位数の値を持つ配列

© 2000-2025, MetaQuotes Ltd.

MathRandomNoncentralT

パラメータnu、deltaを使って、非心t分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、NaNを返します。

```
double  MathRandomNoncentralT(
   const double  nu,              // 分布のパラメータ(自由度の数)
   const double  delta,           // 非心パラメータ
   int&          error_code       // エラーコードを格納する変数
   );
```

パラメータnu、deltaを使って、非心t分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、falseを返します。Rのrt()の類似体です。

```
bool  MathRandomNoncentralT(
   const double  nu,              // 分布のパラメータ(自由度の数)
   const double  delta,           // 非心パラメータ
   const int     data_count,      // 必要なデータ量
   double&       result[]         // 擬似乱数変数の値の配列
   );
```

パラメータ

nu

[in] 分布のパラメータ(自由度の数)

delta

[in] 非心パラメータ.

error_code

[out] エラーコードを格納する変数

data_count

[out] 必要なデータ量

result[]

[out] 擬似乱数の値を取得するための配列

© 2000-2025, MetaQuotes Ltd.

MathMomentsNoncentralT

パラメータnu、deltaを使用して、非心t分布の最初の4次のモーメントの理論的数値を計算します。

```
double  MathMomentsNoncentralT(
   const double  nu,              // 分布のパラメータ(自由度の数)
   const double  delta,           // 非心パラメータ
   double&       mean,            // 平均を表す変数
   double&       variance,        // 分散を表す変数
   double&       skewness,        // 歪みを表す変数
   double&       kurtosis,        // 尖度を表す変数
   int&          error_code       // エラーコードを表す変数
   );
```

パラメータ

nu

[in] 分布のパラメータ(自由度の数)

delta

[in] 非心パラメータ

mean

[out] 平均値を取得するための変数

variance

[out] 分散を取得するための変数

skewness

[out] 歪みを取得するための変数

kurtosis

[out] 尖度を取得するための変数

error_code

[out] エラーコードを取得するための変数

戻り値

モーメントの計算が成功した場合はtrue、そうでない場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

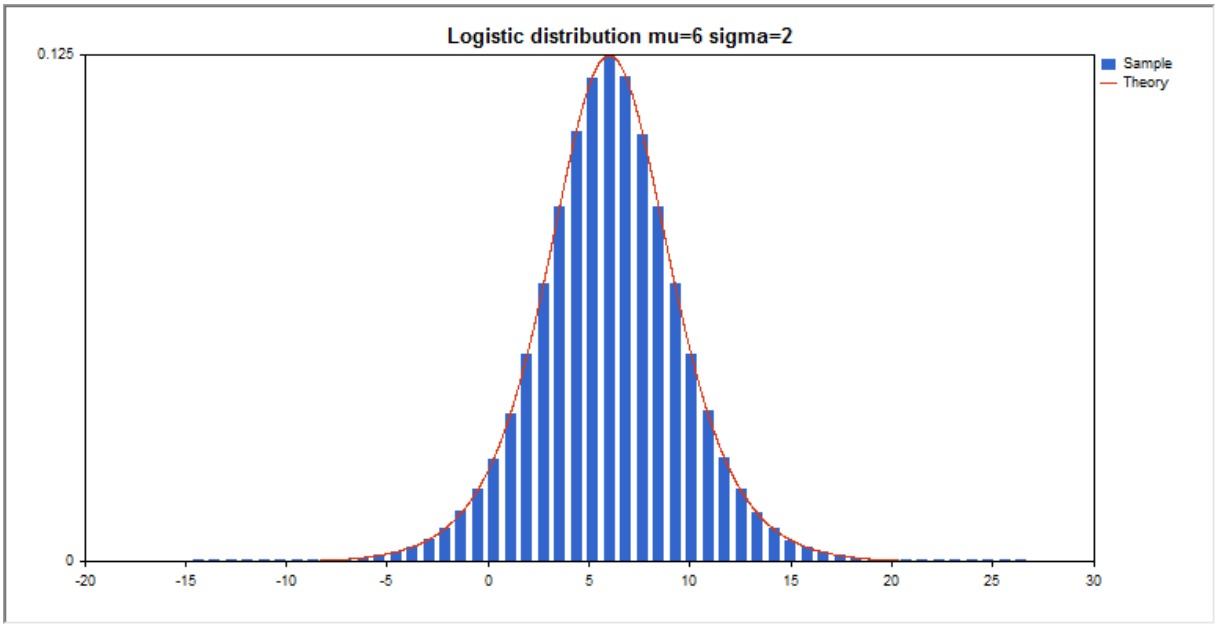
ロジスティック分布

このセクションには、ロジスティック分布を使用するための関数が含まれています。密度、確率、分位数を計算し、ロジスティック法則に従って分布した疑似乱数を生成することができます。ロジスティック分布は下記の式で定義されています。

$$f_{Logistic}(x|\mu,\sigma)=\frac{e^{\frac{x-\mu}{\sigma}}}{\sigma\left(1+e^{\frac{x-\mu}{\sigma}}\right)^2}$$

ここで

- x — 確率変数の値
- μ — 分布の平均パラメータ
- σ — 分布のスケールパラメータ



ライブラリは、個々の確率変数の計算に加えて、確率変数の配列を扱う能力も実装しています。

関数	説明
MathProbabilityDensityLogistic	ロジスティック分布の確率密度関数を計算します
MathCumulativeDistributionLogistic	ロジスティック確率分布関数の値を計算します
MathQuantileLogistic	指定された確率に対するロジスティック逆分布関数の値を計算します
MathRandomLogistic	ロジスティック分布の法則に従って分布した擬似乱数変数/擬似乱数変数の配列を生成します
MathMomentsLogistic	ロジスティック分布の最初の4次のモーメントの理論的数値を計算します

© 2000–2025, MetaQuotes Ltd.

例：

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Logistic.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 入力パラメータ
input double mu_par=6;          // 分布の平均パラメータ
input double sigma_par=2;       // 分布のスケールパラメータ
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 価格チャートを非表示にする
   ChartSetInteger(0,CHART_SHOW,false);
//--- 擬似乱数生成器を初期化する
   MathSrand(GetTickCount());
//--- 確率変数のサンプルを生成する
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // サンプルの値の数
   int ncells=51;          // ヒストグラムの間隔の数
   double x[];             // ヒストグラム間隔の中心
   double y[];             // 間隔内のサンプルからの値の数
   double data[];          // 確率変数値のサンプル
   double max,min;         // サンプルの最大値と最小値
//--- ロジスティク分布からサンプルを取得する
   MathRandomLogistic(mu_par,sigma_par,n,data);
//--- ヒストグラムをプロットするためのデータを計算する
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 理論曲線をプロットするシーケンス境界とステップを取得する
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- [min、max]の間隔で理論的に計算されたデータを取得する
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityLogistic(x2,mu_par,sigma_par,false,y2);
//--- 規模を設定する
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- チャートを出力する
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
```

© 2000-2025, MetaQuotes Ltd.

```
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Logistic distribution mu=%G sigma=%
G",mu_par,sigma_par));
   graphic.BackgroundMainSize(16);
//--- Y軸の自動スケーリングを無効にする
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Max(theor_max);
   graphic.YAxis().Min(0);
//--- カーブをプロットする
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 今度は分布密度の理論曲線をプロットする
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- カーブをプロットする
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|   データセットの頻度を計算する                                   |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double
&frequency[],
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 間隔の中心を定義する
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 間隔内に入る頻度を記入する
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//|   シーケンス生成の値を計算する                                   |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- 正規化の精度を得るためにシーケンスの絶対範囲を計算する
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- 最大値と最小値を指定された精度に正規化する
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 指定された精度に基づいてシーケンス生成ステップも設定される
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityLogistic

確率変数xに対して、パラメータmu、sigmaを用いてロジスティク分布の確率密度関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityLogistic(
   const double  x,              // 確率変数の値
   const double  mu,             // 分布の平均パラメータ
   const double  sigma,          // 分布のスケールパラメータ
   const bool    log_mode,       // 値の対数を計算する。log_mode=trueの場合は、確率密度の自然対数が計算される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータmu、sigmaを用いてロジスティク分布の確率密度関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityLogistic(
   const double  x,              // 確率変数の値
   const double  mu,             // 分布の平均パラメータ
   const double  sigma,          // 分布のスケールパラメータ
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対して、パラメータmu、sigmaを用いてロジスティク分布の確率密度関数の値を計算します。エラーの場合は、falseを返します。Rのdlogis()の類似体です。

```
bool  MathProbabilityDensityLogistic(
   const double& x[],            // 確率変数の値を持つ配列
   const double  mu,             // 分布の平均パラメータ
   const double  sigma,          // 分布のスケールパラメータ
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率密度の自然対数が計算される
   double&       result[]        // 確率密度関数の値の配列
   );
```

確率変数の配列x[]に対して、パラメータmu、sigmaを用いてロジスティク分布の確率密度関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathProbabilityDensityLogistic(
   const double& x[],            // 確率変数の値を持つ配列
   const double  mu,             // 分布の平均パラメータ
   const double  sigma,          // 分布のスケールパラメータ
   double&       result[]        // 確率密度関数の値の配列
   );
```

パラメータ

x

[in] 確率変数の値

x[]

© 2000-2025, MetaQuotes Ltd.

[in] 確率変数の値を持つ配列

mu

[in] 分布の平均パラメータ

sigma

[in] 分布のスケールパラメータ

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率密度の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率密度関数の値の配列

© 2000–2025, MetaQuotes Ltd.

MathCumulativeDistributionLogistic

確率変数xに対して、パラメータmu、sigmaを用いてロジスティク分布の確率分布関数を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionLogistic(
   const double  x,              // 確率変数の値
   const double  mu,             // 分布の平均パラメータ
   const double  sigma,          // 分布のスケールパラメータ
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータmu、sigmaを用いてロジスティク分布の確率分布関数を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionLogistic(
   const double  x,              // 確率変数の値
   const double  mu,             // 分布の平均パラメータ
   const double  sigma,          // 分布のスケールパラメータ
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対して、パラメータmuを用いてロジスティク分布の確率分布関数を計算します。エラーの場合は、falseを返します。

```
bool  MathCumulativeDistributionLogistic(
   const double& x[],            // 確率変数の値を持つ配列
   const double  mu,             // 分布の平均パラメータ
   const double  sigma,          // 分布のスケールパラメータ
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される
   double&       result[]        // 確率関数の値の配列
   );
```

確率変数の配列x[]に対して、パラメータmu、sigmaを用いてロジスティク分布の確率分布関数を計算します。エラーの場合は、falseを返します。Rのplogis()の類似体です。

```
bool  MathCumulativeDistributionLogistic(
   const double& x[],            // 確率変数の値を持つ配列
   const double  mu,             // 分布の平均パラメータ
   const double  sigma,          // 分布のスケールパラメータ
   double&       result[]        // 確率関数の値の配列
   );
```

パラメータ

© 2000–2025, MetaQuotes Ltd.

x

[in] 確率変数の値

x[]

[in] 確率変数の値を持つ配列

mu

[in] 分布の平均パラメータ

sigma

[in] 分布のスケールパラメータ

tail

[in] 計算のフラグ。true の場合、確率変数がx 以下である確率を計算する。

log_mode

[in] 値の対数を計算するためのフラグ。log_ mode=trueの場合、確率の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率関数の値の配列

© 2000–2025, MetaQuotes Ltd.

MathQuantileLogistic

この関数は、指定されたprobabilityに対して、パラメータmu、sigmaを用いてロジスティク逆分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileLogistic(
   const double  probability,    // 確率変数の発生確率値
   const double  mu,             // 分布の平均パラメータ
   const double  sigma,          // 分布のスケールパラメータ
   const bool    tail,           // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,       // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   int&          error_code      // エラーコードを格納する変数
   );
```

この関数は、指定されたprobabilityに対して、パラメータmu、sigmaを用いてロジスティク逆分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileLogistic(
   const double  probability,    // 確率変数の発生確率値
   const double  mu,             // 分布の平均パラメータ
   const double  sigma,          // 分布のスケールパラメータ
   int&          error_code      // エラーコードを格納する変数
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータmu、sigmaを用いてロジスティク逆分布関数の値を計算します。エラーの場合は、falseを返します。Rのqlogis()の類似体です。

```
double  MathQuantileLogistic(
   const double& probability[],  // 確率変数の確率値を持つ配列
   const double  mu,             // 分布の平均パラメータ
   const double  sigma,          // 分布のスケールパラメータ
   const bool    tail,           // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,       // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   double&       result[]        // 分位数の値を持つ配列
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータmu、sigmaを用いてロジスティク逆分布関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathQuantileLogistic(
   const double& probability[],  // 確率変数の確率値を持つ配列
   const double  mu,             // 分布の平均パラメータ
   const double  sigma,          // 分布のスケールパラメータ
   double&       result[]        // 分位数の値を持つ配列
   );
```

パラメータ

probability

© 2000-2025, MetaQuotes Ltd.

[in] 確率変数の確率値

probability[]

[in] 確率変数の確率値を持つ配列

mu

[in] 分布の平均パラメータ

sigma

[in] 分布のスケールパラメータ

tail

[in] 計算のフラグ。falseの場合は1.0確率で計算する

log_mode

[in] 計算のフラグ。log_mode=trueの場合、計算はExp(probability)に対して行われる

error_code

[out] エラーコードを取得するための変数

result[]

[out] 分位数の値を持つ配列

© 2000–2025, MetaQuotes Ltd.

MathRandomLogistic

パラメータmu、sigmaを使って、ロジスティク分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、NaNを返します。

```
double  MathRandomLogistic(
   const double  mu,              // 分布の平均パラメータ
   const double  sigma,           // 分布のスケールパラメータ
   int&          error_code       // エラーコードを格納する変数
   );
```

パラメータmu、sigmaを使って、ロジスティク分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、falseを返します。Rのrlogis()の類似体です。

```
bool  MathRandomLogistic(
   const double  mu,              // 分布の平均パラメータ
   const double  sigma,           // 分布のスケールパラメータ
   const int     data_count,      // 必要なデータ量
   double&       result[]         // 擬似乱数変数の値の配列
   );
```

パラメータ

mu

[in] 分布の平均パラメータ

sigma

[in] 分布のスケールパラメータ

error_code

[out] エラーコードを格納する変数

data_count

[out] 必要なデータ量

result[]

[out] 擬似乱数の値を取得するための配列

© 2000–2025, MetaQuotes Ltd.

MathMomentsLogistic

パラメータmu、sigmaを使用して、ロジスティク分布の最初の4次のモーメントの理論的数値を計算します。

```
double  MathMomentsLogistic(
   const double  mu,              // 分布の平均パラメータ
   const double  sigma,           // 分布のスケールパラメータ
   double&       mean,            // 平均を表す変数
   double&       variance,        // 分散を表す変数
   double&       skewness,        // 歪みを表す変数
   double&       kurtosis,        // 尖度を表す変数
   int&          error_code       // エラーコードを表す変数
   );
```

パラメータ

mu

[in] 分布の平均パラメータ

sigma

[in] 分布のスケールパラメータ

mean

[out] 平均値を取得するための変数

variance

[out] 分散を取得するための変数

skewness

[out] 歪みを取得するための変数

kurtosis

[out] 尖度を取得するための変数

error_code

[out] エラーコードを取得するための変数

戻り値

モーメントの計算が成功した場合はtrue、そうでない場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

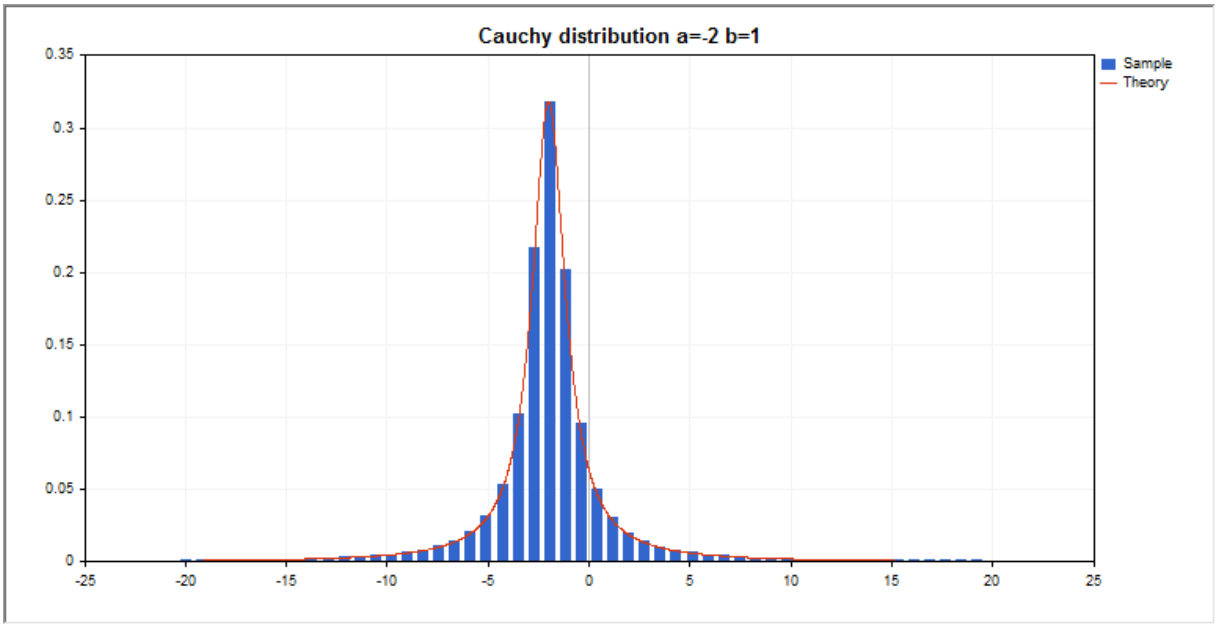
コーシー分布

このセクションには、コーシー分布を使用するための関数が含まれています。密度、確率、分位数を計算し、コーシー法則に従って分布した疑似乱数を生成することができます。コーシー分布は下記の式で定義されています。

$$f_{Cauchy}(x|a,b)=\frac{1}{\pi}\frac{b}{\left(b^2+(x-a)^2\right)}$$

ここで

- x — 確率変数の値
- a — 分布の平均パラメータ
- b — 分布のスケールパラメータ



ライブラリは、個々の確率変数の計算に加えて、確率変数の配列を扱う能力も実装しています。

関数	説明
MathProbabilityDensityCauchy	コーシー分布の確率密度関数を計算します
MathCumulativeDistributionCauchy	コーシー確率分布関数の値を計算します
MathQuantileCauchy	指定された確率に対するコーシー逆分布関数の値を計算します
MathRandomCauchy	コーシー分布の法則に従って分布した擬似乱数変数/擬似乱数変数の配列を生成します
MathMomentsCauchy	コーシー分布の最初の4次のモーメントの理論的数値を計算します

例：

© 2000–2025, MetaQuotes Ltd.

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Cauchy.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 入力パラメータ
input double a_par=-2;       // 分布の平均パラメータ
input double b_par=1;        // 分布のスケールパラメータ
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 価格チャートを非表示にする
   ChartSetInteger(0,CHART_SHOW,false);
//--- 擬似乱数生成器を初期化する
   MathSrand(GetTickCount());
//--- 確率変数のサンプルを生成する
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // サンプルの値の数
   int ncells=51;          // ヒストグラムの間隔の数
   double x[];             // ヒストグラム間隔の中心
   double y[];             // 間隔内のサンプルからの値の数
   double data[];          // 確率変数値のサンプル
   double max,min;         // サンプルの最大値と最小値
//--- コーシー分布からサンプルを取得する
   MathRandomCauchy(a_par,b_par,n,data);
//--- ヒストグラムをプロットするためのデータを計算する
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 理論曲線をプロットするシーケンス境界とステップを取得する
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- [min、max]の間隔で理論的に計算されたデータを取得する
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityCauchy(x2,a_par,b_par,false,y2);
//--- 規模を設定する
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- チャートを出力する
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
```

© 2000-2025, MetaQuotes Ltd.

```
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Cauchy distribution a=%G b=%
G",a_par,b_par));
   graphic.BackgroundMainSize(16);
//--- カーブをプロットする
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 今度は分布密度の理論曲線をプロットする
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- カーブをプロットする
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|   データセットの頻度を計算する                                   |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double
&frequency[],
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1)
      return(false);
   int size=ArraySize(data);
   if(size<cells*10)
      return(false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   Print("min=",minv," max=",maxv);
   minv=-20;
   maxv=20;
   double range=maxv-minv;
   double width=range/cells;
   if(width==0)
      return(false);
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 間隔の中心を定義する
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+i*width;
      frequency[i]=0;
     }
//--- 間隔内に入る頻度を記入する
   for(int i=0; i<size; i++)
     {
      int ind=(int)MathRound((data[i]-minv)/width);
      if(ind>=0 && ind<cells)
         frequency[ind]++;
     }
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//|   シーケンス生成の値を計算する                                   |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- 正規化の精度を得るためにシーケンスの絶対範囲を計算する
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- 最大値と最小値を指定された精度に正規化する
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 指定された精度に基づいてシーケンス生成ステップも設定される
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityCauchy

確率変数xに対して、パラメータa、bを用いてコーシー分布の確率密度関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityCauchy(
   const double  x,              // 確率変数の値
   const double  a,              // 分布の平均パラメータ
   const double  b,              // 分布のスケールパラメータ
   const bool    log_mode,       // 値の対数を計算する。log_mode=trueの場合は、確率密度の自然対数が計算される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータa、bを用いてコーシー分布の確率密度関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityCauchy(
   const double  x,              // 確率変数の値
   const double  a,              // 分布の平均パラメータ
   const double  b,              // 分布のスケールパラメータ
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対して、パラメータa、bを用いてコーシー分布の確率密度関数の値を計算します。エラーの場合は、falseを返します。Rのdcauchy()の類似体です。

```
bool  MathProbabilityDensityCauchy(
   const double& x[],            // 確率変数の値を持つ配列
   const double  a,              // 分布の平均パラメータ
   const double  b,              // 分布のスケールパラメータ
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率密度の自然対数が計算される
   double&       result[]        // 確率密度関数の値の配列
   );
```

確率変数の配列x[]に対して、パラメータa、bを用いてコーシー分布の確率密度関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathProbabilityDensityCauchy(
   const double& x[],            // 確率変数の値を持つ配列
   const double  a,              // 分布の平均パラメータ
   const double  b,              // 分布のスケールパラメータ
   double&       result[]        // 確率密度関数の値の配列
   );
```

パラメータ

x

[in] 確率変数の値

x[]

© 2000-2025, MetaQuotes Ltd.

[in] 確率変数の値を持つ配列

a

[in] 分布の平均パラメータ

b

[in] 分布のスケールパラメータ

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率密度の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率密度関数の値の配列

© 2000–2025, MetaQuotes Ltd.

MathCumulativeDistributionCauchy

確率変数xに対して、パラメータa、bを用いてコーシー分布の確率分布関数を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionCauchy(
   const double  x,              // 確率変数の値
   const double  a,              // 分布の平均パラメータ
   const double  b,              // 分布のスケールパラメータ
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータa、bを用いてコーシー分布の確率分布関数を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionCauchy(
   const double  x,              // 確率変数の値
   const double  a,              // 分布の平均パラメータ
   const double  b,              // 分布のスケールパラメータ
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対してパラメータa、bを用いてコーシー分布の確率分布関数を計算します。エラーの場合は、falseを返します。

```
bool  MathCumulativeDistributionCauchy(
   const double& x[],            // 確率変数の値を持つ配列
   const double  a,              // 分布の平均パラメータ
   const double  b,              // 分布のスケールパラメータ
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される
   double&       result[]        // 確率関数の値の配列
   );
```

確率変数の配列x[]に対してパラメータa、bを用いてコーシー分布の確率分布関数を計算します。エラーの場合は、falseを返します。Rのplogis()の類似体です。

```
bool  MathCumulativeDistributionCauchy(
   const double& x[],            // 確率変数の値を持つ配列
   const double  a,              // 分布の平均パラメータ
   const double  b,              // 分布のスケールパラメータ
   double&       result[]        // 確率関数の値の配列
   );
```

パラメータ

© 2000–2025, MetaQuotes Ltd.

x

[in] 確率変数の値

x[]

[in] 確率変数の値を持つ配列

a

[in] 分布の平均パラメータ

b

[in] 分布のスケールパラメータ

tail

[in] 計算のフラグ。true の場合、確率変数がx 以下である確率を計算する。

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率関数の値の配列

© 2000–2025, MetaQuotes Ltd.

MathQuantileCauchy

この関数は、指定されたprobabilityに対して、パラメータa、bを用いてコーシー逆分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileCauchy(
   const double  probability,     // 確率変数の発生確率値
   const double  a,               // 分布の平均パラメータ
   const double  b,               // 分布のスケールパラメータ
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定されたprobabilityに対して、パラメータa、bを用いてコーシー逆分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileCauchy(
   const double  probability,     // 確率変数の発生確率値
   const double  a,               // 分布の平均パラメータ
   const double  b,               // 分布のスケールパラメータ
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータa、bを用いてコーシー逆分布関数の値を計算します。エラーの場合は、falseを返します。Rのqcauschy()の類似体です。

```
double  MathQuantileCauchy(
   const double& probability[],   // 確率変数の確率値を持つ配列
   const double  a,               // 分布の平均パラメータ
   const double  b,               // 分布のスケールパラメータ
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   double&       result[]         // 分位数の値を持つ配列
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータa、bを用いてコーシー逆分布関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathQuantileCauchy(
   const double& probability[],   // 確率変数の確率値を持つ配列
   const double  a,               // 分布の平均パラメータ
   const double  b,               // 分布のスケールパラメータ
   double&       result[]         // 分位数の値を持つ配列
   );
```

パラメータ

probability

© 2000–2025, MetaQuotes Ltd.

[in] 確率変数の確率値

probability[]

[in] 確率変数の確率値を持つ配列

a

[in] 分布の平均パラメータ

b

[in] 分布のスケールパラメータ

tail

[in] 計算のフラグ。falseの場合は1.0確率で計算する

log_mode

[in] 計算のフラグ。log_mode=trueの場合、計算はExp(probability)に対して行われる

error_code

[out] エラーコードを取得するための変数

result[]

[out] 分位数の値を持つ配列

© 2000–2025, MetaQuotes Ltd.

MathRandomCauchy

パラメータa、bを使って、コーシー分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、NaNを返します。

```
double  MathRandomCauchy(
   const double  a,              // 分布の平均パラメータ
   const double  b,              // 分布のスケールパラメータ
   int&          error_code      // エラーコードを格納する変数
   );
```

パラメータa、bを使って、コーシー分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、falseを返します。Rのrcauchy()の類似体です。

```
bool  MathRandomCauchy(
   const double  a,              // 分布の平均パラメータ
   const double  b,              // 分布のスケールパラメータ
   const int     data_count,     // 必要なデータ量
   double&       result[]        // 擬似乱数変数の値の配列
   );
```

パラメータ

a

[in] 分布の平均パラメータ

b

[in] 分布のスケールパラメータ

error_code

[out] エラーコードを格納する変数

data_count

[out] 必要なデータ量

result[]

[out] 擬似乱数の値を取得するための配列

© 2000-2025, MetaQuotes Ltd.

MathMomentsCauchy

パラメータa、bを使用して、コーシー分布の最初の4次のモーメントの理論的数値を計算します。

```
double  MathMomentsCauchy(
   const double  a,              // 分布の平均パラメータ
   const double  b,              // 分布のスケールパラメータ
   double&       mean,           // 平均を表す変数
   double&       variance,       // 分散を表す変数
   double&       skewness,       // 歪みを表す変数
   double&       kurtosis,       // 尖度を表す変数
   int&          error_code      // エラーコードを表す変数
   );
```

パラメータ

a

[in] 分布の平均パラメータ

b

[in] 分布のスケールパラメータ

mean

[out] 平均値を取得するための変数

variance

[out] 分散を取得するための変数

skewness

[out] 歪みを取得するための変数

kurtosis

[out] 尖度を取得するための変数

error_code

[out] エラーコードを取得するための変数

戻り値

モーメントの計算が成功した場合はtrue、そうでない場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

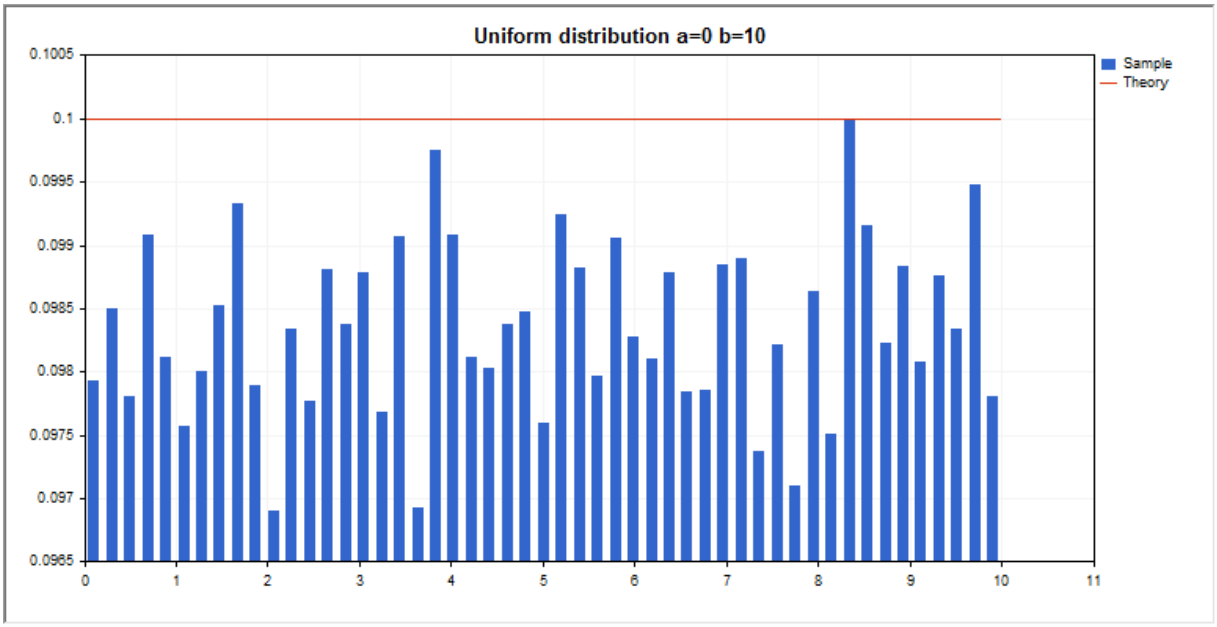
一様分布

このセクションには、一様分布を使用するための関数が含まれています。密度、確率、分位数を計算し、一様分布の法則に従って分布した疑似乱数を生成することができます。一様分布は下記の式で定義されています。

$$f_{Uniform}(x|\,a,b)=\frac{1}{b-a}$$

ここで

- x — 確率変数の値
- a — 分布のパラメータ(下限)
- b — 分布のパラメータ(上限)



ライブラリは、個々の確率変数の計算に加えて、確率変数の配列を扱う能力も実装しています。

関数	説明
MathProbabilityDensityUniform	一様分布の確率密度関数を計算します
MathCumulativeDistributionUniform	一様確率分布関数の値を計算します
MathQuantileUniform	指定された確率に対する一様逆分布関数の値を計算します
MathRandomUniform	一様分布に従って分布した擬似乱数変数/擬似乱数変数の配列を生成します
MathMomentsUniform	一様分布の最初の4次のモーメントの理論的数値を計算します

例:

© 2000–2025, MetaQuotes Ltd.

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Uniform.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 入力パラメータ
input double a_par=0;       // 分布のパラメータa（下限）
input double b_par=10;      // 分布のパラメータb（上限）
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 価格チャートを非表示にする
   ChartSetInteger(0,CHART_SHOW,false);
//--- 擬似乱数生成器を初期化する
   MathSrand(GetTickCount());
//--- 確率変数のサンプルを生成する
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // サンプルの値の数
   int ncells=51;          // ヒストグラムの間隔の数
   double x[];             // ヒストグラム間隔の中心
   double y[];             // 間隔内のサンプルからの値の数
   double data[];          // 確率変数値のサンプル
   double max,min;         // サンプルの最大値と最小値
//--- 一様分布からサンプルを取得する
   MathRandomUniform(a_par,b_par,n,data);
//--- ヒストグラムをプロットするためのデータを計算する
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 理論曲線をプロットするシーケンス境界とステップを取得する
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- [min、max]の間隔で理論的に計算されたデータを取得する
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityUniform(x2,a_par,b_par,false,y2);
//--- 規模を設定する
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- チャートを出力する
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
```

© 2000-2025, MetaQuotes Ltd.

```
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Uniform distribution a=%G b=%
G",a_par,b_par));
   graphic.BackgroundMainSize(16);
//--- カーブをプロットする
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 今度は分布密度の理論曲線をプロットする
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- カーブをプロットする
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  データセットの頻度を計算する                                    |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double
&frequency[],
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 間隔の中心を定義する
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 間隔内に入る頻度を記入する
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  シーケンス生成の値を計算する                                    |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
```

© 2000–2025, MetaQuotes Ltd.

```
//--- 正規化の精度を得るためにシーケンスの絶対範囲を計算する
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- 最大値と最小値を指定された精度に正規化する
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 指定された精度に基づいてシーケンス生成ステップも設定される
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityUniform

確率変数xに対して、パラメータa、bを用いて一様分布の確率密度関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityUniform(
   const double  x,              // 確率変数の値
   const double  a,              // 分布パラメータa（下限）
   const double  b,              // 分布パラメータb（上限）
   const bool    log_mode,       // 値の対数を計算する。log_mode=trueの場合は、確率密度の自然対数が計算される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータa、bを用いて一様分布の確率密度関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityUniform(
   const double  x,              // 確率変数の値
   const double  a,              // 分布パラメータa（下限）
   const double  b,              // 分布パラメータb（上限）
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対して、パラメータa、bを用いて一様分布の確率密度関数の値を計算します。エラーの場合は、falseを返します。Rのdunif()Rのの類似体です。

```
bool  MathProbabilityDensityUniform(
   const double& x[],            // 確率変数の値を持つ配列
   const double  a,              // 分布パラメータa（下限）
   const double  b,              // 分布パラメータb （上限）
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率密度の自然対数が計算される
   double&       result[]        // 確率密度関数の値の配列
   );
```

確率変数の配列x[]に対して、パラメータa、bを用いて一様分布の確率密度関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathProbabilityDensityUniform(
   const double& x[],            // 確率変数の値を持つ配列
   const double  a,              // 分布パラメータa（下限）
   const double  b,              // 分布パラメータb （上限）
   double&       result[]        // 確率密度関数の値の配列
   );
```

パラメータ

x

[in] 確率変数の値

x[]

© 2000–2025, MetaQuotes Ltd.

[in] 確率変数の値を持つ配列

a

[in] 分布パラメータa（下限）.

b

[in] 分布パラメータb（上限）.

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率密度の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率密度関数の値の配列

© 2000–2025, MetaQuotes Ltd.

MathCumulativeDistributionUniform

確率変数xに対して、パラメータa、bを用いて一様分布の確率分布関数を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionUniform(
   const double  x,              // 確率変数の値
   const double  a,              // 分布パラメータa（下限）
   const double  b,              // 分布パラメータb（上限）
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータa、bを用いて一様分布の確率分布関数を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionUniform(
   const double  x,              // 確率変数の値
   const double  a,              // 分布パラメータa（下限）
   const double  b,              // 分布パラメータb（上限）
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対してパラメータa、bを用いて一様分布の確率分布関数を計算します。エラーの場合は、falseを返します。

```
bool  MathCumulativeDistributionUniform(
   const double& x[],            // 確率変数の値を持つ配列
   const double  a,              // 分布パラメータa（下限）
   const double  b,              // 分布パラメータb （上限）
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される
   double&       result[]        // 確率関数の値の配列
   );
```

確率変数の配列x[]に対してパラメータa、bを用いて一様分布の確率分布関数を計算します。エラーの場合は、falseを返します。Rのpunif()の類似体です。

```
bool  MathCumulativeDistributionUniform(
   const double& x[],            // 確率変数の値を持つ配列
   const double  a,              // 分布パラメータa（下限）
   const double  b,              // 分布パラメータb （上限）
   double&       result[]        // 確率関数の値の配列
   );
```

パラメータ

© 2000–2025, MetaQuotes Ltd.

x

[in] 確率変数の値

x[]

[in] 確率変数の値を持つ配列

a

[in] 分布パラメータa（下限）.

b

[in] 分布パラメータb（上限）.

tail

[in] 計算のフラグ。true の場合、確率変数がx 以下である確率を計算する。

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率関数の値の配列

© 2000–2025, MetaQuotes Ltd.

MathQuantileUniform

この関数は、指定されたprobabilityに対して、パラメータa、bを用いて一様逆分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileUniform(
   const double  probability,     // 確率変数の発生確率値
   const double  a,               // 分布パラメータa（下限）
   const double  b,               // 分布パラメータb （上限）
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定されたprobabilityに対して、パラメータa、bを用いて一様逆分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileUniform(
   const double  probability,     // 確率変数の発生確率値
   const double  a,               // 分布パラメータa（下限）
   const double  b,               // 分布パラメータb （上限）
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータa、bを用いて一様逆分布関数の値を計算します。エラーの場合は、falseを返します。Rのqcauschy()の類似体です。

```
double  MathQuantileUniform(
   const double& probability[],   // 確率変数の確率値を持つ配列
   const double  a,               // 分布パラメータa（下限）
   const double  b,               // 分布パラメータb （上限）
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   double&       result[]         // 分位数の値を持つ配列
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータa、bを用いて一様逆分布関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathQuantileUniform(
   const double& probability[],   // 確率変数の確率値を持つ配列
   const double  a,               // 分布パラメータa（下限）
   const double  b,               // 分布パラメータb （上限）
   double&       result[]         // 分位数の値を持つ配列
   );
```

パラメータ

probability

© 2000-2025, MetaQuotes Ltd.

[in] 確率変数の確率値

probability[]

[in] 確率変数の確率値を持つ配列

a

[in] 分布パラメータa（下限）.

b

[in] 分布パラメータb（上限）.

tail

[in] 計算のフラグ。falseの場合は1.0確率で計算する

log_mode

[in] 計算のフラグ。log_mode=trueの場合、計算はExp(probability)に対して行われる

error_code

[out] エラーコードを取得するための変数

result[]

[out] 分位数の値を持つ配列

© 2000–2025, MetaQuotes Ltd.

MathRandomUniform

パラメータa、bを使って、一様分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、NaNを返します。

```
double  MathRandomUniform(
   const double  a,              // 分布パラメータa（下限）
   const double  b,              // 分布パラメータb （上限）
   int&          error_code      // エラーコードを格納する変数
   );
```

パラメータa、bを使って、一様分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、falseを返します。Rのrunif()の類似体です。

```
bool  MathRandomUniform(
   const double  a,              // 分布パラメータa（下限）
   const double  b,              // 分布パラメータb （上限）
   const int     data_count,     // 必要なデータ量
   double&       result[]        // 擬似乱数変数の値の配列
   );
```

パラメータ

a

[in] 分布パラメータa（下限）.

b

[in] 分布パラメータb（上限）.

error_code

[out] エラーコードを格納する変数

data_count

[out] 必要なデータ量

result[]

[out] 擬似乱数の値を取得するための配列

© 2000–2025, MetaQuotes Ltd.

MathMomentsUniform

パラメータa、bを使用して、一様分布の最初の4次のモーメントの理論的数値を計算します。

```
double  MathMomentsUniform(
   const double  a,              // 分布パラメータa（下限）
   const double  b,              // 分布パラメータb （上限）
   double&       mean,           // 平均を表す変数
   double&       variance,       // 分散を表す変数
   double&       skewness,       // 歪みを表す変数
   double&       kurtosis,       // 尖度を表す変数
   int&          error_code      // エラーコードを表す変数
   );
```

パラメータ

a

[in] 分布パラメータa（下限）.

b

[in] 分布パラメータb（上限）.

mean

[out] 平均値を取得するための変数

variance

[out] 分散を取得するための変数

skewness

[out] 歪みを取得するための変数

kurtosis

[out] 尖度を取得するための変数

error_code

[out] エラーコードを取得するための変数

戻り値

モーメントの計算が成功した場合はtrue、そうでない場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

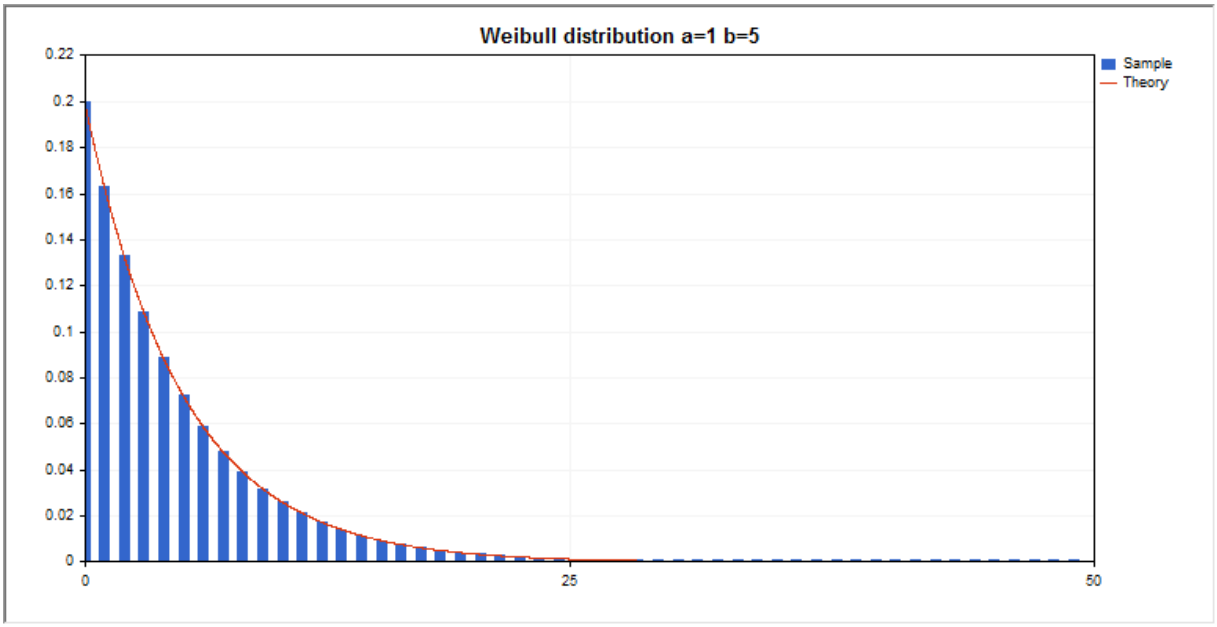
ワイブル分布

このセクションには、ワイブル分布を使用するための関数が含まれています。密度、確率、分位数を計算し、ワイブル分布の法則に従って分布した疑似乱数を生成することができます。ワイブル分布は下記の式で定義されています。

$$f_{Weibull}(x|a,b)=\frac{a}{b}\left(\frac{x}{b}\right)^{a-1}e^{-\left(\frac{x}{b}\right)^{a}}$$

ここで

- x — 確率変数の値
- a — 分布のパラメータ(形状)
- b — 分布のパラメータ(スケール)



ライブラリは、個々の確率変数の計算に加えて、確率変数の配列を扱う能力も実装しています。

関数	説明
MathProbabilityDensityWeibull	ワイブル分布の確率密度関数を計算します
MathCumulativeDistributionWeibull	ワイブル確率分布関数の値を計算します
MathQuantileWeibull	指定された確率に対するワイブル逆分布関数の値を計算します
MathRandomWeibull	ワイブル分布の法則に従って分布した擬似乱数変数/擬似乱数変数の配列を生成します
MathMomentsWeibull	ワイブル分布の最初の4次のモーメントの理論的数値を計算します

例:

© 2000–2025, MetaQuotes Ltd.

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Weibull.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 入力パラメータ
input double a_par=1;       // 分布のパラメータ(形状)
input double b_par=5;       // 分布のパラメータ(スケール)
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 価格チャートを非表示にする
   ChartSetInteger(0,CHART_SHOW,false);
//--- 擬似乱数生成器を初期化する
   MathSrand(GetTickCount());
//--- 確率変数のサンプルを生成する
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // サンプルの値の数
   int ncells=51;          // ヒストグラムの間隔の数
   double x[];             // ヒストグラム間隔の中心
   double y[];             // 間隔内のサンプルからの値の数
   double data[];          // 確率変数値のサンプル
   double max,min;         // サンプルの最大値と最小値
//--- ワイブル分布からサンプルを取得する
   MathRandomWeibull(a_par,b_par,n,data);
//--- ヒストグラムをプロットするためのデータを計算する
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 理論曲線をプロットするシーケンス境界とステップを取得する
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- [min、max]の間隔で理論的に計算されたデータを取得する
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityWeibull(x2,a_par,b_par,false,y2);
//--- 規模を設定する
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- チャートを出力する
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
```

© 2000-2025, MetaQuotes Ltd.

```
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Weibull distribution a=%G b=%
G",a_par,b_par));
   graphic.BackgroundMainSize(16);
//--- Y軸の自動スケーリングを無効にする
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Max(max);
   graphic.XAxis().Min(min);
//--- カーブをプロットする
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 今度は分布密度の理論曲線をプロットする
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- カーブをプロットする
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  データセットの頻度を計算する                                    |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double
&frequency[],
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 間隔の中心を定義する
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+i*width;
      frequency[i]=0;
     }
//--- 間隔内に入る頻度を記入する
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|  シーケンス生成の値を計算する                                                |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- 正規化の精度を得るためにシーケンスの絶対範囲を計算する
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- 最大値と最小値を指定された精度に正規化する
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 指定された精度に基づいてシーケンス生成ステップも設定される
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityWeibull

確率変数xに対して、パラメータa、bを用いてワイブル分布の確率密度関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityWeibull(
   const double  x,              // 確率変数の値
   const double  a,              // 分布のパラメータ(形状)
   const double  b,              // 分布のパラメータ(スケール)
   const bool    log_mode,       // 値の対数を計算する。log_mode=trueの場合は、確率密度の自然対数が計算される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータa、bを用いてワイブル分布の確率密度関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityWeibull(
   const double  x,              // 確率変数の値
   const double  a,              // 分布のパラメータ(形状)
   const double  b,              // 分布のパラメータ(スケール)
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対して、パラメータa、bを用いてワイブル分布の確率密度関数の値を計算します。エラーの場合は、falseを返します。Rのdweibull()の類似体です。

```
bool  MathProbabilityDensityWeibull(
   const double& x[],            // 確率変数の値を持つ配列
   const double  a,              // 分布のパラメータ(形状)
   const double  b,              // 分布のパラメータ(スケール)
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率密度の自然対数が計算される
   double&       result[]        // 確率密度関数の値の配列
   );
```

確率変数の配列x[]に対して、パラメータa、bを用いてワイブル分布の確率密度関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathProbabilityDensityWeibull(
   const double& x[],            // 確率変数の値を持つ配列
   const double  a,              // 分布のパラメータ(形状)
   const double  b,              // 分布のパラメータ(スケール)
   double&       result[]        // 確率密度関数の値の配列
   );
```

パラメータ

x

[in] 確率変数の値

x[]

© 2000-2025, MetaQuotes Ltd.

[in] 確率変数の値を持つ配列

a

[in] 分布のパラメータ(スケール)

b

[in] 分布のパラメータ(形状)

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率密度の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率密度関数の値の配列

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionWeibull

確率変数xに対して、パラメータa、bを用いてワイブル分布の確率分布関数を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionWeibull(
   const double  x,              // 確率変数の値
   const double  a,              // 分布のパラメータ(形状)
   const double  b,              // 分布のパラメータ(スケール)
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータa、bを用いてワイブル分布の確率分布関数を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionWeibull(
   const double  x,              // 確率変数の値
   const double  a,              // 分布のパラメータ(形状)
   const double  b,              // 分布のパラメータ(スケール)
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対して、パラメータa、bを用いてワイブル分布の確率分布関数を計算します。エラーの場合は、falseを返します。Rのpweibull()の類似体です。

```
bool  MathCumulativeDistributionWeibull(
   const double& x[],            // 確率変数の値を持つ配列
   const double  a,              // 分布のパラメータ(形状)
   const double  b,              // 分布のパラメータ(スケール)
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される
   double&       result[]        // 確率関数の値の配列
   );
```

確率変数の配列x[]に対して、パラメータa、bを用いてワイブル分布の確率分布関数を計算します。エラーの場合は、falseを返します。

```
bool  MathCumulativeDistributionWeibull(
   const double& x[],            // 確率変数の値を持つ配列
   const double  a,              // 分布のパラメータ(形状)
   const double  b,              // 分布のパラメータ(スケール)
   double&       result[]        // 確率関数の値の配列
   );
```

パラメータ

© 2000-2025, MetaQuotes Ltd.

x

[in] 確率変数の値

x[]

[in] 確率変数の値を持つ配列

a

[in] 分布のパラメータ（スケール）

b

[in] 分布のパラメータ（形状）

tail

[in] 計算のフラグ。true の場合、確率変数がx 以下である確率を計算する。

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率関数の値の配列

© 2000–2025, MetaQuotes Ltd.

MathQuantileWeibull

この関数は、指定されたprobabilityに対して、パラメータa、bを用いてワイブル逆分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileWeibull(
   const double  probability,     // 確率変数の発生確率値
   const double  a,               // 分布のパラメータ(形状)
   const double  b,               // 分布のパラメータ(スケール)
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定されたprobabilityに対して、パラメータa、bを用いてワイブル逆分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileWeibull(
   const double  probability,     // 確率変数の発生確率値
   const double  a,               // 分布のパラメータ(形状)
   const double  b,               // 分布のパラメータ(スケール)
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータa、bを用いてワイブル逆分布関数の値を計算します。エラーの場合は、falseを返します。Rのqweibull()の類似体です。

```
double  MathQuantileWeibull(
   const double& probability[],   // 確率変数の確率値を持つ配列
   const double  a,               // 分布のパラメータ(形状)
   const double  b,               // 分布のパラメータ(スケール)
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   double&       result[]         // 分位数の値を持つ配列
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータa、bを用いてワイブル逆分布関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathQuantileWeibull(
   const double& probability[],   // 確率変数の確率値を持つ配列
   const double  a,               // 分布のパラメータ(形状)
   const double  b,               // 分布のパラメータ(スケール)
   double&       result[]         // 分位数の値を持つ配列
   );
```

パラメータ

probability

© 2000-2025, MetaQuotes Ltd.

[in] 確率変数の確率値

probability[]

[in] 確率変数の確率値を持つ配列

a

[in] 分布のパラメータ(スケール)

b

[in] 分布のパラメータ(形状)

tail

[in] 計算のフラグ。falseの場合は1.0確率で計算する

log_mode

[in] 計算のフラグ。log_mode=trueの場合、計算はExp(probability)に対して行われる

error_code

[out] エラーコードを取得するための変数

result[]

[out] 分位数の値を持つ配列

© 2000-2025, MetaQuotes Ltd.

MathRandomWeibull

パラメータa、bを使って、ワイブル分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、NaNを返します。

```
double  MathRandomWeibull(
   const double  a,              // 分布のパラメータ(形状)
   const double  b,              // 分布のパラメータ(スケール)
   int&          error_code      // エラーコードを格納する変数
   );
```

パラメータa、bを使って、ワイブル分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、falseを返します。Rのrweibull()の類似体です。

```
bool  MathRandomWeibull(
   const double  a,              // 分布のパラメータ(形状)
   const double  b,              // 分布のパラメータ(スケール)
   const int     data_count,     // 必要なデータ量
   double&       result[]        // 擬似乱数変数の値の配列
   );
```

パラメータ

a

[in] 分布のパラメータ(スケール)

b

[in] 分布のパラメータ(形状)

error_code

[out] エラーコードを格納する変数

data_count

[out] 必要なデータ量

result[]

[out] 擬似乱数の値を取得するための配列

© 2000-2025, MetaQuotes Ltd.

MathMomentsWeibull

パラメータa、bを使用して、ワイブル分布の最初の4次のモーメントの理論的数値を計算します。

```
double  MathMomentsWeibull(
   const double  a,              // 分布のパラメータ(形状)
   const double  b,              // 分布のパラメータ(スケール)
   double&       mean,           // 平均を表す変数
   double&       variance,       // 分散を表す変数
   double&       skewness,       // 歪みを表す変数
   double&       kurtosis,       // 尖度を表す変数
   int&          error_code      // エラーコードを表す変数
   );
```

パラメータ

a

[in] 分布のパラメータ(スケール)

b

[in] 分布のパラメータ(形状)

mean

[out] 平均値を取得するための変数

variance

[out] 分散を取得するための変数

skewness

[out] 歪みを取得するための変数

kurtosis

[out] 尖度を取得するための変数

error_code

[out] エラーコードを取得するための変数

戻り値

モーメントの計算が成功した場合はtrue、そうでない場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

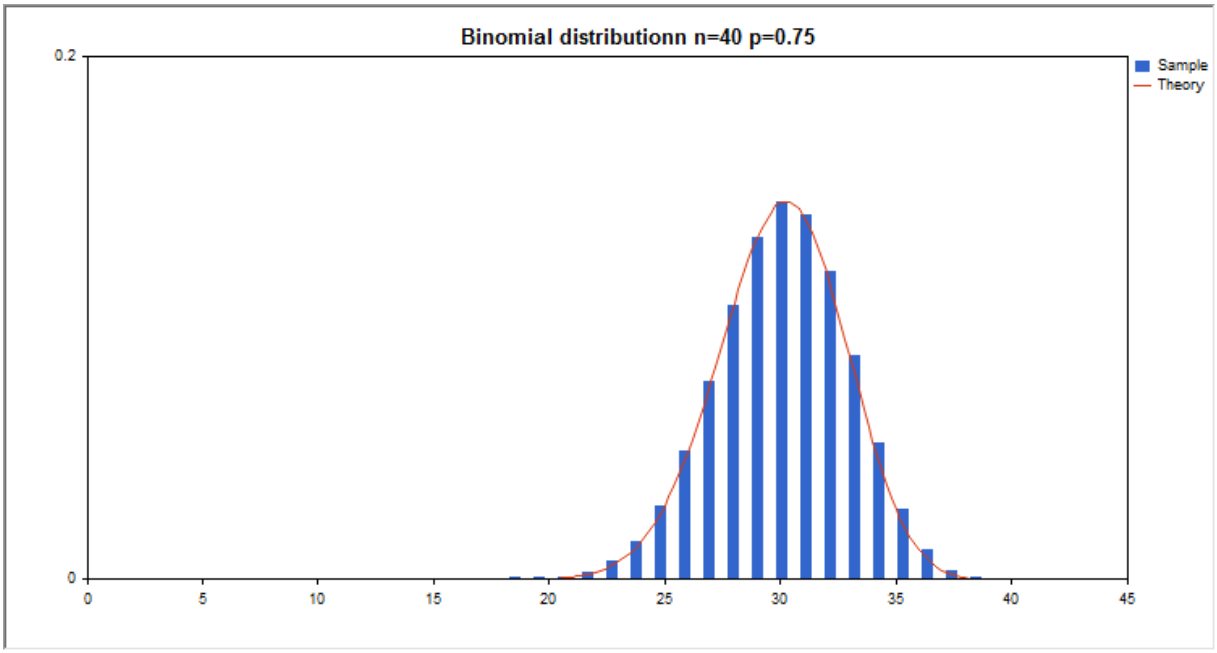
二項分布

このセクションには、二項分布を使用するための関数が含まれています。密度、確率、分位数を計算し、二項法則に従って分布した疑似乱数を生成することができます。二項分布は下記の式で定義されています。

$$f_{Binomial}(x \mid n, p) = \binom{n}{x} p^x (1-p)^{n-x}$$

ここで

- x — 確率変数の値
- n — テスト数
- p — 各テストの成功率



ライブラリは、個々の確率変数の計算に加えて、確率変数の配列を扱う能力も実装しています。

関数	説明
MathProbabilityDensityBinomial	二項分布の確率密度関数を計算します
MathCumulativeDistributionBinomial	二項確率分布関数の値を計算します
MathQuantileBinomial	指定された確率に対する二項逆分布関数の値を計算します
MathRandomBinomial	二項分布に従って分布した擬似乱数変数/擬似乱数変数の配列を生成します
MathMomentsBinomial	二項分布の最初の4次のモーメントの理論的数値を計算します

© 2000–2025, MetaQuotes Ltd.

例：

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Binomial.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 入力パラメータ
input double n_par=40;         // テスト数
input double p_par=0.75;       // 各テストの成功率
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 価格チャートを非表示にする
   ChartSetInteger(0,CHART_SHOW,false);
//--- 擬似乱数生成器を初期化する
   MathSrand(GetTickCount());
//--- 確率変数のサンプルを生成する
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // サンプルの値の数
   int ncells=20;          // ヒストグラムの間隔の数
   double x[];             // ヒストグラム間隔の中心
   double y[];             // 間隔内のサンプルからの値の数
   double data[];          // 確率変数値のサンプル
   double max,min;         // サンプルの最大値と最小値
//--- 二項分布からサンプルを取得する
   MathRandomBinomial(n_par,p_par,n,data);
//--- ヒストグラムをプロットするためのデータを計算する
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- [min、max]の間隔で理論的に計算されたデータを取得する
   double x2[];
   double y2[];
   MathSequence(0,n_par,1,x2);
   MathProbabilityDensityBinomial(x2,n_par,p_par,false,y2);
//--- 規模を設定する
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- チャートを出力する
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.BackgroundMain(StringFormat("Binomial distributionn n=%G p=%
G",n_par,p_par));
   graphic.BackgroundMainSize(16);
//--- カーブをプロットする
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 今度は分布密度の理論曲線をプロットする
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory").LinesSmooth(true);
   graphic.CurvePlotAll();
//--- カーブをプロットする
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|   データセットの頻度を計算する                                   |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double
&frequency[],
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 間隔の中心を定義する
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 間隔内に入る頻度を記入する
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityBinomial

確率変数xに対して、パラメータn、pを用いて二項分布の確率質量関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityBinomial(
   const double  x,              // 確率変数の値
   const double  n,              // 分布のパラメータ(テスト数)
   const double  p,              // 分布のパラメータ(1回のテストでのイベント発生の確率)
   const bool    log_mode,       // 値の対数を計算する。log_mode=trueの場合は、確率密度の自然対数が計算される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータn、pを用いて二項分布の確率質量関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityBinomial(
   const double  x,              // 確率変数の値
   const double  n,              // 分布のパラメータ(テスト数)
   const double  p,              // 分布のパラメータ(1回のテストでのイベント発生の確率)
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対して、パラメータn、pを用いて二項分布の確率質量関数の値を計算します。エラーの場合は、falseを返します。Rのdbinom()の類似体です。

```
bool  MathProbabilityDensityBinomial(
   const double& x[],            // 確率変数の値を持つ配列
   const double  n,              // 分布のパラメータ(テスト数)
   const double  p,              // 分布のパラメータ(1回のテストでのイベント発生の確率)
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率密度の自然対数が計算される
   double&       result[]        // 確率密度関数の値の配列
   );
```

確率変数の配列x[]に対して、パラメータn、pを用いて二項分布の確率質量関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathProbabilityDensityBinomial(
   const double& x[],            // 確率変数の値を持つ配列
   const double  n,              // 分布のパラメータ(テスト数)
   const double  p,              // 分布のパラメータ(1回のテストでのイベント発生の確率)
   double&       result[]        // 確率密度関数の値の配列
   );
```

パラメータ

x

[in] 確率変数の値

x[]

© 2000-2025, MetaQuotes Ltd.

[in] 確率変数の値を持つ配列

n

[in] 分布のパラメータ(テスト数)

p

[in] 分布のパラメータ(1回のテストでのイベント発生の確率)

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率密度の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率密度関数の値の配列

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionBinomial

確率変数xに対して、パラメータn、pを用いて二項分布の確率分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionBinomial(
   const double  x,              // 確率変数の値
   const double  n,              // 分布のパラメータ(テスト数)
   const double  p,              // 分布のパラメータ(1回のテストでのイベント発生の確率)
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータn、pを用いて二項分布の確率分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionBinomial(
   const double  x,              // 確率変数の値
   const double  n,              // 分布のパラメータ(テスト数)
   const double  p,              // 分布のパラメータ(1回のテストでのイベント発生の確率)
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対して、パラメータn、bを用いて二項分布の確率分布関数の値を計算します。エラーの場合は、falseを返します。Rのpweibull()の類似体です。

```
bool  MathCumulativeDistributionBinomial(
   const double& x[],            // 確率変数の値を持つ配列
   const double  n,              // 分布のパラメータ(テスト数)
   const double  p,              // 分布のパラメータ(1回のテストでのイベント発生の確率)
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される
   double&       result[]        // 確率関数の値の配列
   );
```

確率変数の配列x[]に対して、パラメータn、pを用いて二項分布の確率分布関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathCumulativeDistributionBinomial(
   const double& x[],            // 確率変数の値を持つ配列
   const double  n,              // 分布のパラメータ(テスト数)
   const double  p,              // 分布のパラメータ(1回のテストでのイベント発生の確率)
   double&       result[]        // 確率関数の値の配列
   );
```

パラメータ

© 2000-2025, MetaQuotes Ltd.

x

[in] 確率変数の値

x[]

[in] 確率変数の値を持つ配列

n

[in] 分布のパラメータ（テスト数）

p

[in] 分布のパラメータ（1回のテストでのイベント発生の確率）

tail

[in] 計算のフラグ。true の場合、確率変数がx 以下である確率を計算する。

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率関数の値の配列

© 2000–2025, MetaQuotes Ltd.

MathQuantileBinomial

この関数は、指定されたprobabilityに対して、パラメータn、pを用いて二項法の分布関数の逆数値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileBinomial(
   const double  probability,     // 確率変数の発生確率値
   const double  n,               // 分布のパラメータ(テスト数)
   const double  p,               // 分布のパラメータ(1回のテストでのイベント発生の確率)
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定されたprobabilityに対して、パラメータn、pを用いて二項法の分布関数の逆数値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileBinomial(
   const double  probability,     // 確率変数の発生確率値
   const double  n,               // 分布のパラメータ(テスト数)
   const double  p,               // 分布のパラメータ(1回のテストでのイベント発生の確率)
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータn、pを用いて二項法の分布関数の逆数値を計算します。エラーの場合は、falseを返します。Rのqbinom()の類似体です。

```
double  MathQuantileBinomial(
   const double& probability[],   // 確率変数の確率値を持つ配列
   const double  n,               // 分布のパラメータ(テスト数)
   const double  p,               // 分布のパラメータ(1回のテストでのイベント発生の確率)
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   double&       result[]         // 分位数の値を持つ配列
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータn、pを用いて二項法の分布関数の逆数値を計算します。エラーの場合は、falseを返します。

```
bool  MathQuantileBinomial(
   const double& probability[],   // 確率変数の確率値を持つ配列
   const double  n,               // 分布のパラメータ(テスト数)
   const double  p,               // 分布のパラメータ(1回のテストでのイベント発生の確率)
   double&       result[]         // 分位数の値を持つ配列
   );
```

パラメータ

probability

© 2000-2025, MetaQuotes Ltd.

[in] 確率変数の確率値

probability[]

[in] 確率変数の確率値を持つ配列

n

[in] 分布のパラメータ（テスト数）

p

[in] 分布のパラメータ（1回のテストでのイベント発生の確率）

tail

[in] 計算のフラグ。falseの場合は1.0確率で計算する

log_mode

[in] 計算のフラグ。log_mode=trueの場合、計算はExp(probability)に対して行われる

error_code

[out] エラーコードを取得するための変数

result[]

[out] 分位数の値を持つ配列

© 2000-2025, MetaQuotes Ltd.

MathRandomBinomial

パラメータn、pを使って、二項分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、NaNを返します。

```
double  MathRandomBinomial(
   const double  n,              // 分布のパラメータ(テスト数)
   const double  p,              // 分布のパラメータ(1回のテストでのイベント発生の確率)
   int&          error_code      // エラーコードを格納する変数
   );
```

パラメータn、pを使って、二項分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、falseを返します。Rのrweibull()の類似体です。

```
bool  MathRandomBinomial(
   const double  n,              // 分布のパラメータ(テスト数)
   const double  p,              // 分布のパラメータ(1回のテストでのイベント発生の確率)
   const int     data_count,     // 必要なデータ量
   double&       result[]        // 擬似乱数変数の値の配列
   );
```

パラメータ

n

[in] 分布のパラメータ(テスト数)

p

[in] 分布のパラメータ(1回のテストでのイベント発生の確率)

error_code

[out] エラーコードを格納する変数

data_count

[out] 必要なデータ量

result[]

[out] 擬似乱数の値を取得するための配列

© 2000-2025, MetaQuotes Ltd.

MathMomentsBinomial

パラメータn、pを使用して、二項分布の最初の4次のモーメントの理論的数値を計算します。

```
double  MathMomentsBinomial(
   const double  n,              // 分布のパラメータ(テスト数)
   const double  p,              // 分布のパラメータ(1回のテストでのイベント発生の確率)
   double&       mean,           // 平均を表す変数
   double&       variance,       // 分散を表す変数
   double&       skewness,       // 歪みを表す変数
   double&       kurtosis,       // 尖度を表す変数
   int&          error_code      // エラーコードを表す変数
   );
```

パラメータ

n

[in] 分布のパラメータ(テスト数)

p

[in] 分布のパラメータ(1回のテストでのイベント発生の確率)

mean

[out] 平均値を取得するための変数

variance

[out] 分散を取得するための変数

skewness

[out] 歪みを取得するための変数

kurtosis

[out] 尖度を取得するための変数

error_code

[out] エラーコードを取得するための変数

戻り値

モーメントの計算が成功した場合はtrue、そうでない場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

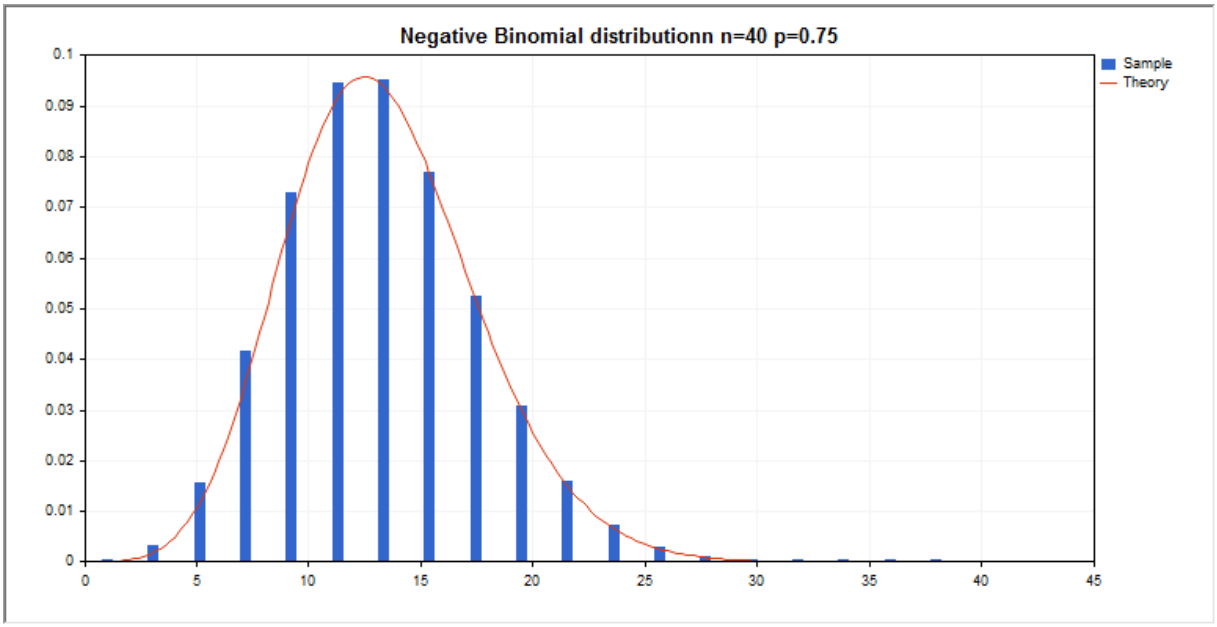
負の二項分布

このセクションには、負の二項分布を使用するための関数が含まれています。密度、確率、分位数を計算し、負の二項法則に従って分布した疑似乱数を生成することができます。負の二項分布は下記の式で定義されています。

$$f_{NegativeBinomial}(x \mid r, p) = \frac{\Gamma(r+x)}{\Gamma(r)\Gamma(x+1)} p^r (1-p)^x$$

ここで

- x — 確率変数の値
- r — 成功したテストの数
- p — 成功率



ライブラリは、個々の確率変数の計算に加えて、確率変数の配列を扱う能力も実装しています。

関数	説明
MathProbabilityDensityNegativeBinomial	負の二項分布の確率密度関数を計算します
MathCumulativeDistributionNegativeBinomial	負の二項確率分布関数の値を計算します
MathQuantileNegativeBinomial	指定された確率に対する負の二項逆分布関数の値を計算します
MathRandomNegativeBinomial	負の二項分布に従って分布した擬似乱数変数/擬似乱数変数の配列を生成します
MathMomentsNegativeBinomial	負の二項分布の最初の4次のモーメントの理論的数値を計算します

© 2000–2025, MetaQuotes Ltd.

例：

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\NegativeBinomial.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 入力パラメータ
input double n_par=40;         // テスト数
input double p_par=0.75;       // 各テストの成功率
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 価格チャートを非表示にする
   ChartSetInteger(0,CHART_SHOW,false);
//--- 擬似乱数生成器を初期化する
   MathSrand(GetTickCount());
//--- 確率変数のサンプルを生成する
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;         // サンプルの値の数
   int ncells=19;         // ヒストグラムの間隔の数
   double x[];            // ヒストグラム間隔の中心
   double y[];            // 間隔内のサンプルからの値の数
   double data[];         // 確率変数値のサンプル
   double max,min;        // サンプルの最大値と最小値
//--- 負の二項分布からサンプルを取得する
   MathRandomNegativeBinomial(n_par,p_par,n,data);
//--- ヒストグラムをプロットするためのデータを計算する
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- [min、max]の間隔で理論的に計算されたデータを取得する
   double x2[];
   double y2[];
   MathSequence(0,n_par,1,x2);
   MathProbabilityDensityNegativeBinomial(x2,n_par,p_par,false,y2);
//--- 規模を設定する
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- チャートを出力する
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.BackgroundMain(StringFormat("Negative Binomial distributionn n=%G p=%
G",n_par,p_par));
   graphic.BackgroundMainSize(16);
//--- カーブをプロットする
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 今度は分布密度の理論曲線をプロットする
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory").LinesSmooth(true);
   graphic.CurvePlotAll();
//--- カーブをプロットする
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|   データセットの頻度を計算する                                   |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double
&frequency[],
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 間隔の中心を定義する
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 間隔内に入る頻度を記入する
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityNegativeBinomial

確率変数xに対して、パラメータr、pを用いて負の二項分布の確率質量関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityNegativeBinomial(
   const double  x,              // 確率変数の値(整数)
   const double  r,              // 成功したテストの数
   const double  p,              // 成功率
   const bool    log_mode,       // 値の対数を計算する。log_mode=trueの場合は、確率密度の自然対数が計算される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータr、pを用いて負の二項分布の確率質量関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityNegativeBinomial(
   const double  x,              // 確率変数の値(整数)
   const double  r,              // 成功したテストの数
   const double  p,              // 成功率
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対して、パラメータr、pを用いて負の二項分布の確率質量関数の値を計算します。エラーの場合は、falseを返します。Rのdnbinom()の類似体です。

```
bool  MathProbabilityDensityNegativeBinomial(
   const double& x[],            // 確率変数の値を持つ配列
   const double  r,              // 成功したテストの数
   const double  p,              // 成功率
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率密度の自然対数が計算される
   double&       result[]        // 確率密度関数の値の配列
   );
```

確率変数の配列x[]に対して、パラメータr、pを用いて負の二項分布の確率質量関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathProbabilityDensityNegativeBinomial(
   const double& x[],            // 確率変数の値を持つ配列
   const double  r,              // 成功したテストの数
   const double  p,              // 成功率
   double&       result[]        // 確率密度関数の値の配列
   );
```

パラメータ

x

[in] 確率変数の値

x[]

© 2000-2025, MetaQuotes Ltd.

[in] 確率変数の値を持つ配列

r

[in] 成功したテストの数

p

[in] 成功率

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率密度の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率密度関数の値の配列

© 2000–2025, MetaQuotes Ltd.

MathCumulativeDistributionNegativeBinomial

確率変数xに対して、パラメータr、pを用いて負の二項分布の確率分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionNegativeBinomial(
   const double  x,              // 確率変数の値(整数)
   const double  r,              // 成功したテストの数
   const double  p,              // 成功率
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータr、pを用いて負の二項分布の確率分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionNegativeBinomial(
   const double  x,              // 確率変数の値(整数)
   const double  r,              // 成功したテストの数
   const double  p,              // 成功率
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対して、パラメータr、pを用いて負の二項分布の確率分布関数の値を計算します。エラーの場合は、falseを返します。Rのpweibull()の類似体です。

```
bool  MathCumulativeDistributionNegativeBinomial(
   const double& x[],            // 確率変数の値を持つ配列
   const double  r,              // 成功したテストの数
   const double  p,              // 成功率
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される
   double&       result[]        // 確率関数の値の配列
   );
```

確率変数の配列x[]に対して、パラメータr、pを用いて負の二項分布の確率分布関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathCumulativeDistributionNegativeBinomial(
   const double& x[],            // 確率変数の値を持つ配列
   const double  r,              // 成功したテストの数
   const double  p,              // 成功率
   double&       result[]        // 確率関数の値の配列
   );
```

パラメータ

© 2000-2025, MetaQuotes Ltd.

x

[in] 確率変数の値

x[]

[in] 確率変数の値を持つ配列

r

[in] 成功したテストの数

p

[in] 成功率

tail

[in] 計算のフラグ。true の場合、確率変数がx 以下である確率を計算する。

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率関数の値の配列

© 2000–2025, MetaQuotes Ltd.

MathQuantileNegativeBinomial

この関数は、指定されたprobabilityに対して、パラメータr、pを用いて負の二項法の分布関数の逆数値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileNegativeBinomial(
   const double  probability,    // 確率変数の発生確率値
   const double  r,              // 成功したテストの数
   const double  p,              // 成功率
   const bool    tail,           // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,       // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   int&          error_code      // エラーコードを格納する変数
   );
```

この関数は、指定されたprobabilityに対して、パラメータr、pを用いて負の二項法の分布関数の逆数値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileNegativeBinomial(
   const double  probability,    // 確率変数の発生確率値
   const double  r,              // 成功したテストの数
   const double  p,              // 成功率
   int&          error_code      // エラーコードを格納する変数
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータr、pを用いて負の二項法の分布関数の逆数値を計算します。エラーの場合は、falseを返します。Rのqnbinom()の類似体です。

```
double  MathQuantileNegativeBinomial(
   const double& probability[],  // 確率変数の確率値を持つ配列
   const double  r,              // 成功したテストの数
   const double  p,              // 成功率
   const bool    tail,           // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,       // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   double&       result[]        // 分位数の値を持つ配列
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータr、pを用いて負の二項法の分布関数の逆数値を計算します。エラーの場合は、falseを返します。

```
bool  MathQuantileNegativeBinomial(
   const double& probability[],  // 確率変数の確率値を持つ配列
   const double  r,              // 成功したテストの数
   const double  p,              // 成功率
   double&       result[]        // 分位数の値を持つ配列
   );
```

パラメータ

probability

© 2000-2025, MetaQuotes Ltd.

[in] 確率変数の確率値

probability[]

[in] 確率変数の確率値を持つ配列

r

[in] 成功したテストの数

p

[in] 成功率

tail

[in] 計算のフラグ。falseの場合は1.0確率で計算する

log_mode

[in] 計算のフラグ。log_mode=trueの場合、計算はExp(probability)に対して行われる

error_code

[out] エラーコードを取得するための変数

result[]

[out] 分位数の値を持つ配列

© 2000-2025, MetaQuotes Ltd.

MathRandomNegativeBinomial

パラメータr、pを使って、負の二項分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、NaNを返します。

```
double  MathRandomNegativeBinomial(
   const double  r,              // 成功したテストの数
   const double  p,              // 成功率
   int&          error_code      // エラーコードを格納する変数
   );
```

パラメータr、pを使って、負の二項分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、falseを返します。Rのrweibull()の類似体です。

```
bool  MathRandomNegativeBinomial(
   const double  r,              // 成功したテストの数
   const double  p,              // 成功率
   const int     data_count,     // 必要なデータ量
   double&       result[]        // 擬似乱数変数の値の配列
   );
```

パラメータ

r

[in] 成功したテストの数

p

[in] 成功率

error_code

[out] エラーコードを格納する変数

data_count

[out] 必要なデータ量

result[]

[out] 擬似乱数の値を取得するための配列

© 2000-2025, MetaQuotes Ltd.

MathMomentsNegativeBinomial

パラメータr、pを使用して、負の二項分布の最初の4次のモーメントの理論的数値を計算します。

```
double  MathMomentsNegativeBinomial(
   const double  r,              // 成功したテストの数
   const double  p,              // 成功率
   double&       mean,           // 平均を表す変数
   double&       variance,       // 分散を表す変数
   double&       skewness,       // 歪みを表す変数
   double&       kurtosis,       // 尖度を表す変数
   int&          error_code      // エラーコードを表す変数
   );
```

パラメータ

r

[in] 成功したテストの数

p

[in] 成功率

mean

[out] 平均値を取得するための変数

variance

[out] 分散を取得するための変数

skewness

[out] 歪みを取得するための変数

kurtosis

[out] 尖度を取得するための変数

error_code

[out] エラーコードを取得するための変数

戻り値

モーメントの計算が成功した場合はtrue、そうでない場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

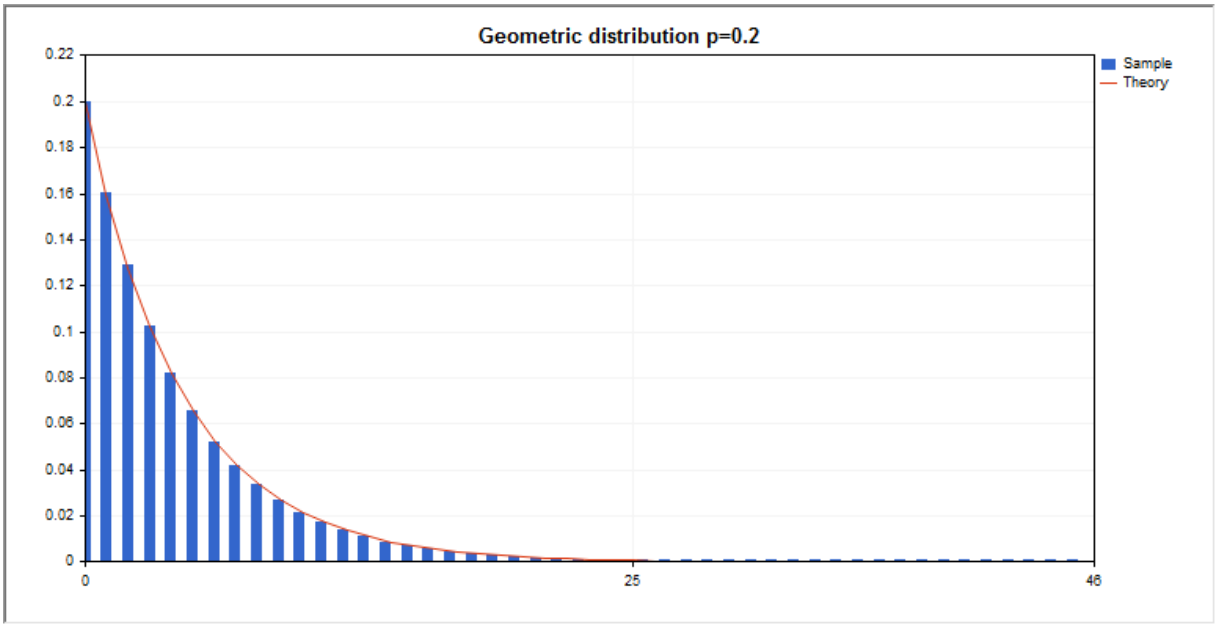
幾何分布

このセクションには、幾何分布を使用するための関数が含まれています。密度、確率、分位数を計算し、幾何法則に従って分布した疑似乱数を生成することができます。幾何分布は下記の式で定義されています。

$$f_{Geometric}(x|p) = p(1-p)^x$$

ここで

- x — 確率変数の値（整数）
- p — 1回のテストでのイベント発生の確率



ライブラリは、個々の確率変数の計算に加えて、確率変数の配列を扱う能力も実装しています。

関数	説明
MathProbabilityDensityGeometric	幾何分布の確率密度関数を計算します
MathCumulativeDistributionGeometric	幾何確率分布関数の値を計算します
MathQuantileGeometric	指定された確率に対する幾何逆分布関数の値を計算します
MathRandomGeometric	幾何分布の法則に従って分布した擬似乱数変数/擬似乱数変数の配列を生成します
MathMomentsGeometric	幾何分布の最初の4次のモーメントの理論的数値を計算します

例：

```
#include <Graphics\Graphic.mqh>
```

© 2000–2025, MetaQuotes Ltd.

```
#include <Math\Stat\Geometric.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 入力パラメータ
input double p_par=0.2;       // 1回のテストでのイベント発生率
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 価格チャートを非表示にする
   ChartSetInteger(0,CHART_SHOW,false);
//--- 擬似乱数生成器を初期化する
   MathSrand(GetTickCount());
//--- 確率変数のサンプルを生成する
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // サンプルの値の数
   int ncells=47;          // ヒストグラムの間隔の数
   double x[];             // ヒストグラム間隔の中心
   double y[];             // 間隔内のサンプルからの値の数
   double data[];          // 確率変数値のサンプル
   double max,min;         // サンプルの最大値と最小値
//--- 幾何分布からサンプルを取得する
   MathRandomGeometric(p_par,n,data);
//--- ヒストグラムをプロットするためのデータを計算する
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 理論曲線をプロットするシーケンス境界とステップを取得する
   double step;
   GetMaxMinStepValues(max,min,step);
   PrintFormat("max=%G min=%G",max,min);
//--- [min、max]の間隔で理論的に計算されたデータを取得する
   double x2[];
   double y2[];
   MathSequence(0,ncells,1,x2);
   MathProbabilityDensityGeometric(x2,p_par,false,y2);
//--- 規模を設定する
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- チャートを出力する
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Geometric distribution p=%G",p_par));
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.BackgroundMainSize(16);
//--- Y軸の自動スケーリングを無効にする
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Max(max);
   graphic.XAxis().Min(min);
//--- カーブをプロットする
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 今度は分布密度の理論曲線をプロットする
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- カーブをプロットする
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|   データセットの頻度を計算する                                   |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double
&frequency[],
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 間隔の中心を定義する
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+i*width;
      frequency[i]=0;
     }
//--- 間隔内に入る頻度を記入する
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|   シーケンス生成の値を計算する                                   |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
```

© 2000–2025, MetaQuotes Ltd.

```
   {
//--- 正規化の精度を得るためにシーケンスの絶対範囲を計算する
    double range=MathAbs(maxv-minv);
    int degree=(int)MathRound(MathLog10(range));
//--- 最大値と最小値を指定された精度に正規化する
    maxv=NormalizeDouble(maxv,degree);
    minv=NormalizeDouble(minv,degree);
//--- 指定された精度に基づいてシーケンス生成ステップも設定される
    stepv=NormalizeDouble(MathPow(10,-degree),degree);
    if((maxv-minv)/stepv<10)
       stepv/=10.;
   }
```

© 2000–2025, MetaQuotes Ltd.

MathProbabilityDensityGeometric

確率変数xに対して、パラメータpを用いて幾何分布の確率質量関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityGeometric(
   const double  x,              // 確率変数の値(整数)
   const double  p,              // 分布のパラメータ(1回のテストでのイベント発生の確率)
   const bool    log_mode,       // 値の対数を計算する。log_mode=trueの場合は、確率密度の自然対数が計算される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータpを用いて幾何分布の確率質量関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityGeometric(
   const double  x,              // 確率変数の値(整数)
   const double  p,              // 分布のパラメータ(1回のテストでのイベント発生の確率)
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対して、パラメータpを用いて幾何分布の確率質量関数の値を計算します。エラーの場合は、falseを返します。Rのdgeom()の類似体です。

```
bool  MathProbabilityDensityGeometric(
   const double& x[],            // 確率変数の値を持つ配列
   const double  p,              // 分布のパラメータ(1回のテストでのイベント発生の確率)
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率密度の自然対数が計算される
   double&       result[]        // 確率密度関数の値の配列
   );
```

確率変数の配列x[]に対して、パラメータpを用いて幾何分布の確率質量関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathProbabilityDensityGeometric(
   const double& x[],            // 確率変数の値を持つ配列
   const double  p,              // 分布のパラメータ(1回のテストでのイベント発生の確率)
   double&       result[]        // 確率密度関数の値の配列
   );
```

パラメータ

x

[in] 確率変数の値

x[]

[in] 確率変数の値を持つ配列

p

[in] 分布のパラメータ(1回のテストでのイベント発生の確率)

© 2000-2025, MetaQuotes Ltd.

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率密度の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率密度関数の値の配列

© 2000–2025, MetaQuotes Ltd.

MathCumulativeDistributionGeometric

確率変数xに対して、パラメータpを用いて幾何分布の確率分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionGeometric(
   const double  x,              // 確率変数の値(整数)
   const double  p,              // 分布のパラメータ(1回のテストでのイベント発生の確率)
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータpを用いて幾何分布の確率分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionGeometric(
   const double  x,              // 確率変数の値(整数)
   const double  p,              // 分布のパラメータ(1回のテストでのイベント発生の確率)
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対して、パラメータpを用いて幾何分布の確率分布関数の値を計算します。エラーの場合は、falseを返します。Rのpgeom()の類似体です。

```
bool  MathCumulativeDistributionGeometric(
   const double& x[],            // 確率変数の値を持つ配列
   const double  p,              // 分布のパラメータ(1回のテストでのイベント発生の確率)
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される
   double&       result[]        // 確率関数の値の配列
   );
```

確率変数の配列x[]に対して、パラメータpを用いて幾何分布の確率分布関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathCumulativeDistributionGeometric(
   const double& x[],            // 確率変数の値を持つ配列
   const double  p,              // 分布のパラメータ(1回のテストでのイベント発生の確率)
   double&       result[]        // 確率関数の値の配列
   );
```

パラメータ

x

[in] 確率変数の値

x[]

© 2000-2025, MetaQuotes Ltd.

[in] 確率変数の値を持つ配列

p

[in] 分布のパラメータ（1回のテストでのイベント発生の確率）

tail

[in] 計算のフラグ。tail=trueの場合、確率変数がx 以下である確率を計算する。

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率関数の値の配列

© 2000–2025, MetaQuotes Ltd.

MathQuantileGeometric

この関数は、指定されたprobabilityに対して、パラメータpを用いて幾何法の分布関数の逆数値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileGeometric(
   const double  probability,     // 確率変数の発生確率値
   const double  p,               // 分布のパラメータ(1回のテストでのイベント発生の確率)
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定されたprobabilityに対して、パラメータpを用いて幾何法の分布関数の逆数値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileGeometric(
   const double  probability,     // 確率変数の発生確率値
   const double  p,               // 分布のパラメータ(1回のテストでのイベント発生の確率)
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータpを用いて幾何法の分布関数の逆数値を計算します。エラーの場合は、falseを返します。Rのqgeom()の類似体です。

```
double  MathQuantileGeometric(
   const double& probability[],   // 確率変数の確率値を持つ配列
   const double  p,               // 分布のパラメータ(1回のテストでのイベント発生の確率)
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   double&       result[]         // 分位数の値を持つ配列
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータpを用いて幾何法の分布関数の逆数値を計算します。エラーの場合は、falseを返します。

```
bool  MathQuantileGeometric(
   const double& probability[],   // 確率変数の確率値を持つ配列
   const double  p,               // 分布のパラメータ(1回のテストでのイベント発生の確率)
   double&       result[]         // 分位数の値を持つ配列
   );
```

パラメータ

probability

[in] 確率変数の確率値

probability[]

[in] 確率変数の確率値を持つ配列

p

© 2000-2025, MetaQuotes Ltd.

[in] 分布のパラメータ（1回のテストでのイベント発生の確率）

tail

[in] 計算のフラグ。falseの場合は1.0確率で計算する

log_mode

[in] 計算のフラグ。log_mode=trueの場合、計算はExp(probability)に対して行われる

error_code

[out] エラーコードを取得するための変数

result[]

[out] 分位数の値を持つ配列

© 2000–2025, MetaQuotes Ltd.

MathRandomGeometric

パラメータpを使って、幾何分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、NaNを返します。

```
double  MathRandomGeometric(
   const double  p,              // 分布のパラメータ(1回のテストでのイベント発生の確率)
   int&          error_code      // エラーコードを格納する変数
   );
```

パラメータpを使って、幾何分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、falseを返します。Rのrgeom()の類似体です。

```
bool  MathRandomGeometric(
   const double  p,              // 分布のパラメータ(1回のテストでのイベント発生の確率)
   const int     data_count,     // 必要なデータ量
   double&       result[]        // 擬似乱数変数の値の配列
   );
```

パラメータ

p

[in] 分布のパラメータ(1回のテストでのイベント発生の確率)

error_code

[out] エラーコードを格納する変数

data_count

[out] 必要なデータ量

result[]

[out] 擬似乱数の値を取得するための配列

© 2000-2025, MetaQuotes Ltd.

MathMomentsGeometric

パラメータn、pを使用して、幾何分布の最初の4次のモーメントの理論的数値を計算します。

```
double  MathMomentsGeometric(
   const double  p,              // 分布のパラメータ(1回のテストでのイベント発生の確率)
   double&       mean,           // 平均を表す変数
   double&       variance,       // 分散を表す変数
   double&       skewness,       // 歪みを表す変数
   double&       kurtosis,       // 尖度を表す変数
   int&          error_code      // エラーコードを表す変数
   );
```

パラメータ

p

[in] 分布のパラメータ(1回のテストでのイベント発生の確率)

mean

[out] 平均値を取得するための変数

variance

[out] 分散を取得するための変数

skewness

[out] 歪みを取得するための変数

kurtosis

[out] 尖度を取得するための変数

error_code

[out] エラーコードを取得するための変数

戻り値

モーメントの計算が成功した場合はtrue、そうでない場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

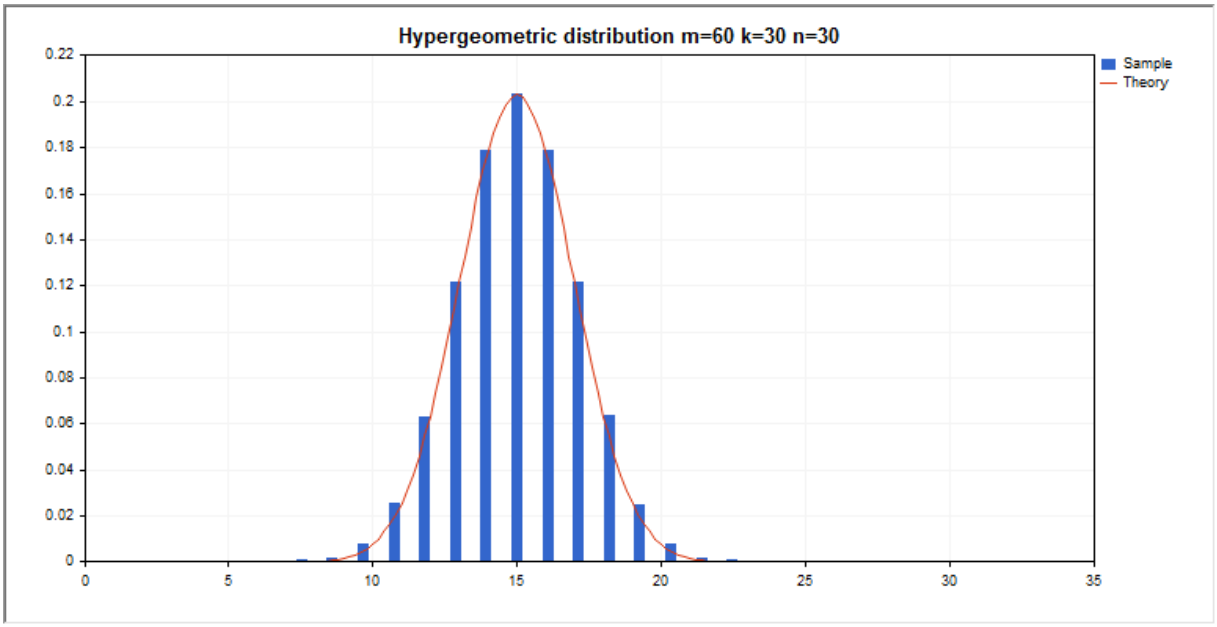
超幾何分布

このセクションには、超幾何分布を使用するための関数が含まれています。密度、確率、分位数を計算し、超幾何法則に従って分布した疑似乱数を生成することができます。超幾何分布は下記の式で定義されています。

$$f_{Hypergeometric}(x|\,m,k,n)=\frac{\binom{k}{x}\binom{m-k}{n-x}}{\binom{m}{n}}$$

ここで

- x — 確率変数の値（整数）
- m — オブジェクトの総数
- k — 所望の特性を有するオブジェクトの数
- n — オブジェクト描画の数



ライブラリは、個々の確率変数の計算に加えて、確率変数の配列を扱う能力も実装しています。

関数	説明
MathProbabilityDensityHypergeometric	超幾何分布の確率密度関数を計算します
MathCumulativeDistributionHypergeometric	超幾何確率分布関数の値を計算します
MathQuantileHypergeometric	指定された確率に対する超幾何逆分布関数の値を計算します
MathRandomHypergeometric	超幾何分布の法則に従って分布した擬似乱数変数/擬似乱数変数の配列を生成します

© 2000–2025, MetaQuotes Ltd.

関数	説明
MathMomentsHypergeometric	超幾何分布の最初の4次のモーメントの理論的数値を計算します

例：

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Hypergeometric.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 入力パラメータ
input double m_par=60;       // オブジェクトの総数
input double k_par=30;       // 所望の特性を有するオブジェクトの数
input double n_par=30;       // オブジェクト描画の数
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 価格チャートを非表示にする
   ChartSetInteger(0,CHART_SHOW,false);
//--- 擬似乱数生成器を初期化する
   MathSrand(GetTickCount());
//--- 確率変数のサンプルを生成する
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // サンプルの値の数
   int ncells=15;          // ヒストグラムの間隔の数
   double x[];             // ヒストグラム間隔の中心
   double y[];             // 間隔内のサンプルからの値の数
   double data[];          // 確率変数値のサンプル
   double max,min;         // サンプルの最大値と最小値
//--- 超幾何分布からサンプルを取得する
   MathRandomHypergeometric(m_par,k_par,n_par,n,data);
//--- ヒストグラムをプロットするためのデータを計算する
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 理論曲線をプロットするシーケンス境界とステップを取得する
   double step;
   GetMaxMinStepValues(max,min,step);
   PrintFormat("max=%G min=%G",max,min);
//--- [min、max]の間隔で理論的に計算されたデータを取得する
   double x2[];
   double y2[];
   MathSequence(0,n_par,1,x2);
   MathProbabilityDensityHypergeometric(x2,m_par,k_par,n_par,false,y2);
//--- 規模を設定する
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
```

© 2000-2025, MetaQuotes Ltd.

```
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- チャートを出力する
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Hypergeometric distribution m=%G k=%G n=%
G",m_par,k_par,n_par));
   graphic.BackgroundMainSize(16);
//--- カーブをプロットする
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 今度は分布密度の理論曲線をプロットする
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory").LinesSmooth(true);
   graphic.CurvePlotAll();
//--- カーブをプロットする
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|   データセットの頻度を計算する                                   |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double
&frequency[],
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 間隔の中心を定義する
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 間隔内に入る頻度を記入する
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
    return (true);
   }
//+------------------------------------------------------------------+
//|   シーケンス生成の値を計算する                                   |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
   {
//--- 正規化の精度を得るためにシーケンスの絶対範囲を計算する
    double range=MathAbs(maxv-minv);
    int degree=(int)MathRound(MathLog10(range));
//--- 最大値と最小値を指定された精度に正規化する
    maxv=NormalizeDouble(maxv,degree);
    minv=NormalizeDouble(minv,degree);
//--- 指定された精度に基づいてシーケンス生成ステップも設定される
    stepv=NormalizeDouble(MathPow(10,-degree),degree);
    if((maxv-minv)/stepv<10)
       stepv/=10.;
   }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityHypergeometric

確率変数xに対して、パラメータm、k、nを用いて超幾何学分布の確率質量関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityHypergeometric(
   const double  x,              // 確率変数の値(整数)
   const double  m,              // オブジェクトの総数(整数)
   const double  k,              // 所望の特性を有するオブジェクトの数(整数)
   const double  n,              // オブジェクト描画の数(整数)
   const bool    log_mode,       // 値の対数を計算する。log_mode=trueの場合は、確率密度の自然対数が計算される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータm、k、nを用いて超幾何学分布の確率質量関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityHypergeometric(
   const double  x,              // 確率変数の値(整数)
   const double  m,              // オブジェクトの総数(整数)
   const double  k,              // 所望の特性を有するオブジェクトの数(整数)
   const double  n,              // オブジェクト描画の数(整数)
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対して、パラメータm、k、nを用いて超幾何学分布の確率質量関数の値を計算します。エラーの場合は、falseを返します。Rのdhyper()の類似体です。

```
bool  MathProbabilityDensityHypergeometric(
   const double& x[],            // 確率変数の値を持つ配列
   const double  m,              // オブジェクトの総数(整数)
   const double  k,              // 所望の特性を有するオブジェクトの数(整数)
   const double  n,              // オブジェクト描画の数(整数)
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率密度の自然対数が計算される
   double&       result[]        // 確率密度関数の値の配列
   );
```

確率変数の配列x[]に対して、パラメータm、k、nを用いて超幾何学分布の確率質量関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathProbabilityDensityHypergeometric(
   const double& x[],            // 確率変数の値を持つ配列
   const double  m,              // オブジェクトの総数(整数)
   const double  k,              // 所望の特性を有するオブジェクトの数(整数)
   const double  n,              // オブジェクト描画の数(整数)
   double&       result[]        // 確率密度関数の値の配列
   );
```

パラメータ

© 2000-2025, MetaQuotes Ltd.

x

[in] 確率変数の値

x[]

[in] 確率変数の値を持つ配列

m

[in] オブジェクトの総数（整数）

k

[in] 所望の特性を有するオブジェクトの数（整数）

n

[in] オブジェクト描画の数（整数）

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率密度の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率密度関数の値の配列

© 2000–2025, MetaQuotes Ltd.

MathCumulativeDistributionHypergeometric

確率変数xに対して、パラメータm、k、nを用いて超幾何学分布の確率分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionHypergeometric(
   const double  x,              // 確率変数の値(整数)
   const double  m,              // オブジェクトの総数(整数)
   const double  k,              // 所望の特性を有するオブジェクトの数(整数)
   const double  n,              // オブジェクト描画の数(整数)
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータm、k、nを用いて超幾何学分布の確率分布関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionHypergeometric(
   const double  x,              // 確率変数の値(整数)
   const double  m,              // オブジェクトの総数(整数)
   const double  k,              // 所望の特性を有するオブジェクトの数(整数)
   const double  n,              // オブジェクト描画の数(整数)
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対して、パラメータm、k、nを用いて超幾何学分布の確率分布関数の値を計算します。エラーの場合は、falseを返します。Rのdhyper()の類似体です。

```
bool  MathCumulativeDistributionHypergeometric(
   const double& x[],            // 確率変数の値を持つ配列
   const double  m,              // オブジェクトの総数(整数)
   const double  k,              // 所望の特性を有するオブジェクトの数(整数)
   const double  n,              // オブジェクト描画の数(整数)
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される
   double&       result[]        // 分布関数の値の配列
   );
```

確率変数の配列x[]に対して、パラメータm、k、nを用いて超幾何学分布の確率分布関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathCumulativeDistributionHypergeometric(
   const double& x[],            // 確率変数の値を持つ配列
   const double  m,              // オブジェクトの総数(整数)
   const double  k,              // 所望の特性を有するオブジェクトの数(整数)
   const double  n,              // オブジェクト描画の数(整数)
```

© 2000-2025, MetaQuotes Ltd.

```
   double&        result[]         // 分布関数の値の配列
   );
```

パラメータ

x
　[in] 確率変数の値

x[]
　[in] 確率変数の値を持つ配列

m
　[in] オブジェクトの総数（整数）

k
　[in] 所望の特性を有するオブジェクトの数（整数）

n
　[in] オブジェクト描画の数（整数）

tail
　[in] 計算のフラグ。true の場合、確率変数がx 以下である確率を計算する。

log_mode
　[in] 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される

error_code
　[out] エラーコードを格納する変数

result[]
　[out] 分布関数の値の配列

© 2000–2025, MetaQuotes Ltd.

MathQuantileHypergeometric

この関数は、指定されたprobabilityに対して、パラメータm、k、nを用いて超幾何学法の分布関数の逆数値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileHypergeometric(
   const double  probability,     // 確率変数の発生確率値
   const double  m,               // オブジェクトの総数(整数)
   const double  k,               // 所望の特性を有するオブジェクトの数(整数)
   const double  n,               // オブジェクト描画の数(整数)
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定されたprobabilityに対して、パラメータm、k、nを用いて超幾何学法の分布関数の逆数値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantileHypergeometric(
   const double  probability,     // 確率変数の発生確率値
   const double  m,               // オブジェクトの総数(整数)
   const double  k,               // 所望の特性を有するオブジェクトの数(整数)
   const double  n,               // オブジェクト描画の数(整数)
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータm、k、nを用いて超幾何学法の分布関数の逆数値を計算します。エラーの場合は、falseを返します。Rのqhyper()の類似体です。

```
double  MathQuantileHypergeometric(
   const double& probability[],   // 確率変数の確率値を持つ配列
   const double  m,               // オブジェクトの総数(整数)
   const double  k,               // 所望の特性を有するオブジェクトの数(整数)
   const double  n,               // オブジェクト描画の数(整数)
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   double&       result[]         // 分位数の値を持つ配列
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータm、k、nを用いて超幾何学法の分布関数の逆数値を計算します。エラーの場合は、falseを返します。

```
bool  MathQuantileHypergeometric(
   const double& probability[],   // 確率変数の確率値を持つ配列
   const double  m,               // オブジェクトの総数(整数)
   const double  k,               // 所望の特性を有するオブジェクトの数(整数)
   const double  n,               // オブジェクト描画の数(整数)
   double&       result[]         // 分位数の値を持つ配列
   );
```

© 2000-2025, MetaQuotes Ltd.

パラメータ

probability

[in] 確率変数の確率値

probability[]

[in] 確率変数の確率値を持つ配列

m

[in] オブジェクトの総数（整数）

k

[in] 所望の特性を有するオブジェクトの数（整数）

n

[in] オブジェクト描画の数（整数）

tail

[in] 計算のフラグ。tail=falseの場合は1.0確率で計算する

log_mode

[in] 計算のフラグ。log_mode=trueの場合、計算はExp(probability)に対して行われる

error_code

[out] エラーコードを取得するための変数

result[]

[out] 分位数の値を持つ配列

© 2000–2025, MetaQuotes Ltd.

MathRandomHypergeometric

パラメータm、n、pを使って、超幾何分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、NaNを返します。

```
double  MathRandomHypergeometric(
   const double  m,              // オブジェクトの総数(整数)
   const double  k,              // 所望の特性を有するオブジェクトの数(整数)
   const double  n,              // オブジェクト描画の数(整数)
   int&          error_code      // エラーコードを格納する変数
   );
```

パラメータm、n、pを使って、超幾何分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、falseを返します。Rのrgeom()の類似体です。

```
bool  MathRandomHypergeometric(
   const double  m,              // オブジェクトの総数(整数)
   const double  k,              // 所望の特性を有するオブジェクトの数(整数)
   const double  n,              // オブジェクト描画の数(整数)
   const int     data_count,     // 必要なデータ量
   double&       result[]        // 擬似乱数変数の値の配列
   );
```

パラメータ

m

[in] オブジェクトの総数(整数)

k

[in] 所望の特性を有するオブジェクトの数(整数)

n

[in] オブジェクト描画の数(整数)

error_code

[out] エラーコードを格納する変数

data_count

[out] 必要なデータ量

result[]

[out] 擬似乱数の値を取得するための配列

© 2000-2025, MetaQuotes Ltd.

MathMomentsHypergeometric

パラメータm、n、kを使用して、超幾何分布の最初の4次のモーメントの理論的数値を計算します。

```
double  MathMomentsHypergeometric(
   const double  m,              // オブジェクトの総数(整数)
   const double  k,              // 所望の特性を有するオブジェクトの数(整数)
   const double  n,              // オブジェクト描画の数(整数)
   double&       mean,           // 平均を表す変数
   double&       variance,       // 分散を表す変数
   double&       skewness,       // 歪みを表す変数
   double&       kurtosis,       // 尖度を表す変数
   int&          error_code      // エラーコードを表す変数
   );
```

パラメータ

m

[in] オブジェクトの総数(整数)

k

[in] 所望の特性を有するオブジェクトの数(整数)

n

[in] オブジェクト描画の数(整数)

mean

[out] 平均値を取得するための変数

variance

[out] 分散を取得するための変数

skewness

[out] 歪みを取得するための変数

kurtosis

[out] 尖度を取得するための変数

error_code

[out] エラーコードを取得するための変数

戻り値

モーメントの計算が成功した場合はtrue、そうでない場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

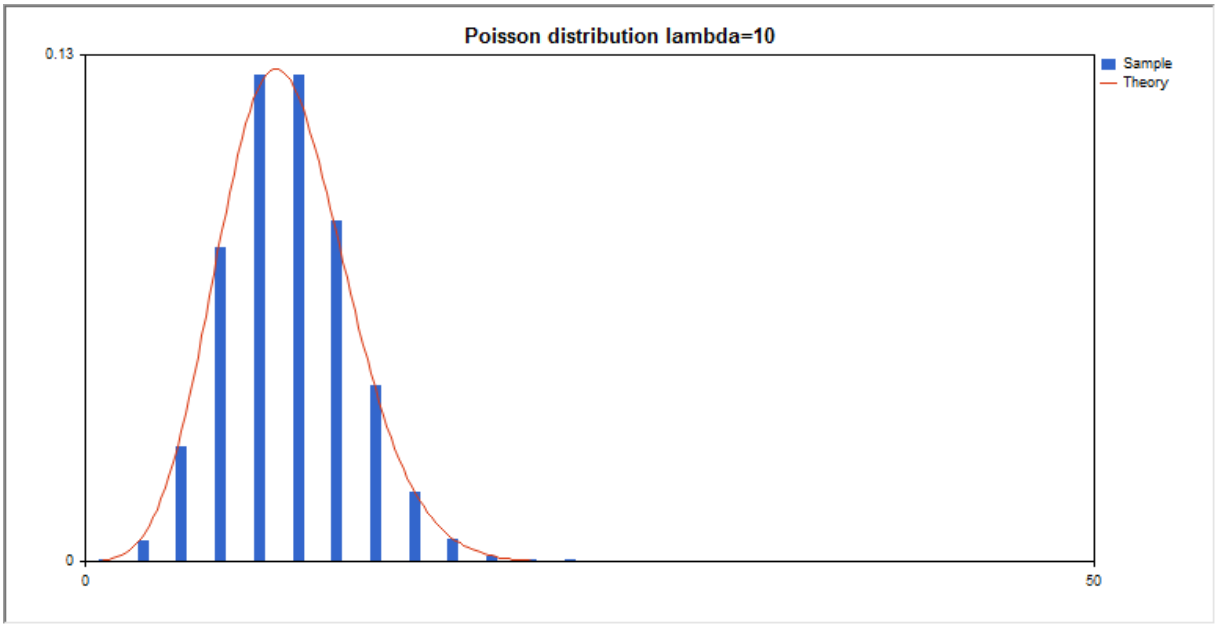
ポアソン分布

このセクションには、ポアソン分布を使用するための関数が含まれています。密度、確率、分位数を計算し、ポアソン法則に従って分布した疑似乱数を生成することができます。ポアソン分布は下記の式で定義されています。

$$f_{Poisson}(x|\lambda) = \frac{\lambda^x}{x!}e^{-\lambda}$$

ここで

- x — 確率変数の値
- λ — 分布のパラメータ(平均)



ライブラリは、個々の確率変数の計算に加えて、確率変数の配列を扱う能力も実装しています。

関数	説明
MathProbabilityDensityPoisson	ポアソン分布の確率密度関数を計算します
MathCumulativeDistributionPoisson	ポアソン確率分布関数の値を計算します
MathQuantilePoisson	指定された確率に対するポアソン逆分布関数の値を計算します
MathRandomPoisson	ポアソン分布の法則に従って分布した擬似乱数変数/擬似乱数変数の配列を生成します
MathMomentsPoisson	ポアソン分布の最初の4次のモーメントの理論的数値を計算します

例:

```
#include <Graphics\Graphic.mqh>
```

© 2000-2025, MetaQuotes Ltd.

```
#include <Math\Stat\Poisson.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 入力パラメータ
input double lambda_par=10;      // 分布のパラメータ(平均)
//+------------------------------------------------------------------+
//| スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 価格チャートを非表示にする
   ChartSetInteger(0,CHART_SHOW,false);
//--- 擬似乱数生成器を初期化する
   MathSrand(GetTickCount());
//--- 確率変数のサンプルを生成する
   long chart=0;
   string name="GraphicNormal";
   int n=100000;        // サンプルの値の数
   int ncells=13;       // ヒストグラムの間隔の数
   double x[];          // ヒストグラム間隔の中心
   double y[];          // 間隔内のサンプルからの値の数
   double data[];       // 確率変数値のサンプル
   double max,min;      // サンプルの最大値と最小値
//--- ポアソン分布からサンプルを取得する
   MathRandomPoisson(lambda_par,n,data);
//--- ヒストグラムをプロットするためのデータを計算する
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 理論曲線をプロットするシーケンス境界とステップを取得する
   double step;
   GetMaxMinStepValues(max,min,step);
   PrintFormat("max=%G min=%G",max,min);
//--- [min、max]の間隔で理論的に計算されたデータを取得する
   double x2[];
   double y2[];
   MathSequence(0,int(MathCeil(max)),1,x2);
   MathProbabilityDensityPoisson(x2,lambda_par,false,y2);
//--- 規模を設定する
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- チャートを出力する
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.BackgroundMain(StringFormat("Poisson distribution lambda=%
G",lambda_par));
   graphic.BackgroundMainSize(16);
//--- Y軸の自動スケーリングを無効にする
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Max(NormalizeDouble(theor_max,2));
   graphic.YAxis().Min(0);
//--- カーブをプロットする
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 今度は分布密度の理論曲線をプロットする
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory").LinesSmooth(true);
   graphic.CurvePlotAll();
//--- カーブをプロットする
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|   データセットの頻度を計算する                                   |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double
&frequency[],
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 間隔の中心を定義する
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 間隔内に入る頻度を記入する
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|   シーケンス生成の値を計算する                                   |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- 正規化の精度を得るためにシーケンスの絶対範囲を計算する
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- 最大値と最小値を指定された精度に正規化する
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 指定された精度に基づいてシーケンス生成ステップも設定される
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityPoisson

確率変数xに対して、パラメータlambdaを用いてポアソン分布の確率質量関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityPoisson(
   const double  x,              // 確率変数の値(整数)
   const double  lambda,         // 分布のパラメータ(平均)
   const bool    log_mode,       // 値の対数を計算する。log_mode=trueの場合は、確率密度の自然対数が計算される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータlambdaを用いてポアソン分布の確率質量関数の値を計算します。エラーの場合は、NaNを返します。

```
double  MathProbabilityDensityPoisson(
   const double  x,              // 確率変数の値(整数)
   const double  lambda,         // 分布のパラメータ(平均)
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対して、パラメータlambdaを用いてポアソン分布の確率質量関数の値を計算します。エラーの場合は、falseを返します。Rのdhyper()の類似体です。

```
bool  MathProbabilityDensityPoisson(
   const double& x[],            // 確率変数の値を持つ配列
   const double  lambda,         // 分布のパラメータ(平均)
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率密度の自然対数が計算される
   double&       result[]        // 確率密度関数の値の配列
   );
```

<t0>確率変数の配列x[]に対して、パラメータlambdaを用いてポアソン分布の確率質量関数の値を計算します。エラーの場合は、falseを返します。

```
bool  MathProbabilityDensityPoisson(
   const double& x[],            // 確率変数の値を持つ配列
   const double  lambda,         // 分布のパラメータ(平均)
   double&       result[]        // 確率密度関数の値の配列
   );
```

パラメータ

x

[in] 確率変数の値

x[]

[in] 確率変数の値を持つ配列

lambda

[in] 分布のパラメータ(平均)

© 2000-2025, MetaQuotes Ltd.

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合、確率密度の自然対数を計算する。

error_code

[out] エラーコードを格納する変数

result[]

[out] 確率密度関数の値の配列

© 2000–2025, MetaQuotes Ltd.

MathCumulativeDistributionPoisson

確率変数xに対して、パラメータlambdaを用いてポアソン分布の確率分布関数を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionPoisson(
   const double  x,              // 確率変数の値(整数)
   const double  lambda,         // 分布のパラメータ(平均)
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数xに対して、パラメータlambdaを用いてポアソン分布の確率分布関数を計算します。エラーの場合は、NaNを返します。

```
double  MathCumulativeDistributionPoisson(
   const double  x,              // 確率変数の値(整数)
   const double  lambda,         // 分布のパラメータ(平均)
   int&          error_code      // エラーコードを格納する変数
   );
```

確率変数の配列x[]に対して、パラメータlambdaを用いてポアソン分布の確率分布関数を計算します。エラーの場合は、falseを返します。Rのdhyper()の類似体です。

```
bool  MathCumulativeDistributionPoisson(
   const double& x[],            // 確率変数の値を持つ配列
   const double  lambda,         // 分布のパラメータ(平均)
   const bool    tail,           // 計算のフラグ。true の場合、確率変数がx 以下である確率が計算される
   const bool    log_mode,       // 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される
   double&       result[]        // 分布関数の値の配列
   );
```

<t0>確率変数の配列x[]に対して、パラメータlambdaを用いてポアソン分布の確率分布関数を計算します。エラーの場合は、falseを返します。

```
bool  MathCumulativeDistributionPoisson(
   const double& x[],            // 確率変数の値を持つ配列
   const double  lambda,         // 分布のパラメータ(平均)
   double&       result[]        // 分布関数の値の配列
   );
```

パラメータ

x

[in] 確率変数の値

x[]

© 2000-2025, MetaQuotes Ltd.

[in] 確率変数の値を持つ配列

lambda

[in] 分布のパラメータ(平均)

tail

[in] 計算のフラグ。true の場合、確率変数がx 以下である確率を計算する。

log_mode

[in] 値の対数を計算するためのフラグ。log_mode=trueの場合は、確率の自然対数が計算される

error_code

[out] エラーコードを格納する変数

result[]

[out] 分布関数の値の配列

© 2000–2025, MetaQuotes Ltd.

MathQuantilePoisson

この関数は、指定されたprobabilityに対して、パラメータlambdaを用いてポアソン分布関数の逆数値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantilePoisson(
   const double  probability,     // 確率変数の発生確率値
   const double  lambda,          // 分布のパラメータ(平均)
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定されたprobabilityに対して、パラメータlambdaを用いてポアソン分布関数の逆数値を計算します。エラーの場合は、NaNを返します。

```
double  MathQuantilePoisson(
   const double  probability,     // 確率変数の発生確率値
   const double  lambda,          // 分布のパラメータ(平均)
   int&          error_code       // エラーコードを格納する変数
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータlambdaを用いてポアソン分布関数の逆数値を計算します。エラーの場合は、falseを返します。Rのqhyper()の類似体です。

```
double  MathQuantilePoisson(
   const double& probability[],   // 確率変数の確率値を持つ配列
   const double  lambda,          // 分布のパラメータ(平均)
   const bool    tail,            // 計算のフラグ。false の場合は1.0確率で計算する
   const bool    log_mode,        // 計算のフラグ。log_mode=true の場合、計算は
Exp(probability)に対して行われる
   double&       result[]         // 分位数の値を持つ配列
   );
```

この関数は、指定された確率値の配列probability[]に対して、パラメータlambdaを用いてポアソン分布関数の逆数値を計算します。エラーの場合は、falseを返します。

```
bool  MathQuantilePoisson(
   const double& probability[],   // 確率変数の確率値を持つ配列
   const double  lambda,          // 分布のパラメータ(平均)
   double&       result[]         // 分位数の値を持つ配列
   );
```

パラメータ

probability

[in] 確率変数の確率値

probability[]

[in] 確率変数の確率値を持つ配列

lambda

© 2000-2025, MetaQuotes Ltd.

[in] 分布のパラメータ（平均）

tail

[in] 計算のフラグ。tail=falseの場合は1.0確率で計算する

log_mode

[in] 計算のフラグ。log_mode=trueの場合、計算はExp(probability)に対して行われる

error_code

[out] エラーコードを取得するための変数

result[]

[out] 分位数の値を持つ配列

© 2000-2025, MetaQuotes Ltd.

MathRandomPoisson

パラメータlambdaを使って、ポアソン分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、NaNを返します。

```
double  MathRandomPoisson(
   const double  lambda,          // 分布のパラメータ(平均)
   int&          error_code       // エラーコードを格納する変数
   );
```

パラメータlambdaを使って、ポアソン分布の法則に従って分布した擬似乱数変数を生成します。エラーの場合は、falseを返します。Rのrgeom()の類似体です。

```
bool  MathRandomPoisson(
   const double  lambda,          // 分布のパラメータ(平均)
   const int     data_count,      // 必要なデータ量
   double&       result[]         // 擬似乱数変数の値の配列
   );
```

パラメータ

lambda

[in] 分布のパラメータ(平均)

error_code

[out] エラーコードを格納する変数

data_count

[out] 必要なデータ量

result[]

[out] 擬似乱数の値を取得するための配列

© 2000-2025, MetaQuotes Ltd.

MathMomentsPoisson

パラメータlambdaを使用して、ポアソン分布の最初の4次のモーメントの理論的数値を計算します。

```
double  MathMomentsPoisson(
   const double  lambda,          // 分布のパラメータ(平均)
   double&       mean,            // 平均を表す変数
   double&       variance,        // 分散を表す変数
   double&       skewness,        // 歪みを表す変数
   double&       kurtosis,        // 尖度を表す変数
   int&          error_code       // エラーコードを表す変数
   );
```

パラメータ

lambda

[in] 分布のパラメータ(平均)

mean

[out] 平均値を取得するための変数

variance

[out] 分散を取得するための変数

skewness

[out] 歪みを取得するための変数

kurtosis

[out] 尖度を取得するための変数

error_code

[out] エラーコードを取得するための変数

戻り値

モーメントの計算が成功した場合はtrue、そうでない場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

サブ関数

ガンマ関数、ベータ関数、階乗、指数関数、異なる基底を持つ対数、平方根などの基本的な数学演算を実行する関数のグループです。

これらは、個々の数値（実数と整数）と値の配列（別々の配列または元の配列への結果の出力）を処理する機能を備えています。

関数	説明
MathRandomNonZero	0.0～1.0の浮動小数点数を持つ乱数を返します。
MathMoments	配列要素の最初の四次モーメント（平均、分散、歪度、尖度）を計算します。
MathPowInt	数を指定された整数乗まで引き上げます。
MathFactorial	指定された整数の階乗を計算します。
MathTrunc	指定された数値や配列要素の整数部分を計算します。
MathRound	数値または数値の配列を指定された小数点以下桁数に丸めます。
MathArctan2	接線が[-pi、pi]の範囲にある指定された2つの数値の比率に等しい角度を計算します。
MathGamma	ガンマ関数の値を計算します。
MathGammaLog	ガンマ関数の対数を計算します。
MathBeta	ベータ関数の値を計算します。
MathBetaLog	ベータ関数の対数を計算します。
MathBetaIncomplete	不完全ベータ関数の値を計算します。
MathGammaIncomplete	不完全ガンマ関数の値を計算します。
MathBinomialCoefficient	二項係数を計算します。
MathBinomialCoefficientLog	二項係数の対数を計算します。
MathHypergeometric2F2	超幾何関数の値を計算します。
MathSequence	最初の要素、最後の要素、シーケンスのステップの値に基づいてシーケンスを生成します。
MathSequenceByCount	最初の要素、最後の要素、シーケンスの要素数に基づいてシーケンスを生成します。
MathReplicate	繰り返し値のシーケンスを生成します。
MathReverse	要素を逆順にした値の配列を生成します。
MathIdentical	2つの値の配列を比較し、すべての要素が一致する場合はtrueを返します。

© 2000-2025, MetaQuotes Ltd.

関数	説明
MathUnique	一意の値のみを持つ配列を生成します。
MathQuickSortAscending	昇順で並べ替える関数です。
MathQuickSortDescending	降順で並べ替える関数です。
MathQuickSort	並べ替え関数です。
MathOrder	並べ替え後の配列要素の順番に従った順列を持つ配列を生成します。
MathBitwiseNot	配列要素のビット単位NOT演算の結果を計算します。
MathBitwiseAnd	配列要素のビット単位AND演算の結果を計算します。
MathBitwiseOr	配列要素のビット単位AND演算の結果を計算します。
MathBitwiseXor	配列要素のビット単位XOR演算の結果を計算します。
MathBitwiseShiftL	配列要素のビット単位SHL演算の結果を計算します。
MathBitwiseShiftR	配列要素のビット単位SHR演算の結果を計算します。
MathCumulativeSum	累積合計で配列を生成します。
MathCumulativeProduct	累積積で配列を生成します。
MathCumulativeMin	累積最小値で配列を生成します。
MathCumulativeMax	累積最大値で配列を生成します。
MathSin	配列要素のsin(x)関数の値を計算します。
MathCos	配列要素のcos(x)関数の値を計算します。
MathTan	配列要素のtan(x)関数の値を計算します。
MathArcsin	配列要素のarcsin(x)関数の値を計算します。
MathArccos	配列要素のarccos(x)関数の値を計算します。
MathArctan	配列要素のarctan(x)関数の値を計算します。
MathSinPi	配列要素のsin(pi*x) 関数の値を計算します。
MathCosPi	配列要素のcos(pi*x) 関数の値を計算します。
MathTanPi	配列要素のtan(pi*x) 関数の値を計算します。
MathAbs	配列要素の絶対値を計算します。

© 2000–2025, MetaQuotes Ltd.

関数	説明
MathCeil	配列要素が最も近い値に切り上げられた場合の整数値を返します。
MathFloor	配列要素が最も近い値に切り下げられた場合の整数値を返します。
MathSqrt	配列要素の平方根を計算します。
MathExp	配列要素のexp(x)関数の値を計算します。
MathPow	配列要素のpow(x, power)関数の値を計算します。
MathLog	配列要素のlog(x)関数の値を計算します。
MathLog2	配列要素の2を底とする対数を計算します。
MathLog10	配列要素の10を底とする対数を計算します。
MathDifference	要素の差がy[i]=x[i+lag]−x[i]の配列を生成します。
MathSample	配列要素からランダムサンプルを生成します。
MathTukeySummary	配列要素のTukeyの五数要約を計算します。
MathRange	配列要素の最小値と最大値を計算します。
MathMin	すべての配列要素の最小値を返します。
MathMax	すべての配列要素の最大値を返します。
MathSum	配列要素の和を返します。
MathProduct	配列要素の積を返します。
MathStandardDeviation	配列要素の標準偏差を計算します。
MathAverageDeviation	配列要素の平均絶対偏差を計算します。
MathMedian	配列要素の中央値を計算します。
MathMean	配列要素の平均値を計算します。
MathVariance	配列要素の分散を計算します。
MathSkewness	配列要素の歪度を計算します。
MathKurtosis	配列要素の尖度を計算します。
MathLog1p	配列要素の log(1+x)関数の値を計算します。
MathExpm1	配列要素のexp(x)−1関数の値を計算します。
MathSinh	配列要素のsinh(x)関数の値を計算します。
MathCosh	配列要素のcosh(x)関数の値を計算します。
MathTanh	配列要素のtanh(x)関数の値を計算します。

© 2000−2025, MetaQuotes Ltd.

関数	説明
MathArcsinh	配列要素のarcsinh(x)関数の値を計算します。
MathArccosh	配列要素のarccosh(x)関数の値を計算します。
MathArctanh	配列要素のarctanh(x)関数の値を計算します。
MathSignif	値を指定された仮数の桁数を持つ値に丸めます。
MathRank	配列要素のランクを計算します。
MathCorrelationPearson	ピアソンの相関係数を計算します。
MathCorrelationSpearman	スピアマンの相関係数を計算します。
MathCorrelationKendall	ケンドールの相関係数を計算します。
MathQuantile	指定された確率に対応するサンプルの分位数を計算します。
MathProbabilityDensityEmpirical	乱数に対する経験的確率密度関数を計算します。
MathCumulativeDistributionEmpirical	乱数に対する経験的な累積分布関数を計算します。

© 2000–2025, MetaQuotes Ltd.

MathRandomNonZero

0.0～1.0の浮動小数点数を持つ乱数を返します。

```
double  MathRandomNonZero()
```

戻り値

0.0～1.0の浮動小数点数を持つ乱数

© 2000-2025, MetaQuotes Ltd.

MathMoments

配列要素の最初の四次モーメント（平均、分散、歪度、尖度）を計算します。

```
bool  MathMoments(
   const double&  array[],            // 値の配列
   double&        mean,               // 平均を示す変数
   double&        variance,           // 分散を示す変数
   double&        skewness,           // 歪度を示す変数
   double&        kurtosis,           // 尖度を示す変数
   const int      start=0,            // 初期のインデックス
   const int      count=WHOLE_ARRAY   // 要素数
   )
```

パラメータ

array[]

[in] 値の配列

mean

[out] 平均値（一次モーメント）の変数

variance

[out] 分散（二次モーメント）の変数

skewness

[out] 歪度（三次モーメント）の変数

kurtosis

[out] 尖度（四次モーメント）の変数

start=0

[in] 計算の初期インデックス

count=WHOLE_ARRAY

[in] 計算される要素数

戻り値

モーメントが正常に計算された場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

MathPowInt

数を指定された整数乗まで引き上げます。

```
double  MathPowInt(
   const double  x,       // 数値
   const int     power    // 累乗
   )
```

パラメータ

x

[in] 累乗される倍精度浮動小数点数

power

[in] 累乗を示す整数

戻り値

xの(指定された)乗

© 2000-2025, MetaQuotes Ltd.

MathFactorial

指定された整数の階乗を計算します。

```
double  MathFactorial(
   const int  n   // 数値
   )
```

パラメータ

n

[in] その階乗が計算される整数

戻り値

数の階乗

© 2000–2025, MetaQuotes Ltd.

MathTrunc

指定された数値や配列要素の整数部分を計算します。

倍精度浮動小数点数を扱うためのバージョン:

```
double  MathTrunc(
   const double   x          // 数値
   )
```

戻り値

指定された数値の整数部分

倍精度浮動小数点数配列を扱うためのバージョン。結果は新しい配列に出力されます。

```
bool  MathTrunc(
   const double&  array[],   // 値の配列
   double&        result[]   // 結果の配列
   )
```

戻り値

成功の場合は true、それ以外の場合は false。

倍精度浮動小数点数配列を扱うためのバージョン。結果は元の配列に出力されます。

```
bool  MathTrunc(
   double&        array[]    // 値の配列
   )
```

戻り値

成功の場合は true、それ以外の場合は false。

パラメータ

x

[in] その整数部分が取得される倍精度浮動小数点数

array[]

[in] その整数部分が取得される倍精度浮動小数点数の配列

array[]

[out] 出力値の配列

result[]

[out] 出力値の配列

© 2000–2025, MetaQuotes Ltd.

MathRound

倍精度浮動小数点数またはそのような数値の配列を指定された小数点以下桁数に丸めます。

倍精度浮動小数点数を指定された小数点以下桁数に丸めるバージョン:

```
double  MathRound(
   const double   x,          // 数値
   const int      digits      // 小数点以下桁数
   )
```

戻り値

桁数に等しい小数点以下の桁数がdigitsに等しい、xパラメータに最も近い数値

倍精度浮動小数点数の配列を指定された小数点以下桁数に丸めるバージョン。結果は新しい配列に出力されます。

```
bool  MathRound(
   const double&  array[],    // 値の配列
   int            digits,     // 小数点以下桁数
   double&        result[]    // 結果の配列
   )
```

戻り値

成功の場合はtrue、それ以外の場合はfalse。

倍精度浮動小数点数の配列を指定された小数点以下桁数に丸めるバージョン。結果は新しい配列に出力されます。

```
bool  MathRound(
   double&        array[],    // 値の配列
   int            digits      // 小数点以下桁数
   )
```

戻り値

成功の場合はtrue、それ以外の場合はfalse。

パラメータ

x

[in] 丸められる倍精度浮動小数点数

digits

[in] 戻り値の小数点以下桁数

array[]

[in] 丸められる倍精度浮動小数点数の配列

array[]

[out] 出力値の配列

© 2000–2025, MetaQuotes Ltd.

result[]

[out] 出力値の配列

© 2000–2025, MetaQuotes Ltd.

MathArctan2

2つの引数（x、y）の商のアークタンジェントを返します。

指定された2つの数値（x、y）の比率を扱うためのバージョン：

```
double  MathArctan2(
   const double       y,          // Y座標
   const double       x           // X座標
   )
```

戻り値

π≤θ≤π及び tan (θ) = yまたはx（ここで (x, y) はデカルト座標系の点）となるような（x、y）ラジアンで測定された角度 θ。

x及びy配列からの要素のペアの比率を扱うためのバージョン：

```
bool  MathArctan2(
   const double&      x[],        // x値の配列
   const double&      y[],        // y値の配列
   double&            result[]    // 結果の配列
   )
```

戻り値

成功の場合は true、それ以外の場合は false。

パラメータ

y

[in] 点のY座標

x

[in] 点のX座標

x[]

[in] 点のX座標の配列

y[]

[in] 点のY座標の配列

result[]

[out] 結果を出力する配列

注意事項

以下にご注意ください。

- 第1象限の（x, y）では、戻り値は0 < θ< π/2になります。
- 第2象限の（x, y）では、戻り値はπ/2 < θ≤ πになります。
- 第3象限の（x, y）では、戻り値は −π< θ<−π/2になります。
- 第4象限の（x, y）では、戻り値は−π/2 < θ< 0になります。

© 2000-2025, MetaQuotes Ltd.

これらの象限以外の点の戻り値は以下のとおりです。

- yが0でxが正の場合はθ= 0。
- yが0でxが負の場合はθ=π。
- yが正でxが0の場合はθ=π/ 2。
- yが負でxが0の場合はθ= −π/2。
- yが0でxが0の場合はθ=−π/ 2。

xまたはyパラメータの値がNaNの場合、またはPositiveInfinityまたはNegativeInfinityの値と等しい場合、このメソッドはNaN値を返します。

© 2000−2025, MetaQuotes Ltd.

MathGamma

実数引数xのガンマ関数の値を計算します。

```
double  MathGamma(
   const double  x        // 関数の引数
   )
```

パラメータ

x

[in] 関数の実数引数

戻り値

ガンマ関数の値

© 2000–2025, MetaQuotes Ltd.

MathGammaLog

実数引数xのガンマ関数の対数を計算します。

```
double  MathGammaLog(
   const double  x      // 関数の引数
   )
```

パラメータ

x

[in] 関数の実数引数

戻り値

関数の対数

© 2000–2025, MetaQuotes Ltd.

MathBeta

実数引数aとbのベータ関数の値を計算します。

```
double  MathBeta(
   const double  a,       // 関数の第1引数
   const double  b        // 関数の第2引数
   )
```

パラメータ

a

[in] 関数のa引数

b

[in] 関数のb引数

戻り値

関数の値

© 2000-2025, MetaQuotes Ltd.

MathBetaLog

実数引数aとbのベータ関数の対数値を計算します。

```
double  MathBetaLog(
   const double  a,       // 関数の第1引数
   const double  b        // 関数の第2引数
   )
```

パラメータ

a

[in] 関数のa引数

b

[in] 関数のb引数

戻り値

関数の対数

© 2000-2025, MetaQuotes Ltd.

MathBetaIncomplete

不完全ベータ関数の値を計算します。

```
double  MathBetaIncomplete(
   const double  x,       // 関数の引数
   const double  p,       // 関数の第1パラメータ
   const double  q        // 関数の第2パラメータ
   )
```

パラメータ

x

[in] 関数の引数

p

[in] ベータ関数の第1パラメータ(>0.0)

q

[in] ベータ関数の第2パラメータ(>0.0)

戻り値

関数の値

© 2000-2025, MetaQuotes Ltd.

MathGammaIncomplete

不完全ガンマ関数の値を計算します。

```
double  MathGammaIncomplete(
   double  x,           // 関数の引数
   double  alpha        // 関数のパラメータ
   )
```

パラメータ

x

[in] 関数の引数

alpha

[in] 不完全ガンマ関数のパラメータ

戻り値

関数の値

© 2000-2025, MetaQuotes Ltd.

MathBinomialCoefficient

二項係数C(n,k)=n!/(k!*(n-k)!)を計算します。

```
long  MathBinomialCoefficient(
   const int  n,       // 要素の総数
   const int  k        // 組み合わせ要素数
   )
```

パラメータ

n

[in] 要素数

k

[in] 各組み合わせの要素数

戻り値

NからKを選択する場合の組み合わせの数

© 2000-2025, MetaQuotes Ltd.

MathBinomialCoefficientLog

二項係数の対数Log(C(n,k))=Log(n!/(k!*(n-k)!))を計算します。

整数引数のバージョン:

```
double  MathBinomialCoefficientLog(
   const int      n,       // 要素の総数
   const int      k        // 組み合わせ要素数
   )
```

実数引数のバージョン:

```
double  MathBinomialCoefficientLog(
   const double   n,       // 要素の総数
   const double   k        // 組み合わせ要素数
   )
```

パラメータ

n

[in] 要素数

k

[in] 各組み合わせの要素数

戻り値

C(n,k)の対数

© 2000-2025, MetaQuotes Ltd.

MathHypergeometric2F2

Taylorの方法を使用してHypergeometric_2F2(a, b, c, d, z)関数の値を計算します。

```
double  MathHypergeometric2F2(
   const double  a,       // 関数の第1パラメータ
   const double  b,       // 関数の第2パラメータ
   const double  c,       //  関数の第3パラメータ
   const double  d,       // 関数の第4パラメータ
   const double  z        // 関数の第5パラメータ
   )
```

パラメータ

a

[in] 関数の第1パラメータ

b

[in] 関数の第2パラメータ

c

[in] 関数の第3パラメータ

d

[in] 関数の第4パラメータ

z

[in] 関数の第5パラメータ

戻り値

関数の値

© 2000-2025, MetaQuotes Ltd.

MathSequence

最初の要素、最後の要素、シーケンスのステップの値に基づいてシーケンスを生成します。

実数値を扱うためのバージョン:

```
bool  MathSequence(
   const double  from,        // 初期値
   const double  to,          // 最終値
   const double  step,        // ステップ
   double&       result[]     // 結果の配列
   )
```

整数値を扱うためのバージョン:

```
bool  MathSequence(
   const int     from,        // 初期値
   const int     to,          // 最終値
   const int     step,        // ステップ
   int&          result[]     // 結果の配列
   )
```

パラメータ

from

[in] シーケンスの最初の値

to

[in] シーケンスの最後の値

step

[in] シーケンスのステップ

result[]

[out] シーケンスを出力するための配列

戻り値

成功の場合は true、それ以外の場合は false。

© 2000-2025, MetaQuotes Ltd.

MathSequenceByCount

最初の要素、最後の要素、シーケンスの要素数に基づいてシーケンスを生成します。

実数値を扱うためのバージョン:

```
bool  MathSequenceByCount(
   const double  from,        // 初期値
   const double  to,          // 最終値
   const int     count,       // カウント
   double&       result[]     // 結果の配列
   )
```

整数値を扱うためのバージョン:

```
bool  MathSequenceByCount(
   const int     from,        // 初期値
   const int     to,          // 最終値
   const int     count,       // カウント
   int&          result[]     // 結果の配列
   )
```

パラメータ

from

[in] シーケンスの最初の値

to

[in] シーケンスの最後の値

count

[in] シーケンス内の要素数

result[]

[out] シーケンスを出力するための配列

戻り値

成功の場合は true、それ以外の場合は false。

© 2000-2025, MetaQuotes Ltd.

MathReplicate

繰り返し値のシーケンスを生成します。

実数値を扱うためのバージョン:

```
bool  MathReplicate(
   const double&  array[],    // 値の配列
   const int      count,      // 繰り返し値
   double&        result[]    // 結果の配列
   )
```

整数値を扱うためのバージョン:

```
bool  MathReplicate(
   const int&     array[],    // 値の配列
   const int      count,      // 繰り返し値
   int&           result[]    // 結果の配列
   )
```

パラメータ

array[]

[in] シーケンスを生成するための配列

count

[in] シーケンス内の配列の繰り返し数

result[]

[out] シーケンスを出力するための配列

戻り値

成功の場合は true、それ以外の場合は false。

© 2000–2025, MetaQuotes Ltd.

MathReverse

要素を逆順にした値の配列を生成します。

実数値を扱って新しい配列に結果を出力するバージョン:

```
bool  MathReverse(
   const double&  array[],      // 値の配列
   double&        result[]      // 結果の配列
   )
```

整数値を扱って新しい配列に結果を出力するバージョン:

```
bool  MathReverse(
   const int&     array[],      // 値の配列
   int&           result[]      // 結果の配列
   )
```

実数値を扱って元の配列に結果を出力するバージョン:

```
bool  MathReverse(
   double&        array[]       // 値の配列
   )
```

整数値を扱って元の配列に結果を出力するバージョン:

```
bool  MathReverse(
   int&           array[]       // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

array[]

[out] 値の逆順を持つ出力配列

result[]

[out] 値の逆順を持つ出力配列

戻り値

成功の場合は true、それ以外の場合は false。

© 2000–2025, MetaQuotes Ltd.

MathIdentical

2つの値の配列を比較し、すべての要素が一致する場合はtrueを返します。

実数配列を扱うためのバージョン:

```
bool  MathIdentical(
   const double&  array1[],      // 1番目の値の配列
   const double&  array2[]       // 2番目の値の配列/t5>
   )
```

整数配列を扱うためのバージョン:

```
bool  MathIdentical(
   const int&     array1[],      // 1番目の値の配列
   const int&     array2[]       // 2番目の値の配列/t5>
   )
```

パラメータ

array1[]

[in] 比較される1番目の配列

array2[]

[in] 比較される2番目の配列<分

戻り値

配列が同一の場合はtrue、それ以外の場合はfalse

© 2000-2025, MetaQuotes Ltd.

MathUnique

一意の値のみを持つ配列を生成します。

実数値を扱うためのバージョン:

```
bool  MathUnique(
   const double&   array[],   // 値の配列
   double&         result[]   // 結果の配列
   )
```

整数値を扱うためのバージョン:

```
bool  MathUnique(
   const int&     array[],      // 値の配列
   int&           result[]      // 結果の配列
   )
```

パラメータ

array[]

[in] ソース配列

result[]

[out] 一意の値を出力するための配列

戻り値

成功の場合は true、それ以外の場合は false。

© 2000–2025, MetaQuotes Ltd.

MathQuickSortAscending

QuickSortアルゴリズムを使用してarray[]とindices[]配列を同時に昇順に並び替える関数です。

```
void  MathQuickSortAscending(
   double&  array[],      // 値の配列
   int&     indices[],    // インデックスの配列
   int      first,        // 初期値
   int      last          // 最終値
   )
```

パラメータ

array[]

[in][out] 並び替えられる配列

indices[]

[in][out] 元の配列のインデックスを格納する配列

first

[in] 並び替えを始めるインデックス

last

[in] 並び替えを終えるインデックス

© 2000-2025, MetaQuotes Ltd.

MathQuickSortDescending

QuickSortアルゴリズムを使用してarray[]とindices[]配列を同時に降順に並び替える関数です。

```
void  MathQuickSortDescending(
   double&  array[],      // 値の配列
   int&     indices[],    // インデックスの配列
   int      first,        // 初期値
   int      last          // 最終値
   )
```

パラメータ

array[]

[in][out] 並び替えられる配列

indices[]

[in][out] 元の配列のインデックスを格納する配列

first

[in] 並び替えを始めるインデックス

last

[in] 並び替えを終えるインデックス

© 2000-2025, MetaQuotes Ltd.

MathQuickSort

QuickSortアルゴリズムを使用してarray[]とindices[]配列を同時に並び替える関数です。

```
void  MathQuickSort(
   double&  array[],      // 値の配列
   int&     indices[],    // インデックスの配列
   int      first,        // 初期値
   int      last,         // 最終値
   int      mode          // 方向
   )
```

パラメータ

array[]

[in][out] 並び替えられる配列

indices[]

[in][out] 元の配列のインデックスを格納する配列

first

[in] 並び替えを始めるインデックス

last

[in] 並び替えを終えるインデックス

mode

[in] 並び替えの方向（>0は昇順、それ以外の場合は降順）

© 2000-2025, MetaQuotes Ltd.

MathOrder

並べ替え後の配列要素の順番に従った順列を持つ整数配列を生成します。

実数配列を扱うためのバージョン:

```
bool  MathOrder(
   const double&  array[],   // 値の配列
   int&           result[]   // 結果の配列
   )
```

整数配列を扱うためのバージョン:

```
bool  MathOrder(
   const int&     array[],   // 値の配列
   int&           result[]   // 結果の配列
   )
```

パラメータ

array[]

[in] 値の配列

result[]

[out]並べ替えられたインデックスを出力する配列

戻り値

成功の場合は true、それ以外の場合は false。

© 2000–2025, MetaQuotes Ltd.

MathBitwiseNot

配列要素のビット単位NOT演算の結果を計算します。

新しい配列に結果を出力するバージョン:

```
bool  MathBitwiseNot(
   const int&  array[],   // 値の配列
   int&        result[]   // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathBitwiseNot(
   int&        array[]    // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

array[]

[out] 出力値の配列

result[]

[out] 出力値の配列

戻り値

成功の場合はtrue、それ以外の場合はfalse。

© 2000–2025, MetaQuotes Ltd.

MathBitwiseAnd

指定された配列要素のビット単位AND演算の結果を計算します。

```
bool  MathBitwiseAnd(
   const int&  array1[],  // 1番目の値の配列
   const int&  array2[],  // 2番目の値の配列/t5>
   int&        result[]   // 結果の配列
   )
```

パラメータ

array1[]

[in] 1番目の値の配列

array2[]

[in] 2番目の値の配列

result[]

[out] 結果を出力する配列.

戻り値

成功の場合はtrue、それ以外の場合はfalse。

© 2000-2025, MetaQuotes Ltd.

MathBitwiseOr

指定された配列要素のビット単位OR演算の結果を計算します。

```
bool  MathBitwiseOr(
   const int&  array1[],  // 1番目の値の配列
   const int&  array2[],  // 2番目の値の配列/t5>
   int&        result[]   // 結果の配列
   )
```

パラメータ

array1[]

[in] 1番目の値の配列

array2[]

[in] 2番目の値の配列

result[]

[out] 結果を出力する配列.

戻り値

成功の場合はtrue、それ以外の場合はfalse。

© 2000–2025, MetaQuotes Ltd.

MathBitwiseXor

指定された配列要素のビット単位XOR演算の結果を計算します。

```
bool  MathBitwiseXor(
   const int&  array1[],  // 1番目の値の配列
   const int&  array2[],  // 2番目の値の配列/t5>
   int&        result[]   // 結果の配列
   )
```

パラメータ

array1[]

[in] 1番目の値の配列

array2[]

[in] 2番目の値の配列

result[]

[out] 結果を出力する配列.

戻り値

成功の場合はtrue、それ以外の場合はfalse。

© 2000-2025, MetaQuotes Ltd.

MathBitwiseShiftL

配列要素のビット単位SHL（ビットシフト左）演算の結果を計算します。

新しい配列に結果を出力するバージョン:

```
bool  MathBitwiseShiftL(
   const int&  array[],    // 値の配列
   const int   n,          // シフト値
   int&        result[]    // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathBitwiseShiftL(
   int&        array[],    // 値の配列
   const int   n           // シフト値
   )
```

パラメータ

array[]
　[in] 値の配列

n
　[in]シフトするビット数

array[]
　[out] 出力値の配列

result[]
　[out] 出力値の配列

戻り値

成功の場合はtrue、それ以外の場合はfalse。

© 2000–2025, MetaQuotes Ltd.

MathBitwiseShiftR

配列要素のビット単位SHR(ビットシフト右)演算の結果を計算します。

新しい配列に結果を出力するバージョン:

```
bool  MathBitwiseShiftR(
   const int&  array[],    // 値の配列
   const int   n,          // シフト値
   int&        result[]    // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathBitwiseShiftR(
   int&        array[],    // 値の配列
   const int   n           // シフト値
   )
```

パラメータ

array[]
  [in] 値の配列

n
  [in]シフトするビット数

array[]
  [out] 出力値の配列

result[]
  [out] 出力値の配列

戻り値

成功の場合はtrue、それ以外の場合はfalse。

© 2000–2025, MetaQuotes Ltd.

MathCumulativeSum

累積合計で配列を生成します。

新しい配列に結果を出力するバージョン:

```
bool  MathCumulativeSum(
   const double&  array[],   // 値の配列
   double&        result[]   // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathCumulativeSum(
   double&        array[]    // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

array[]

[out] 出力値の配列

result[]

[out] 出力値の配列

戻り値

成功の場合は true、それ以外の場合は false。

© 2000–2025, MetaQuotes Ltd.

MathCumulativeProduct

累積積で配列を生成します。

新しい配列に結果を出力するバージョン:

```
bool  MathCumulativeProduct(
   const double&  array[],   // 値の配列
   double&        result[]   // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathCumulativeProduct(
   double&        array[]    // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

result[]

[out] 出力値の配列

array[]

[out] 出力値の配列

戻り値

成功の場合は true、それ以外の場合は false。

© 2000–2025, MetaQuotes Ltd.

MathCumulativeMin

累積最小値で配列を生成します。

新しい配列に結果を出力するバージョン:

```
bool  MathCumulativeMin(
   const double&  array[],   // 値の配列
   double&        result[]   // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathCumulativeMin(
   double&        array[]    // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

result[]

[out] 出力値の配列

array[]

[out] 出力値の配列

戻り値

成功の場合は true、それ以外の場合は false。

© 2000–2025, MetaQuotes Ltd.

MathCumulativeMax

累積最大値で配列を生成します。

新しい配列に結果を出力するバージョン:

```
bool  MathCumulativeMax(
   const double&  array[],   // 値の配列
   double&        result[]   // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathCumulativeMax(
   double&        array[]    // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

result[]

[out] 出力値の配列

array[]

[out] 出力値の配列

戻り値

成功の場合は true、それ以外の場合は false。

© 2000–2025, MetaQuotes Ltd.

MathSin

配列要素のsin(x)関数の値を計算します。

新しい配列に結果を出力するバージョン:

```
bool  MathSin(
   const double&  array[],   // 値の配列
   double&        result[]   // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathSin(
   double&        array[]    // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

result[]

[out] 出力値の配列

array[]

[out] 出力値の配列

戻り値

成功の場合はtrue、それ以外の場合はfalse。

© 2000–2025, MetaQuotes Ltd.

MathCos

配列要素のcos(x)関数の値を計算します。

新しい配列に結果を出力するバージョン:

```
bool  MathCos(
   const double&  array[],   // 値の配列
   double&        result[]   // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathCos(
   double&        array[]    // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

result[]

[out] 出力値の配列

array[]

[out] 出力値の配列

戻り値

成功の場合はtrue、それ以外の場合はfalse。

© 2000–2025, MetaQuotes Ltd.

MathTan

配列要素のtan(x)関数の値を計算します。

新しい配列に結果を出力するバージョン:

```
bool  MathTan(
   const double&  array[],   // 値の配列
   double&        result[]   // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathTan(
   double&        array[]    // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

result[]

[out] 出力値の配列

array[]

[out] 出力値の配列

戻り値

成功の場合はtrue、それ以外の場合はfalse。

© 2000–2025, MetaQuotes Ltd.

MathArcsin

配列要素のarcsin(x)関数の値を計算します。

新しい配列に結果を出力するバージョン:

```
bool  MathArcsin(
   const double&  array[],   // 値の配列
   double&        result[]   // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathArcsin(
   double&        array[]    // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

result[]

[out] 出力値の配列

array[]

[out] 出力値の配列

戻り値

成功の場合はtrue、それ以外の場合はfalse。

© 2000-2025, MetaQuotes Ltd.

MathArccos

配列要素のarccos(x)関数の値を計算します。

新しい配列に結果を出力するバージョン:

```
bool  MathArccos(
   const double&   array[],    // 値の配列
   double&         result[]    // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathArccos(
   double&         array[]    // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

result[]

[out] 出力値の配列

array[]

[out] 出力値の配列

戻り値

成功の場合はtrue、それ以外の場合はfalse。

© 2000-2025, MetaQuotes Ltd.

MathArctan

配列要素のarctan(x)関数の値を計算します。

新しい配列に結果を出力するバージョン:

```
bool  MathArctan(
   const double&   array[],   // 値の配列
   double&         result[]   // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathArctan(
   double&        array[]    // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

result[]

[out] 出力値の配列

array[]

[out] 出力値の配列

戻り値

成功の場合はtrue、それ以外の場合はfalse。

© 2000–2025, MetaQuotes Ltd.

MathSinPi

配列要素のsin(pi*x) 関数の値を計算します。

新しい配列に結果を出力するバージョン:

```
bool  MathSinPi(
   const double&  array[],   // 値の配列
   double&        result[]   // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathSinPi(
   double&        array[]    // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

result[]

[out] 出力値の配列

array[]

[out] 出力値の配列

戻り値

成功の場合は true、それ以外の場合は false。

© 2000–2025, MetaQuotes Ltd.

MathCosPi

配列要素のcos(pi*x) 関数の値を計算します。

新しい配列に結果を出力するバージョン:

```
bool  MathCosPi(
   const double&  array[],   // 値の配列
   double&        result[]   // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathCosPi(
   double&        array[]    // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

result[]

[out] 出力値の配列

array[]

[out] 出力値の配列

戻り値

成功の場合は true、それ以外の場合は false。

© 2000–2025, MetaQuotes Ltd.

MathTanPi

配列要素のtan(pi*x) 関数の値を計算します。

新しい配列に結果を出力するバージョン:

```
bool  MathTanPi(
   const double&  array[],   // 値の配列
   double&        result[]   // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathTanPi(
   double&        array[]    // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

result[]

[out] 出力値の配列

array[]

[out] 出力値の配列

戻り値

成功の場合は true、それ以外の場合は false。

© 2000–2025, MetaQuotes Ltd.

MathAbs

配列要素の絶対値を計算します。

新しい配列に結果を出力するバージョン:

```
bool  MathAbs(
   const double&  array[],   // 値の配列
   double&        result[]   // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathAbs(
   double&        array[]    // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

result[]

[out] 出力値の配列

array[]

[out] 出力値の配列

戻り値

成功の場合は true、それ以外の場合は false。

© 2000–2025, MetaQuotes Ltd.

MathCeil

配列要素が最も近い値に切り上げられた場合の整数値を返します。

新しい配列に結果を出力するバージョン:

```
bool  MathCeil(
   const double&  array[],   // 値の配列
   double&        result[]   // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathCeil(
   double&        array[]    // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

result[]

[out] 出力値の配列

array[]

[out] 出力値の配列

戻り値

成功の場合は true、それ以外の場合は false。

© 2000–2025, MetaQuotes Ltd.

MathFloor

配列要素が最も近い値に切り下げられた場合の整数値を返します。

新しい配列に結果を出力するバージョン:

```
bool  MathFloor(
   const double&  array[],   // 値の配列
   double&        result[]   // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathFloor(
   double&        array[]    // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

result[]

[out] 出力値の配列

array[]

[out] 出力値の配列

戻り値

成功の場合は true、それ以外の場合は false。

© 2000–2025, MetaQuotes Ltd.

MathSqrt

配列要素の平方根を計算します。

新しい配列に結果を出力するバージョン:

```
bool  MathSqrt(
   const double&  array[],   // 値の配列
   double&        result[]   // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathSqrt(
   double&         array[]    // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

result[]

[out] 出力値の配列

array[]

[out] 出力値の配列

戻り値

成功の場合は true、それ以外の場合は false。

© 2000–2025, MetaQuotes Ltd.

MathExp

配列要素のexp(x)関数の値を計算します。

新しい配列に結果を出力するバージョン:

```
bool  MathExp(
   const double&  array[],   // 値の配列
   double&        result[]   // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathExp(
   double&       array[]    // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

result[]

[out] 出力値の配列

array[]

[out] 出力値の配列

戻り値

成功の場合はtrue、それ以外の場合はfalse。

© 2000–2025, MetaQuotes Ltd.

MathPow

配列要素のpow(x, power)関数の値を計算します。

新しい配列に結果を出力するバージョン:

```
bool  MathPow(
   const double&  array[],    // 値の配列
   const double   power,      // 累乗
   double&        result[]    // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathPow(
   double&        array[],    // 値の配列
   const double   power       // 累乗
   )
```

パラメータ

array[]

[in] 値の配列

result[]

[out] 出力値の配列

array[]

[out] 出力値の配列

戻り値

成功の場合はtrue、それ以外の場合はfalse。

© 2000–2025, MetaQuotes Ltd.

MathLog

配列要素のlog(x)関数の値を計算します。

自然対数を計算して新しい配列に結果を出力するバージョン:

```
bool  MathLog(
   const double&  array[],      // 値の配列
   double&        result[]      // 結果の配列
   )
```

自然対数を計算して元の配列に結果を出力するバージョン:

```
bool  MathLog(
   double&        array[]       // 値の配列
   )
```

指定された底までの対数を計算して結果を新しい配列に出力するバージョン:

```
bool  MathLog(
   const double&   array[],     // 値の配列
   const double    base,        // 対数の底
   double&         result[]     // 結果の配列
   )
```

指定された底までの対数を計算して結果を元の配列に出力するバージョン:

```
bool  MathLog(
   double&         array[],     // 値の配列
   const double    base         // 対数の底
   )
```

パラメータ

array[]
[in] 値の配列

base
[in] 対数の底

array[]
[out] 出力値の配列

result[]
[out] 出力値の配列

戻り値

成功の場合はtrue、それ以外の場合はfalse。

© 2000-2025, MetaQuotes Ltd.

MathLog2

配列要素の2を底とする対数を計算します。

新しい配列に結果を出力するバージョン:

```
bool  MathLog2(
   const double&  array[],   // 値の配列
   double&        result[]   // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathLog2(
   double&        array[]    // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

result[]

[out] 出力値の配列

array[]

[out] 出力値の配列

戻り値

成功の場合は true、それ以外の場合は false。

© 2000–2025, MetaQuotes Ltd.

MathLog10

配列要素の10を底とする対数を計算します。

新しい配列に結果を出力するバージョン:

```
bool  MathLog10(
   const double&   array[],   // 値の配列
   double&         result[]   // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathLog10(
   double&        array[]    // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

result[]

[out] 出力値の配列

array[]

[out] 出力値の配列

戻り値

成功の場合は true、それ以外の場合は false。

© 2000–2025, MetaQuotes Ltd.

MathLog1p

配列要素のlog(1+x)関数の値を計算します。

新しい配列に結果を出力するバージョン:

```
bool  MathLog1p(
   const double&  array[],   // 値の配列
   double&        result[]   // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathLog1p(
   double&        array[]    // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

result[]

[out] 出力値の配列

array[]

[out] 出力値の配列

戻り値

成功の場合はtrue、それ以外の場合はfalse。

© 2000-2025, MetaQuotes Ltd.

MathDifference

要素の差がy[i]=x[i+lag]-x[i]の配列を生成します。

実数配列の単一生成のバージョン:

```
bool  MathDifference(
   const double  &array[],     // 値の配列
   const int     lag,          // ラグ
   double        &result[]     // 結果の配列
   )
```

整数配列の単一生成のバージョン:

```
bool  MathDifference(
   const int     &array[],     // 値の配列
   const int     lag,          // ラグ
   int           &result[]     // 結果の配列
   )
```

実数配列の反復生成のバージョン(反復数は入力パラメータで設定):

```
bool  MathDifference(
   const double  &array[],     // 値の配列
   const int     lag,          // ラグ
   const int     differences,  // 反復数
   double        &result[]     // 結果の配列
   )
```

整数配列の反復生成のバージョン(反復数は入力パラメータで設定):

```
bool  MathDifference(
   const int&    array[],      // 値の配列
   const int     lag,          // ラグ
   const int     differences,  // 反復数
   int&          result[]      // 結果の配列
   )
```

パラメータ

array[]

[in] 値の配列

lag

[in] ラグパラメータ

differences

[in] 反復数

result[]

[out] 結果を出力する配列.

戻り値

© 2000-2025, MetaQuotes Ltd.

成功の場合はtrue、それ以外の場合はfalse。

© 2000–2025, MetaQuotes Ltd.

MathSample

配列要素からランダムサンプルを生成します。

実数配列を扱うためのバージョン:

```
bool  MathSample(
   const double&  array[],          // 値の配列
   const int      count,            // カウント
   double&        result[]          // 結果の配列
   )
```

整数配列を扱うためのバージョン:

```
bool  MathSample(
   const int&     array[],          // 値の配列
   const int      count,            // カウント
   int&           result[]          // 結果の配列
   )
```

実数配列を扱うためのバージョン。サンプルは置き換えで得ることが可能です:

```
bool  MathSample(
   const double&  array[],          // 値の配列
   const int      count,            // カウント
   const bool     replace,          // フラグ
   double&        result[],         // 結果の配列
   )
```

整数配列を扱うためのバージョン: サンプルは置き換えで得ることが可能です:

```
bool  MathSample(
   const int&     array[],          // 値の配列
   const int      count,            // カウント
   const bool     replace,          // フラグ
   int&           result[]          // 結果の配列
   )
```

サンプリングの確率が定義されている実数配列を扱うためのバージョン:

```
bool  MathSample(
   const double&  array[],          // 値の配列
   double&        probabilities[],  // 確率配列
   const int      count,            // カウント
   double&        result[]          // 結果の配列
   )
```

サンプリングの確率が定義されている整数配列を扱うためのバージョン:

```
bool  MathSample(
   const int&     array[],          // 値の配列
   double&        probabilities[],  // 確率配列
```

© 2000–2025, MetaQuotes Ltd.

```
   const int       count,              // カウント
   int&            result[]            // 結果の配列
   )
```

サンプリングの確率が定義されている実数配列を扱うためのバージョン：サンプルは置き換えで得ることが可能です：

```
bool  MathSample(
   const double&  array[],            // 値の配列
   double&        probabilities[],    // 確率配列
   const int      count,              // カウント
   const bool     replace,            // フラグ
   double&        result[]            // 結果の配列
   )
```

サンプリングの確率が定義されている整数配列を扱うためのバージョン：サンプルは置き換えで得ることが可能です：

```
bool  MathSample(
   const int&     array[],            // 値の配列
   double&        probabilities[],    // 確率配列
   const int      count,              // カウント
   const bool     replace,            // フラグ
   int&           result[]            // 結果の配列
   )
```

パラメータ

array[]

[in] 整数配列

probabilities[]

[in] 要素をサンプルするための確率配列

count

[in] 要素数

replace

[in] 置き換えによるサンプリングを可能にするパラメータ

result[]

[out] 結果を出力する配列.

戻り値

成功の場合は true、それ以外の場合は false。

注意事項

replace=true引数を使用すると、元のシーケンスに置き換えて元の要素を無作為抽出することができます。

© 2000–2025, MetaQuotes Ltd.

MathTukeySummary

配列要素のTukeyの五数要約（最小値、第1四分位数、第2四分位数、第3四分位数、最大値）を計算します。

```
bool  MathTukeySummary(
   const double&  array[],        // 値の配列
   const bool     removeNAN,      // フラグ
   double&        minimum,        // 最小値
   double&        lower_hinge,    // 第1四分位数
   double&        median,         // 第2四分位数
   double&        upper_hinge,    // 第3四分位数
   double&        maximum         // 最大値
   )
```

パラメータ

array[]

[in] 実数配列

removeNAN

[in] 数値以外の値を削除するかどうかを示すフラグ

minimum

[out] 最小値を格納するための変数

lower_hinge

[out] 第1四分位数を格納するための変数

median

[out] 第2四分位数を格納するための変数

upper_hinge

[out] 第3四分位数を格納するための変数

maximum

[out] 最大値を格納するための変数

戻り値

成功の場合はtrue、それ以外の場合はfalse。

© 2000-2025, MetaQuotes Ltd.

MathRange

配列要素の最小値と最大値を計算します。

```
bool  MathRange(
   const double&  array[],  // 値の配列
   double&        min,      // 最小値
   double&        max       // 最大値
   )
```

パラメータ

array[]

[in] 値の配列

min

[out] 最小値を格納するための変数

max

[out] 最大値を格納するための変数

戻り値

成功の場合は true、それ以外の場合は false。

© 2000–2025, MetaQuotes Ltd.

MathMin

すべての配列要素の最小値を返します。

```
double  MathMin(
   const double&  array[]   // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

戻り値

最小値

© 2000–2025, MetaQuotes Ltd.

MathMax

すべての配列要素の最大値を返します。

```
double  MathMax(
   const double&  array[]   // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

戻り値

最大値

© 2000–2025, MetaQuotes Ltd.

MathSum

配列要素の和を返します。

```
double  MathSum(
   const double&  array[]   // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

戻り値

配列要素の和

© 2000–2025, MetaQuotes Ltd.

MathProduct

配列要素の積を返します。

```
double  MathProduct(
   const double&  array[]   // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

戻り値

要素の積

© 2000–2025, MetaQuotes Ltd.

MathStandardDeviation

配列要素の標準偏差を計算します。

```
double  MathStandardDeviation(
   const double&  array[]   // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

戻り値

標準偏差

© 2000–2025, MetaQuotes Ltd.

MathAverageDeviation

この関数は配列要素の平均絶対偏差を計算します。

```
double  MathAverageDeviation(
   const double&  array[]   // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

戻り値

配列要素の平均絶対偏差

© 2000–2025, MetaQuotes Ltd.

MathMedian

配列要素の中央値を計算します。

```
double  MathMedian(
   double&  array[]   // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

戻り値

中央値

© 2000–2025, MetaQuotes Ltd.

MathMean

配列要素の平均値を計算します。

```
double  MathMean(
   const double&  array[]   // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

戻り値

平均値

© 2000–2025, MetaQuotes Ltd.

MathVariance

この関数は、配列要素の分散（二次モーメント）を計算します。

```
double  MathVariance(
   const double&  array[]   // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

戻り値

分散の値

© 2000–2025, MetaQuotes Ltd.

MathSkewness

この関数は、配列要素の歪度（三次モーメント）を計算します。

```
double  MathSkewness(
   const double&  array[]   // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

戻り値

歪度

© 2000-2025, MetaQuotes Ltd.

MathKurtosis

この関数は、配列要素の尖度（四次モーメント）を計算します。

```
double  MathKurtosis(
   const double&  array[]   // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

戻り値

尖度

© 2000–2025, MetaQuotes Ltd.

MathExpm1

配列要素のexp(x)-1関数の値を計算します。

新しい配列に結果を出力するバージョン:

```
bool  MathExpm1(
   const double&   array[],   // 値の配列
   double&         result[]   // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathExpm1(
   double&         array[]    // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

result[]

[out] 出力値の配列

array[]

[out] 出力値の配列

戻り値

成功の場合はtrue、それ以外の場合はfalse。

© 2000-2025, MetaQuotes Ltd.

MathSinh

配列要素のsinh(x)関数の値を計算します。

新しい配列に結果を出力するバージョン:

```
bool  MathSinh(
   const double&  array[],   // 値の配列
   double&        result[]   // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathSinh(
   double&       array[]    // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

result[]

[out] 出力値の配列

array[]

[out] 出力値の配列

戻り値

成功の場合はtrue、それ以外の場合はfalse。

© 2000–2025, MetaQuotes Ltd.

MathCosh

配列要素のcosh(x)関数の値を計算します。

新しい配列に結果を出力するバージョン:

```
bool  MathCosh(
   const double&  array[],   // 値の配列
   double&        result[]   // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathCosh(
   double&        array[]    // 値の配列
   )
```

パラメータ

array[]
  [in] 値の配列

result[]
  [out] 出力値の配列

array[]
  [out] 出力値の配列

戻り値

成功の場合はtrue、それ以外の場合はfalse。

© 2000–2025, MetaQuotes Ltd.

MathTanh

配列要素のtanh(x)関数の値を計算します。

新しい配列に結果を出力するバージョン:

```
bool  MathTanh(
   const double&  array[],   // 値の配列
   double&        result[]   // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathTanh(
   double&        array[]    // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

result[]

[out] 出力値の配列

array[]

[out] 出力値の配列

戻り値

成功の場合はtrue、それ以外の場合はfalse。

© 2000–2025, MetaQuotes Ltd.

MathArcsinh

配列要素のarcsinh(x)関数の値を計算します。

新しい配列に結果を出力するバージョン:

```
bool  MathArcsinh(
   const double&  array[],   // 値の配列
   double&        result[]   // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathArcsinh(
   double&         array[]    // 値の配列
   )
```

パラメータ

array[]
  [in] 値の配列

result[]
  [out] 出力値の配列

array[]
  [out] 出力値の配列

戻り値

成功の場合はtrue、それ以外の場合はfalse。

© 2000-2025, MetaQuotes Ltd.

MathArccosh

配列要素のarccosh(x)関数の値を計算します。

新しい配列に結果を出力するバージョン:

```
bool  MathArccosh(
   const double&  array[],   // 値の配列
   double&        result[]   // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathArccosh(
   double&        array[]    // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

result[]

[out] 出力値の配列

array[]

[out] 出力値の配列

戻り値

成功の場合はtrue、それ以外の場合はfalse。

© 2000-2025, MetaQuotes Ltd.

MathArctanh

配列要素のarctanh(x)関数の値を計算します。

新しい配列に結果を出力するバージョン:

```
bool  MathArctanh(
   const double&  array[],   // 値の配列
   double&        result[]   // 結果の配列
   )
```

元の配列に結果を出力するバージョン:

```
bool  MathArctanh(
   double&        array[]    // 値の配列
   )
```

パラメータ

array[]

[in] 値の配列

result[]

[out] 出力値の配列

array[]

[out] 出力値の配列

戻り値

成功の場合はtrue、それ以外の場合はfalse。

© 2000–2025, MetaQuotes Ltd.

MathSignif

値を指定された仮数の桁数を持つ値に丸めます。

実数値を扱うためのバージョン:

```
double  MathSignif(
   const double   x,          // 値
   const int      digits      // 小数点以下桁数
   )
```

戻り値

丸められた値

実数配列を扱って別配列に結果を出力するバージョン:

```
bool  MathSignif(
   const double&  array[],    // 値の配列
   int            digits,     // 小数点以下桁数
   double         result[]    // 結果の配列
   )
```

戻り値

成功の場合は true、それ以外の場合は false。

実数配列を扱って元の配列に結果を出力するバージョン:

```
bool  MathSignif(
   double&        array[],    // 値の配列
   int            digits      // 小数点以下桁数
   )
```

戻り値

成功の場合は true、それ以外の場合は false。

パラメータ

x

[in] 丸められる実数

digits

[in] 小数点以下の桁数

array[]

[in] 実数配列

array[]

[out] 出力値の配列

result[]

[out] 出力値の配列

© 2000–2025, MetaQuotes Ltd.

© 2000–2025, MetaQuotes Ltd.

MathRank

配列要素のランクを計算します。

実数配列を扱うためのバージョン:

```
bool  MathRank(
   const double&  array[],  // 値の配列
   double&        rank[]    // ランクの配列
   )
```

整数配列を扱うためのバージョン:

```
bool  MathRank(
   const int&     array[],  // 値の配列
   double&        rank[]    // ランクの配列
   )
```

パラメータ

array[]

[in] 値の配列

rank[]

[out] ランクを出力するための配列

戻り値

成功の場合は true、それ以外の場合は false。

© 2000–2025, MetaQuotes Ltd.

MathCorrelationPearson

ピアソンの相関係数を計算します。

実数配列を扱うためのバージョン:

```
bool  MathCorrelationPearson(
   const double&  array1[],  // 1番目の値の配列
   const double&  array2[],  // 2番目の値の配列/t5>
   double&        r          // 相関係数
   )
```

整数配列を扱うためのバージョン:

```
bool  MathCorrelationPearson(
   const int&     array1[],  // 1番目の値の配列
   const int&     array2[],  // 2番目の値の配列/t5>
   double&        r          // 相関係数
   )
```

パラメータ

array1[]

[in] 1番目の値の配列.

array2[]

[in] 2番目の値の配列

r

[out] 相関係数を格納するための変数

戻り値

成功の場合は true、それ以外の場合は false。

© 2000–2025, MetaQuotes Ltd.

MathCorrelationSpearman

スピアマンの相関係数を計算します。

実数配列を扱うためのバージョン:

```
bool  MathCorrelationSpearman(
   const double&  array1[],  // 1番目の値の配列
   const double&  array2[],  // 2番目の値の配列/t5>
   double&        r          // 相関係数
   )
```

整数配列を扱うためのバージョン:

```
bool  MathCorrelationSpearman(
   const int&     array1[],   // 1番目の値の配列
   const int&     array2[],   // 2番目の値の配列/t5>
   double&        r           // 相関係数
   )
```

パラメータ

array1[]

[in] 1番目の値の配列.

array2[]

[in] 2番目の値の配列

r

[out] 相関係数を格納するための変数

戻り値

成功の場合は true、それ以外の場合は false。

© 2000–2025, MetaQuotes Ltd.

MathCorrelationKendall

ケンドールの相関係数を計算します。

実数配列を扱うためのバージョン:

```
bool  MathCorrelationKendall(
   const double&  array1[],  // 1番目の値の配列
   const double&  array2[],  // 2番目の値の配列/t5>
   double&        tau        // 相関係数
   )
```

整数配列を扱うためのバージョン:

```
bool  MathCorrelationKendall(
   const int&     array1[],  // 1番目の値の配列
   const int&     array2[],  // 2番目の値の配列/t5>
   double&        tau        // 相関係数
   )
```

パラメータ

array1[]

[in] 1番目の値の配列.

array2[]

[in] 2番目の値の配列

tau

[out] 相関係数を格納するための変数

戻り値

成功の場合は true、それ以外の場合は false。

© 2000–2025, MetaQuotes Ltd.

MathQuantile

指定された確率に対応するサンプルの分位数Q[i](p) = (1 - gamma)*x[j] + gamma*x[j+1]を計算します。

```
bool  MathQuantile(
   const double&  array[],     // 値の配列
   const double&  probs[],     // 確率配列
   double&        quantile[]   // 分位数を出力するための配列
   )
```

パラメータ

array[]

[in] 値の配列

probs[]

[in] 確率配列

quantile[]

[out] 分位数を出力するための配列

戻り値

成功の場合はtrue、それ以外の場合はfalse。

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityEmpirical

この関数は、配列からの乱数に対する経験的確率密度関数（pdf）を計算します。

```
bool  MathProbabilityDensityEmpirical(
   const double&  array[],   // 乱数の配列
   const int      count,     // 対の数
   double&        x[],       // x値の配列
   double&        pdf[]      // pdf値の配列
   )
```

パラメータ

array[]

[in] 乱数値の配列

count

[in] (x, pdf(x)) 対の数

x[]

[out] x値を出力するための配列

pdf[]

[out] pdf(x)値を出力するための配列

戻り値

成功の場合はtrue、それ以外の場合はfalse。

© 2000–2025, MetaQuotes Ltd.

MathCumulativeDistributionEmpirical

この関数は、配列からの乱数に対する経験的累積分布関数（cdf）を計算します。

```
bool  MathCumulativeDistributionEmpirical(
   const double&  array[],    // 乱数の配列
   const int      count,      // 対の数
   double&        x[],        // x値の配列
   double&        cdf[]       // cdf値の配列
   )
```

パラメータ

array[]

[in] 乱数値の配列

count

[in] (x, cdf(x)) 対の数

x[]

[out] x値を出力するための配列

cdf[]

[out] cdf(x)値を出力するための配列

戻り値

成功の場合はtrue、それ以外の場合はfalse。

© 2000–2025, MetaQuotes Ltd.

ファジーはファジー論理を扱うライブラリです。

ファジー論理は、真理が言語変数として記述されている伝統的なアリステレオ論理の合成です。古典論理に相当するファジー論理は、定義されたファジー集合上で、独自のファジー論理演算を有します。ファジー集合と通常の集合には同じ演算がありますが、計算ははるかに困難です。また、ファジー集合の組成はファジー集合として構成していることには注意するべきです。

典的論理と隔てるファジー論理の主な原則は現実を反映した最大限の近接性と主観性の高いレベルであり、これらは計算に大きな誤差をもたらす可能性があります。

ファジーモデル（またはシステム）とは、その計算がファジー論理に基づいている数学的モデルです。このようなモデルの構築は、研究対象に弱い形式性があり、厳密に計算しようとすると複雑になりすぎてしまう場合に有効です。これらのモデルの出力値（エラーモデル）の品質は、このモデルを設定するエキスパートアドバイザーのみに直接依存します。誤差を最小化するための最良の方法は、最も完全かつ包括的なモデルを描き、その後、大規模なトレーニングセットをもって機械学習でそれを調整することです。

モデル構築のプロセスは、次の3つの段階に分けることができます。

1. モデルの入出力特性の定義
2. 知識ベースの構築
3. ファジー推論の方法を選択（マムダニまたは菅野）

第一段階の結果は直接2に影響を与え、モデルの将来を決定します。ルールベースなどと呼ばれることもある知識ベースは、入力と出力の関係性を定義する"if, then"ファジールールタイプのセットです。システム内のルールの数は制限されず、エキスパートアドバイザーによっても決定されます。ファジールールの一般形式は下記の通りです。

If ルール条件、then ルール結論

ルール条件は、現在のオブジェクトの状態を記述し、ルール結論は、この条件がオブジェクトにどのような影響を与えるかを記述します。ファジー推論によって決定されるので、条件と結論の一般的な見解は選択できません。

システム内の各ルールは重みを有し、この特性は、モデル内のルールの重要性を定義します。重み係数は、[0, 1]範囲内のルールに割り当てられています。重み付けデータは関連文献のファジーモデルの例では指定されていませんが、存在していない訳ではありません。実際には、データベースの各ルールのような場合、重み付けは1に等しく固定されます。各ルールの用語や結論には2種類が存在できます。

1. シンプル－1ファジー変数を含む
2. コンプレックス－複数のファジー変数を含む

モデルのファジー推論システムは作成された知識ベースに応じて決定されます。ファジー論理推論は、知識ベースとファジー演算を用いた、入力の現在値に応じたファジー集合の形態です。. ファジー推論の2つの主要な種類はマムダニ法と菅野法です。

© 2000–2025, MetaQuotes Ltd.

メンバーシップ関数

メンバーシップ関数　は、ユニバーサルセットのランダム要素のメンバーシップ度をファジー集合に計算することを可能にする関数です。したがって、メンバーシップ関数の領域は[0、1]の範囲内でなければなりません。

ほとんどの場合、メンバーシップ関数は連続的かつ単調です。

メンバーシップ関数のクラス	説明
CConstantMembershipFunction	メンバーシップ関数を座標軸と平行な直線として実装するクラス。
CCompositeMembershipFunction	メンバーシップ関数の構成を実装するためのクラス。
CDifferencTwoSigmoidalMembershipFunction	A1、A2、C1、C2パラメータを持つ2つのシグモイド関数の差の形でメンバーシップ関数を実装するためのクラス。
CGeneralizedBellShapedMembershipFunction	A、B、Cの各パラメータを持つ一般的なベル型メンバーシップ関数を実装するためのクラス。
CNormalCombinationMembershipFunction	B1、B2、Sigma1、Sigma2パラメータを持つ2面ガウスメンバーシップ関数を実装するためのクラス。
CNormalMembershipFunction	BとSigmaパラメータを持つ対称ガウスメンバーシップ関数を実装するためのクラス。
CP_ShapedMembershipFunction	A、B、C、Dパラメータを持つπ型メンバーシップ関数を実装するためのクラス。
CProductTwoSigmoidalMembershipFunction	A1、A2、C1、C2パラメータを持つ2つのシグモイド関数の積の形でメンバーシップ関数を実装するためのクラス。
CS_ShapedMembershipFunction	AとBのパラメータを持つS字メンバーシップ関数を実装するためのクラス。
CTrapezoidMembershipFunction	X1、X2、X3、X4パラメータを持つ台形メンバーシップ関数を実装するためのクラス。
CTriangularMembershipFunction	X1、X2、X3パラメータを持つ三角形メンバーシップ関数を実装するためのクラス。
CSigmoidalMembershipFunction	AとCのパラメータを持つシグモイドメンバーシップ関数を実装するためのクラス。
CZ_ShapedMembershipFunction	AとBのパラメータを持つz字メンバーシップ関数を実装するためのクラス。
IMembershipFunction	すべてのメンバーシップ関数の基本クラス

© 2000–2025, MetaQuotes Ltd.

CConstantMembershipFunction

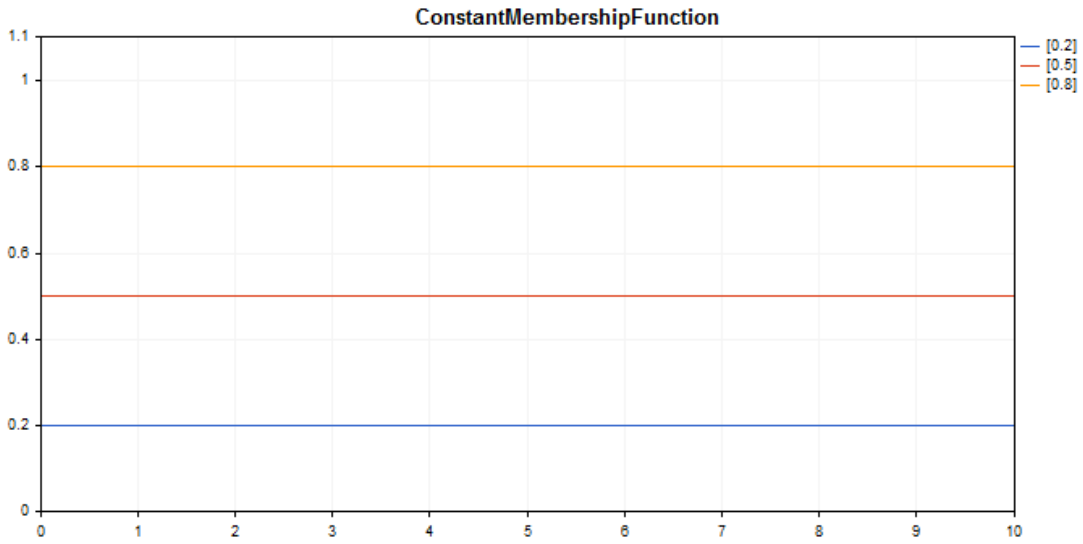
メンバーシップ関数を座標軸と平行な直線として実装するクラス。

説明

この関数は、次の式で表されます。

y(x)=c

したがって、関数のメンバーシップ度は数値軸全体で同じであり、コンストラクタで指定されたパラメータと同じです。



下記はチャートをプロットするための[サンプルコード](#)です。

宣言

```
class CConstantMembershipFuncion : public IMembershipFunction
```

タイトル

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

継承階層

[CObject](#)
  [IMembershipFunction](#)
    CConstantMembershipFunction

クラスメソッド

クラスメソッド	説明
[GetValue](#)	メンバーシップ関数の値を指定された引数で計算します。

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

例

```
//+------------------------------------------------------------------+
//|                                  ConstantMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- メンバーシップ関数を作成します
CConstantMembershipFunction func1(0.2);
CConstantMembershipFunction func2(0.5);
CConstantMembershipFunction func3(0.8);
//--- メンバーシップ関数用のラッパーを作成する
double ConstantMembershipFunction1(double x) { return(func1.GetValue(x)); }
double ConstantMembershipFunction2(double x) { return(func2.GetValue(x)); }
double ConstantMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- グラフィックを作成する
   CGraphic graphic;
   if(!graphic.Create(0,"ConstantMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"ConstantMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("ConstantMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- 曲線を作成する
   graphic.CurveAdd(ConstantMembershipFunction1,0.0,10.0,1.0,CURVE_LINES,"[0.2]");
   graphic.CurveAdd(ConstantMembershipFunction2,0.0,10.0,1.0,CURVE_LINES,"[0.5]");
   graphic.CurveAdd(ConstantMembershipFunction3,0.0,10.0,1.0,CURVE_LINES,"[0.8]");
//--- X軸のプロパティを設定する
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y軸のプロパティを設定する
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- プロット
```

© 2000–2025, MetaQuotes Ltd.

```
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

GetValue

メンバーシップ関数の値を指定された引数で計算します。

```
double  GetValue(
   const double  x      // メンバーシップ関数引数
   )
```

パラメータ

x

[in] メンバーシップ関数引数

戻り値

メンバーシップ関数名

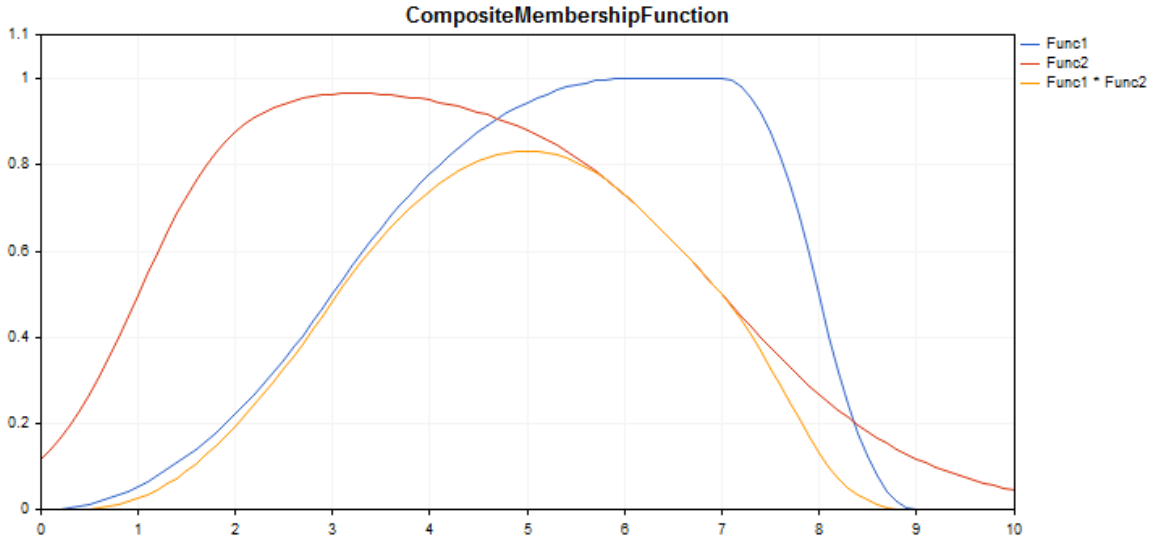
© 2000–2025, MetaQuotes Ltd.

CCompositeMembershipFunction

メンバーシップ関数の構成を実装するためのクラス。

説明

メンバーシップ関数の構成とは、指定された演算子を使用する2つ以上のメンバーシップ関数の組み合わせです。



下記はチャートをプロットするためのサンプルコードです。

宣言

```
class CCompositeMembershipFuncion : public IMembershipFunction
```

タイトル

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

継承階層

CObject
  IMembershipFunction
    CCompositeMembershipFunction

クラスメソッド

クラスメソッド	説明
CompositionType	構成演算子を設定します。
MembershipFunctions	メンバーシップ関数のリストを取得します。
GetValue	メンバーシップ関数の値を指定された引数で計算します。

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

例

```
//+------------------------------------------------------------------+
//|                                  CompositeMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- メンバーシップ関数を作成します
CProductTwoSigmoidalMembershipFunctions func1(2,1,-1,7);
CP_ShapedMembershipFunction func2(0,6,7,9);
CCompositeMembershipFunction composite(ProdMF,GetPointer(func1),GetPointer(func2));
//--- メンバーシップ関数用のラッパーを作成する
double ProductTwoSigmoidalMembershipFunctions(double x)
{ return(func1.GetValue(x)); }
double P_ShapedMembershipFunction(double x) { return(func2.GetValue(x)); }
double CompositeMembershipFunction(double x) { return(composite.GetValue(x)); }
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- グラフィックを作成する
   CGraphic graphic;
   if(!graphic.Create(0,"CompositeMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"CompositeMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("CompositeMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- 曲線を作成する
   graphic.CurveAdd(P_ShapedMembershipFunction,0.0,10.0,0.1,CURVE_LINES,"Func1");
   graphic.CurveAdd(ProductTwoSigmoidalMembershipFunctions,0.0,10.0,0.1,CURVE_LINES,
"Func2");
   graphic.CurveAdd(CompositeMembershipFunction,0.0,10.0,0.1,CURVE_LINES,"Func1 *
Func2");
//--- X軸のプロパティを設定する
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y軸のプロパティを設定する
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
```

© 2000–2025, MetaQuotes Ltd.

```
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- プロット
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

CompositionType

構成演算子を設定します。

```
void  CompositionType(
   MfCompositionType  value      // 演算子の種類
   )
```

パラメータ

value

[in] 構成演算子の種類

注意事項

次の演算子タイプが使用できます。

- MinMF(関数の最小値)
- MaxMF(関数の最大値)
- ProdMF(関数の積)
- SumMF(関数の和)

MembershipFunctions

構成に含まれたメンバーシップ関数のリストを取得します。

```
CList*  MembershipFunctions(
   void        // メンバーシップ関数のリスト
   )
```

戻り値

メンバーシップ関数のリスト

GetValue

メンバーシップ関数の値を指定された引数で計算します。

```
double  GetValue(
   const  x      // メンバーシップ関数引数
   )
```

パラメータ

x

[in] メンバーシップ関数引数

© 2000-2025, MetaQuotes Ltd.

戻り値

メンバーシップ関数名

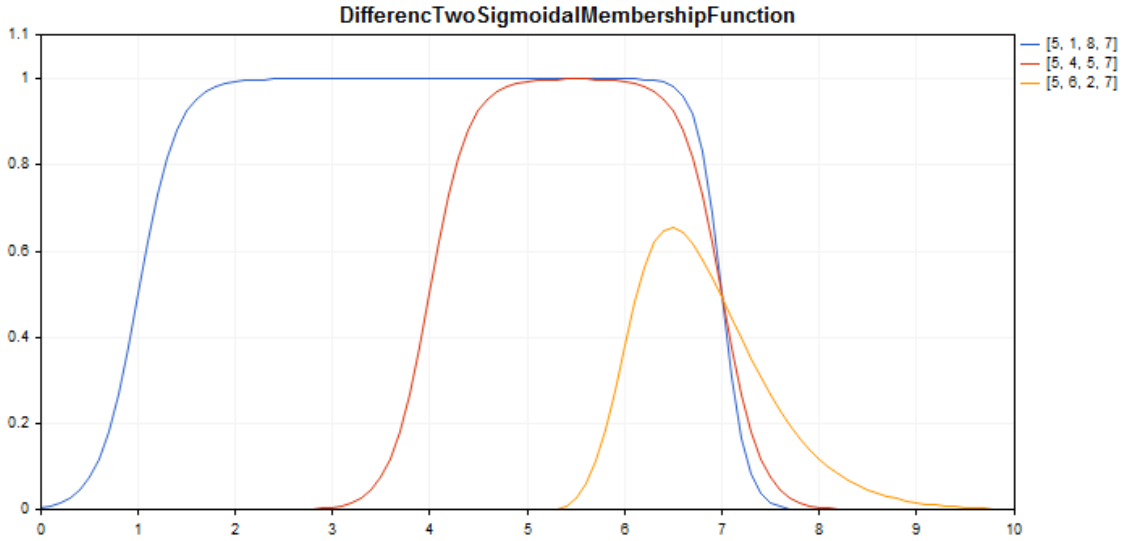
© 2000–2025, MetaQuotes Ltd.

CDifferencTwoSigmoidalMembershipFunction

A1、A2、C1、C2パラメータを持つ2つのシグモイド関数の差の形でメンバーシップ関数を実装するためのクラス。

説明

この関数はシグモイド曲線に基づいています。引数値で始まる値が1のメンバーシップ関数の作成が可能になります。このような関数は、「ショート」や「ロング」などの言語用語を設定する必要がある場合に適しています。



下記はチャートをプロットするためのサンプルコードです。

宣言

```
class CDifferencTwoSigmoidalMembershipFuncion : public IMembershipFunction
```

タイトル

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

継承階層

CObject

IMembershipFunction

CDifferencTwoSigmoidalMembershipFunction

クラスメソッド

クラスメソッド	説明
A1	1番目のメンバーシップ関数の勾配比を取得及び設定します。

© 2000-2025, MetaQuotes Ltd.

クラスメソッド	説 明
A2	2番目のメンバーシップ関数の勾配比を取得及び設定します。
C1	1番目のメンバーシップ関数の屈曲座標パラメータを取得及び設定します。
C2	2番目のメンバーシップ関数の屈曲座標パラメータを取得及び設定します。
GetValue	メンバーシップ関数の値を指定された引数で計算します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

例

```
//+------------------------------------------------------------------+
//|                      DifferencTwoSigmoidalMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- メンバーシップ関数を作成します
CDifferencTwoSigmoidalMembershipFunction func1(5,1,8,7);
CDifferencTwoSigmoidalMembershipFunction func2(5,4,5,7);
CDifferencTwoSigmoidalMembershipFunction func3(5,6,2,7);
//--- メンバーシップ関数用のラッパーを作成する
double DifferencTwoSigmoidalMembershipFunction1(double x)
{ return(func1.GetValue(x)); }
double DifferencTwoSigmoidalMembershipFunction2(double x)
{ return(func2.GetValue(x)); }
double DifferencTwoSigmoidalMembershipFunction3(double x)
{ return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- グラフィックを作成する
   CGraphic graphic;
   if(!graphic.Create(0,"DifferencTwoSigmoidalMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"DifferencTwoSigmoidalMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("DifferencTwoSigmoidalMembershipFunction");
   graphic.BackgroundMainSize(16);
```

© 2000–2025, MetaQuotes Ltd.

```
//--- 曲線を作成する
   graphic.CurveAdd(DifferencTwoSigmoidalMembershipFunction1,0.0,10.0,0.1,CURVE_LINE
S,"[5, 1, 8, 7]");
   graphic.CurveAdd(DifferencTwoSigmoidalMembershipFunction2,0.0,10.0,0.1,CURVE_LINE
S,"[5, 4, 5, 7]");
   graphic.CurveAdd(DifferencTwoSigmoidalMembershipFunction3,0.0,10.0,0.1,CURVE_LINE
S,"[5, 6, 2, 7]");
//--- X軸のプロパティを設定する
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y軸のプロパティを設定する
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- プロット
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A1（取得メソッド）

1番目のメンバーシップ関数の勾配比を取得します。

```
double  A1()
```

戻り値

勾配比値

A1（設定メソッド）

1番目のメンバーシップ関数の勾配比を設定します。

```
void  A1(
   const double  a1      // 勾配比値
   )
```

パラメータ

a1

[in] 勾配比値

A2（取得メソッド）

2番目のメンバーシップ関数の勾配比を取得します。

```
double  A2()
```

戻り値

© 2000–2025, MetaQuotes Ltd.

勾配比値

A2（設定メソッド）

2番目のメンバーシップ関数の勾配比を設定します。

```
void  A2(
   const double  a2      // 勾配比値
   )
```

パラメータ

a2

[in] 勾配比値

C1（取得メソッド）

1番目のメンバーシップ関数の屈曲座標パラメータを取得します。

```
double  C1()
```

戻り値

屈曲座標値

C1（設定メソッド）

1番目のメンバーシップ関数の屈曲座標パラメータを設定します。

```
void  C1(
   const double  c1      // 屈曲座標値
   )
```

パラメータ

c1

[in] 屈曲座標値

C2（取得メソッド）

2番目のメンバーシップ関数の屈曲座標パラメータを取得します。

```
double  C2()
```

戻り値

屈曲座標値

C2（設定メソッド）

2番目のメンバーシップ関数の屈曲座標パラメータを設定します。

© 2000–2025, MetaQuotes Ltd.

```
void  C2(
   const double  c2      // 屈曲座標値
   )
```

パラメータ

c2

[in] 屈曲座標値

GetValue

メンバーシップ関数の値を指定された引数で計算します。

```
double  GetValue(
   const double  x      // メンバーシップ関数引数
   )
```

パラメータ

x

[in] メンバーシップ関数引数

戻り値

メンバーシップ関数名

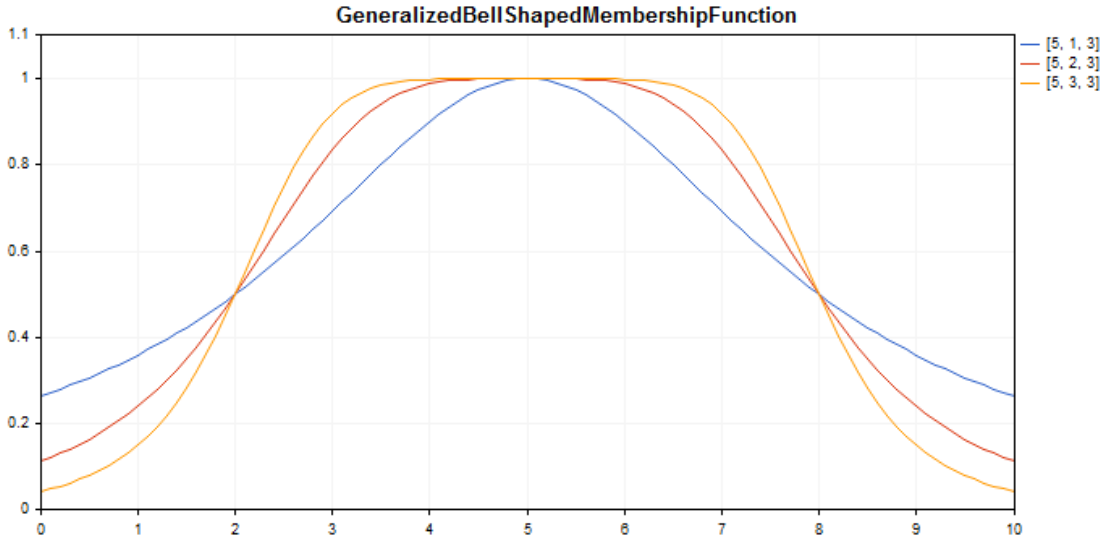
© 2000-2025, MetaQuotes Ltd.

CGeneralizedBellShapedMembershipFunction

A、B、Cの各パラメータを持つ一般的なベル型メンバーシップ関数を実装するためのクラス。

説明

一般化されたベル型メンバーシップ関数の形状はガウス関数に類似しています。関数は滑らかで、定義領域全体に沿ってゼロ以外の値をとります。



下記はチャートをプロットするためのサンプルコードです。

宣言

```
class CGeneralizedBellShapedMembershipFuncion : public IMembershipFunction
```

タイトル

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

継承階層

CObject
  IMembershipFunction
    CGeneralizedBellShapedMembershipFunction

クラスメソッド

クラスメソッド	説明
A	メンバーシップ関数の集中率を取得及び設定します。
B	メンバーシップ関数の勾配比を取得及び設定します。
C	メンバーシップ関数の最大座標を取得及び設定します。

© 2000–2025, MetaQuotes Ltd.

クラスメソッド	説 明
GetValue	メンバーシップ関 数 の値 を指 定 された引 数 で計 算 します。

クラスから継 承 されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

例

```
//+------------------------------------------------------------------+
//|                      GeneralizedBellShapedMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- メンバーシップ関数を作成します
CGeneralizedBellShapedMembershipFunction func1(5, 1, 3);
CGeneralizedBellShapedMembershipFunction func2(5, 2, 3);
CGeneralizedBellShapedMembershipFunction func3(5, 3, 3);
//--- メンバーシップ関数用のラッパーを作成する
double GeneralizedBellShapedMembershipFunction1(double x)
{ return(func1.GetValue(x)); }
double GeneralizedBellShapedMembershipFunction2(double x)
{ return(func2.GetValue(x)); }
double GeneralizedBellShapedMembershipFunction3(double x)
{ return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- グラフィックを作成する
   CGraphic graphic;
   if(!graphic.Create(0,"GeneralizedBellShapedMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"GeneralizedBellShapedMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("GeneralizedBellShapedMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- 曲線を作成する
   graphic.CurveAdd(GeneralizedBellShapedMembershipFunction1,0.0,10.0,0.1,CURVE_LINE
S,"[5, 1, 3]");
   graphic.CurveAdd(GeneralizedBellShapedMembershipFunction2,0.0,10.0,0.1,CURVE_LINE
S,"[5, 2, 3]");
   graphic.CurveAdd(GeneralizedBellShapedMembershipFunction3,0.0,10.0,0.1,CURVE_LINE
S,"[5, 3, 3]");
//--- X軸のプロパティを設定する
```

© 2000–2025, MetaQuotes Ltd.

```
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y軸のプロパティを設定する
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- プロット
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A（取得メソッド）

メンバーシップ関数の集中率を取得します。

```
double  A()
```

戻り値

集中率値

A（設定メソッド）

メンバーシップ関数の集中率を設定します。

```
void  A(
   const double  a      // 集中率値
   )
```

パラメータ

a

[in] メンバーシップ関数集中率値

B（取得メソッド）

メンバーシップ関数の勾配比を取得します。

```
double  B()
```

戻り値

勾配比値

B（設定メソッド）

© 2000–2025, MetaQuotes Ltd.

メンバーシップ関数の勾配比を設定します。

```
void  B(
   const double  b      // 勾配比値
   )
```

パラメータ

b

[in] メンバーシップ関数の勾配比

C（取得メソッド）

メンバーシップ関数の最大座標を取得します。

```
double  C()
```

戻り値

メンバーシップ関数の最大座標

C（設定メソッド）

メンバーシップ関数の最大座標を設定します。

```
void  C(
   const double  c      // 最大座標値
   )
```

パラメータ

c

[in] メンバーシップ関数の最大座標

GetValue

メンバーシップ関数の値を指定された引数で計算します。

```
double  GetValue(
   const  x      // メンバーシップ関数引数
   )
```

パラメータ

x

[in] メンバーシップ関数引数

戻り値

メンバーシップ関数名

© 2000–2025, MetaQuotes Ltd.

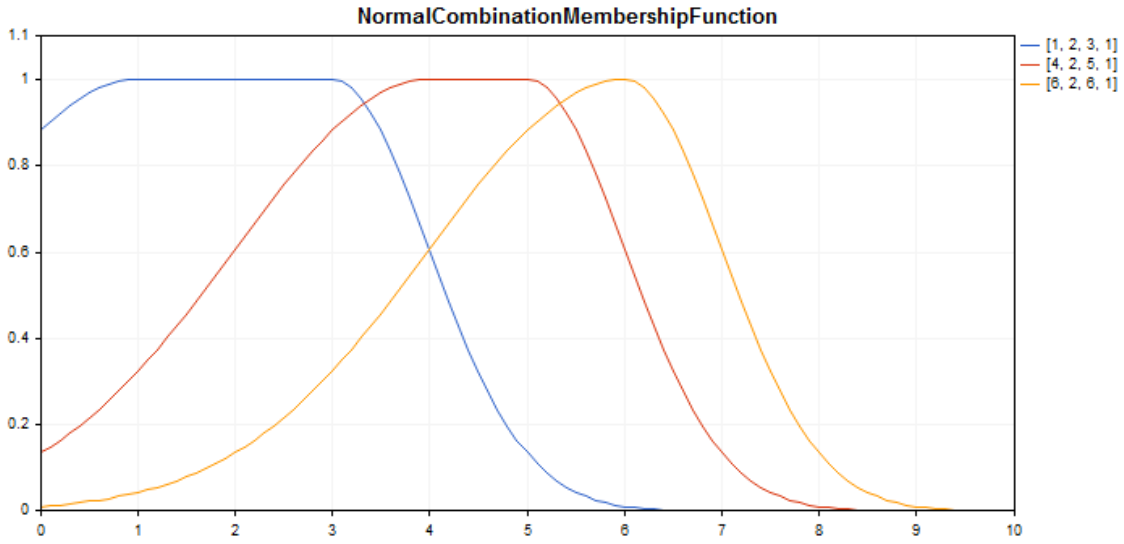
© 2000–2025, MetaQuotes Ltd.

CNormalCombinationMembershipFunction

B1、B2、Sigma1、Sigma2パラメータを持つ2面ガウスメンバーシップ関数を実装するためのクラス。

説明

2面ガウスメンバーシップ関数は、ガウス分布を用いて形成されます。非対称メンバーシップ関数の設定が可能です。関数は滑らかで、定義領域全体に沿ってゼロ以外の値をとります。



下記はチャートをプロットするための[サンプルコード](#)です。

宣言

```
class CNormalCombinationMembershipFuncion : public IMembershipFunction
```

タイトル

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

継承階層

[CObject](#)
  [IMembershipFunction](#)
    CNormalCombinationMembershipFunction

クラスメソッド

クラスメソッド	説明
[B1](#)	1番目のメンバーシップ関数の中心値を取得及び設定します。

© 2000-2025, MetaQuotes Ltd.

クラスメソッド	説明
B2	2番目のメンバーシップ関数の中心値を取得及び設定します。
Sigma1	メンバーシップ関数の曲率の1番目のパラメータを取得及び設定します。
Sigma2	メンバーシップ関数の曲率の2番目のパラメータを取得及び設定します。
GetValue	メンバーシップ関数の値を指定された引数で計算します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

例

```
//+------------------------------------------------------------------+
//|                          NormalCombinationMembershipFunction.mq5 |
//|                       Copyright 2016, MetaQuotes Software Corp.  |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- メンバーシップ関数を作成します
CNormalCombinationMembershipFunction func1(1,2,3,1);
CNormalCombinationMembershipFunction func2(4,2,5,1);
CNormalCombinationMembershipFunction func3(6,2,6,1);
//--- メンバーシップ関数用のラッパーを作成する
double NormalCombinationMembershipFunction1(double x) { return(func1.GetValue(x)); }
double NormalCombinationMembershipFunction2(double x) { return(func2.GetValue(x)); }
double NormalCombinationMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- グラフィックを作成する
   CGraphic graphic;
   if(!graphic.Create(0,"NormalCombinationMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"NormalCombinationMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("NormalCombinationMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- 曲線を作成する
```

© 2000–2025, MetaQuotes Ltd.

```
   graphic.CurveAdd(NormalCombinationMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[
1, 2, 3, 1]");
   graphic.CurveAdd(NormalCombinationMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[
4, 2, 5, 1]");
   graphic.CurveAdd(NormalCombinationMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[
6, 2, 6, 1]");
//--- X軸のプロパティを設定する
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y軸のプロパティを設定する
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- プロット
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

B1（取得メソッド）

1番目のメンバーシップ関数の中心値を取得します。

```
double  B1()
```

戻り値

1番目のメンバーシップ関数の中心値

B1（設定メソッド）

1番目のメンバーシップ関数の中心値を設定します。

```
void  B1(
   const double  b1      // 1番目の中心値
   )
```

パラメータ

b

[in] 1番目のメンバーシップ関数の中心値

B2（取得メソッド）

2番目のメンバーシップ関数の中心値を取得します。

```
double  B2()
```

© 2000-2025, MetaQuotes Ltd.

戻り値

2番目のメンバーシップ関数の中心値

B2（設定メソッド）

2番目のメンバーシップ関数の中心値を設定します。

```
void  B2(
   const double  b2      // 2番目の中心値
   )
```

パラメータ

b2

[in] 2番目のメンバーシップ関数の中心値

Sigma1（取得メソッド）

メンバーシップ関数の曲率の1番目のパラメータを取得します。

```
double  Sigma1()
```

戻り値

メンバーシップ関数の曲率の1番目のパラメータ

Sigma1（設定メソッド）

メンバーシップ関数の曲率の1番目のパラメータを設定します。

```
void  Sigma1(
   const double  sigma1      // 1番目の曲率パラメータの値
   )
```

パラメータ

sigma1

[in] メンバーシップ関数の曲率の1番目のパラメータ

Sigma2（取得メソッド）

メンバーシップ関数の曲率の2番目のパラメータを取得します。

```
double  Sigma2()
```

戻り値

© 2000-2025, MetaQuotes Ltd.

メンバーシップ関数の曲率の2番目のパラメータ

Sigma2（設定メソッド）

メンバーシップ関数の曲率の2番目のパラメータを設定します。

```
void  Sigma2(
   const double  sigma2      // 2番目の曲率パラメータの値
   )
```

パラメータ

sigma2

[in] メンバーシップ関数の曲率の2番目のパラメータ

GetValue

メンバーシップ関数の値を指定された引数で計算します。

```
double  GetValue(
   const  x      // メンバーシップ関数引数
   )
```

パラメータ

x

[in] メンバーシップ関数引数

戻り値

メンバーシップ関数名

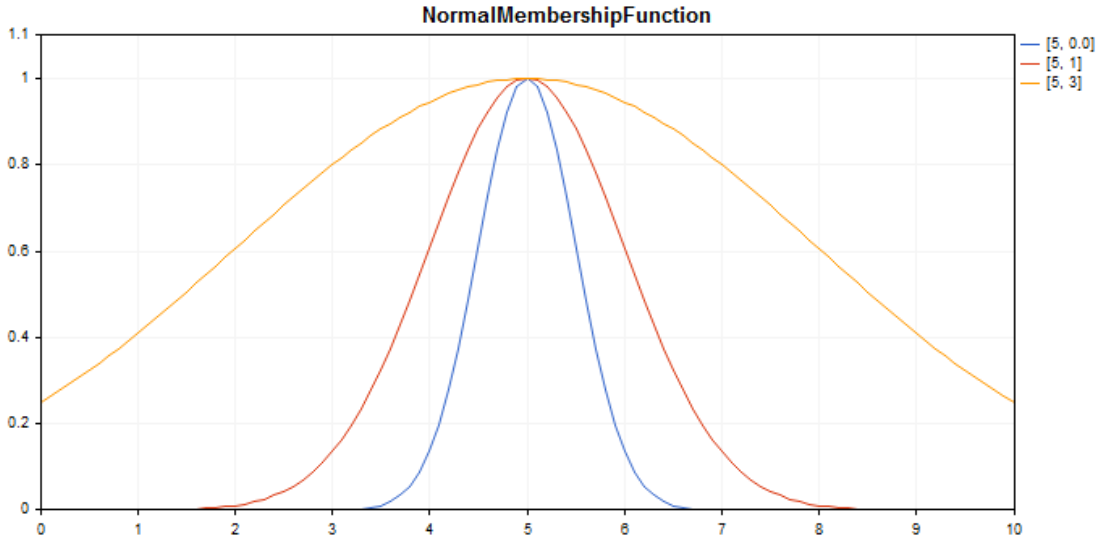
© 2000–2025, MetaQuotes Ltd.

CNormalMembershipFunction

BとSigmaパラメータを持つ対称ガウスメンバーシップ関数を実装するためのクラス。

説明

対称ガウスメンバーシップ関数は、ガウス分布を用いて形成されます。関数は滑らかで、定義領域全体に沿ってゼロ以外の値をとります。



下記はチャートをプロットするための[サンプルコード](#)です。

宣言

```
class CNormalMembershipFuncion : public IMembershipFunction
```

タイトル

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

継承階層

[CObject](#)
  [IMembershipFunction](#)
    CNormalMembershipFunction

クラスメソッド

クラスメソッド	説明
[B](#)	メンバーシップ関数の中心値を取得及び設定します。
[Sigma](#)	メンバーシップ関数の曲率のパラメータを取得及び設定します。

© 2000–2025, MetaQuotes Ltd.

クラスメソッド	説 明
GetValue	メンバーシップ関 数 の値 を指 定 された引 数 で計 算 しま す。

クラスから継 承 されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

例

```
//+------------------------------------------------------------------+
//|                                     NormalMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- メンバーシップ関数を作成します
CNormalMembershipFunction func1(5,0.5);
CNormalMembershipFunction func2(5,1);
CNormalMembershipFunction func3(5,3);
//--- メンバーシップ関数用のラッパーを作成する
double NormalMembershipFunction1(double x) { return(func1.GetValue(x)); }
double NormalMembershipFunction2(double x) { return(func2.GetValue(x)); }
double NormalMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- グラフィックを作成する
   CGraphic graphic;
   if(!graphic.Create(0,"NormalMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"NormalMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("NormalMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- 曲線を作成する
   graphic.CurveAdd(NormalMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[5, 0.0]");
   graphic.CurveAdd(NormalMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[5, 1]");
   graphic.CurveAdd(NormalMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[5, 3]");
//--- X軸のプロパティを設定する
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y軸のプロパティを設定する
```

© 2000–2025, MetaQuotes Ltd.

```
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- プロット
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

B（取得メソッド）

メンバーシップ関数の中心値を取得します。

```
double  B()
```

戻り値

メンバーシップ関数の中心値

B（設定メソッド）

メンバーシップ関数の中心値を設定します。

```
void  B(
   const double  b      // 関数の中心値
   )
```

パラメータ

b

[in] メンバーシップ関数の中心値

Sigma（取得メソッド）

メンバーシップ関数の曲率のパラメータを取得します。

```
double  Sigma()
```

戻り値

メンバーシップ関数の曲率パラメータの値

Sigma（設定メソッド）

メンバーシップ関数の曲率パラメータの値を設定します。

```
void  Sigma(
   const double  sigma      // 曲率パラメータ値
   )
```

パラメータ

sigma

© 2000-2025, MetaQuotes Ltd.

[in] メンバーシップ関数の曲率パラメータの値

GetValue

メンバーシップ関数の値を指定された引数で計算します。

```
double  GetValue(
   const double  x      // 引数
   )
```

パラメータ

x

[in] メンバーシップ関数引数

戻り値

メンバーシップ関数名

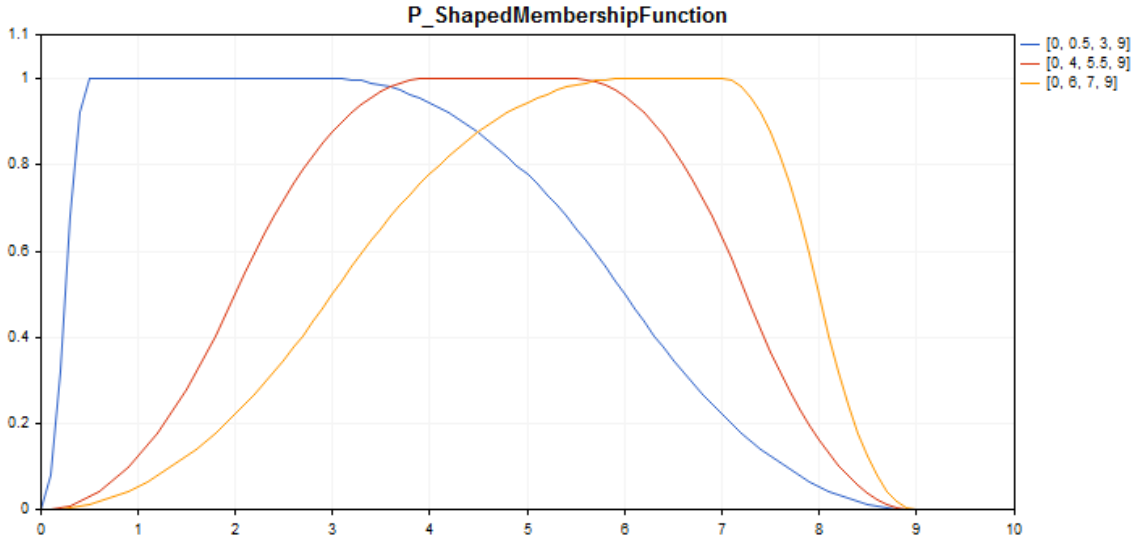
© 2000–2025, MetaQuotes Ltd.

CP_ShapedMembershipFunction

A、B、C、Dパラメータを持つπ型メンバーシップ関数を実装するためのクラス。

説明

π型のメンバーシップ関数は、曲線の台形の形をしています。この関数は、悲観的な数値から楽観的な数値への滑らかなファジー数評価の移行で、非対称メンバーシップ関数を設定するために使用されます。



下記はチャートをプロットするためのサンプルコードです。

宣言

```
class CP_ShapedMembershipFuncion : public IMembershipFunction
```

タイトル

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

継承階層

CObject
  IMembershipFunction
    CP_ShapedMembershipFunction

クラスメソッド

クラスメソッド	説明
A	ファジーセットの先頭のパラメータを取得および設定します。
B	ファジーセットコアの1番目のパラメータを取得および設定します。

© 2000–2025, MetaQuotes Ltd.

クラスメソッド	説 明
C	ファジーセットコアの2番目のパラメータを取得および設定します。<分節
D	ファジーセットの最後のパラメータを取得および設定します。
GetValue	メンバーシップ関数の値を指定された引数で計算します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

例

```
//+------------------------------------------------------------------+
//|                                    P_ShapedMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- メンバーシップ関数を作成します
CP_ShapedMembershipFunction func1(0,0.5,3,9);
CP_ShapedMembershipFunction func2(0,4,5.5,9);
CP_ShapedMembershipFunction func3(0,6,7,9);
//--- メンバーシップ関数用のラッパーを作成する
double P_ShapedMembershipFunction1(double x) { return(func1.GetValue(x)); }
double P_ShapedMembershipFunction2(double x) { return(func2.GetValue(x)); }
double P_ShapedMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- グラフィックを作成する
   CGraphic graphic;
   if(!graphic.Create(0,"P_ShapedMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"P_ShapedMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("P_ShapedMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- 曲線を作成する
   graphic.CurveAdd(P_ShapedMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[0, 0.5,
3, 9]");
   graphic.CurveAdd(P_ShapedMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[0, 4,
5.5, 9]");
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.CurveAdd(P_ShapedMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[0, 6, 7,
9]");
//--- X軸のプロパティを設定する
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y軸のプロパティを設定する
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- プロット
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A（取得メソッド）

ファジーセットの先頭のパラメータを取得します。

```
double  A()
```

戻り値

ファジーセットの先頭のパラメータ

A（設定メソッド）

ファジーセットの先頭のパラメータを設定します。

```
void  A(
   const double  a      // ファジーセットの先頭のパラメータ
   )
```

パラメータ

a

[in] ファジーセットの先頭のパラメータ

B（取得メソッド）

ファジーセットコアの1番目のパラメータを取得します。

```
double  B()
```

戻り値

ファジーセットコアの1番目のパラメータ

B（設定メソッド）

© 2000-2025, MetaQuotes Ltd.

ファジーセットコアの1番目のパラメータを設定します。

```
void  B(
   const double  b      // ファジーセットコアの1番目のパラメータの値
   )
```

パラメータ

b

[in] ファジーセットコアの1番目のパラメータ

C（取得メソッド）

ファジーセットコアの2番目のパラメータを取得します。

```
double  C()
```

戻り値

ファジーセットコアの2番目のパラメータ

C（設定メソッド）

ファジーセットコアの2番目のパラメータを設定します。

```
void  C(
   const double  c      // ファジーセットコアの2番目のパラメータの値
   )
```

パラメータ

c

[in] ファジーセットコアの2番目のパラメータの値

D（取得メソッド）

ファジーセットの最後のパラメータを取得します。

```
double  D()
```

戻り値

ファジーセットの最後のパラメータ

D（設定メソッド）

ファジーセットの最後のパラメータを設定します。

```
void  D(
   const double  d      // ファジーセットの最後のパラメータ
   )
```

パラメータ

d

© 2000-2025, MetaQuotes Ltd.

[in] ファジーセットの最後のパラメータの値

GetValue

メンバーシップ関数の値を指定された引数で計算します。

```
double  GetValue(
   const double  x
   )
```

パラメータ

x

[in] メンバーシップ関数引数

戻り値

メンバーシップ関数値

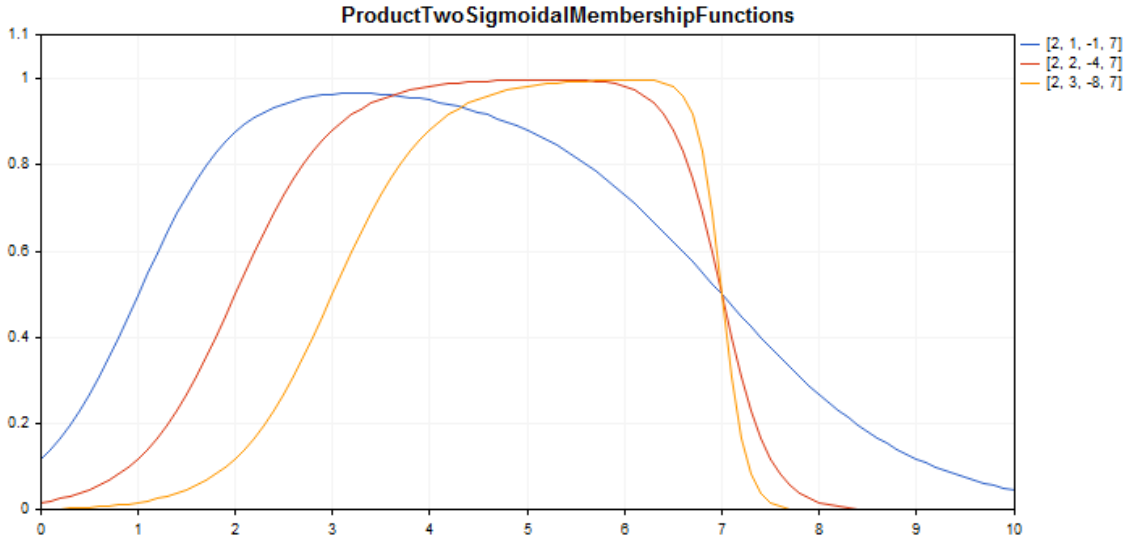
© 2000-2025, MetaQuotes Ltd.

CProductTwoSigmoidalMembershipFunction

A1、A2、C1、C2パラメータを持つ2つのシグモイド関数の積の形でメンバーシップ関数を実装するためのクラス。

説明

平滑な非対称関数を設定するために、2つのシグモイドメンバーシップ関数の積が適用されます。引数値で始まる値が1のメンバーシップ関数の作成が可能になります。このような関数は、「ショート」や「ロング」などの言語用語を設定する必要がある場合に適しています。



下記はチャートをプロットするためのサンプルコードです。

宣言

```
class CProductTwoSigmoidalMembershipFuncion : public IMembershipFunction
```

タイトル

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

継承階層

CObject

IMembershipFunction

CProductTwoSigmoidalMembershipFunctions

クラスメソッド

クラスメソッド	説明
A1	1番目のメンバーシップ関数の勾配比を取得及び設定します。

© 2000-2025, MetaQuotes Ltd.

クラスメソッド	説明
A2	2番目のメンバーシップ関数の勾配比を取得及び設定します。
C1	1番目のメンバーシップ関数の屈曲座標パラメータを取得します。
C2	2番目のメンバーシップ関数の屈曲座標パラメータを取得します。
GetValue	メンバーシップ関数の値を指定された引数で計算します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

例

```
//+------------------------------------------------------------------+
//|                      ProductTwoSigmoidalMembershipFunctions.mq5 |
//|                       Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- メンバーシップ関数を作成します
CProductTwoSigmoidalMembershipFunctions func1(2,1,-1,7);
CProductTwoSigmoidalMembershipFunctions func2(2,2,-4,7);
CProductTwoSigmoidalMembershipFunctions func3(2,3,-8,7);
//--- メンバーシップ関数用のラッパーを作成する
double ProductTwoSigmoidalMembershipFunctions1(double x)
{ return(func1.GetValue(x)); }
double ProductTwoSigmoidalMembershipFunctions2(double x)
{ return(func2.GetValue(x)); }
double ProductTwoSigmoidalMembershipFunctions3(double x)
{ return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//|  スクリプトプログラム開始関数                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- グラフィックを作成する
   CGraphic graphic;
   if(!graphic.Create(0,"ProductTwoSigmoidalMembershipFunctions",0,30,30,780,380))
     {
      graphic.Attach(0,"ProductTwoSigmoidalMembershipFunctions");
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("ProductTwoSigmoidalMembershipFunctions");
   graphic.BackgroundMainSize(16);
//--- 曲線を作成する
   graphic.CurveAdd(ProductTwoSigmoidalMembershipFunctions1,0.0,10.0,0.1,CURVE_LINES
,"[2, 1, -1, 7]");
   graphic.CurveAdd(ProductTwoSigmoidalMembershipFunctions2,0.0,10.0,0.1,CURVE_LINES
,"[2, 2, -4, 7]");
   graphic.CurveAdd(ProductTwoSigmoidalMembershipFunctions3,0.0,10.0,0.1,CURVE_LINES
,"[2, 3, -8, 7]");
//--- X軸のプロパティを設定する
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y軸のプロパティを設定する
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- プロット
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A1（取得メソッド）

1番目のメンバーシップ関数の勾配比を取得します。

```
double  A1()
```

戻り値

1番目のメンバーシップ関数の勾配比

A1（設定メソッド）

1番目のメンバーシップ関数の勾配比を設定します。

```
void  A1(
   const double  a1      // 1番目のメンバーシップ関数の勾配比
   )
```

パラメータ

a1

[in] 1番目のメンバーシップ関数の勾配比

A2（取得メソッド）

2番目のメンバーシップ関数の勾配比を取得します。

© 2000-2025, MetaQuotes Ltd.

```
double  A2()
```

戻り値

2番目のメンバーシップ関数の勾配比

A2（設定メソッド）

2番目のメンバーシップ関数の勾配比を設定します。

```
void  A2(
   const double  a2       // 2番目のメンバーシップ関数の勾配比
   )
```

パラメータ

a2

[in] 2番目のメンバーシップ関数の勾配比

C1（取得メソッド）

1番目のメンバーシップ関数の屈曲座標パラメータを取得します。

```
double  C1()
```

戻り値

1番目のメンバーシップ関数の屈曲座標

C1（設定メソッド）

1番目のメンバーシップ関数の屈曲座標パラメータを設定します。

```
void  C1(
   const double  c1       // 1番目のメンバーシップ関数の屈曲座標
   )
```

パラメータ

c1

[in] 1番目のメンバーシップ関数の屈曲座標

C2（取得メソッド）

2番目のメンバーシップ関数の屈曲座標パラメータを取得します。

```
double  C2()
```

戻り値

2番目のメンバーシップ関数の屈曲座標

C2（設定メソッド）

© 2000–2025, MetaQuotes Ltd.

2番目のメンバーシップ関数の屈曲座標パラメータを設定します。

```
void  C2(
   const double  c2       // 2番目のメンバーシップ関数の屈曲座標
   )
```

パラメータ

c2

[in] 2番目のメンバーシップ関数の屈曲座標

GetValue

メンバーシップ関数の値を指定された引数で計算します。

```
double  GetValue(
   const  x       // メンバーシップ関数引数
   )
```

パラメータ

x

[in] メンバーシップ関数引数

戻り値

メンバーシップ関数名

© 2000–2025, MetaQuotes Ltd.

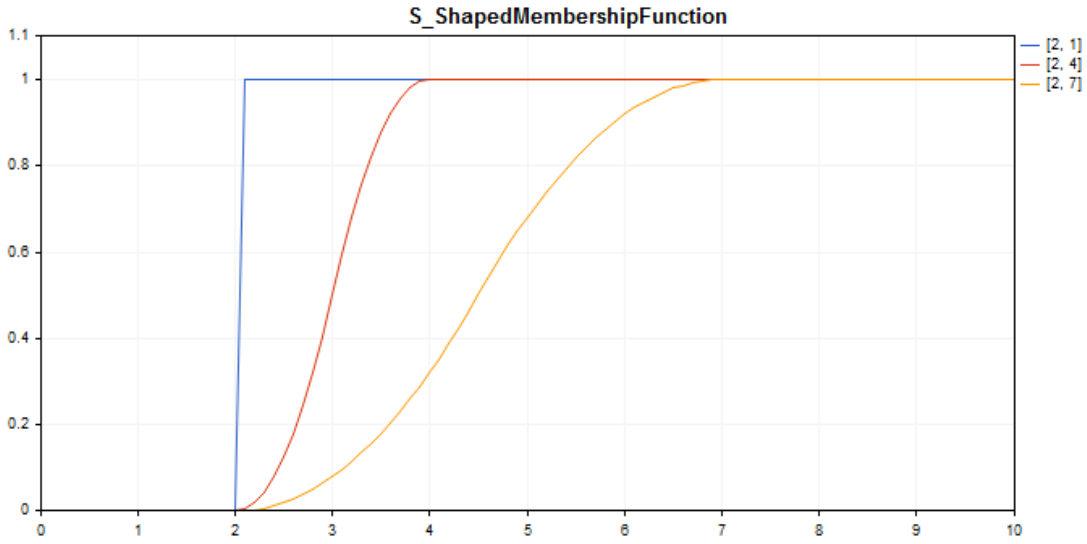
CS_ShapedMembershipFunction

AとBのパラメータを持つS字メンバーシップ関数を実装するためのクラス。

説明

この関数は、2パラメータS字メンバーシップ関数を設定します。これは0から1までの値をとる非減少関数です。Aパラメータと Bパラメータは、関数が非線形軌道で0から1まで増加する間隔を定義します。

関数は、「非常に高い」タイプのファジー集合を表します（すなわち、飽和を伴う非減少メンバーシップ関数が設定されます）。



下記はチャートをプロットするための[サンプルコード](#)です。

宣言

```
class CS_ShapedMembershipFuncion : public IMembershipFunction
```

タイトル

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

継承階層

COBject

IMembershipFunction

CS_ShapedMembershipFunction

クラスメソッド

クラスメソッド	説明
A	増加区間開始のパラメータを取得および設定します。

© 2000–2025, MetaQuotes Ltd.

クラスメソッド	説 明
B	ファジーセットコアの1番目のパラメータを取得および設定します。
GetValue	メンバーシップ関数の値を指定された引数で計算します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

例

```
//+------------------------------------------------------------------+
//|                                  S_ShapedMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- メンバーシップ関数を作成します
CS_ShapedMembershipFunction func1(2,1);
CS_ShapedMembershipFunction func2(2,4);
CS_ShapedMembershipFunction func3(2,7);
//--- メンバーシップ関数用のラッパーを作成する
double S_ShapedMembershipFunction1(double x) { return(func1.GetValue(x)); }
double S_ShapedMembershipFunction2(double x) { return(func2.GetValue(x)); }
double S_ShapedMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- グラフィックを作成する
   CGraphic graphic;
   if(!graphic.Create(0,"S_ShapedMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"S_ShapedMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("S_ShapedMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- 曲線を作成する
   graphic.CurveAdd(S_ShapedMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[2, 1]");
   graphic.CurveAdd(S_ShapedMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[2, 4]");
   graphic.CurveAdd(S_ShapedMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[2, 7]");
//--- X軸のプロパティを設定する
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
```

© 2000–2025, MetaQuotes Ltd.

```
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y軸のプロパティを設定する
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- プロット
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A（取得メソッド）

増加区間開始のパラメータを取得します。

```
double  A()
```

戻り値

増加区間開始のパラメータ

A（設定メソッド）

増加区間開始のパラメータを設定します。

```
void  A(
   const double  a      // 増加区間開始のパラメータ
   )
```

パラメータ

a

[in] 増加区間開始のパラメータ

B（取得メソッド）

ファジーセットコアの1番目のパラメータを取得します。

```
double  B()
```

戻り値

ファジーセットコアの1番目のパラメータ

B（設定メソッド）

ファジーセットコアの1番目のパラメータを設定します。

```
void  B(
   const double  b      // ファジーセットコアの1番目のパラメータ
   )
```

© 2000-2025, MetaQuotes Ltd.

パラメータ

b

[in] ファジーセットコアの1番目のパラメータ

GetValue

メンバーシップ関数の値を指定された引数で計算します。

```
double  GetValue(
   const  x      // メンバーシップ関数引数
   )
```

パラメータ

x

[in] メンバーシップ関数引数

戻り値

メンバーシップ関数名

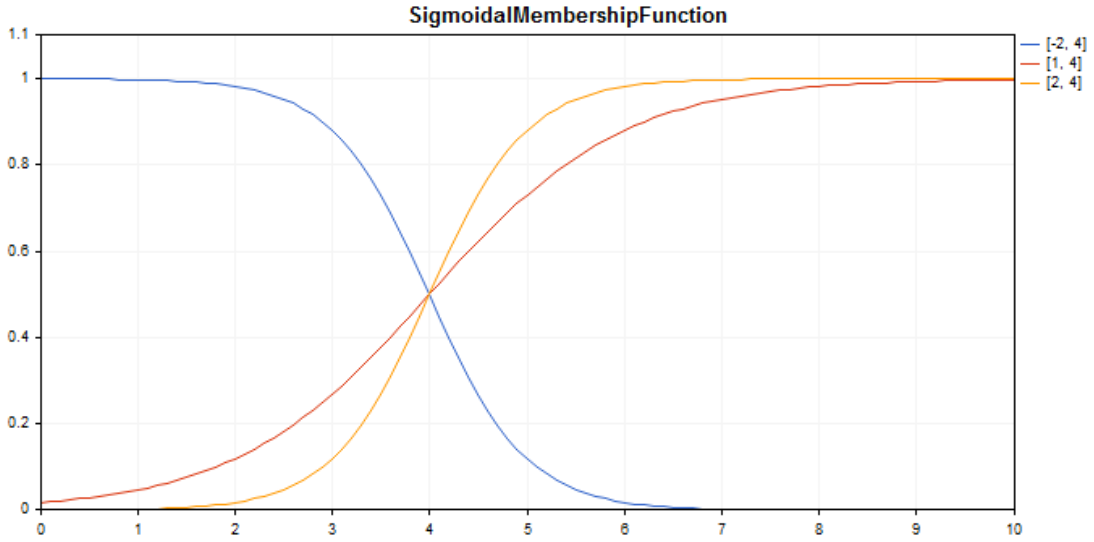
© 2000-2025, MetaQuotes Ltd.

CSigmoidalMembershipFunction

AとCのパラメータを持つシグモイドメンバーシップ関数を実装するためのクラス。

説明

シグモイド関数は、単調なメンバーシップ関数を設定するときに適用されます。引数値で始まる値が1のメンバーシップ関数の作成が可能になります。このような関数は、「ショート」や「ロング」などの言語用語を設定する必要がある場合に適しています。



下記はチャートをプロットするための[サンプルコード](#)です。

宣言

```
class CSigmoidalMembershipFuncion : public IMembershipFunction
```

タイトル

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

継承階層

[CObject](#)
  [IMembershipFunction](#)
    CSigmoidalMembershipFunction

クラスメソッド

クラスメソッド	説明
[A](#)	メンバーシップ関数の勾配比を取得及び設定します。
[C](#)	メンバーシップ関数の屈曲座標パラメータを取得及び設定します。

© 2000–2025, MetaQuotes Ltd.

クラスメソッド	説 明
GetValue	メンバーシップ関 数 の値 を指 定 された引 数 で計 算 します。

クラスから継 承 されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

例

```
//+------------------------------------------------------------------+
//|                                  SigmoidalMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- メンバーシップ関数を作成します
CSigmoidalMembershipFunction func1(-2, 4);
CSigmoidalMembershipFunction func2(1, 4);
CSigmoidalMembershipFunction func3(2, 4);
//--- メンバーシップ関数用のラッパーを作成する
double SigmoidalMembershipFunction1(double x) { return(func1.GetValue(x)); }
double SigmoidalMembershipFunction2(double x) { return(func2.GetValue(x)); }
double SigmoidalMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- グラフィックを作成する
   CGraphic graphic;
   if(!graphic.Create(0,"SigmoidalMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"SigmoidalMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("SigmoidalMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- 曲線を作成する
   graphic.CurveAdd(SigmoidalMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[-2,
4]");
   graphic.CurveAdd(SigmoidalMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[1, 4]");
   graphic.CurveAdd(SigmoidalMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[2, 4]");
//--- X軸のプロパティを設定する
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
```

© 2000–2025, MetaQuotes Ltd.

```
//--- Y軸のプロパティを設定する
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- プロット
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A（取得メソッド）

メンバーシップ関数の勾配比を取得します。

```
double  A()
```

戻り値

メンバーシップ関数の勾配比

A（設定メソッド）

メンバーシップ関数の勾配比を設定します。

```
void  A(
   const double  a      // 1番目のメンバーシップ関数の勾配比
   )
```

パラメータ

a

[in] メンバーシップ関数の勾配比

C（取得メソッド）

メンバーシップ関数の屈曲座標パラメータを取得します。

```
double  C()
```

戻り値

メンバーシップ関数の屈曲座標

C（設定メソッド）

メンバーシップ関数の屈曲座標パラメータを設定します。

```
void  C(
   const double  c      // メンバーシップ関数の屈曲座標
   )
```

パラメータ

c

© 2000–2025, MetaQuotes Ltd.

[in] メンバーシップ関数の屈曲座標

GetValue

メンバーシップ関数の値を指定された引数で計算します。

```
double  GetValue(
   const  x      // メンバーシップ関数引数
   )
```

パラメータ

x

[in] メンバーシップ関数引数

戻り値

メンバーシップ関数名

© 2000–2025, MetaQuotes Ltd.

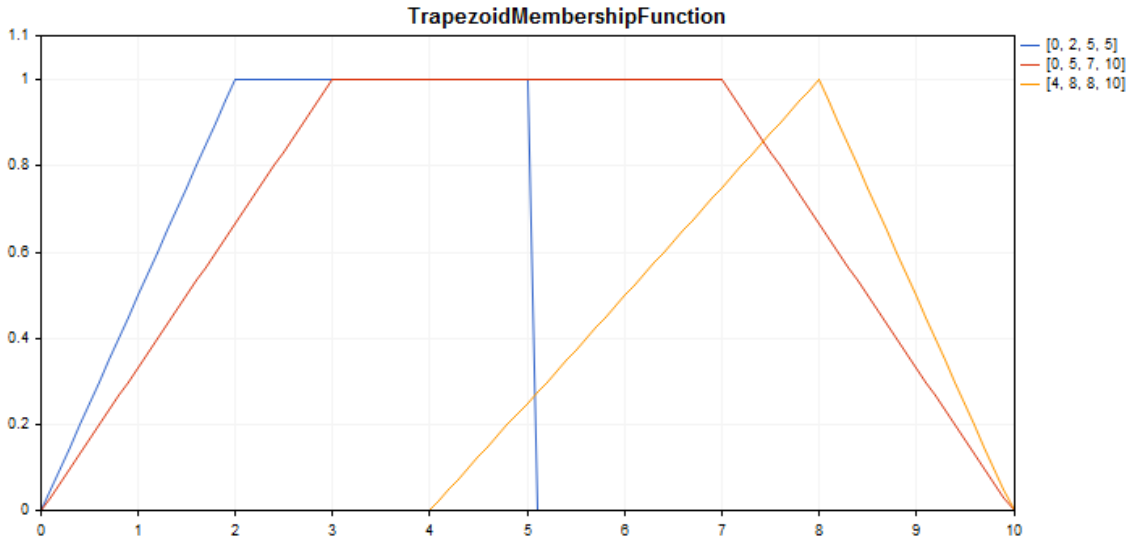
CTrapezoidMembershipFunction

X1、X2、X3、X4パラメータを持つ台形メンバーシップ関数を実装するためのクラス。

説明

この関数は、区分線形近似を用いて形成されます。これは、ファジー集合コアを区間として割り当てることを可能にする三角関数の一般化です。このようなメンバーシップ関数によって、楽観的/悲観的な評価を簡便に解釈することが可能になります。

この関数は、変数の非対称メンバーシップ関数を、特定の間隔内で定義された最も重要な値で設定するために使用されます。



下記はチャートをプロットするための[サンプルコード](#)です。

宣言

```
class CTrapezoidMembershipFuncion : public IMembershipFunction
```

タイトル

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

継承階層

[CObject](#)
  [IMembershipFunction](#)
    CTrapezoidMembershipFunction

クラスメソッド

クラスメソッド	説明
[X1](#)	X軸上の1番目の点の値を取得および設定します。

© 2000-2025, MetaQuotes Ltd.

クラスメソッド	説明
X2	X軸上の2番目の点の値を取得および設定します。
X3	X軸上の3番目の点の値を取得および設定します。
X4	X軸上の4番目の点の値を取得および設定します。
GetValue	メンバーシップ関数の値を指定された引数で計算します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

例

```
//+------------------------------------------------------------------+
//|                                  TrapezoidMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- メンバーシップ関数を作成します
CTrapezoidMembershipFunction func1(0,2,5,5);
CTrapezoidMembershipFunction func2(0,3,7,10);
CTrapezoidMembershipFunction func3(4,8,8,10);
//--- メンバーシップ関数用のラッパーを作成する
double TrapezoidMembershipFunction1(double x) { return(func1.GetValue(x)); }
double TrapezoidMembershipFunction2(double x) { return(func2.GetValue(x)); }
double TrapezoidMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- グラフィックを作成する
   CGraphic graphic;
   if(!graphic.Create(0,"TrapezoidMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"TrapezoidMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("TrapezoidMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- 曲線を作成する
   graphic.CurveAdd(TrapezoidMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[0, 2, 5,
5]");
   graphic.CurveAdd(TrapezoidMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[0, 5, 7,
10]");
```

© 2000–2025, MetaQuotes Ltd.

```
   graphic.CurveAdd(TrapezoidMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[4, 8, 8,
10]");
//--- X軸のプロパティを設定する
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y軸のプロパティを設定する
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- プロット
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

X1（取得メソッド）

X軸上の1番目の点の値を取得します。

```
double  X1()
```

戻り値

X軸上の1番目の点の値

X1（設定メソッド）

X軸上の1番目の点の値を設定します。

```
void  X1(
   const double  x1      // X軸上の1番目の点の値
   )
```

パラメータ

x1

[in] X軸上の1番目の点の値

X2（取得メソッド）

X軸上の2番目の点の値を取得します。

```
double  X2()
```

戻り値

X軸上の2番目の点の値

X2（設定メソッド）

© 2000-2025, MetaQuotes Ltd.

X軸上の2番目の点の値を設定します。

```
void  X2(
   const double  x2      // X軸上の2番目の点の値
   )
```

パラメータ

x2

[in] X軸上の2番目の点の値

X3（取得メソッド）

X軸上の3番目の点の値を取得します。

```
double  X3()
```

戻り値

X軸上の3番目の点の値

X3（設定メソッド）

X軸上の3番目の点の値を設定します。

```
void  X3(
   const double  x3      // X軸上の3番目の点の値
   )
```

パラメータ

x3

[in] X軸上の3番目の点の値

X4（取得メソッド）

X軸上の4番目の点の値を取得します。

```
double  X4()
```

戻り値

X軸上の4番目の点の値

X4（設定メソッド）

X軸上の4番目の点の値を設定します。

```
void  X4(
   const double  x4      // X軸上の4番目の点の値
   )
```

パラメータ

x4

© 2000–2025, MetaQuotes Ltd.

[in] X軸上の4番目の点の値

GetValue

メンバーシップ関数の値を指定された引数で計算します。

```
double  GetValue(
   const  x      // メンバーシップ関数引数
   )
```

パラメータ

x

[in] メンバーシップ関数引数

戻り値

メンバーシップ関数名

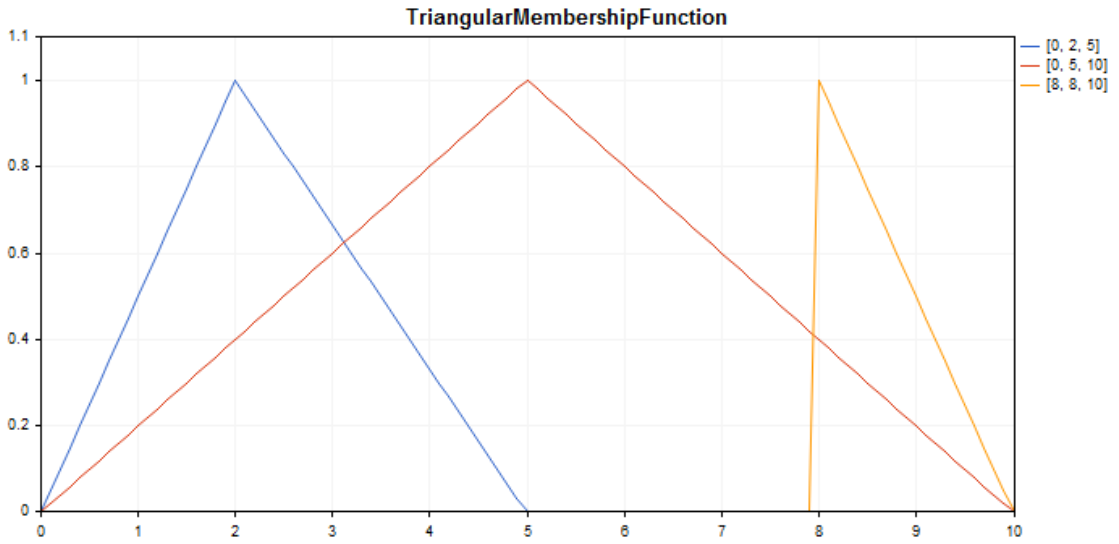
© 2000–2025, MetaQuotes Ltd.

CTriangularMembershipFunction

X1、X2、X3パラメータを持つ三角形メンバーシップ関数を実装するためのクラス。

説明

この関数はメンバーシップ関数を三角形の形で設定します。これは簡単で最も頻繁に適用されるメンバーシップ関数です。



下記はチャートをプロットするための[サンプルコード](#)です。

宣言

```
class CTriangularMembershipFuncion : public IMembershipFunction
```

タイトル

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

継承階層

[CObject](#)
  [IMembershipFunction](#)
    CTriangularMembershipFunction

クラスメソッド

クラスメソッド	説明
[X1](#)	X軸上の1番目の点の値を取得します。
[X2](#)	X軸上の2番目の点の値を取得します。
[X3](#)	X軸上の3番目の点の値を取得します。
[ToNormalMF](#)	三角形メンバーシップ関数をガウス関数に変換します。

© 2000–2025, MetaQuotes Ltd.

クラスメソッド	説 明
GetValue	メンバーシップ関 数 の値 を指 定 された引 数 で計 算 します。

クラスから継 承 されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

例

```
//+------------------------------------------------------------------+
//|                                TriangularMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- メンバーシップ関数を作成します
CTriangularMembershipFunction func1(0,2,5);
CTriangularMembershipFunction func2(0,5,10);
CTriangularMembershipFunction func3(8,8,10);
//--- メンバーシップ関数用のラッパーを作成する
double TriangularMembershipFunction1(double x) { return(func1.GetValue(x)); }
double TriangularMembershipFunction2(double x) { return(func2.GetValue(x)); }
double TriangularMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- グラフィックを作成する
   CGraphic graphic;
   if(!graphic.Create(0,"TriangularMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"TriangularMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("TriangularMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- 曲線を作成する
   graphic.CurveAdd(TriangularMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[0, 2,
5]");
   graphic.CurveAdd(TriangularMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[0, 5,
10]");
   graphic.CurveAdd(TriangularMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[8, 8,
10]");
```

© 2000–2025, MetaQuotes Ltd.

```
//--- X軸のプロパティを設定する
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y軸のプロパティを設定する
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- プロット
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

X1（取得メソッド）

X軸上の1番目の点の値を取得します。

```
double  X1()
```

戻り値

X軸上の1番目の点の値

X1（設定メソッド）

X軸上の1番目の点の値を設定します。

```
void  X1(
   const double  x1      // X軸上の1番目の点の値
   )
```

パラメータ

x1

[in] X軸上の1番目の点の値

X2（取得メソッド）

X軸上の2番目の点の値を取得します。

```
double  X2()
```

戻り値

X軸上の2番目の点の値

X2（設定メソッド）

X軸上の2番目の点の値を設定します。

```
void  X2(
```

© 2000-2025, MetaQuotes Ltd.

```
   const double   x2       // X軸上の2番目の点の値
   )
```

パラメータ

x2

[in] X軸上の2番目の点の値

X3（取得メソッド）

X軸上の3番目の点の値を取得します。

```
double  X3()
```

戻り値

X軸上の3番目の点の値

X3（設定メソッド）

X軸上の3番目の点の値を設定します。

```
void  X3(
   const double   x3       // X軸上の3番目の点の値
   )
```

パラメータ

x3

[in] X軸上の3番目の点の値

ToNormalMF

三角形メンバーシップ関数をガウス関数に変換します。

```
CNormalMembershipFunction*  ToNormalMF()
```

戻り値

ガウスメンバーシップ関数へのポインタ

GetValue

メンバーシップ関数の値を指定された引数で計算します。

```
double  GetValue(
   const  x       // メンバーシップ関数引数
   )
```

パラメータ

x

© 2000-2025, MetaQuotes Ltd.

[in] メンバーシップ関数引数

戻り値

メンバーシップ関数名

© 2000–2025, MetaQuotes Ltd.

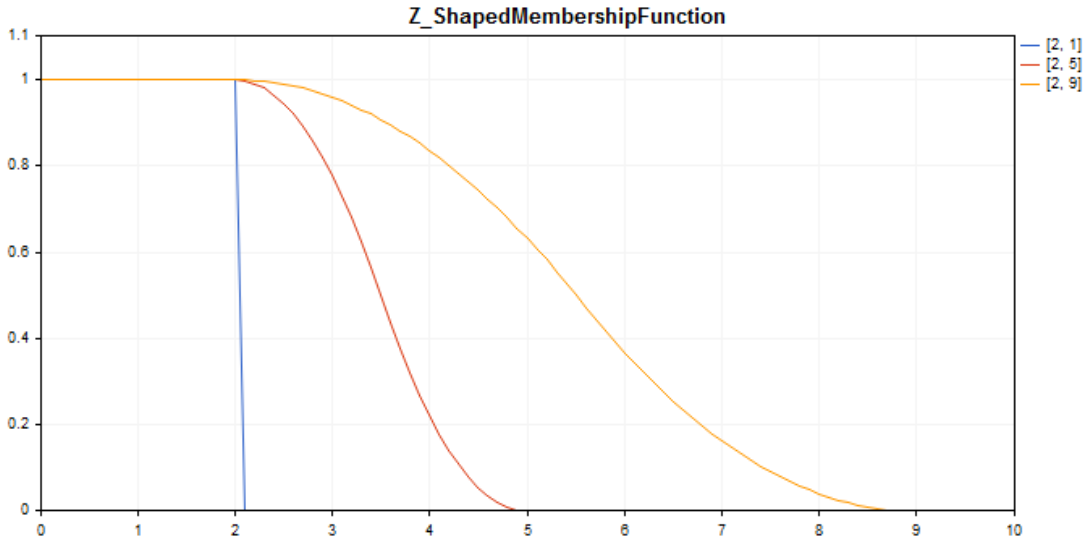
CZ_ShapedMembershipFunction

AとBのパラメータを持つz字メンバーシップ関数を実装するためのクラス。

説明

この関数は、2パラメータz字メンバーシップ関数を設定します。これは1から0までの値をとる非増加メンバーシップ関数です。関数のパラメータは、関数が非線形軌道において1から0まで減少する間隔を定義します。

関数は、「非常に低い」タイプのファジー集合を表します。すなわち、飽和を伴う非増加メンバーシップ関数が設定されます。



下記はチャートをプロットするための[サンプルコード](#)です。

宣言

```
class CZ_ShapedMembershipFuncion : public IMembershipFunction
```

タイトル

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

継承階層

[CObject](#)
[IMembershipFunction](#)
CZ_ShapedMembershipFunction

クラスメソッド

クラスメソッド	説明
[A](#)	減少区間開始のパラメータを取得および設定します。

© 2000–2025, MetaQuotes Ltd.

クラスメソッド	説明
B	減少区間終了のパラメータを取得および設定します。
GetValue	メンバーシップ関数の値を指定された引数で計算します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

例

```
//+------------------------------------------------------------------+
//|                                   Z_ShapedMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- メンバーシップ関数を作成します
CZ_ShapedMembershipFunction func1(2,1);
CZ_ShapedMembershipFunction func2(2,5);
CZ_ShapedMembershipFunction func3(2,9);
//--- メンバーシップ関数用のラッパーを作成する
double Z_ShapedMembershipFunction1(double x) { return(func1.GetValue(x)); }
double Z_ShapedMembershipFunction2(double x) { return(func2.GetValue(x)); }
double Z_ShapedMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- グラフィックを作成する
   CGraphic graphic;
   if(!graphic.Create(0,"Z_ShapedMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"Z_ShapedMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("Z_ShapedMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- 曲線を作成する
   graphic.CurveAdd(Z_ShapedMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[2, 1]");
   graphic.CurveAdd(Z_ShapedMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[2, 5]");
   graphic.CurveAdd(Z_ShapedMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[2, 9]");
//--- X軸のプロパティを設定する
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
```

© 2000–2025, MetaQuotes Ltd.

```
   graphic.XAxis().DefaultStep(1.0);
//--- Y軸のプロパティを設定する
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- プロット
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A（取得メソッド）

減少区間開始のパラメータを取得します。

```
double  A()
```

戻り値

減少区間開始のパラメータ

A（設定メソッド）

減少区間開始のパラメータを設定します。

```
void  A(
   const double  a      // 減少区間開始のパラメータ
   )
```

パラメータ

a

[in] 減少区間開始のパラメータ

B（取得メソッド）

減少区間終了のパラメータを取得します。

```
double  B()
```

戻り値

減少区間終了のパラメータ

B（設定メソッド）

減少区間終了のパラメータを設定します。

```
void  B(
   const double  b      // 減少区間終了のパラメータ
   )
```

パラメータ

© 2000-2025, MetaQuotes Ltd.

b

[in] 減少区間終了のパラメータ

GetValue

メンバーシップ関数の値を指定された引数で計算します。

```
double  GetValue(
   const  x      // メンバーシップ関数引数
   )
```

パラメータ

x

[in] メンバーシップ関数引数

戻り値

メンバーシップ関数名

© 2000–2025, MetaQuotes Ltd.

IMembershipFunction

すべてのメンバーシップ関数の基本クラス

宣言

```
class CZ_ShapedMembershipFuncion : public IMembershipFunction
```

タイトル

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

継承階層

CObject

IMembershipFunction

直接子孫

CCompositeMembershipFunction, CConstantMembershipFunction, CDifferencTwoSigmoidalMembershipFunction, CGeneralizedBellShapedMembershipFunction, CNormalCombinationMembershipFunction, CNormalMembershipFunction, CP_ShapedMembershipFunction, CProductTwoSigmoidalMembershipFunctions, CS_ShapedMembershipFunction, CSigmoidalMembershipFunction, CTrapezoidMembershipFunction, CTriangularMembershipFunction, CZ_ShapedMembershipFunction

クラスメソッド

クラスメソッド	説明
GetValue	メンバーシップ関数の値を指定された引数で計算します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

GetValue

メンバーシップ関数の値を指定された引数で計算します。

```
double  GetValue(
   const  x      // メンバーシップ関数引数
   )
```

パラメータ

x

[in] メンバーシップ関数引数

戻り値

メンバーシップ関数名

© 2000-2025, MetaQuotes Ltd.

ファジーシステムルール

ファジーシステム（ファジー論理推論システム）は、入力の現在の値に対応するファジー集合の形の結論のファジールールとファジー演算を使って受け取りです。

ファジールールは、検査対象の入力と出力の関係を決定します。システム内のルールの数には制限がありません。ファジールールの一般形式は下記の通りです。</t2>

if *ルール条件*、then *ルール結論*

ルール条件　は、オブジェクトの現状態を記述します。*ルール結論*は、この条件がオブジェクトにどのような影響を与えるかを記述します。

ファジーシステムルールのクラス。	説明
CMamdaniFuzzyRule	マムダニアルゴリズムのためのファジー論理ルールを実装するためのクラス。
CSugenoFuzzyRule	菅野アルゴリズムのためのファジー論理ルールを実装するためのクラス。
CSingleCondition	このクラスは「ファジー変数 – ファジー用語」のペアで表されるファジー条件を設定します。
CConditions	このクラスは、演算子によって互いに接続された1組のファジー条件を定義します。
CGenericFuzzyRule	両種類のファジールールを実装するための基本クラス。

© 2000–2025, MetaQuotes Ltd.

CMamdaniFuzzyRule

ファジーシステムの2つの基本型のうちの1つであるマムダニファジー推論。出力変数値は、ファジー用語を使用して設定されます。

説明

Mamdaniアルゴリズムのためのファジー論理ルールは、以下のように記述することができます。

$$if(X_1\ is\ a_1) \wedge (X_2\ is\ a_2) \wedge \ldots \wedge (X_n\ is\ a_n)\ then\ (Y\ is\ d)(W)$$

ここで

- X = (X1, X2, X3 ... Xn) — 入力変数のベクトル
- Y — 出力変数
- a = (a1, a2, a3 ... an) — 入力変数値のベクトル
- d — 出力変数値
- W — ルールの重み

宣言

```
class CMamdaniFuzzyRule : public CGenericFuzzyRule
```

タイトル

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

継承階層

CObject
  IParsableRule
    CGenericFuzzyRule
      CMamdaniFuzzyRule

クラスメソッド

クラスメソッド	説明
Conclusion	マムダニファジールールの結論を取得及び設定します。
Weight	マムダニファジールールの重みを取得及び設定します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

クラスから継承されたメソッド CGenericFuzzyRule

Condition, Condition, CreateCondition, CreateCondition, CreateCondition

Conclusion（取得メソッド）

© 2000–2025, MetaQuotes Ltd.

マムダニファジールールの結論を取得します。

```
CSingleConditon*  Conclusion()
```

戻り値

マムダニファジールールの結論

Conclusion（設定メソッド）

マムダニファジールールの結論を設定します。

```
void  Conclusion(
   CSingleConditon*  value      // マムダニファジールールの結論
   )
```

パラメータ

value

[in] マムダニファジールールの結論

Weight（取得メソッド）

マムダニファジールールの重みを取得します。

```
double  Weight()
```

戻り値

マムダニファジールールの重み

Weight（設定メソッド）

マムダニファジールールの重みを設定します。

```
void  Weight(
   const double  value      // マムダニファジールールの重み
   )
```

パラメータ

value

[in] マムダニファジールールの重みく

© 2000-2025, MetaQuotes Ltd.

CSugenoFuzzyRule

ファジーシステムの2つの基本型のうちの1つである菅野ファジー推論　。出力変数値は、入力変数の線形結合として設定されます。

説明

マムダニルールとは異なり、入力変数値は、ファジー用語ではなくエントリからの一次関数によって設定されます。菅野アルゴリズムのファジー論理ルールは、以下のように記述することができます。

$$if(X_1\ is\ a_1)\wedge(X_2\ is\ a_2)\wedge\ldots\wedge(X_n\ is\ a_n)\ then\ (Y=b_0+b_1\cdot X_1+b_2\cdot X_2+\ldots+b_n\cdot X_n)(W)$$

ここで

- X = (X1, X2, X3 ... Xn) — 入力変数のベクトル
- Y — 出力変数
- a = (a1, a2, a3 ... an) — 入力変数値のベクトル
- b = (b1, b2, b3 ... bn) — free term ratio in the linear function for an output value
- W — ルールの重み

宣言

```
class CSugenoFuzzyRule : public CGenericFuzzyRule
```

タイトル

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

継承階層

CObject
  IParsableRule
    CGenericFuzzyRule
      CSugenoFuzzyRule

クラスメソッド

クラスメソッド	説明
Conclusion	菅野ファジールールの結論を取得します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

クラスから継承されたメソッド CGenericFuzzyRule

Condition, Condition, CreateCondition, CreateCondition, CreateCondition

Conclusion（取得メソッド）

© 2000–2025, MetaQuotes Ltd.

菅野ファジールールの結論を取得します。

```
CSingleConditon*  Conclusion()
```

戻り値

菅野ファジールールの結論

Conclusion（設定メソッド）

菅野ファジールールの結論を設定します。

```
void  Conclusion(
   CSingleCondition*  value      // 菅野ファジールールの結論
   )
```

パラメータ

value

[in] 菅野ファジールールの結論

© 2000-2025, MetaQuotes Ltd.

CSingleCondition

このクラスは「ファジー変数 – ファジー用語」のペアで表されるファジー条件を設定します。

説明

ファジー条件によると、1つの変数は1つの項に対応します。ファジー条件は、次の式で表すことができます。*X is a*,

ここで

- Xはファジー変数です。
- a はファジー変数値（ファジー用語）です。

宣言

```
class CSingleCondition : public ICondition
```

タイトル

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

継承階層

CObject
  ICondition
    CSingleCondition

直接子孫

CFuzzyCondition

クラスメソッド

クラスメソッド	説明
Not	条件に否定を適用する必要があるかどうかを示すフラグを取得及び設定します。
Term	この条件のファジー用語を取得および設定します。
Var	この条件のファジー変数を取得および設定します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Not（取得メソッド）

条件に否定を適用する必要があるかどうかを示すフラグを取得します。

```
bool Not()
```

戻り値

© 2000–2025, MetaQuotes Ltd.

フラグ値

Not（設定メソッド）

条件に否定を適用する必要があるかどうかを示すフラグを設定します。

```
void  Not(
   bool  not      // フラグ値
   )
```

パラメータ

not

[in] フラグ値

Term（取得メソッド）

与えられた条件のファジー用語を取得します。

```
INamedValue* Term()
```

戻り値

与えられた条件のファジー用語

Term（設定メソッド）

与えられた条件のファジー用語を設定します。

```
void  Term(
   INamedValue*&  value      // 与えられた条件のファジー用語
   )
```

パラメータ

value

[in] 与えられた条件のファジー用語

Var（取得メソッド）

与えられた条件のファジー変数を取得します。

```
INamedVariable* Var()
```

戻り値

与えられた条件のファジー変数

Var（設定メソッド）

与えられた条件のファジー変数を設定します。

```
void  Var(
   INamedVariable*&  value      // 与えられた条件のファジー変数
```

© 2000-2025, MetaQuotes Ltd.

```
   )
```

パラメータ

value

[in] ファジー変数

© 2000-2025, MetaQuotes Ltd.

CConditions

このクラスは、演算子によって互いに接続された1組のファジー条件を定義します。

説明

演算子によって互いに接続された1組のファジー条件は、以下のように記述することができます。

$$(X_1 \; is \; a_1) \wedge (X_2 \; is \; a_2) \wedge \ldots \wedge (X_n \; is \; a_n)$$

ここで

- X = (X1, X2, X3 ... Xn) — 入力変数のベクトル
- a = (a1, a2, a3 ... an) — 入力変数の値のベクトル

この例では*and*演算子が使われます。加えて、このクラスでは*or*演算子がも使用できます。

宣言

```
class CConditions : public ICondition
```

タイトル

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

継承階層

CObject
  ICondition
    CConditions

クラスメソッド

クラスメソッド	説明
ConditionsList	全条件のリストを取得します。
Not	条件に否定を適用する必要があるかどうかを示すフラグを取得及び設定します。
Op	条件バンドル演算子の型を取得及び設定します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

ConditionsList

全条件のリストを取得します。

```
CList*  ConditionsList()
```

戻り値

© 2000–2025, MetaQuotes Ltd.

全条件のリスト

Not（取得メソッド）

条件に否定を適用する必要があるかどうかを示すフラグを取得します。

```
bool Not()
```

戻り値

フラグ値

Not（設定メソッド）

条件に否定を適用する必要があるかどうかを示すフラグを設定します。

```
void  Not(
   bool  not      // フラグ値
   )
```

パラメータ

not

[in] フラグ値

Op（取得メソッド）

条件バンドル演算子の型を取得します。*and*及び*or*演算子を使用できます。

```
OperatorType Op()
```

戻り値

条件バンドル演算子の型

Op（設定メソッド）

条件バンドル演算子を設定します。*and*及び*or*演算子を使用できます。

```
void  Op(
   OperatorType  op     // 条件バンドル演算子の型
   )
```

パラメータ

op

[in] 条件バンドル演算子の型

© 2000–2025, MetaQuotes Ltd.

CGenericFuzzyRule

両方の種類のファジールールの基底クラス。

宣言

```
class CGenericFuzzyRule : public IParsableRule
```

タイトル

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

継承階層

COBject

IParsableRule

CGenericFuzzyRule

直接子孫

CMamdaniFuzzyRule, CSugenoFuzzyRule

クラスメソッド

クラスメソッド	説明
Conclusion	ファジールールの結論を取得及び設定します。
Condition	ファジールールの 'if'条件（条件の集合）を取得及び設定します。
CreateCondition	指定されたパラメータでファジールールの条件を作成します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Conclusion（取得メソッド）

ファジールールの結論を取得します。

```
CSingleConditon*  Conclusion()
```

戻り値

ファジールールの結論

Conclusion（設定メソッド）

ファジールールの結論を設定します。

```
virtual void  Conclusion(
   CSingleConditon*  value      // ファジールールの結論
```

© 2000-2025, MetaQuotes Ltd.

```
   )
```

パラメータ

value

[in] ファジールールの結論

Condition（取得メソッド）

ファジールールの 'if'条件（条件の集合）を取得します。

```
CConditons*  Condition()
```

戻り値

ファジールールの 'if'条件（条件の集合）

Condition（設定メソッド）

ファジールールの 'if'条件（条件の集合）を設定します。

```
void  Condition(
   CConditons*  value      // 'ファジールールの 'if' 条件(条件の集合)
   )
```

パラメータ

value

[in] ファジー'条件（条件の集合）.

CreateCondition

指定されたパラメータでファジールールの条件を作成します。

```
CFuzzyCondition*  CreateCondition(
   CFuzzyVariable*  var,      // ファジー変数
   CFuzzyTerm*      term,     // ファジー用語
   )
```

パラメータ

var

[in] ファジー変数

term

[in] ファジー用語

戻り値

ファジールールの状態

CreateCondition

© 2000-2025, MetaQuotes Ltd.

指定されたパラメータでファジールールの条件を作成します。

```
CFuzzyCondition*  CreateCondition(
   CFuzzyVariable*  var,      // ファジー変数
   CFuzzyTerm*      term,     // ファジー用語
   bool             not,      // 条件に否定を適用する必要があるかどうかを示すフラグ
   )
```

パラメータ

var

[in] ファジー変数

term

[in] ファジー用語

not

[in] 条件に否定を適用する必要があるかどうかを示すフラグ

戻り値

ファジールールの状態

CreateCondition

指定されたパラメータでファジールールの条件を作成します。

```
CFuzzyCondition*  CreateCondition(
   CFuzzyVariable* var,      // ファジー変数
   CFuzzyTerm*     term,     // ファジー用語
   bool            not,      // 条件に否定を適用する必要があるかどうかを示すフラグ
   HedgeType       hedge     // 条件バンドルの型
   )
```

パラメータ

var

[in] ファジー変数

term

[in] ファジー用語

not

[in] 条件に否定を適用する必要があるかどうかを示すフラグ

hedge

[in] 条件バンドル型

戻り値

ファジールールの状態

© 2000–2025, MetaQuotes Ltd.

ファジーシステム変数

ファジー(言語)変数はファジーシステムで適応されます。これらは、値が自然言語または人工言語の単語または単語の組み合わせである変数です。

言語変数は、ファジー集合を含みます。ファジー変数の性質と数は、ファジー集合を定義するときに特定のタスクごとに変化します。

クラス	説明
CFuzzyVariable	一般的なファジー変数を作成するためのクラス。
CSugenoVariable	菅野ファジー変数を作成するためのクラス。

© 2000-2025, MetaQuotes Ltd.

CFuzzyVariable

一般的なファジー変数を作成するためのクラス。

説明

ここでは、ファジー変数が以下のパラメータを使用して作成されます。

- 最大可変値
- 最小可変値
- ファジー変数名
- 項集合（言語変数が受け取ることができるすべての可能な値の集合）

宣言

```
class CFuzzyVariable : public CNamedVariableImpl
```

タイトル

```
#include <Math\Fuzzy\fuzzyvariable.mqh>
```

継承階層

COBject
INamedValue
INamedVariable
CNamedVariableImpl
CFuzzyVariable

クラスメソッド

クラスメソッド	説明
AddTerm	ファジー変数に単一のファジー用語を追加します。
GetTermByName	指定された名前のファジー用語を取得します。
Max	ファジー変数の最大値を取得及び設定します。
Min	ファジー変数の最小値を取得及び設定します。
Terms	与えられたファジー変数のファジー用語リストを取得及び設定します。
Values	与えられたファジー変数のファジー値リストを取得及び設定します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

クラスから継承されたメソッド CNamedVariableImpl

Name, Name

© 2000–2025, MetaQuotes Ltd.

AddTerm

ファジー変数に単一のファジー用語を追加します。

```
void  AddTerm(
   CFuzzyTerm*&  term      // ファジー用語
   )
```

パラメータ

term

[in] ファジー用語

GetTermByName

指定された名前のファジー用語を取得します。

```
CFuzzyTerm*  GetTermByName(
   const string  name      // ファジー用語の名前
   )
```

パラメータ

name

[in] ファジー用語の名前

戻り値

指定された名前のファジー用語

Max（取得メソッド）

ファジー変数の最大値を取得します。

```
double  Max()
```

戻り値

ファジー変数の最大値

Max（設定メソッド）

ファジー変数の最大値を設定します。

```
void  Max(
   const double  max      // ファジー変数の最大値
   )
```

パラメータ

max

[in] ファジー変数の最大値

© 2000-2025, MetaQuotes Ltd.

Min（取得メソッド）

ファジー変数の最小値を取得します。

```
double  Min()
```

戻り値

ファジー変数の最小値

Max（設定メソッド）

ファジー変数の最小値を設定します。

```
void  Min(
   const double  min       // ファジー変数の最小値
   )
```

パラメータ

min

[in] ファジー変数の最小値

Terms（取得メソッド）

与えられたファジー変数のファジー用語リストを取得します。

```
CList*  Terms()
```

戻り値

与えられたファジー変数のファジー用語リスト

Terms（設定メソッド）

与えられたファジー変数のファジー用語リストを設定します。

```
void  Terms(
   CList*&  terms       // 与えられたファジー変数のファジー用語リスト
   )
```

パラメータ

terms

[in] 与えられたファジー変数のファジー用語リスト

Values

与えられたファジー変数のファジー用語リストを取得します。

```
CList*  Values()
```

戻り値

© 2000–2025, MetaQuotes Ltd.

与えられたファジー変数のファジー用語リスト

© 2000–2025, MetaQuotes Ltd.

CSugenoVariable

菅野ファジー変数を作成するためのクラス。

説明

菅野ファジー変数は、用語の集合のかわりに一次関数の集合によって設定されるため、一般的な言語変数とは異なります。

宣言

```
class CSugenoVariable : public CNamedVariableImpl
```

タイトル

```
#include <Math\Fuzzy\sugenovariable.mqh>
```

継承階層

CObject
  INamedValue
    INamedVariable
      CNamedVariableImpl
        CSugenoVariable

クラスメソッド

クラスメソッド	説明
Functions	菅野ファジー変数の一次関数のリストを取得します。
GetFuncByName	指定された名前の一次関数を取得します。
Values	菅野ファジー変数の一次関数のリストを取得します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

クラスから継承されたメソッド CNamedVariableImpl

Name, Name

Functions

菅野ファジー変数の一次関数のリストを取得します。

```
CList*  Functions()
```

戻り値

一次関数のリスト

© 2000-2025, MetaQuotes Ltd.

GetFuncByName

指定された名前の一次関数を取得します。

```
ISugenoFunction*  GetFuncByName(
   const string  name      // 一次関数名
   )
```

パラメータ

name

[in] 一次関数の名前

戻り値

指定された名前の一次関数

Values

菅野ファジー変数の一次関数のリストを取得します。

```
CList*  Values()
```

戻り値

菅野ファジー変数の一次関数のリスト

© 2000–2025, MetaQuotes Ltd.

CFuzzyTerm（ファジー用語）

ファジー用語を実行するためのクラス。

説明

項とは項セットの要素です。項は、2つのコンポーネントによって定義されます。

- ファジー用語の名前
- メンバーシップ関数

宣言

```
class CFuzzyTerm : public CNamedValueImpl
```

タイトル

```
#include <Math\Fuzzy\fuzzyterm.mqh>
```

継承階層

CObject
  INamedValue
    CNamedValueImpl
      CFuzzyTerm

クラスメソッド

クラスメソッド	説明
MembershipFunction	ファジー用語のメンバーシップ関数を取得します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

クラスから継承されたメソッド CNamedValueImpl

Name, Name

© 2000–2025, MetaQuotes Ltd.

MembershipFunction

ファジー用語のメンバーシップ関数を取得します。

```
IMembershipFunction*  MembershipFunction()
```

戻り値

メンバーシップ関数

© 2000-2025, MetaQuotes Ltd.

ファジーシステム

ファジーシステム(*ファジーモデル*)とは、その計算がファジー論理に基づいている数学的モデルです。このようなモデルの構築は、研究対象に弱い形式性があり、厳密に計算しようとすると複雑になりすぎてしまう場合に有効です。

モデル構築のプロセスは、次の3つの段階に分けることができます。

1. モデルの入出力特性の定義
2. 知識ベースの構築
3. ファジー推論の方法を選択(マムダニまたは菅野)

第一段階の結果は直接2に影響を与え、モデルの将来を決定します。

知識ベース(*ルールベース*)は、入力と出力の関係性を定義する"if, then"ファジールールタイプのセットです。

ルール条件は現在のオブジェクトの状態を記述し、**ルール結論**はこの条件がオブジェクトにどのような影響を与えるかを記述します。

各ルールの用語や結論には2種類が存在できます。

1. シンプル(Csinglcondへのリンク) — 1ファジー変数を含む
2. コンプレックス(Cconditionsへのリンク) — 複数のファジー変数を含む

システム内の各ルールは、モデル内のルールの重要性を定義する**重み**を有します。重み係数は、[0,1]範囲内のルールに割り当てられています。

モデルのファジー推論システムは作成された知識ベースに応じて決定されます。**ファジー論理推論**は、知識ベースとファジー演算を用いた、入力の現在値に応じたファジー集合の形態です。ファジー推論の2つの主要な種類はマムダニ法と菅野法です。

© 2000-2025, MetaQuotes Ltd.

マムダニシステム

マムダニシステムの出力変数値は、ファジー用語を使用して設定されます。

説明

Mamdaniアルゴリズムのためのファジー論理ルールは、以下のように記述することができます。

$$if(X_1\ is\ a_1) \wedge (X_2\ is\ a_2) \wedge \ldots \wedge (X_n\ is\ a_n)\ then\ (Y\ is\ d)(W)$$

ここで

- X = (X1, X2, X3 ... Xn) — 入力変数のベクトル
- Y — 出力変数
- a = (a1, a2, a3 ... an) — 入力変数値のベクトル
- d — 出力変数値
- W — ルールの重み

クラスメソッド

クラスメソッド	説明
AggregationMethod	条件集約型を設定します
Calculate	システムのファジー推論を計算します。
DefuzzificationMethod	非ファジー化メソッドの型を設定します。
EmptyRule	現在のシステムに基づいて空のマムダニファジールールを作成します。
ImplicationMethod	システム含意演算子の型を設定します。
Output	マムダニファジー出力変数のリストを取得します。
OutputByName	指定された名前のマムダニファジー出力変数を取得します。
ParseRule	特定の行に基づいてマムダニファジールールを作成します。
Rules	マムダニファジールールのリストを返します。

クラスから継承されたメソッド CGenericFuzzySystem

Input, AndMethod, AndMethod, OrMethod, OrMethod, InputByName, Fuzzify

AggregationMethod

条件集約メソッドの型を設定します。

```
void  AggregationMethod(
   AggregationMethod  value      // 集約メソッドの型
```

© 2000–2025, MetaQuotes Ltd.

```
   )
```

パラメータ

value

[in] 条件集約メソッドの型

Calculate

システムのファジー推論を計算します。

```
CList*  Calculate(
   CList*  inputValues      // 入力データ
   )
```

パラメータ

inputValues

[in] 計算のための入力データ

戻り値

計算結果

DefuzzificationMethod

非ファジー化メソッドの型を設定します。

```
void  DefuzzificationMethod(
   DefuzzificationMethod  value      // 非ファジー化メソッドの型
   )
```

パラメータ

value

[in] 非ファジー化メソッドの型

EmptyRule

現在のシステムに基づいて空のマムダニファジールールを作成します。

```
CMamdaniFuzzyRule*  EmptyRule()
```

戻り値

マムダニファジールール

ImplicationMethod

システム含意演算子の型を設定します。

```
void  ImplicationMethod(
   ImplicationMethod  value      // 含意演算子の型
```

© 2000-2025, MetaQuotes Ltd.

```
   )
```

パラメータ

value

[in] 含意演算子の型

Output

マムダニファジー出力変数のリストを取得します。

```
CList*  Output()
```

戻り値

ファジー変数のリスト

OutputByName

指定された名前のマムダニファジー出力変数を取得します。

```
CFuzzyVariable*  OutputByName(
   const string  name      // ファジー変数の名前
   )
```

パラメータ

name

[in] ファジー変数の名前

戻り値

指定された名前のマムダニファジー出力変数

ParseRule

特定の行に基づいてマムダニファジールールを作成します。

```
CMamdaniFuzzyRule*  ParseRule(
   const string  rule      // ファジールールの文字列表現
   )
```

パラメータ

rule

[in] マムダニファジールールの文字列表現

戻り値

マムダニファジールール

Rules

© 2000-2025, MetaQuotes Ltd.

マムダニファジールールのリストを返します。

```
CList*  Rules()
```

戻り値

マムダニファジールールのリスト

© 2000–2025, MetaQuotes Ltd.

CSugenoFuzzyRule

菅野ファジー論理システムは、2つの基本的なファジーシステムの1つです。出力変数値は、入力変数の線形結合として設定されます。

説明

マムダニルールとは異なり、入力変数値は、ファジー用語ではなくエントリからの一次関数によって設定されます。菅野アルゴリズムのファジー論理ルールは、以下のように記述することができます。

$$if(X_1\ is\ a_1)\wedge(X_2\ is\ a_2)\wedge\ldots\wedge(X_n\ is\ a_n)\ then\ (Y=b_0+b_1\cdot X_1+b_2\cdot X_2+\ldots+b_n\cdot X_n)(W)$$

ここで

- X = (X1, X2, X3 ... Xn) — 入力変数のベクトル
- Y — 出力変数
- a = (a1, a2, a3 ... an) — 入力変数値のベクトル
- b = (b1, b2, b3 ... bn) — free term ratio in the linear function for an output value
- W — ルールの重み

クラスメソッド

クラスメソッド	説明
Calculate	システムのファジー推論を計算します。
CreateSugenoFunction	菅野システムの一次関数を作成します。
EmptyRule	現在のシステムに基づいて空の菅野ファジールールを作成します。
Output	菅野ファジー出力変数のリストを取得します。
OutputByName	指定された名前の菅野ファジー出力変数を取得します。
ParseRule	特定の行に基づいて菅野ファジールールを作成します。
Rules	ファジールールのリストを返します。

クラスから継承されたメソッド CGenericFuzzySystem

Input, AndMethod, AndMethod, OrMethod, OrMethod, InputByName, Fuzzify

Calculate

システムのファジー推論を計算します。

```
CList*  Calculate(
   CList*&  inputValues      // 入力データ
   )
```

パラメータ

© 2000–2025, MetaQuotes Ltd.

inputValues

[in] 計算のための入力データ

戻り値

計算結果

CreateSugenoFunction

菅野システムの一次関数を作成します。

```
CLinearSugenoFunction*  CreateSugenoFunction(
   const string  name,           // 関数名
   const double& coeffs[]        // 関数比
   )
```

パラメータ

name

[in] 関数名

coeffs[]

[in] 関数の比

戻り値

菅野一次関数

注意事項

比率配列のサイズの数は入力の数に等しいか1を足したものです。菅野一次関数の定数項は、前者ではゼロに等しく、後者の場合は最後の比率に等しくなります。

CreateSugenoFunction

菅野システムの一次関数を作成します。

```
CLinearSugenoFunction*  CreateSugenoFunction(
   const string  name,           // 関数名
   CList*&       coeffs,         // ファジー変数のペアのリスト - その比率
   const double  constValue      // 関数の定数項の比
   )
```

パラメータ

name

[in] 関数名

coeffs[]

[in] 関数の比

戻り値

菅野一次関数

© 2000-2025, MetaQuotes Ltd.

EmptyRule

現在のシステムに基づいて空の菅野ファジールールを作成します。

```
CSugenoFuzzyRule*  EmptyRule()
```

戻り値

菅野ファジールール

Output

菅野ファジー出力変数のリストを取得します。

```
CList*  Output()
```

戻り値

ファジー変数のリスト

OutputByName

指定された名前の菅野ファジー出力変数を取得します。

```
CSugenoVariable*  OutputByName(
   const string  name      // ファジー変数の名前
   )
```

パラメータ

name

ファジー変数名

戻り値

指定された名前の菅野ファジー変数

ParseRule

特定の行に基づいて菅野ファジールールを作成します。

```
CSugenoFuzzyRule*  ParseRule(
   const string  rule      // 菅野ファジールールの文字列表現
   )
```

パラメータ

rule

[in] 菅野ファジールールの文字列表現

戻り値

菅野ファジールール

© 2000-2025, MetaQuotes Ltd.

Rules

ファジールールのリストを返します。

```
CList*  Rules()
```

戻り値

ファジールールのリスト

© 2000–2025, MetaQuotes Ltd.

OpenCLプログラムを使うためのクラス

COpenCLクラスはOpenCL関数の使用を簡易化するためのラッパーです。場合によっては、計算速度はGPUの使用によって大幅に向上されます。

float値とdouble値に基づく計算のクラス使用例は、MQL5¥Scripts¥Examples¥OpenCL¥フォルダーの対応するサブディレクトリーにあります。OpenCLプログラムのソースコードは、MQL5¥Scripts¥Examples¥OpenCL¥Double¥KernelsおよびMQL5¥Scripts¥Examples¥OpenCL¥Float¥Kernelsサブディレクトリにあります。

- MatrixMult.mq5 – グローバルとローカルメモリを使用した行列乗算の例
- BitonicSort.mq5 – GPUにおける配列要素の並列ソートの例
- FFT.mq5 – 高速フーリエ変換計算の例
- Wavelet.mq5 – モーレットウェーブレットを用いたデータのウェーブレット変換の例

OpenCL用のソースコードは別のCLファイルに書き込むことをお勧めします。これらのファイルは後にリソース変数を使用してMQL5プログラムに組み込むことができます。

宣言

```
class COpenCL
```

タイトル

```
#include <OpenCL\OpenCL.mqh>
```

クラスメソッド

名称	説明
BufferCreate	指定されたインデックスでOpenCLバッファを作成します
BufferFree	指定されたインデックスでOpenCLバッファを削除します
BufferFromArray	指定されたインデックスで値の配列からバッファを作成します
BufferRead	指定されたインデックスでOpenCLバッファを配列に読み込みます
BufferWrite	指定されたインデックスで値の配列をバッファに書き出します
Execute	指定されたインデックスを持つOpenCLカーネルを実行します
GetContext	OpenCLコンテキストのハンドルを返します
GetKernel	指定されたインデックスにあるカーネルオブジェクトのハンドルを返します
GetKernelName	指定されたインデックスにあるカーネルオブジェクトの名前を返します
GetProgram	OpenCLプログラムのハンドルを返します
Initialize	OpenCLプログラムを初期化します
KernelCreate	指定されたインデックスでのOpenCLプログラムへのエントリポイントを作成します
KernelFree	指定されたインデックスでのOpenCL開始関数を削除します

© 2000–2025, MetaQuotes Ltd.

名 称	説 明
SetArgument	指 定 されたインデックスにあるOpenCL関 数 のパラメータを設 定 します
SetArgumentBuffer	指 定 されたインデックスにあるOpenCL関 数 のパラメータとしてOpenCL バッファを設 定 します
SetArgumentLocalMemory	指 定 されたインデックスにあるOpenCL関 数 のパラメータをローカルメモリ で設 定 します
SetBuffersCount	バッファ数 を設 定 します
SetKernelsCount	カーネルオブジェクト の数 を設 定 します
Shutdown	OpenCLプログラムをアンロードします
SupportDouble	浮 動 小 数 点 データ型 がデバイスでサポート されているかどうかをチェック します

© 2000–2025, MetaQuotes Ltd.

BufferCreate

指定されたインデックスでOpenCLバッファを作成します。

```
bool  BufferCreate(
   const int   buffer_index,      // バッファインデックス
   const uint  size_in_bytes,     // バイト単位でのバッファサイズ
   const uint  flags              // バッファプロパティを定義するフラグの組み合わせ
   );
```

パラメータ

buffer_index

[in] インデックスバッファ

size_in_bytes

[in] バイト単位でのバッファサイズ

フラグ

[in] フラグの組み合わせで指定されるバッファプロパティ

戻り値

実行成功の場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

BufferFree

指定されたインデックスでOpenCLバッファを削除します。

```
bool  BufferFree(
   const int  buffer_index      // バッファインデックス
   );
```

パラメータ

buffer_index

[in] インデックスバッファ

戻り値

実行成功の場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

BufferFromArray

指定されたインデックスで値の配列からバッファを作成します。

```
template<typename T>
bool  BufferFromArray(
   const int   buffer_index,          // バッファインデックス
   T           &data[],               // 値の配列
   const uint  data_array_offset,     // 値の配列のオフセット（バイト単位）
   const uint  data_array_count,      // 書き込む配列からの値の数
   const uint  flags                  // バッファのプロパティを定義するフラグの組み合わせ
   );
```

パラメータ

buffer_index

[in] インデックスバッファ

&data[]

[in] OpenCLバッファに書き込まれる値の配列

data_array_offset

[in] 値の書き込みが始まる配列の値のオフセット（バイト単位）

data_array_count

[in] 書き込まれる値の数

フラグ

[in] フラグの組み合わせで指定されるバッファプロパティ

戻り値

実行成功の場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

BufferRead

指定されたインデックスでOpenCLバッファを配列に読み込みます。

```
template<typename T>
bool  BufferRead(
   const int   buffer_index,          // バッファインデックス
   T           &data[],               // 値の配列
   const uint  cl_buffer_offset,      // OpenCL バッファ内のバイト単位のオフセット
   const uint  data_array_offset,     // 配列要素のシフト
   const uint  data_array_count       // バッファから読み込む値の数
   );
```

パラメータ

buffer_index

[in] インデックスバッファ

&data[]

[in] OpenCLバッファの値を取得するのに使用する配列

cl_buffer_offset

[in] OpenCL バッファでの読み込みが始まるオフセット（バイト単位）

data_array_offset

[in] OpenCLのバッファの値を書き込むための最初の配列要素のインデックス

data_array_count

[in] 読まれる値の数

戻り値

実行成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

BufferWrite

指定されたインデックスで値の配列をバッファに書き出します。

```
template<typename T>
bool  BufferWrite(
   const int   buffer_index,          // バッファインデックス
   T           &data[],               // 値の配列
   const uint  cl_buffer_offset,      // OpenCL バッファ内のバイト単位のオフセット
   const uint  data_array_offset,     // 配列要素のシフト
   const uint  data_array_count       // 書き込む配列からの値の数
   );
```

パラメータ

buffer_index

[in] インデックスバッファ

&data[]

[in] OpenCLバッファに書き込まれる値の配列

cl_buffer_offset

[in] OpenCL バッファでの書き出しが始まるオフセット（バイト単位）

data_array_offset

[in] OpenCLのバッファの値を書き込むための最初の配列要素のインデックス

data_array_count

[in] 書き込まれる値の数

戻り値

実行成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Execute

指定されたインデックスを持つOpenCLカーネルを実行します。

```
bool  Execute(
   const int   kernel_index,          // カーネルのインデックス
   const int   work_dim,              // 作業空間の寸法
   const uint  &work_offset[],        // 作業空間の初期オフセット
   const uint  &work_size[]           // 作業の数
   );
```

ローカルグループ内の指定されたインデックスとタスク数でOpenCLカーネルを実行します。

```
bool  Execute(
   const int   kernel_index,          // カーネルのインデックス
   const int   work_dim,              // 作業空間の寸法
   const uint  &work_offset[],        // 作業空間の初期オフセット
   const uint  &work_size[],          // 作業の数
   const uint  &local_work_size[]     // ローカルグループの作業数
   );
```

パラメータ

kernel_index

[in] カーネルオブジェクトのインデックス

work_dim

[in] 作業空間の寸法

&work_offset[]

[in][out] 作業空間の初期オフセット参照で引き渡されます。

&work_size[]

[in] 作業サブセットのサイズ参照で引き渡されます。

&local_work_size[]

[in][out] 作業サブセットのサイズ参照で引き渡されます。

戻り値

実行成功の場合はtrue、それ以外の場合はfalse

© 2000-2025, MetaQuotes Ltd.

GetContext

OpenCL コンテキストのハンドルを返します。

```
int  GetContext();
```

戻り値

OpenCL コンテキストのハンドル

© 2000–2025, MetaQuotes Ltd.

GetKernel

指定されたインデックスにあるカーネルオブジェクトのハンドルを返します。

```
int  GetKernel(
   const int  kernel_index      // カーネルのインデックス
   );
```

パラメータ

kernel_index

[in] カーネルオブジェクトのインデックス

戻り値

カーネルオブジェクトのハンドル

© 2000-2025, MetaQuotes Ltd.

GetKernelName

指定されたインデックスにあるカーネルオブジェクトの名前を返します。

```
string  GetKernelName(
   const int  kernel_index      // カーネルのインデックス
   );
```

パラメータ

kernel_index

[in] カーネルオブジェクトのインデックス

戻り値

カーネルオブジェクトの名前

© 2000-2025, MetaQuotes Ltd.

GetProgram

OpenCLプログラムのハンドルを返します。

```
int  GetProgram();
```

戻り値

OpenCLプログラムのハンドル

© 2000-2025, MetaQuotes Ltd.

Initialize

OpenCLプログラムを初期化します。

```
bool  Initialize(
   const string  program,           // OpenCLプログラムのハンドル
   const bool    show_log=true      // ログを維持する
   );
```

パラメータ

program

[in] OpenCLプログラムのハンドル

show_log=true

[in] メッセージログの維持を有効にする

戻り値

初期化が成功したらtrue、その他の場合はfalse

© 2000-2025, MetaQuotes Ltd.

KernelCreate

指定されたインデックスでのOpenCLプログラムへのエントリポイントを作成します。

```
bool  KernelCreate(
   const int      kernel_index,      // カーネルのインデックス
   const string   kernel_name        // カーネルの名前
   );
```

パラメータ

kernel_index

[in] カーネルオブジェクトのインデックス

kernel_name

[in] カーネルオブジェクトの名前

戻り値

実行成功の場合はtrue、それ以外の場合はfalse

© 2000-2025, MetaQuotes Ltd.

KernelFree

指定されたインデックスでのOpenCL開始関数を削除します。

```
bool  KernelFree(
   const int  kernel_index      // カーネルのインデックス
   );
```

パラメータ

kernel_index

[in] カーネルオブジェクトのインデックス

戻り値

実行成功の場合はtrue、それ以外の場合はfalse

© 2000-2025, MetaQuotes Ltd.

SetArgument

指定されたインデックスにあるOpenCL関数のパラメータを設定します。

```
template<typename T>
bool  SetArgument(
   const int  kernel_index,      // カーネルのインデックス
   const int  arg_index,         // 関数の引数のインデックス
   T          value              // ソースコード
   );
```

パラメータ

kernel_index

[in] カーネルオブジェクトのインデックス

arg_index

[in] 関数の引数のインデックス

value

[in] 関数の引数の値

戻り値

実行成功の場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

SetArgumentBuffer

指定されたインデックスにあるOpenCL関数のパラメータとしてOpenCLバッファを設定します。

```
bool  SetArgumentBuffer(
   const int  kernel_index,      // カーネルのインデックス
   const int  arg_index,         // 関数の引数のインデックス
   const int  buffer_index       // バッファインデックス
   );
```

パラメータ

kernel_index

[in] カーネルオブジェクトのインデックス

arg_index

[in] 関数の引数のインデックス

buffer_index

[in] インデックスバッファ

戻り値

実行成功の場合はtrue、それ以外の場合はfalse

© 2000-2025, MetaQuotes Ltd.

SetArgumentLocalMemory

指定されたインデックスにあるOpenCL関数のパラメータをローカルメモリで設定します。

```
bool  SetArgumentLocalMemory(
   const int  kernel_index,          // カーネルのインデックス
   const int  arg_index,             // 関数の引数のインデックス
   const int  local_memory_size      // ローカルメモリのサイズ
   );
```

パラメータ

kernel_index

[in] カーネルオブジェクトのインデックス

arg_index

[in] 関数の引数のインデックス

local_memory_size

[in] ローカルメモリサイズ

戻り値

実行成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

SetBuffersCount

バッファ数を設定します。

```
bool  SetBuffersCount(
   const int  total_buffers      // バッファ数
   );
```

パラメータ

total_buffers

[in] バッファの総数

戻り値

実行成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

SetKernelsCount

カーネルオブジェクトの数を設定します。

```
bool  SetKernelsCount(
   const int  total_kernels      // カーネル数
   );
```

パラメータ

total_kernels

[in] カーネルの総数

戻り値

実行成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Shutdown

OpenCLプログラムをアンロードします。

```
void  Shutdown();
```

戻り値

なし

© 2000-2025, MetaQuotes Ltd.

SupportDouble

浮動小数点データ型がデバイスでサポートされているかどうかをチェックします。

```
bool  SupportDouble();
```

戻り値

デバイスが浮動小数点データ型をサポートする場合は true を返します。

© 2000-2025, MetaQuotes Ltd.

CObject 基本クラス

CObject は MQL5 標準ライブラリを構成する基本クラスです。

説明

CObject クラスは継承クラスの全てをリンクリストでつなぎます。また、継承クラスが更に実装するための仮想メソッドの数を示します。

宣言

```
class CObject
```

タイトル

```
#include <Object.mqh>
```

継承階層

CObject

直接子孫

CAccountInfo, CArray, CChart, CChartObject, CCurve, CDealInfo, CDictionary_Obj_Double, CDictionary_Obj_Obj, CDictionary_String_Obj, CExpertBase, CFile, CHistoryOrderInfo, CList, COrderInfo, CPositionInfo, CString, CSymbolInfo, CTerminalInfo, CTrade, CTreeNode, CWnd, ICondition, IExpression, IMembershipFunction, INamedValue, IParsableRule

クラスメソッド

属性	
Prev	前の項目の値を取得します。
Prev	前の項目の値を設定します。
Next	次の項目の値を取得します。
Next	次の項目の値を設定します。
比較メソッド	
virtual Compare	別のオブジェクトとの比較の結果を返します。
入出力	
virtual Save	オブジェクトをファイルに書き込みます。
virtual Load	オブジェクトをファイルから読み込みます。
virtual Type	オブジェクトの型を返します。

© 2000–2025, MetaQuotes Ltd.

Prev

リストの前の要素へのポインタを取得します。

```
CObject*  Prev()
```

戻り値

リストの前の要素へのポインタ。最初の要素の場合は NULL。

例：

```
//--- CObject::Prev() の例
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Object create error");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Object create error");
      delete object_first;
      return;
     }
   //--- リンクを設定
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- 前のオブジェクトを使用する
   CObject *object=object_second.Prev();
   //--- オブジェクトを削除する
   delete object_first;
   delete object_second;
  }
```

© 2000–2025, MetaQuotes Ltd.

Prev

リストの前の要素へのポインタを設定します。

```
void  Prev(
   CObject*  object      // リストの前の要素へのポインタ
  )
```

パラメータ

object

[in] リストの前の要素へのポインタの新しい値

例:

```
//--- CObject::Prev(CObject*) の例
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Object create error");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Object create error");
      delete object_first;
      return;
     }
   //--- リンクを設定
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- オブジェクトを使用する
   //--- ...
   //--- オブジェクトを削除する
   delete object_first;
   delete object_second;
  }
```

© 2000-2025, MetaQuotes Ltd.

Next

リストの次の要素へのポインタを取得します。

```
CObject*  Next()
```

戻り値

リストの次の要素へのポインタ。最後の要素の場合は NULL。

例：

```
//--- CObject::Next() の例
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Object create error");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Object create error");
      delete object_first;
      return;
     }
   //--- リンクを設定
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- 次のオブジェクトを使用する
   CObject *object=object_first.Next();
   //--- オブジェクトを削除する
   delete object_first;
   delete object_second;
  }
```

© 2000-2025, MetaQuotes Ltd.

Next

リストの次の要素へのポインタを設定します。

```
void  Next(
   CObject*  object       // リストの次の要素へのポインタ
  )
```

パラメータ

object

[in] リストの次の要素へのポインタの新しい値

例:

```
//--- CObject::Next(CObject*) の例
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Object create error");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Object create error");
      delete object_first;
      return;
     }
   //--- リンクを設定
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- オブジェクトを使用する
   //--- ...
   //--- オブジェクトを削除する
   delete object_first;
   delete object_second;
  }
```

© 2000-2025, MetaQuotes Ltd.

Compare

リスト内のデータ要素を他の要素と比較します。

```
virtual int  Compare(
   const CObject*  node,      // Node to compare with
   const int       mode=0     // Compare mode
   ) const
```

パラメータ

node

[in] 比較する要素へのポインタ

mode=0

[in] 比較するもの

戻り値

両方の要素が等しい場合は 0、1 番目が 2 番目より小さい場合は -1 、1 番目が 2 番目より大きい場合は 1

注意事項

CObject クラスの Compare() メソッドは常に true を戻し何のアクションも取りません。派生クラスでデータを比較したい場合には Compare(...) メソッドが実装されるべきです。「mode」パラメータは、多変量比較を実装する際に使用する必要があります。

例：

```
//--- CObject::Compare(...) の例
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Object create error");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Object create error");
      delete object_first;
      return;
     }
   //--- リンクを設定
   object_first.Next(object_second);
```

© 2000-2025, MetaQuotes Ltd.

```
   object_second.Prev(object_first);
   //--- オブジェクトを比較する
   int result=object_first.Compare(object_second);
   //--- オブジェクトを削除する
   delete object_first;
   delete object_second;
  }
```

© 2000–2025, MetaQuotes Ltd.

Save

データ要素をファイルに保存します。

```
virtual bool  Save(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に FileOpen(...) で開かれたバイナリファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

注意事項

CObject クラスの Save (int) は常に true を戻し何のアクションも取りません。派生クラスでデータを保存したい場合は Save (int) メソッドが実装されるべきです。

例:

```
//--- CObject::Save(int) の例
#include <Object.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CObject *object=new CObject;
   //---
   if(object!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- オブジェクトデータを設定する
   //--- . にて。.
   //--- ファイルを開く
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Save(file_handle))
        {
         //--- ファイル保存エラー
         printf("File save: Error %d!",GetLastError());
         delete object;
         FileClose(file_handle);
         //---
         return;
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      FileClose(file_handle);
     }
   delete object;
  }
```

© 2000–2025, MetaQuotes Ltd.

Load

データ要素をファイルから読み込みます。

```
virtual bool  Load(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に FileOpen() 関数で開かれたバイナリファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

注意事項

CObject クラスの Load (int) は常に true を戻し何のアクションも取りません。派生クラスでデータを読み込みたい場合は Load (int) メソッドが実装されるべきです。

例:

```
//--- CObject::Load(int) の例
#include <Object.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CObject *object=new CObject;
   //---
   if(object!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ファイルを開く
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Load(file_handle))
        {
         //--- ファイル読み込みエラー
         printf("File load: Error %d!",GetLastError());
         delete object;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- オブジェクトを使用する
   //--- . にて。.
   delete object;
  }
```

© 2000–2025, MetaQuotes Ltd.

Type

型の識別子を取得します。

```
virtual int  Type() const
```

戻り値

型の識別子（CObject は 0）

例：

```
//--- CObject::Type() の例
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object=new CObject;
   //---
   object=new CObject;
   if(object ==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- オブジェクトの型を取得する
   int type=object.Type();
   //--- オブジェクトを削除する
   delete object;
  }
```

© 2000–2025, MetaQuotes Ltd.

データ構造体

このセクションには、様々なデータ構造体（配列、リンクリストなど）の操作に関する技術的な詳細と関連するMQL5 標準ライブラリコンポーネントの説明が含まれています。

データ構造体クラスの使用は、（複合データ構造を含む）様々な形式のカスタムデータ格納の作成時に時間を節約します。

MQL5 標準ライブラリのデータ構造体クラスに関連したファイルは、端末の作業ディレクトリの Include¥Arrays フォルダ内に配置されています。

データ配列

データの動的配列のクラスの使用は、（多次元配列を含む）様々な形式のカスタムデータ格納の作成時に時間を節約します。

MQL5 標準ライブラリのデータ配列に関連したファイルは、端末の作業ディレクトリの Include¥Arrays フォルダ内に配置されています。

Class	説明
CArray	動的データ配列の基本クラス
CArrayChar	char または uchar 型の動的配列の変数
CArrayShort	short または ushort 型の動的配列の変数
CArrayInt	int または uint 型の動的配列の変数
CArrayLong	long または ulong 型の動的配列の変数
CArrayFloat	float 型の動的配列の変数
CArrayDouble	double 型の動的配列の変数
CArrayString	string 型の動的配列の変数
CArrayObj	動的配列ポインタ CObject
CList	CObject とその派生クラスのインスタンスのリストを操作する機能を提供
CTreeNode	CTree二進木節点を操作する機能を提供
CTree	CTreeNode とその派生クラスのインスタンスの二進木を操作する機能を提供

© 2000–2025, MetaQuotes Ltd.

CArray

CArray は変数の動的配列の基本クラスです。

説明

CArray クラスは、メモリ割り当て、ソート、及びファイルでの作業などの、動的配列の変数操作を意図しています。

宣言

```
class CArray : public CObject
```

タイトル

```
#include <Arrays\Array.mqh>
```

継承階層

CObject

CArray

直接子孫

CArrayChar, CArrayDouble, CArrayFloat, CArrayInt, CArrayLong, CArrayObj, CArrayShort, CArrayString

クラスメソッド

属性	
Step	配列のステップ増分サイズを取得します。
Step	配列のステップ増分サイズを設定します。
Total	配列の要素数を取得します。
Available	追加のメモリ割り当てをせずに利用可能な配列の要素の数を取得します。
Max	メモリの再割り当てなしで可能な配列の最大サイズを取得します。
IsSorted	指定されたモードでそーとされている配列のフラグを取得します。
SortMode	配列のソートモードを取得します。
クリアのメソッド	
Clear	メモリを解放せずに、配列の全ての要素を削除します。
ソートのメソッド	
Sort	指定されたオプションで配列をソートします。
入出力	
virtual Save	データ配列をファイルに保存します。

© 2000–2025, MetaQuotes Ltd.

属性	
virtual Load	データ配列をファイルから読み込みます。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Type, Compare

派生クラス

- CArrayChar
- CArrayShort
- CArrayInt
- CArrayLong
- CArrayFloat
- CArrayDouble
- CArrayString
- CArrayObj

© 2000-2025, MetaQuotes Ltd.

Step

配列のステップ増分サイズを取得します。

```
int  Step() const
```

戻り値

配列のステップ増分サイズ

例:

```
//--- CArray::Step() の例
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- サイズ変更のステップを取得する
   int step=array.Step();
   //--- 配列を使用する
   //--- ...
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Step

配列のステップ増分サイズを設定します。

```
bool  Step(
   int  step      // ステップ
  )
```

パラメータ

step

[in] 配列の新しいステップ増分サイズ

戻り値

成功の場合は true、ゼロ以下のステップを確立しようとする試みがあった場合は false

例:

```
//--- CArray::Step(int) の例
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- サイズ変更のステップを設定する
   bool result=array.Step(1024);
   //--- 配列を使用する
   //--- ...
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Total

配列の要素数を取得します。

```
int  Total() const;
```

戻り値

配列の要素数

例：

```
//--- CArray::Total() の例
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 合計をチェックする
   int total=array.Total();
   //--- 配列を使用する
   //--- ...
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Available

追加のメモリ割り当てをせずに利用可能な配列の要素の数を取得します。

```
int  Available() const
```

戻り値

追加のメモリ割り当てをせずに利用可能な配列の要素の数

例：

```
//--- CArray::Available() の例
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 利用可能かをチェックする
   int available=array.Available();
   //--- 配列を使用する
   //--- ...
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Max

メモリの再割り当てなしで可能な配列の最大サイズを取得します。

```
int  Max() const
```

戻り値

メモリの再割り当てなしで可能な配列の最大サイズ

例：

```
//--- CArray::Max() の例
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 最大のサイズをチェックする
   int max=array.Max();
   //--- 配列を使用する
   //--- ...
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

IsSorted

指定されたモードでソートされている配列のフラグを取得します。

```
bool  IsSorted(
   int  mode=0       // ソートのモード
   ) const
```

パラメータ

mode=0

[in] テストされたソートモード

戻り値

ソートされたリストのフラグ。リストが指定されたバージョンでソートされた場合は true、それ以外の場合は false。

注意事項

ソートフラグは直接には変更出来ません。それは Sort() メソッドによって設定され InserSort(...) 以外の追加/挿入メソッドでリセットされます。

例:

```
//--- CArray::IsSorted() の例
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソートされたかをチェックする
   if(array.IsSorted())
     {
      //--- ソートされた配列のメソッドを使用する
      //--- ...
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SortMode

配列のソートモードを取得します。

```
int  SortMode() const;
```

戻り値

ソートのモード

例：

```
//--- CArray::SortMode() の例
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソートのモードをチェックする
   int sort_mode=array.SortMode();
   //--- 配列を使用する
   //--- ...
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Clear

メモリを解放せずに、配列の全ての要素を削除します。

```
void  Clear()
```

戻り値

なし

例：

```
//--- CArray::Clear() の例
#include <Arrays\Array.mqh>
//---
void OnStart()
 {
 CArray *array=new CArray;
 //---
 if(array==NULL)
 {
 printf("Object create error");
 return;
 }
 //--- 配列を使用する
 //--- ...
 //--- 配列をクリアする
 array.Clear();
 //--- 配列を削除する
 delete array;
 }
```

© 2000–2025, MetaQuotes Ltd.

Sort

指定されたオプションで配列をソートします。

```
void  Sort(
   int  mode=0       // ソートのモード
  )
```

パラメータ

mode=0

[in] 配列ソートのモード

戻り値

No.

注意事項

配列のソートの順番は常に昇順です。プリミティブデータ型の配列（CArrayChar、CArrayShortなど）の場合は、「 mode 」パラメータは使用されません。CArrayObj の配列のためには、複数の変数によるソートが、派生クラスの Sort (int) で実装する必要があります。

例:

```
//--- CArray::Sort(int) の例
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- モード 0 でのソート
   array.Sort(0);
   //--- 配列を使用する
   //--- ...
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

データ配列をファイルに保存します。

```
virtual bool  Save(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に FileOpen(...) で開かれたファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

例:

```
//--- CArray::Save(int) の例
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   int     file_handle;
   CArray *array=new CArray;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ファイルを開く
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- ファイル保存エラー
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Load

データ配列をファイルから読み込みます。

```
virtual bool  Load(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に FileOpen() 関数で開かれたバイナリファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

例:

```
//--- CArray::Load(...) の例
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   int     file_handle;
   CArray *array=new CArray;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ファイルを開く
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- ファイル読み込みエラー
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

CArrayChar

CArrayChar クラスは char または uchar 型の変数の動的配列のクラスです。

説明

CArrayChar クラスは char または uchar 型の動的配列の変数を操作する能力を提供します。配列要素の追加/挿入/削除、配列のソート、ソートされた配列での検索が実装されています。またファイル操作のメソッドも実装されています。

宣言

```
class CArrayChar : public CArray
```

タイトル

```
#include <Arrays\ArrayChar.mqh>
```

継承階層

CObject
  CArray
    CArrayChar

クラスメソッド

メモリ制御	
Reserve	メモリを割り当てて、配列のサイズを増加します。
Resize	配列の新しい(減少した)サイズを設定します。
Shutdown	配列をクリアして、メモリを完全に開放します。
追加メソッド	
Add	配列の最後に要素を追加します。
AddArray	配列の最後に別の配列の要素を追加します。
AddArray	配列の最後に別の配列の要素を追加します。
Insert	指定された位置に配列の要素を挿入します。
InsertArray	指定された位置に別の配列の要素を挿入します。
InsertArray	指定された位置に別の配列の要素を挿入します。
AssignArray	別の配列の要素を複製します。
AssignArray	別の配列の要素を複製します。
変更メソッド	
Update	指定された位置の配列の要素を変更します。

© 2000–2025, MetaQuotes Ltd.

メモリ制御	
Shift	配列内の指定された位置から要素を指定されたオフセットに移動します。
削除メソッド	
Delete	配列の指定された位置から要素を削除します。
DeleteRange	配列の指定された位置から要素のグループを削除します。
アクセスメソッド	
At	配列の指定された位置から要素を取得します。
比較メソッド	
CompareArray	配列を別の配列と比較します。
CompareArray	配列を別の配列と比較します。
ソートされた配列の操作	
InsertSort	ソートされた配列に要素を挿入します。
Search	ソートされた配列内でサンプルに等しい要素を検索します。
SearchGreat	ソートされた配列内でサンプルより大きい要素を検索します。
SearchLess	ソートされた配列内でサンプルより小さい要素を検索します。
SearchGreatOrEqual	ソートされた配列内でサンプル以上の要素を検索します。
SearchLessOrEqual	ソートされた配列内でサンプル以下の要素を検索します。
SearchFirst	ソートされた配列内でサンプルに等しい最初の要素を検索します。
SearchLast	ソートされた配列内でサンプルに等しい最後の要素を検索します。
SearchLinear	配列内でサンプルに等しい要素を検索します。
入出力	
virtual Save	データ配列をファイルに保存します。
virtual Load	データ配列をファイルから読み込みます。
virtual Type	配列の型識別子を取得します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000–2025, MetaQuotes Ltd.

Reserve

メモリを割り当てて、配列のサイズを増加します。

```
bool  Reserve(
   int  size      // 数
  )
```

パラメータ

size

[in] 配列に追加される要素の数

戻り値

成功の場合は true、ゼロ以下の数を追加しようとする試みがあった場合、または配列が増加しなかった場合は false

注意事項

メモリの断片化を低減するために、配列のサイズは、以前に Step(int) によって決定されたステップのまたはデフォルトの16ステップを使用して変更されます。

例:

```
//--- CArrayChar::Reserve(int) の例
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- メモリを予約する
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

配列の新しい(減少した)サイズを設定します。

```
bool  Resize(
   int  size      // サイズ
  )
```

パラメータ

size

[in] 配列の新しいサイズ

戻り値

成功の場合はtrue、ゼロ以下のサイズを設定しようとする試みがあった場合はfalse

注意事項

配列のサイズ変更はメモリ使用の最適化に役立ちます。右側の余分な要素が失われます。メモリの断片化を低減するために、配列のサイズは、以前に Step(int) によって決定されたステップのまたはデフォルトの16ステップを使用して変更されます。

例:

```
//--- CArrayChar::Resize(int) の例
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列のサイズを変更する
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

配列をクリアして、メモリを完全に開放します。

```
bool  Shutdown()
```

戻り値

成功の場合は true、エラーが発生した場合は false

例：

```
//--- CArrayChar::Shutdown() の例
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をシャットダウンする
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Add

配列の最後に要素を追加します。

```
bool  Add(
   char  element      // 追加する要素
  )
```

パラメータ

element

[in] 配列に追加される要素の値

戻り値

成功の場合は true、要素が追加できなかった場合は false

例:

```
//--- CArrayChar::Add(char) の例
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

配列の最後に別の配列の要素を追加します。

```
bool  AddArray(
   const char&  src[]      // ソース配列
  )
```

パラメータ

src[]

[in] 追加する要素を持つ配列への参照

戻り値

成功の場合は true、要素が追加できなかった場合は false

例:

```
//--- CArrayChar::AddArray(const char &[]) の例
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 別の配列を追加する
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete array;
      return;
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

配列の最後に別の配列の要素を追加します。

```
bool  AddArray(
   const CArrayChar*  src      // コピー元へのポインタ
  )
```

パラメータ

src

[in] 追加する要素へのポインタ（CArrayChar クラスインスタンス）

戻り値

成功の場合は true、要素が追加できなかった場合は false

例：

```
//--- CArrayChar::AddArray(const CArrayChar*) の例
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソース配列を作成する
   CArrayChar *src=new CArrayChar;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ソース配列に要素を加える
   //--- . にて。.
   //--- 別の配列を追加する
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete src;
      delete array;
      return;
     }
   //--- ソース配列を削除する
   delete src;
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Insert

配列の指定された位置に要素を挿入します。

```
bool  Insert(
   char  element,     // 挿入する要素
   int   pos          // 位置
  )
```

パラメータ

element

[in] 配列に挿入される要素の値

pos

[in] 配列内の挿入位置

戻り値

成功の場合は true、要素が挿入できなかった場合は false

例:

```
//--- CArrayChar::Insert(char,int) の例
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 要素を挿入する
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

指定された位置に別の配列の要素を挿入します。

```
bool  InsertArray(
   const char&  src[],     // ソース配列
   int           pos       // 位置
  )
```

パラメータ

src[]

[in] 挿入する要素を持つ配列への参照

pos

[in] 配列内の挿入位置

戻り値

成功の場合は true、要素が挿入できなかった場合は false

例:

```
//--- CArrayChar::InsertArray(const char &[],int) の例
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 別の配列を挿入する
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete array;
      return;
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

指定された位置に別の配列の要素を挿入します。

```
bool  InsertArray(
   CArrayChar*  src,      // ソース配列へのポインタ
   int           pos      // 位置
  )
```

パラメータ

src

[in] 挿入する要素へのポインタ（CArrayChar クラスインスタンス）

pos

[in] 配列内の挿入位置

戻り値

成功の場合は true、要素が挿入できなかった場合は false

例：

```
//--- CArrayChar::InsertArray(const CArrayChar*,int) の例
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソース配列を作成する
   CArrayChar *src=new CArrayChar;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ソース配列に要素を加える
   //--- . にて。.
   //--- 別の配列を挿入する
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete src;
      delete array;
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }
   //--- ソース配列を削除する
   delete src;
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

AssignArray

別の配列の要素を複製します。

```
bool  AssignArray(
   const char&  src[]      // ソース配列
  )
```

パラメータ

src[]

[in] 複製する配列要素への参照

戻り値

成功の場合は true、要素が複製できなかった場合は false

例:

```
//--- CArrayChar::AssignArray(const char &[]) の例
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 別の配列を割り当てる
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete array;
      return;
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

AssignArray

別の配列の要素を複製します。

```
bool  AssignArray(
   const CArrayChar*  src      // コピー元へのポインタ
  )
```

パラメータ

src

[in] 追加する要素へのポインタ（CArrayChar クラスインスタンス）

戻り値

成功の場合は true、要素が複製できなかった場合は false

例：

```
//--- CArrayChar::AssignArray(const CArrayChar*) の例
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソース配列を作成する
   CArrayChar *src  =new CArrayChar;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ソース配列に要素を加える
   //--- . にて。.
   //--- 別の配列を割り当てる
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete src;
      delete array;
      return;
     }
   //--- 配列が同一
   //--- ソース配列を削除する
```

© 2000-2025, MetaQuotes Ltd.

```
   delete src;
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Update

指定された位置の配列の要素を変更します。

```
bool  Update(
   int   pos,          // 位置
   char  element       // 値
  )
```

パラメータ

pos

[in] 変更される要素の配列内での位置

element

[in] 要素の新しい値

戻り値

成功の場合は true、要素が変更できなかった場合は false

例:

```
//--- CArrayChar::Update(int,char) の例
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素を更新する
   if(!array.Update(0,'A'))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shift

配列内の指定された位置から要素を指定されたオフセットに移動します。

```
bool  Shift(
   int  pos,        // 位置
   int  shift       // 値
  )
```

パラメータ

pos

[in] 配列での移動した要素の位置

shift

[in]（正と負の両方）のシフトの値

戻り値

成功の場合は true、要素が移動できなかった場合は false

例:

```
//--- CArrayChar::Shift(int,int) の例
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素をシフトする
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

配列の指定された位置から要素を削除します。

```
bool  Delete(
   int  pos      // 位置
  )
```

パラメータ

pos

[in] 配列から削除される要素の位置

戻り値

成功の場合は true、要素が削除できなかった場合は false

例:

```
//--- CArrayChar::Delete(int) の例
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素を削除する
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

DeleteRange

配列の指定された位置から要素のグループを削除します。

```
bool  DeleteRange(
   int  from,     // 最初の要素の位置
   int  to        // 最後の要素の位置
  )
```

パラメータ

from

[in] 配列の削除される範囲の最初の要素の位置

to

[in] 配列の削除される範囲の最後の要素の位置

戻り値

成功の場合は true、削除が失敗した場合は false

例:

```
//--- CArrayChar::DeleteRange(int,int) の例
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素を削除する
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

At

配列の指定された位置から要素を取得します。

```
char  At(
   int  pos      // 位置
   ) const
```

パラメータ

pos

[in] 配列での望まれた要素の位置

戻り値

成功の場合は要素の値、存在しない位置での要素の取得が試まれた場合CHAR_MAX（直近エラーはERR_OUT_OF_RANGE）

注意事項

CHAR_MAX は配列要素の有効な値である可能性もあるので、この値が取得されたら、常に最終のエラーコードをチェックします。

例:

```
//--- CArrayChar::At(int) の例
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   for(int i=0;i<array.Total();i++)
     {
      char result=array.At(i);
      if(result==CHAR_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- 配列読み込みエラー
         printf("Get element error");
         delete array;
         return;
        }
      //--- 要素を使用する
      //--- . にて。.
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

CompareArray

配列を別の配列と比較します。

```
bool  CompareArray(
   const char&  src[]      // ソース配列
   ) const
```

パラメータ

src[]

[in] 比較元の要素を持つ配列への参照

戻り値

配列が同一なら true、それ以外の場合は false

例:

```
//--- CArrayChar::CompareArray(const char &[]) の例
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 別の配列と比較する
   int result=array.CompareArray(src);
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

配列を別の配列と比較します。

```
bool  CompareArray(
   const CArrayChar*  src      // 比較元へのポインタ
   ) const
```

パラメータ

src

[in] 比較する要素へのポインタ（CArrayChar クラスインスタンス）

戻り値

配列が同一なら true、それ以外の場合は false

例：

```
//--- CArrayChar::CompareArray(const CArrayChar*) の例
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソース配列を作成する
   CArrayChar *src=new CArrayChar;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ソース配列に要素を加える
   //--- . にて。.
   //--- 別の配列と比較する
   int result=array.CompareArray(src);
   //--- 配列を削除する
   delete src;
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

InsertSort

ソートされた配列に要素を挿入します。

```
bool  InsertSort(
   char  element      // 挿入する要素
  )
```

パラメータ

element

[in] ソートされた配列に挿入される要素の値

戻り値

成功の場合は true、要素が挿入できなかった場合は false

例:

```
//--- CArrayChar::InsertSort(char) の例
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を挿入する
   if(!array.InsertSort('A'))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Search

ソートされた配列内でサンプルに等しい要素を検索します。

```
int  Search(
   char  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayChar::Search(char) の例
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.Search('A')!=-1) printf("Element found");
   else                      printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

ソートされた配列内でサンプルより大きい要素を検索します。

```
int  SearchGreat(
   char  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayChar::SearchGreat(char) の例
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchGreat('A')!=-1) printf("Element found");
   else                           printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

ソートされた配列内でサンプルより小さい要素を検索します。

```
int  SearchLess(
   char  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayChar::SearchLess(char) の例
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchLess('A')!=-1) printf("Element found");
   else                          printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

ソートされた配列内でサンプル以上の要素を検索します。

```
int  SearchGreatOrEqual(
   char  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayChar::SearchGreatOrEqual(char) の例
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchGreatOrEqual('A')!=-1) printf("Element found");
   else                                  printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

ソートされた配列内でサンプル以下の要素を検索します。

```
int  SearchLessOrEqual(
   char  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayChar::SearchLessOrEqual(char) の例
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchLessOrEqual('A')!=-1) printf("Element found");
   else                                 printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

ソートされた配列内でサンプルに等しい最初の要素を検索します。

```
int  SearchFirst(
   char  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayChar::SearchFirst(char) の例
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchFirst('A')!=-1) printf("Element found");
   else                           printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

ソートされた配列内でサンプルに等しい最後の要素を検索します。

```
int  SearchLast(
   char  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayChar::SearchLast(char) の例
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchLast('A')!=-1) printf("Element found");
   else                          printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLinear

配列内でサンプルに等しい要素を検索します。

```
int  SearchLinear(
   char  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

注意事項

このメソッドは、ソートされていない配列では線形探索（または順次検索）アルゴリズムを使用します。

例:

```
//--- CArrayChar::SearchLinear(char) の例
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素を探す
   if(array.SearchLinear('A')!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

データ配列をファイルに保存します。

```
virtual bool  Save(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に FileOpen(...) で開かれたファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

例:

```
//--- CArrayChar::Save(int) の例
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayChar *array=new CArrayChar;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ファイルを開く
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- ファイル保存エラー
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

データ配列をファイルから読み込みます。

```
virtual bool  Load(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に FileOpen(...) で開かれたファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

例:

```
//--- CArrayChar::Load(int) の例
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayChar *array=new CArrayChar;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ファイルを開く
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- ファイル読み込みエラー
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- 配列要素を使用する
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = '%c'",i,array.At(i));
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Type

配列の型識別子を取得します。

```
virtual int  Type() const
```

戻り値

配列の型識別子（CArrayChar は 77）

例：

```
//--- CArrayChar::Type() の例
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列の型を取得する
   int type=array.Type();
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

CArrayShort

CArrayShort クラスは short または ushort 型の変数の動的配列のクラスです。

説明

CArrayShort クラスは short または ushort 型の動的配列の変数を操作する能力を提供します。配列要素の追加/挿入/削除、配列のソート、ソートされた配列での検索が実装されています。またファイル操作のメソッドも実装されています。

宣言

```
class CArrayShort : public CArray
```

タイトル

```
#include <Arrays\ArrayShort.mqh>
```

継承階層

CObject
  CArray
    CArrayShort

クラスメソッド

メモリ制御	
Reserve	メモリを割り当てて、配列のサイズを増加します。
Resize	配列の新しい(減少した)サイズを設定します。
Shutdown	配列をクリアして、メモリを完全に開放します。
追加メソッド	
Add	配列の最後に要素を追加します。
AddArray	配列の最後に別の配列の要素を追加します。
AddArray	配列の最後に別の配列の要素を追加します。
Insert	指定された位置に配列の要素を挿入します。
InsertArray	指定された位置に別の配列の要素を挿入します。
InsertArray	指定された位置に別の配列の要素を挿入します。
AssignArray	別の配列の要素を複製します。
AssignArray	別の配列の要素を複製します。
更新メソッド	
Update	指定された位置の配列の要素を変更します。

© 2000-2025, MetaQuotes Ltd.

メモリ制御	
Shift	配列内の指定された位置から要素を指定されたオフセットに移動します。
削除メソッド	
Delete	配列の指定された位置から要素を削除します。
DeleteRange	配列の指定された位置から要素のグループを削除します。
アクセスメソッド	
At	配列の指定された位置から要素を取得します。
比較メソッド	
CompareArray	配列を別の配列と比較します。
CompareArray	配列を別の配列と比較します。
ソートされた配列の操作	
InsertSort	ソートされた配列に要素を挿入します。
Search	ソートされた配列内でサンプルに等しい要素を検索します。
SearchGreat	ソートされた配列内でサンプルより大きい要素を検索します。
SearchLess	ソートされた配列内でサンプルより小さい要素を検索します。
SearchGreatOrEqual	ソートされた配列内でサンプル以上の要素を検索します。
SearchLessOrEqual	ソートされた配列内でサンプル以下の要素を検索します。
SearchFirst	ソートされた配列内でサンプルに等しい最初の要素を検索します。
SearchLast	ソートされた配列内でサンプルに等しい最後の要素を検索します。
SearchLinear	配列内でサンプルに等しい要素を検索します。
入出力	
virtual Save	データ配列をファイルに保存します。
virtual Load	データ配列をファイルから読み込みます。
virtual Type	配列の型識別子を取得します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000–2025, MetaQuotes Ltd.

Reserve

メモリを割り当てて、配列のサイズを増加します。

```
bool  Reserve(
   int  size      // 数
  )
```

パラメータ

size

[in] 配列に追加される要素の数

戻り値

成功の場合は true、ゼロ以下の数を追加しようとする試みがあった場合、または配列が増加しなかった場合は false

注意事項

メモリの断片化を低減するために、配列のサイズは、以前に Step(int) によって決定されたステップのまたはデフォルトの16ステップを使用して変更されます。

例:

```
//--- CArrayShort::Reserve(int) の例
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- メモリを予約する
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

配列の新しい(減少した)サイズを設定します。

```
bool  Resize(
   int  size      // サイズ
  )
```

パラメータ

size

[in] 配列の新しいサイズ

戻り値

成功の場合はtrue、ゼロ以下のサイズを設定しようとする試みがあった場合はfalse

注意事項

配列のサイズ変更はメモリ使用の最適化に役立ちます。右側の余分な要素が失われます。メモリの断片化を低減するために、配列のサイズは、以前に Step(int) によって決定されたステップのまたはデフォルトの16ステップを使用して変更されます。

例:

```
//--- CArrayShort::Resize(int) の例
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列のサイズを変更する
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

配列をクリアして、メモリを完全に開放します。

```
bool  Shutdown()
```

戻り値

成功の場合は true、エラーが発生した場合は false

例：

```
//--- CArrayShort::Shutdown() の例
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をシャットダウンする
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Add

配列の最後に要素を追加します。

```
bool  Add(
   short  element      // 追加する要素
  )
```

パラメータ

element

[in] 配列に追加される要素の値

戻り値

成功の場合は true、要素が追加できなかった場合は false

例:

```
//--- CArrayShort::Add(short) の例
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

配列の最後に別の配列の要素を追加します。

```
bool  AddArray(
   const short&  src[]      // ソース配列
  )
```

パラメータ

src[]

[in] 追加する要素を持つ配列への参照

戻り値

成功の場合は true、要素が追加できなかった場合は false

例:

```
//--- CArrayShort::AddArray(const short &[]) の例
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 別の配列を追加する
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete array;
      return;
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

AddArray

配列の最後に別の配列の要素を追加します。

```
bool  AddArray(
   const CArrayShort*  src      // ソース配列
  )
```

パラメータ

src

[in] 追加する要素へのポインタ（CArrayShort クラスインスタンス）

戻り値

成功の場合は true、要素が追加できなかった場合は false

例：

```
//--- CArrayShort::AddArray(const CArrayShort*) の例
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソース配列を作成する
   CArrayShort *src=new CArrayShort;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ソース配列に要素を加える
   //--- . にて。.
   //--- 別の配列を追加する
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete src;
      delete array;
      return;
     }
   //--- ソース配列を削除する
   delete src;
```

© 2000–2025, MetaQuotes Ltd.

```
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Insert

配列の指定された位置に要素を挿入します。

```
bool  Insert(
   short  element,      // 挿入する要素
   int    pos           // 位置
  )
```

パラメータ

element

[in] 配列に挿入される要素の値

pos

[in] 配列内の挿入位置

戻り値

成功の場合は true、要素が挿入できなかった場合は false

例:

```
//--- CArrayShort::Insert(short,int) の例
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 要素を挿入する
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

指定された位置に別の配列の要素を挿入します。

```
bool  InsertArray(
   const short&  src[],     // 挿入元の配列
   int           pos        // 位置
  )
```

パラメータ

src[]

[in] 挿入する要素を持つ配列への参照

pos

[in] 配列内の挿入位置

戻り値

成功の場合は true、要素が挿入できなかった場合は false

例:

```
//--- CArrayShort::InsertArray(const short &[],int) の例
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 別の配列を挿入する
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete array;
      return;
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

InsertArray

別の配列の要素を指定された位置に挿入します。

```
bool  InsertArray(
   CArrayShort*  src,      // 挿入元へのポインタ
   int           pos       // 位置
  )
```

パラメータ

src

[in] 挿入する要素へのポインタ（CArrayShort クラスインスタンス）

pos

[in] 配列内の挿入位置

戻り値

成功の場合は true、要素が挿入できなかった場合は false

例：

```
//--- CArrayShort::InsertArray(const CArrayShort*,int) の例
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソース配列を作成する
   CArrayShort *src=new CArrayShort;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ソース配列に要素を加える
   //--- . にて。.
   //--- 別の配列を挿入する
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete src;
      delete array;
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }
   //--- ソース配列を削除する
   delete src;
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

AssignArray

別の配列の要素を複製します。

```
bool  AssignArray(
   const short&  src[]      // ソース配列
  )
```

パラメータ

src[]

[in] 複製する配列要素への参照

戻り値

成功の場合は true、要素が複製できなかった場合は false

例:

```
//--- CArrayShort::AssignArray(const short &[]) の例
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 別の配列を割り当てる
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete array;
      return;
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

AssignArray

別の配列の要素を複製します。

```
bool  AssignArray(
   const CArrayShort*  src      // ソース配列
  )
```

パラメータ

src

[in] 複製する要素へのポインタ（CArrayShort クラスインスタンス）

戻り値

成功の場合は true、要素が複製できなかった場合は false

例：

```
//--- CArrayShort::AssignArray(const CArrayShort*) の例
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソース配列を作成する
   CArrayShort *src  =new CArrayShort;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ソース配列に要素を加える
   //--- . にて。.
   //--- 別の配列を割り当てる
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete src;
      delete array;
      return;
     }
   //--- 配列が同一
   //--- ソース配列を削除する
```

© 2000-2025, MetaQuotes Ltd.

```
   delete src;
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Update

指定された位置の配列の要素を変更します。

```
bool  Update(
   int    pos,          // 位置
   short  element       // 値
  )
```

パラメータ

pos

[in] 変更される要素の配列内での位置

element

[in] 要素の新しい値

戻り値

成功の場合は true、要素が変更できなかった場合は false

例:

```
//--- CArrayShort::Update(int,short) の例
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素を更新する
   if(!array.Update(0,100))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shift

配列内の指定された位置から要素を指定されたオフセットに移動します。

```
bool  Shift(
   int  pos,        // 位置
   int  shift       // シフト
  )
```

パラメータ

pos

[in] 配列での移動した要素の位置

shift

[in]（正と負の両方）のシフトの値

戻り値

成功の場合は true、要素が移動できなかった場合は false

例:

```
//--- CArrayShort::Shift(int,int) の例
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素をシフトする
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

配列の指定された位置から要素を削除します。

```
bool  Delete(
   int  pos      // 位置
  )
```

パラメータ

pos

[in] 配列から削除される要素の位置

戻り値

成功の場合は true、要素が削除できなかった場合は false

例:

```
//--- CArrayShort::Delete(int) の例
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素を削除する
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

DeleteRange

配列の指定された位置から要素のグループを削除します。

```
bool  DeleteRange(
   int  from,     // 最初の要素の位置
   int  to        // 最後の要素の位置
  )
```

パラメータ

from

[in] 配列の削除される範囲の最初の要素の位置

to

[in] 配列の削除される範囲の最後の要素の位置

戻り値

成功の場合は true、削除が失敗した場合は false

例:

```
//--- CArrayShort::DeleteRange(int,int) の例
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素を削除する
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

At

配列の指定された位置から要素を取得します。

```
short  At(
   int  pos      // 位置
   ) const
```

パラメータ

pos

[in] 配列での望まれた要素の位置

戻り値

成功の場合は要素の値、存在しない位置での要素の取得が試まれた場合SHORT_MAX(直近エラーはERR_OUT_OF_RANGE)

注意事項

SHORT_MAX は配列要素の有効な値である可能性もあるので、この値が取得されたら、常に最終のエラーコードをチェックします。

例:

```
//--- CArrayShort::At(int) の例
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   for(int i=0;i<array.Total();i++)
     {
      short result=array.At(i);
      if(result==SHORT_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- 配列読み込みエラー
         printf("Get element error");
         delete array;
         return;
        }
      //--- 要素を使用する
      //--- . にて。.
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

CompareArray

配列を別の配列と比較します。

```
bool  CompareArray(
   const short&  src[]      // ソース配列
   ) const
```

パラメータ

src[]

[in] 比較元の要素を持つ配列への参照

戻り値

配列が同一なら true、それ以外の場合は false

例:

```
//--- CArrayShort::CompareArray(const short &[]) の例
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 別の配列と比較する
   int result=array.CompareArray(src);
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

CompareArray

配列を別の配列と比較します。

```
bool  CompareArray(
   const CArrayShort*  src      // ソース配列
   ) const
```

パラメータ

src

[in] 比較する要素へのポインタ（CArrayShort クラスインスタンス）

戻り値

配列が同一なら true、それ以外の場合は false

例：

```
//--- CArrayShort::CompareArray(const CArrayShort*) の例
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソース配列を作成する
   CArrayShort *src=new CArrayShort;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ソース配列に要素を加える
   //--- . にて。.
   //--- 別の配列と比較する
   int result=array.CompareArray(src);
   //--- 配列を削除する
   delete src;
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertSort

ソートされた配列に要素を挿入します。

```
bool  InsertSort(
   short  element      // 挿入する要素
  )
```

パラメータ

element

[in] ソートされた配列に挿入される要素の値

戻り値

成功の場合は true、要素が挿入できなかった場合は false

例:

```
//--- CArrayShort::InsertSort(short) の例
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を挿入する
   if(!array.InsertSort(100))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Search

ソートされた配列内でサンプルに等しい要素を検索します。

```
int  Search(
   short  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayShort::Search(short) の例
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.Search(100)!=-1) printf("Element found");
   else                      printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

ソートされた配列内でサンプルより大きい要素を検索します。

```
int  SearchGreat(
   short  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayShort::SearchGreat(short) の例
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchGreat(100)!=-1) printf("Element found");
   else                           printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

ソートされた配列内でサンプルより小さい要素を検索します。

```
int  SearchLess(
   short  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayShort::SearchLess(short) の例
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchLess(100)!=-1) printf("Element found");
   else                          printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

ソートされた配列内でサンプル以上の要素を検索します。

```
int  SearchGreatOrEqual(
   short  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayShort::SearchGreatOrEqual(short) の例
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchGreatOrEqual(100)!=-1) printf("Element found");
   else                                  printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

ソートされた配列内でサンプル以下の要素を検索します。

```
int  SearchLessOrEqual(
   short  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayShort::SearchLessOrEqual(short) の例
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchLessOrEqual(100)!=-1) printf("Element found");
   else                                 printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

ソートされた配列内でサンプルに等しい最初の要素を検索します。

```
int  SearchFirst(
   short  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayShort::SearchFirst(short) の例
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchFirst(100)!=-1) printf("Element found");
   else                           printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

ソートされた配列内でサンプルに等しい最後の要素を検索します。

```
int  SearchLast(
   short  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayShort::SearchLast(short) の例
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchLast(100)!=-1) printf("Element found");
   else                          printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLinear

配列内でサンプルに等しい要素を検索します。

```
int  SearchLinear(
   short  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

注意事項

このメソッドは、ソートされていない配列では線形探索（または順次検索）アルゴリズムを使用します。

例:

```
//--- CArrayShort::SearchLinear(short) の例
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素を探す
   if(array.SearchLinear(100)!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

データ配列をファイルに保存します。

```
virtual bool  Save(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に FileOpen(...) で開かれたファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

例:

```
//--- CArrayShort::Save(int) の例
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayShort *array=new CArrayShort;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 100要素を追加する
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- ファイルを開く
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- ファイル保存エラー
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Load

データ配列をファイルから読み込みます。

```
virtual bool  Load(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に FileOpen(...) で開かれたファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

例:

```
//--- CArrayShort::Load(int) の例
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayShort *array=new CArrayShort;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ファイルを開く
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- ファイル読み込みエラー
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- 配列要素を使用する
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = %d",i,array.At(i));
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Type

配列の型識別子を取得します。

```
virtual int  Type() const
```

戻り値

配列の型識別子（CArrayShort は 82）

例:

```
//--- CArrayShort::Type() の例
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列の型を取得する
   int type=array.Type();
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

CArrayInt

CArrayInt クラスは int または uint 型の変数の動的配列のクラスです。

説明

CArrayInt クラスは、int または uint 型変数の動的配列を操作する関数を提供します。配列要素の追加/挿入/削除、配列のソート、ソートされた配列での検索が実装されています。またファイル操作のメソッドも実装されています。

宣言

```
class CArrayInt : public CArray
```

タイトル

```
#include <Arrays\ArrayInt.mqh>
```

継承階層

CObject
  CArray
    CArrayInt

直接子孫

CSpreadBuffer

クラスメソッド

メモリ制御	
Reserve	メモリを割り当てて、配列のサイズを増加します。
Resize	配列の新しい(減少した)サイズを設定します。
Shutdown	配列をクリアして、メモリを完全に開放します。
追加メソッド	
Add	配列の最後に要素を追加します。
AddArray	配列の最後に別の配列の要素を追加します。
AddArray	配列の最後に別の配列の要素を追加します。
Insert	指定された位置に配列の要素を挿入します。
InsertArray	指定された位置に別の配列の要素を挿入します。
InsertArray	指定された位置に別の配列の要素を挿入します。
AssignArray	別の配列の要素を複製します。
AssignArray	別の配列の要素を複製します。

© 2000-2025, MetaQuotes Ltd.

メモリ制御	
更新メソッド	
Update	指定された位置の配列の要素を変更します。
Shift	配列内の指定された位置から要素を指定されたオフセットに移動します。
削除メソッド	
Delete	配列の指定された位置から要素を削除します。
DeleteRange	配列の指定された位置から要素のグループを削除します。
アクセスメソッド	
At	配列の指定された位置から要素を取得します。
比較メソッド	
CompareArray	配列を別の配列と比較します。
CompareArray	配列を別の配列と比較します。
ソートされた配列の操作	
InsertSort	ソートされた配列に要素を挿入します。
Search	ソートされた配列内でサンプルに等しい要素を検索します。
SearchGreat	ソートされた配列内でサンプルより大きい要素を検索します。
SearchLess	ソートされた配列内でサンプルより小さい要素を検索します。
SearchGreatOrEqual	ソートされた配列内でサンプル以上の要素を検索します。
SearchLessOrEqual	ソートされた配列内でサンプル以下の要素を検索します。
SearchFirst	ソートされた配列内でサンプルに等しい最初の要素を検索します。
SearchLast	ソートされた配列内でサンプルに等しい最後の要素を検索します。
SearchLinear	配列内でサンプルに等しい要素を検索します。
入出力	
virtual Save	データ配列をファイルに保存します。
virtual Load	データ配列をファイルから読み込みます。
virtual Type	配列の型識別子を取得します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000–2025, MetaQuotes Ltd.

Reserve

メモリを割り当てて、配列のサイズを増加します。

```
bool  Reserve(
   int  size      // 数
  )
```

パラメータ

size

[in] 配列に追加される要素の数

戻り値

成功の場合は true、ゼロ以下の数を追加しようとする試みがあった場合、または配列が増加しなかった場合は false

注意事項

メモリの断片化を低減するために、配列のサイズは、以前に Step(int) によって決定されたステップのまたはデフォルトの16ステップを使用して変更されます。

例:

```
//--- CArrayInt::Reserve(int) の例
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- メモリを予約する
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

配列の新しい(減少した)サイズを設定します。

```
bool  Resize(
   int  size      // 数
  )
```

パラメータ

size

[in] 配列の新しいサイズ

戻り値

成功の場合は true、ゼロ以下のサイズを設定しようとする試みがあった場合は false

注意事項

配列のサイズ変更はメモリ使用の最適化に役立ちます。右側の余分な要素が失われます。メモリの断片化を低減するために、配列のサイズは、以前に Step(int) によって決定されたステップのまたはデフォルトの16ステップを使用して変更されます。

例:

```
//--- CArrayInt::Resize(int) の例
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列のサイズを変更する
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

配列をクリアして、メモリを完全に開放します。

```
bool  Shutdown()
```

戻り値

成功の場合は true、エラーが発生した場合は false

例：

```
//--- CArrayInt::Shutdown() の例
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をシャットダウンする
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Add

配列の最後に要素を追加します。

```
bool  Add(
   int  element      // 追加する要素
  )
```

パラメータ

element

[in] 配列に追加される要素の値

戻り値

成功の場合は true、要素が追加できなかった場合は false

例:

```
//--- CArrayInt::Add(int) の例
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

配列の最後に別の配列の要素を追加します。

```
bool  AddArray(
   const int&  src[]      // ソース配列
  )
```

パラメータ

src[]

[in] 追加する要素を持つ配列への参照

戻り値

成功の場合は true、要素が追加できなかった場合は false

例:

```
//--- CArrayInt::AddArray(const int &[]) の例
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 別の配列を追加する
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete array;
      return;
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

AddArray

配列の最後に別の配列の要素を追加します。

```
bool  AddArray(
   const CArrayInt*  src      // ソース配列
  )
```

パラメータ

src

[in] 追加する要素へのポインタ（CArrayInt クラスインスタンス）

戻り値

成功の場合は true、要素が追加できなかった場合は false

例：

```
//--- CArrayInt::AddArray(const CArrayInt*) の例
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソース配列を作成する
   CArrayInt *src=new CArrayInt;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ソース配列に要素を加える
   //--- . にて。.
   //--- 別の配列を追加する
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete src;
      delete array;
      return;
     }
   //--- ソース配列を削除する
   delete src;
```

© 2000–2025, MetaQuotes Ltd.

```
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Insert

配列の指定された位置に要素を挿入します。

```
bool  Insert(
   int  element,     // 挿入する要素
   int  pos          // 位置
  )
```

パラメータ

element

[in] 配列に挿入される要素の値

pos

[in] 配列内の挿入位置

戻り値

成功の場合は true、要素が挿入できなかった場合は false

例:

```
//--- CArrayInt::Insert(int,int) の例
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 要素を挿入する
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

指定された位置に別の配列の要素を挿入します。

```
bool  InsertArray(
   const int&  src[],     // 挿入元の配列
   int         pos        // 位置
  )
```

パラメータ

src[]

[in] 挿入する要素を持つ配列への参照

pos

[in] 配列内の挿入位置

戻り値

成功の場合は true、要素が挿入できなかった場合は false

例:

```
//--- CArrayInt::InsertArray(const int &[],int) の例
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 別の配列を挿入する
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete array;
      return;
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

InsertArray

指定された位置に別の配列の要素を挿入します。

```
bool  InsertArray(
   CArrayInt*  src,      // 挿入元へのポインタ
   int          pos      // 位置
  )
```

パラメータ

src

[in] 挿入する要素へのポインタ（CArrayInt クラスインスタンス）

pos

[in] 配列内の挿入位置

戻り値

成功の場合は true、要素が挿入できなかった場合は false

例：

```
//--- CArrayInt::InsertArray(const CArrayInt*,int) の例
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソース配列を作成する
   CArrayInt *src=new CArrayInt;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ソース配列に要素を加える
   //--- . にて。.
   //--- 別の配列を挿入する
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete src;
      delete array;
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }
   //--- ソース配列を削除する
   delete src;
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

AssignArray

別の配列の要素を複製します。

```
bool  AssignArray(
   const int&  src[]      // ソース配列
  )
```

パラメータ

src[]

[in] 複製する配列要素への参照

戻り値

成功の場合は true、要素が複製できなかった場合は false

例:

```
//--- CArrayInt::AssignArray(const int &[]) の例
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 別の配列を割り当てる
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete array;
      return;
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

AssignArray

別の配列の要素を複製します。

```
bool  AssignArray(
   const CArrayInt*  src      // ソース配列
  )
```

パラメータ

src

[in] 複製する要素へのポインタ（CArrayInt クラスインスタンス）

戻り値

成功の場合は true、要素が複製できなかった場合は false

例：

```
//--- CArrayInt::AssignArray(const CArrayInt*) の例
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソース配列を作成する
   CArrayInt *src  =new CArrayInt;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ソース配列に要素を加える
   //--- . にて。.
   //--- 別の配列を割り当てる
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete src;
      delete array;
      return;
     }
```

© 2000–2025, MetaQuotes Ltd.

```
   //--- 配列が同一
   //--- ソース配列を削除する
   delete src;
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Update

指定された位置の配列の要素を変更します。

```
bool  Update(
   int  pos,          // 位置
   int  element       // 値
  )
```

パラメータ

pos

[in] 変更される要素の配列内での位置

element

[in] 要素の新しい値

戻り値

成功の場合は true、要素が変更できなかった場合は false

例:

```
//--- CArrayInt::Update(int,int) の例
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素を更新する
   if(!array.Update(0,10000))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Shift

配列内の指定された位置から要素を指定されたオフセットに移動します。

```
bool  Shift(
   int  pos,        // 位置
   int  shift       // シフト
  )
```

パラメータ

pos

[in] 配列での移動した要素の位置

shift

[in]（正と負の両方）のシフトの値

戻り値

成功の場合は true、要素が移動できなかった場合は false

例:

```
//--- CArrayInt::Shift(int,int) の例
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素をシフトする
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

配列の指定された位置から要素を削除します。

```
bool  Delete(
   int  pos      // 位置
  )
```

パラメータ

pos

[in] 配列から削除される要素の位置

戻り値

成功の場合は true、要素が削除できなかった場合は false

例:

```
//--- CArrayInt::Delete(int) の例
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素を削除する
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

DeleteRange

配列の指定された位置から要素のグループを削除します。

```
bool  DeleteRange(
   int  from,     // 最初の要素の位置
   int  to        // 最後の要素の位置
  )
```

パラメータ

from

[in] 配列の削除される範囲の最初の要素の位置

to

[in] 配列の削除される範囲の最後の要素の位置

戻り値

成功の場合は true、削除が失敗した場合は false

例:

```
//--- CArrayInt::DeleteRange(int,int) の例
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素を削除する
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

At

配列の指定された位置から要素を取得します。

```
int  At(
   int  pos      // 位置
   ) const
```

パラメータ

pos

[in] 配列での望まれた要素の位置

戻り値

成功の場合は要素の値、存在しない位置での要素の取得が試まれた場合INT_MAX(直近エラーはERR_OUT_OF_RANGE)

注意事項

INT_MAXは配列要素の有効な値である可能性もあるので、この値が取得されたら、常に最終のエラーコードをチェックします。

例:

```
//--- CArrayInt::At(int) の例
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   for(int i=0;i<array.Total();i++)
     {
      int result=array.At(i);
      if(result==INT_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- 配列読み込みエラー
         printf("Get element error");
         delete array;
         return;
        }
      //--- 要素を使用する
      //--- . にて。.
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

CompareArray

配列を別の配列と比較します。

```
bool  CompareArray(
   const int&  src[]      // ソース配列
   ) const
```

パラメータ

src[]

[in] 比較元の要素を持つ配列への参照

戻り値

配列が同一なら true、それ以外の場合は false

例:

```
//--- CArrayInt::CompareArray(const int &[]) の例
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 別の配列と比較する
   int result=array.CompareArray(src);
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

CompareArray

配列を別の配列と比較します。

```
bool  CompareArray(
   const CArrayInt*  src      // ソース配列
   ) const
```

パラメータ

src

[in] 比較する要素へのポインタ（CArrayInt クラスインスタンス）

戻り値

配列が同一なら true、それ以外の場合は false

例：

```
//--- CArrayInt::CompareArray(const CArrayInt*) の例
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソース配列を作成する
   CArrayInt *src=new CArrayInt;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ソース配列に要素を加える
   //--- . にて。.
   //--- 別の配列と比較する
   int result=array.CompareArray(src);
   //--- 配列を削除する
   delete src;
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

InsertSort

ソートされた配列に要素を挿入します。

```
bool  InsertSort(
   int  element      // 挿入する要素
  )
```

パラメータ

element

[in] ソートされた配列に挿入される要素の値

戻り値

成功の場合は true、要素が挿入できなかった場合は false

例:

```
//--- CArrayInt::InsertSort(int) の例
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を挿入する
   if(!array.InsertSort(10000))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Search

ソートされた配列内でサンプルに等しい要素を検索します。

```
int  Search(
   int  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayInt::Search(int) の例
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.Search(10000)!=-1) printf("Element found");
   else                        printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

ソートされた配列内でサンプルより大きい要素を検索します。

```
int  SearchGreat(
   int  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayInt::SearchGreat(int) の例
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchGreat(10000)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

ソートされた配列内でサンプルより小さい要素を検索します。

```
int  SearchLess(
   int  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayInt::SearchLess(int) の例
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchLess(10000)!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

ソートされた配列内でサンプル以上の要素を検索します。

```
int  SearchGreatOrEqual(
   int  element      // 検索する要素
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayInt::SearchGreatOrEqual(int) の例
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchGreatOrEqual(10000)!=-1) printf("Element found");
   else                                    printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

ソートされた配列内でサンプル以下の要素を検索します。

```
int  SearchLessOrEqual(
   int  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayInt::SearchLessOrEqual(int) の例
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchLessOrEqual(10000)!=-1) printf("Element found");
   else                                   printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

ソートされた配列内でサンプルに等しい最初の要素を検索します。

```
int  SearchFirst(
   int  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayInt::  SearchFirst(int) の例
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchFirst(10000)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

ソートされた配列内でサンプルに等しい最後の要素を検索します。

```
int  SearchLast(
   int  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayInt::SearchLast(int) の例
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchLast(10000)!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLinear

配列内でサンプルに等しい要素を検索します。

```
int  SearchLinear(
   int  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

注意事項

このメソッドは、ソートされていない配列では線形探索（または順次検索）アルゴリズムを使用します。

例:

```
//--- CArrayInt::SearchLinear(int) の例
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素を探す
   if(array.SearchLinear(10000)!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

データ配列をファイルに保存します。

```
virtual bool  Save(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に FileOpen(...) で開かれたファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

例:

```
//--- CArrayInt::Save(int) の例
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   int         file_handle;
   CArrayInt *array=new CArrayInt;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 100要素を追加する
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- ファイルを開く
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- ファイル保存エラー
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Load

データ配列をファイルから読み込みます。

```
virtual bool  Load(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に FileOpen(...) で開かれたファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

例:

```
//--- CArrayInt::Load(int) の例
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   int         file_handle;
   CArrayInt *array=new CArrayInt;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ファイルを開く
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- ファイル読み込みエラー
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- 配列要素を使用する
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = %d",i,array.At(i));
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Type

配列の型識別子を取得します。

```
virtual int  Type() const
```

戻り値

配列の型識別子（CArrayInt は 82）

例：

```
//--- CArrayInt::Type() の例
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列の型を取得する
   int type=array.Type();
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

CArrayLong

CArrayLong クラスは long または ulong 型の変数の動的配列のクラスです。

説明

CArrayLong クラスは、long または ulong 型変数の動的配列を操作する関数を提供します。配列要素の追加/挿入/削除、配列のソート、ソートされた配列での検索が実装されています。またファイル操作のメソッドも実装されています。

宣言

```
class CArrayLong : public CArray
```

タイトル

```
#include <Arrays\ArrayLong.mqh>
```

継承階層

CObject
  CArray
    CArrayLong

直接子孫

CRealVolumeBuffer, CTickVolumeBuffer, CTimeBuffer

クラスメソッド

メモリ制御	
Reserve	メモリを割り当てて、配列のサイズを増加します。
Resize	配列の新しい(減少した)サイズを設定します。
Shutdown	配列をクリアして、メモリを完全に開放します。
追加メソッド	
Add	配列の最後に要素を追加します。
AddArray	配列の最後に別の配列の要素を追加します。
AddArray	配列の最後に別の配列の要素を追加します。
Insert	指定された位置に配列の要素を挿入します。
InsertArray	指定された位置に別の配列の要素を挿入します。
InsertArray	指定された位置に別の配列の要素を挿入します。
AssignArray	別の配列の要素を複製します。
AssignArray	別の配列の要素を複製します。

© 2000-2025, MetaQuotes Ltd.

メモリ制御	
更新メソッド	
Update	指定された位置の配列の要素を変更します。
Shift	配列内の指定された位置から要素を指定されたオフセットに移動します。
削除メソッド	
Delete	配列の指定された位置から要素を削除します。
DeleteRange	配列の指定された位置から要素のグループを削除します。
アクセスメソッド	
At	配列の指定された位置から要素を取得します。
比較メソッド	
CompareArray	配列を別の配列と比較します。
CompareArray	配列を別の配列と比較します。
ソートされた配列の操作	
InsertSort	ソートされた配列に要素を挿入します。
Search	ソートされた配列内でサンプルに等しい要素を検索します。
SearchGreat	ソートされた配列内でサンプルより大きい要素を検索します。
SearchLess	ソートされた配列内でサンプルより小さい要素を検索します。
SearchGreatOrEqual	ソートされた配列内でサンプル以上の要素を検索します。
SearchLessOrEqual	ソートされた配列内でサンプル以下の要素を検索します。
SearchFirst	ソートされた配列内でサンプルに等しい最初の要素を検索します。
SearchLast	ソートされた配列内でサンプルに等しい最後の要素を検索します。
SearchLinear	配列内でサンプルに等しい要素を検索します。
入出力	
virtual Save	データ配列をファイルに保存します。
virtual Load	データ配列をファイルから読み込みます。
virtual Type	配列の型識別子を取得します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000–2025, MetaQuotes Ltd.

© 2000–2025, MetaQuotes Ltd.

Reserve

メモリを割り当てて、配列のサイズを増加します。

```
bool  Reserve(
   int  size      // 数
  )
```

パラメータ

size

[in] 配列に追加される要素の数

戻り値

成功の場合は true、ゼロ以下の数を追加しようとする試みがあった場合、または配列が増加しなかった場合は false

注意事項

メモリの断片化を低減するために、配列のサイズは、以前に Step(int) によって決定されたステップのまたはデフォルトの16ステップを使用して変更されます。

例:

```
//--- CArrayLong::Reserve(int) の例
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- メモリを予約する
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

配列の新しい(減少した)サイズを設定します。

```
bool  Resize(
   int  size      // サイズ
  )
```

パラメータ

size

[in] 配列の新しいサイズ

戻り値

成功の場合はtrue、ゼロ以下のサイズを設定しようとする試みがあった場合はfalse

注意事項

配列のサイズ変更はメモリ使用の最適化に役立ちます。右側の余分な要素が失われます。メモリの断片化を低減するために、配列のサイズは、以前に Step(int) によって決定されたステップのまたはデフォルトの16ステップを使用して変更されます。

例:

```
//--- CArrayLong::Resize(int) の例
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列のサイズを変更する
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

配列をクリアして、メモリを完全に開放します。

```
bool  Shutdown()
```

戻り値

成功の場合は true、エラーが発生した場合は false

例：

```
//--- CArrayLong::Shutdown() の例
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をシャットダウンする
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Add

配列の最後に要素を追加します。

```
bool  Add(
   long  element      // 追加する要素
  )
```

パラメータ

element

[in] 配列に追加される要素の値

戻り値

成功の場合は true、要素が追加できなかった場合は false

例:

```
//--- CArrayLong::Add(long) の例
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

配列の最後に別の配列の要素を追加します。

```
bool  AddArray(
   const long&  src[]      // ソース配列
  )
```

パラメータ

src[]

[in] 追加する要素を持つ配列への参照

戻り値

成功の場合は true、要素が追加できなかった場合は false

例:

```
//--- CArrayLong::AddArray(const long &[]) の例
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 別の配列を追加する
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete array;
      return;
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

AddArray

配列の最後に別の配列の要素を追加します。

```
bool  AddArray(
   const CArrayLong*  src      // ソース配列
  )
```

パラメータ

src

[in] 追加する要素へのポインタ（CArrayLong クラスインスタンス）

戻り値

成功の場合は true、要素が追加できなかった場合は false

例：

```
//--- CArrayLong::AddArray(const CArrayLong*) の例
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソース配列を作成する
   CArrayLong *src=new CArrayLong;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ソース配列に要素を加える
   //--- . にて。.
   //--- 別の配列を追加する
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete src;
      delete array;
      return;
     }
   //--- ソース配列を削除する
   delete src;
```

© 2000–2025, MetaQuotes Ltd.

```
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Insert

配列の指定された位置に要素を挿入します。

```
bool  Insert(
   long  element,     // 挿入する要素
   int   pos          // 位置
  )
```

パラメータ

element

[in] 配列に挿入される要素の値

pos

[in] 配列内の挿入位置

戻り値

成功の場合は true、要素が挿入できなかった場合は false

例:

```
//--- CArrayLong::Insert(long,int) の例
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 要素を挿入する
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

別の配列の要素を指定された位置に挿入します。

```
bool  InsertArray(
   const long&  src[],     // 挿入元の配列
   int          pos        // 位置
  )
```

パラメータ

src[]

[in] 挿入する要素を持つ配列への参照

pos

[in] 配列内の挿入位置

戻り値

成功の場合は true、要素が挿入できなかった場合は false

例:

```
//--- CArrayLong::InsertArray(const long &[],int) の例
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 別の配列を挿入する
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete array;
      return;
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

別の配列の要素を指定された位置に挿入します。

```
bool  InsertArray(
   CArrayLong*  src,      // 挿入元へのポインタ
   int           pos      // 位置
  )
```

パラメータ

src

[in] 挿入する要素へのポインタ（CArrayLong クラスインスタンス）

pos

[in] 配列内の挿入位置

戻り値

成功の場合は true、要素が挿入できなかった場合は false

例：

```
//--- CArrayLong::InsertArray(const CArrayLong*,int) の例
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソース配列を作成する
   CArrayLong *src=new CArrayLong;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ソース配列に要素を加える
   //--- . にて。.
   //--- 別の配列を挿入する
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete src;
      delete array;
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }
   //--- ソース配列を削除する
   delete src;
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

AssignArray

別の配列の要素を複製します。

```
bool  AssignArray(
   const long&  src[]      // ソース配列
  )
```

パラメータ

src[]

[in] 複製する配列要素への参照

戻り値

成功の場合は true、要素が複製できなかった場合は false

例:

```
//--- CArrayLong::AssignArray(const long &[]) の例
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 別の配列を割り当てる
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete array;
      return;
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

別の配列の要素を複製します。

```
bool  AssignArray(
   const CArrayLong*  src      // ソース配列
  )
```

パラメータ

src

[in] 複製する要素へのポインタ（CArrayLong クラスインスタンス）

戻り値

成功の場合は true、要素が複製できなかった場合は false

例：

```
//--- CArrayLong::AssignArray(const CArrayLong*) の例
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソース配列を作成する
   CArrayLong *src  =new CArrayLong;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ソース配列に要素を加える
   //--- . にて。.
   //--- 別の配列を割り当てる
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete src;
      delete array;
      return;
     }
   //--- 配列が同一
   //--- ソース配列を削除する
```

© 2000–2025, MetaQuotes Ltd.

```
   delete src;
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Update

指定された位置の配列の要素を変更します。

```
bool  Update(
   int   pos,          // 位置
   long  element       // 値
  )
```

パラメータ

pos

[in] 変更される要素の配列内での位置

element

[in] 要素の新しい値

戻り値

成功の場合は true、要素が変更できなかった場合は false

例:

```
//--- CArrayLong::Update(int,long) の例
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素を更新する
   if(!array.Update(0,1000000))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shift

配列内の指定された位置から要素を指定されたオフセットに移動します。

```
bool  Shift(
   int  pos,        // 位置
   int  shift       // シフト
  )
```

パラメータ

pos

[in] 配列での移動した要素の位置

shift

[in]（正と負の両方）のシフトの値

戻り値

成功の場合は true、要素が移動できなかった場合は false

例:

```
//--- CArrayLong::Shift(int,int) の例
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素をシフトする
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

配列の指定された位置から要素を削除します。

```
bool  Delete(
   int  pos      // 位置
  )
```

パラメータ

pos

[in] 配列から削除される要素の位置

戻り値

成功の場合は true、要素が削除できなかった場合は false

例:

```
//--- CArrayLong::Delete(int) の例
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素を削除する
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

DeleteRange

配列の指定された位置から要素のグループを削除します。

```
bool  DeleteRange(
   int  from,     // 最初の要素の位置
   int  to        // 最後の要素の位置
  )
```

パラメータ

from

[in] 配列の削除される範囲の最初の要素の位置

to

[in] 配列の削除される範囲の最後の要素の位置

戻り値

成功の場合は true、削除が失敗した場合は false

例:

```
//--- CArrayLong::DeleteRange(int,int) の例
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素を削除する
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

At

配列の指定された位置から要素を取得します。

```
long  At(
   int  pos      // 位置
   ) const
```

パラメータ

pos

[in] 配列での望まれた要素の位置

戻り値

成功の場合は要素の値、存在しない位置での要素の取得が試まれた場合 LONG_MAX(直近エラーは ERR_OUT_OF_RANGE)

注意事項

LONG_MAX は配列要素の有効な値である可能性もあるので、この値が取得されたら、常に最終のエラーコードをチェックします。

例:

```
//--- CArrayLong::At(int) の例
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   for(int i=0;i<array.Total();i++)
     {
      long result=array.At(i);
      if(result==LONG_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- 配列読み込みエラー
         printf("Get element error");
         delete array;
         return;
        }
      //--- 要素を使用する
      //--- . にて。.
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

CompareArray

配列を別の配列と比較します。

```
bool  CompareArray(
   const long&  src[]      // ソース配列
   ) const
```

パラメータ

src[]

[in] 比較元の要素を持つ配列への参照

戻り値

配列が同一なら true、それ以外の場合は false

例:

```
//--- CArrayLong::CompareArray(const long &[]) の例
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 別の配列と比較する
   int result=array.CompareArray(src);
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

CompareArrayconst

配列を別の配列と比較します。

```
bool  CompareArrayconst(
   const CArrayLong*  src      // ソース配列
   ) const
```

パラメータ

src

[in] 比較する要素へのポインタ（CArrayLon クラスインスタンス）

戻り値

配列が同一なら true、それ以外の場合は false

例：

```
//--- CArrayLong::CompareArray(const CArrayLong*) の例
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソース配列を作成する
   CArrayLong *src=new CArrayLong;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ソース配列に要素を加える
   //--- . にて。.
   //--- 別の配列と比較する
   int result=array.CompareArray(src);
   //--- 配列を削除する
   delete src;
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

InsertSort

ソートされた配列に要素を挿入します。

```
bool  InsertSort(
   long  element      // 挿入する要素
  )
```

パラメータ

element

[in] ソートされた配列に挿入される要素の値

戻り値

成功の場合は true、要素が挿入できなかった場合は false

例:

```
//--- CArrayLong::InsertSort(long) の例
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を挿入する
   if(!array.InsertSort(1000000))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Search

ソートされた配列内でサンプルに等しい要素を検索します。

```
int  Search(
   long  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayLong::Search(long) の例
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.Search(1000000)!=-1) printf("Element found");
   else                          printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

ソートされた配列内でサンプルより大きい要素を検索します。

```
int  SearchGreat(
   long  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayLong::SearchGreat(long) の例
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchGreat(1000000)!=-1) printf("Element found");
   else                               printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

ソートされた配列内でサンプルより小さい要素を検索します。

```
int  SearchLess(
   long  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayLong::SearchLess(long) の例
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchLess(1000000)!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

ソートされた配列内でサンプル以上の要素を検索します。

```
int  SearchGreatOrEqual(
   long  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayLong::SearchGreatOrEqual(long) の例
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchGreatOrEqual(1000000)!=-1) printf("Element found");
   else                                      printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

ソートされた配列内でサンプル以下の要素を検索します。

```
int  SearchLessOrEqual(
   long  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayLong::SearchLessOrEqual(long) の例
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchLessOrEqual(1000000)!=-1) printf("Element found");
   else                                     printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

ソートされた配列内でサンプルに等しい最初の要素を検索します。

```
int  SearchFirst(
   long  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayLong::SearchFirst(long) の例
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchFirst(1000000)!=-1) printf("Element found");
   else                               printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

ソートされた配列内でサンプルに等しい最後の要素を検索します。

```
int  SearchLast(
   long  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayLong::SearchLast(long) の例
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchLast(1000000)!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLinear

ソートされた配列内でサンプルに等しい要素を検索します。

```
int  SearchLinear(
   long  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

注意事項

このメソッドは、ソートされていない配列では線形探索（または順次検索）アルゴリズムを使用します。

例:

```
//--- CArrayLong::SearchLinear(long) の例
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素を探す
   if(array.SearchLinear(1000000)!=-1) printf("Element found");
   else                                printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

データ配列をファイルに保存します。

```
virtual bool  Save(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に FileOpen(...) で開かれたファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

例:

```
//--- CArrayLong::Save(int) の例
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayLong *array=new CArrayLong;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 100要素を追加する
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- ファイルを開く
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- ファイル保存エラー
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Load

データ配列をファイルから読み込みます。

```
virtual bool  Load(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に FileOpen(...) で開かれたファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

例:

```
//--- CArrayLong::Load(int) の例
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayLong *array=new CArrayLong;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ファイルを開く
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- ファイル読み込みエラー
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- 配列要素を使用する
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = %I64",i,array.At(i));
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Type

配列の型識別子を取得します。

```
virtual int  Type() const
```

戻り値

配列の型識別子（CArrayLong は 84）

例:

```
//--- CArrayLong::Type() の例
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列の型を取得する
   int type=array.Type();
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

CArrayFloat

CArrayFloat クラスは float 型の変数の動的配列のクラスです。

説明

CArrayFloat クラスは、float 型変数の動的配列を操作する関数を提供します。ソートや検索、配列要素の追加、挿入、及び削除や、配列のソートや検索が提供されています。また、ファイルを操作するためのメソッドを実装されています。

宣言

```
class CArrayFloat : public CArray
```

タイトル

```
#include <Arrays\ArrayFloat.mqh>
```

継承階層

CObject
  CArray
    CArrayFloat

クラスメソッド

属性	
Delta	比較の許容範囲を設定します。
メモリ制御	
Reserve	メモリを割り当てて、配列のサイズを増加します。
Resize	配列の新しい(減少した)サイズを設定します。
Shutdown	配列をクリアして、メモリを完全に開放します。
追加メソッド	
Add	配列の最後に要素を追加します。
AddArray	配列の最後に別の配列の要素を追加します。
AddArray	配列の最後に別の配列の要素を追加します。
Insert	指定された位置に配列の要素を挿入します。
InsertArray	指定された位置に別の配列の要素を挿入します。
InsertArray	指定された位置に別の配列の要素を挿入します。
AssignArray	別の配列の要素を複製します。
AssignArray	別の配列の要素を複製します。

© 2000–2025, MetaQuotes Ltd.

属性	
更新メソッド	
Update	指定された位置の配列の要素を変更します。
Shift	配列内の指定された位置から要素を指定されたオフセットに移動します。
Delete	配列の指定された位置から要素を削除します。
DeleteRange	配列の指定された位置から要素のグループを削除します。
アクセスメソッド	
At	配列の指定された位置から要素を取得します。
比較メソッド	
CompareArray	配列を別の配列と比較します。
CompareArray	配列を別の配列と比較します。
ソートされた配列の操作	
InsertSort	ソートされた配列に要素を挿入します。
Search	ソートされた配列内でサンプルに等しい要素を検索します。
SearchGreat	ソートされた配列内でサンプルより大きい要素を検索します。
SearchLess	ソートされた配列内でサンプルより小さい要素を検索します。
SearchGreatOrEqual	ソートされた配列内でサンプル以上の要素を検索します。
SearchLessOrEqual	ソートされた配列内でサンプル以下の要素を検索します。
SearchFirst	ソートされた配列内でサンプルに等しい最初の要素を検索します。
SearchLast	ソートされた配列内でサンプルに等しい最後の要素を検索します。
SearchLinear	配列内でサンプルに等しい要素を検索します。
入出力	
virtual Save	データ配列をファイルに保存します。
virtual Load	データ配列をファイルから読み込みます。
virtual Type	配列の型識別子を取得します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000–2025, MetaQuotes Ltd.

Delta

比較の許容範囲を設定します。

```
void  Delta(
   float  delta      // 許容範囲
  )
```

パラメータ

delta

[in] 新しい比較の許容範囲

戻り値

None

注意事項

比較の許容範囲は、検索に使用されます。値は、その差が許容範囲以下の場合は等しいと見なされます。デフォルトの許容範囲は 0.0 です。

例:

```
//--- CArrayFloat::Delta(float) の例
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 比較のバリエーションを設定する
   array.Delta(0.001);
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Reserve

メモリを割り当てて、配列のサイズを増加します。

```
bool  Reserve(
   int  size      // 数
  )
```

パラメータ

size

[in] 配列に追加される要素の数

戻り値

成功の場合は true、ゼロ以下の数を追加しようとする試みがあった場合、または配列が増加しなかった場合は false

注意事項

メモリの断片化を低減するために、配列のサイズは、以前に Step(int) によって決定されたステップのまたはデフォルトの16ステップを使用して変更されます。

例:

```
//--- CArrayFloat::Reserve(int) の例
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- メモリを予約する
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

配列の新しい(減少した)サイズを設定します。

```
bool  Resize(
   int  size      // サイズ
  )
```

パラメータ

size

[in] 配列の新しいサイズ

戻り値

成功の場合はtrue、ゼロ以下のサイズを設定しようとする試みがあった場合はfalse

注意事項

配列のサイズ変更はメモリ使用の最適化に役立ちます。右側の余分な要素が失われます。メモリの断片化を低減するために、配列のサイズは、以前に Step(int) によって決定されたステップのまたはデフォルトの16ステップを使用して変更されます。

例:

```
//--- CArrayFloat::Resize(int) の例
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列のサイズを変更する
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

配列をクリアして、メモリを完全に開放します。

```
bool  Shutdown()
```

戻り値

成功の場合は true、エラーが発生した場合は false

例：

```
//--- CArrayFloat::Shutdown() の例
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をシャットダウンする
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Add

配列の最後に要素を追加します。

```
bool  Add(
   float  element      // 追加する要素
  )
```

パラメータ

element

[in] 配列に追加される要素の値

戻り値

成功の場合は true、要素が追加できなかった場合は false

例:

```
//--- CArrayFloat::Add(float) の例
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

配列の最後に別の配列の要素を追加します。

```
bool  AddArray(
   const float&  src[]      // ソース配列
  )
```

パラメータ

src[]

[in] 追加する要素を持つ配列への参照

戻り値

成功の場合は true、要素が追加できなかった場合は false

例:

```
//--- CArrayFloat::AddArray(const float &[]) の例
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 別の配列を追加する
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete array;
      return;
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

AddArray

配列の最後に別の配列の要素を追加します。

```
bool  AddArray(
   const CArrayFloat*  src      // ソース配列
  )
```

パラメータ

src

[in] 追加する要素へのポインタ（CArrayFloat クラスインスタンス）

戻り値

成功の場合は true、要素が追加できなかった場合は false

例：

```
//--- CArrayFloat::AddArray(const CArrayFloat*) の例
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソース配列を作成する
   CArrayFloat *src=new CArrayFloat;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ソース配列に要素を加える
   //--- . にて。.
   //--- 別の配列を追加する
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete src;
      delete array;
      return;
     }
   //--- ソース配列を削除する
   delete src;
```

© 2000–2025, MetaQuotes Ltd.

```
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Insert

配列の指定された位置に要素を挿入します。

```
bool  Insert(
   float  element,      // 挿入する要素
   int    pos           // 位置
  )
```

パラメータ

element

[in] 配列に挿入される要素の値

pos

[in] 配列内の挿入位置

戻り値

成功の場合は true、要素が挿入できなかった場合は false

例:

```
//--- CArrayFloat::Insert(float,int) の例
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 要素を挿入する
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

指定された位置に別の配列の要素を挿入します。

```
bool  InsertArray(
   const float&  src[],     // 挿入元の配列
   int           pos        // 位置
  )
```

パラメータ

src[]

[in] 挿入する要素を持つ配列への参照

pos

[in] 配列内の挿入位置

戻り値

成功の場合は true、要素が挿入できなかった場合は false

例:

```
//--- CArrayFloat::InsertArray(const float &[],int) の例
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 別の配列を挿入する
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete array;
      return;
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

InsertArray

指定された位置に別の配列の要素を挿入します。

```
bool  InsertArray(
   CArrayFloat*  src,      // 挿入元へのポインタ
   int           pos       // 位置
  )
```

パラメータ

src

[in] 挿入する要素へのポインタ（CArrayFloat クラスインスタンス）

pos

[in] 配列内の挿入位置

戻り値

成功の場合は true、要素が挿入できなかった場合は false

例：

```
//--- CArrayFloat::InsertArray(const CArrayFloat*,int) の例
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソース配列を作成する
   CArrayFloat *src=new CArrayFloat;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ソース配列に要素を加える
   //--- . にて。.
   //--- 別の配列を挿入する
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete src;
      delete array;
```

© 2000–2025, MetaQuotes Ltd.

```
      return;
     }
   //--- ソース配列を削除する
   delete src;
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

AssignArray

別の配列の要素を複製します。

```
bool  AssignArray(
   const float&  src[]      // ソース配列
  )
```

パラメータ

src[]

[in] 複製する配列要素への参照

戻り値

成功の場合は true、要素が複製できなかった場合は false

例:

```
//--- CArrayFloat::AssignArray(const float &[]) の例
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 別の配列を割り当てる
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete array;
      return;
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

AssignArray

別の配列の要素を複製します。

```
bool  AssignArray(
   const CArrayFloat*  src      // ソース配列
  )
```

パラメータ

src

[in] 複製する要素へのポインタ（CArrayFloat クラスインスタンス）

戻り値

成功の場合は true、要素が複製できなかった場合は false

例：

```
//--- CArrayFloat::AssignArray(const CArrayFloat*) の例
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソース配列を作成する
   CArrayFloat *src  =new CArrayFloat;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ソース配列に要素を加える
   //--- . にて。.
   //--- 別の配列を割り当てる
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete src;
      delete array;
      return;
     }
   //--- 配列が同一
   //--- ソース配列を削除する
```

© 2000–2025, MetaQuotes Ltd.

```
   delete src;
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Update

指定された位置の配列の要素を変更します。

```
bool  Update(
   int    pos,          // 位置
   float  element       // 値
  )
```

パラメータ

pos

[in] 変更される要素の配列内での位置

element

[in] 要素の新しい値

戻り値

成功の場合は true、要素が変更できなかった場合は false

例:

```
//--- CArrayFloat::Update(int,float) の例
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素を更新する
   if(!array.Update(0,100.0))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shift

配列内の指定された位置から要素を指定されたオフセットに移動します。

```
bool  Shift(
   int  pos,        // 位置
   int  shift       // シフト
  )
```

パラメータ

pos

[in] 配列での移動した要素の位置

shift

[in]（正と負の両方）のシフトの値

戻り値

成功の場合は true、要素が移動できなかった場合は false

例:

```
//--- CArrayFloat::Shift(int,int) の例
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素をシフトする
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

配列の指定された位置から要素を削除します。

```
bool  Delete(
   int  pos      // 位置
  )
```

パラメータ

pos

[in] 配列から削除される要素の位置

戻り値

成功の場合は true、要素が削除できなかった場合は false

例:

```
//--- CArrayFloat::Delete(int) の例
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素を削除する
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

DeleteRange

配列の指定された位置から要素のグループを削除します。

```
bool  DeleteRange(
   int  from,     // 最初の要素の位置
   int  to        // 最後の要素の位置
  )
```

パラメータ

from

[in] 配列の削除される範囲の最初の要素の位置

to

[in] 配列の削除される範囲の最後の要素の位置

戻り値

成功の場合は true、削除が失敗した場合は false

例:

```
//--- CArrayFloat::DeleteRange(int,int) の例
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素を削除する
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

At

配列の指定された位置から要素を取得します。

```
float  At(
   int  pos      // 位置
   ) const
```

パラメータ

pos

[in] 配列での望まれた要素の位置

戻り値

成功の場合は要素の値、存在しない位置での要素の取得が試まれた場合FLT_MAX(直近エラーはERR_OUT_OF_RANGE)

注意事項

FLT_MAX は配列要素の有効な値である可能性もあるので、この値が取得されたら、常に最終のエラーコードをチェックします。

例:

```
//--- CArrayFloat::At(int) の例
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   for(int i=0;i<array.Total();i++)
     {
      float result=array.At(i);
      if(result==FLT_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- 配列読み込みエラー
         printf("Get element error");
         delete array;
         return;
        }
      //--- 要素を使用する
      //--- . にて。.
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

CompareArray

配列を別の配列と比較します。

```
bool  CompareArray(
   const float&  src[]      // ソース配列
   ) const
```

パラメータ

src[]

[in] 比較元の要素を持つ配列への参照

戻り値

配列が同一なら true、それ以外の場合は false

例:

```
//--- CArrayFloat::CompareArray(const float &[]) の例
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 別の配列と比較する
   int result=array.CompareArray(src);
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArrayconst

別の配列の要素を複製します。

```
bool  AssignArrayconst(
   const CArrayFloat*  src      // ソース配列
   ) const
```

パラメータ

src

[in] 複製する要素へのポインタ（CArrayFloat クラスインスタンス）

戻り値

成功の場合は true、要素が複製できなかった場合は false

例：

```
//--- CArrayFloat::CompareArray(const CArrayFloat*) の例
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソース配列を作成する
   CArrayFloat *src=new CArrayFloat;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ソース配列に要素を加える
   //--- . にて。.
   //--- 別の配列と比較する
   int result=array.CompareArray(src);
   //--- 配列を削除する
   delete src;
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

InsertSort

ソートされた配列に要素を挿入します。

```
bool  InsertSort(
   float  element      // 挿入する要素
  )
```

パラメータ

element

[in] ソートされた配列に挿入される要素の値

戻り値

成功の場合は true、要素が挿入できなかった場合は false

例:

```
//--- CArrayFloat::InsertSort(float) の例
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を挿入する
   if(!array.InsertSort(100.0))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Search

ソートされた配列内でサンプルに等しい要素を検索します。

```
int  Search(
   float  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayFloat::Search(float) の例
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.Search(100.0)!=-1) printf("Element found");
   else                        printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

ソートされた配列内でサンプルより大きい要素を検索します。

```
int  SearchGreat(
   float  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayFloat::SearchGreat(float) の例
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchGreat(100.0)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

ソートされた配列内でサンプルより小さい要素を検索します。

```
int  SearchLess(
   float  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayFloat::  SearchLess(float) の例
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchLess(100.0)!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

ソートされた配列内でサンプル以上の要素を検索します。

```
int  SearchGreatOrEqual(
   float  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayFloat::SearchGreatOrEqual(float) の例
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchGreatOrEqual(100.0)!=-1) printf("Element found");
   else                                    printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

ソートされた配列内でサンプル以下の要素を検索します。

```
int  SearchLessOrEqual(
   float  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayFloat::SearchLessOrEqual(float) の例
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchLessOrEqual(100.0)!=-1) printf("Element found");
   else                                   printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

ソートされた配列内でサンプルに等しい最初の要素を検索します。

```
int  SearchFirst(
   float  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayFloat::SearchFirst(float) の例
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchFirst(100.0)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

ソートされた配列内でサンプルに等しい最後の要素を検索します。

```
int  SearchLast(
   float  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayFloat::SearchLast(float) の例
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchLast(100.0)!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLinear

配列内でサンプルに等しい要素を検索します。

```
int  SearchLinear(
   float  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

注意事項

このメソッドは、ソートされていない配列では線形探索（または順次検索）アルゴリズムを使用します。

例:

```
//--- CArrayFloat::SearchLinear(float) の例
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素を探す
   if(array.SearchLinear(100.0)!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

データ配列をファイルに保存します。

```
virtual bool  Save(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に FileOpen(...) で開かれたファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

例:

```
//--- CArrayFloat::Save(int) の例
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 100要素を追加する
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- ファイルを開く
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- ファイル保存エラー
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Load

データ配列をファイルから読み込みます。

```
virtual bool  Load(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に FileOpen(...) で開かれたファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

例:

```
//--- CArrayFloat::Load(int) の例
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ファイルを開く
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- ファイル読み込みエラー
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- 配列要素を使用する
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = %f",i,array.At(i));
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Type

配列の型識別子を取得します。

```
virtual int  Type() const
```

戻り値

配列の型識別子（CArrayFloat は 87）

例：

```
//--- CArrayFloat::Type() の例
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列の型を取得する
   int type=array.Type();
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

CArrayDouble

CArrayDouble クラスは double 型の変数の動的配列のクラスです。

説明

CArrayDouble クラスは、double 型変数の動的配列を操作する関数を提供します。ソートや検索、配列要素の追加、挿入、及び削除や、配列のソートや検索が提供されています。また、ファイルを操作するためのメソッドを実装されています。

宣言

```
class CArrayDouble : public CArray
```

タイトル

```
#include <Arrays\ArrayDouble.mqh>
```

継承階層

CObject
  CArray
    CArrayDouble

直接子孫

CDoubleBuffer

クラスメソッド

属性	
Delta	比較の許容範囲を設定します。
メモリ制御	
Reserve	メモリを割り当てて、配列のサイズを増加します。
Resize	配列の新しい（減少した）サイズを設定します。
Shutdown	配列をクリアして、メモリを完全に開放します。
追加メソッド	
Add	配列の最後に要素を追加します。
AddArray	配列の最後に別の配列の要素を追加します。
AddArray	配列の最後に別の配列の要素を追加します。
Insert	指定された位置に配列の要素を挿入します。
InsertArray	指定された位置に別の配列の要素を挿入します。
InsertArray	指定された位置に別の配列の要素を挿入します。

© 2000–2025, MetaQuotes Ltd.

属性	
AssignArray	別の配列の要素を複製します。
AssignArray	別の配列の要素を複製します。
更新メソッド	
Update	指定された位置の配列の要素を変更します。
Shift	配列内の指定された位置から要素を指定されたオフセットに移動します。
削除メソッド	
Delete	配列の指定された位置から要素を削除します。
DeleteRange	配列の指定された位置から要素のグループを削除します。
アクセスメソッド	
At	配列の指定された位置から要素を取得します。
比較メソッド	
CompareArray	配列を別の配列と比較します。
CompareArray	配列を別の配列と比較します。
最安値と最高値の検索	
Minimum	配列の指定された範囲内の最小要素のインデックスを取得します。
Maximum	配列の指定された範囲内の最大要素のインデックスを取得します。
ソートされた配列の操作	
InsertSort	ソートされた配列に要素を挿入します。
Search	ソートされた配列内でサンプルに等しい要素を検索します。
SearchGreat	ソートされた配列内でサンプルより大きい要素を検索します。
SearchLess	ソートされた配列内でサンプルより小さい要素を検索します。
SearchGreatOrEqual	ソートされた配列内でサンプル以上の要素を検索します。
SearchLessOrEqual	ソートされた配列内でサンプル以下の要素を検索します。
SearchFirst	ソートされた配列内でサンプルに等しい最初の要素を検索します。
SearchLast	ソートされた配列内でサンプルに等しい最後の要素を検索します。
SearchLinear	配列内でサンプルに等しい要素を検索します。
入出力	
virtual Save	データ配列をファイルに保存します。
virtual Load	データ配列をファイルから読み込みます。
virtual Type	配列の型識別子を取得します。

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000–2025, MetaQuotes Ltd.

Delta

比較の許容範囲を設定します。

```
void  Delta(
   double  delta      // 許容範囲
  )
```

パラメータ

delta

[in] 新しい比較の許容範囲

戻り値

No

注意事項

比較の許容範囲は、検索に使用されます。値は、その差が許容範囲以下の場合は等しいと見なされます。デフォルトの許容範囲は 0.0 です。

例:

```
//--- CArrayDouble::Delta(double) の例
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 比較のバリエーションを設定する
   array.Delta(0.001);
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Reserve

メモリを割り当てて、配列のサイズを増加します。

```
bool  Reserve(
   int  size      // 数
  )
```

パラメータ

size

[in] 配列に追加される要素の数

戻り値

成功の場合は true、ゼロ以下の数を追加しようとする試みがあった場合、または配列が増加しなかった場合は false

注意事項

メモリの断片化を低減するために、配列のサイズは、以前に Step(int) によって決定されたステップのまたはデフォルトの16ステップを使用して変更されます。

例:

```
//--- CArrayDouble::Reserve(int) の例
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- メモリを予約する
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

配列の新しい(減少した)サイズを設定します。

```
bool  Resize(
   int  size      // サイズ
  )
```

パラメータ

size

[in] 配列の新しいサイズ

戻り値

成功の場合はtrue、ゼロ以下のサイズを設定しようとする試みがあった場合はfalse

注意事項

配列のサイズ変更はメモリ使用の最適化に役立ちます。右側の余分な要素が失われます。メモリの断片化を低減するために、配列のサイズは、以前に Step(int) によって決定されたステップのまたはデフォルトの16ステップを使用して変更されます。

例:

```
//--- CArrayDouble::Resize(int) の例
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列のサイズを変更する
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

配列をクリアして、メモリを完全に開放します。

```
bool  Shutdown()
```

戻り値

成功の場合は true、エラーが発生した場合は false

例：

```
//--- CArrayDouble::Shutdown() の例
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をシャットダウンする
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Add

配列の最後に要素を追加します。

```
bool  Add(
   double  element      // 追加する要素
  )
```

パラメータ

element

[in] 配列に追加される要素の値

戻り値

成功の場合は true、要素が追加できなかった場合は false

例:

```
//--- CArrayDouble::Add(double) の例
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

配列の最後に別の配列の要素を追加します。

```
bool  AddArray(
   const double&  src[]      // ソース配列
  )
```

パラメータ

src[]

[in] 追加する要素を持つ配列への参照

戻り値

成功の場合は true、要素が追加できなかった場合は false

例:

```
//--- CArrayDouble::AddArray(const double &[]) の例
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 別の配列を追加する
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete array;
      return;
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

AddArray

配列の最後に別の配列の要素を追加します。

```
bool  AddArray(
   const CArrayDouble*  src      // ソース配列
  )
```

パラメータ

src

[in] 追加する要素へのポインタ（CArrayDouble クラスインスタンス）

戻り値

成功の場合は true、要素が追加できなかった場合は false

例：

```
//--- CArrayDouble::AddArray(const CArrayDouble*) の例
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソース配列を作成する
   CArrayDouble *src=new CArrayDouble;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ソース配列に要素を加える
   //--- . にて。.
   //--- 別の配列を追加する
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete src;
      delete array;
      return;
     }
   //--- ソース配列を削除する
   delete src;
```

© 2000–2025, MetaQuotes Ltd.

```
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Insert

配列の指定された位置に要素を挿入します。

```
bool  Insert(
   double  element,     // 挿入する要素
   int     pos          // 位置
  )
```

パラメータ

element

[in] 配列に挿入される要素の値

pos

[in] 配列内の挿入位置

戻り値

成功の場合は true、要素が挿入できなかった場合は false

例:

```
//--- CArrayDouble::Insert(double,int) の例
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 要素を挿入する
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

指定された位置に別の配列の要素を挿入します。

```
bool  InsertArray(
   const double&  src[],      // ソース配列
   int            pos         // 位置
  )
```

パラメータ

src[]

[in] 挿入する要素を持つ配列への参照

pos

[in] 配列内の挿入位置

戻り値

成功の場合は true、要素が挿入できなかった場合は false

例:

```
//--- CArrayDouble::InsertArray(const double &[],int) の例
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 別の配列を挿入する
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete array;
      return;
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

InsertArray

指定された位置に別の配列の要素を挿入します。

```
bool  InsertArray(
   CArrayDouble*  src,      // ソース配列
   int            pos       // 位置
  )
```

パラメータ

src

[in] 挿入する要素へのポインタ（CArrayDouble クラスインスタンス）

pos

[in] 配列内の挿入位置

戻り値

成功の場合は true、要素が挿入できなかった場合は false

例：

```
//--- CArrayDouble::InsertArray(const CArrayDouble*,int) の例
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソース配列を作成する
   CArrayDouble *src=new CArrayDouble;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ソース配列に要素を加える
   //--- . にて。.
   //--- 別の配列を挿入する
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete src;
      delete array;
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }
   //--- ソース配列を削除する
   delete src;
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

AssignArray

別の配列の要素を複製します。

```
bool  AssignArray(
   const double&  src[]      // ソース配列
  )
```

パラメータ

src[]

[in] 複製する配列要素への参照

戻り値

成功の場合は true、要素が複製できなかった場合は false

例:

```
//--- CArrayDouble::AssignArray(const double &[]) の例
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 別の配列を割り当てる
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete array;
      return;
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

AssignArray

別の配列の要素を複製します。

```
bool  AssignArray(
   const CArrayDouble*  src      // ソース配列
  )
```

パラメータ

src

[in] 複製する要素へのポインタ（CArrayDouble クラスインスタンス）

戻り値

成功の場合は true、要素が複製できなかった場合は false

例：

```
//--- CArrayDouble::AssignArray(const CArrayDouble*) の例
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソース配列を作成する
   CArrayDouble *src  =new CArrayDouble;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ソース配列に要素を加える
   //--- . にて。.
   //--- 別の配列を割り当てる
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete src;
      delete array;
      return;
     }
   //--- 配列が同一
   //--- ソース配列を削除する
```

© 2000–2025, MetaQuotes Ltd.

```
   delete src;
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Update

指定された位置の配列の要素を変更します。

```
bool  Update(
   int     pos,          // 位置
   double  element       // 値
  )
```

パラメータ

pos

[in] 変更される要素の配列内での位置

element

[in] 新しい値

戻り値

成功の場合は true、要素が変更できなかった場合は false

例:

```
//--- CArrayDouble::Update(int,double) の例
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素を更新する
   if(!array.Update(0,100.0))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shift

配列内の指定された位置から要素を指定されたオフセットに移動します。

```
bool  Shift(
   int  pos,        // 位置
   int  shift       // シフト
  )
```

パラメータ

pos

[in] 配列での移動した要素の位置

shift

[in]（正と負の両方）のシフトの値

戻り値

成功の場合は true、要素が移動できなかった場合は false

例：

```
//--- CArrayDouble::Shift(int,int) の例
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素をシフトする
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

配列の指定された位置から要素を削除します。

```
bool  Delete(
   int  pos      // 位置
  )
```

パラメータ

pos

[in] 配列から削除される要素の位置

戻り値

成功の場合は true、要素が削除できなかった場合は false

例:

```
//--- CArrayDouble::Delete(int) の例
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素を削除する
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

DeleteRange

配列の指定された位置から要素のグループを削除します。

```
bool  DeleteRange(
   int  from,     // 最初の要素の位置
   int  to        // 最後の要素の位置
  )
```

パラメータ

from

[in] 配列の削除される範囲の最初の要素の位置

to

[in] 配列の削除される範囲の最後の要素の位置

戻り値

成功の場合は true、削除が失敗した場合は false

例:

```
//--- CArrayDouble::DeleteRange(int,int) の例
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素を削除する
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

At

配列の指定された位置から要素を取得します。

```
double  At(
   int  pos      // 位置
   ) const
```

パラメータ

pos

[in] 配列での望まれた要素の位置

戻り値

成功の場合は要素の値、存在しない位置での要素の取得が試まれた場合DBL_MAX(直近エラーはERR_OUT_OF_RANGE)

注意事項

DBL_MAX は配列要素の有効な値である可能性もあるので、この値が取得されたら、常に最終のエラーコードをチェックします。

例:

```
//--- CArrayDouble::At(int) の例
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   for(int i=0;i<array.Total();i++)
     {
      double result=array.At(i);
      if(result==DBL_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //---  配列読み込みエラー
         printf("Get element error");
         delete array;
         return;
        }
      //--- 要素を使用する
      //--- . にて。.
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

CompareArray

配列を別の配列と比較します。

```
bool  CompareArray(
   const double&  src[]      // ソース配列
   ) const
```

パラメータ

src[]

[in] 比較元の要素を持つ配列への参照

戻り値

配列が同一なら true、それ以外の場合は false

例:

```
//--- CArrayDouble::CompareArray(const double &[]) の例
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 別の配列と比較する
   int result=array.CompareArray(src);
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

配列を別の配列と比較します。

```
bool  CompareArray(
   const CArrayDouble*  src      // ソース配列
   ) const
```

パラメータ

src

[in] 比較する要素へのポインタ（CArrayDouble クラスインスタンス）

戻り値

配列が同一なら true、それ以外の場合は false

例：

```
//--- CArrayDouble::CompareArray(const CArrayDouble*) の例
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソース配列を作成する
   CArrayDouble *src=new CArrayDouble;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ソース配列に要素を加える
   //--- . にて。.
   //--- 別の配列と比較する
   int result=array.CompareArray(src);
   //--- 配列を削除する
   delete src;
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Minimum

配列の指定された範囲内の最小要素のインデックスを取得します。

```
int  Minimum(
   int  start,      // 開始インデックス
   int  count       // 扱う要素の数
   ) const
```

パラメータ

start

[in] 配列の開始インデックス

count

[in] 扱う要素の数

戻り値

配列の指定された範囲内の最小要素のインデックス

© 2000–2025, MetaQuotes Ltd.

Maximum

配列の指定された範囲内の最大要素のインデックスを取得します。

```
int  Maximum(
   int  start,      // 開始インデックス
   int  count       // 扱う要素の数
   ) const
```

パラメータ

start

[in] 配列の開始インデックス

count

[in] 扱う要素の数

戻り値

配列の指定された範囲内の最大要素のインデックス

© 2000–2025, MetaQuotes Ltd.

InsertSort

ソートされた配列に要素を挿入します。

```
bool  InsertSort(
   double  element      // 挿入する要素
  )
```

パラメータ

element

[in] ソートされた配列に挿入される要素の値

戻り値

成功の場合は true、要素が挿入できなかった場合は false

例:

```
//--- CArrayDouble::InsertSort(double) の例
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を挿入する
   if(!array.InsertSort(100.0))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Search

ソートされた配列内でサンプルに等しい要素を検索します。

```
int  Search(
   double  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayDouble::Search(double) の例
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.Search(100.0)!=-1) printf("Element found");
   else                        printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

ソートされた配列内でサンプルより大きい要素を検索します。

```
int  SearchGreat(
   double  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayDouble::SearchGreat(double) の例
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchGreat(100.0)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

ソートされた配列内でサンプルより小さい要素を検索します。

```
int  SearchLess(
   double  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayDouble::  SearchLess(double) の例
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchLess(100.0)!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

ソートされた配列内でサンプル以上の要素を検索します。

```
int  SearchGreatOrEqual(
   double  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayDouble::SearchGreatOrEqual(double) の例
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchGreatOrEqual(100.0)!=-1) printf("Element found");
   else                                    printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

ソートされた配列内でサンプル以下の要素を検索します。

```
int  SearchLessOrEqual(
   double  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayDouble::SearchLessOrEqual(double) の例
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchLessOrEqual(100.0)!=-1) printf("Element found");
   else                                   printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

ソートされた配列内でサンプルに等しい最初の要素を検索します。

```
int  SearchFirst(
   double  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayDouble::SearchFirst(double) の例
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchFirst(100.0)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

ソートされた配列内でサンプルに等しい最後の要素を検索します。

```
int  SearchLast(
   double  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayDouble::SearchLast(double) の例
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchLast(100.0)!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLinear

ソートされた配列内でサンプルに等しい要素を検索します。

```
int  SearchLinear(
   double  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

注意事項

このメソッドは、ソートされていない配列では線形探索（または順次検索）アルゴリズムを使用します。

例:

```
//--- CArrayDouble::SearchLinear(double) の例
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素を探す
   if(array.SearchLinear(100.0)!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

データ配列をファイルに保存します。

```
virtual bool  Save(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に FileOpen(...) で開かれたファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

例:

```
//--- CArrayDouble::Save(int) の例
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   int            file_handle;
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 100要素を追加する
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- ファイルを開く
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- ファイル保存エラー
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Load

データ配列をファイルから読み込みます。

```
virtual bool  Load(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に FileOpen() 関数で開かれたバイナリファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

例:

```
//--- CArrayDouble::Load(int) の例
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   int            file_handle;
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ファイルを開く
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- ファイル読み込みエラー
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- 配列要素を使用する
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = %f",i,array.At(i));
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Type

配列の型識別子を取得します。

```
virtual int  Type() const
```

戻り値

配列の型識別子（CArrayDouble は 87）

例：

```
//--- CArrayDouble::Type() の例
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列の型を取得する
   int type=array.Type();
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

CArrayString

CArrayString は string 型の変数の動的配列のクラスです。

説明

CArrayString クラスは、string 型変数の動的配列を操作する関数を提供します。ソートや検索、配列要素の追加、挿入、及び削除や、配列のソートや検索が提供されています。また、ファイルを操作するためのメソッドを実装されています。

宣言

```
class CArrayString : public CArray
```

タイトル

```
#include <Arrays\ArrayString.mqh>
```

継承階層

CObject
  CArray
    CArrayString

クラスメソッド

メモリ制御	
Reserve	メモリを割り当てて、配列のサイズを増加します。
Resize	配列の新しい（減少した）サイズを設定します。
Shutdown	配列の新しい（減少した）サイズを設定します。
追加メソッド	
Add	配列の最後に要素を追加します。
AddArray	配列の最後に別の配列の要素を追加します。
AddArray	配列の最後に別の配列の要素を追加します。
Insert	指定された位置に配列の要素を挿入します。
InsertArray	指定された位置に別の配列の要素を挿入します。
InsertArray	指定された位置に別の配列の要素を挿入します。
AssignArray	別の配列の要素を複製します。
AssignArray	別の配列の要素を複製します。
更新メソッド	
Update	指定された位置の配列の要素を変更します。

© 2000–2025, MetaQuotes Ltd.

メモリ制御	
Shift	配列内の指定された位置から要素を指定されたオフセットに移動します。
削除メソッド	
Delete	配列の指定された位置から要素を削除します。
DeleteRange	配列の指定された位置から要素のグループを削除します。
アクセスメソッド	
At	配列の指定された位置から要素を取得します。
比較メソッド	
CompareArray	配列を別の配列と比較します。
CompareArray	配列を別の配列と比較します。
ソートされた配列の操作	
InsertSort	ソートされた配列に要素を挿入します。
Search	ソートされた配列内でサンプルに等しい要素を検索します。
SearchGreat	ソートされた配列内でサンプルより大きい要素を検索します。
SearchLess	ソートされた配列内でサンプルより小さい要素を検索します。
SearchGreatOrEqual	ソートされた配列内でサンプル以上の要素を検索します。
SearchLessOrEqual	ソートされた配列内でサンプル以下の要素を検索します。
SearchFirst	ソートされた配列内でサンプルに等しい最初の要素を検索します。
SearchLast	ソートされた配列内でサンプルに等しい最後の要素を検索します。
SearchLinear	配列内でサンプルに等しい要素を検索します。
入出力	
virtual Save	データ配列をファイルに保存します。
virtual Load	データ配列をファイルから読み込みます。
virtual Type	配列の型識別子を取得します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000–2025, MetaQuotes Ltd.

Reserve

メモリを割り当てて、配列のサイズを増加します。

```
bool  Reserve(
   int  size      // 数
  )
```

パラメータ

size

[in] 配列に追加される要素の数

戻り値

成功の場合は true、ゼロ以下の数を追加しようとする試みがあった場合、または配列が増加しなかった場合は false

注意事項

メモリの断片化を低減するために、配列のサイズは、以前に Step(int) によって決定されたステップのまたはデフォルトの16ステップを使用して変更されます。

例:

```
//--- CArrayString::Reserve(int) の例
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- メモリを予約する
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

配列の新しい(減少した)サイズを設定します。

```
bool  Resize(
   int  size      // サイズ
  )
```

パラメータ

size

[in] 配列の新しいサイズ

戻り値

成功の場合は true、ゼロ以下のサイズを設定しようとする試みがあった場合は false

注意事項

配列のサイズ変更はメモリ使用の最適化に役立ちます。右側の余分な要素が失われます。メモリの断片化を低減するために、配列のサイズは、以前に Step(int) によって決定されたステップのまたはデフォルトの16ステップを使用して変更されます。

例:

```
//--- CArrayString::Resize(int) の例
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列のサイズを変更する
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

配列をクリアして、メモリを完全に開放します。

```
bool  Shutdown()
```

戻り値

成功の場合は true、エラーが発生した場合は false

例：

```
//--- CArrayString::Shutdown() の例
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をシャットダウンする
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Add

配列の最後に要素を追加します。

```
bool  Add(
   string  element      // 追加する要素
  )
```

パラメータ

element

[in] 配列に追加される要素の値

戻り値

成功の場合は true、要素が追加できなかった場合は false

例:

```
//--- CArrayString::Add(string) の例
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   for(int i=0;i<100;i++)
     {
      if(!array.Add(IntegerToString(i)))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

配列の最後に別の配列の要素を追加します。

```
bool  AddArray(
   const string&  src[]      // ソース配列の配列
  )
```

パラメータ

src[]

[in] 追加する要素を持つ配列への参照

戻り値

成功の場合は true、要素が追加できなかった場合は false

例:

```
//--- CArrayString::AddArray(const string &[]) の例
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 別の配列を追加する
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete array;
      return;
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

AddArray

配列の最後に別の配列の要素を追加します。

```
bool  AddArray(
   const CArrayString*  src      // ソース配列
  )
```

パラメータ

src

[in] 追加する要素へのポインタ（CArrayString クラスインスタンス）

戻り値

成功の場合は true、要素が追加できなかった場合は false

例：

```
//--- CArrayString::AddArray(const CArrayString*) の例
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソース配列を作成する
   CArrayString *src=new CArrayString;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ソース配列に要素を加える
   //--- . にて。.
   //--- 別の配列を追加する
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete src;
      delete array;
      return;
     }
   //--- ソース配列を削除する
   delete src;
```

© 2000–2025, MetaQuotes Ltd.

```
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Insert

配列の指定された位置に要素を挿入します。

```
bool  Insert(
   string  element,     // 挿入する要素
   int     pos          // 位置
  )
```

パラメータ

element

[in] 配列に挿入される要素の値

pos

[in] 配列内の挿入位置

戻り値

成功の場合は true、要素が挿入できなかった場合は false

例:

```
//--- CArrayString::Insert(string,int) の例
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 要素を挿入する
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(IntegerToString(i),0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

指定された位置に別の配列の要素を挿入します。

```
bool  InsertArray(
   const string&  src[],      // 挿入元の配列
   int            pos         // 位置
  )
```

パラメータ

src[]

[in] 挿入する要素を持つ配列への参照

pos

[in] 配列内の挿入位置

戻り値

成功の場合は true、要素が挿入できなかった場合は false

例:

```
//--- CArrayString::InsertArray(const string &[],int) の例
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 別の配列を挿入する
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete array;
      return;
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

InsertArray

指定された位置に別の配列の要素を挿入します。

```
bool  InsertArray(
   CArrayString*  src,      // 挿入元へのポインタ
   int             pos      // 位置
  )
```

パラメータ

src

[in] 挿入する要素へのポインタ（CArrayString クラスインスタンス）

pos

[in] 配列内の挿入位置

戻り値

成功の場合は true、要素が挿入できなかった場合は false

例：

```
//--- CArrayString::InsertArray(const CArrayString*,int) の例
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソース配列を作成する
   CArrayString *src=new CArrayString;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ソース配列に要素を加える
   //--- . にて。.
   //--- 別の配列を挿入する
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete src;
      delete array;
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }
   //--- ソース配列を削除する
   delete src;
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

AssignArray

別の配列の要素を複製します。

```
bool  AssignArray(
   const string&  src[]      // ソース配列の配列
  )
```

パラメータ

src[]

[in] 複製する配列要素への参照

戻り値

成功の場合は true、要素が複製できなかった場合は false

例:

```
//--- CArrayString::AssignArray(const string &[]) の例
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 別の配列を割り当てる
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete array;
      return;
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

AssignArray

別の配列の要素を複製します。

```
bool  AssignArray(
   const CArrayString*  src      // ソース配列
  )
```

パラメータ

src

[in] 複製する要素へのポインタ（CArrayString クラスインスタンス）

戻り値

成功の場合は true、要素が複製できなかった場合は false

例：

```
//--- CArrayString::AssignArray(const CArrayString*) の例
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソース配列を作成する
   CArrayString *src  =new CArrayString;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ソース配列に要素を加える
   //--- . にて。.
   //--- 別の配列を割り当てる
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete src;
      delete array;
      return;
     }
   //--- 配列が同一
   //--- ソース配列を削除する
```

© 2000–2025, MetaQuotes Ltd.

```
   delete src;
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Update

指定された位置の配列の要素を変更します。

```
bool  Update(
   int     pos,          // 位置
   string  element       // 値
  )
```

パラメータ

pos

[in] 変更される要素の配列内での位置

element

[in] 要素の新しい値

戻り値

成功の場合は true、要素が変更できなかった場合は false

例:

```
//--- CArrayString::Update(int, string) の例
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素を更新する
   if(!array.Update(0,"ABC"))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Shift

配列内の指定された位置から要素を指定されたオフセットに移動します。

```
bool  Shift(
   int  pos,        // 位置
   int  shift       // シフト
  )
```

パラメータ

pos

[in] 配列での移動した要素の位置

shift

[in]（正と負の両方）のシフトの値

戻り値

成功の場合は true、要素が移動できなかった場合は false

例：

```
//--- CArrayString::Shift(int,int) の例
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素をシフトする
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Delete

配列の指定された位置から要素を削除します。

```
bool  Delete(
   int  pos      // 位置
  )
```

パラメータ

pos

[in] 配列から削除される要素の位置

戻り値

成功の場合は true、要素が削除できなかった場合は false

例:

```
//--- CArrayString::Delete(int) の例
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素を削除する
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

DeleteRange

配列の指定された位置から要素のグループを削除します。

```
bool  DeleteRange(
   int  from,      // 最初の要素の位置
   int  to         // 最後の要素の位置
  )
```

パラメータ

from

[in] 配列の削除される範囲の最初の要素の位置

to

[in] 配列の削除される範囲の最後の要素の位置

戻り値

成功の場合は true、削除が失敗した場合は false

例:

```
//--- CArrayString::DeleteRange(int,int) の例
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素を削除する
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

At

配列の指定された位置から要素を取得します。

```
string  At(
   int  pos      // 位置
   ) const
```

パラメータ

pos

[in] 配列での望まれた要素の位置

戻り値

The value of the element in case of success, "- if there was an attempt to get an element from not existing position (in this case the last error will be ERR_OUT_OF_RANGE).

注意事項

STRING_MAX は配列要素の有効な値である可能性もあるので、この値が取得されたら、常に最終のエラーコードをチェックします。

例:

```
//--- CArrayString::At(int) の例
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   for(int i=0;i<array.Total();i++)
     {
      string result=array.At(i);
      if(result=="" && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- 配列読み込みエラー
         printf("Get element error");
         delete array;
         return;
        }
      //--- 要素を使用する
      //--- . にて。.
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

CompareArray

配列を別の配列と比較します。

```
bool  CompareArray(
   const string&  src[]      // ソース配列の配列
   ) const
```

パラメータ

src[]

[in] 比較元の要素を持つ配列への参照

戻り値

配列が同一なら true、それ以外の場合は false

例:

```
//--- CArrayString::CompareArray(const string &[]) の例
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 別の配列と比較する
   int result=array.CompareArray(src);
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

配列を別の配列と比較します。

```
bool  CompareArrays(
   const CArrayString*  src      // ソース配列
   ) const
```

パラメータ

src

[in] 比較する要素へのポインタ（CArrayString クラスインスタンス）

戻り値

配列が同一なら true、それ以外の場合は false

例：

```
//--- CArrayString::CompareArray(const CArrayString*) の例
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソース配列を作成する
   CArrayString *src=new CArrayString;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ソース配列に要素を加える
   //--- . にて。.
   //--- 別の配列と比較する
   int result=array.CompareArray(src);
   //--- 配列を削除する
   delete src;
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

InsertSort

ソートされた配列に要素を挿入します。

```
bool  InsertSort(
   string  element      // 挿入する要素
  )
```

パラメータ

element

[in] ソートされた配列に挿入される要素の値

戻り値

成功の場合は true、要素が挿入できなかった場合は false

例:

```
//--- CArrayString::InsertSort(string) の例
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を挿入する
   if(!array.InsertSort("ABC"))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Search

ソートされた配列内でサンプルに等しい要素を検索します。

```
int  Search(
   string  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayString::Search(string) の例
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.Search("ABC")!=-1) printf("Element found");
   else                        printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

ソートされた配列内でサンプルより大きい要素を検索します。

```
int  SearchGreat(
   string  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayString::SearchGreat(string) の例
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchGreat("ABC")!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

ソートされた配列内でサンプルより小さい要素を検索します。

```
int  SearchLess(
   string  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayString::  SearchLess(string)  の例
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchLess("ABC")!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

ソートされた配列内でサンプル以上の要素を検索します。

```
int  SearchGreatOrEqual(
   string  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayString::  SearchGreatOrEqual(string) の例
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchGreatOrEqual("ABC")!=-1) printf("Element found");
   else                                    printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

ソートされた配列内でサンプル以下の要素を検索します。

```
int  SearchLessOrEqual(
   string  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayString::  SearchLessOrEqual(string) の例
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchLessOrEqual("ABC")!=-1) printf("Element found");
   else                                   printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

ソートされた配列内でサンプルに等しい最初の要素を検索します。

```
int  SearchFirst(
   string  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayString::  SearchFirst(string) の例
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchFirst("ABC")!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

ソートされた配列内でサンプルに等しい最後の要素を検索します。

```
int  SearchLast(
   string  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayString::  SearchLast(string) の例
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を探す
   if(array.SearchLast("ABC")!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLinear

配列内でサンプルに等しい要素を検索します。

```
int  SearchLinear(
   string  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

注意事項

このメソッドは、ソートされていない配列では線形探索（または順次検索）アルゴリズムを使用します。

例:

```
//--- CArrayString::SearchLinear(string) の例
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素を探す
   if(array.SearchLinear("ABC")!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

データ配列をファイルに保存します。

```
virtual bool  Save(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に FileOpen(...) で開かれたファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

例:

```
//--- CArrayString::Save(int) の例
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   int            file_handle;
   CArrayString *array=new CArrayString;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 100要素を追加する
   for(int i=0;i<100;i++)
     {
      array.Add(IntegerToString(i));
     }
   //--- ファイルを開く
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- ファイル保存エラー
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Load

データ配列をファイルから読み込みます。

```
virtual bool  Load(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に FileOpen(...) で開かれたファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

例:

```
//--- CArrayString::Load(int) の例
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   int            file_handle;
   CArrayString *array=new CArrayString;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ファイルを開く
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- ファイル読み込みエラー
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- 配列要素を使用する
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = '%s'",i,array.At(i));
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Type

配列の型識別子を取得します。

```
virtual int  Type() const
```

戻り値

配列の型識別子（CArrayString は 89）

例：

```
//--- CArrayString::Type() の例
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列の型を取得する
   int type=array.Type();
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

CArrayObj

CArrayObj クラスは、CObjectと派生クラスのインスタンスへのポインタの動的配列のクラスです。

説明

CArrayObj クラスは、CObject とその派生クラスのインスタンスの動的配列のポインタを操作する機能を提供します。これは、基本データ型のデータ及びより複雑な構成を持つデータ構造体を持つ多次元動的配列を操作することが出来ます。

配列要素の追加/挿入/削除、配列のソート、ソートされた配列での検索が実装されています。またファイル操作のメソッドも実装されています。

CArrayObj クラスにはいくつかの注意点があります。

宣言

```
class CArrayObj : public CArray
```

タイトル

```
#include <Arrays\ArrayObj.mqh>
```

継承階層

CObject
  CArray
    CArrayObj

直接子孫

CIndicators, CSeries

クラスメソッド

属性	
FreeMode	メモリ管理フラグを取得します。
FreeMode	メモリ管理フラグを設定します。
メモリ制御	
Reserve	メモリを割り当てて、配列のサイズを増加します。
Resize	配列の新しい(減少した)サイズを設定します。
Shutdown	配列をクリアして、配列メモリ(要素メモリ以外)を完全に開放します。
追加メソッド	
Add	配列の最後に要素を追加します。
AddArray	配列の最後に要素を追加します。
Insert	指定された位置に配列の要素を挿入します。

© 2000-2025, MetaQuotes Ltd.

属性	
InsertArray	指定された位置に別の配列の要素を挿入します。
AssignArray	別の配列の要素を複製します。
更新メソッド	
Update	指定された位置の配列の要素を変更します。
Shift	配列内の指定された位置から要素を指定されたオフセットに移動します。
削除メソッド	
Detach	配列の指定された位置から要素を取得して、配列から削除します。
Delete	配列の指定された位置から要素を削除します。
DeleteRange	配列の指定された位置から要素のグループを削除します。
Clear	メモリを解放せずに、配列の全ての要素を削除します。
アクセスメソッド	
At	配列の指定された位置から要素を取得します。
比較メソッド	
CompareArray	配列を別の配列と比較します。
ソートされた配列の操作	
InsertSort	ソートされた配列に要素を挿入します。
Search	ソートされた配列内でサンプルに等しい要素を検索します。
SearchGreat	ソートされた配列内でサンプルより大きい要素を検索します。
SearchLess	ソートされた配列内でサンプルより小さい要素を検索します。
SearchGreatOrEqual	ソートされた配列内でサンプル以上の要素を検索します。
SearchLessOrEqual	ソートされた配列内でサンプル以下の要素を検索します。
SearchFirst	ソートされた配列内でサンプルに等しい最初の要素を検索します。
SearchLast	ソートされた配列内でサンプルに等しい最後の要素を検索します。
入出力	
Save	データ配列をファイルに保存します。
Load	データ配列をファイルから読み込みます。
Type	配列の型識別子を取得します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

実用的な配列は（標準ライブラリの全てのクラスを含む）CObject クラスから派生します。

例として、2 次元配列を見てみます。

```
#include <Arrays\ArrayDouble.mqh>
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int i,j;
   int first_size=10;
   int second_size=100;
//--- 配列を作成する
   CArrayObj    *array=new CArrayObj;
   CArrayDouble *sub_array;
//---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
//--- 子配列を作成する
   for(i=0;i<first_size;i++)
     {
      sub_array=new CArrayDouble;
      if(sub_array==NULL)
        {
         delete array;
         printf("Object create error");
         return;
        }
      //--- 配列に書き込む
      for(j=0;j<second_size;j++)
        {
         sub_array.Add(i*j);
        }
      array.Add(sub_array);
     }
//--- 配列作成に成功
   for(i=0;i<first_size;i++)
     {
      sub_array=array.At(i);
      for(j=0;j<second_size;j++)
        {
```

© 2000–2025, MetaQuotes Ltd.

```
         double element=sub_array.At(j);
         //--- 配列要素を使用する
        }
      }
    delete array;
   }
```

注意点

このクラスは動的メモリを制御するので、配列の要素を操作する際には注意するべきです。

メモリ管理のメカニズムは、FreeMode（bool）メソッドを使用して有効/無効に出来ます。デフォルトではメカニズムは有効です。

よって、CArrayObj クラス操作の方法は 2 つあります。

1. メモリ管理のメカニズムが有効（デフォルト）

この場合、CArrayObj は、要素が配列から除去された後、要素に使用されていたメモリを解放する責任を負います。このプログラムでは、ユーザは、配列要素を解放してはいけません。

例：

```
    int i;
//--- 配列を作成する
    CArrayObj *array=new CArrayObj;
//--- 配列要素に書き込む
    for(i=0;i<10;i++) array.Add(new CObject);
//--- 何かする
    for(i=0;i<array.Total();i++)
      {
       CObject *object=array.At(i);
       //--- 要素に何かする
      にて。にて。にて。
      }
//--- 配列とその要素を削除する
    delete array;
```

2. メモリ管理のメカニズムが無効

この場合、CArrayObj は、要素が配列から除去された後、要素に使用されていたメモリを解放する責任を負いません。ユーザプログラムが配列要素を解放するべきです。

例：

```
    int i;
//--- 配列を作成する
    CArrayObj *array=new CArrayObj;
//--- メモリ管理のメカニズムを無効にする
    array.FreeMode(false);
//--- 配列要素に書き込む
```

© 2000-2025, MetaQuotes Ltd.

```
   for(i=0;i<10;i++) array.Add(new CObject);
//--- 何かする
   for(i=0;i<array.Total();i++)
     {
      CObject *object=array.At(i);
      //--- 要素に何かする
     にて。にて。にて。
     }
//--- 配列要素を削除する
   while(array.Total()) delete array.Detach();
//--- 空の配列を削除する
   delete array;
```

© 2000–2025, MetaQuotes Ltd.

FreeMode

メモリ管 理フラグを取 得します。

```
bool  FreeMode() const
```

戻 り値

メモリ管 理 のフラグ

例 :

```
//--- CArrayObj::FreeMode() の例
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- メモリ管理フラグを取得する
   bool array_free_mode=array.FreeMode();
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

FreeMode

メモリ管理フラグを設定します。

```
void  FreeMode(
   bool  mode      // 新しいフラグ
  )
```

パラメータ

mode

[in] メモリ管理フラグの新しい値

戻り値

なし

注意事項

メモリ管理のフラグの設定は CArrayObj クラス使用にあたって重要な部分です。配列要素は動的オブジェクトへのポインタであるため、配列から削除する際にどのように処理するかを決定することが重要です。

フラグが設定されている場合は、配列から除去された際に、要素は自動的に delete オペレータによって削除されます。フラグが設定されていない場合は、削除されたオブジェクトへのポインタがユーザプログラムのどこかに残っていて、後にプログラムによって解放されると想定されています。

ユーザプログラムががメモリ管理フラグをリセットした場合は、ユーザは、プログラムの終了前に配列除去の責任があることを理解する必要があります。さもないと、 new 演算子によって要素に割り当てられたメモリが解放されません。

データが大量の場合、端末の動作が停止することもあります。

ユーザがメモリ管理フラグをリセットしない場合は、別の問題が起こる可能性があります。配列要素ポインタがローカル変数に格納されている場合、配列除去は重大なエラーにつながり、ユーザプログラムが強制停止されます。デフォルトでは、メモリ管理フラグが設定されていて、メモリ要素を解放する責任は配列クラスにあります。

例:

```
//--- CArrayObj::FreeMode(bool) の例
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- メモリ管理フラグをリセットする
   array.FreeMode(false);
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Reserve

メモリを割り当てて、配列のサイズを増加します。

```
bool  Reserve(
   int  size      // 数
  )
```

パラメータ

size

[in] 配列に追加される要素の数

戻り値

成功の場合は true、ゼロ以下の数を追加しようとする試みがあった場合、または配列が増加しなかった場合は false

注意事項

メモリの断片化を低減するために、配列のサイズは、以前に Step(int) によって決定されたステップのまたはデフォルトの16ステップを使用して変更されます。

例:

```
//--- CArrayObj::Reserve(int) の例
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

配列の新しい(減少した)サイズを設定します。

```
bool  Resize(
   int  size      // サイズ
  )
```

パラメータ

size

[in] 配列の新しいサイズ

戻り値

成功の場合は true、ゼロ以下のサイズを設定しようとする試みがあった場合は false

注意事項

配列のサイズ変更はメモリ使用の最適化に役立ちます。右側の余分な要素が失われます。失われた要素のためのメモリは、メモリ管理のモードに応じて解放されます。

メモリの断片化を低減するために、配列のサイズは、以前に Step(int) によって決定されたステップのまたはデフォルトの16ステップを使用して変更されます。

例:

```
//--- CArrayObj::Resize(int) の例
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列のサイズを変更する
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Clear

メモリを解放せずに、配列の全ての要素を削除します。

```
void  Clear()
```

戻り値

No.

注意事項

メモリ管理フラグが有効であれば、削除された要素に使用されていたメモリが解放されます。

例：

```
//--- CArrayObj::Clear() の例
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をクリアする
   array.Clear();
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

配列をクリアして、配列メモリ(要素メモリ以外)を完全に開放します。

```
bool  Shutdown()
```

戻り値

成功の場合は true、エラーが発生した場合は false

注意事項

メモリ管理フラグが有効であれば、削除された要素に使用されていたメモリが解放されます。

例:

```
//--- CArrayObj::Shutdown() の例
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をシャットダウンする
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CreateElement

指定された位置に新しい要素を作成します。

```
bool  CreateElement(
   int  index      // 位置
  )
```

パラメータ

index

[in] 新しい要素が作成される位置

戻り値

成功の場合は true、要素が作成できなかった場合は false

注意事項

MethoCArrayObj クラスの CreateElement (int) メソッドは常に false を戻し何のアクションも取りません。派生クラスで CreateElement (int) メソッドが必要に応じて実装されるべきです。

例:

```
//--- CArrayObj::CreateElement(int) の例
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int         size=100;
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列に書き込む
   array.Reserve(size);
   for(int i=0;i<size;i++)
     {
      if(!array.CreateElement(i))
        {
         printf("Element create error");
         delete array;
         return;
        }
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
```

© 2000-2025, MetaQuotes Ltd.

```
   }
```

© 2000–2025, MetaQuotes Ltd.

Add

配列の最後に要素を追加します。

```
bool  Add(
   CObject*  element      // 追加する要素
  )
```

パラメータ

element

[in] 配列に追加される要素の値

戻り値

成功の場合は true、要素が追加できなかった場合は false

注意事項

NULL などの不正なポインタがパラメータに渡された場合、要素は配列に追加されません。

例:

```
//--- CArrayObj::Add(CObject*) の例
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 100要素を追加する
   for(int i=0;i<100;i++)
     {
      if(!array.Add(new CObject))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

配列の最後に別の配列の要素を追加します。

```
bool  AddArray(
   const CArrayObj *  src      // ソース配列へのポインタ
  )
```

パラメータ

src

[in] 追加する要素へのポインタ(CArrayDouble クラスインスタンス)

戻り値

成功の場合は true、要素が追加できなかった場合は false

注意事項

配列への配列要素の追加は、実際にはポインタのコピーです。 そのため、メソッドを呼び出す時に、 複数の変数に動的オブジェクトへのポインタが存在する可能性があります。

```
//--- 例
extern bool        make_error;
extern int         error;
extern CArrayObj *src;
//--- CArrayObj  の新しいインスタンスを作成する
//--- デフォルトのメモリ管理が有効
CArrayObj *array=new CArrayObj;
//--- ソース配列の要素を追加(コピー)する
if(array!=NULL)
   bool result=array.AddArray(src);
if(make_error)
  {
   //--- 不正なアクションを取る
   switch(error)
     {
      case 0:
         //--- メモリ管理のフラグをチェックしないでソース配列を削除する
         delete src;
         //--- 結果:
         //--- 受け取り側の配列での不正なポインタでの要素アドレス指定が可能
         break;
      case 1:
         //--- ソース配列のメモリ管理のメカニズムを無効にする
         if(src.FreeMode()) src.FreeMode(false);
         //--- ソース配列を削除しない
         //--- 結果:
         //--- 受け取り側の配列の削除後、ソース配列での不正なポインタでの要素アドレス指定が可能
         break;
      case 2:
```

© 2000-2025, MetaQuotes Ltd.

```
            //--- ソース配列のメモリ管理のメカニズムを無効にする
            src.FreeMode(false);
            //--- 受け取り側の配列のメモリ管理のメカニズムを無効にする
            array.FreeMode(false);
            //--- 結果:
            //--- プログラム終了後、「メモリリーク」が発生する
            break;
         }
      }
   else
      {
      //--- ソース配列のメモリ管理のメカニズムを無効にする
      if(src.FreeMode()) src.FreeMode(false);
      //--- ソース配列を削除する
      delete src;
      //--- 結果
      //--- 受け取り側の配列要素のアドレス指定は正しい
      //--- 受け取り側の配列の削除はその要素の削除につながる
      }
```

例:

```
//--- CArrayObj::AddArray(const CArrayObj*) の例
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
   {
    CArrayObj *array=new CArrayObj;
    //---
    if(array==NULL)
      {
       printf("Object create error");
       return;
      }
    //--- ソース配列を作成する
    CArrayObj *src=new CArrayObj;
    if(src==NULL)
      {
       printf("Object create error");
       delete array;
       return;
      }
    //--- メモリ管理フラグをリセットする
    src.FreeMode(false);
    //--- ソース配列に書き込む
    //--- . にて。.
    //--- 別の配列を追加する
    if(!array.AddArray(src))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       printf("Array addition error");
       delete src;
       delete array;
       return;
      }
   //--- 追加元の空配列を削除する
   delete src;
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Insert

配列の指定された位置に要素を挿入します。

```
bool  Insert(
   CObject*  element,      // 挿入する要素
   int       pos           // 位置
  )
```

パラメータ

element

[in] 配列に挿入される要素の値

pos

[in] 配列内の挿入位置

戻り値

成功の場合は true、要素が挿入できなかった場合は false

注意事項

NULL などの不正なポインタがパラメータに渡された場合、要素は配列に追加されません。

例:

```
//--- CArrayObj::Insert(CObject*,int) の例
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 要素を挿入する
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(new CObject,0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
```

© 2000-2025, MetaQuotes Ltd.

```
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

InsertArray

指定された位置に別の配列の要素を挿入します。

```
bool  InsertArray(
   const CArrayObj*  src,      // 挿入元へのポインタ
   int               pos       // 位置
  )
```

パラメータ

src

[in] 挿入する要素へのポインタ（CArrayObj クラスインスタンス）

pos

[in] 配列内の挿入位置

戻り値

成功の場合は true、要素が挿入できなかった場合は false

注意事項

CArrayObj::AddArray(const CArrayObj*) をご参照ください。

例：

```
//--- CArrayObj::InsertArray(const CArrayObj*,int) の例
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソース配列を作成する
   CArrayObj *src=new CArrayObj;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- メモリ管理フラグをリセットする
   src.FreeMode(false);
   //--- ソース配列に書き込む
   //--- . にて。.
   //--- 別の配列を挿入する
```

© 2000–2025, MetaQuotes Ltd.

```
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete src;
      delete array;
      return;
     }
   //--- 追加元の空配列を削除する
   delete src;
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

AssignArray

別の配列の要素を複製します。

```
bool  AssignArray(
   const CArrayObj*  src      // コピー元へのポインタ
  )
```

パラメータ

src

[in] 複製する要素へのポインタ（CArrayObj クラスインスタンス）

戻り値

成功の場合は true、要素が複製できなかった場合は false

注意事項

AssignArray の呼び出し時に受け取り側の配列が空でない場合は、その全ての要素が削除され、メモリ管理フラグが有効であれば、削除された要素に使用されていたメモリが解放されます。受け取り側の配列はソース配列と同一のコピーです。CArrayObj::AddArray(const CArrayObj*) をご参照ください。

例：

```
//--- CArrayObj::AssignArray(const CArrayObj*) の例
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソース配列を作成する
   CArrayObj *src=new CArrayObj;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- メモリ管理フラグをリセットする
   src.FreeMode(false);
   //--- ソース配列に書き込む
   //--- . にて。.
   //--- 別の配列を割り当てる
   if(!array.AssignArray(src))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      printf("Array assigned error");
      delete src;
      delete array;
      return;
     }
   //--- 配列が同一
   //--- 追加元の空配列を削除する
   delete src;
   //--- 配列を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Update

指定された位置の配列の要素を変更します。

```
bool  Update(
   int       pos,          // 位置
   CObject*  element       // 値
  )
```

パラメータ

pos

[in] 変更される要素の配列内での位置

element

[in] 要素の新しい値

戻り値

成功の場合は true、要素が変更できなかった場合は false

注意事項

NULL などの不正なポインタがパラメータに渡された場合、要素は変更されません。メモリ管理フラグが有効であれば、変更された要素に使用されていたメモリが解放されます。

例:

```
//--- CArrayObj::Update(int,CObject*) の例
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素を更新する
   if(!array.Update(0,new CObject))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Shift

配列内の指定された位置から要素を指定されたオフセットに移動します。

```
bool  Shift(
   int  pos,        // 位置
   int  shift       // シフト
  )
```

パラメータ

pos

[in] 配列での移動した要素の位置

shift

[in]（正と負の両方）のシフトの値

戻り値

成功の場合は true、要素が移動できなかった場合は false

例:

```
//--- CArrayObj::Shift(int,int) の例
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素をシフトする
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Detach

配列の指定された位置から要素を削除します。

```
CObject*  Detach(
   int  pos      // 位置
  )
```

パラメータ

pos

[in] 削除される要素の位置

戻り値

成功の場合は削除された要素へのポインタ、要素が削除できない場合は NULL

注意事項

配列から削除された要素に使用されていたメモリはメモリ管理フラグに関係なく解放されません。配列要素ポインタは使用されたら解放されなければなりません。

例:

```
//--- CArrayObj::Detach(int) の例
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   CObject *object=array.Detach(0);
   if(object==NULL)
     {
      printf("Detach error");
      delete array;
      return;
     }
   //--- 要素を使用する
   //--- . にて。.
   //--- 要素を削除する
   delete object;
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

配列の指定された位置から要素を削除します。

```
bool  Delete(
   int  pos      // 位置
  )
```

パラメータ

pos

[in] 配列から削除される要素の位置

戻り値

成功の場合は true、要素が削除できなかった場合は false

注意事項

メモリ管理フラグが有効であれば、削除された要素に使用されていたメモリが解放されます。

例:

```
//--- CArrayObj::Delete(int) の例
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteRange

配列の指定された位置から要素のグループを削除します。

```
bool  DeleteRange(
   int  from,     // 最初の要素の位置
   int  to        // 最後の要素の位置
  )
```

パラメータ

from

[in] 配列の削除される範囲の最初の要素の位置

to

[in] 配列の削除される範囲の最後の要素の位置

戻り値

成功の場合は true、削除が失敗した場合は false

注意事項

メモリ管理フラグが有効であれば、削除された要素に使用されていたメモリが解放されます。

例:

```
//--- CArrayObj::DeleteRange(int,int) の例
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 要素を削除する
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

At

配列の指定された位置から要素を取得します。

```
CObject*  At(
   int  pos      // 位置
  )
```

パラメータ

pos

[in] 配列での望まれた要素の位置

戻り値

成功の場合は要素の値、存在しない位置での要素の取得が試まれた場合NULL

例:

```
//--- CArrayObj::At(int) の例
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 要素を追加する
   //--- . にて。.
   for(int i=0;i<array.Total();i++)
     {
      CObject *result=array.At(i);
      if(result==NULL)
        {
         //--- 配列読み込みエラー
         printf("Get element error");
         delete array;
         return;
        }
      //--- 要素を使用する
      //--- . にて。.
     }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

配列を別の配列と比較します。

```
bool  CompareArray(
   const CArrayObj*  src      // コピー元へのポインタ
   ) const
```

パラメータ

src

[in] 比較する要素へのポインタ（CArrayObj クラスインスタンス）

戻り値

配列が同一なら true、それ以外の場合は false

例：

```
//--- CArrayObj::CompareArray(const CArrayObj*) の例
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソース配列を作成する
   CArrayObj *src=new CArrayObj;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ソース配列に書き込む
   //--- . にて。.
   //--- 別の配列と比較する
   int result=array.CompareArray(src);
   //--- 配列を削除する
   delete src;
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

InsertSort

ソートされた配列に要素を挿入します。

```
bool  InsertSort(
   CObject*  element      // 挿入する要素
  )
```

パラメータ

element

[in] ソートされた配列に挿入される要素の値

戻り値

成功の場合は true、要素が挿入できなかった場合は false

注意事項

NULL などの不正なポインタがパラメータに渡された場合、要素は配列に追加されません。

例:

```
//--- CArrayObj::InsertSort(CObject*) の例
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- 要素を挿入する
   if(!array.InsertSort(new CObject))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Search

ソートされた配列内でサンプルに等しい要素を検索します。

```
int  Search(
   CObject*  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayObj::Search(CObject*) の例
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- サンプルを作成する
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete array;
      return;
     }
   //--- サンプルの属性を設定する
   //--- . にて。.
   //--- 要素を探す
   if(array.Search(sample)!=-1) printf("Element found");
   else                         printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

ソートされた配列内でサンプルより大きい要素を検索します。

```
int  SearchGreat(
   CObject*  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayObj::SearchGreat(CObject*) の例
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- サンプルを作成する
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete array;
      return;
     }
   //--- サンプルの属性を設定する
   //--- . にて。.
   //--- 要素を探す
   if(array.SearchGreat(sample)!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

ソートされた配列内でサンプルより小さい要素を検索します。

```
int  SearchLess(
   CObject*  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayObj::SearchLess(CObject*) の例
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- サンプルを作成する
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete array;
      return;
     }
   //--- サンプルの属性を設定する
   //--- . にて。.
   //--- 要素を探す
   if(array.SearchLess(sample)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

ソートされた配列内でサンプル以上の要素を検索します。

```
int  SearchGreatOrEqual(
   CObject*  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayObj::SearchGreatOrEqual(CObject*) の例
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- サンプルを作成する
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete array;
      return;
     }
   //--- サンプルの属性を設定する
   //--- . にて。.
   //--- 要素を探す
   if(array.SearchGreatOrEqual(sample)!=-1) printf("Element found");
   else                                     printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

ソートされた配列内でサンプル以下の要素を検索します。

```
int  SearchLessOrEqual(
   CObject*  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayObj::SearchLessOrEqual(CObject*) の例
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- サンプルを作成する
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete array;
      return;
     }
   //--- サンプルの属性を設定する
   //--- . にて。.
   //--- 要素を探す
   if(array.SearchLessOrEqual(sample)!=-1) printf("Element found");
   else                                    printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

ソートされた配列内でサンプルに等しい最初の要素を検索します。

```
int  SearchFirst(
   CObject*  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayObj::SearchFirst(CObject*) の例
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- サンプルを作成する
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete array;
      return;
     }
   //--- サンプルの属性を設定する
   //--- . にて。.
   //--- 要素を探す
   if(array.SearchFirst(sample)!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

ソートされた配列内でサンプルに等しい最後の要素を検索します。

```
int  SearchLast(
   CObject*  element      // サンプル
   ) const
```

パラメータ

element

[in] 配列で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は -1

例:

```
//--- CArrayObj::SearchLast(CObject*) の例
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- 配列をソートする
   array.Sort();
   //--- サンプルを作成する
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete array;
      return;
     }
   //--- サンプルの属性を設定する
   //--- . にて。.
   //--- 要素を探す
   if(array.SearchLast(sample)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- 配列を削除する
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

データ配列をファイルに保存します。

```
virtual bool  Save(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に FileOpen(...) で開かれたバイナリファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

例:

```
//--- CArrayObj::Save(int) の例
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int         file_handle;
   CArrayObj *array=new CArrayObj;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列要素を追加する
   //--- . にて。.
   //--- ファイルを開く
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- ファイル保存エラー
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Load

配列データをファイルから読み込みます。

```
virtual bool  Load(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に FileOpen(...) で開かれたファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

注意事項

ファイルから配列を読み込んで要素を作成する時には CArrayObj::CreateElement(int) メソッドが呼ばれます。

例:

```
//--- CArrayObj::Load(int) の例
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int         file_handle;
   CArrayObj *array=new CArrayObj;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ファイルを開く
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- ファイル読み込みエラー
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- 配列要素を使用する
   //--- . にて。.
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

Type

配列の型識別子を取得します。

```
virtual int  Type() const
```

戻り値

配列の型識別子（CArrayObj は 7778）

例:

```
//--- CArrayObj::Type() の例
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 配列の型を取得する
   int type=array.Type();
   //--- 配列を削除する
   delete array;
  }
```

© 2000–2025, MetaQuotes Ltd.

CList

CList は CObject と派生クラスのインスタンスへのポインタの動的配列のクラスです。

説明

CList クラスは CObject とその派生クラスのインスタンスのリストを操作する機能を提供します。配列要素の追加/挿入/削除、配列のソート、ソートされた配列での検索が実装されています。またファイル操作のメソッドも実装されています。

CList の操作にはいくつかの注意点があります。このクラスは動的メモリを制御するので、配列の要素を操作する際には注意するべきです。

メモリ管理の注意点は CArrayObj において説明されたものと同じです。

宣言

```
class CList : public CObject
```

タイトル

```
#include <Arrays\List.mqh>
```

継承階層

CObject
  CList

クラスメソッド

属性	
FreeMode	リスト要素の削除に使用されるメモリ管理のフラグを取得します。
FreeMode	リスト要素の削除に使用されるメモリ管理のフラグを設定します。
Total	リストの要素数を取得します。
IsSorted	リストがソートされているかを示すフラグを取得します。
SortMode	ソートモードを取得します。
作成メソッド	
CreateElement	リストで新しい項目を作成します。
追加メソッド	
Add	リストの最後に要素を追加します。
Insert	指定された位置にリストの要素を挿入します。
削除メソッド	
DetachCurrent	リストの現在の位置から要素を「物理的に」削除せずに削除します。

© 2000-2025, MetaQuotes Ltd.

属性	
DeleteCurrent	リストの現在の位置から要素を削除します。
Delete	リストの指定された位置から要素を削除します。
Clear	リストの全要素を削除します。
ナビゲート	
IndexOf	リスト要素のインデックスを取得します。
GetNodeAtIndex	リストで指定されたインデックスを持つ項目を取得します。
GetFirstNode	リストの最初の要素を取得します。
GetPrevNode	リストの前の要素を取得します。
GetCurrentNode	リストの現在の要素を取得します。
GetNextNode	リストの次の要素を取得します。
GetLastNode	リストの最後の要素を取得します。
順序付けメソッド	
Sort	リストをソートします。
MoveToIndex	リストの現在の要素を指定された位置に移動します。
Exchange	リストの 2 つの要素を交換します。
比較メソッド	
CompareList	リストを別のリストと比較します。
検索メソッド	
Search	ソートされたリスト内でサンプルに等しい要素を検索します。
入出力	
virtual Save	リストデータをファイルに保存します。
virtual Load	リストデータをファイルから読み込みます。
virtual Type	リストの型識別子を取得します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

© 2000–2025, MetaQuotes Ltd.

FreeMode

リスト要素の削除に使用されるメモリ管理のフラグを取得します。

```
bool  FreeMode() const
```

戻り値

メモリ管理のフラグ

例：

```
//--- CList::FreeMode() の例
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- メモリ管理フラグを取得する
   bool list_free_mode=list.FreeMode();
   //--- リストを削除する
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

FreeMode

リスト要素の削除に使用されるメモリ管理のフラグを取得します。

```
void  FreeMode(
   bool  mode       // 新しい値
  )
```

パラメータ

mode

[in] メモリ管理フラグの新しい値

注意事項

メモリ管理のフラグの設定は CList クラス使用にあたって重要な部分です。リスト要素は動的オブジェクトへのポインタであるため、リストから削除する際にどのように処理するかを決定することが重要です。

フラグが設定されている場合は、リストから除去された際に、要素は自動的に delete オペレータによって削除されます。フラグが設定されていない場合は、削除されたオブジェクトへのポインタがユーザプログラムのどこかに残っていて、後にプログラムによって解放されると想定されています。

ユーザプログラムががメモリ管理フラグをリセットした場合は、ユーザは、プログラムの終了前に配列除去の責任があることを理解する必要があります。さもないと、 new 演算子によって要素に割り当てられたメモリが解放されません。

データが大量の場合、端末の動作が停止することもあります。

ユーザがメモリ管理フラグをリセットしない場合は、別の問題が起こる可能性があります。リストポインタがローカル変数に格納されている場合、リスト除去は重大なエラーにつながり、ユーザプログラムをクラッシュします。デフォルトでは、メモリ管理フラグが設定されていて、メモリ要素を解放する責任はリストクラスにあります。

例:

```
//--- CList::FreeMode(bool) の例
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- メモリ管理フラグをリセットする
   list.FreeMode(false);
   //--- リストを使用する
   //--- . にて。にて。
   //--- リストを削除する
   delete list;
```

© 2000-2025, MetaQuotes Ltd.

```
   }
```

© 2000–2025, MetaQuotes Ltd.

Total

リストの要素数を取得します。

```
int  Total() const
```

戻り値

リストの要素数

例：

```
//--- CList::Total() の例
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 合計をチェックする
   int total=list.Total();
   //--- リストを使用する
   //--- ...
   //--- リストを削除する
   delete list;
  }
```

© 2000–2025, MetaQuotes Ltd.

IsSorted

Gets the sorted list flag.

```
bool  IsSorted(
   int  mode=0       // ソートのモード
   ) const
```

パラメータ

mode=0

[in] チェックされたソート のモード

戻り値

ソート されたリスト のフラグ。リスト がソート されている場 合 は true、それ以 外 の場 合 は false

注 意 事 項

このフラグは直 接 変 更 出 来 ません。フラグは Sort (int) によって設 定 され、追 加 /挿 入 メソッド でリセット されます。

例 :

```
//--- CList::IsSorted() の例
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソートされているかをチェックする
   if(list.IsSorted(0))
     {
      //--- ソートされたリストのメソッドを使用する
      //--- ...
     }
   //--- リストを削除する
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

SortMode

ソートのモードを取得します。

```
int  SortMode() const
```

戻り値

ソートのオプション。リストがソートされていない場合は -1

例：

```
//--- CList::SortMode() の例
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ソートのモードをチェックする
   int sort_mode=list.SortMode();
   //--- リストを使用する
   //--- ...
   //--- リストを削除する
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

CreateElement

リストで新しい項目を作成します。

```
CObject*  CreateElement()
```

戻り値

成功の場合は作成された要素へのポインタ、要素が作成出来ない場合は NULL

注意事項

CList クラスの CreateElement() メソッドは常に NULL を戻し何のアクションも取りません。派生クラスで CreateElement() メソッドが必要に応じて実装されるべきです。

例:

```
//--- CList::CreateElement(int) の例
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   int     size=100;
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- リストに書き込む
   for(int i=0;i<size;i++)
     {
      CObject *object=list.CreateElement();
      if(object==NULL)
        {
         printf("Element create error");
         delete list;
         return;
        }
      list.Add(object);
     }
   //--- リストを使用する
   //--- . にて。にて。
   //--- リストを削除する
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

Add

リストの最後に要素を追加します。

```
int  Add(
   CObject*  element      // 追加する要素
  )
```

パラメータ

element

[in] リストに追加される要素の値

戻り値

成功の場合は追加された要素のインデックス、エラーの場合は -1

注意事項

NULLなどの不正なポインタがパラメータに渡された場合、要素はリストに追加されません。

例:

```
//--- CList::Add(CObject*) の例
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 100 要素を追加する
   for(int i=0;i<100;i++)
     {
      if(list.Add(new CObject)==-1)
        {
         printf("Element addition error");
         delete list;
         return;
        }
     }
   //--- リストを使用する
   //--- . にて。にて。
   //--- リストを削除する
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

Insert

指定された位置にリストの要素を挿入します。

```
int  Insert(
   CObject*  element,      // 挿入する要素
   int       pos           // 位置
  )
```

パラメータ

element

[in] リストに挿入される要素の値

pos

[in] リスト内の挿入位置

戻り値

成功の場合は挿入された要素のインデックス、エラーの場合は -1

注意事項

NULLなどの不正なポインタがパラメータに渡された場合、要素はリストに追加されません。

例:

```
//--- CList::Insert(CObject*,int) の例
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 100 要素を挿入する
   for(int i=0;i<100;i++)
     {
      if(list.Insert(new CObject,0)==-1)
        {
         printf("Element insert error");
         delete list;
         return;
        }
     }
   //--- リストを使用する
   //--- . にて。にて。
   //--- リストを削除する
```

© 2000-2025, MetaQuotes Ltd.

```
   delete list;
  }
```

© 2000–2025, MetaQuotes Ltd.

DetachCurrent

現在の位置から要素を「物理的に」削除せずに抽出します。

```
CObject*  DetachCurrent()
```

戻り値

成功の場合は削除された要素へのポインタ、要素が削除できない場合は NULL

注意事項

When removed from the list, the item is not removed in any state of the memory management flag.
抽出された要素へのポインタは使用後に解放されなければいけません。

例:

```
//--- CList::DetachCurrent() の例
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- リスト要素を追加する
   //--- . にて。にて。
   CObject *object=list.DetachCurrent();
   if(object==NULL)
     {
      printf("Detach error");
      delete list;
      return;
     }
   //--- 要素を使用する
   //--- . にて。にて。
   //--- 要素を削除する
   delete object;
   //--- リストを削除する
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteCurrent

リストの現在の位置から要素を削除します。

```
bool  DeleteCurrent()
```

戻り値

成功の場合はtrue、要素が削除できなかった場合はfalse

注意事項

メモリ管理フラグが有効であれば、削除された要素に使用されていたメモリが解放されます。

例：

```
//--- CList::DeleteCurrent() の例
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- リスト要素を追加する
   //--- . にて。にて。
   if(!list.DeleteCurrent())
     {
      printf("Delete error");
      delete list;
      return;
     }
   //--- リストを削除する
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

リストの指定された位置から要素を削除します。

```
bool  Delete(
   int  pos      // 位置
  )
```

パラメータ

pos

[in] リストから削除される要素の位置

戻り値

成功の場合は true、要素が削除できなかった場合は false

注意事項

メモリ管理フラグが有効であれば、削除された要素に使用されていたメモリが解放されます。

例:

```
//--- CList::Delete(int) の例
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- リスト要素を追加する
   //--- . にて。にて。
   if(!list.Delete(0))
     {
      printf("Delete error");
      delete list;
      return;
     }
   //--- リストを削除する
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

Clear

リストの全要素を削除します。

```
void  Clear()
```

注意事項

メモリ管理フラグが有効であれば、削除された要素に使用されていたメモリが解放されます。

例:

```
//--- CList::Clear() の例
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- リスト要素を追加する
   //--- . にて。にて。
   //--- リストをクリアする
   list.Clear();
   //--- リストを削除する
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

IndexOf

リスト要素のインデックスを取得します。

```
int  IndexOf(
   CObject*  element       // 要素へのポインタ
  )
```

パラメータ

element

[in] リスト要素へのポインタ

戻り値

リスト要素のインデックス、もしくは -1

例:

```
//--- CList::IndexOf(CObject*) の例
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   CObject *object=new CObject;
   if(object==NULL)
     {
      printf("Element create error");
      delete list;
      return;
     }
   if(list.Add(object))
     {
      int pos=list.IndexOf(object);
     }
   //--- リストを削除する
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

GetNodeAtIndex

リストで指定されたインデックスを持つ項目を取得します。

```
CObject*  GetNodeAtIndex(
   int  pos      // 位置
  )
```

パラメータ

pos

[in] リスト内の項目のインデックス

戻り値

成功の場合は最初の要素へのポインタ、要素が取得出来ない場合は NULL

例:

```
//--- CList::GetNodeAtIndex(int) の例
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- リスト要素を追加する
   //--- . にて。にて。
   CObject *object=list.GetNodeAtIndex(10);
   if(object==NULL)
     {
      printf("Get node error");
      delete list;
      return;
     }
   //--- 要素を使用する
   //--- . にて。にて。
   //--- 要素を削除しない
   //--- リストを削除する
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

GetFirstNode

リストの最初の要素を取得します。

```
CObject*  GetFirstNode()
```

戻り値

成功の場合は最初の要素へのポインタ、要素が取得出来ない場合は NULL

例：

```
//--- CList::GetFirstNode() の例
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- リスト要素を追加する
   //--- . にて。にて。
   CObject *object=list.GetFirstNode();
   if(object==NULL)
     {
      printf("Get node error");
      delete list;
      return;
     }
   //--- 要素を使用する
   //--- . にて。にて。
   //--- 要素を削除しない
   //--- リストを削除する
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

GetPrevNode

リストの前の要素を取得します。

```
CObject*  GetPrevNode()
```

戻り値

成功の場合は前の要素へのポインタ、要素が取得出来ない場合は NULL

例：

```
//--- CList::GetPrevNode() の例
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- リスト要素を追加する
   //--- . にて。にて。
   CObject *object=list.GetPrevNode();
   if(object==NULL)
     {
      printf("Get node error");
      delete list;
      return;
     }
   //--- 要素を使用する
   //--- . にて。にて。
   //--- 要素を削除しない
   //--- リストを削除する
   delete list;
  }
```

© 2000–2025, MetaQuotes Ltd.

GetCurrentNode

Gets the current list element.

```
CObject*  GetCurrentNode()
```

戻り値

成功の場合は現在の要素へのポインタ、要素が取得出来ない場合は NULL

例:

```
//--- CList::GetCurrentNode() の例
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- リスト要素を追加する
   //--- . にて。にて。
   CObject *object=list.GetCurrentNode();
   if(object==NULL)
     {
      printf("Get node error");
      delete list;
      return;
     }
   //--- 要素を使用する
   //--- . にて。にて。
   //--- 要素を削除しない
   //--- リストを削除する
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

GetNextNode

Gets the next element in the list.

```
CObject*  GetNextNode()
```

戻り値

成功の場合は次の要素へのポインタ、要素が取得出来ない場合は NULL

例：

```
//--- CList::GetNextNode() の例
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- リスト要素を追加する
   //--- . にて。にて。
   CObject *object=list.GetNextNode();
   if(object==NULL)
     {
      printf("Get node error");
      delete list;
      return;
     }
   //--- 要素を使用する
   //--- . にて。にて。
   //--- 要素を削除しない
   //--- リストを削除する
   delete list;
  }
```

© 2000–2025, MetaQuotes Ltd.

GetLastNode

リストの最後の要素を取得します。

```
CObject*  GetLastNode()
```

戻り値

成功の場合は最後の要素へのポインタ、要素が取得出来ない場合は NULL

例：

```
//--- CList::GetLastNode() の例
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- リスト要素を追加する
   //--- . にて。にて。
   CObject *object=list.GetLastNode();
   if(object==NULL)
     {
      printf("Get node error");
      delete list;
      return;
     }
   //--- 要素を使用する
   //--- . にて。にて。
   //--- 要素を削除しない
   //--- リストを削除する
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

Sort

リストをソートします。

```
void  Sort(
   int  mode      // ソートのモード
  )
```

パラメータ

mode

[in] ソートのモード

戻り値

No.

注意事項

リストのソートは常に昇順です。

例:

```
//--- CList::Sort(int) の例
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- モード 0 でのソート
   list.Sort(0);
   //--- リストを使用する
   //--- ...
   //--- リストを削除する
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

MoveToIndex

リストの現在の要素を指定された位置に移動します。

```
bool  MoveToIndex(
   int  pos      // 位置
  )
```

パラメータ

pos

[in] 移動する要素のリスト内の位置

戻り値

成功の場合は true、要素が移動できなかった場合は false

例:

```
//--- CList::MoveToIndex(int) の例
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 初めに現在の要素を移動する
   list.MoveToIndex(0);
   //--- リストを使用する
   //--- . にて。にて。
   //--- リストを削除する
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

Exchange

リストの 2 つの要素を交換します。

```
bool  Exchange(
   CObject*  node1,     // リスト要素
   CObject*  node2      // リスト要素
  )
```

パラメータ

node1

[in] リスト要素

node2

[in] リスト要素

戻り値

成功の場合は true、要素が交換できなかった場合は false

例:

```
//--- CList::Exchange(CObject*, CObject*) の例
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 交換する
   list.Exchange(list.GetFirstNode(),list.GetLastNode());
   //--- リストを使用する
   //--- . にて。にて。
   //--- リストを削除する
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareList

リストを別のリストと比較します。

```
bool  CompareList(
   CList*  list      // 比較先
  )
```

パラメータ

list

[in] 比較する要素へのポインタ（CList クラスインスタンス）

戻り値

リストが同一なら true、それ以外の場合は false

例：

```
//--- CList::CompareList(const CList*) の例
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- 比較元のリストを作成する
   CList *src=new CList;
   if(src==NULL)
     {
      printf("Object create error");
      delete list;
      return;
     }
   //--- 比較元のリストに書き込む
   //--- . にて。にて。
   //--- あと 1 つのリストと比較する
   bool result=list.CompareList(src);
   //--- リストを削除する
   delete src;
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

Search

ソートされたリスト内でサンプルに等しい要素を検索します。

```
CObject*  Search(
   CObject*  element      // サンプル
  )
```

パラメータ

element

[in] リスト内で探すサンプル要素

戻り値

成功の場合は見つかった要素のインデックス、要素が見つからなかった場合は NULL

例:

```
//--- CList::Search(CObject*) の例
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- リスト要素を追加する
   //--- . にて。にて。
   //--- リストをソートする
   list.Sort(0);
   //--- サンプルを作成する
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete list;
      return;
     }
   //--- サンプルの属性を設定する
   //--- . にて。にて。
   //--- 要素を探す
   if(list.Search(sample)!=NULL) printf("Element found");
   else                          printf("Element not found");
   //--- リストを削除する
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

Saves list data in a file.

```
virtual bool  Save(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に FileOpen() 関数で開かれたバイナリファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

例:

```
//--- CList::Save(int) の例
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CList *list=new CList;
   //---
   if(list!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- リスト要素を追加する
   //--- . にて。にて。
   //--- ファイルを開く
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!list.Save(file_handle))
        {
         //--- ファイル保存エラー
         printf("File save: Error %d!",GetLastError());
         delete list;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- リストを削除する
   delete list;
```

© 2000–2025, MetaQuotes Ltd.

```
   }
```

© 2000–2025, MetaQuotes Ltd.

Load

リスト データをファイルから読み込みます。

```
virtual bool  Load(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に FileOpen() 関数で開かれたバイナリファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

注意事項

When reading list elements from the file, the CList::CreateElement(int) method is called to create each element.

例:

```
//--- CLoad::Load(int) の例
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CList *list=new CList;
   //---
   if(list!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ファイルを開く
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!list.Load(file_handle))
        {
         //--- ファイル読み込みエラー
         printf("File load: Error %d!",GetLastError());
         delete list;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- リスト要素を使用する
   //--- .  にて。にて。
   //--- リストを削除する
   delete list;
  }
```

© 2000–2025, MetaQuotes Ltd.

Type

Gets the type identifier of a list.

```
virtual int  Type()
```

戻り値

型の識別子（CList は 7779）

例：

```
//--- CList::Type() の例
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- リストの型を取得する
   int type=list.Type();
   //--- リストを削除する
   delete list;
  }
```

© 2000–2025, MetaQuotes Ltd.

CTreeNode

CTreeNode は CTree 二進木の節点（ノード）のクラスです。

説明

CTreeNode は CTree 二進木の節点を操作する機能を提供します。ツリーのナビゲーションのオプションは、このクラスで実装されています。その他に、ファイル操作のメソッドも実装されています。

宣言

```
class CTreeNode : public CObject
```

タイトル

```
#include <Arrays\TreeNode.mqh>
```

継承階層

CObject
CTreeNode

直接子孫

CTree

クラスメソッド

属性	
Owner	所有者ノードのポインタを取得/設定します。
Left	左ノードのポインタを取得/設定します。
Right	右ノードのポインタを取得/設定します。
Balance	ノードのバランスを取得します。
BalanceL	ノードの左サブブランチのバランスを取得します。
BalanceR	ノードの右サブブランチのバランスを取得します。
新しい要素の作成	
CreateSample	新しいノードインスタンスを作成します。
比較	
RefreshBalance	ノードのバランスを再計算します。
検索	
GetNext	次のノードのポインタを取得します。
入出力	
SaveNode	ノードデータをファイルに保存します。

© 2000–2025, MetaQuotes Ltd.

属性	
LoadNode	ノードデータをファイルから読み込みます。
virtual Type	ノード型の識別子を取得します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Compare

実際には CTreeNode クラスの派生クラスのツリーが使用されます。

CTreeNode の派生クラスは事前に定義されたメソッドを持たなければなりません。CreateSample は CTreeNode の派生クラスの新しいインスタンスを作成します。Compare は CTreeNode の派生クラスのキーフィールドを比較します。Type はノードの識別に必要で、SaveNode 及び LoadNode はファイル操作に必要です。

CTree 派生クラスの例を見てみましょう。

```
//+------------------------------------------------------------------+
//|                                                   MyTreeNode.mq5 |
//|                        Copyright 2010, MetaQuotes Software Corp. |
//|                                        https://www.metaquotes.net/ |
//+------------------------------------------------------------------+
#property copyright "2010, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
//---
#include <Arrays\TreeNode.mqh>
//+------------------------------------------------------------------+
//| CTreeNode から派生した CMyTreeNode クラスを記述する               |
//+------------------------------------------------------------------+
//| CMyTreeNode クラス                                               |
//| 目的:二分木の要素のクラス                                        |
//|              CTreeNode の派生クラス                              |
//+------------------------------------------------------------------+
class CMyTreeNode : public CTreeNode
  {
protected:
   //--- ユーザデータ
   long              m_long;           // long 型のキーフィールド
   double            m_double;         // double 型のカスタム変数
   string            m_string;         // string 型のカスタム変数
   datetime          m_datetime;       // datetime 型のカスタム変数

public:
                     CMyTreeNode();
   //--- ユーザデータアクセスに使用されるメソッド
   long              GetLong(void)               { return(m_long); }
   void              SetLong(long value)         { m_long=value;   }
   double            GetDouble(void)             { return(m_double); }
   void              SetDouble(double value)     { m_double=value;   }
```

© 2000-2025, MetaQuotes Ltd.

```
   string            GetString(void)               { return(m_string); }
   void              SetString(string value)       { m_string=value;  }
   datetime          GetDateTime(void)             { return(m_datetime); }
   void              SetDateTime(datetime value)   { m_datetime=value;  }
   //--- ファイル操作のメソッド
   virtual bool      Save(int file_handle);
   virtual bool      Load(int file_handle);
protected:
   virtual int       Compare(const CObject *node,int mode);
   //--- クラスインスタンス作成のメソッド
   virtual CTreeNode* CreateSample();
  };
//+------------------------------------------------------------------+
//| CMyTreeNode クラスコンストラクタ                                 |
//| 入力:なし                                                        |
//| 出力:なし                                                        |
//| メモ:なし                                                        |
//+------------------------------------------------------------------+
void CMyTreeNode::CMyTreeNode()
  {
//--- ユーザデータの初期化
   m_long         =0;
   m_double       =0.0;
   m_string       ="";
   m_datetime     =0;
  }
//+------------------------------------------------------------------+
//| 指定されたアルゴリズムでの他のツリーノードとの比較               |
//| 入力: node - 比較されるツリー要素                                |
//|          mode - 比較アルゴリズムの識別子                         |
//| 出力:比較結果(>0、0、<0 )                                        |
//| メモ:なし                                                        |
//+------------------------------------------------------------------+
int CMyTreeNode::Compare(const CObject *node,int mode)
  {
//--- ツリー構築アルゴリズムが 1 つだけであるため、パラメータモードが無視される
   int res=0;
//--- 明示的な型キャスト
   CMyTreeNode *n=node;
   res=(int)(m_long-n.m_long);
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| 新しいクラスインスタンスの作成                                   |
//| 入力:なし                                                        |
//| 出力:CMyTreeNodeクラスの新しいインスタンスへのポインタ           |
//| メモ:なし                                                        |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
CTreeNode* CMyTreeNode::CreateSample()
  {
   CMyTreeNode *result=new CMyTreeNode;
//---
   return(result);
  }
//+------------------------------------------------------------------+
//| ツリーノードデータをファイルに書き込みます。                                  |
//| 入力:file_handle - 既に書き込みのために開かれたファイルのハンドル    |
//| 出力:成功の場合は true、それ以外の場合は false                              |
//| メモ:なし                                                        |
//+------------------------------------------------------------------+
bool CMyTreeNode::Save(int file_handle)
  {
   uint i=0,len;
//--- チェック
   if(file_handle<0) return(false);
//--- ユーザデータの書き込み
//--- long 型のカスタム変数の書き込み
   if(FileWriteLong(file_handle,m_long)!=sizeof(long))           return(false);
//--- double 型のカスタム変数の書き込み
   if(FileWriteDouble(file_handle,m_double)!=sizeof(double))     return(false);
//--- string 型のカスタム変数の書き込み
   len=StringLen(m_string);
//--- 文字列の長さを書き込む
   if(FileWriteInteger(file_handle,len,INT_VALUE)!=INT_VALUE)    return(false);
//--- 文字列を書き込む
   if(len!=0 && FileWriteString(file_handle,m_string,len)!=len) return(false);
//--- datetime 型のカスタム変数の書き込み
   if(FileWriteLong(file_handle,m_datetime)!=sizeof(long))       return(false);
//---
   return(true);
  }
//+------------------------------------------------------------------+
//| ファイルからツリーノードデータを読み込む                                  |
//| 入力: file_handle - 既に読み込みのために開かれたファイルのハンドル    |
//| 出力:成功の場合は true、それ以外の場合は false                              |
//| メモ:なし                                                        |
//+------------------------------------------------------------------+
bool CMyTreeNode::Load(int file_handle)
  {
   uint i=0,len;
//--- チェック
   if(file_handle<0) return(false);
//--- 読み込み
   if(FileIsEnding(file_handle)) return(false);
//--- char 型のカスタム変数の読み込み
//--- long 型のカスタム変数の読み込み
   m_long=FileReadLong(file_handle);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- double 型のカスタム変数の読み込み
   m_double=FileReadDouble(file_handle);
//--- string 型のカスタム変数の読み込み
//--- 文字列の長さを読み込む
   len=FileReadInteger(file_handle,INT_VALUE);
//--- 文字列を読み込む
   if(len!=0) m_string=FileReadString(file_handle,len);
   else       m_string="";
//--- datetime 型のカスタム変数の読み込み
   m_datetime=FileReadLong(file_handle);
//---
   return(true);
  }
```

© 2000–2025, MetaQuotes Ltd.

Owner

所有者ノードのポインタを取得します。

```
CTreeNode*  Owner()
```

戻り値

所有者ノードへのポインタ

Owner

所有者ノードのポインタを設定します。

```
void  Owner(
   CTreeNode*  node      // ノード
  )
```

パラメータ

node

[in] 所有者ノードの新しいポインタの値

戻り値

なし

© 2000-2025, MetaQuotes Ltd.

Left

左ノードのポインタを取得します。

```
CTreeNode*  Left()
```

戻り値

左ノードのポインタ

Left

左ノードのポインタを設定します。

```
void  Left(
   CTreeNode*  node      // ノード
  )
```

パラメータ

node

[in] 新しい左ノードへのポインタ

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

Right

右ノードのポインタを取得します。

```
CTreeNode*  Right()
```

戻り値

右ノードのポインタ

Right

右ノードのポインタを設定します。

```
void  Right(
   CTreeNode*  node      // ノード
  )
```

パラメータ

node

[in] 右ノードへの新しいポインタの値

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

Balance

ノードのバランスを取得します。

```
int  Balance() const
```

戻り値

ノードのバランス

© 2000–2025, MetaQuotes Ltd.

BalanceL

ノードの左サブブランチのバランスを取得します。

```
int  BalanceL() const
```

戻り値

ノードの左サブブランチのバランス

© 2000–2025, MetaQuotes Ltd.

BalanceR

ノードの右サブブランチのバランスを取得します。

```
int  BalanceR() const
```

戻り値

ノードの右サブブランチのバランス

© 2000–2025, MetaQuotes Ltd.

CreateSample

新しいノード サンプルを作 成します。

```
virtual CTreeNode*  CreateSample()
```

戻り値

新しいノード サンプルへのポインタ、または NULL

© 2000–2025, MetaQuotes Ltd.

RefreshBalance

ノードのバランスを再計算します。

```
int  RefreshBalance()
```

戻り値

ノードのバランス

© 2000–2025, MetaQuotes Ltd.

GetNext

次のノードのポインタを取得します。

```
CTreeNode*  GetNext(
   CTreeNode*  node      // ノード
  )
```

パラメータ

node

[in] 基準となるノード

戻り値

次のノードのポインタ

© 2000–2025, MetaQuotes Ltd.

SaveNode

ノードデータをファイルに書き込みます。

```
bool  SaveNode(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に書き込みのために開かれたバイナリファイルのハンドル

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

LoadNode

ファイルからノードデータを読み込みます。

```
bool  LoadNode(
   int          file_handle,     // ファイルハンドル
   CTreeNode*   main             // ノード
  )
```

パラメータ

file_handle

[in] 既に読み込みのために開かれたバイナリファイルのハンドル

main

[in] ノードのデータ

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

Type

ノード型の識別子を取得します。

```
virtual int  Type() const
```

戻り値

ノード型の識別子

© 2000–2025, MetaQuotes Ltd.

CTree

CTree はCTreeNode とその派生クラスを節点とする二進木のクラスです。

説明

CTree は CTreeNode クラスインスタンスとその派生クラスを節点（ノード）とした二進木の操作を提供します。ツリー要素の追加/挿入/削除のオプションとツリー内の検索は、クラスに実装されています。ファイル操作のメソッドも実装されています。

CTree では動的メモリ管理は実装されていません（これは CList 及び CArrayObj とは異なります）。全てのツリーノードは、メモリ解放で削除されます。

宣言

```
class CTree : public CTreeNode
```

タイトル

```
#include <Arrays\Tree.mqh>
```

継承階層

CObject
  CTreeNode
    CTree

クラスメソッド

属性	
Root	ツリーの根（ルート）を取得します。
新しい要素の作成	
CreateElement	ノードの新しいインスタンスを作成します。
書き込み	
Insert	ツリーにノードを追加します。
削除	
Detach	指定されたノードをツリーから取り外します。
Delete	指定されたノードをツリーから削除します。
Clear	ツリーの全てのノードを削除します。
検索	
Find	ツリー内でサンプルによってノードを探します。
入出力	
virtual Save	全てのツリーデータをファイルに保存します。

© 2000-2025, MetaQuotes Ltd.

属性	
virtual Load	ファイルからツリーデータを読み込みます。
virtual Type	ツリー型の識別子を取得します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CTreeNode

Parent, Parent, Left, Left, Right, Right, Balance, BalanceL, BalanceR, RefreshBalance, GetNext, SaveNode, LoadNode

実際には CTreeNode クラスの派生クラスを持った CTree 派生クラスのツリーが使用されます。

CTree の派生クラスは事前に定義された CreateElement メソッドを持たなければいけません。このメソッドは新しい CTreeNode の派生クラスのノードを作成します。

CTree の派生クラスの例を見てみましょう。

```
//+------------------------------------------------------------------+
//|                                                       MyTree.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//| www.metaquotes.net |
//+------------------------------------------------------------------+
#property copyright "2010, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
//---
#include <Arrays\Tree.mqh>
#include "MyTreeNode.mqh"
//---
input int extCountedNodes = 100;
//+------------------------------------------------------------------+
//| CTree. から派生された CMyTree クラスを記述する                          |
//+------------------------------------------------------------------+
//| CMyTree クラス                                                    |
//| 目的:二分探索木の建築とナビゲーション    |
//+------------------------------------------------------------------+
class CMyTree : public CTree
  {
public:
   //--- カスタムデータによるツリーの検索のメソッド
   CMyTreeNode*         FindByLong(long find_long);
   //--- ツリー要素作成のメソッド
   virtual CTreeNode *CreateElement();
  };
//---
CMyTree MyTree;
//+------------------------------------------------------------------+
//| 新しいツリーノードの作成                                          |
```

© 2000-2025, MetaQuotes Ltd.

```
//|  入力:なし                                                       |
//|  出力:成功した場合は、新しいツリーノードのポインタ、または NULL          |
//|  メモ:なし                                                       |
//+------------------------------------------------------------------+
CTreeNode *CMyTree::CreateElement()
  {
   CMyTreeNode *node=new CMyTreeNode;
//---
   return(node);
  }
//+------------------------------------------------------------------+
//|  m_long の値によってのリスト内の要素検索                          |
//|  入力:find_long - 探される値                                      |
//|  出力:見つかった要素のポインタ、または NULL                       |
//|  メモ:なし                                                       |
//+------------------------------------------------------------------+
CMyTreeNode* CMyTree::FindByLong(long find_long)
  {
   CMyTreeNode *res=NULL;
   CMyTreeNode *node;
//--- 検索パラメータを渡すためにツリーノードを作成する
   node=new CMyTreeNode;
   if(node==NULL) return(NULL);
   node.SetLong(find_long);
//---
   res=Find(node);
   delete node;
//---
   return(res);
  }
//+------------------------------------------------------------------+
//|  「CMyTree クラスのテスト」スクリプト                             |
//+------------------------------------------------------------------+
//---  文字列初期化の配列
string str_array[11]
={"p","oo","iii","uuuu","yyyyy","ttttt","rrrr","eee","ww","q","999"};
//---
int OnStart() export
  {
   int           i;
   uint          pos;
   int           beg_time,end_time;
   CMyTreeNode *node; //--- CMyTreeNode クラスサンプルへの一時的なポインタ
//---
   printf("Start test %s.",__FILE__);
//--- MyTree に extCountedNodes 個の MyTreeNode クラスインスタンスのノードを書き込む
   beg_time=GetTickCount();
   for(i=0;i<extCountedNodes;i++)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      node=MyTree.CreateElement();
      if(node==NULL)
        {
         //--- 緊急終了
         printf("%s (%4d): create error",__FILE__,__LINE__);
         return(__LINE__);
        }
      NodeSetData(node,i);
      node.SetLong(i);
      MyTree.Insert(node);
     }
   end_time=GetTickCount();
   printf("Filling time of MyTree is %d ms.",end_time-beg_time);
//--- 一時的な TmpMyTree ツリーを作成する
   CMyTree TmpMyTree;
//--- 半分(偶数)のツリー要素を取り外す
//--- 一時的な TmpMyTree に追加する
   beg_time=GetTickCount();
   for(i=0;i<extCountedNodes;i+=2)
     {
      node=MyTree.FindByLong(i);
      if(node!=NULL)
         if(MyTree.Detach(node)) TmpMyTree.Insert(node);
     }
   end_time=GetTickCount();
   printf("Deletion time of %d elements from MyTree is %d
ms.",extCountedNodes/2,end_time-beg_time);
//--- 取り外した物を返す
   node=TmpMyTree.Root();
   while(node!=NULL)
     {
      if(TmpMyTree.Detach(node)) MyTree.Insert(node);
      node=TmpMyTree.Root();
     }
//--- Save(int file_handle); メソッドの操作をチェックする
   int file_handle;
   file_handle=FileOpen("MyTree.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!MyTree.Save(file_handle))
        {
         //--- ファイル書き込みエラー
         //--- 緊急終了
         printf("%s: Error %d in %d!",__FILE__,GetLastError(),__LINE__);
         //--- 終了前にファイルを閉じる
         FileClose(file_handle);
         return(__LINE__);
        }
      FileClose(file_handle);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- Load(int file_handle); メソッドの操作をチェックする
   file_handle=FileOpen("MyTree.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!TmpMyTree.Load(file_handle))
        {
         //--- ファイル読み込みエラー
         //--- 緊急終了
         printf("%s: Error %d in %d!",__FILE__,GetLastError(),__LINE__);
         //--- 終了前にファイルを閉じる
         FileClose(file_handle);
         return(__LINE__);
        }
      FileClose(file_handle);
     }
//---
   MyTree.Clear();
   TmpMyTree.Clear();
//---
   printf("End test %s. OK!",__FILE__);
//---
   return(0);
  }
//+------------------------------------------------------------------+
//| ノードの内容を操作ログに出力する関数                             |
//+------------------------------------------------------------------+
void NodeToLog(CMyTreeNode *node)
  {
   printf("   %I64d,%f,'%s','%s'",
               node.GetLong(),node.GetDouble(),
               node.GetString(),TimeToString(node.GetDateTime()));
  }
//+------------------------------------------------------------------+
//| ノードに乱数を書き込む関数                                       |
//+------------------------------------------------------------------+
void NodeSetData(CMyTreeNode *node,int mode)
  {
   if(mode%2==0)
     {
      node.SetLong(mode*MathRand());
      node.SetDouble(MathPow(2.02,mode)*MathRand());
     }
   else
     {
      node.SetLong(mode*(long)(-1)*MathRand());
      node.SetDouble(-MathPow(2.02,mode)*MathRand());
     }
   node.SetString(str_array[mode%10]);
```

© 2000-2025, MetaQuotes Ltd.

```
   node.SetDateTime(10000*mode);
  }
```

© 2000–2025, MetaQuotes Ltd.

Root

ツリーの根（ルート）を取得します。

```
CTreeNode*  Root() const
```

戻り値

ツリーの根（ルート）のポインタ

© 2000–2025, MetaQuotes Ltd.

CreateElement

ノードの新しいインスタンスを作成します。

```
virtual CTreeNode*  CreateElement()
```

戻り値

ノードの新しいインスタンスへのポインタ、または NULL

© 2000–2025, MetaQuotes Ltd.

Insert

ツリーにノードを追加します。

```
CTreeNode*  Insert(
   CTreeNode*  new_node      // ノード
  )
```

パラメータ

new_node

[in] ツリーに追加されるノードのポインタ

戻り値

追加されたノードのポインタ、または NULL

© 2000-2025, MetaQuotes Ltd.

Detach

指定されたノードをツリーから取り外します。

```
bool  Detach(
   CTreeNode*  node      // ノード
  )
```

パラメータ

node

[in] 取り外されるノードポインタ

戻り値

成功の場合は true、それ以外の場合は false

注意事項

ノードポインタは取り外し後に解放されません。ツリーはバランスされます。

© 2000–2025, MetaQuotes Ltd.

Delete

指定されたノードをツリーから削除します。

```
bool  Delete(
   CTreeNode*  node      // ノード
  )
```

パラメータ

node

[in] 削除されるノードポインタ

戻り値

成功の場合は true、それ以外の場合は false

注意事項

ノードポインタは削除後に解放されます。ツリーはバランスされます。

© 2000-2025, MetaQuotes Ltd.

Clear

ツリーの全てのノードを削除します。

```
void  Clear()
```

戻り値

なし

注意事項

ノードポインタは削除後に解放されます。

© 2000–2025, MetaQuotes Ltd.

Find

ツリー内でサンプルによってノードを探します。

```
CTreeNode*  Find(
   CTreeNode*  node      // ノード
  )
```

パラメータ

node

[in] 探されるサンプルを含むノード

戻り値

見つかった要素のポインタ、または NULL

© 2000-2025, MetaQuotes Ltd.

Save

ツリーデータをファイルに書き込みます。

```
virtual bool  Save(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に書き込みのために開かれたバイナリファイルのハンドル

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

Load

ファイルからツリーデータを読み込みます。

```
virtual bool  Load(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に読み込みのために開かれたバイナリファイルのハンドル

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Type

ツリー型の識別子を取得します。

```
virtual int  Type() const
```

戻り値

ツリー型の識別子

© 2000–2025, MetaQuotes Ltd.

ジェネリックデータコレクション

このライブラリは、ジェネリックコレクションを定義するクラスとインターフェイスを提供し、ユーザが厳密に型指定されたコレクションを作成できるようにします。これらのコレクションは非ジェネリックの型指定されたコレクションよりも利便性とデータ処理パフォーマンスの面で優れています。

ライブラリは、ターミナル作業ディレクトリのInclude¥Genericフォルダにあります。

オブジェクト：

オブジェクト	説明	Type
ICollection	ジェネリックデータコレクションを実装するためのインターフェイス	インターフェイス
IEqualityComparable	比較可能なオブジェクトを実装するためのインターフェイス	インターフェイス
IComparable	「より大きい、より小さい、等しい」によって比較可能なオブジェクトを実装するためのインタフェース	インターフェイス
IComparer	T型の2つのオブジェクトをそのうち1つがあと1つと比べて「より大きい、より小さい、等しい」で比較するジェネリッククラスを実装するためのインタフェース	インターフェイス
IEqualityComparer	T型の2つのオブジェクトが等しいかどうかを比較するジェネリッククラスを実装するためのインタフェース	インターフェイス
IList	ジェネリックデータリストを実装するためのインタフェース	インターフェイス
IMap	キー/値ペアのジェネリックコレクションを実装するためのインターフェイス	インターフェイス
ISet	ジェネリックデータセットを実装するためのインタフェイス	インターフェイス
CDefaultComparer	Compareグローバルメソッドに基づいてIComparer<T>ジェネリックインターフェイスを実装するヘルパークラス	クラス
CDefaultEqualityComparer	Equals<T>およびGetHashCodeグローバルメソッドを使用してIEqualityComparer<T>ジェネリックインターフェイスを実装するヘルパークラス	クラス
CArrayList	IList<T>インターフェイスを実装するジェネリッククラス	クラス
CKeyValuePair	キー/値ペアを実装するクラス	クラス
CHashMap	IMap<TKey, TValue>インターフェイスを実装するジェネリッククラス	クラス

© 2000–2025, MetaQuotes Ltd.

オブジェクト	説明	Type
CHashSet	ISet<T>インターフェイスを実装するジェネリッククラス	クラス
CLinkedListNode	CLinkedListNode<T>クラスを実装するためのヘルパークラス	クラス
CLinkedList	ICollection<T>インターフェイスを実装するジェネリッククラス	クラス
CQueue	ICollection<T>インターフェイスを実装するジェネリッククラス	クラス
CRedBlackTreeNode	CRedBlackTree<T>クラスの実装に使用されるヘルパークラス	クラス
CRedBlackTree	ICollection<T>インターフェイスを実装するジェネリッククラス	クラス
CSortedMap	IMap<TKey, TValue>インターフェイスを実装するジェネリッククラス	クラス
CSortedSet	ISet<T>インターフェイスを実装するジェネリッククラス	クラス
CStack	ICollection<T>インターフェイスを実装するジェネリッククラス	クラス

グローバルメソッド：

メソッド	説明
ArrayBinarySearch	IComparable<T>を使用して要素を比較して昇順ソートされた1次元配列で指定された値を検索します。
ArrayIndexOf	1次元配列で値の最初の出現を検索します。
ArrayLastIndexOf	1次元配列で値の最後の出現を検索します。
ArrayReverse	1次元配列の要素の順序を変更します。
Compare	2つの値をそのうち1つがあと1つと比べて「より大きい、より小さい、等しい」かで比較します。
Equals	2つの値が等しいかどうかを比較します。
GetHashCode	ハッシュコード値を計算します。

© 2000–2025, MetaQuotes Ltd.

ICollection<T>

ICollection<T>はジェネリックデータコレクションを実装するためのインターフェイスです。

説明

ICollection<T>インターフェイスは、要素数をカウントするメソッド、コレクションをクリアするメソッド、要素を追加または削除するメソッドなどのコレクションを処理するための基本的なメソッドを提供します。

Declaration

```
template<typename T>
interface ICollection
```

ヘッダ

```
#include <Generic\Interfaces\ICollection.mqh>
```

継承階層

ICollection

直系子孫

CLinkedList, CQueue, CRedBlackTree, CStack, IList, IMap, ISet

クラスメソッド

メソッド	説明
Add	コレクションに要素を追加します。
kaCount	コレクションの要素数を返します。
Contains	コレクションに指定された値を持つ要素が含まれているかどうかを判断します。
CopyTo	コレクションのすべての要素を、指定された配列に指定されたインデックスから開始してコピーします。
Clear	コレクションからすべての要素を削除します。
Remove	コレクションから指定された要素の最初のオカレンスを削除します。

© 2000-2025, MetaQuotes Ltd.

Add

コレクションに要素を追加します。

```
bool Add(
   T  value     // 要素値
   );
```

パラメータ

value

[in] 追加する要素の値

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

kaCount

コレクションの要素数を返します。

```
int Count();
```

戻り値

要素数を返します。

© 2000–2025, MetaQuotes Ltd.

Contains

コレクションに指定された値を持つ要素が含まれているかどうかを判断します。

```
bool Contains(
   T  item     // 検索値
   );
```

パラメータ

item

[in] 検索値

戻り値

指定された値を持つ要素がコレクションにある場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

CopyTo

コレクションのすべての要素を、指定された配列に指定されたインデックスから開始してコピーします。

```
int CopyTo(
   T&          dst_array[],      // 書き入れ対象の配列
   const int   dst_start=0       // 書き始めのインデックス
   );
```

パラメータ

&dst_array[]

[out] コレクションの要素が書き込まれる配列

dst_start=0

[in] コピーを開始する配列のインデックス

戻り値

複製された要素の数を返します。

© 2000-2025, MetaQuotes Ltd.

Clear

コレクションからすべての要素を削除します。

```
void Clear();
```

© 2000–2025, MetaQuotes Ltd.

Remove

コレクションから指定された要素の最初のオカレンスを削除します。

```
bool Remove(
   T  item     // 要素値
   );
```

パラメータ

item

[in] 削除する要素の値

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

IEqualityComparable<T>

IEqualityComparable<T>は比較可能なオブジェクトを実装するためのインターフェイスです。

説明

The IEqualityComparable<T>インターフェイスは、現在のオブジェクトのハッシュコードを取得し、同じ型の別のオブジェクトと等しいかどうかをチェックするメソッドを定義します。

Declaration

```
template<typename T>
interface IEqualityComparable
```

ヘッダ

```
#include <Generic\Interfaces\IEqualityComparable.mqh>
```

継承階層

IEqualityComparable

直系子孫

IComparable

クラスメソッド

メソッド	説明
Equals	現在のオブジェクトを指定された値と比較します。
HashCode	現在のオブジェクトのハッシュコード値を計算します。

© 2000-2025, MetaQuotes Ltd.

Equals

現在のオブジェクトを指定された値と比較します。

```
bool Equals(
   T  value     // 比較する値
   );
```

パラメータ

value

[in] 現在のオブジェクトと比較する値

戻り値

オブジェクトが等しい場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

HashCode

現在のオブジェクトのハッシュコード値を計算します。

```
int HashCode();
```

戻り値

ハッシュコードを返します。

© 2000-2025, MetaQuotes Ltd.

IComparable<T>

IComparable<T>は「より大きい、より小さい、等しい」によって比較可能なオブジェクトを実装するためのインタフェースです。

説明

IComparable<T>インタフェースは、現在のオブジェクトを同じ型の別のオブジェクトと比較するメソッドを定義します。これらのオブジェクトのコレクションはこれを基にしてソートできます。

Declaration

```
template<typename T>
interface IComparable : public IEqualityComparable<T>
```

ヘッダ

```
#include <Generic\Interfaces\IComparable.mqh>
```

継承階層

IEqualityComparable

IComparable

直系子孫

CKeyValuePair

クラスメソッド

メソッド	説明
Compare	現在のオブジェクトを指定された値と比較します。

© 2000–2025, MetaQuotes Ltd.

Compare

現在のオブジェクトを指定された値と比較します。

```
int Compare(
   T  value     // 比較する値
   );
```

パラメータ

value

[in] 現在のオブジェクトと比較する値

戻り値

現在のオブジェクトと渡されたオブジェクトのの比率を表す数値を返します。

- 結果が0より小さい場合、現在のオブジェクトは渡されたものより小さいです。
- 結果が0の場合、現在のオブジェクトは渡されたものと等しいです。
- 結果が0より大きい場合、現在のオブジェクトは渡されたものより大きいです。

© 2000–2025, MetaQuotes Ltd.

IComparer<T>

IComparer<T>はT型の2つのオブジェクトをそのうち1つがあと1つと比べて「より大きい、より小さい、等しい」で比較するジェネリッククラスを実装するためのインタフェースです。

説明

The IComparer<T>インターフェイスは、T型の2つのオブジェクトを比較するメソッドを決定します。これらのオブジェクトのコレクションはこれを基にしてソートできます。

Declaration

```
template<typename T>
interface IComparer
```

ヘッダ

```
#include <Generic\Interfaces\IComparer.mqh>
```

継承階層

IComparer

直系子孫

CDefaultComparer

クラスメソッド

メソッド	説明
Compare	T型の2つの値を比較します。

© 2000–2025, MetaQuotes Ltd.

Compare

T型の2つの値を比較します。

```
int Compare(
   T  x,      // 1番目の値
   T  y       // 2番目の値
   );
```

パラメータ

x

[in] 比較される1番目の値

y

[in] 比較される1番目の値

戻り値

比較された2つの値の比を表す数値を返します。

- 結果が0より小さい場合、xはyより小さい(x <y)です。
- 結果がゼロに等しい場合、xはyに等しい(x = y)です。
- 結果がゼロより大きい場合、xはyより大きい(x> y)です。

© 2000-2025, MetaQuotes Ltd.

IEqualityComparer<T>

IEqualityComparer<T>はT型 の2つのオブジェクト を比 較 するジェネリッククラスを実 装 するためのインタフェース です。

説 明

IEqualityComparer<T>インタフェースは、T型 オブジェクト のハッシュコード を取 得 し、T型 の2つのオブジェクト が 等 しいかどうかを確 認 するメソッド を定 義 します。

Declaration

```
template<typename T>
interface IEqualityComparer
```

ヘッダ

```
#include <Generic\Interfaces\IEqualityComparer.mqh>
```

継 承 階 層

IEqualityComparer

直 系 子 孫

CDefaultEqualityComparer

クラスメソッド

メソッド	説 明
Equals	T型 の2つの値 を比 較 します。
HashCode	T型 オブジェクト に基 づいてハッシュコード 値 を計 算 します。

© 2000–2025, MetaQuotes Ltd.

Equals

T型の2つの値を比較します。

```
bool Equals(
   T  x,      // 1番目の値
   T  y       // 2番目の値
   );
```

パラメータ

x

[in] 比較される1番目の値

y

[in] 比較される2番目の値

戻り値

値が等しい場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

HashCode

T型オブジェクトに基づいてハッシュコード値を計算します。

```
int HashCode(
   T  value     // 計算されるオブジェクト
   );
```

パラメータ

value

[in] ハッシュコードを取得したいオブジェクト

戻り値

ハッシュコードを返します。

© 2000–2025, MetaQuotes Ltd.

IList<T>

IList<T>はジェネリックデータリストを実装するためのインターフェイスです。

説明

IList<T>インターフェースは、インデックスによる要素へのアクセス、要素の検索や削除、ソートなどのリストを操作するための基本的なメソッドを定義します。

Declaration

```
template<typename T>
interface IList : public ICollection<T>
```

ヘッダ

```
#include <Generic\Interfaces\IList.mqh>
```

継承階層

ICollection

IList

直系子孫

CArrayList

クラスメソッド

メソッド	説明
TryGetValue	指定されたインデックスにあるリスト要素の値を設定します。
TrySetValue	指定されたインデックスにあるリスト要素の値を変更します。
Insert	リストに指定されたインデックスで要素を挿入します。
IndexOf	リストで値が最初に出現した位置を検索します。
LastIndexOf	リストで値が最後に出現した位置を検索します。
RemoveAt	指定されたインデックスにあるリスト要素を削除します。

© 2000-2025, MetaQuotes Ltd.

TryGetValue

指定されたインデックスにあるリスト要素の値を取得します。

```
bool TryGetValue(
   const int   index,      // 要素のインデックス
   T&          value       // 書き入れられる値
   );
```

パラメータ

index

[in] リスト要素のインデックス

&value

[out] リスト要素の指定された値が書き込まれる変数

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

TrySetValue

指定されたインデックスにあるリスト要素の値を変更します。

```
bool TrySetValue(
   const int   index,      // 要素インデックス
   T           value       // 新しい値
   );
```

パラメータ

index

[in] リスト要素のインデックス

value

[in] 指定されたリスト要素の新しい値

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

Insert

リストに指定されたインデックスで要素を挿入します。

```
bool Insert(
   const int   index,      // 挿入位置のインデックス
   T           item        // 挿入値
   );
```

パラメータ

index

[in] 挿入位置のインデックス

item

[in] 指定されたインデックスで挿入される値

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

IndexOf

リストで値の最初の出現を検索します。

```
int IndexOf(
   T  item     // 検索値
   );
```

パラメータ

item

[in] 検索値

戻り値

初めに見つかった要素のインデックスを返します。値が見つからなかった場合は-1を返します。

© 2000-2025, MetaQuotes Ltd.

LastIndexOf

リストで値の最後の出現を検索します。

```
int LastIndexOf(
   T  item     // 検索値
   );
```

パラメータ

item

[in] 検索値

戻り値

最後に見つかった要素のインデックスを返します。値が見つからなかった場合は-1を返します。

© 2000-2025, MetaQuotes Ltd.

RemoveAt

指定されたインデックスにあるリスト要素を削除します。

```
bool RemoveAt(
   const int  index     // 要素のインデックス
   );
```

パラメータ

index

[in] 削除する要素のインデックス

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

IMap<TKey, TValue>

IMap<TKey, TValue>はキー/値 ペアのジェネリックコレクションを実 装 するためのインターフェイスです。

説 明

IMap<TKey, TValue> interface defines basic methods to work with collections whose data are stored as key/value pairs.

Declaration

```
template<typename TKey, typename TValue>
interface IMap : public ICollection<TKey>
```

ヘッダ

```
#include <Generic\Interfaces\IMap.mqh>
```

継 承 階 層

ICollection

IMap

直 系 子 孫

CHashMap, CSortedMap

クラスメソッド

メソッド	説 明
Add	コレクションにキー/値 ペアを追 加します。
Contains	コレクションに指 定されたキーを持 つキー/値 のペアが含 まれているかどうかを判 断します。
Remove	コレクションから指 定されたキー/値 ペアの最 初 のオカレンスを削 除します。
TryGetValue	コレクションから指 定されたキーを持 つ要 素を取 得します。
TrySetValue	コレクションの指 定されたキーでのキー/値 ペアの値 を変 更します。
CopyTo	コレクションのすべてのキー/値 ペアを指 定された配 列 に、指 定されたインデックスから開 始してコピーします。

© 2000–2025, MetaQuotes Ltd.

Add

コレクションにキー/値 ペアを追 加します。

```
bool Add(
   TKey    key,       // キー
   TValue  value      // 値
   );
```

パラメータ

key

[in] キー

value

[in] 値。

戻り値

成 功 の場 合 はtrue、それ以 外 の場 合 はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

Contains

コレクションに指定されたキーを持つキー/値のペアが含まれているかどうかを判断します。

```
bool Contains(
   TKey    key,      // キー
   TValue  value     // 値
   );
```

パラメータ

key

[in] キー

value

[in] 値。

戻り値

Returnstrue, if the collection contains the key/value pair with the specified key and value, orfalse otherwise.

© 2000-2025, MetaQuotes Ltd.

Remove

コレクションから指定されたキー/値ペアの最初のオカレンスを削除します。

```
bool Remove(
   TKey  key      // キー
   );
```

パラメータ

key

[in] キー

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

TryGetValue

コレクションから指定されたキーを持つ要素を取得します。

```
bool TryGetValue(
   TKey      key,       // キー
   TValue&   value      // 値を書き入れるための変数
   );
```

パラメータ

key

[in] キー

&value

[out] 指定されたキー/値ペアの値が書き込まれる変数

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

TrySetValue

コレクションの指定されたキーでのキー/値ペアの値を変更します。

```
bool TrySetValue(
   TKey    key,       // キー
   TValue  value      // 新しい値
   );
```

パラメータ

key

[in] キー

value

[in] 指定されたキー/値ペアに割り当てる新しい値

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

CopyTo

コレクションのすべてのキー/値 ペアを指 定 された配 列 に、指 定 されたインデックスから開 始 してコピーします。

```
int CopyTo(
   TKey&       dst_keys[],       // キーを書き入れ対象の配列
   TValue&     dst_values[],     // 値を書き入れ対象の配列
   const int   dst_start=0       // 書き始めのインデックス
   );
```

パラメータ

&dst_keys[]

[out] コレクションのすべてのキーが書 き込 まれる配 列

&dst_values[]

[out] コレクションのキーに対 応 する値 が書 き込 まれる配 列

dst_start=0

[in] コピーを開 始 する配 列 のインデックス

戻 り値

複 製 されたキー/値 ペアの数 を返 します。

© 2000-2025, MetaQuotes Ltd.

ISet<T>

ISet<T>はジェネリックデータセットを実装するためのインターフェイスです。

説明

ISetインターフェイスは、集合の組合や交差、厳密なおよび非厳密な下位集合の定義などの集合を扱うための基本的なメソッドを定義します。

Declaration

```
template<typename T>
interface ISet : public ICollection<T>
```

ヘッダ

```
#include <Generic\Interfaces\ISet.mqh>
```

継承階層

ICollection

ISet

直系子孫

CHashSet, CSortedSet

クラスメソッド

メソッド	説明
ExceptWith	現在のコレクションと渡されたコレクション(配列)の差集合操作を生成します。
IntersectWith	現在のコレクションと渡されたコレクション(配列)の共通集合操作を生成します。
SymmetricExceptWith	現在の集合と指定されたコレクション(配列)の対称差集合操作を生成します。
UnionWith	現在の集合と指定されたコレクションまたは配列の和集合操作を生成します。
IsProperSubsetOf	現在の集合が指定されたコレクションまたは配列の真下位集合であるかどうかを判断します。
IsProperSupersetOf	現在の集合が指定されたコレクションまたは配列の真上位集合であるかどうかを判断します。
IsSubsetOf	現在の集合が指定されたコレクションまたは配列の下位集合であるかどうかを判断します。
IsSupersetOf	現在の集合が指定されたコレクションまたは配列の上位集合であるかどうかを判断します。

© 2000-2025, MetaQuotes Ltd.

メソッド	説明
Overlaps	現在の集合が指定されたコレクションまたは配列と重複するかどうかを判断します。
SetEquals	現在の集合が指定されたコレクションまたは配列のすべての要素を含んでいるかどうかを判断します。

© 2000–2025, MetaQuotes Ltd.

ExceptWith

現在のコレクションと渡されたコレクション(配列)の差集合操作を生成します。現在のコレクション(配列)から指定されたコレクション(配列)に存在するすべての要素を削除します。

ICollection<T>インターフェイスを実装するコレクションを扱うためのバージョン:

```
void ExceptWith(
   ICollection<T>*  collection     // コレクション
   );
```

配列を扱うためのバージョン:

```
void ExceptWith(
   T&  array[]                     // 配列
   );
```

パラメータ

collection

[in] 現在の集合との差が計算されるコレクション

&collection[]

[in] 現在の集合との差が計算される配列

注意事項

結果は現在のコレクション(配列)に書き込まれます。

© 2000-2025, MetaQuotes Ltd.

IntersectWith

現在のコレクションと渡されたコレクション(配列)の共通集合操作を生成します。現在のコレクションを変更して、指定されたコレクション(配列)に存在する要素のみを含むようにします。

ICollection<T>インターフェイスを実装するコレクションを扱うためのバージョン:

```
void IntersectWith(
   ICollection<T>*  collection     // コレクション
   );
```

配列を扱うためのバージョン:

```
void IntersectWith(
   T&  array[]                     // 配列
   );
```

パラメータ

**collection*

[in] 現在の集合が交差するコレクション

&collection[]

[in] 現在の集合が交差する配列

注意事項

結果は現在のコレクション(配列)に書き込まれます。

© 2000-2025, MetaQuotes Ltd.

SymmetricExceptWith

現在の集合と指定されたコレクション(配列)の対称差集合操作を生成します。現在のコレクションを変更して、ソースオブジェクトまたは指定されたコレクション(配列)の一方にのみ存在する要素のみを含むようにします。

ICollection<T>インターフェイスを実装するコレクションを扱うためのバージョン:

```
void SymmetricExceptWith(
   ICollection<T>*  collection     // コレクション
   );
```

配列を扱うためのバージョン:

```
void SymmetricExceptWith(
   T&  array[]                     // 配列
   );
```

パラメータ

collection

[in] 対称差を生成するコレクション

&collection[]

[in] 対称差を生成する配列

注意事項

結果は現在のコレクション(配列)に書き込まれます。

© 2000-2025, MetaQuotes Ltd.

UnionWith

現在の集合と指定されたコレクションまたは配列の和集合操作を生成します。現在のコレクション(配列)に、指定されたコレクション(配列)から不足している要素を追加します。

ICollection<T>インターフェイスを実装するコレクションを扱うためのバージョン:

```
void UnionWith(
   ICollection<T>*  collection     // コレクション
   );
```

配列を扱うためのバージョン:

```
void UnionWith(
   T&  array[]                     // 配列
   );
```

パラメータ

collection

[in] 現在の集合と和結合するコレクション

&collection[]

[in] 現在の集合と和結合する配列

注意事項

結果は現在のコレクション(配列)に書き込まれます。

© 2000-2025, MetaQuotes Ltd.

IsProperSubsetOf

現在の集合が指定されたコレクションまたは配列の真下位集合であるかどうかを判断します。

ICollection<T>インターフェイスを実装するコレクションを扱うためのバージョン：

```
bool IsProperSubsetOf(
   ICollection<T>*  collection     // 関係を判断するコレクション
   );
```

配列を扱うためのバージョン：

```
bool IsProperSubsetOf(
   T&  array[]                     // 関係を判断する配列
   );
```

パラメータ

collection

[in] 関係を判断するコレクション

&collection[]

[in] 関係を判断する配列

戻り値

現在の集合が真下位集合である場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

IsProperSupersetOf

現在の集合が指定されたコレクションまたは配列の真上位集合であるかどうかを判断します。

ICollection<T>インターフェイスを実装するコレクションを扱うためのバージョン：

```
bool IsProperSupersetOf(
   ICollection<T>*  collection     // 関係を判断するコレクション
   );
```

配列を扱うためのバージョン：

```
bool IsProperSupersetOf(
   T&  array[]                     // 関係を判断する配列
   );
```

パラメータ

collection

[in] 関係を判断するコレクション

&collection[]

[in] 関係を判断する配列

戻り値

現在の集合が真上位集合である場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

IsSubsetOf

現在の集合が指定されたコレクションまたは配列の下位集合であるかどうかを判断します。

ICollection<T>インターフェイスを実装するコレクションを扱うためのバージョン：

```
bool IsSubsetOf(
   ICollection<T>*  collection     // 関係を判断するコレクション
   );
```

配列を扱うためのバージョン：

```
bool IsSubsetOf(
   T&  array[]                     // 関係を判断する配列
   );
```

パラメータ

collection

[in] 関係を判断するコレクション

&collection[]

[in] 関係を判断する配列

戻り値

現在の集合が下位集合である場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

IsSupersetOf

現在の集合が指定されたコレクションまたは配列の上位集合であるかどうかを判断します。

ICollection<T>インターフェイスを実装するコレクションを扱うためのバージョン：

```
bool IsSupersetOf(
   ICollection<T>*  collection     // 関係を判断するコレクション
   );
```

配列を扱うためのバージョン：

```
bool IsSupersetOf(
   T&  array[]                     // 関係を判断する配列
   );
```

パラメータ

collection

[in] 関係を判断するコレクション

&collection[]

[in] 関係を判断する配列

戻り値

現在の集合が上位集合である場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

Overlaps

現在の集合が指定されたコレクションまたは配列と重複するかどうかを判断します。

ICollection<T>インターフェイスを実装するコレクションを扱うためのバージョン：

```
bool Overlaps(
   ICollection<T>*  collection     // 比較するコレクション
   );
```

配列を扱うためのバージョン：

```
bool Overlaps(
   T&  array[]                     // 比較する配列
   );
```

パラメータ

collection

[in] 重複を判断するコレクション

&collection[]

[in] 重複を判断する配列

戻り値

現在の集合とコレクションまたは配列が重なっている場合はtrue、それ例外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

SetEquals

現在の集合が指定されたコレクションまたは配列のすべての要素を含んでいるかどうかを判断します。

ICollection<T>インターフェイスを実装するコレクションを扱うためのバージョン：

```
bool SetEquals(
   ICollection<T>*  collection     // 比較するコレクション
   );
```

配列を扱うためのバージョン：

```
bool SetEquals(
   T&  array[]                     // 比較する配列
   );
```

パラメータ

collection

[in] 要素を比較するコレクション

&collection[]

[in] 要素を比較するコレクション

戻り値

現在の集合が指定されたコレクションまたは配列のすべての要素を含んでいる場合はtruep、それ例外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

CDefaultComparer<T>

CDefaultComparer<T>はCompareグローバルメソッドに基づいてIComparer<T>ジェネリックインターフェイスを実装するヘルパークラスです。

説明

The CDefaultComparer<T>クラスは、ユーザがIComparer<T>インタフェースを実装する別のクラスを暗黙的に使用していない限り、デフォルトでジェネリックデータコレクションで使用されます。

Declaration

```
template<typename T>
class CDefaultComparer : public IComparer<T>
```

ヘッダ

```
#include <Generic\Internal\DefaultComparer.mqh>
```

継承階層

IComparer

CDefaultComparer

クラスメソッド

メソッド	説明
Compare	T型の2つの値を比較します。

© 2000-2025, MetaQuotes Ltd.

Compare

T型の2つの値を比較します。

```
int Compare(
   T  x,      // 1番目の値
   T  y       // 2番目の値
   );
```

パラメータ

x

[in] 比較される1番目の値

y

[in] 比較される2番目の値

戻り値

比較された2つの値の比を表す数値を返します。

- 結果が0より小さい場合、xはyより小さい(x <y)です。
- 結果がゼロに等しい場合、xはyに等しい(x = y)です。
- 結果がゼロより大きい場合、xはyより大きい(x> y)です。

注意事項

xとyの値は、Tの型に応じてCompareグローバルメソッドのオーバーロードの1つに基づいて比較されます。

© 2000-2025, MetaQuotes Ltd.

CDefaultEqualityComparer<T>

CDefaultEqualityComparer<T>はEquals<T>およびGetHashCodeグローバルメソッドを使用してIEqualityComparer<T>ジェネリックインターフェイスを実装するヘルパークラスです。

説明

The CDefaultEqualityComparer<T>クラスは、ユーザがIEqualityComparer<T>インタフェースを実装する別のクラスを暗黙的に使用していない限り、デフォルトでジェネリックデータコレクションで使用されます。

Declaration

```
template<typename T>
class CDefaultEqualityComparer : public IEqualityComparer<T>
```

ヘッダ

```
#include <Generic\Internal\DefaultEqualityComparer.mqh>
```

継承階層

IEqualityComparer

CDefaultEqualityComparer

クラスメソッド

メソッド	説明
Equals	T型の2つの値を比較します。
HashCode	T型オブジェクトに基づいてハッシュコード値を計算します。

© 2000–2025, MetaQuotes Ltd.

Equals

T型の2つの値を比較します。

```
bool Equals(
   T  x,      // 1番目の値
   T  y       // 2番目の値
   );
```

パラメータ

x

[in] 比較される1番目の値

y

[in] 比較される2番目の値

戻り値

値が等しい場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

HashCode

T型オブジェクトに基づいてハッシュコード値を計算します。

```
int HashCode(
   T  value     // 計算されるオブジェクト
   );
```

パラメータ

value

[in] ハッシュコードを取得したいオブジェクト

戻り値

ハッシュコードを返します。

© 2000-2025, MetaQuotes Ltd.

CRedBlackTreeNode<T>

CRedBlackTreeNode<T>はCRedBlackTree<T>クラスを実装するためのヘルパークラスです。

説明

CRedBlackTreeNode<T>クラスはCRedBlackTree<T>のノードです。ツリーナビゲーションメソッドは、このクラスで実装されています。

Declaration

```
template<typename T>
class CRedBlackTreeNode
```

ヘッダ

```
#include <Generic\RedBlackTree.mqh>
```

クラスメソッド

メソッド	説明
値	ノード値を返します/設定します。
Parent	親ノードへのポインタを返します/設定します。
Left	左ノードへのポインタを返します/設定します。
Right	右ノードへのポインタを返します/設定します。
Color	ノードの色を返します/設定します。
IsLeaf	ノードが葉であるかどうかを判断します。
CreateEmptyNode	親や子を持たない新しい黒ノードを作成し、そのポインタを返します。

© 2000-2025, MetaQuotes Ltd.

Value（取得メソッド）

ノード値を返します。

```
T Value();
```

戻り値

ノード値を返します。

Value（設定メソッド）

ノード値を設定します。

```
void Value(
   T  value     // ノード値
   );
```

パラメータ

value

[in] ノード値

© 2000–2025, MetaQuotes Ltd.

Parent（取得メソッド）

親ノードへのポインタを返します。

```
CRedBlackTreeNode<T>* Parent();
```

戻り値

親ノードへのポインタを返します。

Parent（設定メソッド）

親ノードへのポインタを設定します。

```
void Parent(
   CRedBlackTreeNode<T>*  node      // 親ノードへのポインタ
   );
```

パラメータ

node

[in] 親ノードへのポインタ

© 2000-2025, MetaQuotes Ltd.

Left（取得メソッド）

左ノードへのポインタを返します。

```
CRedBlackTreeNode<T>* Left();
```

戻り値

左ノードへのポインタを返します。

Left（設定メソッド）

左ノードへのポインタを設定します。

```
void Left(
   CRedBlackTreeNode<T>*  node      // 左ノードへのポインタ
   );
```

パラメータ

node

[in] 左ノードへのポインタ

© 2000–2025, MetaQuotes Ltd.

Right（取得メソッド）

右ノードへのポインタを返します。

```
CRedBlackTreeNode<T>* Right();
```

戻り値

右ノードへのポインタを返します。

Right（設定メソッド）

右ノードへのポインタを設定します。

```
void Right(
   CRedBlackTreeNode<T>*  node      // 右ノードへのポインタ
   );
```

パラメータ

node

[in] 右ノードへのポインタ

© 2000–2025, MetaQuotes Ltd.

Color（Get メソッド）

ノードの色を返します。

```
ENUM_RED_BLACK_TREE_NODE_TYPE Color();
```

戻り値

ノードの色を返します。

Color（Set メソッド）

ノードの色を設定します。

```
void Color(
   ENUM_RED_BLACK_TREE_NODE_TYPE  clr     // ノードの色
   );
```

パラメータ

clr

[in] ノードの色

注意事項

ノードの色はENUM_RED_BLACK_TREE_NODE_TYPEの値を使って設定されます。これには2つあります。

- RED_BLACK_TREE_NODE_RED — 赤ノード
- RED_BLACK_TREE_NODE_BLACK — 黒ノード

© 2000–2025, MetaQuotes Ltd.

IsLeaf

ノードが葉であるかどうかを判断します。

```
bool IsLeaf();
```

戻り値

ノードが葉である場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

CreateEmptyNode

親や子を持たない新しい黒ノードを作成し、そのポインタを返します。

```
static CRedBlackTreeNode<T>* CreateEmptyNode();
```

戻り値

新しいノードへのポインタを返します。

© 2000–2025, MetaQuotes Ltd.

CLinkedListNode<T>

CLinkedListNode<T>はCLinkedListNode<T>クラスを実装するためのヘルパークラスです。

説明

CLinkedListNode<T>クラスは、二重連結リストのノードCLinkedListNode<T>です。リストナビゲーションメソッドは、このクラスで実装されています。

Declaration

```
template<typename T>
class CLinkedListNode
```

ヘッダ

```
#include <Generic\LinkedList.mqh>
```

クラスメソッド

メソッド	説明
List	CLinkedList<T>へのポインタを返します/設定します。
Next	次のノードへのポインタを返します/設定します。
Previous	前のノードへのポインタを返します/設定します。
値	ノード値を返します/設定します。

© 2000-2025, MetaQuotes Ltd.

List（取得メソッド）

CLinkedList<T>へのポインタを返します。

```
CLinkedList<T>* List();
```

戻り値

CLinkedList<T>連結リストへのポインタを返します。

List（設定メソッド）

CLinkedList<T>へのポインタを設定します。

```
void List(
   CLinkedList<T>*   value      // リストへのポインタ
   );
```

パラメータ

value

[in] CLinkedList<T>連結リストへのポインタ

© 2000-2025, MetaQuotes Ltd.

Next（取得メソッド）

次のノードへのポインタを返します。

```
CLinkedListNode<T>* Next();
```

戻り値

次のノードへのポインタを返します。

Next（設定メソッド）

次のノードへのポインタを/設定します。

```
void Next(
   CLinkedListNode<T>*  value     // 次のノードへのポインタ
   );
```

パラメータ

value

[in] 次のノードへのポインタ

© 2000-2025, MetaQuotes Ltd.

Previous（取得メソッド）

前のノードへのポインタを返します。

```
CLinkedListNode<T>* Previous();
```

戻り値

前のノードへのポインタを返します。

Previous（設定メソッド）

前のノードへのポインタを設定します。

```
void Previous(
   CLinkedListNode<T>*  value     // 前のノードへのポインタ
   );
```

パラメータ

value

[in] 前のノードへのポインタ

© 2000-2025, MetaQuotes Ltd.

Value（取得メソッド）

ノード値を返します。

```
T Value();
```

戻り値

ノード値を返します。

Value（設定メソッド）

ノード値を設定します。

```
void Value(
   T  value     // ノード値
   );
```

パラメータ

value

[in] ノード値

© 2000–2025, MetaQuotes Ltd.

CKeyValuePair<TKey, TValue>

CKeyValuePair<TKey, TValue>クラスはキー/値 ペアを実 装します。

説 明

CKeyValuePair<TKey, TValue>クラスは、キーとキー/値 のペアの値 を操 作 するためのメソッドを実 装します。

Declaration

```
template<typename TKey, typename TValue>
class CKeyValuePair : public IComparable<CKeyValuePair<TKey,TValue>*>
```

ヘッダ

```
#include <Generic\HashMap.mqh>
```

継 承 階 層

IEqualityComparable
  IComparable
    CKeyValuePair

クラスメソッド

メソッド	説 明
Key	キー/値 ペアのキーを取 得 /設 定します。
値	キー/値 ペアの値 を取 得 /設 定します。
Clone	キー/値 ペアが現 在 のものと等しい新しいキー/値 ペアを作 成します。
Compare	現 在 のキー/値 ペアを指 定されたものと比 較します。
Equals	現 在 のキー/値 ペアが指 定されたものと等しいかどうかをチェックします。
HashCode	キー/値 ペアに基 づいてハッシュコード 値 を計 算します。

© 2000–2025, MetaQuotes Ltd.

Key（取得メソッド）

キー/値 ペアのキーを取得します。

```
TKey Key();
```

戻り値

キーを返します。

Key（設定メソッド）

キー/値 ペアのキーを設定します。

```
void Key(
   TKey  key     // キー
   );
```

パラメータ

key

[in] キー

© 2000–2025, MetaQuotes Ltd.

Value（取得メソッド）

キー/値 ペアの値を取得します。

```
TValue Value();
```

戻り値

値を返します。

Value（設定メソッド）

キー/値 ペアの値を/設定します。

```
void Value(
   TValue  value     // 値
   );
```

パラメータ

value

[in] 値

© 2000–2025, MetaQuotes Ltd.

Clone

キー/値 ペアが現 在 のものと等 しい新 しいキー/値 ペアを作 成 します。

```
TValue>* Clone();
```

戻 り値

新 しいキー/値 ペアを返 します。

© 2000–2025, MetaQuotes Ltd.

Compare

現在のキー/値ペアを指定されたものと比較します。

```
int Compare(
   CKeyValuePair<TKeyTValue>*  pair     // 比較するペア
   );
```

パラメータ

pair

[in] 比較されるペア

戻り値

現在のキーと値のペアの比率を表す数値を返します。

- 結果が0より小さい場合、ペアは渡されたものより小さいです。
- 結果が0より小さい場合、ペアは渡されたものと等しいです。
- 結果がゼロより大きい場合、ペアは渡されたものより大きいです。

注意事項

キー/値ペアはキーに基づいて比較されます。

© 2000-2025, MetaQuotes Ltd.

Equals

現在のキー/値ペアが指定されたものと等しいかどうかをチェックします。

```
bool Equals(
   CKeyValuePair<TKeyTValue>*  pair     // 比較するペア
   );
```

パラメータ

pair

[in] 比較するペア

戻り値

キー/値ペアが等しい場合はtrue、それ以外の場合はfalseを返します。

注意事項

キー/値ペアはキーに基づいて比較されます。

© 2000-2025, MetaQuotes Ltd.

HashCode

キー/値 ペアに基 づいてハッシュコード 値 を計 算します。

```
int HashCode();
```

戻り値

ハッシュコードを返します。

注 意 事 項

キー/値 ペアのハッシュコード はキーのハッシュコードと同じです。

© 2000-2025, MetaQuotes Ltd.

CArrayList<T>

CArrayList<T>はIList<T>インターフェイスを実装するジェネリッククラスです。

説明

CArrayList<T>クラスは、T型の動的データリストの実装です。このクラスは、インデックスで要素にアクセスする、要素を検索および削除する、ソートするなどのリストを操作するための基本的なメソッドを提供します。

Declaration

```
template<typename T>
class CArrayList : public IList<T>
```

ヘッダ

```
#include <Generic\ArrayList.mqh>
```

継承階層

ICollection
  IList
    CArrayList

クラスメソッド

メソッド	説明
Capacity	リストの現在の容量を取得および設定します。
kaCount	リストの要素数を返します。
Contains	リストに指定された値を持つ要素が含まれているかどうかを判断します。
TrimExcess	リストの容量を実際の要素数に設定します。
TryGetValue	リストの指定されたインデックスにある要素を取得します。
TrySetValue	指定されたインデックスにあるリスト要素の値を設定します。
Add	リストに要素を追加します。
AddRange	コレクションまたは要素の配列をリストに追加します。
Insert	リストに指定されたインデックスで要素を挿入します。
InsertRange	リストに指定されたインデックスで要素のコレクションまたは配列を挿入します。
CopyTo	リストのすべての要素を、指定された配列に指定されたインデックスから開始してコピーします。
BinarySearch	指定された値を昇順ソートリストで検索します。
IndexOf	リストで値が最初に出現した位置を検索します。

© 2000–2025, MetaQuotes Ltd.

メソッド	説明
LastIndexOf	リストで値が最後に出現した位置を検索します。
Clear	コレクションからすべての要素を削除します。
Remove	リストから指定された要素の最初のオカレンスを削除します。
RemoveAt	リストの指定されたインデックスから要素を削除します。
RemoveRange	リストから要素の範囲を削除します。
リバース。	リストの要素の順序を逆にします。
Sort	リスト内の要素をソートします。

© 2000–2025, MetaQuotes Ltd.

Capacity（取得メソッド）

現在のリスト容量を返します。

```
int Capacity();
```

戻り値

現在のリスト容量を返します。

Capacity（設定メソッド）

現在のリスト容量を設定します。

```
void Capacity(
   const int  capacity     // 容量値
   );
```

パラメータ

capacity

[in] 新しい容量値

© 2000-2025, MetaQuotes Ltd.

kaCount

リストの要素数を返します。

```
int Count();
```

戻り値

要素数を返します。

© 2000–2025, MetaQuotes Ltd.

Contains

リストに指定された値を持つ要素が含まれているかどうかを判断します。

```
bool Contains(
   T  item     // 検索値
   );
```

パラメータ

item

[in] 検索値

戻り値

指定された値を持つ要素がリストにある場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

TrimExcess

リストの容量を実際の要素数に設定して、未使用メモリを解放します。

```
void TrimExcess();
```

© 2000–2025, MetaQuotes Ltd.

TryGetValue

リストの指定されたインデックスにある要素を取得します。

```
bool TryGetValue(
   const int   index,      // インデックス
   T&          value       // 書く変数
   );
```

パラメータ

index

[in] 取得したい値のリスト要素のインデックス

&value

[out] 要素値を書き込む変数

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

TrySetValue

指定されたインデックスにあるリスト要素の値を設定します。

```
bool TrySetValue(
   const int   index,      // インデックス
   T           value       // 要素値
   );
```

パラメータ

index

[in] 設定したい値のリスト要素のインデックス

value

[in] リスト要素の値を設定します

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

Add

リストに要素を追加します。

```
bool Add(
   T  value     // 要素値
   );
```

パラメータ

value

[in] 追加する要素の値

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

AddRange

コレクションまたは要素の配列をリストに追加します。

配列バージョン:

```
bool AddRange(
   const T&  array[]                // 追加される配列
   );
```

コレクションバージョン:

```
bool AddRange(
   ICollection<T>*  collection      // 追加されるコレクション
   );
```

パラメータ

&array[]

[in] 追加される配列

**collection*

[in] 追加されるコレクション

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

Insert

リストに指定されたインデックスで要素を挿入します。

```
bool Insert(
   const int   index,      // 挿入位置のインデックス
   T           item        // 挿入値
   );
```

パラメータ

index

[in] 挿入位置のインデックス

item

[in] 指定されたインデックスで挿入される値

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

InsertRange

リストに指定されたインデックスで要素のコレクションまたは配列を挿入します。

配列を挿入するバージョン:

```
bool InsertRange(
   const int  index,                // 挿入位置のインデックス
   const T&   array[]               // 挿入される配列
   );
```

コレクションを挿入するバージョン:

```
bool InsertRange(
   const int         index,         // 挿入位置のインデックス
   ICollection<T>*   collection     // 挿入されるコレクション
   );
```

パラメータ

index

[in] 挿入位置のインデックス

&array[]

[in] 指定されたインデックスで挿入される配列

**collection*

[in] 指定されたインデックスで挿入されるコレクション

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

CopyTo

リストのすべての要素を、指定された配列に指定されたインデックスから開始してコピーします。

```
int CopyTo(
   T&          dst_array[],      // 書き入れ対象の配列
   const int   dst_start=0       // 書き始めのインデックス
   );
```

パラメータ

&dst_array[]

[out] リストの要素が書き込まれる配列

dst_start=0

[in] コピーを開始する配列のインデックス

戻り値

複製された要素の数を返します。

© 2000-2025, MetaQuotes Ltd.

BinarySearch

指定された値を昇順ソートリストで検索します。

IComparable<T>インターフェイスを実装するクラスを使用して要素を比較して、指定された値の範囲で検索するバージョン:

```
int BinarySearch(
   const int       index,        // 開始インデックス
   const int       count,        // 検索範囲
   T               item,         // 検索値
   IComparer<T>*   comparer      // 比較インターフェイス
   );
```

IComparable<T>インターフェイスを実装するクラスを使用して要素を比較して検索するバージョン:

```
int BinarySearch(
   T               item,         // 検索値
   IComparer<T>*   comparer      // 比較インターフェイス
   );
```

::Compareグローバルメソッドを使用して要素を比較するバージョン:

```
int BinarySearch(
   T  item                       // 検索値
   );
```

パラメータ

index

[in] 検索開始インデックス

count

[in] 検索範囲の長さ

item

[in] 検索値

**comparer*

[in] 要素比較インターフェイス

戻り値

見つかった要素のインデックスを返します。検索値が見つからない場合は、値が最も小さい最小の要素のインデックスを返します。

© 2000–2025, MetaQuotes Ltd.

IndexOf

リストで値の最初の出現を検索します。

リスト全体を検索するバージョン:

```
int IndexOf(
   T  item                    // 検索値
   );
```

指定された位置からリストの終わりまでを検索するバージョン:

```
int IndexOf(
   T           item,          // 検索値
   const int   start_index    // 開始インデックス
   );
```

指定された範囲内の指定された位置から検索するバージョン:

```
int IndexOf(
   T           item,          // 検索値
   const int   start_index,   // 開始インデックス
   const int   count          // 検索範囲
   );
```

パラメータ

item

[in] 検索値

start_index

[in] 検索開始インデックス

count

[in] 検索範囲の長さ

戻り値

初めに見つかった要素のインデックスを返します。値が見つからなかった場合は-1を返します。

© 2000–2025, MetaQuotes Ltd.

LastIndexOf

リストで値の最後の出現を検索します。

リスト全体を検索するバージョン:

```
int LastIndexOf(
   T  item                     // 検索値
   );
```

指定された位置からリストの終わりまでを検索するバージョン:

```
int LastIndexOf(
   T           item,           // 検索値
   const int   start_index     // 開始インデックス
   );
```

指定された範囲内の指定された位置から検索するバージョン:

```
int LastIndexOf(
   T           item,           // 検索値
   const int   start_index,    // 開始インデックス
   const int   count           // 検索範囲
   );
```

パラメータ

item

[in] 検索値

start_index

[in] 検索開始インデックス

count

[in] 検索範囲の長さ

戻り値

最後に見つかった要素のインデックスを返します。値が見つからなかった場合は-1を返します。

© 2000–2025, MetaQuotes Ltd.

Clear

コレクションからすべての要素を削除します。

```
void Clear();
```

© 2000–2025, MetaQuotes Ltd.

Remove

リストから指定された要素の最初のオカレンスを削除します。

```
bool Remove(
   T  item     // 要素値
   );
```

パラメータ

item

[in] 削除する要素の値

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

RemoveAt

リストの指定されたインデックスから要素を削除します。

```
bool RemoveAt(
   const int  index     // インデックス
   );
```

パラメータ

index

[in] 削除される曲線のインデックス

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

RemoveRange

リストから要素の範囲を削除します。

```
bool RemoveRange(
   const int  start_index,      // 開始インデックス
   const int  count             // 要素数
   );
```

パラメータ

start_index

[in] 削除開始インデックス

count

[in] 削除される要素の数

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

リバース。

リストの要素の順序を逆にします。

リスト全体を操作するバージョン:

```
bool Reverse();
```

指定された範囲のリスト要素を処理するバージョン:

```
bool Reverse(
   const int  start_index,     // 開始インデックス
   const int  count            // 要素数
   );
```

パラメータ

start_index

[in] 開始インデックス

count

[in] 操作に関与する配列要素の数

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

Sort

リスト内の要素をソートします。

リスト内のすべての要素をソートするバージョン:

```
bool Sort();
```

要素を比較にIComparable<T>インターフェイスを実装するクラスを使用して、リスト内のすべての要素をソートするバージョン:

```
bool Sort(
   IComparer<T>*  comparer          // 比較インターフェイス
   );
```

要素を比較にIComparable<T>インターフェイスを実装するクラスを使用して、リスト内の指定された範囲の要素をソートするバージョン:

```
bool Sort(
   const int  start_index,      // 開始インデックス
   const int  count             // 要素数
   IComparer<T>*  comparer      // 比較インターフェイス
   );
```

パラメータ

comparer
[in] 要素比較インターフェイス

start_index
[in] ソート開始インデックス

count
[in] ソート範囲の長さ

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

CHashMap<TKey, TValue>

CHashMap<TKey, TValue>はIMap<TKey, TValue>インターフェイスを実装するジェネリッククラスです。

説明

CHashMap<TKey, TValue>クラスは、データがソートされず、キーの一意性要件を考慮してキー/値ペアのとして格納される動的ハッシュテーブルの実装です。このクラスは、キーで値にアクセスし、キー/値ペアを検索して削除するなどのハッシュテーブルを操作するための基本的なメソッドを提供します。

Declaration

```
template<typename TKey, typename TValue>
class CHashMap : public IMap<TKey, TValue>
```

ヘッダ

```
#include <Generic\HashMap.mqh>
```

継承階層

ICollection
  IMap
    CHashMap

クラスメソッド

メソッド	説明
Add	ハッシュテーブルにキー/値ペアを追加します。
kaCount	ハッシュテーブルの要素数を返します。
Comparer	ハッシュテーブルを整理するために使用されるIEqualityComparer<T>インターフェイスへのポインタを返します。
Contains	ハッシュテーブルに指定されたキー/値ペアが含まれているかどうかを判断します。
ContainsKey	ハッシュテーブルに指定されたキーを持つキー/値のペアが含まれているかどうかを判断します。
ContainsValue	CHashMap<TKey, TValue>はIMap<TKey, TValue>インターフェイスを実装するジェネリッククラスです。
CopyTo	ハッシュテーブルのすべてのキー/値ペアを指定された配列に、指定されたインデックスから開始してコピーします。
Clear	ハッシュテーブルからすべての要素を削除します。
Remove	ハッシュテーブルから指定されたキー/値ペアの最初のオカレンスを削除します。
TryGetValue	ハッシュテーブルから指定されたキーを持つ要素を取得します。

© 2000-2025, MetaQuotes Ltd.

メソッド	説 明
TrySetValue	ハッシュテーブルの指 定されたキーでのキー/値 ペアの値 を変 更します。

© 2000–2025, MetaQuotes Ltd.

Add

ハッシュテーブルにキー/値 ペアを追 加します。

生 成 されたキー/値 ペアを追 加 するバージョン:

```
bool Add(
   CKeyValuePair<TKeyTValue>*  pair     // キー/値ペア
   );
```

指 定 されたキーと値 でキー/値 ペアを追 加 するバージョン:

```
bool Add(
   TKey    key,                         // キー
   TValue  value                        // 値
   );
```

パラメータ

pair
[in] キー/値 ペア

key
[in] キー

value
[in] 値 。

戻 り値

成 功 の場 合 はtrue、それ以 外 の場 合 はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

kaCount

ハッシュテーブルの要素数を返します。

```
int Count();
```

© 2000–2025, MetaQuotes Ltd.

Comparer

ハッシュテーブルを整理するために使用されるIEqualityComparer<T>インターフェイスへのポインタを返します。

```
IEqualityComparer<TKey>* Comparer() const;
```

戻り値

IEqualityComparer<T>インターフェイスへのポインタを返します。

© 2000-2025, MetaQuotes Ltd.

Contains

ハッシュテーブルに指定されたキー/値ペアが含まれているかどうかを判断します。

生成されたキー/値ペアを処理するバージョン:

```
bool Contains(
   CKeyValuePair<TKeyTValue>*  item     // キー/値ペア
   );
```

個別に設定されたキーと値の形式でキー/値ペアを処理するバージョン.

```
bool Contains(
   TKey   key,                          // キー
   TValue value                         // 値
   );
```

パラメータ

item

[in] キー/値ペア

key

[in] キー

value

[in] 値。

戻り値

ハッシュテーブルに指定されたキーを持つキー/値のペアが含まれている場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

ContainsKey

ハッシュテーブルに指定されたキーを持つキー/値のペアが含まれているかどうかを判断します。

```
bool ContainsKey(
   TKey  key     // キー
   );
```

パラメータ

key

[in] キー

戻り値

ハッシュテーブルに指定されたキーを持つキー/値のペアが含まれている場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

ContainsValue

ハッシュテーブルに指定された値持つキー/値のペアが含まれているかどうかを判断します。

```
bool ContainsValue(
   TValue  value     // 値
   );
```

パラメータ

value

[in] 値。

戻り値

ハッシュテーブルに指定された値を持つキー/値のペアが含まれている場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

CopyTo

ハッシュテーブルのすべてのキー/値 ペアを 指 定 された配 列 に、指 定 されたインデックスから開 始 してコピーします。

ハッシュテーブルをキー/値 ペアの配 列 にコピーするバージョン:

```
int CopyTo(
   CKeyValuePair<TKeyTValue>*&  dst_array[],     // キー/値ペアを書き入れる配列
   const int                    dst_start=0      // 書き始めのインデックス
   );
```

ハッシュテーブルを別 々 のキー配 列 と値 配 列 にコピーするバージョン:

```
int CopyTo(
   TKey&       dst_keys[],                       // キーを書き入れ対象の配列
   TValue&     dst_values[],                     // 値を書き入れ対象の配列
   const int   dst_start=0                       // 書き始めのインデックス
   );
```

パラメータ

*&dst_array[]

[out] ハッシュテーブルのすべてのペアが書 き込 まれる配 列

&dst_keys[]

[out] ハッシュテーブルのすべてのキーが書 き込 まれる配 列

&dst_values[]

[out] ハッシュテーブルのすべての値 が書 き込 まれる配 列

dst_start=0

[in] コピー開 始 インデックス

戻 り値

複 製 されたキー/値 ペアの数 を返します。

© 2000–2025, MetaQuotes Ltd.

Clear

ハッシュテーブルからすべての要素を削除します。

```
void Clear();
```

© 2000–2025, MetaQuotes Ltd.

Remove

ハッシュテーブルから指定されたキー/値ペアの最初のオカレンスを削除します。

生成されたキー/値ペアを削除するバージョン:

```
bool Remove(
   CKeyValuePair<TKeyTValue>*  item     // キー/値ペア
   );
```

キーに基づいてキー/値ペアを削除するバージョン:

```
bool Remove(
   TKey  key                            // キー
   );
```

パラメータ

item

[in] キー/値ペア

key

[in] キー

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

TryGetValue

ハッシュテーブルから指定されたキーを持つ要素を取得します。

```
bool TryGetValue(
   TKey      key,       // キー
   TValue&   value      // 値を書き入れるための変数
   );
```

パラメータ

key

[in] キー

&value

[out] 指定されたキー/値ペアの値が書き込まれる変数

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

TrySetValue

ハッシュテーブルの指定されたキーでのキー/値ペアの値を変更します。

```
bool TrySetValue(
   TKey    key,       // キー
   TValue  value      // 新しい値
   );
```

パラメータ

key

[in] キー

value

[in] 指定されたキー/値ペアに割り当てる新しい値

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

CHashSet<T>

CHashSet<T>はISet<T>インターフェイスを実装するジェネリッククラスです。

説明

CHashSet<T>クラスは、各値が一意な、T型の順序付けられていない動的データ集合の実装です。このクラスは、集合の操作とその関連操作（和集合と共通部分、下位集合や真下位集合の定義など）のための基本的なメソッドを提供します。

Declaration

```
template<typename T>
class CHashSet : public ISet<T>
```

ヘッダ

```
#include <Generic\HashSet.mqh>
```

継承階層

ICollection
ISet
CHashSet

クラスメソッド

メソッド	説明
Add	集合に要素を追加します。
kaCount	集合の要素数を返します。
Comparer	集合に指定された値を持つ要素が含まれているかどうかを判断します。
Contains	集合を整理するために使用されるIEqualityComparer<T>インターフェイスへのポインタを返します。
TrimExcess	集合の容量を実際の要素数に設定して、未使用メモリを解放します。
CopyTo	集合のすべての要素を、指定された配列に指定されたインデックスから開始してコピーします。
Clear	集合からすべての要素を削除します。
Remove	集合から指定された要素を削除します。
ExceptWith	現在のコレクションと渡されたコレクション（配列）の差集合操作を生成します。
IntersectWith	現在のコレクションと渡されたコレクション（配列）の共通集合操作を生成します。
SymmetricExceptWith	現在の集合と指定されたコレクション（配列）の対称差集合操作を生成します。

© 2000–2025, MetaQuotes Ltd.

メソッド	説明
UnionWith	現在の集合と指定されたコレクションまたは配列の和集合操作を生成します。
IsProperSubsetOf	現在の集合が指定されたコレクションまたは配列の真下位集合であるかどうかを判断します。
IsProperSupersetOf	現在の集合が指定されたコレクションまたは配列の真上位集合であるかどうかを判断します。
IsSubsetOf	現在の集合が指定されたコレクションまたは配列の下位集合であるかどうかを判断します。
IsSupersetOf	現在の集合が指定されたコレクションまたは配列の上位集合であるかどうかを判断します。
Overlaps	現在の集合が指定されたコレクションまたは配列と重複するかどうかを判断します。
SetEquals	現在の集合が指定されたコレクションまたは配列のすべての要素を含んでいるかどうかを判断します。

© 2000–2025, MetaQuotes Ltd.

Add

集合に要素を追加します。

```
bool Add(
   T  value     // 要素値
   );
```

パラメータ

value

[in] 追加する要素の値

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

kaCount

集合の要素数を返します。

```
int Count();
```

戻り値

要素数を返します。

© 2000–2025, MetaQuotes Ltd.

Contains

集合に指定された値を持つ要素が含まれているかどうかを判断します。

```
bool Contains(
   T  item     // 検索値
   );
```

パラメータ

item

[in] 検索値

戻り値

指定された値を持つ要素が集合にある場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

Comparer

集合を整理するために使用されるIEqualityComparer<T>インターフェイスへのポインタを返します。

```
IEqualityComparer<T>* Comparer() const;
```

戻り値

IEqualityComparer<T>インターフェイスへのポインタを返します。

© 2000-2025, MetaQuotes Ltd.

TrimExcess

集合の容量を実際の要素数に設定して、未使用メモリを解放します。

```
void TrimExcess();
```

© 2000–2025, MetaQuotes Ltd.

CopyTo

集合のすべての要素を、指定された配列に指定されたインデックスから開始してコピーします。

```
int CopyTo(
   T&          dst_array[],      // 書き入れ対象の配列
   const int   dst_start=0       // 書き始めのインデックス
   );
```

パラメータ

&dst_array[]

[out] 集合の要素が書き込まれる配列

dst_start=0

[in] コピーを開始する配列のインデックス

戻り値

複製された要素の数を返します。

© 2000-2025, MetaQuotes Ltd.

Clear

集合からすべての要素を削除します。

```
void Clear();
```

© 2000–2025, MetaQuotes Ltd.

Remove

集合から指定された要素を削除します。

```
bool Remove(
   T  item     // 要素値
   );
```

パラメータ

item

[in] 削除する要素の値

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

ExceptWith

現在のコレクションと渡されたコレクション(配列)の差集合操作を生成します。現在のコレクション(配列)から指定されたコレクション(配列)に存在するすべての要素を削除します。

ICollection<T>インターフェイスを実装するコレクションを扱うためのバージョン:

```
void ExceptWith(
   ICollection<T>*  collection     // コレクション
   );
```

配列を扱うためのバージョン:

```
void ExceptWith(
   T&  array[]                     // 配列
   );
```

パラメータ

collection

[in] 現在の集合との差が計算されるコレクション

&collection[]

[in] 現在の集合との差が計算される配列

注意事項

結果は現在のコレクション(配列)に書き込まれます。

© 2000-2025, MetaQuotes Ltd.

IntersectWith

現在のコレクションと渡されたコレクション(配列)の共通集合操作を生成します。現在のコレクションを変更して、指定されたコレクション(配列)に存在する要素のみを含むようにします。

ICollection<T>インターフェイスを実装するコレクションを扱うためのバージョン:

```
void IntersectWith(
   ICollection<T>*  collection     // コレクション
   );
```

配列を扱うためのバージョン:

```
void IntersectWith(
   T&  array[]                     // 配列
   );
```

パラメータ

collection

[in] 現在の集合が交差するコレクション

&collection[]

[in] 現在の集合が交差する配列

注意事項

結果は現在のコレクション(配列)に書き込まれます。

© 2000-2025, MetaQuotes Ltd.

SymmetricExceptWith

現在の集合と指定されたコレクション(配列)の対称差集合操作を生成します。現在のコレクションを変更して、ソースオブジェクトまたは指定されたコレクション(配列)の一方にのみ存在する要素のみを含むようにします。

ICollection<T>インターフェイスを実装するコレクションを扱うためのバージョン:

```
void SymmetricExceptWith(
   ICollection<T>*  collection     // コレクション
   );
```

配列を扱うためのバージョン:

```
void SymmetricExceptWith(
   T&  array[]                     // 配列
   );
```

パラメータ

collection

[in] 対称差を生成するコレクション

&collection[]

[in] 対称差を生成する配列

注意事項

結果は現在のコレクション(配列)に書き込まれます。

© 2000-2025, MetaQuotes Ltd.

UnionWith

現在の集合と指定されたコレクションまたは配列の和集合操作を生成します。現在のコレクション(配列)に、指定されたコレクション(配列)から不足している要素を追加します。

ICollection<T>インターフェイスを実装するコレクションを扱うためのバージョン:

```
void UnionWith(
   ICollection<T>*  collection     // コレクション
   );
```

配列を扱うためのバージョン:

```
void UnionWith(
   T&  array[]                     // 配列
   );
```

パラメータ

collection

[in] 現在の集合と和結合するコレクション

&collection[]

[in] 現在の集合と和結合する配列

注意事項

結果は現在のコレクション(配列)に書き込まれます。

© 2000-2025, MetaQuotes Ltd.

IsProperSubsetOf

現在の集合が指定されたコレクションまたは配列の真下位集合であるかどうかを判断します。

ICollection<T>インターフェイスを実装するコレクションを扱うためのバージョン：

```
bool IsProperSubsetOf(
   ICollection<T>*  collection     // 関係を判断するコレクション
   );
```

配列を扱うためのバージョン：

```
bool IsProperSubsetOf(
   T&  array[]                     // 関係を判断する配列
   );
```

パラメータ

collection

[in] 関係を判断するコレクション

&collection[]

[in] 関係を判断する配列

戻り値

現在の集合が真下位集合である場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

IsProperSupersetOf

現在の集合が指定されたコレクションまたは配列の真上位集合であるかどうかを判断します。

ICollection<T>インターフェイスを実装するコレクションを扱うためのバージョン：

```
bool IsProperSupersetOf(
   ICollection<T>*  collection     // 関係を判断するコレクション
   );
```

配列を扱うためのバージョン：

```
bool IsProperSupersetOf(
   T&  array[]                     // 関係を判断する配列
   );
```

パラメータ

collection

[in] 関係を判断するコレクション

&collection[]

[in] 関係を判断する配列

戻り値

現在の集合が真上位集合である場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

IsSubsetOf

現在の集合が指定されたコレクションまたは配列の下位集合であるかどうかを判断します。

ICollection<T>インターフェイスを実装するコレクションを扱うためのバージョン：

```
bool IsSubsetOf(
   ICollection<T>*  collection     // 関係を判断するコレクション
   );
```

配列を扱うためのバージョン：

```
bool IsSubsetOf(
   T&  array[]                     // 関係を判断する配列
   );
```

パラメータ

collection

[in] 関係を判断するコレクション

&collection[]

[in] 関係を判断する配列

戻り値

現在の集合が下位集合である場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

IsSupersetOf

現在の集合が指定されたコレクションまたは配列の上位集合であるかどうかを判断します。

ICollection<T>インターフェイスを実装するコレクションを扱うためのバージョン：

```
bool IsSupersetOf(
   ICollection<T>*  collection     // 関係を判断するコレクション
   );
```

配列を扱うためのバージョン：

```
bool IsSupersetOf(
   T&  array[]                     // 関係を判断する配列
   );
```

パラメータ

collection

[in] 関係を判断するコレクション

&collection[]

[in] 関係を判断する配列

戻り値

現在の集合が上位集合である場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

Overlaps

現在の集合が指定されたコレクションまたは配列と重複するかどうかを判断します。

ICollection<T>インターフェイスを実装するコレクションを扱うためのバージョン：

```
bool Overlaps(
   ICollection<T>*  collection     // 比較するコレクション
   );
```

配列を扱うためのバージョン：

```
bool Overlaps(
   T&  array[]                     // 比較する配列
   );
```

パラメータ

collection

[in] 重複を判断するコレクション

&collection[]

[in] 重複を判断する配列

戻り値

現在の集合とコレクションまたは配列が重なっている場合はtrue、それ例外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

SetEquals

現在の集合が指定されたコレクションまたは配列のすべての要素を含んでいるかどうかを判断します。

ICollection<T>インターフェイスを実装するコレクションを扱うためのバージョン：

```
bool SetEquals(
   ICollection<T>*  collection     // 比較するコレクション
   );
```

配列を扱うためのバージョン：

```
bool SetEquals(
   T&  array[]                     // 比較する配列
   );
```

パラメータ

collection

[in] 要素を比較するコレクション

&collection[]

[in] 要素を比較する配列

戻り値

現在の集合が指定されたコレクションまたは配列のすべての要素を含んでいる場合はtruep、それ例外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

CLinkedList<T>

CLinkedList<T>はICollection<T>インターフェイスを実装するジェネリッククラスです。

説明

CLinkedList<T>クラスは、T型の動的二重連結データリストの実装です。このクラスは、追加、削除、検索要素などの二重連結リストを操作するための基本的なメソッドを提供します。

Declaration

```
template<typename T>
class CLinkedList : public ICollection<T>
```

ヘッダ

```
#include <Generic\LinkedList.mqh>
```

継承階層

ICollection

CLinkedList

クラスメソッド

メソッド	説明
Add	連結リストに要素を追加します。
AddAfter	連結リストの指定されたノードの後に要素を追加します。
AddBefore	連結リストの指定されたノードの前に要素を追加します。
AddFirst	連結リストの初めに要素を追加します。
AddLast	連結リストの終わりに要素を追加します。
kaCount	連結リストの要素数を返します。
Head	連結リストの最初のノードへのポインタを返します。
First	連結リストの最初のノードへのポインタを返します。
Last	連結リストの最後のノードへのポインタを返します。
Contains	連結リストに指定された値を持つ要素が含まれているかどうかを判断します。
CopyTo	連結リストのすべての要素を、指定された配列に指定されたインデックスから開始してコピーします。
Clear	連結リストからすべての要素を削除します。
Remove	連結リストから指定された要素の最初のオカレンスを削除します。
RemoveFirst	連結リストから最初の要素を削除します。

© 2000-2025, MetaQuotes Ltd.

メソッド	説明
RemoveLast	連結リストから最後の要素を削除します。
Find	連結リストで値が最初に出現した位置を検索します。
FindLast	連結リストで値が最後に出現した位置を検索します。

© 2000–2025, MetaQuotes Ltd.

Add

連結リストに要素を追加します。

```
bool Add(
   T  value     // 要素値
   );
```

パラメータ

value

[in] 追加する要素の値

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

AddAfter

連結リストの指定されたノードの後に要素を追加します。

要素を値によって追加するバージョン:

```
CLinkedListNode<T>* AddAfter(
   CLinkedListNode<T>*  node,          // 要素をその後に追加するノード
   T                    value          // 追加する要素
   );
```

戻り値

追加されたノードへのポインタを返します。

要素を値によって形成されたノードとして追加するバージョン:

```
bool AddAfter(
   CLinkedListNode<T>*  node,          // 要素をその後に追加するノード
   CLinkedListNode<T>*  new_node       // 追加されるノード
   );
```

パラメータ

node

[in] 新しい要素をその後に追加するノード

value

[in] 追加される要素

**new_node*

[in] 追加されるノード

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

AddBefore

連結リストの指定されたノードの前に要素を追加します。

要素を値によって追加するバージョン:

```
CLinkedListNode<T>* AddBefore(
   CLinkedListNode<T>*  node,          // 要素をその前に追加するノード
   T                    value          // 追加する要素
   );
```

戻り値

追加されたノードへのポインタを返します。

要素を値によって形成されたノードとして追加するバージョン:

```
bool AddBefore(
   CLinkedListNode<T>*  node,          // 要素をその前に追加するノード
   CLinkedListNode<T>*  new_node       // 追加されるノード
   );
```

パラメータ

node

[in] 新しい要素をその前に追加するノード

value

[in] 追加される要素

**new_node*

[in] 追加されるノード

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

AddFirst

連結リストの初めに要素を追加します。

要素を値によって追加するバージョン:

```
CLinkedListNode<T>* AddFirst(
   T    value                     // 追加する要素
   );
```

戻り値

追加されたノードへのポインタを返します。

要素を値によって形成されたノードとして追加するバージョン:

```
bool AddFirst(
   CLinkedListNode<T>*  node      // 追加するノード
   );
```

パラメータ

value

[in] 追加される要素

**node*

[in] 追加されるノード

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

AddLast

連結リストの終わりに要素を追加します。

要素を値によって追加するバージョン:

```
CLinkedListNode<T>* AddLast(
   T  value                        // 追加する要素
   );
```

戻り値

追加されたノードへのポインタを返します。

要素を値によって形成されたノードとして追加するバージョン:

```
bool AddLast(
   CLinkedListNode<T>*  node      // 追加するノード
   );
```

パラメータ

value

[in] 追加される要素

**node*

[in] 追加されるノード

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

kaCount

連結リストの要素数を返します。

```
int Count();
```

戻り値

要素数を返します。

© 2000–2025, MetaQuotes Ltd.

Head

連結リストの最初のノードへのポインタを返します。

```
CLinkedListNode<T>* Head();
```

戻り値

最初のノードへのポインタを返します。

© 2000-2025, MetaQuotes Ltd.

First

連結リストの最初のノードへのポインタを返します。

```
CLinkedListNode<T>* First();
```

戻り値

最初のノードへのポインタを返します。

© 2000-2025, MetaQuotes Ltd.

Last

連結リストの最後のノードへのポインタを返します。

```
CLinkedListNode<T>* Last();
```

戻り値

最後のノードへのポインタを返します。

© 2000-2025, MetaQuotes Ltd.

Contains

Determines whether the linked list contains an element with the specified value.

```
bool Contains(
   T  item     // 検索値
   );
```

パラメータ

item

[in] 検索値

戻り値

指定された値を持つ要素が連結リストにある場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

CopyTo

連結リストのすべての要素を、指定された配列に指定されたインデックスから開始してコピーします。

```
int CopyTo(
   T&          dst_array[],      // 書き入れ対象の配列
   const int   dst_start=0       // 書き始めのインデックス
   );
```

パラメータ

&dst_array[]

[out] 連結リストの要素が書き込まれる配列

dst_start=0

[in] コピーを開始する配列のインデックス

戻り値

複製された要素の数を返します。

© 2000-2025, MetaQuotes Ltd.

Clear

コレクションからすべての要素を削除します。

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

連結リストから指定された要素の最初のオカレンスを削除します。

要素を値によって削除するバージョン:

```
bool Remove(
   T  item                          // 要素値
   );
```

要素をノードへのポインタを介して削除するバージョン:

```
bool Remove(
   CLinkedListNode<T>*  node      // 要素ノード
   );
```

パラメータ

item

[in] 削除する要素の値

**node*

[in] 削除する要素のノード

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

RemoveFirst

連結リストから最初の要素を削除します。

```
bool RemoveFirst();
```

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

RemoveLast

連結リストから最後の要素を削除します。

```
bool RemoveLast();
```

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

Find

連結リストで値が最初に出現した位置を検索します。

```
CLinkedListNode<T>* Find(
   T  value     // 検索値
   );
```

パラメータ

value

[in] 検索値

戻り値

成功した場合には最初に見つかった検索値を含むノードへのポインタ、それ以外の場合はNULLを返します。

© 2000-2025, MetaQuotes Ltd.

FindLast

連結リストで値が最後に出現した位置を検索します。

```
CLinkedListNode<T>* FindLast(
   T  value     // 検索値
   );
```

パラメータ

value

[in] 検索値

戻り値

成功した場合には最後に見つかった検索値を含むノードへのポインタ、それ以外の場合はNULLを返します。

© 2000-2025, MetaQuotes Ltd.

CQueue<T>

CQueue<T>はICollection<T>インターフェイスを実装するジェネリッククラスです。

説明

CQueue<T>クラスはT型データの動的コレクションで、FIFO(先入れ先出し)原理で動作するキューとして構成されています。

Declaration

```
template<typename T>
class CQueue : public ICollection<T>
```

ヘッダ

```
#include <Generic\Queue.mqh>
```

継承階層

ICollection

CQueue

クラスメソッド

メソッド	説明
Add	キューに要素を追加します。
Enqueue	キューに要素を追加します。
kaCount	キューの要素数を返します。
Contains	キューに指定された値を持つ要素が含まれているかどうかを判断します。
TrimExcess	キューの容量を実際の要素数に設定して、未使用メモリを解放します。
CopyTo	キューのすべての要素を、指定された配列に指定されたインデックスから開始してコピーします。
Clear	キューからすべての要素を削除します。
Remove	キューから指定された要素の最初のオカレンスを削除します。
Dequeue	最初の要素をキューから削除して返します。
Peek	最初の要素をキューから削除せずに返します。

© 2000-2025, MetaQuotes Ltd.

Add

キューに要素を追加します。

```
bool Add(
   T  value     // 要素値
   );
```

パラメータ

value

[in] 追加する要素の値

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

Enqueue

キューに要素を追加します。

```
bool Enqueue(
   T  value     // 追加する要素
   );
```

パラメータ

value

[in] 追加される要素

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

kaCount

キューの要素数を返します。

```
int Count();
```

戻り値

要素数を返します。

© 2000–2025, MetaQuotes Ltd.

Contains

キューに指定された値を持つ要素が含まれているかどうかを判断します。

```
bool Contains(
   T  item     // 検索値
   );
```

パラメータ

item

[in] 検索値

戻り値

指定された値を持つ要素がキューにある場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

TrimExcess

キューの容量を実際の要素数に設定して、未使用メモリを解放します。

```
void TrimExcess();
```

© 2000-2025, MetaQuotes Ltd.

CopyTo

キューのすべての要素を、指定された配列に指定されたインデックスから開始してコピーします。

```
int CopyTo(
   T&          dst_array[],      // 書き入れ対象の配列
   const int   dst_start=0       // 書き始めのインデックス
   );
```

パラメータ

&dst_array[]

[out] キューの要素が書き込まれる配列

dst_start=0

[in] コピーを開始する配列のインデックス

戻り値

複製された要素の数を返します。

© 2000-2025, MetaQuotes Ltd.

Clear

キューからすべての要素を削除します。

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

キューから指定された要素の最初のオカレンスを削除します。

```
bool Remove(
   T  item     // 要素値
   );
```

パラメータ

item

[in] 削除する要素の値

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

Dequeue

最初の要素をキューから削除して返します。

```
T Dequeue();
```

戻り値

最初の要素を返します。

© 2000-2025, MetaQuotes Ltd.

Peek

最初の要素をキューから削除せずに返します。

```
T Peek();
```

戻り値

最初の要素を返します。

© 2000-2025, MetaQuotes Ltd.

CRedBlackTree<T>

CRedBlackTree<T>はICollection<T>インターフェイスを実装するジェネリッククラスです。

説明

CRedBlackTree<T>クラスは、ノードがT型のデータを格納する動的な赤黒木の実装です。このクラスでは、追加、削除、最大値と最小値の検索などの赤黒木を操作するための基本的なメソッドを提供しています。

Declaration

```
template<typename T>
class CRedBlackTree : public ICollection<T>
```

ヘッダ

```
#include <Generic\RedBlackTree.mqh>
```

継承階層

ICollection

CRedBlackTree

クラスメソッド

メソッド	説明
Add	赤黒木に要素を追加します。
Root	赤黒木の根へのポインタを返します。
kaCount	赤黒木の要素数を返します。
Contains	赤黒木に指定された値を持つ要素が含まれているかどうかを判断します。
Comparer	赤黒木を整理するために使用されるIComparer<T>インターフェイスへのポインタを返します。
TryGetMin	赤黒木の最小要素を取得します。
TryGetMax	赤黒木の最大要素を取得します。
CopyTo	赤黒木のすべての要素を、指定された配列に指定されたインデックスから開始してコピーします。
Clear	赤黒木からすべての要素を削除します。
Remove	赤黒木から指定された要素の最初のオカレンスを削除します。
RemoveMin	赤黒木から最小値を持つ要素を削除します。
RemoveMax	赤黒木から最大値を持つ要素を削除します。
Find	赤黒木で指定された要素を検索します。

© 2000–2025, MetaQuotes Ltd.

メソッド	説明
FindMax	赤黒木で最大値を持つ要素を検索します。
FindMin	赤黒木で最小値を持つ要素を検索します。

© 2000–2025, MetaQuotes Ltd.

Add

赤黒木に要素を追加します。

```
bool Add(
   T  value      // 追加する要素
   );
```

パラメータ

value

[in] 追加される要素

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

kaCount

赤黒木の要素数を返します。

```
int Count();
```

戻り値

要素数を返します。

© 2000–2025, MetaQuotes Ltd.

Root

赤黒木の根へのポインタを返します。

```
CRedBlackTreeNode<T>* Root();
```

戻り値

根へのポインタを返します。

© 2000–2025, MetaQuotes Ltd.

Contains

赤黒木に指定された値を持つ要素が含まれているかどうかを判断します。

```
bool Contains(
   T  item     // 検索値
   );
```

パラメータ

item

[in] 検索値

戻り値

指定された値を持つ要素が赤黒木にある場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

Comparer

赤黒木を整理するために使用されるIComparer<T>インターフェイスへのポインタを返します。

```
IComparer<T>* Comparer() const;
```

戻り値

IComparer<T>インターフェイスへのポインタを返します。

© 2000-2025, MetaQuotes Ltd.

TryGetMin

赤黒木の最小要素を取得します。

```
bool TryGetMin(
   T&  min     // 値を書き入れるための変数
   );
```

パラメータ

&min

[out] 最小値を書き入れるための変数

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

TryGetMax

赤黒木の最大要素を取得します。

```
bool TryGetMax(
   T&  max      // 値を書き入れるための変数
   );
```

パラメータ

&max

[out] 最大値を書き入れるための変数

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

CopyTo

赤黒木のすべての要素を、指定された配列に指定されたインデックスから開始してコピーします。

```
int CopyTo(
   T&          dst_array[],      // 書き入れ対象の配列
   const int   dst_start=0       // 書き始めのインデックス
   );
```

パラメータ

&dst_array[]

[out] 赤黒木の要素が書き込まれる配列

dst_start=0

[in] コピーを開始する配列のインデックス

戻り値

複製された要素の数を返します。

© 2000-2025, MetaQuotes Ltd.

Clear

赤黒木からすべての要素を削除します。

```
void Clear();
```

© 2000–2025, MetaQuotes Ltd.

Remove

赤黒木から指定された要素の最初のオカレンスを削除します。

指定された値を持つ要素を削除するバージョン:

```
bool Remove(
   T  value                         // 要素値
   );
```

要素をノードへのポインタを介して削除するバージョン:

```
bool Remove(
   CRedBlackTreeNode<T>*  node      // 要素ノード
   );
```

パラメータ

item

[in] 削除する要素の値

**node*

[in] 削除する要素のノード

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

RemoveMin

赤黒木から最小値を持つ要素を削除します。

```
bool RemoveMin();
```

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

RemoveMax

赤黒木から最大値を持つ要素を削除します。

```
bool RemoveMax();
```

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

Find

赤黒木で指定された要素を検索します。

```
CRedBlackTreeNode<T>* Find(
   T  value     // 検索値
   );
```

パラメータ

value

[in] 検索値

戻り値

成功した場合には見つかった検索値を含むノードへのポインタ、それ以外の場合はNULLを返します。

© 2000-2025, MetaQuotes Ltd.

FindMin

赤黒木で最小値を持つ要素を検索します。

```
CRedBlackTreeNode<T>* FindMin();
```

戻り値

成功した場合には見つかった最小値を含むノードへのポインタ、それ以外の場合はNULLを返します。

© 2000-2025, MetaQuotes Ltd.

FindMax

赤黒木で最大値を持つ要素を検索します。

```
CRedBlackTreeNode<T>* FindMax();
```

戻り値

成功した場合には見つかった最大値を含むノードへのポインタ、それ以外の場合はNULLを返します。

© 2000-2025, MetaQuotes Ltd.

CSortedMap<TKey, TValue>

CSortedMap<TKey, TValue>はIMap<TKey, TValue>インターフェイスを実装するジェネリッククラスです。

説明

CSortedMap<TKey, TValue>クラスは、データがキーでソートされ、キーの一意性の要件を考慮してキー/値ペアとして格納される動的ハッシュテーブルの実装です。このクラスは、キーで値にアクセスし、キー/値ペアを検索して削除するなどのハッシュテーブルを操作するための基本的なメソッドを提供します。

Declaration

```
template<typename TKey, typename TValue>
class CSortedMap : public IMap<TKey, TValue>
```

ヘッダ

```
#include <Generic\SortedMap.mqh>
```

継承階層

ICollection
  IMap
    CSortedMap

クラスメソッド

メソッド	説明
Add	ハッシュテーブルにキー/値ペアを追加します。
kaCount	ソート済みハッシュテーブルの要素数を返します。
Contains	ソート済みハッシュテーブルに指定されたキー/値ペアが含まれているかどうかを判断します。
ContainsKey	ソート済みハッシュテーブルに指定されたキーを持つキー/値のペアが含まれているかどうかを判断します。
ContainsValue	ソート済みハッシュテーブルに指定された値を持つキー/値のペアが含まれているかどうかを判断します。
Comparer	ソート済みハッシュテーブルを整理するために使用されるIComparer<T>インターフェイスへのポインタを返します。
CopyTo	ソート済みハッシュテーブルのすべてのキー/値ペアを指定された配列に、指定されたインデックスから開始してコピーします。
Clear	ソート済みハッシュテーブルからすべての要素を削除します。
Remove	ソート済みハッシュテーブルから指定されたキー/値ペアの最初のオカレンスを削除します。
TryGetValue	ソート済みハッシュテーブルから指定されたキーを持つ要素を取得します。

© 2000–2025, MetaQuotes Ltd.

メソッド	説明
TrySetValue	ソート済みハッシュテーブルで指定されたキーを持つ要素を変更します。

© 2000–2025, MetaQuotes Ltd.

Add

ハッシュテーブルにキー/値 ペアを追 加します。

生 成 されたキー/値 ペアを追 加 するバージョン:

```
bool Add(
   CKeyValuePair<TKeyTValue>*  pair     // キー/値ペア
   );
```

指 定 されたキーと値 でキー/値 ペアを追 加 するバージョン:

```
bool Add(
   TKey    key,                         // キー
   TValue  value                        // 値
   );
```

パラメータ

pair
  [in] キー/値 ペア

key
  [in] キー

value
  [in] 値 。

戻 り値

成 功 の場 合 はtrue、それ以 外 の場 合 はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

kaCount

ソート済みハッシュテーブルの要素数を返します。

```
int Count();
```

© 2000–2025, MetaQuotes Ltd.

Comparer

ソート済みハッシュテーブルを整理するために使用されるIComparer<T>インターフェイスへのポインタを返します。

```
IComparer<TKey>* Comparer() const;
```

戻り値

IComparer<T>インターフェイスへのポインタを返します。

© 2000-2025, MetaQuotes Ltd.

Contains

ソート済みハッシュテーブルに指定されたキー/値ペアが含まれているかどうかを判断します。

生成されたキー/値ペアを処理するバージョン:

```
bool Contains(
   CKeyValuePair<TKeyTValue>*  item     // キー/値ペア
   );
```

個別に設定されたキーと値の形式でキー/値ペアを処理するバージョン.

```
bool Contains(
   TKey    key,                         // キー
   TValue  value                        // 値
   );
```

パラメータ

item

[in] キー/値ペア

key

[in] キー

value

[in] 値。

戻り値

ソート済みハッシュテーブルに指定されたキーを持つキー/値のペアが含まれている場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

ContainsKey

ソート済みハッシュテーブルに指定されたキーを持つキー/値のペアが含まれているかどうかを判断します。

```
bool ContainsKey(
   TKey  key     // キー
   );
```

パラメータ

key

[in] キー

戻り値

ソート済みハッシュテーブルに指定されたキーを持つキー/値のペアが含まれている場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

ContainsValue

ソート済みハッシュテーブルに指定された値を持つキー/値のペアが含まれているかどうかを判断します。

```
bool ContainsValue(
   TValue  value     // 値
   );
```

パラメータ

value

[in] 値。

戻り値

ソート済みハッシュテーブルに指定された値を持つキー/値のペアが含まれている場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

CopyTo

ソート済みハッシュテーブルのすべてのキー/値ペアを指定された配列に、指定されたインデックスから開始してコピーします。

ハッシュテーブルをキー/値ペアの配列にコピーするバージョン:

```
int CopyTo(
   CKeyValuePair<TKeyTValue>*&   dst_array[],      // キー/値ペアを書き入れる配列
   const int                     dst_start=0       // 書き始めのインデックス
   );
```

ハッシュテーブルを別々のキー配列と値配列にコピーするバージョン:

```
int CopyTo(
   TKey&       dst_keys[],                         // キーを書き入れ対象の配列
   TValue&     dst_values[],                       // 値を書き入れ対象の配列
   const int   dst_start=0                         // 書き始めのインデックス
   );
```

パラメータ

&dst_array[]

[out] ハッシュテーブルのすべてのペアが書き込まれる配列

&dst_keys[]

[out] ハッシュテーブルのすべてのキーが書き込まれる配列

&dst_values[]

[out] ハッシュテーブルのすべての値が書き込まれる配列

dst_start=0

[in] コピーを開始する配列のインデックス

戻り値

複製されたキー/値ペアの数を返します。

© 2000-2025, MetaQuotes Ltd.

Clear

ソート済みハッシュテーブルからすべての要素を削除します。

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

ソート済みハッシュテーブルから指定されたキー/値ペアの最初のオカレンスを削除します。

生成されたキー/値ペアを削除するバージョン:

```
bool Remove(
   CKeyValuePair<TKeyTValue>*  item     // キー/値ペア
   );
```

キーに基づいてキー/値ペアを削除するバージョン:

```
bool Remove(
   TKey  key                            // キー
   );
```

パラメータ

item

[in] キー/値ペア

key

[in] キー

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

TryGetValue

ソート済みハッシュテーブルから指定されたキーを持つ要素を取得します。

```
bool TryGetValue(
   TKey     key,       // キー
   TValue&  value      // 値を書き入れるための変数
   );
```

パラメータ

key

[in] キー

&value

[out] 指定されたキー/値ペアの値が書き込まれる変数

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

TrySetValue

ハッシュテーブルの指定されたキーでのキー/値ペアの値を変更します。

```
bool TrySetValue(
   TKey    key,       // キー
   TValue  value      // 新しい値
   );
```

パラメータ

key

[in] キー

value

[in] 指定されたキー/値ペアに割り当てる新しい値

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

CSortedSet<T>

CSortedSet<T>はISet<T>インターフェイスを実装するジェネリッククラスです。

説明

CHashSet<T>クラスは、各値が一意な、T型の順序付けられている動的データセットの実装です。このクラスは、集合の操作とその関連操作（和集合と共通部分、下位集合や真下位集合の定義など）のための基本的なメソッドを提供します。

Declaration

```
template<typename T>
class CSortedSet : public ISet<T>
```

ヘッダ

```
#include <Generic\SortedSet.mqh>
```

継承階層

ICollection
  ISet
    CSortedSet

クラスメソッド

メソッド	説明
Add	ソート済み集合に要素を追加します。
kaCount	ソート済み集合の要素数を返します。
Contains	ソートされた集合に指定された値を持つ要素が含まれているかどうかを判断します。
Comparer	ソート済み集合を整理するために使用されるIComparer<T>インターフェイスへのポインタを返します。
TryGetMin	ソート済み集合の指定された要素を取得します。
TryGetMax	ソート済み集合の最大要素を取得します。
CopyTo	ソート済み集合のすべての要素を、指定された配列に指定されたインデックスから開始してコピーします。
Clear	ソート済み集合からすべての要素を削除します。
Remove	ソート済み集合から指定された要素を削除します。
ExceptWith	現在のコレクションと渡されたコレクション（配列）の差集合操作を生成します。
IntersectWith	現在のコレクションと渡されたコレクション（配列）の共通集合操作を生成します。

© 2000–2025, MetaQuotes Ltd.

メソッド	説明
SymmetricExceptWith	現在の集合と指定されたコレクション（配列）の対称差集合操作を生成します。
UnionWith	現在の集合と指定されたコレクションまたは配列の和集合操作を生成します。
IsProperSubsetOf	現在のソート済み集合が指定されたコレクションまたは配列の真下位集合であるかどうかを判断します。
IsProperSupersetOf	現在のソート済み集合が指定されたコレクションまたは配列の真上位集合であるかどうかを判断します。
IsSubsetOf	現在のソート済み集合が指定されたコレクションまたは配列の下位集合であるかどうかを判断します。
IsSupersetOf	現在のソート済み集合が指定されたコレクションまたは配列の上位集合であるかどうかを判断します。
Overlaps	現在のソート済み集合が指定されたコレクションまたは配列と重複するかどうかを判断します。
SetEquals	現在のソート済み集合が指定されたコレクションまたは配列のすべての要素を含んでいるかどうかを判断します。
GetViewBetween	現在のソート済み集合から、最小値と最大値で指定された下位集合を取得します。
GetReverse	現在のソート済み集合のすべての要素が逆の順序で配置されているコピーを取得します。

© 2000–2025, MetaQuotes Ltd.

Add

ソート済み集合に要素を追加します。

```
bool Add(
   T  value     // 要素値
   );
```

パラメータ

value

[in] 追加する要素の値

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

kaCount

ソート済み集合の要素数を返します。

```
int Count();
```

戻り値

要素数を返します。

© 2000–2025, MetaQuotes Ltd.

Contains

ソートされた集合に指定された値を持つ要素が含まれているかどうかを判断します。

```
bool Contains(
   T  item     // 検索値
   );
```

パラメータ

item

[in] 検索値

戻り値

指定された値を持つ要素が集合にある場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

Comparer

ソート済み集合を整理するために使用されるIComparer<T>インターフェイスへのポインタを返します。

```
IComparer<T>* Comparer() const;
```

戻り値

IComparer<T>インターフェイスへのポインタを返します。

© 2000-2025, MetaQuotes Ltd.

TryGetMin

ソート済み集合の指定された要素を取得します。

```
bool TryGetMin(
   T&  min     // 値を書き入れるための変数
   );
```

パラメータ

&min

[out] 最小値を書き入れるための変数

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

TryGetMax

ソート済み集合の最大要素を取得します。

```
bool TryGetMax(
   T&  max      // 値を書き入れるための変数
   );
```

パラメータ

&max

[out] 最大値を書き入れるための変数

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

CopyTo

ソート済み集合のすべての要素を、指定された配列に指定されたインデックスから開始してコピーします。

```
int CopyTo(
   T&          dst_array[],      // 書き入れ対象の配列
   const int   dst_start=0       // 書き始めのインデックス
   );
```

パラメータ

&dst_array[]

[out] 集合の要素が書き込まれる配列

dst_start=0

[in] コピーを開始する配列のインデックス

戻り値

複製された要素の数を返します。

© 2000-2025, MetaQuotes Ltd.

Clear

ソート済み集合からすべての要素を削除します。

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

ソート済み集合から指定された要素を削除します。

```
bool Remove(
   T  item     // 要素値
   );
```

パラメータ

item

[in] 削除する要素の値

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

ExceptWith

現在のコレクションと渡されたコレクション(配列)の差集合操作を生成します。現在のコレクション(配列)から指定されたコレクション(配列)に存在するすべての要素を削除します。

ICollection<T>インターフェイスを実装するコレクションを扱うためのバージョン:

```
void ExceptWith(
   ICollection<T>*  collection     // コレクション
   );
```

配列を扱うためのバージョン:

```
void ExceptWith(
   T&  array[]                     // 配列
   );
```

パラメータ

collection

[in] 現在のソート済み集合との差が計算されるコレクション

&collection[]

[in] 現在のソート済み集合との差が計算される配列

注意事項

結果は現在のコレクション(配列)に書き込まれます。

© 2000-2025, MetaQuotes Ltd.

IntersectWith

現在のコレクションと渡されたコレクション(配列)の共通集合操作を生成します。現在のコレクションを変更して、指定されたコレクション(配列)に存在する要素のみを含むようにします。

ICollection<T>インターフェイスを実装するコレクションを扱うためのバージョン:

```
void IntersectWith(
   ICollection<T>*  collection     // コレクション
   );
```

配列を扱うためのバージョン:

```
void IntersectWith(
   T&  array[]                     // 配列
   );
```

パラメータ

**collection*

[in] 現在の集合が交差するコレクション

&collection[]

[in] 現在の集合が交差する配列

注意事項

結果は現在のコレクション(配列)に書き込まれます。

© 2000-2025, MetaQuotes Ltd.

SymmetricExceptWith

現在の集合と指定されたコレクション(配列)の対称差集合操作を生成します。現在のコレクションを変更して、ソースオブジェクトまたは指定されたコレクション(配列)の一方にのみ存在する要素のみを含むようにします。

ICollection<T>インターフェイスを実装するコレクションを扱うためのバージョン:

```
void SymmetricExceptWith(
   ICollection<T>*  collection     // コレクション
   );
```

配列を扱うためのバージョン:

```
void SymmetricExceptWith(
   T&  array[]                     // 配列
   );
```

パラメータ

collection

[in] 対称差を生成するコレクション

&collection[]

[in] 対称差を生成する配列

注意事項

結果は現在のコレクション(配列)に書き込まれます。

© 2000-2025, MetaQuotes Ltd.

UnionWith

現在の集合と指定されたコレクションまたは配列の和集合操作を生成します。現在のコレクション(配列)に、指定されたコレクション(配列)から不足している要素を追加します。

ICollection<T>インターフェイスを実装するコレクションを扱うためのバージョン:

```
void UnionWith(
   ICollection<T>*  collection     // コレクション
   );
```

配列を扱うためのバージョン:

```
void UnionWith(
   T&  array[]                     // 配列
   );
```

パラメータ

collection

[in] 現在の集合と和結合するコレクション

&collection[]

[in] 現在の集合と和結合する配列

注意事項

結果は現在のコレクション(配列)に書き込まれます。

© 2000-2025, MetaQuotes Ltd.

IsProperSubsetOf

現在のソート済み集合が指定されたコレクションまたは配列の真下位集合であるかどうかを判断します。

ICollection<T>インターフェイスを実装するコレクションを扱うためのバージョン：

```
bool IsProperSubsetOf(
   ICollection<T>*  collection     // 関係を判断するコレクション
   );
```

配列を扱うためのバージョン：

```
bool IsProperSubsetOf(
   T&  array[]                     // 関係を判断する配列
   );
```

パラメータ

collection

[in] 関係を判断するコレクション

&collection[]

[in] 関係を判断する配列

戻り値

現在のソート済み集合が真下位集合である場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

IsProperSupersetOf

現在のソート済み集合が指定されたコレクションまたは配列の真上位集合であるかどうかを判断します。

ICollection<T>インターフェイスを実装するコレクションを扱うためのバージョン：

```
bool IsProperSupersetOf(
   ICollection<T>*  collection     // 関係を判断するコレクション
   );
```

配列を扱うためのバージョン：

```
bool IsProperSupersetOf(
   T&  array[]                     // 関係を判断する配列
   );
```

パラメータ

collection

[in] 関係を判断するコレクション

&collection[]

[in] 関係を判断する配列

戻り値

現在のソート済み集合が真上位集合である場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

IsSubsetOf

現在のソート済み集合が指定されたコレクションまたは配列の下位集合であるかどうかを判断します。

ICollection<T>インターフェイスを実装するコレクションを扱うためのバージョン：

```
bool IsSubsetOf(
   ICollection<T>*  collection     // 関係を判断するコレクション
   );
```

配列を扱うためのバージョン：

```
bool IsSubsetOf(
   T&  array[]                     // 関係を判断する配列
   );
```

パラメータ

collection

[in] 関係を判断するコレクション

&collection[]

[in] 関係を判断する配列

戻り値

現在のソート済み集合が下位集合である場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

IsSupersetOf

現在のソート済み集合が指定されたコレクションまたは配列の上位集合であるかどうかを判断します。

ICollection<T>インターフェイスを実装するコレクションを扱うためのバージョン：

```
bool IsSupersetOf(
   ICollection<T>*  collection     // 関係を判断するコレクション
   );
```

配列を扱うためのバージョン：

```
bool IsSupersetOf(
   T&  array[]                     // 関係を判断する配列
   );
```

パラメータ

collection

[in] 関係を判断するコレクション

&collection[]

[in] 関係を判断する配列

戻り値

現在のソート済み集合が上位集合である場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

Overlaps

現在のソート済み集合が指定されたコレクションまたは配列と重複するかどうかを判断します。

ICollection<T>インターフェイスを実装するコレクションを扱うためのバージョン：

```
bool Overlaps(
   ICollection<T>*  collection     // 比較するコレクション
   );
```

配列を扱うためのバージョン：

```
bool Overlaps(
   T&  array[]                     // 比較する配列
   );
```

パラメータ

collection

[in] 重複を判断するコレクション

&collection[]

[in] 重複を判断する配列

戻り値

現在のソート済み集合とコレクションまたは配列が重なっている場合はtrue、それ例外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

SetEquals

現在のソート済み集合が指定されたコレクションまたは配列のすべての要素を含んでいるかどうかを判断します。

ICollection<T>インターフェイスを実装するコレクションを扱うためのバージョン：

```
bool SetEquals(
   ICollection<T>*  collection     // 比較するコレクション
   );
```

配列を扱うためのバージョン：

```
bool SetEquals(
   T&  array[]                     // 比較する配列
   );
```

パラメータ

collection

[in] 要素を比較するコレクション

&collection[]

[in] 要素を比較する配列

戻り値

現在のソート済み集合が指定されたコレクションまたは配列のすべての要素を含んでいる場合はtruep、それ例外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

GetViewBetween

現在のソート済み集合から、最小値と最大値で指定された下位集合を取得します。

```
bool GetViewBetween(
   T&  array[],          // 書き入れ対象の配列
   T   lower_value,      // 最高値
   T   upper_value       // 最小値
   );
```

パラメータ

&array[]

[out] 下位集合を書き入れるための配列

lower_value

[in] 範囲の最小値

upper_value

[in] 範囲の最大値

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

GetReverse

現在のソート済み集合のすべての要素が逆の順序で配置されているコピーを取得します。

```
bool GetReverse(
   T&  array[]      // 書き入れ対象の配列
   );
```

パラメータ

&array[]

[out] 書き入れ対象の配列

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

CStack<T>

CStack<T>はICollection<T>インターフェイスを実装するジェネリッククラスです。

説明

CStack<T>classはT型データの動的コレクションで、LIFO(ラスト・イン・ファースト・アウト)原理で動作するスタックとして編成されています。

Declaration

```
template<typename T>
class CStack : public ICollection<T>
```

ヘッダ

```
#include <Generic\Stack.mqh>
```

継承階層

ICollection

CStack

クラスメソッド

メソッド	説明
Add	スタックに要素を追加します。
kaCount	スタックの要素数を返します。
Contains	スタックに指定された値を持つ要素が含まれているかどうかを判断します。
TrimExcess	スタックの容量を実際の要素数に設定します。
CopyTo	スタックのすべての要素を、指定された配列に指定されたインデックスから開始してコピーします。
Clear	スタックからすべての要素を削除します。
Remove	スタックから指定された要素の最初のオカレンスを削除します。
Push	スタックに要素を追加します。
Peek	先頭要素をスタックから削除せずに返します。
Pop	先頭要素をスタックから削除してに返します。

© 2000-2025, MetaQuotes Ltd.

Add

スタックに要素を追加します。

```
bool Add(
   T  value     // 要素値
   );
```

パラメータ

value

[in] 追加する要素の値

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

kaCount

スタックの要素数を返します。

```
int Count();
```

戻り値

要素数を返します。

© 2000–2025, MetaQuotes Ltd.

Contains

スタックに指定された値を持つ要素が含まれているかどうかを判断します。

```
bool Contains(
   T  item     // 検索値
   );
```

パラメータ

item

[in] 検索値

戻り値

指定された値を持つ要素がスタックにある場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

TrimExcess

スタックの容量を実際の要素数に設定して、未使用メモリを解放します。

```
void TrimExcess();
```

© 2000-2025, MetaQuotes Ltd.

CopyTo

スタックのすべての要素を、指定された配列に指定されたインデックスから開始してコピーします。

```
int CopyTo(
   T&          dst_array[],      // 書き入れ対象の配列
   const int   dst_start=0       // 書き始めのインデックス
   );
```

パラメータ

&dst_array[]

[out] スタックの要素が書き込まれる配列

dst_start=0

[in] コピーを開始する配列のインデックス

戻り値

複製された要素の数を返します。

© 2000-2025, MetaQuotes Ltd.

Clear

スタックからすべての要素を削除します。

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

スタックから指定された要素の最初のオカレンスを削除します。

```
bool Remove(
   T  item     // 要素値
   );
```

パラメータ

item

[in] 削除する要素の値

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

Push

スタックに要素を追加します。

```
bool Push(
   T  value      // 追加する要素
   );
```

パラメータ

value

[in] 追加される要素

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

Peek

先頭要素をスタックから削除せずに返します。

```
T Peek();
```

戻り値

先頭要素を返します。

© 2000-2025, MetaQuotes Ltd.

Pop

先頭要素をスタックから削除してに返します。

```
T Pop();
```

戻り値

先頭要素を返します。

© 2000-2025, MetaQuotes Ltd.

ArrayBinarySearch

IComparable<T>を使用して要素を比較して昇順ソートされた1次元配列で指定された値を検索します。

```
template<typename T>
int ArrayBinarySearch(
   T&              array[],          <t 5// 検索対象の配列
   const int       start_index,      // 開始インデックス
   const int       count,            // 検索範囲
   T               value,            // 検索値
   IComparer<T>*   comparer          // 比較インターフェイス
   );
```

パラメータ

&array[]

[out] 検索される配列

value

[in] 検索値

**comparer*

[in] 要素比較インターフェイス

start_index

[in] 検索開始インデックス

count

[in] 検索範囲の長さ

戻り値

見つかった要素のインデックスを返します。検索値が見つからない場合は、値が最も小さい最小の要素のインデックスを返します。

© 2000–2025, MetaQuotes Ltd.

ArrayIndexOf

1次元配列で値の最初の出現を検索します。

```
template<typename T>
int ArrayIndexOf(
   T&          array[],          <t 5// 検索対象の配列
   T           value,            // 検索値
   const int   start_index,      // 開始インデックス
   const int   count             // 検索範囲
   );
```

パラメータ

&array[]

[out] 検索される配列

value

[in] 検索値

start_index

[in] 検索開始インデックス

count

[in] 検索範囲の長さ

戻り値

初めに見つかった要素のインデックスを返します。値が見つからなかった場合は-1を返します。

© 2000-2025, MetaQuotes Ltd.

ArrayLastIndexOf

1次元配列で値の最後の出現を検索します。

```
template<typename T>
int ArrayLastIndexOf(
   T&          array[],          <t 5// 検索対象の配列
   T           value,            // 検索値
   const int   start_index,      // 開始インデックス
   const int   count             // 検索範囲
   );
```

パラメータ

&array[]

[out] 検索される配列

value

[in] 検索値

start_index

[in] 検索開始インデックス

count

[in] 検索範囲の長さ

戻り値

最後に見つかった要素のインデックスを返します。値が見つからなかった場合は-1を返します。

© 2000-2025, MetaQuotes Ltd.

ArrayReverse

1次元配列の要素の順序を変更します。

```
template<typename T>
bool ArrayReverse(
   T&          array[],          // ソース配列
   const int   start_index,      // 開始インデックス
   const int   count             // 要素数
   );
```

パラメータ

&array[]

[out] ソース配列

start_index

[in] 開始インデックス

count

[in] 操作に関与する配列要素の数

戻り値

成功の場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

Compare

2つの値をそのうち1つがあと1つと比べて「より大きい、より小さい、等しい」かで比較します。

2つのブール値を扱うためのバージョン：

```
int Compare(
   const bool   x,          // 1番目の値
   const bool   y           // 2番目の値
   );
```

2つのchar値を扱うためのバージョン：

```
int Compare(
   const char   x,          // 1番目の値
   const char   y           // 2番目の値
   );
```

2つのuchar値を扱うためのバージョン：

```
int Compare(
   const uchar   x,         // 1番目の値
   const uchar   y          // 2番目の値
   );
```

2つのshort値を扱うためのバージョン：

```
int Compare(
   const short   x,         // 1番目の値
   const short   y          // 2番目の値
   );
```

2つのushort値を扱うためのバージョン：

```
int Compare(
   const ushort   x,        // 1番目の値
   const ushort   y         // 2番目の値
   );
```

2つのcolor値を扱うためのバージョン：

```
int Compare(
   const color   x,         // 1番目の値
   const color   y          // 2番目の値
   );
```

2つのint値を扱うためのバージョン：

```
int Compare(
   const int   x,           // 1番目の値
   const int   y            // 2番目の値
   );
```

© 2000-2025, MetaQuotes Ltd.

2つのuint値 を扱 うためのバージョン:

```
int Compare(
   const uint  x,          // 1番目の値
   const uint  y           // 2番目の値
   );
```

2つのdatetime値 を扱 うためのバージョン:

```
int Compare(
   const datetime  x,      // 1番目の値
   const datetime  y       // 2番目の値
   );
```

2つのlong値 を扱 うためのバージョン:

```
int Compare(
   const long  x,          // 1番目の値
   const long  y           // 2番目の値
   );
```

2つのulong値 を扱 うためのバージョン:

```
int Compare(
   const ulong  x,         // 1番目の値
   const ulong  y          // 2番目の値
   );
```

2つの float値 を扱 うためのバージョン:

```
int Compare(
   const float  x,         // 1番目の値
   const float  y          // 2番目の値
   );
```

2つのdouble値 を扱 うためのバージョン:

```
int Compare(
   const double  x,        // 1番目の値
   const double  y         // 2番目の値
   );
```

2つのstring値 を扱 うためのバージョン:

```
int Compare(
   const string  x,        // 1番目の値
   const string  y         // 2番目の値
   );
```

2つのその他 の型 の値 を扱 うためのバージョン:

```
template<typename T>
int Compare(
```

© 2000–2025, MetaQuotes Ltd.

```
   T  x,                    // 1番目の値
   T  y                     // 2番目の値
   );
```

パラメータ

x

[in] 1番目の値

y

[in] 2番目の値

戻り値

比較された2つの値の比を表す数値を返します。

- 結果が0より小さい場合、xはyより小さい(x <y)です。
- 結果がゼロに等しい場合、xはyに等しい(x = y)です。
- 結果がゼロより大きい場合、xはyより大きい(x> y)です。

注意事項

T型がIComparable<T>インターフェイスを実装するオブジェクトである場合、オブジェクトはそのCompareメソッドに基づいて比較されます。その他のすべての場合には0が返されます。

© 2000-2025, MetaQuotes Ltd.

Equals

2つの値が等しいかどうかを比較します。

```
template<typename T>
bool Equals(
   T  x,      // 1番目の値
   T  y       // 2番目の値
   );
```

パラメータ

x

[in] 1番目の値

y

[in] 2番目の値

戻り値

オブジェクトが等しい場合はtrue、それ以外の場合はfalseを返します。

注意事項

T型がIEqualityComparable<T>インターフェイスを実装するオブジェクトである場合、オブジェクトはそのEqualsメソッドに基づいて比較されます。他のすべての場合には、平等のための標準的な比較が使用されます。

© 2000-2025, MetaQuotes Ltd.

GetHashCode

ハッシュコード値を計算します。

ブール型を扱うためのバージョン：

```
int GetHashCode(
   const bool   value         // 値
   );
```

char型を扱うためのバージョン：

```
int GetHashCode(
   const char   value         // 値
   );
```

uchar型を扱うためのバージョン：

```
int GetHashCode(
   const uchar   value        // 値
   );
```

short型を扱うためのバージョン：

```
int GetHashCode(
   const short   value        // 値
   );
```

ushort型を扱うためのバージョン：

```
int GetHashCode(
   const ushort   value       // 値
   );
```

color型を扱うためのバージョン：

```
int GetHashCode(
   const color   value        // 値
   );
```

int型を扱うためのバージョン：

```
int GetHashCode(
   const int   value          // 値
   );
```

uint型を扱うためのバージョン：

```
int GetHashCode(
   const uint   value         // 値
   );
```

datetime型を扱うためのバージョン：

```
int GetHashCode(
```

© 2000–2025, MetaQuotes Ltd.

```
   const datetime   value      // 値
   );
```

long型を扱うためのバージョン：

```
int GetHashCode(
   const long   value          // 値
   );
```

ulong型を扱うためのバージョン：

```
int GetHashCode(
   const ulong   value         // 値
   );
```

float型を扱うためのバージョン：

```
int GetHashCode(
   const float   value         // 値
   );
```

double型を扱うためのバージョン：

```
int GetHashCode(
   const double   value        // 値
   );
```

string型を扱うためのバージョン：

```
int GetHashCode(
   const string   value        // 値
   );
```

その他の型を扱うためのバージョン：

```
template<typename T>
int GetHashCode(
   T   value                   // 値
   );
```

パラメータ

value

[in] ハッシュコードを取得したい値

戻り値

ハッシュコードを返します。

注意事項

T型がIEqualityComparable<T>インターフェイスを実装するオブジェクトである場合、ハッシュコードはそのHashCodeメソッドに基づいて取得されます。それ以外の場合では全て、ハッシュコードは値型名のハッシュ値として計算されます。

© 2000–2025, MetaQuotes Ltd.

© 2000–2025, MetaQuotes Ltd.

ファイル操作

このセクションには、ファイル操作のクラスの操作に関する技術的な詳細と関連するMQL5　標準ライブラリコンポーネントの説明が含まれています。

ファイル操作クラスの使用は、ファイル入出力操作を使用するアプリケーションの開発時に時間を節約します。

MQL5 標準ライブラリのファイル操作に関連したファイルは、端末の作業ディレクトリの Include¥Files フォルダ内に配置されています。

Class	説明
CFile	ファイル操作の基本クラス
CFileBin	バイナリファイル操作のクラス
CFileTxt	テキストファイル操作のクラス

© 2000–2025, MetaQuotes Ltd.

CFile

CFile は CFileBin 及び CFileTxt クラスの基本クラスです。

説明

CFile クラスは派生クラスの MQL5 API 関数への容易化されたアクセスを提供します。

宣言

```
class CFile: public CObject
```

タイトル

```
#include <Files\File.mqh>
```

継承階層

CObject
  CFile

直接子孫

CFileBin, CFilePipe, CFileTxt

クラスメソッド

属性	
Handle	ファイルハンドルを取得します。
Filename	ファイル名を取得します。
Flags	ファイルフラグを取得します。
SetUnicode	FILE_UNICODE フラグを設定/クリアします。
SetCommon	FILE_COMMON フラグを設定/クリアします。
一般的なファイルメソッド	
Open	ファイルを開きます。
Close	ファイルを閉じます。
Delete	ファイルを削除します。
IsExist	ファイルの存在をチェックします。
Copy	ファイルを複製します。
Move	ファイルを名前変更/移動します。
Size	ファイルサイズを取得します。
Tell	現在のファイル位置を取得します。

© 2000-2025, MetaQuotes Ltd.

属性	
Seek	現在のファイル位置を設定します。
Flush	データをディスクに描きだします。
IsEnding	ファイルの終わりをチェックします。
IsLineEnding	行の終わりをチェックします。
一般的なフォルダメソッド	
FolderCreate	フォルダを作成します。
FolderDelete	フォルダを削除します。
FolderClean	フォルダをクリアします。
検索メソッド	
FileFindFirst	検索を開始します。
FileFindNext	ファイルの検索を続行します。
FileFindClose	検索ハンドルを閉じます。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

Handle

開いているファイルのハンドルを取得します。

```
int  Handle()
```

戻り値

クラスインスタンスに割り当てられた開いているファイルのハンドル（割り当てられたオブジェクトが存在しない場合は -1 ）

© 2000-2025, MetaQuotes Ltd.

FileName

開いているファイルの名 称 を取 得します。

```
string  FileName()
```

戻 り値

クラスインスタンスに割 り当 てられた開 いているファイルの名 称 割 り当 てられたファイルが存 在 しない場 合 は ""

© 2000–2025, MetaQuotes Ltd.

Flags

開いているファイルのフラグを取得します。

```
int  Flags()
```

戻り値

クラスインスタンスに割り当てられた開いているファイルのフラグ

© 2000–2025, MetaQuotes Ltd.

SetUnicode

FILE_UNICODE フラグを設定/クリアします。

```
void  SetUnicode(
   bool  unicode      // 新しいフラグ値
  )
```

パラメータ

unicode

[in] FILE_UNICODE フラグの新しい値

注意事項

文字列操作の結果は FILE_UNICODE フラグに依存します。false の場合には、ANSI コードが使用されます(1バイトのシンボル)。true の場合には、UNICODE コードが使用されます(2 バイトのシンボル)。ファイルが既に開いている場合、フラグは変更出来ません。

© 2000-2025, MetaQuotes Ltd.

SetCommon

FILE_COMMON フラグを設定/クリアします。

```
void  SetCommon(
   bool  common      // 新しいフラグ値
  )
```

パラメータ

common

[in] FILE_COMMON フラグの新しい値

注意事項

FILE_COMMON フラグは現在の操作フォルダを決定します。false の場合は、ローカル端末フォルダが使用されます。true の場合は、Common Data Folder フォルダ(全端末に共通なデータフォルダ)が使用されます。ファイルが既に開いている場合、フラグは変更出来ません。

© 2000-2025, MetaQuotes Ltd.

Open

指定されたファイルを開き、成功した場合はクラスインスタンスに割り当てます。

```
int  Open(
   const string  file_name,        // ファイル名
   int           flags,            // フラグ
   short         delimiter=9       // 区切り文字
  )
```

パラメータ

file_name

[in] 開かれるファイルの名称

flags

[in] ファイルを開く際のフラグ

delimiter=9

[in] CSV ファイルの区切り文字

戻り値

開かれたファイルのハンドル

注意事項

作業フォルダは SetCommon() で設定された FILE_COMMON フラグに依存します。

© 2000–2025, MetaQuotes Ltd.

Close

クラスインスタンスに割り当てたファイルを閉じます。

```
void  Close()
```

© 2000–2025, MetaQuotes Ltd.

Delete

クラスインスタンスに割り当てたファイルを削除します。

```
void  Delete()
```

Delete

指定されたファイルを削除します。

```
void  Delete(
   const string  file_name      // ファイル名
  )
```

パラメータ

file_name

[in] 削除されるファイルの名称

注意事項

作業フォルダは SetCommon() で設定された FILE_COMMON フラグに依存します。

© 2000–2025, MetaQuotes Ltd.

IsExist

ファイルの存 在 をチェックします。

```
bool  IsExist(
   const string  file_name      // ファイル名
  )
```

パラメータ

file_name

[in] チェックされるファイルの名 称

戻 り値

ファイルが存 在 する場 合 は true

© 2000–2025, MetaQuotes Ltd.

Copy

ファイルを複製します。

```
bool  Copy(
   const string  src_name,      // コピー元のファイル名
   int           src_flag,      // フラグ
   const string  dst_name,      // 複製先のファイル名
   int           dst_flags      // フラグ
  )
```

パラメータ

src_name

[in] コピー元のファイル名

src_flag

[in] コピー元のファイルのフラグ（FILE_COMMONのみが使用されます）

dst_name

[in] 複製先のファイル名

dst_flags

[in] 複製先のファイルのフラグ（FILE_REWRITE及びFILE_COMMONが使用されます）

戻り値

成功の場合はtrue、複製されなかった場合はfalse

© 2000-2025, MetaQuotes Ltd.

Move

ファイルを名前変更/移動します。

```
bool  Move(
   const string  src_name,      // コピー元のファイル名
   int           src_flag,      // フラグ
   const string  dst_name,      // 複製先のファイル名
   int           dst_flags      // フラグ
  )
```

パラメータ

src_name

[in] 移動されるファイルの名称

src_flag

[in] コピー元のファイルのフラグ（FILE_COMMONのみが使用されます）

dst_name

[in] 複製先のファイル名

dst_flags

[in] 複製先のファイルのフラグ（FILE_REWRITE及びFILE_COMMONが使用されます）

戻り値

成功の場合はtrue、ファイルが移動されなかった場合はfalse

© 2000-2025, MetaQuotes Ltd.

Size

ファイルサイズをバイト単位で取得します。

```
ulong  Size()
```

戻り値

バイト単位のファイルサイズ割り当てられたファイルが存在しない場合は ULONG_MAX

© 2000–2025, MetaQuotes Ltd.

Tell

現在のファイル位置を取得します。

```
ulong  Tell()
```

戻り値

現在のファイル位置割り当てられたファイルが存在しない場合は ULONG_MAX

© 2000–2025, MetaQuotes Ltd.

Seek

現在のファイル位置を設定します。

```
void  Seek(
   long                offset,     // オフセット
   ENUM_FILE_POSITION  origin      // 元
  )
```

パラメータ

offset

[in] バイト単位のファイルオフセット（負も可）

origin

[in] オフセットの元

戻り値

成功の場合は true、位置が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Flush

全てのファイル出入力データをディスクに書き出します。

```
void  Flush()
```

© 2000-2025, MetaQuotes Ltd.

IsEnding

ファイルの終わりをチェックします。ファイル読み込み作業中に使用されます。

```
bool  IsEnding()
```

戻り値

読み込みまたは検索操作の後でファイルの終わりに届いた場合は true

© 2000-2025, MetaQuotes Ltd.

IsLineEnding

行の終わりをチェックします。ファイル読み込み作業中に使用されます。

```
bool  IsLineEnding()
```

戻り値

txt または csv ファイルの読み込みの後で行の終わりに届いた場合は true（CR-LF 文字）

© 2000-2025, MetaQuotes Ltd.

FolderCreate

新しいフォルダを作成します。

```
bool  FolderCreate(
   const string  folder_name      // フォルダ名
  )
```

パラメータ

folder_name

[in] 作成されるフォルダの名称FILE_COMMONフラグで定義されたフォルダとの相対パスが含まれます。

戻り値

成功の場合はtrue、フォルダが作成できなかった場合はfalse

注意事項

作業フォルダはSetCommon()で設定されたFILE_COMMONフラグに依存します。

© 2000-2025, MetaQuotes Ltd.

FolderDelete

指定されたフォルダを削除します。

```
bool  FolderDelete(
   const string  folder_name      // フォルダ名
  )
```

パラメータ

folder_name

[in] 削除されるフォルダの名称FILE_COMMONフラグで定義されたフォルダとの相対パスが含まれます。

戻り値

成功の場合はtrue、フォルダが削除できなかった場合はfalse

注意事項

作業フォルダはSetCommon()で設定されたFILE_COMMONフラグに依存します。

© 2000–2025, MetaQuotes Ltd.

FolderClean

指定されたフォルダを削除します。

```
bool  FolderClean(
   const string  folder_name      // フォルダ名
  )
```

パラメータ

folder_name

[in] 削除されるフォルダの名称FILE_COMMONフラグで定義されたフォルダとの相対パスが含まれます。

戻り値

成功の場合はtrue、フォルダが削除できなかった場合はfalse

注意事項

作業フォルダはSetCommon()で設定されたFILE_COMMONフラグに依存します。

© 2000-2025, MetaQuotes Ltd.

FileFindFirst

指定されたフィルタを使用してファイル検索を始めます。

```
int  FileFindFirst(
   const string  filter,         // 検索フィルタ
   string&       file_name       // 文字列の参照
  )
```

パラメータ

filter

[in] 検索フィルタ。

file_name

[out] 初めに見つかったファイルの文字列参照

戻り値

If successful, it returns the handle that can be used for further file search using FileFindNext, or it returns INVALID_HANDLE if there isn't any file corresponding to the filter specified.

注意事項

作業フォルダは SetCommon() で設定された FILE_COMMON フラグに依存します。

© 2000–2025, MetaQuotes Ltd.

FileFindNext

FileFindFirst() 関数で始められたファイル検索を続けます。

```
bool  FileFindNext(
   int       search_handle,     // 検索ハンドル
   string&   file_name          // 次に見つかったファイルの文字列参照
  )
```

パラメータ

search_handle

[in] FileFindFirst() メソッドに返された検索ハンドル

file_name

[in] 成功の場合、次に見つかったファイル名の文字列参照

戻り値

成功した場合は true、指定されたフィルタに対応するファイルが存在しない場合は false

© 2000–2025, MetaQuotes Ltd.

FileFindClose

検索ハンドルを閉じます。

```
void  FileFindClose(
   int  search_handle      // 検索ハンドル
  )
```

パラメータ

search_handle

[in] FileFindFirst() メソッドに返された検索ハンドル

© 2000–2025, MetaQuotes Ltd.

CFileBin

CFileBin はバイナリファイルへの簡易化されたアクセスを可能にするクラスです。

説明

CFileBin クラスはバイナリファイルへのアクセスを提供します。

宣言

```
class CFileBin: public CFile
```

タイトル

```
#include <Files\FileBin.mqh>
```

継承階層

CObject
  CFile
    CFileBin

クラスメソッド

オープンメソッド	
Open	バイナリファイルを開きます。
書き込みメソッド	
WriteChar	char または uchar 型の変数を書き込みます。
WriteShort	short または ushort 型の変数を書き込みます。
WriteInteger	int または uint 型の変数を書き込みます。
WriteLong	long または ulong 型の変数を書き込みます。
WriteFloat	float 型の変数を書き込みます。
WriteDouble	double 型の変数を書き込みます。
WriteString	string 型の変数を書き込みます。
WriteCharArray	char または uchar 型の変数の配列を書き込みます。
WriteShortArray	short または ushort 型の変数の配列を書き込みます。
WriteIntegerArray	int または uint 型の変数の配列を書き込みます。
WriteLongArray	long または ulong 型の変数の配列を書き込みます。
WriteFloatArray	float 型の変数の配列を書き込みます。
WriteDoubleArray	double 型の変数の配列を書き込みます。
WriteObject	CObject 継承クラスのインスタンスのデータを書き込みます。

© 2000–2025, MetaQuotes Ltd.

オープンメソッド	
読み込みメソッド	
ReadChar	char または uchar 型の変数を読み込みます。
ReadShort	short または ushort 型の変数を読み込みます。
ReadInteger	int または uint 型の変数を読み込みます。
ReadLong	long または ulong 型の変数を読み込みます。
ReadFloat	float 型の変数を読み込みます。
ReadDouble	double 型の変数を読み込みます。
ReadString	string 型の変数を読み込みます。
ReadCharArray	char または uchar 型の変数の配列を読み込みます。
ReadShortArray	short または ushort 型の変数の配列を読み込みます。
ReadIntegerArray	int または uint 型の変数の配列を読み込みます。
ReadLongArray	long または ulong 型の変数の配列を読み込みます。
ReadFloatArray	float 型の変数の配列を読み込みます。
ReadDoubleArray	double 型の変数の配列を読み込みます。
ReadObject	CObject 継承クラスのインスタンスのデータを読み込みます。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

クラスから継承されたメソッド CFile

Handle, FileName, Flags, SetUnicode, SetCommon, Open, Close, Delete, Size, Tell, Seek, Flush, IsEnding, IsLineEnding, Delete, IsExist, Copy, Move, FolderCreate, FolderDelete, FolderClean, FileFindFirst, FileFindNext, FileFindClose

© 2000–2025, MetaQuotes Ltd.

Open

指定されたバイナリファイルを開いて、成功した場合クラスインスタンスに割り当てます。

```
int  Open(
   const string  file_name,      // ファイル名
   int           flags           // フラグ
  )
```

パラメータ

file_name

[in] 開かれるファイルの名称

flags

[in] ファイルを開く際のフラグ（FILE_BIN フラグは強制されます）

戻り値

開かれたファイルのハンドル

© 2000-2025, MetaQuotes Ltd.

WriteChar

char または uchar 型の変数をファイルに書き込みます。

```
uint  WriteChar(
   char  value      // 値
  )
```

パラメータ

value

[in] 書かれる変数

戻り値

書かれたバイト数

© 2000–2025, MetaQuotes Ltd.

WriteShort

short または ushort 型の変数をファイルに書き込みます。

```
uint  WriteShort(
   short  value      // 値
  )
```

パラメータ

value

[in] 書かれる変数

戻り値

書かれたバイト数

© 2000–2025, MetaQuotes Ltd.

WriteInteger

int または uint 型の変数をファイルに書き込みます。

```
uint  WriteInteger(
   int  value      // 値
  )
```

パラメータ

value

[in] 書かれる変数

戻り値

書かれたバイト数

© 2000–2025, MetaQuotes Ltd.

WriteLong

long または ulong 型の変数をファイルに書き込みます。

```
uint  WriteLong(
   long  value      // 値
  )
```

パラメータ

value

[in] 書かれる変数

戻り値

書かれたバイト数

© 2000–2025, MetaQuotes Ltd.

WriteFloat

float 型の変数をファイルに書き込みます。

```
uint  WriteFloat(
   float  value      // 値
  )
```

パラメータ

value

[in] 書かれる変数

戻り値

書かれたバイト数

© 2000–2025, MetaQuotes Ltd.

WriteDouble

double 型の変数をファイルに書き込みます。

```
uint  WriteDouble(
   double  value      // 値
  )
```

パラメータ

value

[in] 書かれる変数

戻り値

書かれたバイト数

© 2000–2025, MetaQuotes Ltd.

WriteString

string 型の変数をファイルに書き込みます。

```
uint  WriteString(
   const string  value      // 値
  )
```

パラメータ

value

[in] 書かれる文字列

戻り値

書かれたバイト数

WriteString

string 型の変数をファイルに書き込みます。

```
uint  WriteString(
   const string  value,     // 値
   int           size       // サイズ
  )
```

パラメータ

value

[in] 書かれる文字列

size

[in] 書かれるバイト数

戻り値

書かれたバイト数

© 2000–2025, MetaQuotes Ltd.

WriteCharArray

char または uchar 型の変数の配列をファイルに書き込みます。

```
uint  WriteCharArray(
   char&  array[],            // 配列への参照
   int    start_item=0,       // 開始要素
   int    items_count=-1      // 要素数
  )
```

パラメータ

array[]

[in] 書かれる配列

start_item=0

[in] 書き込みが始まる要素

items_count=-1

[in] 書かれる要素の数（-1 – 配列全体）

戻り値

書かれたバイト数

© 2000–2025, MetaQuotes Ltd.

WriteShortArray

short または ushort 型の変数の配列をファイルに書き込みます。

```
uint  WriteShortArray(
   short&  array[],              // 書かれる配列
   int     start_item=0,         // 開始要素
   int     items_count=-1        // 書かれる要素数
  )
```

パラメータ

array[]

[in] 書かれる配列

start_item=0

[in] 書き込みが始まる要素

items_count=-1

[in] 書かれる要素の数（-1 – 配列全体）

戻り値

書かれたバイト数

© 2000–2025, MetaQuotes Ltd.

WriteIntegerArray

int または uint 型の変数の配列をファイルに書き込みます。

```
uint  WriteIntegerArray(
   int&  array[],             // 書かれる配列
   int   start_item=0,        // 開始要素
   int   items_count=-1       // 書かれる要素数
  )
```

パラメータ

array[]

[in] 書かれる配列

start_item=0

[in] 書き込みが始まる要素

items_count=-1

[in] 書かれる要素の数（-1 – 配列全体）

戻り値

書かれたバイト数

© 2000–2025, MetaQuotes Ltd.

WriteLongArray

long または ulong 型の変数の配列をファイルに書き込みます。

```
uint  WriteLongArray(
   long&  array[],             // 書かれる配列
   int    start_item=0,        // 開始要素
   int    items_count=-1       // 書かれる要素数
  )
```

パラメータ

array[]

[in] 書かれる配列

start_item=0

[in] 書き込みが始まる要素

items_count=-1

[in] 書かれる要素の数（-1 – 配列全体）

戻り値

書かれたバイト数

© 2000–2025, MetaQuotes Ltd.

WriteFloatArray

float 型の変数の配列をファイルに書き込みます。

```
uint  WriteFloatArray(
   float&  array[],              // 書かれる配列
   int     start_item=0,         // 開始要素
   int     items_count=-1        // 書かれる要素数
  )
```

パラメータ

array[]

[in] 書かれる配列

start_item=0

[in] 書き込みが始まる要素

items_count=-1

[in] 書かれる要素の数（-1 – 配列全体）

戻り値

書かれたバイト数

© 2000–2025, MetaQuotes Ltd.

WriteDoubleArray

double 型の変数の配列をファイルに書き込みます。

```
uint  WriteDoubleArray(
   double&  array[],             // 書かれる配列
   int      start_item=0,        // 開始要素
   int      items_count=-1       // 書かれる要素数
  )
```

パラメータ

array[]

[in] 書かれる配列

start_item=0

[in] 書き込みが始まる要素

items_count=-1

[in] 書かれる要素の数（-1 – 配列全体）

戻り値

書かれたバイト数

© 2000–2025, MetaQuotes Ltd.

WriteObject

CObject 継承クラスのインスタンスのデータをファイルに書き込みます。

```
bool  WriteObject(
   CObject*  object      // オブジェクトへの参照
  )
```

パラメータ

object

[in] 書き込む CObject 継承クラスのインスタンスへの参照

戻り値

成功の場合は true、データが書き込まれなかった場合は false

© 2000-2025, MetaQuotes Ltd.

ReadChar

char または uchar 型の変数をファイルから読み込みます。

```
bool  ReadChar(
   char&  value      // 読まれる変数
  )
```

パラメータ

value

[in] char 型の読まれる変数

戻り値

成功の場合は true、データが読み込まれなかった場合は false

© 2000-2025, MetaQuotes Ltd.

ReadShort

short または ushort 型の変数をファイルから読み込みます。

```
bool  ReadShort(
   short&  value
  )
```

パラメータ

value

[in] short または ushort 型の読まれる変数

戻り値

成功の場合は true、データが読み込まれなかった場合は false

© 2000–2025, MetaQuotes Ltd.

ReadInteger

int または uint 型の変数をファイルから読み込みます。

```
bool  ReadInteger(
   int&  value      // 読まれる変数
  )
```

パラメータ

value

[in] int または uint 型の読み込まれる変数

戻り値

成功の場合は true、データが読み込まれなかった場合は false

© 2000–2025, MetaQuotes Ltd.

ReadLong

long または ulong 型の変数をファイルから読み込みます。

```
bool  ReadLong(
   long&  value
  )
```

パラメータ

value

[in] long または ulong 型の読まれる変数

戻り値

成功の場合は true、データが読み込まれなかった場合は false

© 2000–2025, MetaQuotes Ltd.

ReadFloat

float 型の変数をファイルから読み込みます。

```
bool  ReadFloat(
   float&  value      // 読まれる変数
  )
```

パラメータ

value

[in] float 型の読まれる変数

戻り値

成功の場合は true、データが読み込まれなかった場合は false

© 2000–2025, MetaQuotes Ltd.

ReadDouble

double 型の変数を読み込みます。

```
bool  ReadDouble(
   double&  value
  )
```

パラメータ

value

[in] double 型の読まれる変数

戻り値

成功の場合は true、データが読み込まれなかった場合は false

© 2000–2025, MetaQuotes Ltd.

ReadString

string 型の変数をファイルから読み込みます。

```
bool  ReadString(
   string&  value      // 読まれる文字列
  )
```

パラメータ

value

[in] string 型の読まれる変数

戻り値

成功の場合は true、データが読み込まれなかった場合は false

ReadString

string 型の変数をファイルから読み込みます。

```
bool  ReadString(
   string&  value
  )
```

パラメータ

value

[in] string 型の読まれる変数

戻り値

成功の場合は true、データが読み込まれなかった場合は false

© 2000–2025, MetaQuotes Ltd.

ReadCharArray

char または uchar 型の変数の配列をファイルから読み込みます。

```
bool  ReadCharArray(
   char&  array[],              // 読まれる配列
   int    start_item=0,         // 開始要素
   int    items_count=-1        // 読まれる要素数
  )
```

パラメータ

array[]

[in] 読まれる char または uchar 型の変数の配列への参照

start_item=0

[in] 読み込みが始まる要素

items_count=-1

[in] 読み込まれる要素の数（-1 – ファイルの終わりまで読み込む）

戻り値

成功の場合は true、データが読み込まれなかった場合は false

© 2000–2025, MetaQuotes Ltd.

ReadShortArray

short または ushort 型の変数の配列をファイルから読み込みます。

```
bool  ReadShortArray(
   short&  array[],              // 読まれる配列
   int     start_item=0,         // 開始要素
   int     items_count=-1        // 読まれる要素数
  )
```

パラメータ

array[]

[in] Reference to the target array of type short or ushort.

start_item=0

[in] 読み込みが始まる要素

items_count=-1

[in] 読み込まれる要素の数（-1 – ファイルの終わりまで読み込む）

戻り値

成功の場合は true、データが読み込まれなかった場合は false

© 2000–2025, MetaQuotes Ltd.

ReadIntegerArray

int または uint 型の変数の配列をファイルから読み込みます。

```
bool  ReadIntegerArray(
   int&  array[],             // 読まれる配列
   int   start_item=0,        // 開始要素
   int   items_count=-1       // 読まれる要素数
  )
```

パラメータ

array[]

[in] 読まれる int または uint 型の変数の配列への参照

start_item=0

[in] 読み込みが始まる要素

items_count=-1

[in] 読み込まれる要素の数（-1 – ファイルの終わりまで読み込む）

戻り値

成功の場合は true、データが読み込まれなかった場合は false

© 2000–2025, MetaQuotes Ltd.

ReadLongArray

long または ulong 型の変数の配列をファイルから読み込みます。

```
bool  ReadLongArray(
   long&  array[],              // 読まれる配列
   int    start_item=0,         // 開始要素
   int    items_count=-1        // 読まれる要素数
  )
```

パラメータ

array[]

[in] 読まれる long または ulong 型の変数の配列への参照

start_item=0

[in] 読み込みが始まる要素

items_count=-1

[in] 読み込まれる要素の数（-1 – ファイルの終わりまで読み込む）

戻り値

成功の場合は true、データが読み込まれなかった場合は false

© 2000–2025, MetaQuotes Ltd.

ReadFloatArray

float 型の変数の配列をファイルから読み込みます。

```
bool  ReadFloatArray(
   float&  array[],              // 読まれる配列
   int     start_item=0,         // 開始要素
   int     items_count=-1        // 読まれる要素数
  )
```

パラメータ

array[]

[in] 読まれる float 型の変数の配列への参照

start_item=0

[in] 読み込みが始まる要素

items_count=-1

[in] 読み込まれる要素の数（-1 – ファイルの終わりまで読み込む）

戻り値

成功の場合は true、データが読み込まれなかった場合は false

© 2000–2025, MetaQuotes Ltd.

ReadDoubleArray

double 型の変数の配列をファイルから読み込みます。

```
bool  ReadDoubleArray(
   double&  array[],              // 読まれる配列
   int      start_item=0,         // 開始要素
   int      items_count=-1        // 読まれる要素数
  )
```

パラメータ

array[]

[in] 読まれる double 型の変数の配列への参照

start_item=0

[in] 読み込みが始まる要素

items_count=-1

[in] 読み込まれる要素の数（-1 – ファイルの終わりまで読み込む）

戻り値

成功の場合は true、データが読み込まれなかった場合は false

© 2000–2025, MetaQuotes Ltd.

ReadObject

CObject 継承クラスのインスタンスのデータをファイルから読み込みます。

```
bool  ReadObject(
   CObject*  object       // オブジェクトポインタ
  )
```

パラメータ

object

[in] 読み込まれる CObject 継承クラスのインスタンスへの参照

戻り値

成功の場合は true、データが読み込まれなかった場合は false

© 2000-2025, MetaQuotes Ltd.

CFileTxt

CFileTxt はテキスト ファイルへの簡 易 化 されたアクセスを可 能 にするクラスです。

説 明

CFileTxt クラスはテキスト ファイルへのアクセスを提 供 します。

宣 言

```
class CFileTxt: public CFile
```

タイト ル

```
#include <Files\FileTxt.mqh>
```

継 承 階 層

COBject
CFile
CFileTxt

クラスメソッド

オープンメソッド	
Open	テキスト ファイルを開きます。
書 き込 みメソッド	
WriteString	string 型 の変 数 をファイルに書 き込 みます。
読 み込 みメソッド	
ReadString	string 型 の変 数 をファイルから読 み込 みます。

クラスから継 承 されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

クラスから継 承 されたメソッド CFile

Handle, FileName, Flags, SetUnicode, SetCommon, Open, Close, Delete, Size, Tell, Seek, Flush, IsEnding, IsLineEnding, Delete, IsExist, Copy, Move, FolderCreate, FolderDelete, FolderClean, FileFindFirst, FileFindNext, FileFindClose

© 2000–2025, MetaQuotes Ltd.

Open

指定されたテキストファイルを開いて、成功した場合クラスインスタンスに割り当てます。

```
int  Open(
   const string  file_name,      // ファイル名
   int           flags           // フラグ
  )
```

パラメータ

file_name

[in] 開かれるファイルの名称

flags

[in] ファイルを開く際のフラグ（FILE_TXT フラグは強制されます）

戻り値

開かれたファイルのハンドル

© 2000–2025, MetaQuotes Ltd.

WriteString

string 型の変数をファイルに書き込みます。

```
uint  WriteString(
   const string  value      // 書かれる文字列
  )
```

パラメータ

value

[in] 書かれる文字列

戻り値

書かれたバイト数

© 2000–2025, MetaQuotes Ltd.

ReadString

string 型の変数をファイルから読み込みます。

```
string  ReadString()
```

戻り値

読み込まれた文字列

© 2000–2025, MetaQuotes Ltd.

文字列操作

このセクションには、文字列操作のクラスの操作に関する技術的な詳細と関連するMQL5 標準ライブラリコンポーネントの説明が含まれています。

文字列操作クラスの使用は、ファイル入出力操作を使用するアプリケーションの開発時に時間を節約します。

MQL5 標準ライブラリの文字列操作に関連したファイルは、端末の作業ディレクトリの Include¥Strings フォルダ内に配置されています。

クラス	説明
CString	文字列操作クラス

© 2000–2025, MetaQuotes Ltd.

CString

CString は string 型の変数への簡易化されたアクセスを可能にするクラスです。

説明

CString クラスは派生クラスの MQL5 API 文字列関数への容易化されたアクセスを提供します。

宣言

```
class CString: public CObject
```

タイトル

```
#include <Strings\String.mqh>
```

継承階層

CObject
  CString

クラスメソッド

データアクセスメソッド	
Str	文字列を取得します。
Len	文字列の長さを取得します。
Copy	文字列を複製します。
書き込みメソッド	
Fill	文字列全体に指定された文字を書き込みます。
代入	文字列を代入します。
Append	文字列を追加します。
Insert	文字列を挿入します。
比較メソッド	
Compare	文字列を比較します。
CompareNoCase	大文字と小文字を区別しないで文字列を比較します。
サブストリングメソッド	
Left	文字列の左側から指定された数の文字を取得します。
Right	文字列の右側から指定された数の文字を取得します。
Mid	文字列から指定された数の文字を取得します。
トリム/削除メソッド	

© 2000-2025, MetaQuotes Ltd.

データアクセスメソッド	
Trim	文字列の先頭と末尾から指定された文字のセットの全てを削除します。
TrimLeft	文字列の先頭から指定された文字のセットの全てを削除します。
TrimRight	文字列の末尾から指定された文字のセットの全てを削除します。
Clear	文字列をクリアします。
変換メソッド	
ToUpper	文字列を大文字に変換します。
ToLower	文字列を小文字に変換します。
リバース。	文字列を反転します。
検索メソッド	
Find	サブストリングの最初のマッチを検索します。
FindRev	サブストリングの最後のマッチを検索します。
Remove	文字列からサブストリングを削除します。
Replace	サブストリングを置き換えます。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type

© 2000–2025, MetaQuotes Ltd.

Str

文字列を取得します。

```
string  Str() const;
```

戻り値

文字列のコピー

© 2000–2025, MetaQuotes Ltd.

Len

文字列の長さを取得します。

```
uint  Len() const;
```

戻り値

文字列の長さ

© 2000–2025, MetaQuotes Ltd.

Copy

文字列を参照によって複製します。

```
void  Copy(
   string&  copy      // 参照
   ) const;
```

パラメータ

copy

[in] 複製する文字列への参照

Copy

文字列を CString クラスインスタンスに複製します。

```
void  Copy(
   CString*  copy      // オブジェクト記述子
   ) const;
```

パラメータ

copy

[in] CString クラスオブジェクト記述子

© 2000–2025, MetaQuotes Ltd.

Fill

文字列全体に指定された文字を書き込みます。

```
bool  Fill(
   short  character      // 文字
  )
```

パラメータ

character

[in] 文字列全体に書き込む文字

戻り値

成功の場合は true、文字列が全体的に書き込まれなかった場合は false

© 2000–2025, MetaQuotes Ltd.

代入

文字列を代入します。

```
void  Assign(
   const string  str      // 代入する文字列
  )
```

パラメータ

str

[in] 代入する文字列

代入

文字列を CString クラスインスタンスに代入します。

```
void  Assign(
   CString*  str      // オブジェクト記述子
  )
```

パラメータ

str

[in] 代入する CString クラスオブジェクト記述子

© 2000–2025, MetaQuotes Ltd.

Append

文字列を追加します。

```
void  Append(
   const string  str       // 追加する文字列
  )
```

パラメータ

str

[in] 追加する文字列

Append

文字列を CString クラスインスタンスに追加します。

```
void  Append(
   CString*  string       // オブジェクト記述子
  )
```

パラメータ

string

[in] 追加する CString クラスオブジェクト記述子

© 2000–2025, MetaQuotes Ltd.

Insert

指定された位置に文字列を挿入します。

```
uint  Insert(
   uint           pos,      // 位置
   const string   str       // 挿入する文字列
  )
```

パラメータ

pos

[in] 挿入位置

str

[in] 挿入する文字列

戻り値

結果の文字列の長さ

Insert

指定された位置で CString クラスインスタンスに文字列を挿入します。

```
uint  Insert(
   uint       pos,      // 位置
   CString*   str       // オブジェクト記述子
  )
```

パラメータ

pos

[in] 挿入位置

str

[in] 挿入する CString クラスオブジェクト記述子

戻り値

結果の文字列の長さ

© 2000-2025, MetaQuotes Ltd.

Compare

文字列を比較します。

```
int  Compare(
   const string  str       // 比較される文字列
   ) const;
```

パラメータ

str

[in] 比較される文字列

戻り値

文字列が同一な場合は 0、クラス文字列が比較される文字列より低い場合は -1、クラス文字列が比較される文字列より高い場合は 1

Compare

文字列を CString クラスインスタンス文字列と比較します。

```
int  Compare(
   CString*  str       // オブジェクト記述子
   ) const;
```

パラメータ

str

[in] 比較される CString クラスオブジェクト記述子

戻り値

文字列が同一な場合は 0、クラス文字列が比較される文字列より低い場合は -1、クラス文字列が比較される文字列より高い場合は 1

© 2000-2025, MetaQuotes Ltd.

CompareNoCase

大文字と小文字を区別しないで文字列を比較します。

```
int  CompareNoCase(
   const string  str      // 比較される文字列
   ) const;
```

パラメータ

str

[in] 比較される文字列

戻り値

文字列が同一な場合は 0、クラス文字列が比較される文字列より低い場合は -1、クラス文字列が比較される文字列より高い場合は 1

CompareNoCase

大文字と小文字を区別しないで文字列を CString クラスインスタンス文字列と比較します。

```
int  CompareNoCase(
   CString*  str      // オブジェクト記述子
   ) const;
```

パラメータ

str

[in] 比較される CString クラスオブジェクト記述子

戻り値

文字列が同一な場合は 0、クラス文字列が比較される文字列より低い場合は -1、クラス文字列が比較される文字列より高い場合は 1

© 2000-2025, MetaQuotes Ltd.

Left

文字列の左側から指定された数の文字を取得します。

```
string  Left(
   uint  count       // 文字数
  )
```

パラメータ

count

[in] 文字数

戻り値

結果サブストリング

© 2000–2025, MetaQuotes Ltd.

Right

文字列の右側から指定された数の文字を取得します。

```
string  Right(
   uint  count       // 文字数
  )
```

パラメータ

count

[in] 文字数

戻り値

結果サブストリング

© 2000–2025, MetaQuotes Ltd.

Mid

文字列から指定された数の文字を取得します。

```
string  Mid(
   uint  pos,        // 位置
   uint  count       // 文字数
  )
```

パラメータ

pos

[in] 文字列の位置

count

[in] 文字数

戻り値

結果サブストリング

© 2000–2025, MetaQuotes Ltd.

Trim

文字列の先頭と末尾から指定された文字のセットの全て（及び' '、'¥t'、'¥r'、'¥n'）を削除します。

```
int  Trim(
   const string  targets      // 削除する文字のセット
  )
```

パラメータ

targets

[in] 削除する文字のセット

戻り値

削除された文字の数

例：

```
//--- CString::Trim の例
#include <Strings\String.mqh>
//---
void OnStart()
  {
   CString str;
   //---
   str.Assign("   \t\tABCD\r\n");
   printf("Source string '%s'",str.Str());
   //---
   str.Trim("DA-DA-DA");
   printf("Result string '%s'",str.Str());
  }
```

© 2000-2025, MetaQuotes Ltd.

TrimLeft

文字列の先頭から指定された文字のセットの全て（及び' '、'¥t'、'¥r'、'¥n'）を削除します。

```
int  TrimLeft(
   const string  targets      // 削除する文字のセット
  )
```

パラメータ

targets

[in] 削除する文字のセット

戻り値

削除された文字の数

© 2000–2025, MetaQuotes Ltd.

TrimRight

文字列の先頭から指定された文字のセットの全て（及び' '、'¥t'、'¥r'、'¥n'）を削除します。

```
int  TrimRight(
   const string  targets      // 削除する文字のセット
  )
```

パラメータ

targets

[in] 削除する文字のセット

戻り値

削除された文字の数

© 2000–2025, MetaQuotes Ltd.

Clear

文字列をクリアします。

```
bool  Clear()
```

戻り値

成功の場合は true、文字列がクリア出来なかった場合は false

© 2000–2025, MetaQuotes Ltd.

ToUpper

文字列を大文字に変換します。

```
bool  ToUpper()
```

戻り値

成功の場合は true、文字列が変換出来なかった場合は false

© 2000–2025, MetaQuotes Ltd.

ToLower

文字列を小文字に変換します。

```
bool  ToLower()
```

戻り値

成功の場合は true、文字列が変換出来なかった場合は false

© 2000–2025, MetaQuotes Ltd.

リバース。

文字列を反転します。

```
void  Reverse()
```

© 2000-2025, MetaQuotes Ltd.

Find

サブストリングの最初のマッチを検索します。

```
int  Find(
   uint           start,          // 位置
   const string   substring       // 検索するサブストリング
   ) const;
```

パラメータ

start

[in] 検索が開始される文字列内のインデックス。初めから検索するには 0。

substring

[in] 検索するサブストリング

戻り値

リクエストされたサブストリングの初めの文字のインデックス。サブストリングが見つからなかった場合は -1

© 2000-2025, MetaQuotes Ltd.

FindRev

サブストリングの最後のマッチを検索します。

```
int  FindRev(
   const string  substring      // サブストリング
   ) const;
```

パラメータ

substring

[in] 検索するサブストリング

戻り値

リクエストされたサブストリングの最後の文字のインデックス。サブストリングが見つからなかった場合は -1

© 2000-2025, MetaQuotes Ltd.

Remove

文字列からサブストリングを削除します。

```
uint  Remove(
   const string  substring      // 削除するサブストリング
  )
```

パラメータ

substring

[in] 検索するサブストリング

戻り値

削除されたサブストリングの数

© 2000–2025, MetaQuotes Ltd.

Replace

文字列内でサブストリングを置換します。

```
uint  Replace(
   const string  substring,      // 置き換えられるサブストリング
   const string  newstring       // 新しいサブストリング
  )
```

パラメータ

substring

[in] 検索するサブストリング

newstring

[in] 置き換えに使用されるサブストリング

戻り値

置き換えられたサブストリングの数

© 2000–2025, MetaQuotes Ltd.

グラフィックオブジェクト

このセクションには、グラフィックオブジェクトのクラスの操作に関する技術的な詳細と関連するMQL5 標準ライブラリコンポーネントの説明が含まれています。

グラフィックオブジェクトのクラスの使用は、カスタムプログラム(スクリプト、エキスパート)の作成時に時間を節約します。

MQL5 標準ライブラリのグラフィックオブジェクトに関連したファイルは、端末の作業ディレクトリのInclude¥ChartObjectsフォルダ内に配置されています。

クラス/グループ	説明
CChartObject	グラフィックオブジェクトの基本クラス
ライン	「Lines(線)」グループクラス
チャンネル	「Channels」グループクラス
ギャンツール	「Gann」グループクラス
フィボナッチツール	「Fibonacci」グループクラス
エリオットツール	「Elliott」グループクラス
図形	「Shapes」グループクラス
矢印	「Arrows」グループクラス
制御	「Controls」グループクラス

© 2000-2025, MetaQuotes Ltd.

CChartObject

CChartObject は標準 MQL5 ライブラリのチャート型のグラフィックオブジェクトの基本クラスです。

説明

CChartObject クラスは派生クラスの MQL5 API 関数への容易化されたアクセスを提供します。

宣言

```
class CChartObject : public CObject
```

タイトル

```
#include <ChartObjects\ChartObject.mqh>
```

継承階層

CObject

CChartObject

直接子孫

CChartObjectArrow, CChartObjectBitmap, CChartObjectBmpLabel, CChartObjectCycles, CChartObjectElliottWave3, CChartObjectEllipse, CChartObjectFiboArc, CChartObjectFiboFan, CChartObjectFiboTimes, CChartObjectHLine, CChartObjectRectangle, CChartObjectSubChart, CChartObjectText, CChartObjectTrend, CChartObjectTriangle, CChartObjectVLine

クラスメソッド

属性	
ChartId	グラフィックオブジェクトが属するチャートの識別子を取得します。
Window	チャートがグラフィックであるウィンドウの数を取得します。
Name	グラフィックオブジェクトの名称を取得/設定します。
NumPoints	アンカーポイントの数を取得します。
代入	
Attach	グラフィックチャートを結合します。
SetPoint	アンカーポイントを設定します。
削除	
Delete	グラフィックチャートを削除します。
Detach	グラフィックチャートを解きます。
シフト	
ShiftObject	オブジェクトの相対的な動き。
ShiftPoint	オブジェクトポイントの相対的な動き。

© 2000–2025, MetaQuotes Ltd.

属性	
オブジェクトプロパティ	
Time	オブジェクトポイントの時間座標を取得/設定します。
Price	オブジェクトポイントの価格座標を取得/設定します。
Color	オブジェクトの色を取得/設定します。
Style	ラインスタイルオブジェクトを取得/設定します。
Width	線オブジェクトの幅を取得/設定します。
BackGround	背景オブジェクト描画のフラグを取得/設定します。
Selected	オブジェクトの「selected」フラグを取得/設定します。
Selectable	フラグ設定が可能なオブジェクトを取得/設定します。
Description	オブジェクトテキストを取得/設定します。
Tooltip	オブジェクトのツールヒントを取得/設定します。
Timeframes	オブジェクト可視フラグのマスクを取得/設定します。
Z_Order	チャートクリックの優先順位を取得/設定します。
CreateTime	オブジェクト作成時刻を取得します。
オブジェクトのレベルプロパティ	
LevelsCount	オブジェクトのレベル数を取得/設定します。
LevelColor	ラインレベルの色を取得/設定します。
LevelStyle	ラインスタイルレベルを取得/設定します。
LevelWidth	ラインレベルの幅を取得/設定します。
LevelValue	レベルを取得/設定します。
LevelDescription	テキストレベルを取得/設定します。
MQL5 API 関数へのアクセス	
GetInteger	オブジェクトプロパティの値を取得します。
SetInteger	オブジェクトプロパティを設定します。
GetDouble	オブジェクトプロパティの値を取得します。
SetDouble	オブジェクトプロパティを設定します。
GetString	オブジェクトプロパティの値を取得します。
SetString	オブジェクトプロパティを設定します。
入出力	
virtual Save	ファイルエントリの仮想メソッド。

© 2000–2025, MetaQuotes Ltd.

属性	
virtual Load	ファイル読み込みの仮想メソッド。
virtual Type	識別の仮想メソッド。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

© 2000–2025, MetaQuotes Ltd.

ChartId

グラフィックオブジェクトが属するチャートの識別子を取得します。

```
long  ChartId() const
```

戻り値

グラフィックオブジェクトが属するチャートの識別子（オブジェクトが存在しない場合は -1 ）

例：

```
//--- CChartObject::ChartId の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- チャートオブジェクトのチャート識別子を取得する
   long chart_id=object.ChartId();
  }
```

© 2000-2025, MetaQuotes Ltd.

Window

グラフィックオブジェクト が属 するウィンド ウのインデックスを取 得します。

```
int  Window() const
```

戻り値

グラフィックオブジェクト が配 置されているチャート ウィンド ウの番 号（ 0 はメインウィンド ウ）（オブジェクト が存在しない場 合 は -1 ）

例 :

```
//--- CChartObject::Window の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- チャートオブジェクトのウィンド ウを取得する
   int window=object.Window();
  }
```

© 2000-2025, MetaQuotes Ltd.

Name（Get メソッド）

グラフィックオブジェクトの名称を取得します。

```
string  Name() const
```

戻り値

クラスインスタンスに結合されているグラフィックオブジェクトの名称（オブジェクトが見つからなかった場合はNULL）

Name（Set メソッド）

グラフィックオブジェクトの名称を設定します。

```
bool  Name(
   string  name      // 新しい名称
  )
```

パラメータ

name

[in] グラフィックオブジェクトの新しい名称

戻り値

成功の場合はtrue、名称が変更できなかった場合はfalse

例：

```
//--- CChartObject::Name の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- チャートオブジェクトの名称を取得する
   string object_name=object.Name();
   if(object_name!="MyChartObject")
     {
     //--- チャートオブジェクトの名称を設定する
     object.Name("MyChartObject");
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

NumPoints

グラフィックオブジェクト のアンカーポイント の数 を取 得します。

```
int  NumPoints() const
```

戻り値

クラスインスタンスに結 合 されているグラフィックオブジェクト をリンクするポイント の数（結 合 されたオブジェクト が存 在しない場 合 は 0 ）

例 :

```
//--- CChartObject::NumPoints の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- チャートオブジェクトのポイント数を取得する
   int points=object.NumPoints();
  }
```

© 2000–2025, MetaQuotes Ltd.

Attach

グラフィックオブジェクトをクラスインスタンスに接続します。

```
bool  Attach(
   long     chart_id,      // チャート識別子
   string   name,          // オブジェクトの名称
   int      window,        // チャートウィンドウ
   int      points         // ポイント数
  )
```

パラメータ

chart_id

[out] チャート識別子

name

[in] グラフィックオブジェクトの名称

window

[in] チャートウィンドウの番号（0はメインウィンドウ）

points

[in] グラフィックオブジェクトのアンカーポイントの数

戻り値

成功の場合はtrue、エラーが発生した場合はfalse

例：

```
//--- CChartObject::Attach の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- チャートオブジェクトを接続する
   if(!object.Attach(ChartID(),"MyObject",0,2))
     {
      printf("Object attach error");
      return;
     }
  }
```

© 2000–2025, MetaQuotes Ltd.

SetPoint

グラフィックオブジェクトのアンカーポイントの新しい座標を設定します。

```
bool  SetPoint(
   int       point,          // ポイント数
   datetime  new_time,       // 時間座標
   double    new_price       // 価格座標
  )
```

パラメータ

point

[in] アンカーポイントの番号

new_time

[in] 指定されたアンカーポイントの新しい時間座標の値

new_price

[in] 指定されたアンカーポイントの新しい価格座標の値

戻り値

成功の場合は true、座標が変更できなかった場合は false

例:

```
//--- CChartObject::SetPoint の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   double       price;
   //---
   if(object.NumPoints()>0)
     {
      //--- チャートオブジェクトのポイントを設定する
      object.SetPoint(0,CurrTime(),price);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

チャート からグラフィックオブジェクト を削 除します。

```
bool  Delete()
```

戻り値

成功の場合は true、削除が失敗した場合は false

例 :

```
//--- CChartObject::Delete の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- チャートオブジェクトを切り離す
   if(!object.Delete())
     {
      printf("Object delete error");
      return;
     }
  }
```

© 2000–2025, MetaQuotes Ltd.

Detach

グラフオブジェクトを切り離します。

```
void  Detach()
```

戻り値

なし

例：

```
//--- CChartObject::Detach の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- グラフオブジェクトを切り離す
   object.Detach();
  }
```

© 2000–2025, MetaQuotes Ltd.

ShiftObject

グラフィックオブジェクトを移動します。

```
bool  ShiftObject(
   datetime  d_time,       // 時間座標の移動距離
   double    d_price       // 価格座標の移動距
  )
```

パラメータ

d_time

[in] 全てのアンカーポイントの時間座標の移動距離

d_price

[in] 全てのアンカーポイントの価格座標の移動距離

戻り値

成功の場合は true、オブジェクトがシフトできなかった場合は false

例:

```
//--- CChartObject::ShiftObject の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   datetime     d_time;
   double       d_price;
   //--- チャートオブジェクトをシフトする
   object.ShiftObject(d_time,d_price);
  }
```

© 2000-2025, MetaQuotes Ltd.

ShiftPoint

グラフィックオブジェクトの指定されたアンカーポイントをシフトします。

```
bool  ShiftPoint(
   int       point,       // ポイントの番号
   datetime  d_time,      // 時間座標の移動距離
   double    d_price      // 価格座標の移動距
  )
```

パラメータ

point

[in] アンカーポイントの番号

d_time

[in] 指定されたポイントの時間座標の移動距離

d_price

[in] 指定されたポイントの価格座標の移動距離

戻り値

成功の場合は true、ポイントがシフトできなかった場合は false

例:

```
//--- CChartObject::ShiftPoint の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   datetime     d_time;
   double       d_price;
   //---
   if(object.NumPoints()>0)
     {
      //--- チャートオブジェクトのポイントをシフトする
      object.ShiftPoint(0,d_time,d_price);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Time（Get メソッド）

グラフィックオブジェクトの指定されたアンカーポイントの時間座標を取得します。

```
datetime  Time(
   int  point      // ポイント数
   ) const
```

パラメータ

point

[in] アンカーポイントの番号

戻り値

クラスインスタンスに結合されたグラフィックオブジェクトの指定されたアンカーポイントの時間座標（結合されたオブジェクトが存在しない場合やオブジェクトがポイントを所有しない場合は 0）

Time（Set メソッド）

グラフィックオブジェクトの指定されたアンカーポイントの時間座標を設定します。

```
bool  Time(
   int       point,        // ポイントの番号
   datetime  new_time      // 時間
  )
```

パラメータ

point

[in] アンカーポイントの番号

new_time

[in] 指定されたアンカーポイントの新しい時間座標の値

戻り値

成功の場合は true、時間座標が変更できなかった場合は false

例：

```
//--- CChartObject::Time の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.NumPoints();i++)
     {
      //--- チャートオブジェクトポイントの時間を取得する
      datetime point_time=object.Time(i);
      if(point_time==0)
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          //--- チャートオブジェクトポイントの時間を設定する
          object.Time(i,TimeCurrent());
         }
      }
   }
```

© 2000-2025, MetaQuotes Ltd.

Price（Get メソッド）

グラフィックオブジェクトの指定されたアンカーポイントの価格座標を取得します。

```
double  Price(
   int  point      // ポイント数
   ) const
```

パラメータ

point

[in] アンカーポイントの番号

戻り値

クラスインスタンスに結合されたグラフィックオブジェクトの指定されたアンカーポイントの価格座標（結合されたオブジェクトが存在しない場合やオブジェクトがポイントを所有しない場合は EMPTY_VALUE ）

Price（Set メソッド）

グラフィックオブジェクトの指定されたアンカーポイントの価格座標を設定します。

```
bool  Price(
   int     point,          // ポイント数
   double  new_price       // 価格
  )
```

パラメータ

point

[in] アンカーポイントの番号

new_price

[in] 指定されたアンカーポイントの新しい価格座標の値

戻り値

成功の場合は true、価格座標が変更できなかった場合は false

例：

```
//--- CChartObject::Price の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   double       price;
   //---
   for(int i=0;i<object.NumPoints();i++)
     {
      //--- チャートオブジェクトポイントの価格を取得する
      double point_price=object.Price(i);
```

© 2000-2025, MetaQuotes Ltd.

```
      if(point_price!=price)
        {
         //--- チャートオブジェクトポイントの価格を設定する
         object.Price(i,price);
        }
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Color（Get メソッド）

グラフィックオブジェクトの線の色を取得します。

```
color  Color() const
```

戻り値

クラスインスタンスに結合されたグラフィックオブジェクトの線の色（結合されたオブジェクトが存在しない場合は CLR_NONE）

Color（Set メソッド）

グラフィックオブジェクトの線の色を設定します。

```
bool  Color(
   color  new_color      // 新しい色
  )
```

パラメータ

new_color

[in] 線の色の新しい値.

戻り値

成功の場合は true、色が変更できなかった場合は false

例：

```
//--- CChartObject::Color の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- チャートオブジェクトの色を取得する
   color object_color=object.Color();
   if(object_color!=clrRed)
     {
     //--- チャートオブジェクトの色を設定する
     object.Color(clrRed);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Style（Get メソッド）

グラフィックオブジェクトの線のスタイルを取得します。

```
ENUM_LINE_STYLE  Style() const
```

戻り値

クラスインスタンスに結合されたグラフィックオブジェクトの線のスタイル（結合されたオブジェクトが存在しない場合は WRONG_VALUE）

Style（Set メソッド）

グラフィックオブジェクトの線のスタイルを設定します。

```
bool  Style(
   ENUM_LINE_STYLE  new_style      // スタイル
  )
```

パラメータ

new_style

[in] 線のスタイルの新しい値

戻り値

成功の場合は true、スタイルが変更できなかった場合は false

例：

```
//--- CChartObject::Style の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- チャートオブジェクトのスタイルを取得する
   ENUM_LINE_STYLE style=object.Style();
   if(style!=STYLE_SOLID)
     {
      //--- チャートオブジェクトのスタイルを設定する
      object.Style(STYLE_SOLID);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Width（Get メソッド）

グラフィックオブジェクトの線の幅を取得します。

```
int  Width() const
```

戻り値

クラスインスタンスに結合されたグラフィックオブジェクトの線の幅（結合されたオブジェクトが存在しない場合は -1 ）

Width（Set メソッド）

グラフィックオブジェクトの線の幅を設定します。

```
bool  Width(
   int  new_width      // 幅
  )
```

パラメータ

new_width

[in] 線の幅の新しい値

戻り値

成功の場合は true、幅が変更できなかった場合は false

例：

```
//--- CChartObject::Width の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- チャートオブジェクトの幅を取得する
   int width=object.Width();
   if(width!=1)
     {
      //--- チャートオブジェクトの幅を設定する
      object.Width(1);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Background（Get メソッド）

背景オブジェクト描画のフラグを取得します。

```
bool  Background() const
```

戻り値

クラスインスタンスに結合されているグラフィックオブジェクトを背景に描画するためのフラグ（結合されたオブジェクトが存在しない場合は false ）

Background（Set メソッド）

背景オブジェクト描画のフラグを設定します。

```
bool  Background(
   bool  background      // フラグ値
  )
```

パラメータ

background

[in] 背景オブジェクト描画のフラグの新しい値

戻り値

成功の場合は true、フラグが変更できなかった場合は false

例：

```
//--- CChartObject::Background の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- チャートオブジェクトの背景フラグを取得する
   bool background_flag=object.Background();
   if(!background_flag)
     {
     //--- チャートオブジェクトの背景フラグを設定する
     object.Background(true);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Selected（Get メソッド）

グラフィックオブジェクトの選択状態を示すフラグを取得します。言い換えると、グラフィックオブジェクトが選択されているかどうかのフラグです。

```
bool  Selected() const
```

戻り値

クラスインスタンスに結合されているグラフィックオブジェクトの選択状態（結合されたオブジェクトが存在しない場合は false ）

Selected（Set メソッド）

グラフィックオブジェクトの選択状態を示すフラグを設定します。

```
bool  Selected(
   bool  selected      // フラグ値
  )
```

パラメータ

selected

[in] フラグの新しい値

戻り値

成功の場合は true、フラグが変更できなかった場合は false

例：

```
//--- CChartObject::Selected の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- チャートオブジェクトの「selected」フラグを取得する
   bool selected_flag=object.Selected();
   if(selected_flag)
     {
     //--- チャートオブジェクトの「selected」フラグを設定する
     object.Selected(false);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Selectable（Get メソッド）

グラフィックオブジェクトの選択能力を示すフラグを取得します。言い換えると、グラフィックオブジェクトが選択出来るかどうかのフラグです。

```
bool  Selectable() const
```

戻り値

クラスインスタンスに結合されているグラフィックオブジェクトの選択能力を示すフラグ（結合されたオブジェクトが存在しない場合は false ）

Selectable（Set メソッド）

グラフィックオブジェクトの選択能力を示すフラグを選択します。

```
bool  Selectable(
   bool  selectable      // フラグ値
  )
```

パラメータ

selectable

[in] フラグの新しい値

戻り値

成功の場合は true、フラグが変更できなかった場合は false

例：

```
//--- CChartObject::Selectable の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- チャートオブジェクトの「selectable」フラグを取得する
   bool selectable_flag=object.Selectable();
   if(selectable_flag)
     {
     //--- チャートオブジェクトの「selectable」フラグを設定する
     object.Selectable(false);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Description（Get メソッド）

グラフィックオブジェクトの記述を取得します。

```
string  Description() const
```

戻り値

クラスインスタンスに結合されているグラフィックオブジェクトの記述（テキスト）（結合されたオブジェクトが存在しない場合は NULL）

Description（Set メソッド）

グラフィックオブジェクトの記述を設定します。

```
bool  Description(
   string  text      // テキスト
  )
```

パラメータ

text

[in] 新しい記述（テキスト）

戻り値

成功の場合は true、記述（テキスト）が変更できなかった場合は false

例：

```
//--- CChartObject::Description の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- チャートオブジェクトの記述を取得する
   string description=object.Description();
   if(description=="")
     {
      //--- チャートオブジェクトの記述を設定する
      object.Description("MyObject");
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Tooltip（Get メソッド）

グラフィックオブジェクト のツールヒント のテキスト を取 得します。

```
string  Tooltip() const
```

戻り値

クラスインスタンスに結 合 されたグラフィックオブジェクト のツールヒント のテキスト（結 合 されたオブジェクト が存 在しない場 合 は NULL ）

Tooltip（Set メソッド）

グラフィックオブジェクト のツールヒント のテキスト を設 定します。

```
bool  Tooltip(
   string  new_tooltip       // 新しいツールヒントのテキスト
  )
```

パラメータ

new_tooltip

[in] 新しいツールヒント のテキスト

戻り値

成 功 の場 合 は true、ツールヒント が変 更 できなかった場 合 は false

Note:

プロパティが設 定されていない場 合 は、端 末 によって自 動 的 に生 成されたツールヒント が表 示されます。ツールヒント は「￥n」（改 行 ）の値 を設 定することによって無 効 に出 来 ます。

© 2000–2025, MetaQuotes Ltd.

Timeframes（Get メソッド）

グラフィックオブジェクトの可視性フラグを取得します。

```
int  Timeframes() const
```

戻り値

クラスインスタンスに結合されているグラフィックオブジェクトの可視性フラグ（結合されたオブジェクトが存在しない場合は 0）

Timeframes（Set メソッド）

グラフィックオブジェクトの可視性フラグを設定します。

```
bool  Timeframes(
   int  new_timeframes      // 可視性フラグ
  )
```

パラメータ

new_timeframes

[in] グラフィックオブジェクトの新しい可視性フラグ

戻り値

成功の場合は true、可視性フラグが変更できなかった場合は false

例：

```
//--- CChartObject::Timeframes の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- チャートオブジェクトの時間軸を取得する
   int timeframes=object.Timeframes();
   if(!(timeframes&OBJ_PERIOD_H1))
     {
      //--- チャートオブジェクトの時間軸を設定する
      object.Timeframes(timeframes|OBJ_PERIOD_H1);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Z_Order（Get メソッド）

チャート クリック（CHARTEVENT_CLICK）の優先順位を取得します。

```
long  Z_Order() const
```

戻り値

クラスインスタンスに結合されたグラフィックオブジェクトの優先順位（結合されたオブジェクトが存在しない場合は 0）

Z_Order（Set メソッド）

チャート クリック（CHARTEVENT_CLICK）の優先順位を設定します。

```
bool  Z_Order(
   long  value      // 新しい優先順位
  )
```

パラメータ

value

[in] チャート クリック（CHARTEVENT_CLICK）の新しい優先順位

戻り値

成功の場合は true、優先順位が変更できなかった場合は false

注意事項

Z_Order はグラフィックオブジェクトのチャート クリックイベント（CHARTEVENT_CLICK）を受信する優先順位です。0（初期値）よりも大きい値を設定することにより、オブジェクトの優先順位を高めることが出来ます。

© 2000–2025, MetaQuotes Ltd.

CreateTime

グラフィックオブジェクトの作成時刻を取得します。

```
datetime  CreateTime() const
```

戻り値

Creation time of the graphical object attached to the instance of the class.（結合されたオブジェクトが存在しない場合は 0 ）

例：

```
//--- CChartObject::CreateTime の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- チャートオブジェクトの作成時刻を取得する
   datetime create_time=object.CreateTime();
  }
```

© 2000-2025, MetaQuotes Ltd.

LevelsCount (Get Method)

グラフィックオブジェクトのレベルの数を取得します。

```
int  LevelsCount() const
```

戻り値

クラスインスタンスに結合されているグラフィックオブジェクトのレベルの数（結合されたオブジェクトが存在しない場合は 0）

LevelsCount（Set メソッド）

グラフィックオブジェクトのレベルの数を設定します。

```
bool  LevelsCount(
   int  levels      // レベル数
  )
```

パラメータ

levels

[in] グラフィックオブジェクトの新しいレベルの数

戻り値

成功の場合は true、レベル数が変更できなかった場合は false

例：

```
//--- CChartObject::LevelsCount の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- チャートオブジェクトのレベル数を取得する
   int levels_count=object.LevelsCount();
   //--- チャートオブジェクトのレベル数を設定する
   object.LevelsCount(levels_count+1);
  }
```

© 2000-2025, MetaQuotes Ltd.

LevelColor（Get メソッド）

グラフィックオブジェクトの指定されたレベルの色を取得します。

```
color  LevelColor(
   int  level      // レベル番号
   ) const
```

パラメータ

level

[in] レベル番号

戻り値

クラスインスタンスに結合されたグラフィックオブジェクトの指定されたレベルの色（結合されたオブジェクトが存在しない場合は CLR_NONE ）

LevelColor（Set メソッド）

グラフィックオブジェクトの指定されたレベルの色を設定します。

```
bool  LevelColor(
   int    level,          // レベル番号
   color  new_color       // 新しい色
  )
```

パラメータ

level

[in] レベル番号

new_color

[in] 指定されたレベルの新しい線の色

戻り値

成功の場合は true、色が変更できなかった場合は false

例：

```
//--- CChartObject::LevelColor の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- チャートオブジェクトのレベルの色を取得する
      color level_color=object.LevelColor(i);
      if(level_color!=clrRed)
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          //--- チャートオブジェクトのレベルの色を設定する
          object.LevelColor(i,clrRed);
         }
      }
   }
```

© 2000-2025, MetaQuotes Ltd.

LevelStyle（Get メソッド）

グラフィックオブジェクト の指 定 されたレベルの線 のスタイルを取 得 します。

```
ENUM_LINE_STYLE  LevelStyle(
   int  level      // レベル番号
   ) const
```

パラメータ

level

[in] レベル番 号

戻 り値

クラスインスタンスに結 合 されたグラフィックオブジェクト の指 定 されたレベルの線 のスタイル（結 合 されたオブジェクト が存 在 しない場 合 や指 定 されたレベルのオブジェクト が存 在 しない場 合 は WRONG_VALUE ）

LevelStyle（Set メソッド）

グラフィックオブジェクト の指 定 されたレベルの線 のスタイルを設 定 します。

```
int  LevelStyle(
   int               level,     // レベル番号
   ENUM_LINE_STYLE   style      // 線のスタイル
  )
```

パラメータ

level

[in] レベル番 号

style

[in] 指 定 されたレベルの新 しい線 のスタイル

戻 り値

成 功 の場 合 は true、スタイルが変 更 できなかった場 合 は false

例 :

```
//--- CChartObject::LevelStyle の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- チャートオブジェクトのレベルのスタイルを取得する
      ENUM_LINE_STYLE level_style=object.LevelStyle(i);
      if(level_style!=STYLE_SOLID)
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          //--- チャートオブジェクトのレベルのスタイルを設定する
          object.LevelStyle(i,STYLE_SOLID);
         }
      }
   }
```

© 2000-2025, MetaQuotes Ltd.

LevelWidth（Get メソッド）

グラフィックオブジェクトの指定されたレベルの線の幅を取得します。

```
int  LevelWidth(
   int  level      // レベル番号
   ) const
```

パラメータ

level

[in] レベル番号

戻り値

クラスインスタンスに結合されたグラフィックオブジェクトの指定されたレベルの線の幅（結合されたオブジェクトが存在しない場合やオブジェクトが指定されたレベルを持たない場合は -1 ）

LevelWidth（Set メソッド）

グラフィックオブジェクトの指定されたレベルの線の幅を設定します。

```
bool  LevelWidth(
   int  level,          // レベル番号
   int  new_width       // 新しい幅
  )
```

パラメータ

level

[in] レベル番号

new_width

[in] 指定されたレベルの新しい線の幅

戻り値

成功の場合は true、幅が変更できなかった場合は false

例：

```
//--- CChartObject::LevelWidth の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- チャートオブジェクトのレベルの幅を取得する
      int level_width=object.LevelWidth(i);
      if(level_width!=1)
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          //--- チャートオブジェクトのレベルの幅を設定する
          object.LevelWidth(i,1);
         }
      }
   }
```

© 2000-2025, MetaQuotes Ltd.

LevelValue（Get メソッド）

グラフィックオブジェクトの指定されたレベルの値を取得します。

```
double  LevelValue(
   int  level      // レベル番号
   ) const
```

パラメータ

level

[in] レベル番号

戻り値

クラスインスタンスに結合されているグラフィックオブジェクトの指定されたレベルの値 If there is no bound object or the object has no level specified, returns EMPTY_VALUE.

LevelValue（Set メソッド）

グラフィックオブジェクトの指定されたレベルの値を設定します。

```
bool  LevelValue(
   int     level,          // レベル番号
   double  new_value       // 新しい値
  )
```

パラメータ

level

[in] レベル番号

new_value

[in] 指定されたレベルの新しい値

戻り値

成功の場合は true、値が変更できなかった場合は false

例：

```
//--- CChartObject::LevelValue の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- チャートオブジェクトのレベルの値を取得する
      double level_value=object.LevelValue(i);
      if(level_value!=0.1*i)
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          //--- チャートオブジェクトのレベルの値を設定する
          object.LevelValue(i,0.1*i);
         }
      }
   }
```

© 2000-2025, MetaQuotes Ltd.

LevelDescription（Get メソッド）

グラフィックオブジェクト のレベルの記 述 を取 得 します。

```
string  LevelDescription(
   int  level       // レベル番号
   ) const
```

パラメータ

level

[in] グラフィックオブジェクト のレベルの数

戻 り値

クラスインスタンスに結 合 されているグラフィックオブジェクト のレベルの記 述（テキスト）（結 合 されたオブジェクト が存 在 しない場 合 やオブジェクト が指 定 されたレベルを持 たない場 合 は -1 ）

LevelDescription（Set メソッド）

グラフィックオブジェクト のレベルの記 述 を設 定 します。

```
bool  LevelDescription(
   int     level ,     // レベル番号
   string  text        // テキスト
  )
```

パラメータ

level

[in] グラフィックオブジェクト のレベルの数

text

[in] グラフィックオブジェクト のレベルの記 述 の新 しい値

戻 り値

成 功 の場 合 は true、記 述（テキスト）が変 更 できなかった場 合 は false

例 :

```
//--- CChartObject::LevelDescription の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- チャートオブジェクトのレベルの記述を取得する
      string level_description=object.LevelDescription(i);
      if(level_description=="")
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          //--- チャートオブジェクトのレベルの記述を設定する
          object.LevelDescription(i,"Level_"+IntegerToString(i));
         }
      }
   }
```

© 2000–2025, MetaQuotes Ltd.

GetInteger

Provides simplified access to the functions of API MQL5 ObjectGetInteger() for properties with integer values (of type bool, char, uchar, short, ushort, int, uint, long, ulong, datetime, color) bound to an instance of the class graphic. この関数の呼び出し方は 2 つあります。

正当性をチェックしないプロパティの値の取得

```
long  GetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,          // 整数型プロパティの識別子
   int                           modifier=-1       // 修飾子
   ) const
```

パラメータ

prop_id

[in] 整数型グラフィックプロパティの識別子

modifier=-1

[in] 整数型修飾子（インデックス）

戻り値

成功した場合は整数型のプロパティの値、失敗した場合は 0

正当性をチェックするプロパティの値の取得

```
bool  GetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,       // 整数型プロパティの識別子
   int                           modifier,      // 修飾子
   long&                         value          // 変数へのリンク
   ) const
```

パラメータ

prop_id

[in] オブジェクトの整数型グラフィックプロパティの識別子

modifier

[in] 整数型修飾子（インデックス）

value

[out] 整数型プロパティの値を格納する変数への参照

戻り値

成功の場合は true、整数型プロパティが取得できなかった場合は false

例：

```
//--- CChartObject::GetInteger の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
```

© 2000–2025, MetaQuotes Ltd.

```
  {
   CChartObject object;
   //--- 簡単な方法でチャートオブジェクトの色を取得する
   printf("Objects color is %
s",ColorToString(object.GetInteger(OBJPROP_COLOR),true));
   //--- 典型的な方法でチャートオブジェクトの色を取得する
   long color_value;
   if(!object.GetInteger(OBJPROP_COLOR,0,color_value))
     {
      printf("Get integer property error %d",GetLastError());
      return;
     }
   else
      printf("Objects color is %s",color_value);
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- 簡単な方法でレベルを取得する
      printf("Level %d width is %d",i,object.GetInteger(OBJPROP_LEVELWIDTH,i));
      //--- 典型的な方法でレベルを取得する
      long width_value;
      if(!object.GetInteger(OBJPROP_LEVELWIDTH,i,width_value))
        {
         printf("Get integer property error %d",GetLastError());
         return;
        }
      else
         printf("Level %d width is %d",i,width_value);
     }
  }
```

© 2000–2025, MetaQuotes Ltd.

SetInteger

Provides simplified access to the functions of API MQL5 ObjectSetInteger() to change properties with integer values (with types bool, char, uchar, short, ushort, int, uint, long, ulong, datetime, color) bound to an instance of the class graphic.There are two versions of a function call:

修飾語なしでのプロパティ値の設定

```
bool  SetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER   prop_id,      // 整数型プロパティの識別子
   long                           value         // 値
  )
```

パラメータ

prop_id

[in] オブジェクトの整数型グラフィックプロパティの識別子

value

[in] 整数型プロパティの新しい値

修飾子を使用してのプロパティ値の設定

```
bool  SetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER   prop_id,      // 整数型プロパティの識別子
   int                            modifier,     // 修飾子
   long                           value         // 値
  )
```

パラメータ

prop_id

[in] オブジェクトの整数型グラフィックプロパティの識別子

modifier

[in] 整数型修飾子（インデックス）

value

[in] プロパティの新しい整数値

戻り値

成功の場合は true、整数型プロパティが変更できなかった場合は false

例：

```
//--- CChartObject::SetInteger の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- チャートオブジェクトの新しい色を設定する
```

© 2000–2025, MetaQuotes Ltd.

```
   if(!object.SetInteger(OBJPROP_COLOR,clrRed))
     {
      printf("Set integer property error %d",GetLastError());
      return;
     }
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- レベルの幅を設定する
      if(!object.SetInteger(OBJPROP_LEVELWIDTH,i,i))
        {
         printf("Set integer property error %d",GetLastError());
         return;
        }
     }
  }
```

© 2000–2025, MetaQuotes Ltd.

GetDouble

クラスインスタンスに割り当てられたグラフィックオブジェクトの double 型プロパティ（float 型とdouble 型）の値を取得するための API MQL5 のObjectGetDouble() 関数への簡易化したアクセスを提供します。この関数の呼び出し方は 2 つあります。

正当性をチェックしないプロパティの値の取得

```
double  GetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE  prop_id,          // double プロパティの識別子
   int                          modifier=-1      // 修飾子
   ) const
```

パラメータ

prop_id

[in] double 型グラフィックプロパティの識別子

modifier=-1

[in] double 修飾子（インデックス）

戻り値

成功した場合は double 型のプロパティの値、失敗した場合は EMPTY_VALUE

正当性をチェックするプロパティの値の取得

```
bool  GetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE  prop_id,       // double 型プロパティの識別子
   int                          modifier,      // 修飾子
   double&                      value          // 変数へのリンク
   ) const
```

パラメータ

prop_id

[in] double 型グラフィックプロパティの識別子

modifier

[in] double 修飾子（インデックス）

value

[out] double 型プロパティの値を格納する変数への参照

戻り値

成功の場合は true、double 型プロパティが取得出来なかった場合は false

例：

```
//--- CChartObject::GetDouble の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
```

© 2000–2025, MetaQuotes Ltd.

```
{
 CChartObject object;
 //---
 for(int i=0;i<object.LevelsCount();i++)
   {
    //--- 簡単な方法でレベルを取得する
    printf("Level %d value=%f",i,object.GetDouble(OBJPROP_LEVELVALUE,i));
    //--- 典型的な方法でレベルを取得する
    double value;
    if(!object.SetDouble(OBJPROP_LEVELVALUE,i,value))
      {
       printf("Get double property error %d",GetLastError());
       return;
      }
    else
       printf("Level %d value=%f",i,value);
   }
}
```

© 2000–2025, MetaQuotes Ltd.

SetDouble

クラスインスタンスに割り当てられたグラフィックオブジェクトの double 型プロパティ（float 型とdouble 型）の値を設定するための API MQL5 の ObjectSetDouble() 関数への簡易化したアクセスを提供します。この関数の呼び出し方は 2 つあります。

修飾語なしでのプロパティ値の設定

```
bool  SetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,      // double 型プロパティの識別子
   double                        value         // 値
  )
```

パラメータ

prop_id

[in] double 型グラフィックプロパティの識別子

value

[in] double 型プロパティの新しい値

修飾子を使用してのプロパティ値の設定

```
bool  SetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,      // double 型プロパティの識別子
   int                           modifier,     // 修飾子
   double                        value         // 値
  )
```

パラメータ

prop_id

[in] double 型グラフィックプロパティの識別子

modifier

[in] double 型プロパティの修飾子（インデックス）

value

[in] double 型プロパティの新しい値

戻り値

成功の場合は true、double 型プロパティが変更出来なかった場合は false

例：

```
//--- CChartObject::SetDouble の列
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
//---
```

© 2000–2025, MetaQuotes Ltd.

```
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- チャートオブジェクトのレベルの値を設定する
      if(!object.SetDouble(OBJPROP_LEVELVALUE,i,0.1*i))
        {
         printf("Set double property error %d",GetLastError());
         return;
        }
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

GetString

クラスインスタンスに割り当てられたグラフィックオブジェクトの string 型プロパティの値を取得するための API MQL5 の ObjectGetString() 関数への簡易化したアクセスを提供します。この関数の呼び出し方は 2 つあります。

正当性をチェックしないプロパティの値の取得

```
string  GetString(
   ENUM_OBJECT_PROPERTY_STRING  prop_id,          // string 型プロパティの識別子
   int                          modifier=-1       // 修飾子
   ) const
```

パラメータ

prop_id

[in] グラフィックオブジェクトの string 型プロパティの識別子

modifier=-1

[in] string 型修飾子（インデックス）

戻り値

string 型プロパティの値

正当性をチェックするプロパティの値の取得

```
bool  GetString(
   ENUM_OBJECT_PROPERTY_STRING  prop_id,       // string 型プロパティの識別子
   int                          modifier,      // 修飾子
   string&                      value          // 変数へのリンク
   ) const
```

パラメータ

prop_id

[in] グラフィックオブジェクトの string 型プロパティの識別子

modifier

[in] string 型修飾子（インデックス）

value

[out] string 型プロパティの値を格納する変数への参照

戻り値

成功の場合は true、string 型プロパティが取得出来なかった場合は false

例：

```
//--- CChartObject::GetString の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
```

© 2000–2025, MetaQuotes Ltd.

```
  {
   CChartObject object;
   string       value;
   //--- 簡単な方法でチャートオブジェクトの名称を取得する
   printf("Object name is '%s'",object.GetString(OBJPROP_NAME));
   //--- 典型的な方法でチャートオブジェクトの名称を取得する
   if(!object.GetString(OBJPROP_NAME,0,value))
     {
      printf("Get string property error %d",GetLastError());
      return;
     }
   else
      printf("Object name is '%s'",value);
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- 簡単な方法でレベルの記述を取得する
      printf("Level %d description is '%
s'",i,object.GetString(OBJPROP_LEVELTEXT,i));
      //--- 典型的な方法でレベルの記述を取得する
      if(!object.GetString(OBJPROP_LEVELTEXT,i,value))
        {
         printf("Get string property error %d",GetLastError());
         return;
        }
      else
         printf("Level %d description is '%s'",i,value);
     }
  }
```

© 2000–2025, MetaQuotes Ltd.

SetString

クラスインスタンスに割り当てられたグラフィックオブジェクトの string 型プロパティの値を変更するための API MQL5 の ObjectSetString() 関数への簡易化したアクセスを提供します。この関数の呼び出し方は 2 つあります。

修飾語なしでのプロパティ値の設定

```
bool  SetString(
   ENUM_OBJECT_PROPERTY_STRING   prop_id,      // string 型プロパティの識別子
   string                        value         // 値
  )
```

パラメータ

prop_id

[in] グラフィックオブジェクトの string 型プロパティの識別子

value

[in] string プロパティの新しい値

修飾子を使用してのプロパティ値の設定

```
bool  SetString(
   ENUM_OBJECT_PROPERTY_STRING   prop_id,      // string 型プロパティの識別子
   int                           modifier,     // 修飾子
   string                        value         // 値
  )
```

パラメータ

prop_id

[in] グラフィックオブジェクトの string 型プロパティの識別子

modifier

[in] string 型修飾子（インデックス）

value

[in] string 型プロパティの新しい値

戻り値

成功の場合は true、string 型プロパティが変更出来なかった場合は false

例：

```
//--- CChartObject::SetString の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- チャートオブジェクトの新しい名称を設定する
```

© 2000–2025, MetaQuotes Ltd.

```
   if(!object.SetString(OBJPROP_NAME,"MyObject"))
     {
      printf("Set string property error %d",GetLastError());
      return;
     }
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- レベルの説明を設定する
      if(!object.SetString(OBJPROP_LEVELTEXT,i,"Level_"+IntegerToString(i)))
        {
         printf("Set string property error %d",GetLastError());
         return;
        }
     }
  }
```

© 2000–2025, MetaQuotes Ltd.

Save

オブジェクト パラメータをファイルに保 存します。

```
virtual bool  Save(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に FileOpen(...) で開かれたファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

例:

```
//--- CChartObject::Save の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CChartObject object=new CChartObject;
   //--- オブジェクトパラメータを設定する
   //--- . にて。にて。
   //--- ファイルを開ける
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Save(file_handle))
        {
         //--- ファイル保存エラー
         printf("File save: Error %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
  }
```

© 2000–2025, MetaQuotes Ltd.

Load

オブジェクト パラメータをファイルから読み込みます。

```
virtual bool  Load(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に FileOpen(...) で開かれたファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

例:

```
//--- CChartObject::Load の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CChartObject object;
   //--- ファイルを開く
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Load(file_handle))
        {
         //--- ファイル読み込みエラー
         printf("File load: Error %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- オブジェクトを使用する
   //--- . にて。にて。
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

グラフィックオブジェクト型の識別子を取得します。

```
virtual int  Type() const
```

戻り値

オブジェクト型の識別子（0x8888 は CChartObject）

例：

```
//--- CChartObject::Type の例
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- オブジェクトの型を取得する
   int type=object.Type();
  }
```

© 2000–2025, MetaQuotes Ltd.

ラインオブジェクト

「Lines（線）」グラフィックオブジェクトのグループ

このセクションには、グラフィックオブジェクトの「Lines（線）」クラスの操作に関する技術的な詳細と関連するMQL5 標準ライブラリコンポーネントの説明が含まれています。

クラス名	オブジェクト
CChartObjectVLine	「Vertical Line（垂直線）」グラフィックオブジェクト
CChartObjectHLine	「Horizontal Line（水平線）」グラフィックオブジェクト
CChartObjectTrend	「Trend Line（トレンドライン）」グラフィックオブジェクト
CChartObjectTrendByAngle	「Trend Line by Angle（アングルトレンド）」グラフィックオブジェクト
CChartObjectCycles	「Cyclic Lines（サイクルライン）」グラフィックオブジェクト

参照

オブジェクトの型、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

CChartObjectVLine

CChartObjectVLine は「Vertical Line（垂直線）」グラフィックオブジェクトのプロパティに簡単にアクセスするためのクラスです。

説明

CChartObjectVLine クラスは「Vertical Line（垂直線）」グラフィックオブジェクトのプロパティへのアクセスを提供します。

宣言

```
class CChartObjectVLine : public CChartObject
```

タイトル

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

継承階層

CObject
  CChartObject
    CChartObjectVLine

クラスメソッド

作成	
Create	「Vertical Line（垂直線）」グラフィックオブジェクトを作成します。
入出力	
virtual Type	識別の仮想メソッド。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

参照

オブジェクトの型、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

「Vertical Line（垂直線）」グラフィックオブジェクトを作成します。

```
bool  Create(
   long       chart_id,      // チャート識別子
   string     name,          // オブジェクト名
   int        window,        // チャートウィンドウ
   datetime   time           // 時間座標
  )
```

パラメータ

chart_id

[in] チャート識別子（0は現在のチャート）

name

[in] 作成するオブジェクトの一意の名称

window

[in] チャートウィンドウの番号（0はメインウィンドウ）

time

[in] アンカーポイントの時間座標

戻り値

成功の場合はtrue、エラーが発生した場合はfalse

© 2000–2025, MetaQuotes Ltd.

Type

グラフィックオブジェクト 型 の識 別 子 を返します。

```
int  Type() const
```

戻り値

オブジェクト 型 の識 別 子（OBJ_ VLINE は CChartObjectVLine）

© 2000–2025, MetaQuotes Ltd.

CChartObjectHLine

CChartObjectHLine は「Horizontal Line（水平線）」グラフィックオブジェクトのプロパティに簡単にアクセスするためのクラスです。

説明

CChartObjectHLine クラスは「Horizontal Line（水平線）」グラフィックオブジェクトのプロパティへのアクセスを提供します。

宣言

```
class CChartObjectHLine : public CChartObject
```

タイトル

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

継承階層

CObject
  CChartObject
    CChartObjectHLine

クラスメソッド

作成	
Create	「Horizontal Line（水平線）」グラフィックオブジェクトを作成します。
入出力	
virtual Type	識別の仮想メソッド。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

参照

オブジェクトの型、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

「Horizontal Line（水平線）」グラフィックオブジェクトを作成します。

```
bool  Create(
   long     chart_id,      // チャート識別子
   string   name,          // オブジェクト名
   long     window,        // チャートウィンドウ
   double   price          // 価格座標
  )
```

パラメータ

chart_id

[in] チャート識別子（0は現在のチャート）

name

[in] 作成するオブジェクトの一意の名称

window

[in] チャートウィンドウの番号（0はメインウィンドウ）

価格

[in] アンカーポイントの価格座標

戻り値

成功の場合はtrue、エラーが発生した場合はfalse

© 2000–2025, MetaQuotes Ltd.

Type

グラフィックオブジェクト 型 の識 別 子 を返します。

```
int  Type() const
```

戻 り値

オブジェクト 型 の識 別 子（OBJ_ HLINE は CChartObjectHLine）

© 2000–2025, MetaQuotes Ltd.

CChartObjectTrend

CChartObjectTrend は「Trend Line（トレンドライン）」グラフィックオブジェクトのプロパティに簡単にアクセスするためのクラスです。

説明

CChartObjectTrend クラスは「Trend Line（トレンドライン）」グラフィックオブジェクトのプロパティへのアクセスを提供します。

宣言

```
class CChartObjectTrend : public CChartObject
```

タイトル

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

継承階層

CObject
  CChartObject
    CChartObjectTrend

直接子孫

CChartObjectChannel, CChartObjectFibo, CChartObjectFiboChannel, CChartObjectFiboExpansion, CChartObjectGannFan, CChartObjectGannGrid, CChartObjectPitchfork, CChartObjectRegression, CChartObjectStdDevChannel, CChartObjectTrendByAngle

クラスメソッド

作成	
Create	「Trend Line（トレンドライン）」グラフィックオブジェクトを作成します。
プロパティ	
RayLeft	「左に延長」プロパティを取得/設定します。
RayRight	「右に延長」プロパティを取得/設定します。
入出力	
virtual Save	ファイル書き込みの仮想メソッド。
virtual Load	ファイル読み込みの仮想メソッド。
virtual Type	識別の仮想メソッド。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CChartObject

© 2000-2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

参照

オブジェクトの型、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

「Trend Line（トレンドライン）」グラフィックオブジェクトを作成します。

```
bool  Create(
   long       chart_id,      // チャート識別子
   string     name,          // オブジェクト名
   int        window,        // チャートウィンドウ
   datetime   time1,         // 1 番目の時間座標
   double     price1,        // 1 番目の価格座標
   datetime   time2,         // 2 番目の時間座標
   double     price2         // 2 番目の価格座標
  )
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] 作成するオブジェクトの一意の名称

window

[in] チャートウィンドウの番号（0 はメインウィンドウ）

time1

[in] 1 番目のアンカーポイントの時間座標

price1

[in] 1 番目のアンカーポイントの価格座標

time2

[in] 2 番目のアンカーポイントの時間座標

price2

[in] 2 番目のアンカーポイントの価格座標

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

RayLeft（Get メソッド）

「左に延長」プロパティの値を取得します。

```
bool  RayLeft() const
```

戻り値

クラスインスタンスに割り当てられた「左に延長」プロパティの値（割り当てられたオブジェクトが存在しない場合は false）

RayLeft（Set メソッド）

「左に延長」プロパティの新しいフラグ値を設定します。

```
bool  RayLeft(
   bool  ray      // フラグ
  )
```

パラメータ

ray

[in]「左に延長」プロパティの新しい値

戻り値

成功の場合は true、フラグが変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

RayRight（Get メソッド）

「右に延長」プロパティの値を取得します。

```
bool  RayRight() const
```

戻り値

クラスインスタンスに割り当てられた「右に延長」プロパティの値（割り当てられたオブジェクトが存在しない場合は false）

RayRight（Set メソッド）

「右に延長」プロパティの新しいフラグ値を設定します。"

```
bool  RayRight(
   bool  ray      // フラグ
  )
```

パラメータ

ray

[in]「右に延長」プロパティの新しい値

戻り値

成功の場合は true、フラグが変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Save

オブジェクト パラメータをファイルに保 存します。

```
virtual bool  Save(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に FileOpen(...) で開 かれたファイルのハンド ル

戻り値

成 功 の場 合 は true、エラーが発 生した場 合 は false

© 2000-2025, MetaQuotes Ltd.

Load

オブジェクト パラメータをファイルから読み込みます。

```
virtual bool  Load(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に FileOpen(...) で開かれたファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

Type

グラフィックオブジェクト型の識別子を返します。

```
int  Type() const
```

戻り値

オブジェクト型の識別子（OBJ_TREND は CChartObjectTrend）

© 2000–2025, MetaQuotes Ltd.

CChartObjectTrendByAngle

CChartObjectTrendByAngle は「Trend Line by Angle（アングルトレンド）」グラフィックオブジェクトのプロパティに簡単にアクセスするためのクラスです。

説明

CChartObjectTrendByAngle クラスは「Trend Line by Angle（アングルトレンド）」グラフィックオブジェクトのプロパティへのアクセスを提供します。

宣言

```
class CChartObjectTrendByAngle : public CChartObjectTrend
```

タイトル

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

継承階層

CObject
CChartObject
CChartObjectTrend
CChartObjectTrendByAngle

直接子孫

CChartObjectGannLine

クラスメソッド

作成	
Create	「Trend Line by Angle（アングルトレンド）」グラフィックオブジェクトを作成します。
プロパティ	
Angle	「Angle（度）」プロパティを取得/設定します。
入出力	
virtual Type	識別の仮想メソッド。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

クラスから継承されたメソッド CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

参照

オブジェクトの型、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

「Trend Line by Angle（アングルトレンド）」グラフィックオブジェクトを作成します。

```
bool  Create(
   long       chart_id,      // チャート識別子
   string     name,          // オブジェクト名
   long       window,        // チャートウィンドウ
   datetime   time1,         // 1 番目の時間座標
   double     price1,        // 1 番目の価格座標
   datetime   time2,         // 2 番目の時間座標
   double     price2         // 2 番目の価格座標
  )
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] 作成するオブジェクトの一意の名称

window

[in] チャートウィンドウの番号（0 はメインウィンドウ）

time1

[in] 1 番目のアンカーポイントの時間座標

price1

[in] 1 番目のアンカーポイントの価格座標

time2

[in] 2 番目のアンカーポイントの時間座標

price2

[in] 2 番目のアンカーポイントの価格座標

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

Angle（Get メソッド）

「Angle（度）」プロパティの値を取得します。

```
double  Angle() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「Angle（度）」プロパティの値（割り当てられたオブジェクトが存在しない場合は EMPTY_VALUE ）

Angle（Set メソッド）

「Angle（度）」プロパティの新しい値を設定します。

```
bool  Angle(
   double  angle      // 角度
  )
```

パラメータ

angle

[in]「Angle（度）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Type

グラフィックオブジェクト 型 の識 別 子 を返します。

```
int  Type() const
```

戻り値

オブジェクト 型 の識 別 子（OBJ_ TRENDBYANGLE は CChartObjectTrendByAngle）

© 2000–2025, MetaQuotes Ltd.

CChartObjectCycles

CChartObjectCycles は「Cyclic Lines（サイクルライン）」グラフィックオブジェクトのプロパティに簡単にアクセスするためのクラスです。

説明

CChartObjectCycles クラスは「Cyclic Lines（サイクルライン）」グラフィックオブジェクトのプロパティへのアクセスを提供します。

宣言

```
class CChartObjectCycles : public CChartObject
```

タイトル

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

継承階層

CObject
  CChartObject
    CChartObjectCycles

クラスメソッド

作成	
Create	「Cyclic Lines（サイクルライン）」グラフィックオブジェクトを作成します。
入出力	
virtual Type	識別の仮想メソッド。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

参照

オブジェクトの型、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

「Cyclic Lines（サイクルライン）」グラフィックオブジェクトを作成します。

```
bool  Create(
   long      chart_id,      // チャート識別子
   string    name,          // オブジェクト名
   long      window,        // チャートウィンドウ
   datetime  time1,         // 1 番目の時間座標
   double    price1,        // 1 番目の価格座標
   datetime  time2,         // 2 番目の時間座標
   double    price2         // 2 番目の価格座標
  )
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] 作成するオブジェクトの一意の名称

window

[in] チャートウィンドウの番号（0 はメインウィンドウ）

time1

[in] 1 番目のアンカーポイントの時間座標

price1

[in] 1 番目のアンカーポイントの価格座標

time2

[in] 2 番目のアンカーポイントの時間座標

price2

[in] 2 番目のアンカーポイントの価格座標

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

Type

グラフィックオブジェクト 型 の識 別 子 を返します。

```
int  Type() const
```

戻 り値

オブジェクト 型 の識 別 子（OBJ_ CYCLES はCChartObjectCycles）

© 2000–2025, MetaQuotes Ltd.

チャンネルオブジェクト

「Channels（チャンネル）」グラフィックオブジェクトのグループ

このセクションには、グラフィックオブジェクトの「Channels（チャンネル）」クラスの操作に関する技術的な詳細と関連するMQL5 標準ライブラリコンポーネントの説明が含まれています。

クラス名	オブジェクト
CChartObjectChannel	「Equidistant Channel（等距離チャンネル）」グラフィックオブジェクト
CChartObjectRegression	「Linear Regression Channel（線形回帰チャンネル）」グラフィックオブジェクト
CChartObjectStdDevChannel	「Standard Deviation Channel（標準偏差チャンネル）」グラフィックオブジェクト
CChartObjectPitchfork	「Andrew's Pitchfork（アンドリューズピッチフォーク）」グラフィックオブジェクト

参照

オブジェクトの型、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

CChartObjectChannel

CChartObjectChannel クラスは「Equidistant Channel（等距離チャンネル）」オブジェクトのプロパティに簡単にアクセスするためのクラスです。

説明

CChartObjectChannel クラスは「Equidistant Channel（等距離チャンネル）」オブジェクトのプロパティへのアクセスを簡易化します。

宣言

```
class CChartObjectChannel : public CChartObjectTrend
```

タイトル

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

継承階層

CObject
CChartObject
CChartObjectTrend
CChartObjectChannel

クラスメソッド

作成	
Create	「Equidistant Channel（等距離チャンネル）」グラフィックオブジェクトを作成します。
入出力	
virtual Type	識別の仮想メソッド。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

クラスから継承されたメソッド CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

© 2000–2025, MetaQuotes Ltd.

参照

オブジェクトの型、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

「Equidistant Channel（等距離チャンネル）」グラフィックオブジェクトを作成します。

```
bool  Create(
   long      chart_id,      // チャート識別子
   string    name,          // オブジェクト名
   int       window,        // チャートウィンドウ
   datetime  time1,         // 1 番目のアンカーポイントの時間座標
   double    price1,        // 1 番目のアンカーポイントの価格座標
   datetime  time2,         // 2 番目のアンカーポイントの時間座標
   double    price2,        // 2 番目のアンカーポイントの価格座標
   datetime  time3,         // 3 番目のアンカーポイントの時間座標
   double    price3         // 3 番目のアンカーポイントの価格座標
  )
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] 作成するオブジェクトの一意の名称

window

[in] チャートウィンドウの番号（0 はメインウィンドウ）

time1

[in] 1 番目のアンカーポイントの時間座標

price1

[in] 1 番目のアンカーポイントの価格座標

time2

[in] 2 番目のアンカーポイントの時間座標

price2

[in] 2 番目のアンカーポイントの価格座標

time3

[in] 3 番目のアンカーポイントの時間座標

price3

[in] 3 番目のアンカーポイントの価格座標

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000-2025, MetaQuotes Ltd.

Type

グラフィックオブジェクト 型 の識 別 子 を返します。

```
int  Type() const
```

戻り値

オブジェクト 型 の識 別 子（OBJ_ CHANNEL は CChartObjectChannel）

© 2000–2025, MetaQuotes Ltd.

CChartObjectRegression

CChartObjectRegression クラスは「Linear Regression Channel（線形回帰チャンネル）」オブジェクトのプロパティに簡単にアクセスするためのクラスです。

説明

CChartObjectRegression クラスは「Linear Regression Channel（線形回帰チャンネル）」のオブジェクトプロパティへのアクセスを簡易化します。

宣言

```
class CChartObjectRegression : public CChartObjectTrend
```

タイトル

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

継承階層

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectRegression

クラスメソッド

作成	
Create	「Linear Regression Channel（線形回帰チャンネル）」グラフィックオブジェクトを作成します。
入出力	
virtual Type	識別の仮想メソッド。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

クラスから継承されたメソッド CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

© 2000–2025, MetaQuotes Ltd.

参照

オブジェクトの型、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

「Linear Regression Channel（線形回帰チャンネル）」グラフィックオブジェクトを作成します。

```
bool  Create(
   long       chart_id,      // チャート識別子
   string     name,          // オブジェクト名
   long       window,        // チャートウィンドウ
   datetime   time1,         // 1 番目の時間座標
   datetime   time2          // 2 番目の時間座標
  )
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] 作成するオブジェクトの一意の名称

window

[in] チャートウィンドウの番号（0 はメインウィンドウ）

time1

[in] 1 番目のアンカーポイントの時間座標

time2

[in] 2 番目のアンカーポイントの時間座標

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

Type

グラフィックオブジェクト型の識別子を返します。

```
int  Type() const
```

戻り値

オブジェクト型の識別子（OBJ_REGRESSION は CChartObjectRegression）

© 2000–2025, MetaQuotes Ltd.

CChartObjectStdDevChannel

CChartObjectStdDevChannel クラスは「Standard Deviation Channel（標準偏差チャンネル）」オブジェクトのプロパティに簡単にアクセスするためのクラスです。

説明

CChartObjectStdDevChannel クラスは「Standard Deviation Channel（標準偏差チャンネル）」のオブジェクトプロパティへのアクセスを簡易化します。

宣言

```
class CChartObjectStdDevChannel : public CChartObjectTrend
```

タイトル

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

継承階層

CObject
CChartObject
CChartObjectTrend
CChartObjectStdDevChannel

クラスメソッド

作成	
Create	「Standard Deviation Channel（標準偏差チャンネル）」グラフィックオブジェクトを作成します。
プロパティ	
Deviations	「Deviation（偏差）」プロパティを取得/設定します。
入出力	
virtual Save	ファイル書き込みの仮想メソッド。
virtual Load	ファイル読み込みの仮想メソッド。
virtual Type	識別の仮想メソッド。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

© 2000-2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

クラスから継承されたメソッド CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create

参照

オブジェクトの型、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

「Standard Deviation Channel（標準偏差チャンネル）」グラフィックオブジェクトを作成します。

```
bool  Create(
   long       chart_id,       // チャート識別子
   string     name,           // オブジェクト名
   int        window,         // チャートウィンドウ
   datetime   time1,          // 1 番目の時間座標
   datetime   time2,          // 2 番目の時間座標
   double     deviation       // 偏差値
  )
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] 作成するオブジェクトの一意の名称

window

[in] チャートウィンドウの番号（0 はメインウィンドウ）

time1

[in] 1 番目のアンカーポイントの時間座標

time2

[in] 2 番目のアンカーポイントの時間座標

偏差

[in]「Deviation（偏差値）」プロパティの数値

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

Deviation（Get メソッド）

「Deviation（偏差値）」プロパティの数値を取得します。

```
double  Deviation() const
```

戻り値

クラスインスタンスに割り当てられた「Deviation（偏差値）」プロパティの数値（割り当てられたオブジェクトが存在しない場合は EMPTY_VALUE ）

Deviation（Set メソッド）

「Deviation（偏差値）」プロパティの数値を設定します。

```
bool  Deviation(
   double  deviation      // 偏差値
  )
```

パラメータ

偏差

[in]「Deviation（偏差値）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Save

オブジェクト パラメータをファイルに保 存します。

```
virtual bool  Save(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に FileOpen(...) で開 かれたファイルのハンド ル

戻り値

成 功 の場 合 は true、エラーが発 生した場 合 は false

© 2000-2025, MetaQuotes Ltd.

Load

オブジェクト パラメータをファイルから読み込みます。

```
virtual bool  Load(
   int  file_handle      // ファイル ハンドル
  )
```

パラメータ

file_handle

[in] 既に FileOpen(...) で開かれたファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

Type

グラフィックオブジェクト 型 の識 別 子 を返します。

```
int  Type() const
```

戻り値

オブジェクト 型 の識 別 子（OBJ_ STDDEVCHANNEL は CChartObjectStdDevChannel）

© 2000–2025, MetaQuotes Ltd.

CChartObjectPitchfork

CChartObjectPitchfork は「Andrew's Pitchfork（アンドリューズピッチフォーク）」グラフィックオブジェクト のプロパティに簡単にアクセスするためのクラスです。

説明

CChartObjectPitchfork は「Andrew's Pitchfork（アンドリューズピッチフォーク）」グラフィックのオブジェクト プロパティへのアクセスを簡易化します。

宣言

```
class CChartObjectPitchfork : public CChartObjectTrend
```

タイトル

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

継承階層

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectPitchfork

クラスメソッド

作成	
Create	「Andrew's Pitchfork（アンドリューズピッチフォーク）」グラフィックオブジェクト を作成します。
入出力	
virtual Type	識別の仮想メソッド。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

クラスから継承されたメソッド CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

© 2000–2025, MetaQuotes Ltd.

参照

オブジェクトの型、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

「Andrew's Pitchfork（アンドリューズピッチフォーク）」グラフィックオブジェクトを作成します。

```
bool  Create(
   long      chart_id,      // チャート識別子
   string    name,          // オブジェクト名
   long      window,        // チャートウィンドウ
   datetime  time1,         // 1 番目のアンカーポイントの時間座標
   double    price1,        // 1 番目のアンカーポイントの価格座標
   datetime  time2,         // 2 番目のアンカーポイントの時間座標
   double    price2,        // 2 番目のアンカーポイントの価格座標
   datetime  time3,         // 3 番目のアンカーポイントの時間座標
   double    price3         // 3 番目のアンカーポイントの価格座標
  )
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] 作成するオブジェクトの一意の名称

window

[in] チャートウィンドウの番号（0 はメインウィンドウ）

time1

[in] 1 番目のアンカーポイントの時間座標

price1

[in] 1 番目のアンカーポイントの価格座標

time2

[in] 2 番目のアンカーポイントの時間座標

price2

[in] 2 番目のアンカーポイントの価格座標

time3

[in] 3 番目のアンカーポイントの時間座標

price3

[in] 3 番目のアンカーポイントの価格座標

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

Type

グラフィックオブジェクト 型 の識 別 子 を返します。

```
int  Type() const
```

戻り値

オブジェクト 型 の識 別 子（OBJ_ PITCHFORK は CChartObjectPitchfork）

© 2000–2025, MetaQuotes Ltd.

ギャンツール

「Gann Tools（ギャンツール）」グラフィックオブジェクトのグループ

このセクションには、グラフィックオブジェクトの「Gann Tools（ギャンツール）」クラスの操作に関する技術的な詳細と関連するMQL5 標準ライブラリコンポーネントの説明が含まれています。

クラス名	オブジェクト
CChartObjectGannLine	「ギャンライン」グラフィックオブジェクト
CChartObjectGannFan	「ギャンファン」グラフィックオブジェクト
CChartObjectGannGrid	「Gann Grid（ギャングリッド）」グラフィックオブジェクト

参照

オブジェクトの型、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

CChartObjectGannLine

CChartObjectGannLine は「Gann Line（ギャンライン）」グラフィックオブジェクト のプロパティに簡 単 にアクセスするためのクラスです。

説 明

CChartObjectGannLine クラスは「Gann Line（ギャンライン）」グラフィックオブジェクト のプロパティへのアクセスを簡 易 化します。

宣 言

```
class CChartObjectGannLine : public CChartObjectTrendByAngle
```

タイト ル

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

継 承 階 層

CObject
CChartObject
CChartObjectTrend
CChartObjectTrendByAngle
CChartObjectGannLine

クラスメソッド

作 成	
Create	「Gann Line（ギャンライン）」グラフィックオブジェクト を作 成します。
プロパティ	
PipsPerBar	「Scale（スケール）」プロパティを取 得 /設 定します。
入 出 力	
virtual Save	ファイル書き込 みの仮 想 メソッド。
virtual Load	ファイル読 み込 みの仮 想 メソッド。
virtual Type	識 別 の仮 想 メソッド。

クラスから継 承 されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継 承 されたメソッド CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor,

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

クラスから継承されたメソッド CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create

クラスから継承されたメソッド CChartObjectTrendByAngle

Angle, Angle, Create

参照

オブジェクトの型、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

「ギャンライン」グラフィックオブジェクトを作成します。

```
bool  Create(
   long       chart_id,      // チャート識別子
   string     name,          // オブジェクト名
   int        window,        // チャートウィンドウ
   datetime   time1,         // 1 番目の時間座標
   double     price1,        // 1 番目の価格座標
   datetime   time2,         // 2 番目の時間座標
   double     ppb            // バーあたりのピップ
  )
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] 作成するオブジェクトの一意の名称

window

[in] チャートウィンドウの番号（0 はメインウィンドウ）

time1

[in] 1 番目のアンカーポイントの時間座標

price1

[in] 1 番目のアンカーポイントの価格座標

time2

[in] 2 番目のアンカーポイントの時間座標

ppb

[in] バーあたりのピップ

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

PipsPerBar（Get メソッド）

「Pips per bar（ピップス/バー）」プロパティの値を取得します。

```
double  PipsPerBar() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「Pips per bar」プロパティの値（割り当てられたオブジェクトが存在しない場合は EMPTY_VALUE ）

PipsPerBar（Set メソッド）

「Pips per bar（ピップス/バー）」プロパティの新しい値を設定します。

```
bool  PipsPerBar(
   double  ppb       // バーあたりのピップ
  )
```

パラメータ

ppb

[in]「Pips per bar（ピップス/バー）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Save

オブジェクト パラメータをファイルに保 存します。

```
virtual bool  Save(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] すでにFileOpen(...)で開 かれたバイナリファイルのハンド ル

戻り値

成 功 の場 合 は true、エラーが発 生した場 合 は false

© 2000–2025, MetaQuotes Ltd.

Load

オブジェクト パラメータをファイルから読み込みます。

```
virtual bool  Load(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] すでにFileOpen(...)で開かれたバイナリファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

Type

グラフィックオブジェクトの型の識別子を取得します。

```
virtual int  Type() const
```

戻り値

オブジェクト型の識別子（OBJ_GANNLINEはCChartObjectGannLine）

© 2000-2025, MetaQuotes Ltd.

CChartObjectGannFan

CChartObjectGannFan は「Gann Fan（ギャンファン）」グラフィックオブジェクトのプロパティに簡単にアクセスするためのクラスです。

説明

CChartObjectGannFan クラスは「Gann Fan（ギャンファン）」グラフィックオブジェクトのプロパティへのアクセスを簡易化します。

宣言

```
class CChartObjectGannFan : public CChartObjectTrend
```

タイトル

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

継承階層

CObject
CChartObject
CChartObjectTrend
CChartObjectGannFan

クラスメソッド

作成	
Create	「Gann Fan（ギャンファン）」グラフィックオブジェクトを作成します。
プロパティ	
PipsPerBar	「Pips per bar（ピップス/バー）」プロパティの値を取得/設定します。
Downtrend	「Downtrend（下降トレンド）」プロパティの値を取得/設定します。
入出力	
virtual Save	ファイル書き込みの仮想メソッド。
virtual Load	ファイル読み込みの仮想メソッド。
virtual Type	識別の仮想メソッド。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor,

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

クラスから継承されたメソッド CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create

参照

オブジェクトの型、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

「ギャンファン」グラフィックオブジェクトを作成します。

```
bool  Create(
   long       chart_id,      // チャート識別子
   string     name,          // オブジェクト名
   int        window,        // チャートウィンドウ
   datetime   time1,         // 1 番目の時間座標
   double     price1,        // 1 番目の価格座標
   datetime   time2,         // 2 番目の時間座標
   double     ppb            // バーあたりのピップ
  )
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] 作成するオブジェクトの一意の名称

window

[in] チャートウィンドウの番号（0 はメインウィンドウ）

time1

[in] 1 番目のアンカーポイントの時間座標

price1

[in] 1 番目のアンカーポイントの価格座標

time2

[in] 2 番目のアンカーポイントの時間座標

ppb

[in] バーあたりのピップ

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

PipsPerBar（Get メソッド）

「Pips per bar（ピップス/バー）」プロパティの値を取得します。

```
double  PipsPerBar() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「Pips per bar」プロパティの値（割り当てられたオブジェクトが存在しない場合は EMPTY_VALUE ）

PipsPerBar（Set メソッド）

「Pips per bar（ピップス/バー）」プロパティの新しい値を設定します。

```
bool  PipsPerBar(
   double  ppb       // バーあたりのピップ
  )
```

パラメータ

ppb

[in]「Pips per bar（ピップス/バー）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Downtrend（Get メソッド）

「downtrend（下降トレンド）」プロパティの値を取得します。

```
bool  Downtrend() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「downtrend（下降トレンド）」プロパティの値（割り当てられたオブジェクトが存在しない場合は false）

Downtrend（Set メソッド）

「downtrend（下降トレンド）」プロパティの新しい値を設定します。

```
bool  Downtrend(
   bool  downtrend      // フラグ値
  )
```

パラメータ

downtrend

[in]「downtrend（下降トレンド）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Save

オブジェクト パラメータをファイルに保 存します。

```
virtual bool  Save(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] すでにFileOpen(...)で開 かれたバイナリファイルのハンド ル

戻り値

成 功 の場 合 は true、エラーが発 生した場 合 は false

© 2000–2025, MetaQuotes Ltd.

Load

オブジェクト パラメータをファイルから読み込みます。

```
virtual bool  Load(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] すでにFileOpen(...)で開かれたバイナリファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

Type

グラフィックオブジェクトの型の識別子を取得します。

```
virtual int  Type() const
```

戻り値

オブジェクト型の識別子（OBJ_GANNFANはCChartObjectGannFan）

© 2000–2025, MetaQuotes Ltd.

CChartObjectGannGrid

CChartObjectGannGrid は「 Gann Grid（ギャングリッド）」グラフィックオブジェクト のプロパティに簡 単 にアクセスするためのクラスです。

説 明

CChartObjectGannGrid クラス は「 Gann Grid（ギャングリッド）」グラフィックオブジェクト のプロパティへのアクセスを簡 易 化します。

宣 言

```
class CChartObjectGannGrid : public CChartObjectTrend
```

タイト ル

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

継 承 階 層

CObject
CChartObject
CChartObjectTrend
CChartObjectGannGrid

クラスメソッド

作 成	
Create	「 Gann Grid（ギャングリッド）」グラフィックオブジェクト を作 成します。
プロパティ	
PipsPerBar	「Pips per bar（ピップス/バー）」プロパティの値 を取 得 /設 定します。
Downtrend	「Downtrend（下 降トレンド）」プロパティの値 を取 得 /設 定します。
入 出 力	
virtual Save	ファイル書き込 みの仮 想 メソッド。
virtual Load	ファイル読 み込 みの仮 想 メソッド。
virtual Type	識 別 の仮 想 メソッド。

クラスから継 承 されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継 承 されたメソッド CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor,

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

クラスから継承されたメソッド CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create

参照

オブジェクトの型、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

「Gann Grid（ギャングリッド）」グラフィックオブジェクトを作成します。

```
bool  Create(
   long       chart_id,      // チャート識別子
   string     name,          // オブジェクト名
   int        window,        // チャートウィンドウ
   datetime   time1,         // 1 番目の時間座標
   double     price1,        // 1 番目の価格座標
   datetime   time2,         // 2 番目の時間座標
   double     ppb            // バーあたりのピップ
  )
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] 作成するオブジェクトの一意の名称

window

[in] チャートウィンドウの番号（0 はメインウィンドウ）

time1

[in] 1 番目のアンカーポイントの時間座標

price1

[in] 1 番目のアンカーポイントの価格座標

time2

[in] 2 番目のアンカーポイントの時間座標

ppb

[in] バーあたりのピップ

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

PipsPerBar（Get メソッド）

「Pips per bar（ピップス/バー）」プロパティの値を取得します。

```
double  PipsPerBar() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「Pips per bar」プロパティの値（割り当てられたオブジェクトが存在しない場合は EMPTY_VALUE ）

PipsPerBar（Set メソッド）

「Pips per bar（ピップス/バー）」プロパティの新しい値を設定します。

```
bool  PipsPerBar(
   double  ppb       // バーあたりのピップ
  )
```

パラメータ

ppb

[in]「Pips per bar（ピップス/バー）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Downtrend（Get メソッド）

「 downtrend（下降トレンド）」プロパティの値を取得します。

```
bool  Downtrend() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「 downtrend （下降トレンド）」プロパティの値（割り当てられたオブジェクトが存在しない場合は false ）

Downtrend（Set メソッド）

「 downtrend（下降トレンド）」プロパティの新しい値を設定します。

```
bool  Downtrend(
   bool  downtrend      // フラグ値
  )
```

パラメータ

downtrend

[in]「 downtrend（下降トレンド）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Save

オブジェクト パラメータをファイルに保 存します。

```
virtual bool  Save(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] すでにFileOpen(...)で開 かれたバイナリファイルのハンド ル

戻り値

成 功 の場 合 は true、エラーが発 生した場 合 は false

© 2000–2025, MetaQuotes Ltd.

Load

オブジェクト パラメータをファイルから読み込みます。

```
virtual bool  Load(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] すでにFileOpen(...)で開かれたバイナリファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

Type

グラフィックオブジェクトの型の識別子を取得します。

```
virtual int  Type() const
```

戻り値

オブジェクト型の識別子（OBJ_GANNGRID は CChartObjectGannGrid）

© 2000–2025, MetaQuotes Ltd.

フィボナッチツール

「Fibonacci Tools（フィボナッチツール）」グラフィックオブジェクトのグループ

このセクションには、グラフィックオブジェクトの「Fibonacci Tools（フィボナッチツール）」クラスの操作に関する技術的な詳細と関連するMQL5 標準ライブラリコンポーネントの説明が含まれています。

クラス名	オブジェクト
CChartObjectFibo	「Fibonacci Retracement（フィボナッチリトレースメント）」グラフィックオブジェクト
CChartObjectFiboTimes	「Fibonacci Time Zones（フィボナッチタイムゾーン）」グラフィックオブジェクト
CChartObjectFiboFan	「Fibonacci Fan（フィボナッチファン）」グラフィックオブジェクト
CChartObjectFiboArc	「Fibonacci Arc（フィボナッチアーク）」グラフィックオブジェクト
CChartObjectFiboChannel	「Fibonacci Channel（フィボナッチチャンネル）」グラフィックオブジェクト
CChartObjectFiboExpansion	「Fibonacci Expansion（フィボナッチ拡張）」グラフィックオブジェクト

参照

オブジェクトの型、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

CChartObjectFibo

CChartObjectFibo は「Fibonacci Retracement（フィボナッチリトレースメント）」グラフィックオブジェクトのプロパティに簡単にアクセスするためのクラスです。

説明

CChartObjectFibo は「Fibonacci Retracement（フィボナッチリトレースメント）」グラフィックオブジェクトのプロパティへのアクセスを簡易化します。

宣言

```
class CChartObjectFibo : public CChartObjectTrend
```

タイトル

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

継承階層

CObject
CChartObject
CChartObjectTrend
CChartObjectFibo

クラスメソッド

作成	
Create	「Fibonacci Retracement（フィボナッチリトレースメント）」グラフィックオブジェクトを作成します。
入出力	
virtual Type	識別の仮想メソッド。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

クラスから継承されたメソッド CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

© 2000–2025, MetaQuotes Ltd.

参照

オブジェクトの型、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

「Fibonacci Retracement（フィボナッチリトレースメント）」グラフィックオブジェクトを作成します。

```
bool  Create(
   long      chart_id,      // チャート識別子
   string    name,          // オブジェクト名
   int       window,        // チャートウィンドウ
   datetime  time1,         // 1 番目の時間座標
   double    price1,        // 1 番目の価格座標
   datetime  time2,         // 2 番目の時間座標
   double    price2         // 2 番目の価格座標
  )
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] 作成するオブジェクトの一意の名称

window

[in] チャートウィンドウの番号（0 はメインウィンドウ）

time1

[in] 1 番目のアンカーポイントの時間座標

price1

[in] 1 番目のアンカーポイントの価格座標

time2

[in] 2 番目のアンカーポイントの時間座標

price2

[in] 2 番目のアンカーポイントの価格座標

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000-2025, MetaQuotes Ltd.

Type

グラフィックオブジェクトの型の識別子を取得します。

```
virtual int  Type() const
```

戻り値

オブジェクト型の識別子（OBJ_FIBOはCChartObjectFibo）

© 2000-2025, MetaQuotes Ltd.

CChartObjectFiboTimes

CChartObjectFiboTimes は「Fibonacci Time Zones（フィボナッチタイムゾーン）」グラフィックオブジェクト のプロパティに簡 単 にアクセスするためのクラスです。

説 明

CChartObjectFiboTimes クラスは「Fibonacci Time Zones（フィボナッチタイムゾーン）」グラフィックオブジェクトのプロパティへのアクセスを簡 易 化します。

宣 言

```
class CChartObjectFiboTimes : public CChartObject
```

タイト ル

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

継 承 階 層

CObject
  CChartObject
    CChartObjectFiboTimes

クラスメソッド

作 成	
Create	「Fibonacci Time Zones（フィボナッチタイムゾーン）」グラフィックオブジェクト を作成します。
入 出 力	
virtual Type	識 別 の仮 想 メソッド。

クラスから継 承 されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継 承 されたメソッド CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

参 照

オブジェクト の型 、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

「Fibonacci Time Zones（フィボナッチタイムゾーン）」グラフィックオブジェクトを作成します。

```
bool  Create(
   long       chart_id,      // チャート識別子
   string     name,          // オブジェクト名
   int        window,        // チャートウィンドウ
   datetime   time1,         // 1 番目の時間座標
   double     price1,        // 1 番目の価格座標
   datetime   time2,         // 2 番目の時間座標
   double     price2         // 2 番目の価格座標
  )
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] 作成するオブジェクトの一意の名称

window

[in] チャートウィンドウの番号（0 はメインウィンドウ）

time1

[in] 1 番目のアンカーポイントの時間座標

price1

[in] 1 番目のアンカーポイントの価格座標

time2

[in] 2 番目のアンカーポイントの時間座標

price2

[in] 2 番目のアンカーポイントの価格座標

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

Type

グラフィックオブジェクトの型の識別子を取得します。

```
virtual int  Type() const
```

戻り値

オブジェクト型の識別子（OBJ_FIBOTIMES は CChartObjectFiboTimes）

© 2000–2025, MetaQuotes Ltd.

CChartObjectFiboFan

CChartObjectFiboFan は「Fibonacci Fan（フィボナッチファン）」グラフィックオブジェクト のプロパティに簡 単 にアクセスするためのクラスです。

説 明

CChartObjectFiboFan は「Fibonacci Fan（フィボナッチファン）」グラフィックオブジェクト のプロパティへのアクセスを簡 易 化します。

宣 言

```
class CChartObjectFiboFan : public CChartObject
```

タイト ル

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

継 承 階 層

CObject
  CChartObject
    CChartObjectFiboFan

クラスメソッド

作 成	
Create	「Fibonacci Fan（フィボナッチファン）」グラフィックオブジェクト を作 成 します。
入 出 力	
virtual Type	識 別 の仮 想 メソッド。

クラスから継 承 されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継 承 されたメソッド CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

参 照

オブジェクト の型 、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

「Fibonacci Fan（フィボナッチファン）」グラフィックオブジェクトを作成します。

```
bool  Create(
   long      chart_id,      // チャート識別子
   string    name,          // オブジェクト名
   int       window,        // チャートウィンドウ
   datetime  time1,         // 1 番目の時間座標
   double    price1,        // 1 番目の価格座標
   datetime  time2,         // 2 番目の時間座標
   double    price2         // 2 番目の価格座標
  )
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] 作成するオブジェクトの一意の名称

window

[in] チャートウィンドウの番号（0 はメインウィンドウ）

time1

[in] 1 番目のアンカーポイントの時間座標

price1

[in] 1 番目のアンカーポイントの価格座標

time2

[in] 2 番目のアンカーポイントの時間座標

price2

[in] 2 番目のアンカーポイントの価格座標

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

Type

グラフィックオブジェクトの型の識別子を取得します。

```
virtual int  Type() const
```

戻り値

オブジェクト型の識別子（OBJ_FIBOFAN は CChartObjectFiboFan）

© 2000-2025, MetaQuotes Ltd.

CChartObjectFiboArc

CChartObjectFiboArc は「Fibonacci Arc（フィボナッチアーク）」グラフィックオブジェクトのプロパティに簡単にアクセスするためのクラスです。

説明

CChartObjectFiboArc は「Fibonacci Arc（フィボナッチアーク）」グラフィックオブジェクトのプロパティへのアクセスを簡易化します。

宣言

```
class CChartObjectFiboArc : public CChartObject
```

タイトル

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

継承階層

CObject
  CChartObject
    CChartObjectFiboArc

クラスメソッド

作成	
Create	「Fibonacci Arc（フィボナッチアーク）」グラフィックオブジェクトを作成します。
プロパティ	
Scale	「Scale（スケール）」プロパティを取得/設定します。
Ellipse	「Ellipse（楕円形）」プロパティを取得/設定します。
入出力	
virtual Save	ファイル書き込みの仮想メソッド。
virtual Load	ファイル読み込みの仮想メソッド。
virtual Type	識別の仮想メソッド。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

参照

オブジェクトの型、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

「Fibonacci Arc（フィボナッチアーク）」グラフィックオブジェクトを作成します。

```
bool  Create(
   long      chart_id,      // チャート識別子
   string    name,          // オブジェクト名
   int       window,        // チャートウィンドウ
   datetime  time1,         // 1 番目の時間座標
   double    price1,        // 1 番目の価格座標
   datetime  time2,         // 2 番目の時間座標
   double    price2,        // 2 番目の時間座標
   double    scale          // スケール
  )
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] 作成するオブジェクトの一意の名称

window

[in] チャートウィンドウの番号（0 はメインウィンドウ）

time1

[in] 1 番目のアンカーポイントの時間座標

price1

[in] 1 番目のアンカーポイントの価格座標

time2

[in] 2 番目のアンカーポイントの時間座標

price2

[in] 2 番目のアンカーポイントの価格座標

scale

[in] スケール

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

Scale（Get メソッド）

「Scale（スケール）」プロパティの値を取得します。

```
double  Scale() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「Scale（スケール）」プロパティの値（割り当てられたオブジェクトが存在しない場合は EMPTY_VALUE ）

Scale（Set メソッド）

「Scale（スケール）」プロパティの新しい値を設定します。

```
bool  Scale(
   double  scale      // スケール
  )
```

パラメータ

scale

[in]「Scale（スケール）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000-2025, MetaQuotes Ltd.

Ellipse（Get メソッド）

「Ellipse（楕円形）」プロパティの値を取得します。

```
bool  Ellipse() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「Ellipse（楕円形）」プロパティの値（割り当てられたオブジェクトが存在しない場合は false）

Ellipse（Set メソッド）

「Ellipse（楕円形）」プロパティの新しいフラグ値を設定します。

```
bool  Ellipse(
   bool  ellipse      // フラグ値
  )
```

パラメータ

ellipse

[in]「Scale（スケール）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Save

オブジェクト パラメータをファイルに保 存します。

```
virtual bool  Save(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] すでにFileOpen(...)で開 かれたバイナリファイルのハンド ル

戻り値

成 功 の場 合 は true、エラーが発 生した場 合 は false

© 2000–2025, MetaQuotes Ltd.

Load

オブジェクト パラメータをファイルから読み込みます。

```
virtual bool  Load(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] すでにFileOpen(...)で開かれたバイナリファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

Type

グラフィックオブジェクトの型の識別子を取得します。

```
virtual int  Type() const
```

戻り値

オブジェクト型の識別子（OBJ_FIBOARCはCChartObjectFiboArc）

© 2000–2025, MetaQuotes Ltd.

CChartObjectFiboChannel

CChartObjectFiboChannel は「 Fibonacci Channel（フィボナッチチャンネル）」グラフィックオブジェクト のプロパティに簡 単 にアクセスするためのクラスです。

説 明

CChartObjectFiboChannel は「 Fibonacci Channel（フィボナッチチャンネル）」グラフィックオブジェクト のプロパティへのアクセスを簡 易 化します。

宣 言

```
class CChartObjectFiboChannel : public CChartObjectTrend
```

タイト ル

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

継 承 階 層

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectFiboChannel

クラスメソッド

作 成	
Create	「 Fibonacci Channel（フィボナッチチャンネル）」グラフィックオブジェクト を作 成します。
入 出 力	
virtual Type	識 別 の仮 想 メソッド。

クラスから継 承 されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継 承 されたメソッド CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

クラスから継 承 されたメソッド CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

© 2000–2025, MetaQuotes Ltd.

参照

オブジェクトの型、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

「Fibonacci Channel（フィボナッチチャンネル）」グラフィックオブジェクトを作成します。

```
bool  Create(
   long      chart_id,      // チャート識別子
   string    name,          // オブジェクト名
   int       window,        // チャートウィンドウ
   datetime  time1,         // 1 番目の時間座標
   double    price1,        // 1 番目の価格座標
   datetime  time2,         // 2 番目の時間座標
   double    price2,        // 2 番目の時間座標
   datetime  time3,         // 3 番目の時間座標
   double    price3         // 3 番目の時間座標
  )
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] 作成するオブジェクトの一意の名称

window

[in] チャートウィンドウの番号（0 はメインウィンドウ）

time1

[in] 1 番目のアンカーポイントの時間座標

price1

[in] 1 番目のアンカーポイントの価格座標

time2

[in] 2 番目のアンカーポイントの時間座標

price2

[in] 2 番目のアンカーポイントの価格座標

time3

[in] 3 番目のアンカーポイントの時間座標

price3

[in] 3 番目のアンカーポイントの価格座標

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000-2025, MetaQuotes Ltd.

Type

グラフィックオブジェクトの型の識別子を取得します。

```
virtual int  Type() const
```

戻り値

オブジェクト型の識別子（OBJ_FIBOCHANNELはCChartObjectFiboChannel）

© 2000–2025, MetaQuotes Ltd.

CChartObjectFiboExpansion

CChartObjectFiboExpansion は「 Fibonacci Expansion（フィボナッチ拡張）」グラフィックオブジェクトのプロパティに簡単にアクセスするためのクラスです。

説明

CChartObjectFiboExpansion は「 Fibonacci Expansion（フィボナッチ拡張）」グラフィックオブジェクトのプロパティへのアクセスを簡易化します。

宣言

```
class CChartObjectFiboExpansion : public CChartObjectTrend
```

タイトル

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

継承階層

CObject
CChartObject
CChartObjectTrend
CChartObjectFiboExpansion

クラスメソッド

作成	
Create	「 Fibonacci Expansion（フィボナッチ拡張）」グラフィックオブジェクトを作成します。
入出力	
virtual Type	識別の仮想メソッド。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

クラスから継承されたメソッド CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

© 2000–2025, MetaQuotes Ltd.

参照

オブジェクトの型、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

「Fibonacci Expansion（フィボナッチ拡張）」グラフィックオブジェクトを作成します。

```
bool  Create(
   long      chart_id,      // チャート識別子
   string    name,          // オブジェクト名
   int       window,        // チャートウィンドウ
   datetime  time1,         // 1 番目の時間座標
   double    price1,        // 1 番目の価格座標
   datetime  time2,         // 2 番目の時間座標
   double    price2,        // 2 番目の時間座標
   datetime  time3,         // 3 番目の時間座標
   double    price3         // 3 番目の時間座標
  )
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] 作成するオブジェクトの一意の名称

window

[in] チャートウィンドウの番号（0 はメインウィンドウ）

time1

[in] 1 番目のアンカーポイントの時間座標

price1

[in] 1 番目のアンカーポイントの価格座標

time2

[in] 2 番目のアンカーポイントの時間座標

price2

[in] 2 番目のアンカーポイントの価格座標

time3

[in] 3 番目のアンカーポイントの時間座標

price3

[in] 3 番目のアンカーポイントの価格座標

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

Type

グラフィックオブジェクトの型の識別子を取得します。

```
virtual int  Type() const
```

戻り値

オブジェクト型の識別子（OBJ_EXPANSIONはCChartObjectFiboExpansion）

© 2000–2025, MetaQuotes Ltd.

エリオット ツール

「Elliott Tools（エリオット ツール）」グラフィックオブジェクト のグループ

このセクションには、グラフィックオブジェクト の「Elliott Tools（エリオット ツール）」クラスの操 作 に関 する技 術 的 な詳 細 と関 連 するMQL5 標 準 ライブラリコンポーネント の説 明 が含 まれています。

クラス名	オブジェクト
CChartObjectElliottWave3	「Correcting Wave（修 正 波 ）」グラフィックオブジェクト
CChartObjectElliottWave5	「 Impulse Wave（エリオット 推 進 波 ）」グラフィックオブジェクト

参 照

オブジェクト の型 、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

CChartObjectElliottWave3

CChartObjectElliottWave3 は「Correcting Wave（修正波）」グラフィックオブジェクトのプロパティに簡単にアクセスするためのクラスです。

説明

CChartObjectElliottWave3 は「Correcting Wave（修正波）」グラフィックオブジェクトのプロパティへのアクセスを簡易化します。

宣言

```
class CChartObjectElliottWave3 : public CChartObject
```

タイトル

```
#include <ChartObjects\ChartObjectsElliott.mqh>
```

継承階層

CObject
  CChartObject
    CChartObjectElliottWave3

直接子孫

CChartObjectElliottWave5

クラスメソッド

作成	
Create	「Correcting Wave（修正波）」グラフィックオブジェクトを作成します。
プロパティ	
Degree	「Degree（度）」プロパティを取得/設定します。
Lines	「Lines（線）」プロパティを取得/設定します。
入出力	
virtual Save	ファイル書き込みの仮想メソッド。
virtual Load	ファイル読み込みの仮想メソッド。
virtual Type	識別の仮想メソッド。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order,

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

参照

オブジェクトの型、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

「Correcting Wave（修正波）」グラフィックオブジェクトを作成します。

```
bool  Create(
   long      chart_id,      // チャート識別子
   string    name,          // オブジェクト名
   int       window,        // チャートウィンドウ
   datetime  time1,         // 1 番目の時間座標
   double    price1,        // 1 番目の価格座標
   datetime  time2,         // 2 番目の時間座標
   double    price2,        // 2 番目の時間座標
   datetime  time3,         // 3 番目の時間座標
   double    price3         // 3 番目の時間座標
  )
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] 作成するオブジェクトの一意の名称

window

[in] チャートウィンドウの番号（0 はメインウィンドウ）

time1

[in] 1 番目のアンカーポイントの時間座標

price1

[in] 1 番目のアンカーポイントの価格座標

time2

[in] 2 番目のアンカーポイントの時間座標

price2

[in] 2 番目のアンカーポイントの価格座標

time3

[in] 3 番目のアンカーポイントの時間座標

price3

[in] 3 番目のアンカーポイントの時間座標

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

Degree（Get メソッド）

「Degree（度）」プロパティの値を取得します。

```
ENUM_ELLIOT_WAVE_DEGREE  Degree() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「Degree（度）」プロパティの値（割り当てられたオブジェクトが存在しない場合は WRONG_VALUE）

Degree（Set メソッド）

「Degree（度）」プロパティの新しい値を設定します。

```
bool  Degree(
   ENUM_ELLIOT_WAVE_DEGREE  degree      // プロパティ値
  )
```

パラメータ

degree

[in]「Degree（度）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Lines（Get メソッド）

「Lines（線）」プロパティの値を取得します。

```
bool  Lines() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「Lines（線）」プロパティの値（割り当てられたオブジェクトが存在しない場合は false）

Lines（Set メソッド）

「Lines（線）」プロパティの新しい値を設定します。

```
bool  Lines(
   bool  lines      // フラグ値
  )
```

パラメータ

lines

[in]「Lines（線）」プロパティの新しい値

戻り値

成功の場合は true、フラグが変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Save

オブジェクト パラメータをファイルに保 存します。

```
virtual bool  Save(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] すでにFileOpen(...)で開 かれたバイナリファイルのハンド ル

戻り値

成 功 の場 合 は true、エラーが発 生した場 合 は false

© 2000-2025, MetaQuotes Ltd.

Load

オブジェクト パラメータをファイルから読み込みます。

```
virtual bool  Load(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] すでにFileOpen(...)で開かれたバイナリファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

Type

グラフィックオブジェクトの型の識別子を取得します。

```
virtual int  Type() const
```

戻り値

オブジェクト型の識別子（OBJ_ELLIOTWAVE3 は CChartObjectElliottWave3）

© 2000–2025, MetaQuotes Ltd.

CChartObjectElliottWave5

CChartObjectElliottWave5 は「 Impulse Wave（エリオット 推 進 波 ）」グラフィックオブジェクト のプロパティに簡 単 にアクセスするためのクラスです 。

説 明

CChartObjectElliottWave5 は「 Impulse Wave（エリオット 推 進 波 ）」グラフィックオブジェクト のプロパティへのア クセスを簡 易 化 します 。

宣 言

```
class CChartObjectElliottWave5 : public CChartObjectElliottWave3
```

タイト ル

```
#include <ChartObjects\ChartObjectsElliott.mqh>
```

継 承 階 層

CObject
  CChartObject
    CChartObjectElliottWave3
      CChartObjectElliottWave5

クラスメソッド

作 成	
Create	「 Impulse Wave（エリオット 推 進 波 ）」グラフィックオブジェクト を作 成 します 。
入 出 力	
virtual Type	識 別 の仮 想 メソッド 。

クラスから継 承 されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継 承 されたメソッド CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

クラスから継 承 されたメソッド CChartObjectElliottWave3

Degree, Degree, Lines, Lines, Create, Save, Load

参 照

© 2000–2025, MetaQuotes Ltd.

オブジェクト の型 、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

「Impulse Wave（エリオット推進波）」グラフィックオブジェクトを作成します。

```
bool  Create(
   long      chart_id,      // チャート識別子
   string    name,          // オブジェクト名
   int       window,        // チャートウィンドウ
   datetime  time1,         // 1 番目の時間座標
   double    price1,        // 1 番目の価格座標
   datetime  time2,         // 2 番目の時間座標
   double    price2,        // 2 番目の時間座標
   datetime  time3,         // 3 番目の時間座標
   double    price3,        // 3 番目の価格座標
   datetime  time4,         // 4 番目の時間座標
   double    price4,        // 4 番目の価格座標
   datetime  time5          // 5 番目の時間座標
   double    price5,        // 5 番目の価格座標
  )
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] 作成するオブジェクトの一意の名称

window

[in] チャートウィンドウの番号（0 はメインウィンドウ）

time1

[in] 1 番目のアンカーポイントの時間座標

price1

[in] 1 番目のアンカーポイントの価格座標

time2

[in] 2 番目のアンカーポイントの時間座標

price2

[in] 2 番目のアンカーポイントの価格座標

time3

[in] 3 番目のアンカーポイントの時間座標

price3

[in] 3 番目のアンカーポイントの価格座標

time4

[in] 4 番目のアンカーポイントの時間座標

© 2000–2025, MetaQuotes Ltd.

price4

[in] 4 番目のアンカーポイントの価格座標

time5

[in] 5 番目のアンカーポイントの時間座標.

price5

[in] 5 番目のアンカーポイントの価格座標

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

Type

グラフィックオブジェクトの型の識別子を取得します。

```
virtual int  Type() const
```

戻り値

オブジェクト型の識別子（OBJ_ELLIOTWAVE5 は CChartObjectElliottWave5）

© 2000–2025, MetaQuotes Ltd.

図形オブジェクト

「Shapes（図形）」グラフィックオブジェクトのグループ

このセクションには、グラフィックオブジェクトの「Shapes（図形）」クラスの操作に関する技術的な詳細と関連するMQL5 標準ライブラリコンポーネントの説明が含まれています。

クラス名	オブジェクト
CChartObjectRectangle	「Rectangle（四角形）」グラフィックオブジェクト
CChartObjectTriangle	「Triangle（三角形）」グラフィックオブジェクト
CChartObjectEllipse	「Ellipse（楕円形）」グラフィックオブジェクト

参照

オブジェクトの型、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

CChartObjectRectangle

CChartObjectRectangle は「Rectangle（四角形）」グラフィックオブジェクトのプロパティに簡単にアクセスするためのクラスです。

説明

クラスは「Rectangle（四角形）」グラフィックオブジェクトのプロパティへのアクセスを提供します。

宣言

```
class CChartObjectRectangle : public CChartObject
```

タイトル

```
#include <ChartObjects\ChartObjectsShapes.mqh>
```

継承階層

CObject
  CChartObject
    CChartObjectRectangle

クラスメソッド

作成	
Create	「Rectangle（四角形）」オブジェクトを作成します。
入出力	
virtual Type	識別の仮想メソッド。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

参照

オブジェクトの型、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

「Rectangle（四角形）」オブジェクトを作成します。

```
bool  Create(
   long       chart_id,      // チャート識別子
   string     name,          // オブジェクト名
   long       window,        // チャートウィンドウ
   datetime   time1,         // 1 番目の時間座標
   double     price1,        // 1 番目の価格座標
   datetime   time2,         // 2 番目の時間座標
   double     price2         // 2 番目の価格座標
  )
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] 作成するオブジェクトの一意の名称

window

[in] チャートウィンドウの番号（0 はメインウィンドウ）

time1

[in] 1 番目のアンカーポイントの時間座標

price1

[in] 1 番目のアンカーポイントの価格座標

time2

[in] 2 番目のアンカーポイントの時間座標

price2

[in] 2 番目のアンカーポイントの価格座標

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

Type

グラフィックオブジェクト 型 の識 別 子 を返します。

```
int  Type() const
```

戻り値

オブジェクト 型 の識 別 子（OBJ_ RECTANGLE は CChartObjectRectangle）

© 2000–2025, MetaQuotes Ltd.

CChartObjectTriangle

CChartObjectTriangle は「Triangle（三角形）」グラフィックオブジェクトのプロパティに簡単にアクセスするためのクラスです。

説明

CChartObjectTriangle クラスは「Triangle（三角形）」オブジェクトプロパティへのアクセスを提供します。

宣言

```
class CChartObjectTriangle : public CChartObject
```

タイトル

```
#include <ChartObjects\ChartObjectsShapes.mqh>
```

継承階層

CObject
  CChartObject
    CChartObjectTriangle

クラスメソッド

作成	
Create	「Triangle（三角形）」オブジェクトを作成します。
入出力	
virtual Type	識別の仮想メソッド。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

参照

オブジェクトの型、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

「Triangle（三角形）」オブジェクトを作成します。

```
bool  Create(
   long      chart_id,      // チャート識別子
   string    name,          // オブジェクト名
   long      window,        // チャートウィンドウ
   datetime  time1,         // 1 番目の時間座標
   double    price1,        // 1 番目の価格座標
   datetime  time2,         // 2 番目の時間座標
   double    price2,        // 2 番目の時間座標
   datetime  time3,         // 3 番目の時間座標
   double    price3         // 3 番目の時間座標
  )
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] 作成するオブジェクトの一意の名称

window

[in] チャートウィンドウの番号（0 はメインウィンドウ）

time1

[in] 1 番目のアンカーポイントの時間座標

price1

[in] 1 番目のアンカーポイントの価格座標

time2

[in] 2 番目のアンカーポイントの時間座標

price2

[in] 2 番目のアンカーポイントの価格座標

time3

[in] 3 番目のアンカーポイントの時間座標

price3

[in] 3 番目のアンカーポイントの価格座標

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

Type

グラフィックオブジェクト 型 の識 別 子 を返します。

```
int  Type() const
```

戻り値

オブジェクト 型 の識 別 子（OBJ_ TRIANGLE は CChartObjectTriangle）

© 2000–2025, MetaQuotes Ltd.

CChartObjectEllipse

CChartObjectEllipse は「Ellipse（楕円形）」グラフィックオブジェクトのプロパティに簡単にアクセスするためのクラスです。

説明

CChartObjectEllipse クラスは「Ellipse（楕円形）」オブジェクトプロパティへのアクセスを提供します。

宣言

```
class CChartObjectEllipse : public CChartObject
```

タイトル

```
#include <ChartObjects\ChartObjectsShapes.mqh>
```

継承階層

CObject
  CChartObject
    CChartObjectEllipse

クラスメソッド

作成	
Create	「Ellipse（楕円形）」オブジェクトを作成します。
入出力	
virtual Type	識別の仮想メソッド。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

参照

オブジェクトの型、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

「Ellipse（楕円形）」オブジェクトを作成します。

```
bool  Create(
   long       chart_id,      // チャート識別子
   string     name,          // オブジェクト名
   int        window,        // チャートウィンドウ
   datetime   time1,         // 1 番目の時間座標
   double     price1,        // 1 番目の価格座標
   datetime   time2,         // 2 番目の時間座標
   double     price2,        // 2 番目の時間座標
   datetime   time3,         // 3 番目の時間座標
   double     price3         // 3 番目の時間座標
  )
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] 作成するオブジェクトの一意の名称

window

[in] チャートウィンドウの番号（0 はメインウィンドウ）

time1

[in] 1 番目のアンカーポイントの時間座標

price1

[in] 1 番目のアンカーポイントの価格座標

time2

[in] 2 番目のアンカーポイントの時間座標

price2

[in] 2 番目のアンカーポイントの価格座標

time3

[in] 3 番目のアンカーポイントの時間座標

price3

[in] 3 番目のアンカーポイントの価格座標

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

Type

グラフィックオブジェクト 型 の識 別 子 を返します。

```
int  Type() const
```

戻 り値

オブジェクト 型 の識 別 子（OBJ_ ELLIPSE は CChartObjectEllipse）

© 2000–2025, MetaQuotes Ltd.

矢印オブジェクト

Arrows（矢印）グラフィックオブジェクトのグループ

このセクションには、グラフィックオブジェクトの「Arrows（矢印）」クラスの操作に関する技術的な詳細と関連するMQL5 標準ライブラリコンポーネントの説明が含まれています。本質的には、矢印は、特定のコードと一致するいくつかのアイコンです。チャート上にアイコンを表示する「Arrow（矢印）」グラフィックオブジェクトには2つのタイプがあります。

- コードのアイコン表示されたオブジェクトを指定を可能にする「Arrow（矢印）」オブジェクト
- アイコンの特定のタイプ（及び対応する特定の固定コード）を表示するグループオブジェクト

矢印のクラスは、任意のコードのアイコンを表示します。

クラス名	矢印オブジェクトの名称
CChartObjectArrow	矢印。

固定コードを持つ矢印のクラス

クラス名	矢印オブジェクトの名称
CChartObjectArrowCheck	チェック。
CChartObjectArrowDown	上矢印。
CChartObjectArrowUp	下矢印。
CChartObjectArrowStop	ストップサイン。
CChartObjectArrowThumbDown	サムズアップ。
CChartObjectArrowThumbUp	サムズダウン。
CChartObjectArrowLeftPrice	左プライスラベル。
CChartObjectArrowRightPrice	右プライスラベル。

参照

オブジェクトの型、オブジェクト結合の方法、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

CChartObjectArrow

CChartObjectArrow クラスは「Arrow（矢印）」グラフィックオブジェクトのプロパティに簡単にアクセスするためのクラスです。

説明

CChartObjectArrow クラスは「Arrow（矢印）」オブジェクトの共通プロパティへのアクセスを全ての派生クラスに提供します。

宣言

```
class CChartObjectArrow : public CChartObject
```

タイトル

```
#include <ChartObjects\ChartObjectsArrows.mqh>
```

継承階層

CObject
  CChartObject
    CChartObjectArrow

直接子孫

CChartObjectArrowCheck, CChartObjectArrowDown, CChartObjectArrowLeftPrice, CChartObjectArrowRightPrice, CChartObjectArrowStop, CChartObjectArrowThumbDown, CChartObjectArrowThumbUp, CChartObjectArrowUp

クラスメソッド

作成	
Create	「Arrow（矢印）」グラフィックオブジェクトを作成します。
プロパティ	
ArrowCode	「Arrow Code（矢印のコード）」プロパティを取得/設定します。
Anchor	「Anchor（アンカー）」プロパティを取得/設定します。
入出力	
virtual Save	ファイル書き込みの仮想メソッド。
virtual Load	ファイル読み込みの仮想メソッド。
virtual Type	識別の仮想メソッド。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CChartObject

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

参照

オブジェクトの型、オブジェクト結合の方法、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

「Arrow（矢印）」グラフィックオブジェクトを作成します。

```
bool  Create(
   long       chart_id,      // チャート識別子
   string     name,          // オブジェクト名
   int        window,        // チャートウィンドウ
   datetime   time,          // 時間
   double     price,         // 価格
   char       code           // 矢印コード
  )
```

パラメータ

chart_id

[in] チャート識別子（0は現在のチャート）

name

[in] 一意のオブジェクト名

window

[in] チャートウィンドウの番号（0はメインウィンドウ）

time

[in] 時間座標

価格

[in] 価格座標

code

[in]「Arrow（矢印）」コード（Wingdingsフォント）

戻り値

成功の場合はtrue、それ以外の場合はfalse

例：

```
//--- CChartObjectArrow::Create の例
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
//--- オブジェクトパラメータを設定する
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,181))
     {
      //--- 矢印作成エラー
      printf("Arrow create: Error %d!",GetLastError());
      //---
```

© 2000-2025, MetaQuotes Ltd.

```
        return;
      }
//--- 矢印を使用する
//--- . にて。にて。
   }
```

© 2000–2025, MetaQuotes Ltd.

ArrowCode（Get メソッド）

「Arrow（矢印）」のシンボルのコードを取得します。

```
char  ArrowCode() const
```

戻り値

クラスインスタンスに割り当てられた「Arrow（矢印）」オブジェクトのシンボルのコード（割り当てられたオブジェクトが存在しない場合は 0）

ArrowCode（Set メソッド）

「Arrow（矢印）」のシンボルのコードを設定します。

```
bool  ArrowCode(
   char  code      // コード値
  )
```

パラメータ

code

[in]「Arrow（矢印）」コードの新しい値（Wingdingsフォント）

戻り値

成功の場合は true、コードが変更できなかった場合は false

例：

```
//--- CChartObjectArrow::ArrowCode の例
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
   char              code=181;
//--- オブジェクトパラメータを設定する
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,code))
     {
      //--- 矢印作成エラー
      printf("Arrow create: Error %d!",GetLastError());
      //---
      return;
     }
//--- 矢印のコードを変更する
//--- . にて。にて。
//--- 矢印のコードを取得する
   if(arrow.ArrowCode()!=code)
     {
      //--- 矢印のコードを設定する
```

© 2000-2025, MetaQuotes Ltd.

```
      arrow.ArrowCode(code);
      }
//--- 矢印を使用する
//--- . にて。にて。
   }
```

© 2000–2025, MetaQuotes Ltd.

Anchor（Get メソッド）

「Arrow（矢印）」オブジェクトのアンカーを取得します。

```
ENUM_ARROW_ANCHOR  Anchor() const
```

戻り値

クラスインスタンスに割り当てられた「Arrow（矢印）」オブジェクトのアンカーの型（割り当てられたオブジェクトが存在しない場合は WRONG_VALUE）

Anchor（Set メソッド）

「Arrow（矢印）」オブジェクトのアンカーを設定します。

```
bool  Anchor(
   ENUM_ARROW_ANCHOR  anchor      // 新しいアンカー型
  )
```

パラメータ

anchor

[in] 新しいアンカー型

戻り値

成功の場合は true、アンカー型 が変更できなかった場合は false

例：

```
//--- CChartObject::Anchor の例
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
   ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM;
//--- オブジェクトパラメータを設定する
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,181))
     {
      //--- 矢印作成エラー
      printf("Arrow create: Error %d!",GetLastError());
      //---
      return;
     }
//--- 矢印のアンカーを取得する
   if(arrow.Anchor()!=anchor)
     {
     //--- 矢印のアンカーを設定する
     arrow.Anchor(anchor);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 矢印を使用する
//--- . にて。にて。
   }
```

© 2000–2025, MetaQuotes Ltd.

Save

オブジェクト パラメータをファイルに保 存します。

```
virtual bool  Save(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に FileOpen(...) で開 かれたファイルのハンド ル

戻り値

成 功 の場 合 は true、それ以 外 の場 合 は false

例 :

```
//--- CChartObjectArrow::Save の例
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   int               file_handle;
   CChartObjectArrow arrow;
//--- オブジェクトパラメータを設定する
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,181))
     {
      //--- 矢印作成エラー
      printf("Arrow create: Error %d!",GetLastError());
      //---
      return;
     }
//--- ファイルを開く
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!arrow.Save(file_handle))
        {
         //--- ファイル保存エラー
         printf("File save: Error %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
  }
```

© 2000–2025, MetaQuotes Ltd.

Load

オブジェクト パラメータをファイルから読み込みます。

```
virtual bool  Load(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] 既に FileOpen(...) で開かれたファイルのハンドル

戻り値

成功の場合は true、それ以外の場合は false

例:

```
//--- CChartObjectArrow::Load の例
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   int                file_handle;
   CChartObjectArrow arrow;
//--- ファイルを開く
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!arrow.Load(file_handle))
        {
         //--- ファイル読み込みエラー
         printf("File load: Error %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
//--- 矢印を使用する
//--- . にて。にて。
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

グラフィックオブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

オブジェクト型の識別子（例えば OBJ_ARROW は CChartObjectArrow）

例：

```
//--- CChartObjectArrow::Type の例
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
   //--- 矢印の型を取得する
   int type=arrow.Type();
  }
```

© 2000-2025, MetaQuotes Ltd.

固定コードを持つ矢印

「Arrows with fixed code（固定コードを持つ矢印）」クラスは下記のグラフィックオブジェクトプロパティへに簡単にアクセスするためのクラスです。

クラス名	矢印オブジェクト名
CChartObjectArrowCheck	「Arrow Check」（チェック矢印）
CChartObjectArrowDown	「Arrow Down」（下矢印）
CChartObjectArrowUp	「Arrow Up」（上矢印）
CChartObjectArrowStop	「Arrow Stop」（停止矢印）
CChartObjectArrowThumbDown	「いいね」（「サムズアップ」）
CChartObjectArrowThumbUp	「良くない」（「サムズダウン」）
CChartObjectArrowLeftPrice	「Left price」（左方価格）矢印
CChartObjectArrowRightPrice	「Right price」（右方価格）矢印

説明

「Arrows with fixed code（固定コードを持つ矢印）」クラスはオブジェクトプロパティへのアクセスを提供します。

宣言

```
class CChartObjectArrowCheck      : public CChartObjectArrow;
class CChartObjectArrowDown       : public CChartObjectArrow;
class CChartObjectArrowUp         : public CChartObjectArrow;
class CChartObjectArrowStop       : public CChartObjectArrow;
class CChartObjectArrowThumbDown  : public CChartObjectArrow;
class CChartObjectArrowThumbUp    : public CChartObjectArrow;
class CChartObjectArrowLeftPrice  : public CChartObjectArrow;
class CChartObjectArrowRightPrice : public CChartObjectArrow;
```

タイトル

```
<ChartObjects\ChartObjectsArrows.mqh>
```

クラスメソッド

Create	
Create	指定されたグラフィックオブジェクトを作成します。
プロパティ	
ArrowCode	コード変更メソッドの「スタブ」
入出力	

© 2000–2025, MetaQuotes Ltd.

Create	
virtual Type	識別の仮想メソッド。

参照

オブジェクトの型、オブジェクト結合の方法、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

「Arrow with fixed code（固定コードを持つ矢印）」グラフィックオブジェクトを作成します。

```
bool  Create(
   long       chart_id,      // チャート識別子
   string     name,          // オブジェクト名
   int        window,        // チャートウィンドウ
   datetime   time,          // 時間
   double     price          // 価格
  )
```

パラメータ

chart_id

[in] チャート識別子（0は現在のチャート）

name

[in] 作成するオブジェクトの一意の名称

window

[in] チャートウィンドウの番号（0はメインウィンドウ）

time

[in] 時間座標

価格

[in] 価格座標

戻り値

成功の場合はtrue、エラーが発生した場合はfalse

例：

```
//--- CChartObjectArrowCheck::Create の例
//--- CChartObjectArrowDown::Create の例
//--- CChartObjectArrowUp::Create の例
//--- CChartObjectArrowStop::Create の例
//--- CChartObjectArrowThumbDown::Create の例
//--- CChartObjectArrowThumbUp::Create の例
//--- CChartObjectArrowLeftPrice::Create の例
//--- CChartObjectArrowRightPrice::Create の例
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
//--- 例として CChartObjectArrowCheck を見る
   CChartObjectArrowCheck arrow;
//--- オブジェクトパラメータを設定する
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"ArrowCheck",0,TimeCurrent(),price))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- 矢印作成エラー
       printf("Arrow create: Error %d!",GetLastError());
       //---
       return;
      }
//--- 矢印を使用する
//--- . にて。にて。
  }
```

© 2000–2025, MetaQuotes Ltd.

ArrowCode

「Arrow（矢印）」のコード変更を禁止します。

```
bool  ArrowCode(
   char  code       // コード値
  )
```

パラメータ

code

[in] 任意の値

戻り値

常に false

例：

```
//--- CChartObjectArrowCheck::ArrowCode の例
//--- CChartObjectArrowDown::ArrowCode の例
//--- CChartObjectArrowUp::ArrowCode の例
//--- CChartObjectArrowStop::ArrowCode の例
//--- CChartObjectArrowThumbDown::ArrowCode の例
//--- CChartObjectArrowThumbUp::ArrowCode の例
//--- CChartObjectArrowLeftPrice::ArrowCode の例
//--- CChartObjectArrowRightPrice::ArrowCode の例
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
//--- 例として CChartObjectArrowCheck を見る
   CChartObjectArrowCheck arrow;
//--- オブジェクトパラメータを設定する
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"ArrowCheck",0,TimeCurrent(),price))
     {
      //--- 矢印作成エラー
      printf("Arrow create: Error %d!",GetLastError());
      //---
      return;
     }
//--- 矢印のコードを設定する
   if(!arrow.ArrowCode(181))
     {
      //--- エラーではない
      printf("Arrow code can not be changed");
     }
//--- 矢印を使用する
//--- . にて。にて。
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

グラフィックオブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

オブジェクトの型の識別子（OBJ_ARROW_CHECK は CChartObjectArrowCheck、OBJ_ARROW_DOWN は CChartObjectArrowDown、OBJ_ARROW_UP は CChartObjectArrowUp、OBJ_ARROW_STOP は CChartObjectArrowStop、OBJ_ARROW_THUMB_DOWN は CChartObjectArrowThumbDown、OBJ_ARROW_THUMB_UP は CChartObjectArrowThumbUp、OBJ_ARROW_LEFT_PRICE は CChartObjectArrowLeftPrice、OBJ_ARROW_RIGHT_PRICE は CChartObjectArrowRightPrice）

例：

```
//--- CChartObjectArrowCheck::Type の例
//--- CChartObjectArrowDown::Type の例
//--- CChartObjectArrowUp::Type の例
//--- CChartObjectArrowStop::Type の例
//--- CChartObjectArrowThumbDown::Type の例
//--- CChartObjectArrowThumbUp::Type の例
//--- CChartObjectArrowLeftPrice::Type の例
//--- CChartObjectArrowRightPrice::Type の例
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
//--- 例として CChartObjectArrowCheck を見る
   CChartObjectArrowCheck arrow;
//--- 矢印の型を取得する
   int type=arrow.Type();
  }
```

© 2000-2025, MetaQuotes Ltd.

オブジェクト制御

「Object Controls（オブジェクト制御）」グラフィックオブジェクトのグループ

このセクションには、グラフィックオブジェクトの「Object Controls（オブジェクト制御）」クラスの操作に関する技術的な詳細と関連するMQL5 標準ライブラリコンポーネントの説明が含まれています。

クラス名	オブジェクト
CChartObjectText	「Text（テキスト）」グラフィックオブジェクト
CChartObjectLabel	「Text Label（ラベル）」グラフィックオブジェクト
CChartObjectEdit	「Edit（編集）」グラフィックオブジェクト
CChartObjectButton	「Button（ボタン）」グラフィックオブジェクト
CChartObjectSubChart	「Chart（チャート）」グラフィックオブジェクト
CChartObjectBitmap	「Bitmap（ビットマップ）」グラフィックオブジェクト
CChartObjectBmpLabel	「Bitmap Label（ビットマップラベル）」グラフィックオブジェクト
CChartObjectRectLabel	「Rectangle Label（四角形ラベル）」グラフィックオブジェクト

参照

オブジェクトの型、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

CChartObjectText

CChartObjectText は「Text(テキスト)」グラフィックオブジェクトのプロパティに簡単にアクセスするためのクラスです。

説明

CChartObjectText は「Text(テキスト)」オブジェクトのプロパティへのアクセスを提供します。

宣言

```
class CChartObjectText : public CChartObject
```

タイトル

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

継承階層

CObject
CChartObject
CChartObjectText

直接子孫

CChartObjectLabel

クラスメソッド

作成	
Create	「Text(テキスト)」グラフィックオブジェクトを作成します。
プロパティ	
Angle	「Angle(度)」プロパティを取得/設定します。
Font	「Font(フォント)」プロパティを取得/設定します。
FontSize	「FontSize(フォントサイズ)」プロパティを取得/設定します。
Anchor	「Anchor(アンカー)」プロパティを取得/設定します。
入出力	
virtual Save	ファイル書き込みの仮想メソッド。
virtual Load	ファイル読み込みの仮想メソッド。
virtual Type	識別の仮想メソッド。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CChartObject

© 2000-2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

派生クラス

- CChartObjectLabel

参照

オブジェクトの型、オブジェクトプロパティ、オブジェクト結合の方法、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

「Text（テキスト）」グラフィックオブジェクトを作成します。

```
bool  Create(
   long       chart_id,      // チャート識別子
   string     name,          // オブジェクト名
   int        window,        // チャートウィンドウ
   datetime   time,          // 時間座標
   double     price          // 価格座標
  )
```

パラメータ

chart_id

[in] チャート識別子（0は現在のチャート）

name

[in] 作成するオブジェクトの一意の名称

window

[in] チャートウィンドウの番号（0はメインウィンドウ）

time

[in] アンカーポイントの時間座標

価格

[in] アンカーポイントの価格座標

戻り値

成功の場合はtrue、エラーが発生した場合はfalse

© 2000–2025, MetaQuotes Ltd.

Angle（Get メソッド）

「Angle（度）」プロパティの値を取得します。

```
double  Angle() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「Angle（度）」プロパティの値（割り当てられたオブジェクトが存在しない場合は EMPTY_VALUE ）

Angle（Set メソッド）

「Angle（度）」プロパティの新しい値を設定します。

```
bool  Angle(
   double  angle      // 新しい angle
  )
```

パラメータ

angle

[in]「Angle（度）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Font（Get メソッド）

「Font（フォント）」プロパティの値を取得します。

```
string  Font() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「Font（フォント）」プロパティの値（割り当てられたオブジェクトが存在しない場合は ""）

Font（Set メソッド）

「Font（フォント）」プロパティの新しい値

```
bool  Font(
   string  font      // 新しいフォント
  )
```

パラメータ

font

[in]「Font（フォント）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

FontSize（Get メソッド）

「FontSize（フォント サイズ）」プロパティの値を取得します。

```
int  FontSize() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「FontSize（フォント サイズ）」プロパティの値（割り当てられたオブジェクトが存在しない場合は 0）

FontSize（Set メソッド）

「FontSize（フォント サイズ）」プロパティの新しい値を設定します。

```
bool  FontSize(
   int  size      // 新しいフォントサイズ
  )
```

パラメータ

size

[in] New value for "Font size" property.

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Anchor（Get メソッド）

「Anchorr（アンカー）」プロパティの値を取得します。

```
ENUM_ANCHOR_POINT  Anchor() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「Anchorr（アンカー）」プロパティの値（割り当てられたオブジェクトが存在しない場合は WRONG_VALUE）

Anchor（Set メソッド）

「Anchorr（アンカー）」プロパティの新しい値を設定します。

```
bool  Anchor(
   ENUM_ANCHOR_POINT  anchor      // 新しい値
  )
```

パラメータ

anchor

[in]「Anchorr（アンカー）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Save

オブジェクト パラメータをファイルに保 存します。

```
virtual bool  Save(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] すでに FileOpen 関 数 で開 かれたバイナリファイルのハンド ル

戻 り値

成 功 の場 合 は true、エラーが発 生 した場 合 は false

© 2000–2025, MetaQuotes Ltd.

Load

オブジェクト パラメータをファイルから読み込みます。

```
virtual bool  Load(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] すでに FileOpen 関数で開かれたバイナリファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

Type

グラフィックオブジェクトの型の識別子を取得します。

```
virtual int  Type() const
```

戻り値

オブジェクト型の識別子（OBJ_TEXTはCChartObjectText）

© 2000–2025, MetaQuotes Ltd.

CChartObjectLabel

CChartObjectLabel は「Label（ラベル）」グラフィックオブジェクト のプロパティに簡 単 にアクセスするためのクラスです。

説 明

CChartObjectLabel クラスは「Label（ラベル）」グラフィックオブジェクト のプロパティへのアクセスを簡 易 化します。

宣 言

```
class CChartObjectLabel : public CChartObjectText
```

タイト ル

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

継 承 階 層

CObject
CChartObject
CChartObjectText
CChartObjectLabel

直 接 子 孫

CChartObjectEdit, CChartObjectRectLabel

クラスメソッド

作 成	
Create	「Label（ラベル）」グラフィックオブジェクト を作 成します。
プロパティ	
X_Distance	「X_Distance（ X の距 離 ）」プロパティを取 得 /設 定します。
Y_Distance	「Y_Distance（ Y の距 離 ）」プロパティを取 得 /設 定します。
X_Size	「X_Size（ X サイズ）」プロパティを取 得 /設 定します。
Y_Size	「Y_Size（ Y サイズ）」プロパティを取 得 /設 定します。
Corner	「Corner（コーナー）」プロパティを取 得 /設 定します。
Time	時 間 座 標 変 更 のスタブ
Price	価 格 座 標 変 更 のスタブ
入 出 力	
virtual Save	ファイル書 き込 みの仮 想 メソッド 。
virtual Load	ファイル読 み込 みの仮 想 メソッド 。

© 2000–2025, MetaQuotes Ltd.

作成	
virtual Type	識別の仮想メソッド。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

クラスから継承されたメソッド CChartObjectText

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

参照

オブジェクトの型、オブジェクトプロパティ、チャート角度、オブジェクト結合の方法, グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

「Label（ラベル）」グラフィックオブジェクトを作成します。

```
bool  Create(
   long     chart_id,      // チャート識別子
   string   name,          // オブジェクト名
   int      window,        // チャートウィンドウ
   int      X,             // X 座標
   int      Y              // Y 座標
  )
```

パラメータ

chart_id

[in] チャート識別子（0は現在のチャート）

name

[in] 作成するオブジェクトの一意の名称

window

[in] チャートウィンドウの番号（0はメインウィンドウ）

X

[in] X座標

Y

[in] Y座標

戻り値

成功の場合はtrue、エラーが発生した場合はfalse

© 2000–2025, MetaQuotes Ltd.

X_ Distance（Get メソッド）

「X_ Distance（ X の距 離 ）」プロパティの値 を取 得します。

```
int  X_Distance() const
```

戻 り値

クラスインスタンスに割り当 てられたオブジェクト の「X_ Distance（ X の距 離 ）」プロパティの値（割り当 てられたオブジェクト が存 在しない場 合 は 0 ）

X_ Distance（Set メソッド）

「X_ Distance（ X の距 離 ）」プロパティの新しい値 を設 定します。

```
bool  X_Distance(
   int  X      // 新しい値
  )
```

パラメータ

X

[in]「X_ Distance（ X の距 離 ）」プロパティの新しい値

戻 り値

成 功 の場 合 は true、プロパティ値 が変 更 できなかった場 合 は false

© 2000–2025, MetaQuotes Ltd.

Y_ Distance（Get メソッド）

「Y_ Distance（ Y の距 離 ）」プロパティの値 を取 得します。

```
int  Y_Distance() const
```

戻 り値

クラスインスタンスに割り当 てられたオブジェクト の「Y_ Distance（ Y の距 離 ）」プロパティの値（割り当 てられたオブジェクト が存 在しない場 合 は 0 ）

Y_ Distance（Set メソッド）

「Y_ Distance（ Y の距 離 ）」プロパティの新しい値 を設 定します。

```
bool  Y_Distance(
   int  Y      // 新しい値
  )
```

パラメータ

Y

[in]「Y_ Distance（ Y の距 離 ）」プロパティの新しい値

戻 り値

成 功 の場 合 は true、プロパティ値 が変 更 できなかった場 合 は false

© 2000–2025, MetaQuotes Ltd.

X_Size

「X_Size（ X サイズ）」プロパティの値を取得します。

```
int  X_Size() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「X_Size（ X サイズ）」プロパティの値（割り当てられたオブジェクトが存在しない場合は 0 ）

© 2000–2025, MetaQuotes Ltd.

Y_Size

「Y_Size（ Y サイズ）」プロパティの値を取得します。

```
int  Y_Size() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「Y_Size（ Y サイズ）」プロパティの値（割り当てられたオブジェクトが存在しない場合は 0 ）

© 2000–2025, MetaQuotes Ltd.

Corner（Get メソッド）

「Corner（コーナー）」プロパティの値を取得します。

```
ENUM_BASE_CORNER  Corner() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「Corner（コーナー）」プロパティの値（割り当てられたオブジェクトが存在しない場合は WRONG_VALUE）

Corner（Set メソッド）

「Corner（コーナー）」プロパティの新しい値を設定します。

```
bool  Corner(
   ENUM_BASE_CORNER  corner      // 新しい値
  )
```

パラメータ

corner

[in]「Corner（コーナー）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Time

時間座標の変更を禁止します。

```
bool  Time(
   datetime  time      // 任意の値
  )
```

パラメータ

time

[in] datetime 型の任意の値

戻り値

常に false

© 2000–2025, MetaQuotes Ltd.

Price

価格座標の変更を禁止します。

```
bool  Price(
   double  price      // 任意の値
  )
```

パラメータ

価格

[in] double 型の任意の値

戻り値

常に false

© 2000–2025, MetaQuotes Ltd.

Save

オブジェクト パラメータをファイルに保 存します。

```
virtual bool  Save(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] すでに FileOpen 関 数 で開 かれたバイナリファイルのハンド ル

戻 り値

成 功 の場 合 は true、エラーが発 生 した場 合 は false

© 2000–2025, MetaQuotes Ltd.

Load

オブジェクト パラメータをファイルから読み込みます。

```
virtual bool  Load(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] すでに FileOpen 関数で開かれたバイナリファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

Type

グラフィックオブジェクトの型の識別子を取得します。

```
virtual int  Type() const
```

戻り値

オブジェクト型の識別子（OBJ_LABELはCChartObjectLabel）

© 2000–2025, MetaQuotes Ltd.

CChartObjectEdit

CChartObjectEdit は「Edit（編集）」グラフィックオブジェクトのプロパティに簡単にアクセスするためのクラスです。

説明

CChartObjectEdit クラスは「Edit（編集）」グラフィックオブジェクトのプロパティへのアクセスを簡易化します。

宣言

```
class CChartObjectEdit : public CChartObjectLabel
```

タイトル

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

継承階層

CObject
CChartObject
CChartObjectText
CChartObjectLabel
CChartObjectEdit

直接子孫

CChartObjectButton

クラスメソッド

作成	
Create	「Edit（編集）」グラフィックオブジェクトを作成します。
プロパティ	
TextAlign	「TextAlign」プロパティを取得します。
X_Size	「X Size」プロパティを取得します。
Y_Size	「Y Size」プロパティを取得します。
BackColor	「Background Color（背景色）」プロパティを取得/設定します。
BorderColor	「Border Color（境界線の色）」プロパティを取得/設定します。
ReadOnly	「Read Only（読み込み専用）」プロパティを取得/設定します。
Angle	「Angle（角度）」プロパティを取得/設定します。
入出力	
virtual Save	ファイル書き込みの仮想メソッド。
virtual Load	ファイル読み込みの仮想メソッド。

© 2000–2025, MetaQuotes Ltd.

作成	
virtual Type	識別の仮想メソッド。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

クラスから継承されたメソッド CChartObjectText

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

クラスから継承されたメソッド CChartObjectLabel

X_Distance, X_Distance, Y_Distance, Y_Distance, X_Size, Y_Size, Corner, Corner, Time, Price, Create

参照

オブジェクトの型、オブジェクトプロパティ、チャート角度、オブジェクト結合の方法, グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

「Edit（編集）」グラフィックオブジェクトを作成します。

```
bool  Create(
   long     chart_id,      // チャート識別子
   string   name,          // オブジェクト名
   int      window,        // チャートウィンドウ
   int      X,             // X 座標
   int      Y,             // Y 座標
   int      sizeX,         // X サイズ
   int      sizeY          // Y サイズ
  )
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] 作成するオブジェクトの一意の名称

window

[in] チャートウィンドウの番号（0 はメインウィンドウ）

X

[in] X 座標

Y

[in] Y 座標

sizeX

[in] X サイズ

sizeY

[in] Y サイズ

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

TextAlign（Get メソッド）

「TextAlign」プロパティ（テキスト アラインメント モード）の値を取得します。

```
ENUM_ALIGN_MODE  TextAlign() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「TextAlign」プロパティの値

TextAlign（Set メソッド）

「TextAlign」プロパティ（テキスト アラインメント モード）の値を設定します。

```
bool  TextAlign(
   ENUM_ALIGN_MODE  align      // 新しい値
  )
```

パラメータ

align

[in]「TextAlign」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

X_Size

「X_Size(X サイズ)」プロパティの新しい値を設定します。

```
bool  X_Size(
   int  size      // 新しいサイズ
  )
```

パラメータ

size

[in]「X_Size(X サイズ)」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000-2025, MetaQuotes Ltd.

Y_Size

「Y_Size(Y サイズ)」プロパティの新しい値を設定します。

```
bool  Y_Size(
   int  size      // 新しいサイズ
  )
```

パラメータ

size

[in] 「Y_Size(Y サイズ)」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000-2025, MetaQuotes Ltd.

BackColor（Get メソッド）

「BackColor（背景）」プロパティの値を取得します。

```
color  BackColor() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「BackColor（背景）」プロパティの値（割り当てられたオブジェクトが存在しない場合は CLR_NONE）

BackColor（Set メソッド）

「BackColor（背景）」プロパティの新しい値を設定します。

```
bool  BackColor(
   color  new_color      // 新しい背景色
  )
```

パラメータ

new_color

[in]「BackColor（背景）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

BorderColor（Get メソッド）

「Border Color（境界線の色）」プロパティの値を取得します。

```
color  BorderColor() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「BorderColor」プロパティの値（割り当てられたオブジェクトが存在しない場合は CLR_NONE）

BorderColor（Set メソッド）

「Border Color（境界線の色）」プロパティの新しい値を設定します。

```
bool  BorderColor(
   color  new_color      // 新しい境界線の色
  )
```

パラメータ

new_color

[in]「Border Color（境界線の色）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

ReadOnly（Get メソッド）

「Read Only（読み込み専用）」プロパティの値を取得します。

```
bool  ReadOnly() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「「Read Only」（読み込み専用）」プロパティの値（割り当てられたオブジェクトが存在しない場合は false）

ReadOnly（Set メソッド）

「Read Only（読み込み専用）」プロパティの新しい値を設定します。

```
bool  ReadOnly(
   const  bool  flag      // 新しい値
  )
```

パラメータ

flag

[in]「Read Only（読み込み専用）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Angle

「Angle(度)」プロパティの変更を禁止します。

```
bool  Angle(
   double  angle      // 任意の値
  )
```

パラメータ

angle

[in] double 型の任意の値

戻り値

常に false

© 2000-2025, MetaQuotes Ltd.

Save

オブジェクト パラメータをファイルに保 存します。

```
virtual bool  Save(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] すでに FileOpen 関 数 で開 かれたバイナリファイルのハンド ル

戻 り値

成 功 の場 合 は true、エラーが発 生 した場 合 は false

© 2000–2025, MetaQuotes Ltd.

Load

オブジェクト パラメータをファイルから読み込みます。

```
virtual bool  Load(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] すでに FileOpen 関数で開かれたバイナリファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

Type

グラフィックオブジェクトの型の識別子を取得します。

```
virtual int  Type() const
```

戻り値

オブジェクト型の識別子（OBJ_EDITはCChartObjectEdit）

© 2000-2025, MetaQuotes Ltd.

CChartObjectButton

CChartObjectButton は「Button（ボタン）」グラフィックオブジェクト のプロパティに簡 単 にアクセスするためのクラスです。

説 明

CChartObjectButton クラスは「Button（ボタン）」グラフィックオブジェクト のプロパティへのアクセスを簡 易 化 します。

宣 言

```
class CChartObjectButton : public CChartObjectEdit
```

タイト ル

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

継 承 階 層

CObject
CChartObject
CChartObjectText
CChartObjectLabel
CChartObjectEdit
CChartObjectButton

直 接 子 孫

CChartObjectPanel

クラスメソッド

作 成	
Create	CChartObjectEdit クラスから継 承 されています。
プロパティ	
State	ボタンの状 態（押 された/放 された）を取 得 /設 定 します。
入 出 力	
virtual Save	ファイル書 き込 みの仮 想 メソッド。
virtual Load	ファイル読 み込 みの仮 想 メソッド。
virtual Type	識 別 の仮 想 メソッド。

クラスから継 承 されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継 承 されたメソッド CChartObject

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

クラスから継承されたメソッド CChartObjectText

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

クラスから継承されたメソッド CChartObjectLabel

X_Distance, X_Distance, Y_Distance, Y_Distance, X_Size, Y_Size, Corner, Corner, Time, Price, Create

クラスから継承されたメソッド CChartObjectEdit

X_Size, Y_Size, BackColor, BackColor, BorderColor, BorderColor, ReadOnly, ReadOnly, TextAlign, TextAlign, Angle, Create

参照

オブジェクトの型、オブジェクトプロパティ、チャート角度、オブジェクト結合の方法、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

State（Get メソッド）

「State(（ステート）」プロパティの値を取得します。

```
bool  State() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「State(（ステート）」プロパティの値（割り当てられたオブジェクトが存在しない場合は false）

State（Set メソッド）

「State(（ステート）」プロパティの新しい値を設定します。

```
bool  State(
   bool  state      // 新しいstate 値
  )
```

パラメータ

X

[in] 「State(（ステート）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Save

オブジェクト パラメータをファイルに保 存します。

```
virtual bool  Save(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] すでに FileOpen 関 数 で開 かれたバイナリファイルのハンド ル

戻 り値

成 功 の場 合 は true、エラーが発 生 した場 合 は false

© 2000–2025, MetaQuotes Ltd.

Load

オブジェクト パラメータをファイルから読み込みます。

```
virtual bool  Load(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] すでに FileOpen 関数で開かれたバイナリファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

Type

グラフィックオブジェクトの型の識別子を取得します。

```
virtual int  Type() const
```

戻り値

オブジェクト型の識別子（OBJ_BUTTONはCChartObjectButton）

© 2000-2025, MetaQuotes Ltd.

CChartObjectSubChart

CChartObjectSubChart クラスは「Chart（チャート）」グラフィックオブジェクトのプロパティに簡単にアクセスするためのクラスです。

説明

CChartObjectSubChart クラスは「Chart（チャート）」オブジェクトプロパティへのアクセスを提供します。

宣言

```
class CChartObjectSubChart : public CChartObject
```

タイトル

```
#include <ChartObjects\ChartObjectSubChart.mqh>
```

継承階層

CObject
  CChartObject
    CChartObjectSubChart

クラスメソッド

作成	
Create	「Chart（チャート）」グラフィックオブジェクトを作成します。
プロパティ	
X_Distance	「X_Distance（X の距離）」プロパティを取得/設定します。
Y_Distance	「Y_Distance（Y の距離）」プロパティを取得/設定します。
Corner	「Corner（コーナー）」プロパティを取得/設定します。
X_Size	「X_Size（X サイズ）」プロパティを取得/設定します。
Y_Size	「Y_Size（Y サイズ）」プロパティを取得/設定します。
Symbol	「Symbol（シンボル）」プロパティを取得/設定します。
Period	「Period（期間）」プロパティを取得/設定します。
Scale	「Scale（スケール）」プロパティを取得/設定します。
DateScale	「Show date scale」プロパティを取得/設定します。
PriceScale	「Show price scale」プロパティを取得/設定します。
Time	時間座標変更のスタブ
Price	価格座標変更のスタブ
入出力	

© 2000–2025, MetaQuotes Ltd.

作成	
virtual Save	ファイル書き込みの仮想メソッド。
virtual Load	ファイル読み込みの仮想メソッド。
virtual Type	識別の仮想メソッド。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

参照

オブジェクトの型、オブジェクトプロパティ、チャート角度、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

「SubChart（サブチャート）」グラフィックオブジェクトを作成します。

```
bool  Create(
   long     chart_id,      // チャート識別子
   string   name,          // オブジェクト名
   int      window,        // チャートウィンドウ
   int      X,             // X 座標
   int      Y,             // Y 座標
   int      sizeX,         // X サイズ
   int      sizeY          // Y サイズ
  )
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] 作成するオブジェクトの一意の名称

window

[in] チャートウィンドウの番号（0 はメインウィンドウ）

X

[in] X 座標

Y

[in] Y 座標

sizeX

[in] X サイズ

sizeY

[in] Y サイズ

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

X_ Distance（Get メソッド）

「X_ Distance（ X の距 離 ）」プロパティの値 を取 得します。

```
int  X_Distance() const
```

戻 り値

クラスインスタンスに割り当 てられたオブジェクト の「X_ Distance（ X の距 離 ）」プロパティの値（割り当 てられたオブジェクト が存 在しない場 合 は 0 ）

X_ Distance（Set メソッド）

「X_ Distance（ X の距 離 ）」プロパティの新しい値 を設 定します。

```
bool  X_Distance(
   int  X      // 新しい値
  )
```

パラメータ

X

[in]「X_ Distance（ X の距 離 ）」プロパティの新しい値

戻 り値

成 功 の場 合 は true、プロパティ値 が変 更 できなかった場 合 は false

© 2000–2025, MetaQuotes Ltd.

Y_ Distance（Get メソッド）

「Y_ Distance（ Y の距 離 ）」プロパティの値 を取 得します。

```
int  Y_Distance() const
```

戻 り値

クラスインスタンスに割り当 てられたオブジェクト の「Y_ Distance（ Y の距 離 ）」プロパティの値（割り当 てられたオブジェクト が存 在しない場 合 は 0 ）

Y_ Distance（Set メソッド）

「Y_ Distance（ Y の距 離 ）」プロパティの新しい値 を設 定します。

```
bool  Y_Distance(
   int  Y      // 新しい値
  )
```

パラメータ

Y

[in]「Y_ Distance（ Y の距 離 ）」プロパティの新しい値

戻 り値

成 功 の場 合 は true、プロパティ値 が変 更 できなかった場 合 は false

© 2000–2025, MetaQuotes Ltd.

Corner（Get メソッド）

「Corner（コーナー）」プロパティの値を取得します。

```
ENUM_BASE_CORNER Corner() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「Corner（コーナー）」プロパティの値（割り当てられたオブジェクトが存在しない場合は WRONG_VALUE）

Corner（Set メソッド）

「Corner（コーナー）」プロパティの新しい値を設定します。

```
bool  Corner(
   ENUM_BASE_CORNER  corner      // 新しい値
  )
```

パラメータ

corner

[in]「Corner（コーナー）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

X_ Size（Get メソッド）

「X_ Size（ X サイズ）」プロパティの値を取得します。

```
int  X_Size() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「X_ Size（ X サイズ）」プロパティの値（割り当てられたオブジェクトが存在しない場合は 0 ）

X_ Size（Set メソッド）

「X_ Size（ X サイズ）」プロパティの新しい値を設定します。

```
bool  X_Size(
   int  X      // 新しい値
  )
```

パラメータ

X

[in]「X_ Size（ X サイズ）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Y_Size（Get メソッド）

「Y_Size（Y サイズ）」プロパティの値を取得します。

```
int  Y_Size() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「Y_Size（Y サイズ）」プロパティの値（割り当てられたオブジェクトが存在しない場合は 0）

Y_Size（Set メソッド）

「Y_Size（Y サイズ）」プロパティの新しい値を設定します。

```
bool  Y_Size(
   int  Y      // 新しい値
  )
```

パラメータ

Y

[in]「Y_Size（Y サイズ）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000-2025, MetaQuotes Ltd.

Symbol（Get メソッド）

「Symbol（シンボル）」プロパティの値を取得します。

```
string  Symbol() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「Symbol（シンボル）」プロパティの値（割り当てられたオブジェクトが存在しない場合は ""）

Symbol（Set メソッド）

「Symbol（シンボル）」プロパティの新しい値を設定します。

```
bool  Symbol(
   string  symbol       // 新しいシンボル
  )
```

パラメータ

symbol

[in]「Symbol（シンボル）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Period (Get Method)

Gets the value of "Period" property.

```
int  Period() const
```

戻り値

Value of "Period" property of the object, assigned to the class instance.（割り当てられたオブジェクトが存在しない場合は 0 ）

Period（Set メソッド）

「Period（期間）」プロパティの新しい値を設定します。

```
bool  Period(
   int  period      // 新しい期間
  )
```

パラメータ

period

[in]「Period（期間）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Scale（Get メソッド）

「Scale（スケール）」プロパティの値を取得します。

```
double  Scale() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「Scale（スケール）」プロパティの値（割り当てられたオブジェクトが存在しない場合は EMPTY_VALUE ）

Scale（Set メソッド）

「Scale（スケール）」プロパティの新しい値を設定します。

```
bool  Scale(
   double  scale      // 新しいスケール
  )
```

パラメータ

scale

[in]「Scale（スケール）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000-2025, MetaQuotes Ltd.

DateScale（Get メソッド）

「DateScale」プロパティの値を取得します。

```
bool  DateScale() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「DateScale」プロパティの値（割り当てられたオブジェクトが存在しない場合は false）

DateScale（Set メソッド）

「DateScale」プロパティの値を設定します。

```
bool  DateScale(
   bool  scale      // 新しい値
  )
```

パラメータ

scale

[in]「DateScale」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

PriceScale（Get メソッド）

「PriceScale」プロパティの値を取得します。

```
bool  PriceScale() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「PriceScale」プロパティの値（割り当てられたオブジェクトが存在しない場合は false）

PriceScale（Set メソッド）

「PriceScale」プロパティの新しい値を設定します。

```
bool  PriceScale(
   bool  scale      // 新しい値
  )
```

パラメータ

scale

[in]「PriceScale」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Time

時間座標の変更を禁止します。

```
bool  Time(
   datetime  time      // 任意の値
  )
```

パラメータ

time

[in] datetime 型の任意の値

戻り値

常に false

© 2000–2025, MetaQuotes Ltd.

Price

価格座標の変更を禁止します。

```
bool  Price(
   double  price      // 任意の値
  )
```

パラメータ

価格

[in] double 型の任意の値

戻り値

常に false

© 2000–2025, MetaQuotes Ltd.

Save

オブジェクト パラメータをファイルに保 存します。

```
virtual bool  Save(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] すでに FileOpen 関 数 で開 かれたバイナリファイルのハンド ル

戻 り値

成 功 の場 合 は true、エラーが発 生 した場 合 は false

© 2000–2025, MetaQuotes Ltd.

Load

オブジェクト パラメータをファイルから読み込みます。

```
virtual bool  Load(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] すでに FileOpen 関数で開かれたバイナリファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

Type

グラフィックオブジェクトの型の識別子を取得します。

```
virtual int  Type() const
```

戻り値

Object type identifier（OBJ_CHART for CChartObjectSubChart).

© 2000–2025, MetaQuotes Ltd.

CChartObjectBitmap

CChartObjectBitmap は「Bitmap（ビットマップ）」グラフィックオブジェクトのプロパティに簡単にアクセスするためのクラスです。

説明

CChartObjectBitmap クラスは「Bitmap（ビットマップ）」グラフィックオブジェクトのプロパティへのアクセスを簡易化します。

宣言

```
class CChartObjectBitmap : public CChartObject
```

タイトル

```
#include <ChartObjects\ChartObjectsBmpControls.mqh>
```

継承階層

CObject
  CChartObject
    CChartObjectBitmap

クラスメソッド

作成	
Create	「Bitmap（ビットマップ）」グラフィックオブジェクトを作成します。
プロパティ	
BmpFile	「BMP Filename」プロパティを取得/設定します。
X_Offset	「X_Offset（Xのオフセット）」プロパティを取得/設定します。
Y_Offset	「Y_Offset（Yのオフセット）」プロパティを取得/設定します。
入出力	
virtual Save	ファイル書き込みの仮想メソッド。
virtual Load	ファイル読み込みの仮想メソッド。
virtual Type	識別の仮想メソッド。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor,

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

参照

オブジェクトの型、オブジェクトプロパティ、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

「Bitmap（ビットマップ）」グラフィックオブジェクトを作成します。

```
bool  Create(
   long       chart_id,      // チャート識別子
   string     name,          // オブジェクト名
   int        window,        // チャートウィンドウ
   datetime   time,          // 時間座標
   double     price          // 価格座標
  )
```

パラメータ

chart_id

[in] チャート識別子（0は現在のチャート）

name

[in] 作成するオブジェクトの一意の名称

window

[in] チャートウィンドウの番号（0はメインウィンドウ）

time

[in] アンカーポイントの時間座標

価格

[in] アンカーポイントの価格座標

戻り値

成功の場合はtrue、エラーが発生した場合はfalse

© 2000-2025, MetaQuotes Ltd.

BmpFile（Get メソッド）

「BmpFile（ビット マップファイル）」プロパティの値を取得します。

```
string  BmpFile() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「BmpFile（ビット マップファイル）」プロパティの値（割り当てられたオブジェクトが存在しない場合は false）

BmpFile（Set メソッド）

「BmpFile（ビット マップファイル）」プロパティの値を設定します。

```
bool  BmpFile(
   string  name      // 新しいファイル名
  )
```

パラメータ

name

[in]「BmpFile（ビット マップファイル）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

X_ Offset（Get メソッド）

「X_ Offset（ X のオフセット）」プロパティの値 （グラフィカルオブジェクト の表 示 領 域 の左 上 隅 のX 座 標）を取得します。

```
int  X_Offset() const
```

戻り値

クラスインスタンスに割り当 てられたオブジェクト の「X_ Offset（ X のオフセット）」プロパティの値（割り当 てられたオブジェクト が存 在しない場 合 は 0 ）

X_ Offset（Set メソッド）

新しい「X_ Offset（ X のオフセット）」プロパティの値 （グラフィカルオブジェクト の表 示 領 域 の左 上 隅 のX 座 標）を設 定します。値 は元 の画 像 の左 上 隅 を基 準 にピクセル単 位 で設 定されています。

```
bool  X_Offset(
   int  X      // 新しい値
  )
```

パラメータ

X

[in] 「X_ Offset（ X のオフセット）」プロパティの新 しい値

戻り値

成 功 の場 合 は true、プロパティ値 が変 更 できなかった場 合 は false

© 2000–2025, MetaQuotes Ltd.

Y_ Offset（Get メソッド）

「Y_ Offset（ Y のオフセット）」プロパティの値 （グラフィカルオブジェクトの表示領域の左上隅のY 座標）を取得します。

```
int  Y_Offset() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの"Y_ Offset" プロパティの値（割り当てられたオブジェクトが存在しない場合は 0 ）

Y_ Offset（Set メソッド）

新しい「Y_ Offset（ Y のオフセット）」プロパティの値（グラフィカルオブジェクトの表示領域の左上隅のX 座標）を設定します。値は元の画像の左上隅を基準にピクセル単位で設定されています。

```
bool  Y_Offset(
   int  Y      // 新しい値
  )
```

パラメータ

Y

[in] 「Y_ Offset（ Y のオフセット）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Save

オブジェクト パラメータをファイルに保 存します。

```
virtual bool  Save(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] すでに FileOpen 関 数 で開 かれたバイナリファイルのハンド ル

戻 り値

成 功 の場 合 は true、エラーが発 生 した場 合 は false

© 2000–2025, MetaQuotes Ltd.

Load

オブジェクト パラメータをファイルから読み込みます。

```
virtual bool  Load(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] すでに FileOpen 関数で開かれたバイナリファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

Type

グラフィックオブジェクトの型の識別子を取得します。

```
virtual int  Type() const
```

戻り値

オブジェクト型の識別子（OBJ_BITMAPはCChartObjectBitmap）

© 2000–2025, MetaQuotes Ltd.

CChartObjectBmpLabel

CChartObjectBmpLabel は「Bitmap Label（ビットマップラベル）」グラフィックオブジェクトのプロパティに簡単にアクセスするためのクラスです。

説明

CChartObjectBmpLabel クラスは「Bitmap Label（ビットマップラベル）」グラフィックオブジェクトのプロパティへのアクセスを簡易化します。

宣言

```
class CChartObjectBmpLabel : public CChartObject
```

タイトル

```
#include <ChartObjects\ChartObjectsBmpControls.mqh>
```

継承階層

CObject
  CChartObject
    CChartObjectBmpLabel

クラスメソッド

作成	
Create	「Bitmap Label（ビットマップラベル）」グラフィックオブジェクトを作成します。
プロパティ	
X_Distance	「X_Distance（X の距離）」プロパティを取得/設定します。
Y_Distance	「Y_Distance（Y の距離）」プロパティを取得/設定します。
X_Offset	「X_Offset（X のオフセット）」プロパティを取得/設定します。
Y_Offset	「Y_Offset（Y のオフセット）」プロパティを取得/設定します。
Corner	「Corner（コーナー）」プロパティを取得/設定します。
X_Size	「X_Size（X サイズ）」プロパティを取得/設定します。
Y_Size	「Y_Size（Y サイズ）」プロパティを取得/設定します。
BmpFileOn	ボタンの押された状態（On）を表す「BmpFileOn」プロパティを取得/設定します。
BmpFileOff	ボタンの放された状態（Off）を表す「BmpFileOff」プロパティを取得/設定します。
State	ボタンの状態（押された/放された）を表す「Button State」プロパティを取得/設定します。
Time	時間座標変更のスタブ

© 2000–2025, MetaQuotes Ltd.

作成	
Price	価格座標変更のスタブ
入出力	
virtual Save	ファイル書き込みの仮想メソッド。
virtual Load	ファイル読み込みの仮想メソッド。
virtual Type	識別の仮想メソッド。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

参照

オブジェクトの型、オブジェクトプロパティ、チャート角度、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

「BmpLabel」グラフィックオブジェクトを作成します。

```
bool  Create(
   long     chart_id,      // チャート識別子
   string   name,          // オブジェクト名
   int      window,        // チャートウィンドウ
   int      X,             // X 座標
   int      Y              // Y 座標
  )
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] 作成するオブジェクトの一意の名称

window

[in] チャートウィンドウの番号（0 はメインウィンドウ）

X

[in] X 座標

Y

[in] Y 座標

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

X_ Distance（Get メソッド）

「X_ Distance（ X の距 離 ）」プロパティの値 を取 得します。

```
int  X_Distance() const
```

戻 り値

クラスインスタンスに割り当 てられたオブジェクト の「X_ Distance（ X の距 離 ）」プロパティの値（割り当 てられたオブジェクト が存 在しない場 合 は 0 ）

X_ Distance（Set メソッド）

「X_ Distance（ X の距 離 ）」プロパティの新しい値 を設 定します。

```
bool  X_Distance(
   int  X      // 新しい値
  )
```

パラメータ

X

[in]「X_ Distance（ X の距 離 ）」プロパティの新しい値

戻 り値

成 功 の場 合 は true、プロパティ値 が変 更 できなかった場 合 は false

© 2000–2025, MetaQuotes Ltd.

Y_ Distance（Get メソッド）

「Y_ Distance（ Y の距 離 ）」プロパティの値 を取 得します。

```
int  Y_Distance() const
```

戻 り値

クラスインスタンスに割り当 てられたオブジェクト の「Y_ Distance（ Y の距 離 ）」プロパティの値（割り当 てられたオブジェクト が存 在しない場 合 は 0 ）

Y_ Distance（Set メソッド）

「Y_ Distance（ Y の距 離 ）」プロパティの新しい値 を設 定します。

```
bool  Y_Distance(
   int  Y      // 新しい値
  )
```

パラメータ

Y

[in]「Y_ Distance（ Y の距 離 ）」プロパティの新しい値

戻 り値

成 功 の場 合 は true、プロパティ値 が変 更 できなかった場 合 は false

© 2000–2025, MetaQuotes Ltd.

X_ Offset（Get メソッド）

「X_ Offset（ X のオフセット）」プロパティの値 （グラフィカルオブジェクト の表 示 領 域 の左 上 隅 のX 座 標）を取得します。

```
int  X_Offset() const
```

戻り値

クラスインスタンスに割り当 てられたオブジェクト の「X_ Offset（ X のオフセット）」プロパティの値（割り当 てられたオブジェクト が存 在しない場 合 は 0 ）

X_ Offset（Set メソッド）

新しい「X_ Offset（ X のオフセット）」プロパティの値 （グラフィカルオブジェクト の表 示 領 域 の左 上 隅 のX 座 標）を設 定します。値 は元 の画 像 の左 上 隅 を基 準 にピクセル単 位 で設 定されています。

```
bool  X_Offset(
   int  X      // 新しい値
  )
```

パラメータ

X

[in] 「X_ Offset（ X のオフセット）」プロパティの新 しい値

戻り値

成 功 の場 合 は true、プロパティ値 が変 更 できなかった場 合 は false

© 2000–2025, MetaQuotes Ltd.

Y_ Offset（Get メソッド）

「Y_ Offset（ Y のオフセット）」プロパティの値 （グラフィカルオブジェクトの表示領域の左上隅のY 座標）を取得します。

```
int  Y_Offset() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの"Y_ Offset" プロパティの値（割り当てられたオブジェクトが存在しない場合は 0 ）

Y_ Offset（Set メソッド）

新しい「Y_ Offset（ Y のオフセット）」プロパティの値（グラフィカルオブジェクトの表示領域の左上隅のX 座標）を設定します。値は元の画像の左上隅を基準にピクセル単位で設定されています。

```
bool  Y_Offset(
   int  Y      // 新しい値
  )
```

パラメータ

Y

[in] 「Y_ Offset（ Y のオフセット）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Corner（Get メソッド）

「Corner（コーナー）」プロパティの値を取得します。

```
ENUM_BASE_CORNER Corner() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「Corner（コーナー）」プロパティの値（割り当てられたオブジェクトが存在しない場合は WRONG_VALUE）

Corner（Set メソッド）

「Corner（コーナー）」プロパティの新しい値を設定します。

```
bool  Corner(
   ENUM_BASE_CORNER  corner      // 新しい値
  )
```

パラメータ

corner

[in]「Corner（コーナー）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

X_Size

「X_Size（ X サイズ）」プロパティの値を取得します。

```
int  X_Size() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「X_Size（ X サイズ）」プロパティの値（割り当てられたオブジェクトが存在しない場合は 0 ）

© 2000–2025, MetaQuotes Ltd.

Y_Size

「Y_Size（ Y サイズ）」プロパティの値を取得します。

```
int  Y_Size() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「Y_Size（ Y サイズ）」プロパティの値（割り当てられたオブジェクトが存在しない場合は 0 ）

© 2000–2025, MetaQuotes Ltd.

BmpFileOn（Get メソッド）

「BmpFileOn」プロパティの値を取得します。

```
string  BmpFileOn() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「BmpFileOn」プロパティの値（割り当てられたオブジェクトが存在しない場合は ""）

BmpFileOn（Set メソッド）

「BmpFileOn」プロパティの新しい値を設定します。

```
bool  BmpFileOn(
   string  name      // ファイル名
  )
```

パラメータ

name

[in]「BmpFileOn」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

BmpFileOff（Get メソッド）

「BmpFileOff」プロパティの値を取得します。

```
string  BmpFileOff() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「BmpFileOff」プロパティの値（割り当てられたオブジェクトが存在しない場合は ""）

BmpFileOff（Set メソッド）

「BmpFileOff」プロパティの新しい値を設定します。

```
bool  BmpFileOff(
   string  name      // ファイル名
  )
```

パラメータ

name

[in]「BmpFileOff」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

State（Get メソッド）

「State(（ステート）」プロパティの値を取得します。

```
bool  State() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「State(（ステート）」プロパティの値（割り当てられたオブジェクトが存在しない場合は false）

State（Set メソッド）

「State(（ステート）」プロパティの新しい値を設定します。

```
bool  State(
   bool  state      // 新しいstate 値
  )
```

パラメータ

state

[in]「State(（ステート）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Time

時間座標の変更を禁止します。

```
bool  Time(
   datetime  time      // 任意の値
  )
```

パラメータ

time

[in] datetime 型の任意の値

戻り値

常に false

© 2000–2025, MetaQuotes Ltd.

Price

価格座標の変更を禁止します。

```
bool  Price(
   double  price      // 任意の値
  )
```

パラメータ

価格

[in] double 型の任意の値

戻り値

常に false

© 2000–2025, MetaQuotes Ltd.

Save

オブジェクト パラメータをファイルに保 存します。

```
virtual bool  Save(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] すでに FileOpen 関 数 で開 かれたバイナリファイルのハンド ル

戻 り値

成 功 の場 合 は true、エラーが発 生 した場 合 は false

© 2000–2025, MetaQuotes Ltd.

Load

オブジェクト パラメータをファイルから読み込みます。

```
virtual bool  Load(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] すでに FileOpen 関数で開かれたバイナリファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

Type

グラフィックオブジェクトの型の識別子を取得します。

```
virtual int  Type() const
```

戻り値

オブジェクト型の識別子（OBJ_BITMAP_LABELはCChartObjectBmpLabel）

© 2000–2025, MetaQuotes Ltd.

CChartObjectRectLabel

CChartObjectRectLabel は「Rectangle Label（四角形ラベル）」グラフィックオブジェクトのプロパティに簡単にアクセスするためのクラスです。

説明

CChartObjectRectLabel クラスは「Rectangle Label（四角形ラベル）」グラフィックオブジェクトのプロパティへのアクセスを簡易化します。

宣言

```
class CChartObjectRectLabel : public CChartObjectLabel
```

タイトル

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

継承階層

CObject
  CChartObject
    CChartObjectText
      CChartObjectLabel
        CChartObjectRectLabel

クラスメソッド

Create	
Create	「Rectangle Label（四角形ラベル）」グラフィックオブジェクトを作成します。
プロパティ	
X_Size	水平方向のサイズを設定
Y_Size	垂直方向のサイズを設定
BackColor	背景色を取得/設定
Angle	スタブメソッド
BorderType	境界線の種類を取得/設定
入出力	
virtual Save	ファイル書き込みの仮想メソッド。
virtual Load	ファイル読み込みの仮想メソッド。
virtual Type	識別の仮想メソッド。

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

クラスから継承されたメソッド CChartObjectText

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

クラスから継承されたメソッド CChartObjectLabel

X_Distance, X_Distance, Y_Distance, Y_Distance, X_Size, Y_Size, Corner, Corner, Time, Price, Create

参照

オブジェクトの型、オブジェクトプロパティ、グラフィックオブジェクト

© 2000–2025, MetaQuotes Ltd.

Create

CChartObjectRectLabel グラフィックオブジェクトを作成します。

```
bool  Create(
   long     chart_id,      // チャート識別子
   string   name,          // オブジェクト名
   int      window,        // チャートウィンドウ
   int      X,             // X 座標
   int      Y,             // Y 座標
   int      sizeX,         // 水平方向のサイズ
   int      sizeY          // 垂直方向のサイズ
  )
```

パラメータ

chart_id

[in] チャート識別子（0 は現在のチャート）

name

[in] 作成するオブジェクトの一意の名称

window

[in] チャートウィンドウの番号（0 はメインウィンドウ）

X

[in] X 座標

Y

[in] Y 座標

sizeX

[in] 水平方向のサイズ

sizeY

[in] 垂直方向のサイズ

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

X_Size

「X_Size(X サイズ)」プロパティの値を設定します。

```
bool  X_Size(
   int  size      // 水平方向のサイズ
  )
```

パラメータ

size

[in] 新しい水平方向のサイズ

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

注意事項

「X_Size(X サイズ)」及び「Y_Size(Y サイズ)」プロパティの取得には CChartObjectLabel 親クラスの X_Size 及び Y_Size メソッドが使用されます。

© 2000-2025, MetaQuotes Ltd.

Y_Size

「Y_Size(Y サイズ)」プロパティの値を設定します。

```
bool  Y_Size(
   int  size      // 垂直方向のサイズ
  )
```

パラメータ

size

[in] 新しい垂直方向のサイズ

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

注意事項

「X_Size(X サイズ)」及び「Y_Size(Y サイズ)」プロパティの取得には CChartObjectLabel 親クラスの X_Size 及び Y_Size メソッドが使用されます。

© 2000-2025, MetaQuotes Ltd.

BackColor

背景色を取得します。

```
color  BackColor() const
```

戻り値

クラスインスタンスに結合されたグラフィックオブジェクトの背景色（割り当てられたオブジェクトが存在しない場合は 0）

BackColor

背景色を/設定します。

```
bool  BackColor(
   color  new_color      // 新しい色
  )
```

パラメータ

new_color

[in] 新しい背景色

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Angle

スタブメソッド

```
bool  Angle(
   double  angle       // 任意の値
  )
```

パラメータ

angle

[in] double 型の任意の値

戻り値

常に false

© 2000–2025, MetaQuotes Ltd.

BorderType

境界線の種類を取得します。

```
int  BorderType() const
```

戻り値

クラスインスタンスに結合されたオブジェクト境界線の種類（割り当てられたオブジェクトが存在しない場合は 0）

BorderType

境界線の種類を設定します。

```
bool  BorderType(
   int  type      // 境界線の種類
  )
```

パラメータ

型

[in] 新しい境界線の種類

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Save

オブジェクト パラメータをファイルに保 存します。

```
virtual bool  Save(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] すでに FileOpen 関 数 で開 かれたバイナリファイルのハンド ル

戻 り値

成 功 の場 合 は true、エラーが発 生 した場 合 は false

© 2000–2025, MetaQuotes Ltd.

Load

オブジェクト パラメータをファイルから読み込みます。

```
virtual bool  Load(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] すでに FileOpen 関数で開かれたバイナリファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

Type

グラフィックオブジェクトの型の識別子を取得します。

```
virtual int  Type() const
```

戻り値

オブジェクト型の識別子（OBJ_RECTANGLE_LABELはCChartObjectRectangleLabel）

© 2000–2025, MetaQuotes Ltd.

カスタムグラフィックス

このセクションでは、カスタムグラフィックスを扱うためのツールを提供します。

カスタムチャート、図面、およびデータの視覚化はその使用によって大幅に簡易化されます。

さまざまなタイプの円グラフや曲線の描画やグラフィカルオブジェクトとプリミティブの作成のための個別のクラスが存在します。線のスタイルや色の変更、塗りつぶし、チャート上の一連のデータの操作などの、オブジェクトを表示するためのさまざまなオプションが実装されています。

クラス	説明
CCanvas	カスタムイメージの簡単な作成のためのクラス
CChartCanvas	チャートとその要素を描画するためのクラスを実装するための基本クラス
CHistogramChart	ヒストグラムをプロットするためのクラス
CLineChart	曲線をプロットするためのクラス
CPieChart	円グラフをプロットするためのクラス

© 2000–2025, MetaQuotes Ltd.

CCanvas

CCanvas は、カスタムイメージの作成を簡素化するクラスです。

説明

CCanvas は（チャートオブジェクトに結合するまたはしない）グラフィックリソースの作成及びグラフィックプリミティブの描画を提供します。

宣言

```
class CCanvas
```

タイトル

```
#include <Canvas\Canvas.mqh>
```

継承階層

CCanvas

直接子孫

CChartCanvas, CFlameCanvas

グループ分けされたクラスメソッド

作成	
Attach	OBJ_BITMAP_LABELオブジェクトをCCanvasクラスインスタンスに接続します。
Create	チャートオブジェクトに結合しないグラフィックリソースを作成します。
CreateBitmap	チャートオブジェクトに結合するグラフィックリソースを作成します。
CreateBitmapLabel	チャートオブジェクトに結合するグラフィックリソースを作成します。
Destroy	グラフィックリソースを破壊します。
プロパティ	
ChartObjectName	結合されたチャートオブジェクトの名称を取得します。
ResourceName	グラフィックリソースの名称を取得します。
Width	グラフィックリソースの幅を取得します。
Height	グラフィックリソースの高さを取得します。
LineStyleSet	線のスタイルを設定します。
画面上のオブジェクトを更新します。	
Update	変更を画面に表示します。

© 2000–2025, MetaQuotes Ltd.

作成	
Resize	グラフィックリソースの大きさを変更します。
色の充填/消去	
Erase	指定された色で充填/消去します。
データアクセス	
PixelGet	指定された座標での点の色を取得します。
PixelSet	指定された座標での点の色を設定します。
プリミティブを描画します。	
LineVertical	垂直線を描画します。
LineHorizontal	水平線を描画します。
Line	フリーハンドの線を描画します。
Polyline	ポリラインを描画します。
Polygon	多辺形を描画します。
Rectangle	四角形を描画します。
Circle	円を描画します。
Triangle	三角形を描画します。
Ellipse	楕円を描画します。
Arc	楕円の弧を描きます。
Pie	楕円のセクタを描きます。
塗りつぶされたプリミティブを描画します。	
FillRectangle	塗りつぶされた四角形を描画します。
FillCircle	塗りつぶされた円を描画します。
FillTriangle	塗りつぶされた三角形を描画します。
FillPolygon	塗りつぶされた多角形を描画します。
FillEllipse	塗りつぶされた楕円を描画します。
Fill	エリアを塗りつぶします。
アンチエイリアスでプリミティブを描画します。	
PixelSetAA	ピクセルを描画します。
LineAA	線を引きます。
PolylineAA	ポリラインを描画します。

© 2000–2025, MetaQuotes Ltd.

作成	
PolygonAA	多辺形を描画します。
TriangleAA	三角形を描画します。
CircleAA	円を描画します。
EllipseAA	楕円を描画します。
LineWu	線を引きます。
PolylineWu	ポリラインを描画します。
PolygonWu	多辺形を描画します。
TriangleWu	三角形を描画します。
CircleWu	円を描画します。
EllipseWu	楕円を描画します。
LineThick	アンチエイリアシングアルゴリズムを使用して、指定された幅を持つフリーハンドラインの線分を描画します。
LineThickVertical	アンチエイリアシングアルゴリズムを使用して、指定された幅を持つフリーハンドラインの垂直線分を描画します。
LineThickHorizontal	アンチエイリアシングアルゴリズムを使用して、指定された幅を持つフリーハンドラインの水平線分を描画します。
PolygonSmooth	2つのアンチエイリアシングアルゴリズムを使用して多角形を描画します。
PolygonThick	アンチエイリアシングアルゴリズムを使用して指定された幅を持つ多角形を描画します。
PolylineSmooth	2つのアンチエイリアシングアルゴリズムを使用して指定された幅を持つポリラインを描画します。
PolylineThick	アンチエイリアシングアルゴリズムを使用して指定された幅を持つポリラインを描画します。
テキスト。	
FontSet	フォントパラメータを設定します。
FontNameSet	フォント名を設定します。
FontSizeSet	フォントサイズを設定します。
FontFlagsSet	フォントフラグを設定します。
FontAngleSet	フォントの傾斜角度を設定します。
FontGet	フォントパラメータを取得します。
FontNameGet	ファイル名を取得します。
FontSizeGet	フォントサイズを取得します。

© 2000–2025, MetaQuotes Ltd.

作 成	
FontFlagsGet	フォント フラグを取 得します。
FontAngleGet	フォント の傾 斜 角 度 を取 得します。
TextOut	テキスト を表 示します。
TextWidth	テキスト 幅 を取 得します。
TextHeight	テキスト の高 さを取 得します。
TextSize	テキスト のサイズを取 得します。
透 明 度	
TransparentLevelSet	透 明 度 を設 定します。
入 出 力	
LoadFromFile	BMP ファイルから画 像 を読 み込 みます。

© 2000–2025, MetaQuotes Ltd.

Attach

OBJ_BITMAP_LABELオブジェクト からグラフィックオブジェクト を取 得して CCanvasクラスインスタンスに接 続します。

```
bool  Attach(
   const long          chart_id,                          // チャート識別子
   const string        objname,                           // オブジェクト名
   ENUM_COLOR_FORMAT   clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // 色処理方法
  );
```

OBJ_BITMAP_LABELのためのグラフィックリソースを作 成してCCanvasクラスインスタンスに接 続します。

```
bool  Attach(
   const long          chart_id,                          // チャート識別子
   const string        objname,                           // オブジェクト名
   const int           width,                             // 画像幅(ピクセル単位)
   const int           height,                            // 画像の高さ(ピクセル単位)
   ENUM_COLOR_FORMAT   clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // 色処理方法
  );
```

パラメータ

chart_id

[out] チャート 識 別 子

objname

[in] グラフィックオブジェクト の名 称

幅

[in] リソースの画 像 幅

height

[in] リソースの画 像 の高 さ

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] アルファチャンネル処 理 方 法 。アルファチャンネルはデフォルト では無 視 されます。

戻り値

成 功 の場 合 は true、オブジェクト がシフト できなかった場 合 は false

© 2000-2025, MetaQuotes Ltd.

Arc

角が(x1、y1)と(x2、y2)の四角形に内接する楕円の円弧を描きます。円弧の境界は、楕円の中心から座標(x3、y3)と(x4、y4)の2つの点に延びる線で切り取られます。

```
void  Arc(
   int          x1,       // 四角形の左上隅のX座標
   int          y1,       // 四角形の左上隅のY座標
   int          x2,       // 四角形の右下隅のX座標
   int          y2,       // 四角形の右下隅のY座標
   int          x3,       // 円弧の境界を見つけるための最初の点のX座標
   int          y3,       // 円弧の境界を見つけるための最初の点のY座標
   int          x4,       // 円弧の境界を見つけるための2番目の点のX座標
   int          y4,       // 円弧の境界を見つけるための2番目の点のY座標
   const uint   clr       // 色
   );
```

パラメータ

x1

[in] 四角形の左上隅のX座標

y1

[in] 四角形の左上隅のY座標

x2

[in] 四角形の右下隅のX座標

y2

[in] 四角形の右下隅のY座標

x3

[in] 円弧の境界を取得するために、四角形の中心からの線が描画される最初の点のX座標

y3

[in] 円弧の境界を取得するために、四角形の中心からの線が描画される最初の点のY座標

x4

[in] 円弧の境界を取得するために、四角形の中心からの線が描画される2番目の点のX座標

y4

[in] 円弧の境界を取得するために、四角形の中心からの線が描画される2番目の点のY座標

clr

[in] ARGB形式での色。色をARGB形式に変換するにはColorToARGB()関数を使用します。

半径rxとryを持つ長方形に内接し、点(x、y)を中心とする楕円の円弧を描きます。円弧の境界は、楕円の中心から角度fi3およびfi4によって形成される線によって切り取られます。

```
void  Arc(
   int          x,        // 楕円中心のX座標
```

© 2000-2025, MetaQuotes Ltd.

```
   int          y,        // 楕円中心のY座標
   int          rx,       // X軸上の楕円半径
   int          ry,       // Y軸上の楕円半径
   int          fi3,      // 円弧の最初の境界を定義する楕円中心からの線の角度
   int          fi4,      // 円弧の2番目の境界を定義する楕円中心からの線の角度
   const uint   clr       // 色
   );
```

Draws an arc of an ellipse with center at point (x,y), inscribed in rectangle, with radii rx and ry, and also returns the coordinates of the arc boundaries. 円弧の境界は、楕円の中心から角度fi3およびfi4によって形成される線によって切り取られます。

```
void  Arc(
   int          x,        // 楕円中心のX座標
   int          y,        // 楕円中心のY座標
   int          rx,       // X軸上の楕円半径
   int          ry,       // Y軸上の楕円半径
   int          fi3,      // 円弧の最初の境界を定義する楕円中心からの線の角度
   int          fi4,      // 円弧の2番目の境界を定義する楕円中心からの線の角度
   int&         x3,       // 円弧の最初の境界のX座標
   int&         y3,       // 円弧の最初の境界のY座標
   int&         x4,       // 円弧の2番目の境界のX座標
   int&         y4,       // 円弧の2番目の境界のY座標
   const uint   clr       // 色
   );
```

パラメータ

x

[in] 楕円中心のX座標

y

[in] 楕円中心のY座標

rx

[in] X軸上の楕円半径（ピクセル単位）

ry

[in] Y軸上の楕円半径（ピクセル単位）

fi3

[in] 円弧の最初の境界を定義する角度（ラジアン単位）.

fi4

[in] 円弧の2番目の境界を定義する角度（ラジアン単位）.

x3

[out] 円弧の最初の境界のX座標を取得するための変数

y3

[out] 円弧の最初の境界のY座標を取得するための変数

© 2000–2025, MetaQuotes Ltd.

x4

[out] 円弧の2番目の境界のX座標を取得するための変数

y4

[out] 円弧の2番目の境界のY座標を取得するための変数

clr

[in] ARGB形式での色。色をARGB形式に変換するにはColorToARGB()関数を使用します。

クラスメソッドの呼び出し例：

```
#include <Canvas\Canvas.mqh>
CCanvas canvas;
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
   int      Width=600;
   int      Height=400;
//--- canvasを作成する
   if(!canvas.CreateBitmapLabel(0,0,"CirclesCanvas",30,30,Width,Height))
     {
      Print("Error creating canvas: ",GetLastError());
     }
//--- canvasをクリアする
   canvas.Erase(clrWhite);
//--- 長方形を描画する
   canvas.Rectangle(215-190,215-120,215+190,215+120,clrGray);
//--- 最初の円弧を描く
   canvas.Arc(215,215, 190,120,M_PI_4,2*M_PI-M_PI_4,ColorToARGB(clrRed));
   int x1,y1,x2,y2;
//--- 2番目の円弧を描く
   canvas.Arc(215,215, 190,120,2*M_PI-
M_PI_4,2*M_PI+M_PI_4,x1,y1,x2,y2,ColorToARGB(clrGreen));
//--- 円弧の座標を出力する
   PrintFormat("First point of arc at (%G,%G), second point of arc at (%G,%
G)",x1,y1,x2,y2);
   canvas.CircleAA(x1,y1,3, ColorToARGB(clrRed));
   canvas.CircleAA(x2,y2,3, ColorToARGB(clrBlue));
//--- 更新されたcanvasを表示する
   canvas.Update();
  }
```

© 2000-2025, MetaQuotes Ltd.

Pie

角が(x1、y1)と(x2、y2)の四角形に内接する楕円の塗りつぶされたセクタを描画します。セクタの境界は、楕円の中心から座標(x3、y3)と(x4、y4)の2つの点に延びる線で切り取られます。

```
void  Pie(
   int          x1,       // 四角形の左上隅のX座標
   int          y1,       // 四角形の左上隅のY座標
   int          x2,       // 四角形の右下隅のX座標
   int          y2,       // 四角形の右下隅のY座標
   int          x3,       // 円弧の境界を見つけるための最初の点のX座標
   int          y3,       // 円弧の境界を見つけるための最初の点のY座標
   int          x4,       // 円弧の境界を見つけるための2番目の点のX座標
   int          y4,       // 円弧の境界を見つけるための2番目の点のY座標
   const uint   clr,      // 線の色
   const uint   fill_clr // 塗りつぶしの色
   );
```

パラメータ

x1

[in] 四角形の左上隅のX座標

y1

[in] 四角形の左上隅のY座標

x2

[in] 四角形の右下隅のX座標

y2

[in] 四角形の右下隅のY座標

x3

[in] 円弧の境界を取得するために、四角形の中心からの線が描画される最初の点のX座標

y3

[in] 円弧の境界を取得するために、四角形の中心からの線が描画される最初の点のY座標

x4

[in] 円弧の境界を取得するために、四角形の中心からの線が描画される2番目の点のX座標

y4

[in] 円弧の境界を取得するために、四角形の中心からの線が描画される2番目の点のY座標

clr

[in] ARGB形式での境界の色

fill_clr

[in] ARGB形式でのセクタ塗りつぶしの色色をARGB形式に変換するにはColorToARGB()関数を使用します。

© 2000-2025, MetaQuotes Ltd.

半径rxとryを持つ長方形に内接し、点(x、y)を中心とする楕円のセクタを描きます。セクタの境界は、楕円の中心から角度fi3およびfi4によって形成される線によって切り取られます。

```
void  Pie(
   int          x,        // 楕円中心のX座標
   int          y,        // 楕円中心のY座標
   int          rx,       // X軸上の楕円半径
   int          ry,       // Y軸上の楕円半径
   int          fi3,      // 円弧の最初の境界を定義する楕円中心からの線の角度
   int          fi4,      // 円弧の2番目の境界を定義する楕円中心からの線の角度
   const uint   clr,      // 線の色
   const uint   fill_clr // 塗りつぶしの色
   );
```

半径rxとryを持つ長方形に内接し、点(x、y)を中心とする楕円のセクタを描き、円弧の境界の座標を返します。セクタの境界は、楕円の中心から角度fi3およびfi4によって形成される線によって切り取られます。

```
void  Pie(
   int          x,        // 楕円中心のX座標
   int          y,        // 楕円中心のY座標
   int          rx,       // X軸上の楕円半径
   int          ry,       // Y軸上の楕円半径
   int          fi3,      // 円弧の最初の境界を定義する楕円中心からの線の角度
   int          fi4,      // 円弧の2番目の境界を定義する楕円中心からの線の角度
   int&         x3,       // 円弧の最初の境界のX座標
   int&         y3,       // 円弧の最初の境界のY座標
   int&         x4,       // 円弧の2番目の境界のX座標
   int&         y4,       // 円弧の2番目の境界のY座標
   const uint   clr,      // 線の色
   const uint   fill_clr // 塗りつぶしの色
   );
```

パラメータ

x

[in] 楕円中心のX座標

y

[in] 楕円中心のY座標

rx

[in] X軸上の楕円半径(ピクセル単位)

ry

[in] X軸上の楕円半径(ピクセル単位)

fi3

[in] 円弧の最初の境界を定義する角度(ラジアン単位).

fi4

[in] 円弧の2番目の境界を定義する角度(ラジアン単位).

© 2000-2025, MetaQuotes Ltd.

x3

[out] 円弧の最初の境界のX座標を取得するための変数

y3

[out] 円弧の最初の境界のY座標を取得するための変数

x4

[out] 円弧の2番目の境界のX座標を取得するための変数

y4

[out] 円弧の2番目の境界のY座標を取得するための変数

clr

[in] ARGB形式での境界の色

fill_clr

[in] ARGB形式でのセクタ塗りつぶしの色色をARGB形式に変換するにはColorToARGB()関数を使用します。

クラスメソッドの呼び出し例：

```
#include <Canvas\Canvas.mqh>
CCanvas canvas;
//+------------------------------------------------------------------+
//|  スクリプトプログラムを開始する関数                                |
//+------------------------------------------------------------------+
void OnStart()
  {
   int      Width=600;
   int      Height=400;
//--- canvasを作成する
   if(!canvas.CreateBitmapLabel(0,0,"CirclesCanvas",30,30,Width,Height))
     {
      Print("Error creating canvas: ",GetLastError());
     }
//--- canvasをクリアする
   canvas.Erase(clrWhite);
//--- 長方形を描画する
   canvas.Rectangle(215-190,215-120,215+190,215+120,clrGray);
//--- 最初のパイを描画する
   canvas.Pie(215,215, 190,120,M_PI_4,2*M_PI-
M_PI_4,ColorToARGB(clrBlue),ColorToARGB(clrRed));
//--- 2番目のパイを描画する
   canvas.Pie(215,215, 190,120,2*M_PI-
M_PI_4,2*M_PI+M_PI_4,ColorToARGB(clrGreen),ColorToARGB(clrGreen));
//--- 更新されたcanvasを表示する
   canvas.Update();
   DebugBreak();
  }
```

© 2000-2025, MetaQuotes Ltd.

FillPolygon

塗りつぶされた多角形を描画します。

```
void  FillPolygon(
   int&          x,       // 多角形の頂点のX座標を持つ配列
   int&          y,       // 多角形の頂点のY座標を持つ配列
   const uint    clr      // 色
   );
```

パラメータ

x

[in] 多辺形の頂点のX座標の配列

y

[in] 多辺形の頂点のY座標の配列

clr

[in] ARGB形式での色

© 2000–2025, MetaQuotes Ltd.

FillEllipse

指定された座標を持つ長方形に内接する塗りつぶされた楕円を描画します。

```
void  FillPolygon(
   int          x1,       // 四角形の左上隅のX座標
   int          y1,       // 四角形の左上隅のY座標
   int          x2,       // 四角形の右下隅のX座標
   int          y2,       // 四角形の右下隅のY座標
   const uint   clr        // 楕円の色
   );
```

パラメータ

x1

[in] 四角形の左上隅のX座標

y1

[in] 四角形の左上隅のY座標

x2

[in] 四角形の右下隅のX座標

y2

[in] 四角形の右下隅のY座標

clr

[in] ARGB形式での色

© 2000–2025, MetaQuotes Ltd.

GetDefaultColor

インデックスによって定義済みの色を返します。

```
static uint  GetDefaultColor(
   const uint  i       // インデックス
   );
```

パラメータ

i

[in] 色を取得するインデックス

戻り値

色

© 2000–2025, MetaQuotes Ltd.

ChartObjectName

結合されたチャートオブジェクトの名称を受け取ります。

```
string  ChartObjectName();
```

戻り値

結合されたチャートオブジェクトの名称

© 2000-2025, MetaQuotes Ltd.

Circle

円を描画します。

```
void  Circle(
   int          x,        // X 座標
   int          y,        // Y 座標
   int          r,        // 半径
   const uint   clr       // 色
   );
```

パラメータ

x

[in] 円の中心の X 座標

y

[in] 円の中心の Y 座標

r

[in] 円の半径

clr

[in] ARGB 形式での色

© 2000–2025, MetaQuotes Ltd.

CircleAA

アンチエイリアシングアルゴリズムを使用して円を描画します。

```
void  CircleAA(
   const int      x,        // X 座標
   const int      y,        // Y 座標
   const double   r,        // 半径
   const uint     clr       // 色
   );
```

パラメータ

x

[in] 円の中心の X 座標

y

[in] 円の中心の Y 座標

r

[in] 円の半径

clr

[in] ARGB 形式での色

© 2000–2025, MetaQuotes Ltd.

CircleWu

Wuのアンチエイリアスアルゴリズムを使用して円を描きます。

```
void  CircleWu(
   const int      x,        // X座標
   const int      y,        // Y座標
   const double   r,        // 半径
   const uint     clr       // 色
   );
```

パラメータ

x

[in] 円の中心のX座標

y

[in] 円の中心のY座標

r

[in] 円の半径

clr

[in] ARGB形式での色

© 2000–2025, MetaQuotes Ltd.

Create

チャートオブジェクトに結合しないグラフィックリソースを作成します。

```
virtual bool  Create(
   const string       name,                                  // 名称
   const int          width,                                 // 幅
   const int          height,                                // 高さ
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // フォーマット
   );
```

パラメータ

name

[in] グラフィックリソース名のベースリソース名は、擬似乱数列を追加することにより、作成時に生成されます。

幅

[in] X軸に沿ったピクセル単位での幅

height

[in] Y軸に沿ったピクセル単位での高さ

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] 色処理方法色処理方法についてはResourceCreate() 関数の説明で詳しく知ることが出来ます。

戻り値

成功の場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

CreateBitmap

チャート オブジェクト に結 合 するグラフィックリソースを作 成します。

1. 現 在 のチャート のメインウィンドウにグラフィックリソースを作 成します。

```
bool  CreateBitmap(
   const string       name,                                  // 名称
   const datetime     time,                                  // 時刻
   const double       price,                                 // 価格
   const int          width,                                 // 幅
   const int          height,                                // 高さ
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // フォーマット
   );
```

2. チャート ID とサブウィンドウの番 号を使 用してグラフィックリソースを作 成します。

```
bool  CreateBitmap(
   const long         chart_id,                              // チャート識別子
   const int          subwin,                                // サブウィンドウ番号
   const string       name,                                  // 名称
   const datetime     time,                                  // 時刻
   const double       price,                                 // 価格
   const int          width,                                 // 幅
   const int          height,                                // 高さ
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // フォーマット
   );
```

パラメータ

chart_id

[in] オブジェクト 作 成 のためのチャート 識 別 子

subwin

[in] オブジェクト 作 成 のためのチャート サブウィンドウの番 号

name

[in] チャート オブジェクト 名 称 及 びグラフィックリソース名 のベース

time

[in] チャート オブジェクト のアンカーポイント の時 間 座 標

価格

[in] チャート オブジェクト のアンカーポイント の価 格 座 標

幅

[in] X 軸 に沿 ったピクセル単 位 でのグラフィックリソースの幅

height

[in] Y 軸 に沿 ったピクセル単 位 でのグラフィックリソースの高 さ

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

© 2000–2025, MetaQuotes Ltd.

[in] 色処理方法色処理方法についてはResourceCreate() 関数の説明で詳しく知ることが出来ます。

戻り値

成功の場合は true、それ以外の場合は false

注意事項

1 つ目の関数バージョンが使用された場合、オブジェクトは現在のチャートのメインウィンドウに作成されます。

オブジェクトサイズはグラフィックリソースのサイズと同じです。

© 2000–2025, MetaQuotes Ltd.

CreateBitmapLabel

チャート オブジェクト に結 合 するグラフィックリソースを作 成します。

1. 現 在 のチャート のメインウィンドウにグラフィックリソースを作 成します。

```
bool  CreateBitmapLabel(
   const string       name,                              // 名称
   const int          x,                                 // X 座標
   const int          y,                                 // Y 座標
   const int          width,                             // 幅
   const int          height,                            // 高さ
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // フォーマット
   );
```

2. チャート ID とサブウィンドウの番 号 を使 用してグラフィックリソースを作 成します。

```
bool  CreateBitmapLabel(
   const long         chart_id,                          // チャート識別子
   const int          subwin,                            // サブウィンドウ番号
   const string       name,                              // 名称
   const int          x,                                 // X 座標
   const int          y,                                 // Y 座標
   const int          width,                             // 幅
   const int          height,                            // 高さ
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // フォーマット
   );
```

パラメータ

chart_id

[in] オブジェクト 作 成 のためのチャート 識 別 子

subwin

[in] オブジェクト 作 成 のためのチャート サブウィンドウの番 号

name

[in] チャート オブジェクト 名 称 及 びグラフィックリソース名 のベース

x

[in] チャート オブジェクト のアンカーポイント の X 座 標

y

[in] チャート オブジェクト のアンカーポイント の Y 座 標

幅

[in] X 軸 に沿 ったピクセル単 位 でのグラフィックリソースの幅

height

[in] Y 軸 に沿 ったピクセル単 位 でのグラフィックリソースの高 さ

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

© 2000–2025, MetaQuotes Ltd.

[in] 色処理方法色処理方法についてはResourceCreate() 関数の説明で詳しく知ることが出来ます。

戻り値

成功の場合は true、それ以外の場合は false

注意事項

1 つ目の関数バージョンが使用された場合、オブジェクトは現在のチャートのメインウィンドウに作成されます。

オブジェクトサイズはグラフィックリソースのサイズと同じです。

© 2000–2025, MetaQuotes Ltd.

Destroy

グラフィックリソースを破 壊します。

```
void  Destroy();
```

注 意 事 項

グラフィックリソースがチャート オブジェクト に結 合 されている場 合 チャート オブジェクト が削 除 されます。

© 2000–2025, MetaQuotes Ltd.

Ellipse

2つの点に基づいて楕円を描きます。

```
void  Ellipse(
   int          x1,      // X座標
   int          y1,      // Y座標
   int          x2,      // X座標
   int          y2,      // Y座標
   const uint   clr      // 色
   );
```

パラメータ

x1

[in] 楕円を形成する1番目の点のX座標

y1

[in] 楕円を形成する1番目の点のY座標

x2

[in] 楕円を形成する2番目の点のX座標

y2

[in] 楕円を形成する2番目の点のY座標

clr

[in] ARGB形式での色

© 2000–2025, MetaQuotes Ltd.

EllipseAA

アンチエイリアスアルゴリズムを使用して2つの点に基づいて楕円を描きます。

```
void  EllipseAA(
   int          x1,      // X座標
   int          y1,      // Y座標
   int          x2,      // X座標
   int          y2,      // Y座標
   const uint   clr      // 色
   );
```

パラメータ

x1

[in] 楕円を形成する1番目の点のX座標

y1

[in] 楕円を形成する1番目の点のY座標

x2

[in] 楕円を形成する2番目の点のX座標

y2

[in] 楕円を形成する2番目の点のY座標

clr

[in] ARGB形式での色

© 2000–2025, MetaQuotes Ltd.

EllipseWu

Wuのアンチエイリアスアルゴリズムを使用して2つの点に基づいて楕円を描きます。

```
void  EllipseWu(
   int          x1,       // X座標
   int          y1,       // Y座標
   int          x2,       // X座標
   int          y2,       // Y座標
   const uint   clr       // 色
   );
```

パラメータ

x1

[in] 楕円を形成する1番目の点のX座標

y1

[in] 楕円を形成する1番目の点のY座標

x2

[in] 楕円を形成する2番目の点のX座標

y2

[in] 楕円を形成する2番目の点のY座標

clr

[in] ARGB形式での色

© 2000-2025, MetaQuotes Ltd.

Erase

指定された色で消去します。

```
void  Erase(
   const uint  clr=0      // 色
   );
```

パラメータ

clr=0

[in] ARGB形式での色

© 2000–2025, MetaQuotes Ltd.

Fill

エリアを塗りつぶします。

```
void  Fill(
   int          x,        // X 座標
   int          y,        // Y 座標
   const uint   clr       // 色
   );
```

パラメータ

x

[in] 塗りつぶしが始まる X 座標

y

[in] 塗りつぶしが始まる Y 座標

clr

[in] ARGB 形式での色

© 2000–2025, MetaQuotes Ltd.

FillCircle

塗りつぶされた円を描画します。

```
void  FillCircle(
   int          x,        // X 座標
   int          y,        // Y 座標
   int          r,        // 半径
   const uint   clr       // 色
   );
```

パラメータ

x

[in] 塗りつぶされた円の中心の X 座標

y

[in] 塗りつぶされた円の中心の Y 座標

r

[in] 塗りつぶされた円の半径

clr

[in] ARGB 形式での色

© 2000–2025, MetaQuotes Ltd.

FillRectangle

塗りつぶされた四角形を描画します。

```
void  FillRectangle(
   int          x1,      // X 座標
   int          y1,      // Y 座標
   int          x2,      // X 座標
   int          y2,      // Y 座標
   const uint   clr      // 色
   );
```

パラメータ

x1

[in] 四角形を形成する1番目の点のX座標

y1

[in] 四角形を形成する1番目の点のY座標

x2

[in] 四角形を形成する2番目の点のX座標

y2

[in] 四角形を形成する2番目の点のY座標

clr

[in] ARGB形式での色

© 2000–2025, MetaQuotes Ltd.

FillTriangle

塗りつぶされた三角形を描画します。

```
void  FillTriangle(
   int          x1,      // X 座標
   int          y1,      // Y 座標
   int          x2,      // X 座標
   int          y2,      // Y 座標
   int          x3,      // X 座標
   int          y3,      // Y 座標
   const uint   clr      // 色
   );
```

パラメータ

x1

[in] 三角形の1 番目の頂点のX 座標

y1

[in] 三角形の1 番目の頂点のY 座標

x2

[in] 三角形の2 番目の頂点のX 座標

y2

[in] 三角形の2 番目の頂点のY 座標

x3

[in] 三角形の3 番目の頂点のX 座標

y3

[in] 三角形の3 番目の頂点のY 座標

clr

[in] ARGB 形式での色

© 2000–2025, MetaQuotes Ltd.

FontAngleGet

フォントの傾斜角度を受け取ります。

```
uint  FontAngleGet();
```

戻り値

フォントの傾斜角度

© 2000–2025, MetaQuotes Ltd.

FontAngleSet

フォントの傾斜角度を設定します。

```
bool  FontAngleSet(
   uint  angle      // 角度
   );
```

パラメータ

angle

[in] 0.1 度単位でのフォントの傾斜角度

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

FontFlagsGet

フォント フラグを受け取ります。

```
uint  FontFlagsGet();
```

戻り値

フォント フラグ

© 2000–2025, MetaQuotes Ltd.

FontFlagsSet

フォントフラグを設定します。

```
bool  FontFlagsSet(
   uint  flags      // フラグ
   );
```

パラメータ

flags

[in] フォント作成フラグフラグについて詳しく知るには TextSetFont() 関数の説明をご覧ください。

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

FontGet

現在のフォントパラメータを受け取ります。

```
void  FontGet(
   string&  name,      // 名称
   int&     size,      // サイズ
   uint&    flags,     // フラグ
   uint&    angle      // 傾斜角度
   );
```

パラメータ

name

[out] フォント名を返すための変数への参照

size

[out] フォントサイズを返すための変数への参照

flags

[out] フォントフラグを返すための変数への参照

angle

[out] フォント傾斜角度を返すための変数への参照

© 2000–2025, MetaQuotes Ltd.

FontNameGet

フォント名を受け取ります。

```
string  FontNameGet();
```

戻り値

フォント名

© 2000–2025, MetaQuotes Ltd.

FontNameSet

フォント名を設定します。

```
bool  FontNameSet(
   string  name      // 名称
   );
```

パラメータ

name

[in] フォント名 例えば "Arial"

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

FontSet

現在のフォントを設定します。

```
bool  FontSet(
   const string  name,         // 名称
   const int     size,         // サイズ
   const uint    flags=0,      // フラグ
   const uint    angle=0       // 角度
   );
```

パラメータ

name

[in] フォント名例えば "Arial"

size

[in] フォントサイズサイズ設定について詳しく知るには TextSetFont() 関数の説明をご覧ください。

flags=0

[in] フォント作成フラグフラグについて詳しく知るには TextSetFont() 関数の説明をご覧ください。

angle=0

[in] 0.1 度単位でのフォントの傾斜角度

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

FontSizeGet

フォント サイズを受け取ります。

```
int  FontSizeGet();
```

戻り値

フォント サイズ

© 2000-2025, MetaQuotes Ltd.

FontSizeSet

フォント サイズを設 定します。

```
bool  FontSizeSet(
   int  size      // サイズ
   );
```

パラメータ

size

[in] フォント サイズサイズ設 定について詳しく知 るには TextSetFont() 関 数 の説 明をご覧ください。

戻り値

成 功 の場 合 は true、それ以 外 の場 合 は false

© 2000–2025, MetaQuotes Ltd.

Height

グラフィックリソースの高さを受け取ります。

```
int  Height();
```

戻り値

グラフィックリソースの高さ

© 2000-2025, MetaQuotes Ltd.

Line

フリーハンドの線のセグメントを描画します。

```
void  Line(
   int          x1,      // X 座標
   int          y1,      // Y 座標
   int          x2,      // X 座標
   int          y2,      // Y 座標
   const uint   clr      // 色
   );
```

パラメータ

x1

[in] セグメントの1番目の点の X 座標

y1

[in] セグメントの1番目の点の Y 座標

x2

[in] セグメントの2番目の点の X 座標

y2

[in] セグメントの2番目の点の Y 座標

clr

[in] ARGB 形式での色

© 2000–2025, MetaQuotes Ltd.

LineAA

アンチエイリアシングアルゴリズムを使用してフリーハンドの線のセグメントを描画します。

```
void  LineAA(
   const int   x1,                  // X 座標
   const int   y1,                  // Y 座標
   const int   x2,                  // X 座標
   const int   y2,                  // Y 座標
   const uint  clr,                 // 色
   const uint  style=UINT_MAX       // 線のスタイル
   );
```

パラメータ

x1

[in] セグメントの1番目の点のX座標

y1

[in] セグメントの1番目の点のY座標

x2

[in] セグメントの2番目の点のX座標

y2

[in] セグメントの2番目の点のY座標

clr

[in] ARGB形式での色

style=UINT_MAX

[in] 線のスタイルはENUM_LINE_STYLE列挙値またはカスタム値

© 2000–2025, MetaQuotes Ltd.

LineWu

Wuのアンチエイリアスアルゴリズムを使用してフリーハンドで線分を描画します。

```
void  LineWu(
   const int   x1,                  // X 座標
   const int   y1,                  // Y 座標
   const int   x2,                  // X 座標
   const int   y2,                  // Y 座標
   const uint  clr,                 // 色
   const uint  style=UINT_MAX       // 線のスタイル
   );
```

パラメータ

x1

[in] セグメントの1番目の点のX座標

y1

[in] セグメントの1番目の点のY座標

x2

[in] セグメントの2番目の点のX座標

y2

[in] セグメントの2番目の点のY座標

clr

[in] ARGB形式での色

style=UINT_MAX

[in] 線のスタイルはENUM_LINE_STYLE列挙値またはカスタム値

© 2000–2025, MetaQuotes Ltd.

LineHorizontal

水平線のセグメントを描画します。

```
void  LineHorizontal(
   int          x1,       // X 座標
   int          x2,       // X 座標
   int          y,        // Y 座標
   const uint   clr       // 色
   );
```

パラメータ

x1

[in] セグメントの1番目の点のX座標

x2

[in] セグメントの2番目の点のX座標

y

[in] セグメントのY座標

clr

[in] ARGB形式での色

© 2000–2025, MetaQuotes Ltd.

LineVertical

垂直線のセグメントを描画します。

```
void  LineVertical(
   int          x,        // X 座標
   int          y1,       // Y 座標
   int          y2,       // Y 座標
   const uint   clr       // 色
   );
```

パラメータ

x

[in] セグメントの X 座標

y1

[in] セグメントの1 番目の点の Y 座標

y2

[in] セグメントの2 番目の点の Y 座標

clr

[in] ARGB 形式での色

© 2000–2025, MetaQuotes Ltd.

LineStyleSet

線のスタイルを設定します。

```
void  LineStyleSet(
   const uint  style      // スタイル
   );
```

パラメータ

style

[in] 線のスタイル

注意事項

入力パラメータは ENUM_LINE_STYLE 列挙値のいずれかです。その上、カスタム線描画スタイルを作成することも可能です。

© 2000–2025, MetaQuotes Ltd.

LineThick

アンチエイリアシングアルゴリズムを使用して、指定された幅を持つフリーハンドラインの線分を描画します。

```
void  LineThick(
   const int       x1,            // 線分の1  番目の点のX座標
   const int       y1,            // 線分の1番目の点のY座標
   const int       x2,            // 線分の2番目の点のX座標
   const int       y2,            // 線分の2番目の点のY座標
   const uint      clr,           // 色
   const int       size,          // 線の幅
   const uint      style,         // 線のスタイル
   ENUM_LINE_END   end_style      // 線の端のスタイル
  )
```

パラメータ

x1

[in] セグメントの1番目の点のX座標

y1

[in] セグメントの1番目の点のY座標

x2

[in] セグメントの2番目の点のX座標

y2

[in] セグメントの2番目の点のY座標

clr

[in] ARGB形式での色

size

[in] 線の幅

style

[in] 線のスタイルはENUM_LINE_STYLE列挙値またはカスタム値</t2>

end_style

[in] 線のスタイルはENUM_LINE_END列挙値

ENUM_LINE_END

ID	説明
LINE_END_ROUND	線の端は丸められています。
LINE_END_BUTT	線の端は切られています。
LINE_END_SQUARE	線は塗りつぶされた四角形で終わります。

© 2000–2025, MetaQuotes Ltd.

LineThickVertical

アンチエイリアシングアルゴリズムを使用して、指定された幅を持つフリーハンドラインの垂直線分を描画します。

```
void  LineThickVertical(
   const int       x,              // 線分の1番目の点のX座標<
   const int       y1,             // 線分の1番目の点のY座標
   const int       y2,             // 線分の2番目の点のY座標
   const uint      clr,            // 色
   const int       size,           // 線の幅
   const uint      style,          // 線のスタイル
   ENUM_LINE_END   end_style       // 線の端のスタイル
  )
```

パラメータ

x

[in] セグメントのX座標

y1

[in] セグメントの1番目の点のY座標

y2

[in] セグメントの2番目の点のY座標

clr

[in] ARGB形式での色

size

[in] 線の幅

style

[in] 線のスタイルはENUM_LINE_STYLE列挙値またはカスタム値</t2>

end_style

[in] 線のスタイルはENUM_LINE_END列挙値

© 2000-2025, MetaQuotes Ltd.

LineThickHorizontal

アンチエイリアシングを使用して、指定された幅を持つフリーハンドラインの水平線分を描画します。

```
void  LineThickHorizontal(
   const int       x1,             // 線分の1  番目の点のX座標
   const int       x2,             // 線分の2番目の点のX座標
   const int       y,              // 線分の1  番目の点のY座標
   const uint      clr,            // 色
   const int       size,           // 線の幅
   const uint      style,          // 線のスタイル
   ENUM_LINE_END   end_style       // 線の端のスタイル
  )
```

パラメータ

x1

[in] セグメントの1番目の点のX座標

x2

[in] セグメントの2番目の点のX座標

y

[in] セグメントのY座標

clr

[in] ARGB形式での色

size

[in] 線の幅

style

[in] 線のスタイルはENUM_LINE_STYLE列挙値またはカスタム値</t2>

end_style

[in] 線のスタイルはENUM_LINE_END列挙値

© 2000-2025, MetaQuotes Ltd.

LoadFromFile

BMP ファイルから画像を読み込みます。

```
bool  LoadFromFile(
   const string  filename      // ファイル名
   );
```

パラメータ

filename

[in] ファイル名（「BMP」拡張子を含む）

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

PixelGet

指定された座標での点の色を受け取ります。

```
uint  PixelGet(
   const int  x,      // X 座標
   const int  y       // Y 座標
   );
```

パラメータ

x

[in] 点のX座標

y

[in] 点のY座標

戻り値

ARGB形式での点の色

© 2000–2025, MetaQuotes Ltd.

PixelSet

指定された座標での点の色を設定します。

```
void  PixelSet(
   const int   x,        // X 座標
   const int   y,        // Y 座標
   const uint  clr       // 色
   );
```

パラメータ

x

[in] 点のX座標

y

[in] 点のY座標

clr

[in] ARGB形式での色

© 2000–2025, MetaQuotes Ltd.

PixelSetAA

アンチエイリアシングアルゴリズムを使用して点を描画します。

```
void  PixelSetAA(
   const double  x,        // X 座標
   const double  y,        // Y 座標
   const uint    clr       // 色
   );
```

パラメータ

x

[in] 点のX座標

y

[in] 点のY座標

clr

[in] ARGB形式での色

© 2000–2025, MetaQuotes Ltd.

Polygon

多辺形を描画します。

```
void  Polygon(
   int&         x[],     // X 座標の配列
   int&         y[],     // Y 座標の配列
   const uint   clr      // 色
   );
```

パラメータ

x[]

[in] 多辺形の頂点の X 座標の配列

y[]

[in] 多辺形の頂点の Y 座標の配列

clr

[in] ARGB 形式での色

© 2000–2025, MetaQuotes Ltd.

PolygonAA

アンチエイリアシングアルゴリズムを使用してポリラインを描画します。

```
void  PolygonAA(
   int&         x[],                 // X 座標の配列
   int&         y[],                 // Y 座標の配列
   const uint   clr,                 // 色
   const uint   style=UINT_MAX       // 線のスタイル
   );
```

パラメータ

x[]

[in] 多辺形の頂点の X 座標の配列

y[]

[in] 多辺形の頂点の Y 座標の配列

clr

[in] ARGB 形式での色

style=UINT_MAX

[in] 線のスタイルはENUM_LINE_STYLE 列挙値またはカスタム値

© 2000–2025, MetaQuotes Ltd.

PolygonWu

Wuのアンチエイリアスアルゴリズムを使用して多辺形を描きます。

```
void  PolygonWu(
   int&         x[],                 // X 座標の配列
   int&         y[],                 // Y 座標の配列
   const uint   clr,                 // 色
   const uint   style=UINT_MAX       // 線のスタイル
   );
```

パラメータ

x[]

[in] 多辺形の頂点のX座標の配列

y[]

[in] 多辺形の頂点のY座標の配列

clr

[in] ARGB形式での色

style=UINT_MAX

[in] 線のスタイルはENUM_LINE_STYLE列挙値またはカスタム値

© 2000–2025, MetaQuotes Ltd.

PolygonThick

アンチエイリアシングアルゴリズムを使用して指定された幅を持つ多角形を描画します。

```
void  PolygonThick(
   const int&      x[],          // 多角形の頂点のX座標を持つ配列
   const int&      y[],          // 多角形の頂点のY 座標を持つ配列
   const uint      clr,          // 色
   const int       size,         // 線の幅
   const uint      style,        // 線のスタイル
   ENUM_LINE_END   end_style     // 線の端のスタイル
  )
```

パラメータ

x[]

[in] 多辺形の頂点のX座標の配列

y[]

[in] 多辺形の頂点のY座標の配列

clr

[in] ARGB形式での色

size

[in] 線の幅

style

[in] 線のスタイルはENUM_LINE_STYLE列挙値またはカスタム値</t2>

end_style

[in] 線のスタイルはENUM_LINE_END列挙値

© 2000–2025, MetaQuotes Ltd.

PolygonSmooth

2つのアンチエイリアシングアルゴリズムを使用して、指定された幅のポリゴンを連続して描画します。まず、個々の線分がベジェ曲線に基づいて平滑化されます。次に、レンダリング品質を向上するために、これらの線分から構築されたポリゴンにラスタアンチエイリアシングアルゴリズムを適用します。

```
void  PolygonSmooth(
   int&             x[],                    // 多角形の頂点のX座標を持つ配列
   int&             y[],                    // 多角形の頂点のY座標を持つ配列
   const uint       clr,                    // 色
   const int        size,                   // 線の幅
   ENUM_LINE_STYLE  style=STYLE_SOLID,      // 線のスタイル
   ENUM_LINE_END    end_style=LINE_END_ROUND,  // 線の端のスタイル
   double           tension=0.5,            // アンチエイリアスパラメータ値
   double           step=10                 // 近似線の長さ
  )
```

パラメータ

&x[]

[in] 多辺形の頂点のX座標の配列

&y[]

[in] 多辺形の頂点のY座標の配列

clr

[in] ARGB形式での色

size

[in] 線の幅

style=STYLE_SOLID

[in] 線のスタイルはENUM_LINE_STYLE列挙値またはカスタム値</t2>

end_style=LINE_END_ROUND

[in] 線のスタイルはENUM_LINE_END列挙値

tension=0.5

[in] 平滑化パラメータ値

step=10

[in] 近似線の長さ

© 2000-2025, MetaQuotes Ltd.

Polyline

ポリラインを描画します。

```
void  Polyline(
   int&         x[],     // X 座標の配列
   int&         y[],     // Y 座標の配列
   const uint   clr      // 色
   );
```

パラメータ

x[]

[in] ポリラインの X 座標の配列

y[]

[in] ポリラインの Y 座標の配列

clr

[in] ARGB 形式での色

© 2000–2025, MetaQuotes Ltd.

PolylineSmooth

2つのアンチエイリアシングアルゴリズムを使用して指定された幅を持つポリラインを描画します。まず、個々の線分がベジェ曲線に基づいて平滑化されます。次に、レンダリング品質を向上するために、これらの線分から構築されたポリゴンにラスタアンチエイリアシングアルゴリズムを適用します。

```
void  PolylineSmooth(
   const int&        x[],                    // ポリライン点のX座標を持つ配列
   const int&        y[],                    // ポリライン点のY座標を持つ配列
   const uint        clr,                    // 色
   const int         size,                   // 線の幅
   ENUM_LINE_STYLE   style=STYLE_SOLID,      // 線のスタイル
   ENUM_LINE_END     end_style=LINE_END_ROUND,  // 線の端のスタイル
   double            tension=0.5,            // アンチエイリアスパラメータ値
   double            step=10                 // 近似ステップ
  )
```

パラメータ

&x[]

[in] ポリラインのX座標の配列

&y[]

[in] ポリラインのY座標の配列

clr

[in] ARGB形式での色

size

[in] 線の幅

style=STYLE_SOLID

[in] 線のスタイルはENUM_LINE_STYLE列挙値またはカスタム値</t2>

end_style=LINE_END_ROUND

[in] 線のスタイルはENUM_LINE_END列挙値

tension=0.5

[in] 平滑化パラメータ値

step=10

[in] 近似値ステップ

© 2000-2025, MetaQuotes Ltd.

PolylineThick

アンチエイリアシングアルゴリズムを使用して指定された幅を持つポリラインを描画します。

```
void  PolylineThick(
   const int       &x[],          // ポリライン点のX座標を持つ配列
   const int       &y[],          // ポリライン点のY座標を持つ配列
   const uint      clr,           // 色
   const int       size,          // 線の幅
   const uint      style,         // 線のスタイル
   ENUM_LINE_END   end_style      // 線の端のスタイル
  )
```

パラメータ

&x[]

[in] ポリラインのX座標の配列

&y[]

[in] ポリラインのY座標の配列

clr

[in] ARGB形式での色

size

[in] 線の幅

style

[in] 線のスタイルはENUM_LINE_STYLE列挙値またはカスタム値</t2>

end_style

[in] 線のスタイルはENUM_LINE_END<t2>列挙値

© 2000–2025, MetaQuotes Ltd.

PolylineWu

Wuのアンチエイリアスアルゴリズムを使用してポリラインを描きます。

```
void  PolylineWu(
   int&         x[],                 // X 座標の配列
   int&         y[],                 // Y 座標の配列
   const uint   clr,                 // 色
   const uint   style=UINT_MAX       // 線のスタイル
   );
```

パラメータ

x[]

[in] ポリラインのX座標の配列

y[]

[in] ポリラインのY座標の配列

clr

[in] ARGB形式での色

style=UINT_MAX

[in] 線のスタイルはENUM_LINE_STYLE列挙値またはカスタム値

© 2000–2025, MetaQuotes Ltd.

PolylineAA

アンチエイリアシングアルゴリズムを使用してポリラインを描画します。

```
void  PolylineAA(
   int&         x[],                 // X 座標の配列
   int&         y[],                 // Y 座標の配列
   const uint   clr,                 // 色
   const uint   style=UINT_MAX       // 線のスタイル
   );
```

パラメータ

x[]

[in] ポリラインの X 座標の配列

y[]

[in] ポリラインの Y 座標の配列

clr

[in] ARGB 形式での色

style=UINT_MAX

[in] 線のスタイルはENUM_LINE_STYLE 列挙値またはカスタム値

© 2000–2025, MetaQuotes Ltd.

Rectangle

2 つの点を使用して四角形を描画します。

```
void  Rectangle(
   int          x1,      // X 座標
   int          y1,      // Y 座標
   int          x2,      // X 座標
   int          y2,      // Y 座標
   const uint   clr      // 色
   );
```

パラメータ

x1

[in] 四角形を形成する1 番目の点の X 座標

y1

[in] 四角形を形成する1 番目の点の Y 座標

x2

[in] 四角形を形成する2 番目の点の X 座標

y2

[in] 四角形を形成する2 番目の点の Y 座標

clr

[in] ARGB 形式での色

© 2000–2025, MetaQuotes Ltd.

Resize

グラフィックリソースの大きさを変えます。

```
bool  Resize(
   const int  width,       // 幅
   const int  height       // 高さ
   );
```

パラメータ

幅

[in] グラフィックリソースの新しい幅

height

[in] グラフィックリソースの新しい高さ

戻り値

成功の場合は true、それ以外の場合は false

注意事項

大きさを変更する際、依然の画像は保存されません。

© 2000–2025, MetaQuotes Ltd.

ResourceName

グラフィックリソースの名 称を受 け取 ります。

```
string  ResourceName();
```

戻 り値

グラフィックリソースの名 称

© 2000–2025, MetaQuotes Ltd.

TextHeight

テキストの高さを受け取ります。

```
int  TextHeight(
   const string  text      // テキスト
   );
```

パラメータ

text

[in] 図るためのテキスト

戻り値

ピクセル単位でのテキストの高さ

注意事項

現在のフォントはテキストを図るのに使用されます。

© 2000–2025, MetaQuotes Ltd.

TextOut

テキストを表示します。

```
void  TextOut(
   int          x,                // X 座標
   int          y,                // Y 座標
   string       text,             // テキスト
   const uint   clr,              // 色
   uint         alignment=0       // 位置合わせ
   );
```

パラメータ

x

[in] テキストのアンカーポイントの X 座標

y

[in] テキストのアンカーポイントの Y 座標

text

[in] 表示されるテキスト

clr

[in] ARGB 形式での色

alignment=0

[in] テキストアンカー方法 アンカー方法については TextOut() 関数の説明で詳しく知ることが出来ます。

注意事項

テキスト表示には現在のフォントが使用されます。

© 2000–2025, MetaQuotes Ltd.

TextSize

テキスト サイズを受け取ります。

```
void  TextSize(
   const string  text,        // テキスト
   int&          width,       // 幅
   int&          height       // 高さ
   );
```

パラメータ

text

[in] 図るためのテキスト

幅

[out] テキスト幅を返すための変数への参照

height

[out] テキストの高さを返すための変数への参照.

注意事項

現在のフォントはテキストを図るのに使用されます。

© 2000–2025, MetaQuotes Ltd.

TextWidth

テキスト幅を受け取ります。

```
int  TextWidth(
   const string  text      // テキスト
   );
```

パラメータ

text

[in] 図るためのテキスト

戻り値

ピクセル単位でのテキストの高さ

注意事項

現在のフォントはテキストを図るのに使用されます。

© 2000–2025, MetaQuotes Ltd.

TransparentLevelSet

透明度を設定します。

```
void  TransparentLevelSet(
   const uchar  value      // 値
   );
```

パラメータ

value

[in] 透明度の新しい値

注意事項

0 は完全な透明性、255 は完全な不透明性を表します。

透明度の設定は、以前に描かれたもの全てに影響します。指定された透明度レベルは更に構造に影響を与えません。

© 2000–2025, MetaQuotes Ltd.

Triangle

三角形を描画します。

```
void  Triangle(
   int          x1,      // X 座標
   int          y1,      // Y 座標
   int          x2,      // X 座標
   int          y2,      // Y 座標
   int          x3,      // X 座標
   int          y3,      // Y 座標
   const uint   clr      // 色
   );
```

パラメータ

x1

[in] 三角形の1 番目の頂点のX 座標

y1

[in] 三角形の1 番目の頂点のY 座標

x2

[in] 三角形の2 番目の頂点のX 座標

y2

[in] 三角形の2 番目の頂点のY 座標

x3

[in] 三角形の3 番目の頂点のX 座標

y3

[in] 三角形の3 番目の頂点のY 座標

clr

[in] ARGB 形式での色

© 2000–2025, MetaQuotes Ltd.

TriangleAA

アンチエイリアシングアルゴリズムを使用して三角形を描画します。

```
void  TriangleAA(
   const int   x1,                  // X 座標
   const int   y1,                  // Y 座標
   const int   x2,                  // X 座標
   const int   y2,                  // Y 座標
   const int   x3,                  // X 座標
   const int   y3,                  // Y 座標
   const uint  clr,                 // 色
   const uint  style=UINT_MAX       // 線のスタイル
   );
```

パラメータ

x1

[in] 三角形の1番目の頂点のX座標

y1

[in] 三角形の1番目の頂点のY座標

x2

[in] 三角形の2番目の頂点のX座標

y2

[in] 三角形の2番目の頂点のY座標

x3

[in] 三角形の3番目の頂点のX座標

y3

[in] 三角形の3番目の頂点のY座標

clr

[in] ARGB形式での色

style=UINT_MAX

[in] 線のスタイルはENUM_LINE_STYLE列挙値またはカスタム値

© 2000-2025, MetaQuotes Ltd.

TriangleWu

Wuのアンチエイリアスアルゴリズムを使用して三角形を描きます。

```
void  TriangleWu(
   const int   x1,                  // X 座標
   const int   y1,                  // Y 座標
   const int   x2,                  // X 座標
   const int   y2,                  // Y 座標
   const int   x3,                  // X 座標
   const int   y3,                  // Y 座標
   const uint  clr,                 // 色
   const uint  style=UINT_MAX       // 線のスタイル
   );
```

パラメータ

x1

[in] 三角形の1番目の頂点のX座標

y1

[in] 三角形の1番目の頂点のY座標

x2

[in] 三角形の2番目の頂点のX座標

y2

[in] 三角形の2番目の頂点のY座標

x3

[in] 三角形の3番目の頂点のX座標

y3

[in] 三角形の3番目の頂点のY座標

clr

[in] ARGB形式での色

style=UINT_MAX

[in] 線のスタイルはENUM_LINE_STYLE列挙値またはカスタム値

© 2000-2025, MetaQuotes Ltd.

Update

変更を画面に表示します。

```
void  Update(
   const bool  redraw=true      // フラグ
   );
```

パラメータ

redraw=true

チャートの再描画の必要性を示すフラグ

© 2000–2025, MetaQuotes Ltd.

Width

グラフィックリソースの幅を受け取ります。

```
int  Width();
```

戻り値

グラフィックリソースの幅

© 2000–2025, MetaQuotes Ltd.

CChartCanvas

チャートとその要素を描画するためのクラスを実装するための基本クラス

説明

このクラスには、座標軸とそのマーク、チャートの凡例、グリッド、背景などのチャートの基本要素を操作するためのメソッドが含まれています。ここでは、要素を表示するためのオプション(可視性、テキストの色など)をカスタマイズできます。

宣言

```
class CChartCanvas : public CCanvas
```

タイトル

```
#include <Canvas\Charts\ChartCanvas.mqh>
```

継承階層

CCanvas

CChartCanvas

直接子孫

CHistogramChart, CLineChart, CPieChart

クラスメソッド

メソッド	アクション
ColorBackground	背景色を返します/設定します。
ColorBorder	境界色を返します/設定します。
ColorText	テキストの色を返します/設定します。
ColorGrid	グリッドの色を返します/設定します。
MaxData	許容されるデータ(系列)の最大量を返します/設定します。
MaxDescrLen	記述子の最大長を返します/設定します。
ShowFlags	チャート要素の可視性フラグを返します/設定します。
IsShowLegend	チャート上の凡例の可視性フラグを返します/設定します。
IsShowScaleLeft	左側の値のスケールの可視性フラグを返します。
IsShowScaleRight	右側の値のスケールの可視性フラグを返します。

© 2000-2025, MetaQuotes Ltd.

メソッド	アクション
IsShowScaleTop	上側の値のスケールの可視性フラグを返します。
IsShowScaleBottom	下側の値のスケールの可視性フラグを返します。
IsShowGrid	チャート上のグリッドの可視性フラグを返します。
IsShowDescriptors	チャート上の記述子の可視性フラグを返します。
IsShowPercent	チャート上のパーセンテージの可視性フラグを返します。
VScaleMin	値の垂直スケールの最小値を返します/設定します。
VScaleMax	値の垂直スケールの最大値を返します/設定します。
NumGrid	チャートグリッドをプロットするときの縦軸の分割数を返します/設定します。
DataOffset	データのオフセット値を返します/設定します。
DataTotal	チャート上のデータ系列の総数を返します。
DrawDescriptors	記述子を描画するための仮想メソッド。
DrawData	指定されたインデックスでデータ系列を描画するための仮想メソッド。
Create	グラフィカルリソースを作成する仮想メソッド。
AllowedShowFlags	チャート要素に許可された可視性フラグのセットを設定します。
ShowLegend	凡例の可視性フラグを設定します。
ShowScaleLeft	左のスケールの可視性フラグを設定します。
ShowScaleRight	右のスケールの可視性フラグを設定します。
ShowScaleTop	上のスケールの可視性フラグを設定します。
ShowScaleBottom	下のスケールの可視性フラグを設定します。
ShowGrid	グリッドの可視性フラグを設定します。
ShowDescriptors	記述子の可視性フラグを設定します。
ShowValue	値の可視性フラグを設定します。
ShowPercent	パーセンテージの可視性フラグを設定します。
LegendAlignment	凡例のテキストの配置を設定します。.

© 2000–2025, MetaQuotes Ltd.

メソッド	アクション
Accumulative	シリーズの値の累積フラグを設定します。
VScaleParams	値の垂直スケールのパラメータを設定します。
DescriptorUpdate	指定された位置の系列記述子の値を更新します。
ColorUpdate	指定された位置の系列の色を更新します。
ValuesCheck	チャートをプロットするための内部計算を実行します。
Redraw	チャートを再描画します。
DrawBackground	背景を描画します。
DrawLegend	凡例を再描画します。
DrawLegendVertical	垂直の凡例を描画します。
DrawLegendHorizontal	水平の凡例を描画します。
CalcScales	スケールの座標を計算します。
DrawScales	値のすべてのスケールを再描画します。
DrawScaleLeft	値の左のスケールを再描画します。
DrawScaleRight	値の右のスケールを再描画します。
DrawScaleTop	値の上のスケールを再描画します。
DrawScaleBottom	値の下のスケールを再描画します。
DrawGrid	チャートを再描画します。
DrawChart	チャートを再描画します。

クラスから継承されたメソッド CCanvas

CreateBitmap, CreateBitmap, CreateBitmapLabel, CreateBitmapLabel, Attach, Attach, Destroy, ChartObjectName, ResourceName, Width, Height, Update, Resize, Erase, PixelGet, PixelSet, LineVertical, LineHorizontal, Line, Polyline, Polygon, Rectangle, Triangle, Circle, Ellipse, Arc, Arc, Arc, Pie, Pie, FillRectangle, FillTriangle, FillPolygon, FillCircle, FillEllipse, Fill, Fill, PixelSetAA, LineAA, PolylineAA, PolygonAA, TriangleAA, CircleAA, EllipseAA, LineWu, PolylineWu, PolygonWu, TriangleWu, CircleWu, EllipseWu, LineThickVertical, LineThickHorizontal, LineThick, PolylineThick, PolygonThick, PolylineSmooth, PolygonSmooth, FontSet, FontNameSet, FontSizeSet, FontFlagsSet, FontAngleSet, FontGet, FontNameGet, FontSizeGet, FontFlagsGet, FontAngleGet, TextOut, TextWidth, TextHeight, TextSize, GetDefaultColor, TransparentLevelSet, LoadFromFile, LineStyleGet, LineStyleSet

© 2000–2025, MetaQuotes Ltd.

ColorBackground（Getメソッド）

背景色を返します。

```
uint  ColorBackground()
```

戻り値

背景色

ColorBackground（Setメソッド）

背景色を設定します。

```
void  ColorBackground(
  const uint  value,  // 背景色
  )
```

パラメータ

value

[in] 背景色

© 2000–2025, MetaQuotes Ltd.

ColorBorder（Getメソッド）

境界色を返します。

```
uint  ColorBorder()
```

戻り値

境界色

ColorBorder（Setメソッド）

境界色を設定します。

```
void  ColorBorder(
  const uint  value,  // 境界色
  )
```

パラメータ

value

[in] 境界色

© 2000–2025, MetaQuotes Ltd.

ColorText（Getメソッド）

テキストの色を返します。

```
uint  ColorText()
```

戻り値

テキストの色

ColorText（Setメソッド）

テキストの色を設定します。

```
void  ColorText(
  const uint  value,  // テキストの色
  )
```

パラメータ

value

[in] テキストの色

© 2000-2025, MetaQuotes Ltd.

ColorGrid（Getメソッド）

グリッドの色を返します。

```
uint  ColorGrid()
```

戻り値

グリッドの色

ColorGrid（Setメソッド）

グリッドの色を設定します。

```
void  ColorGrid(
  const uint  value,  // グリッドの色
  )
```

パラメータ

value

[in] グリッドの色

© 2000-2025, MetaQuotes Ltd.

MaxData（Getメソッド）

許容されるデータ（系列）の最大量を返します。

```
uint  MaxData()
```

戻り値

データ（系列）の最大量

MaxData（Setメソッド）

許容されるデータ（系列）の最大量を設定します。

```
void  MaxData(
  const uint  value,  // データの量
  )
```

パラメータ

value

[in] データ（系列）の最大量

© 2000-2025, MetaQuotes Ltd.

MaxDescrLen（Getメソッド）

記述子の最大長を返します。

```
uint  MaxDescrLen()
```

戻り値

記述子の最大長

MaxDescrLen（Setメソッド）

記述子の最大長を設定します。

```
void  MaxDescrLen(
  const uint  value,  // 最大長
  )
```

パラメータ

value

[in] 記述子の最大長の値

© 2000–2025, MetaQuotes Ltd.

ShowFlags（Getメソッド）

チャート要素の可視性フラグを返します/設定します。

```
bool  ShowFlags()
```

戻り値

チャート要素の可視性フラグの値

ShowFlags（Setメソッド）

チャート要素の可視性フラグを設定します。

```
void  ShowFlags(
  const uint  flags,  // フラグ
  )
```

パラメータ

flags

[in] チャート要素の可視性フラグの値

© 2000–2025, MetaQuotes Ltd.

IsShowLegend

チャート上の凡例の可視性フラグを返します/設定します。

```
bool  IsShowLegend()
```

戻り値

凡例が可視であればtrue、そうでない場合はfalse

© 2000–2025, MetaQuotes Ltd.

IsShowScaleLeft

左側の値のスケールの可視性フラグを返します。

```
bool  IsShowScaleLeft()
```

戻り値

値のスケールが可視であればtrue、そうでない場合はfalse

© 2000-2025, MetaQuotes Ltd.

IsShowScaleRight

右側の値のスケールの可視性フラグを返します。

```
bool  IsShowScaleRight()
```

戻り値

値のスケールが可視であればtrue、そうでない場合はfalse

© 2000–2025, MetaQuotes Ltd.

IsShowScaleTop

上側の値のスケールの可視性フラグを返します。

```
bool  IsShowScaleTop()
```

戻り値

値のスケールが可視であればtrue、そうでない場合はfalse

© 2000–2025, MetaQuotes Ltd.

IsShowScaleBottom

下側の値のスケールの可視性フラグを返します。

```
bool  IsShowScaleBottom()
```

戻り値

値のスケールが可視であればtrue、そうでない場合はfalse

© 2000-2025, MetaQuotes Ltd.

IsShowGrid

チャート上のグリッドの可視性フラグを返します。

```
bool  IsShowGrid()
```

戻り値

グリッドが可視であればtrue、そうでない場合はfalse

© 2000-2025, MetaQuotes Ltd.

IsShowDescriptors

チャート上の記述子の可視性フラグを返します。

```
bool  IsShowDescriptors()
```

戻り値

記述子が可視であればtrue、そうでない場合はfalse

© 2000–2025, MetaQuotes Ltd.

IsShowPercent

チャート上のパーセンテージの可視性フラグを返します。

```
bool  IsShowPercent()
```

戻り値

パーセンテージが可視であればtrue、そうでない場合はfalse

© 2000-2025, MetaQuotes Ltd.

VScaleMin（Getメソッド）

値の垂直スケールの最小値を返します。

```
double  VScaleMin()
```

戻り値

値の垂直スケールの最小値

VScaleMin（Setメソッド）

値の垂直スケールの最小値を設定します。

```
void  VScaleMin(
  const double  value,    // 垂直スケールの値
  )
```

パラメータ

value

[in] 最小値

© 2000-2025, MetaQuotes Ltd.

VScaleMax

値の垂直スケールの最大値を返します。

```
double  VScaleMax()
```

戻り値

値の垂直スケールの最大値

VScaleMax

値の垂直スケールの最大値を設定します。

```
void  VScaleMax(
  const double  value,  //  値の垂直スケールの値
  )
```

パラメータ

value

[in] 最大値

© 2000-2025, MetaQuotes Ltd.

NumGrid

チャート グリッドをプロットするときの縦軸の分割数を返します。

```
uint  NumGrid()
```

戻り値

分割数

NumGrid

チャート グリッドをプロットするときの縦軸の分割数を設定します。

```
void  NumGrid(
  const uint  value,  // 分割数
  )
```

パラメータ

value

[in] 分割数

© 2000–2025, MetaQuotes Ltd.

DataOffset

データのオフセット値を返します。

```
int  DataOffset()
```

戻り値

データのオフセット

DataOffset

データのオフセット値を設定します。

```
void  DataOffset(
  const int  value,  // オフセット
  )
```

パラメータ

value

[in] データのオフセット

© 2000–2025, MetaQuotes Ltd.

DataTotal

チャート上のデータ系列の総数を返します。

```
uint  DataTotal()
```

戻り値

系列の数

© 2000–2025, MetaQuotes Ltd.

DrawDescriptors

記述子を描画するための仮想メソッド。

```
virtual void  DrawDescriptors()
```

© 2000–2025, MetaQuotes Ltd.

DrawData

指定されたインデックスでデータ系列を描画するための仮想メソッド。

```
virtual void  DrawData(
   const uint  idx=0,  // インデックス
   )
```

パラメータ

idx=0

[in] 系列のインデックス

© 2000–2025, MetaQuotes Ltd.

Create

グラフィカルリソースを作成する仮想メソッド。

```
virtual bool  Create(
  const string       name,      // リソース名
  const int          width,     // 幅
  const int          height,    // 高さ
  ENUM_COLOR_FORMAT  clrfmt,    // 形式
  )
```

パラメータ

name

[in] グラフィカルリソース名の基底。リソース名は、作成中に擬似乱数文字列を追加することによって生成されます。

width

[in] ピクセル単位での幅（X軸に沿ったサイズ）

height

[in] ピクセル単位での高さ（Y軸に沿ったサイズ）

clrfmt

[in] 色処理メソッド。色処理メソッドの詳細については、ResourceCreate()関数の説明をご参照ください。

戻り値

成功した場合はtrue、それ以外の場合はfalse

© 2000-2025, MetaQuotes Ltd.

AllowedShowFlags

チャート要素に許可された可視性フラグのセットを設定します。

```
void  AllowedShowFlags(
  const uint  flags,  // フラグ
  )
```

パラメータ

flags

[in] 許可されたフラグ

© 2000–2025, MetaQuotes Ltd.

ShowLegend

凡例の可視性フラグ(FLAG_SHOW_LEGEND)を設定します。

```
void  ShowLegend(
  const bool  flag,  // フラグ値
  )
```

パラメータ

flag

[in]フラグ値：

- true — 凡例が可視
- false — 凡例が可視でない

© 2000–2025, MetaQuotes Ltd.

ShowScaleLeft

左のスケールの可視性フラグ(FLAG_SHOW_SCALE_LEFT)を設定します。

```
void  ShowScaleLeft(
  const bool  flag,  // フラグ値
  )
```

パラメータ

flag

[in]フラグ値：

- true — 左のスケールが可視
- false — 左のスケールが可視でない

© 2000–2025, MetaQuotes Ltd.

ShowScaleRight

右のスケールの可視性フラグ(FLAG_SHOW_SCALE_RIGHT)を設定します。

```
void  ShowScaleRight(
  const bool  flag,  // フラグ値
  )
```

パラメータ

flag

[in]フラグ値：

- true — 右のスケールが可視
- false — 右のスケールが可視でない

© 2000–2025, MetaQuotes Ltd.

ShowScaleTop

上のスケールの可視性フラグ(FLAG_SHOW_SCALE_TOP)を設定します。

```
void  ShowScaleTop(
  const bool  flag,  // フラグ値
  )
```

パラメータ

flag

[in]フラグ値：

- true — 上のスケールが可視
- false — 上のスケールが可視でない

© 2000–2025, MetaQuotes Ltd.

ShowScaleBottom

下のスケールの可視性フラグ（FLAG_SHOW_SCALE_BOTTOM）を設定します。

```
void  ShowScaleBottom(
  const bool  flag,     // フラグ値
  )
```

パラメータ

flag

[in]フラグ値：

- true — 下のスケールが可視
- false — 下のスケールが可視でない

© 2000–2025, MetaQuotes Ltd.

ShowGrid

グリッドの可視性フラグ(FLAG_SHOW_GRID)を設定します。

```
void  ShowGrid(
  const bool  flag,  // フラグ値
  )
```

パラメータ

flag

[in]フラグ値：

- true — グリッドが可視
- false — グリッドが可視でない

© 2000–2025, MetaQuotes Ltd.

ShowDescriptors

記述子の可視性フラグ(FLAG_SHOW_DESCRIPTORS)を設定します。

```
void  ShowDescriptors(
  const bool  flag,    // フラグ値
  )
```

パラメータ

flag

[in]フラグ値：

- true — 記述子が可視
- false — 記述子が可視でない

© 2000–2025, MetaQuotes Ltd.

ShowValue

値の可視性フラグ(FLAG_SHOW_VALUE)を設定します。

```
void  ShowValue(
  const bool  flag,  // フラグ値
  )
```

パラメータ

flag

[in]フラグ値：

- true — 値が可視
- false — 値が可視でない

© 2000–2025, MetaQuotes Ltd.

ShowPercent

パーセンテージの可視性フラグ（FLAG_SHOW_PERCENT）を設定します。

```
void  ShowPercent(
  const bool  flag,  // フラグ値
  )
```

パラメータ

flag

[in]フラグ値：

- true — パーセンテージが可視
- false — パーセンテージが可視でない

© 2000-2025, MetaQuotes Ltd.

LegendAlignment

凡例のテキストの配置を設定します。.

```
void  LegendAlignment(
  const ENUM_ALIGNMENT  value,  // フラグ
  )
```

パラメータ

value

[in] ENUM_ALIGNMENT列挙の値の1つをとります。

- ALIGNMENT_LEFT — 左寄せ
- ALIGNMENT_TOP — 上寄せ
- ALIGNMENT_RIGHT — 右寄せ
- ALIGNMENT_BOTTOM — 下寄せ

© 2000–2025, MetaQuotes Ltd.

Accumulative

シリーズの値の累積フラグを設定します。

```
void  Accumulative(
  const bool  flag=true,  // フラグ値
  )
```

パラメータ

flag=true

[in] フラグ値：

- true — 系列の現在値は前のすべて値の合計で置き換えられます。
- false — 系列を描画するための標準モードです。

© 2000–2025, MetaQuotes Ltd.

VScaleParams

値の垂直スケールのパラメータを設定します。

```
void  VScaleParams(
  const double  max,   // 最大値
  const double  min,   // 最小値
  const uint    grid,  // 分割数
  )
```

パラメータ

max

[in] 最小値

min

[in] 最大値

grid

[in] スケール分割数

© 2000–2025, MetaQuotes Ltd.

DescriptorUpdate

指定された位置の系列記述子の値を更新します。

```
bool  DescriptorUpdate(
  const uint    pos,    // インデックス
  const string  descr,  // 値
  )
```

パラメータ

pos

[in] 系列のインデックス－0から始まったその追加の系列番号

descr

[in] 記述子の名前

戻り値

成功した場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

ColorUpdate

指定された位置の系列の色を更新します。

```
bool  ColorUpdate(
  const uint  pos,  // インデックス
  const uint  clr,  // 色
  )
```

パラメータ

pos

[in] 系列のインデックス－0から始まったその追加の系列番号

clr

[in] 色の値

戻り値

成功した場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

ValuesCheck

チャートをプロットするための内部計算を実行する補助的仮想メソッド。

```
virtual void  ValuesCheck()
```

© 2000–2025, MetaQuotes Ltd.

Redraw

チャートを再描画するための仮想メソッド。

```
virtual void  Redraw()
```

© 2000–2025, MetaQuotes Ltd.

DrawBackground

背景を再描画するための仮想メソッド。

```
virtual void  DrawBackground()
```

© 2000–2025, MetaQuotes Ltd.

DrawLegend

凡例を再描画するための仮想メソッド。

```
virtual void  DrawLegend()
```

© 2000–2025, MetaQuotes Ltd.

DrawLegendVertical

垂直の凡例を描画します。

```
int  DrawLegendVertical(
  const int  w,  // 幅
  const int  h,  // 高さ
  )
```

パラメータ

w

[in] 凡例のテキストの最大幅

h

[in] 凡例のテキストの最大の高さ

戻り値

ピクセル単位での凡例の幅

© 2000–2025, MetaQuotes Ltd.

DrawLegendHorizontal

水平の凡例を描画します。

```
int  DrawLegendHorizontal(
  const int  w,  //
  const int  h,  //
  )
```

パラメータ

w

[in] 凡例のテキストの最大幅

h

[in] 凡例のテキストの最大の高さ

戻り値

ピクセル単位での凡例の高さ

© 2000–2025, MetaQuotes Ltd.

CalcScales

値のスケールのラベルの座標を計算する仮想メソッド。

```
virtual void  CalcScales()
```

© 2000–2025, MetaQuotes Ltd.

DrawScales

値のすべてのスケールを再描画する仮想メソッド。

```
virtual void  DrawScales()
```

© 2000–2025, MetaQuotes Ltd.

DrawScaleLeft

値の左のスケールを再描画する仮想メソッド。

```
virtual int  DrawScaleLeft(
  const bool  draw,  // フラグ
  )
```

パラメータ

draw

[in] スケールを再描画する必要があるかどうかを示すフラグ

戻り値

値のスケールの幅

© 2000–2025, MetaQuotes Ltd.

DrawScaleRight

値の右のスケールを再描画する仮想メソッド。

```
virtual int  DrawScaleRight(
  const bool  draw,  // フラグ
  )
```

パラメータ

draw

[in] スケールを再描画する必要があるかどうかを示すフラグ

戻り値

値のスケールの幅

© 2000–2025, MetaQuotes Ltd.

DrawScaleTop

値の上のスケールを再描画する仮想メソッド。

```
virtual int  DrawScaleTop(
  const bool  draw,  // フラグ
  )
```

パラメータ

draw

[in] スケールを再描画する必要があるかどうかを示すフラグ

戻り値

値のスケールの高さ

© 2000–2025, MetaQuotes Ltd.

DrawScaleBottom

値の下のスケールを再描画する仮想メソッド。

```
virtual int  DrawScaleBottom(
  const bool  draw,  // フラグ
  )
```

パラメータ

draw

[in] スケールを再描画する必要があるかどうかを示すフラグ

戻り値

値のスケールの高さ

© 2000–2025, MetaQuotes Ltd.

DrawGrid

グリッドを再描画するための仮想メソッド。

```
virtual void  DrawGrid()
```

© 2000–2025, MetaQuotes Ltd.

DrawChart

チャートを再描画するための仮想メソッド。

```
virtual void  DrawChart()
```

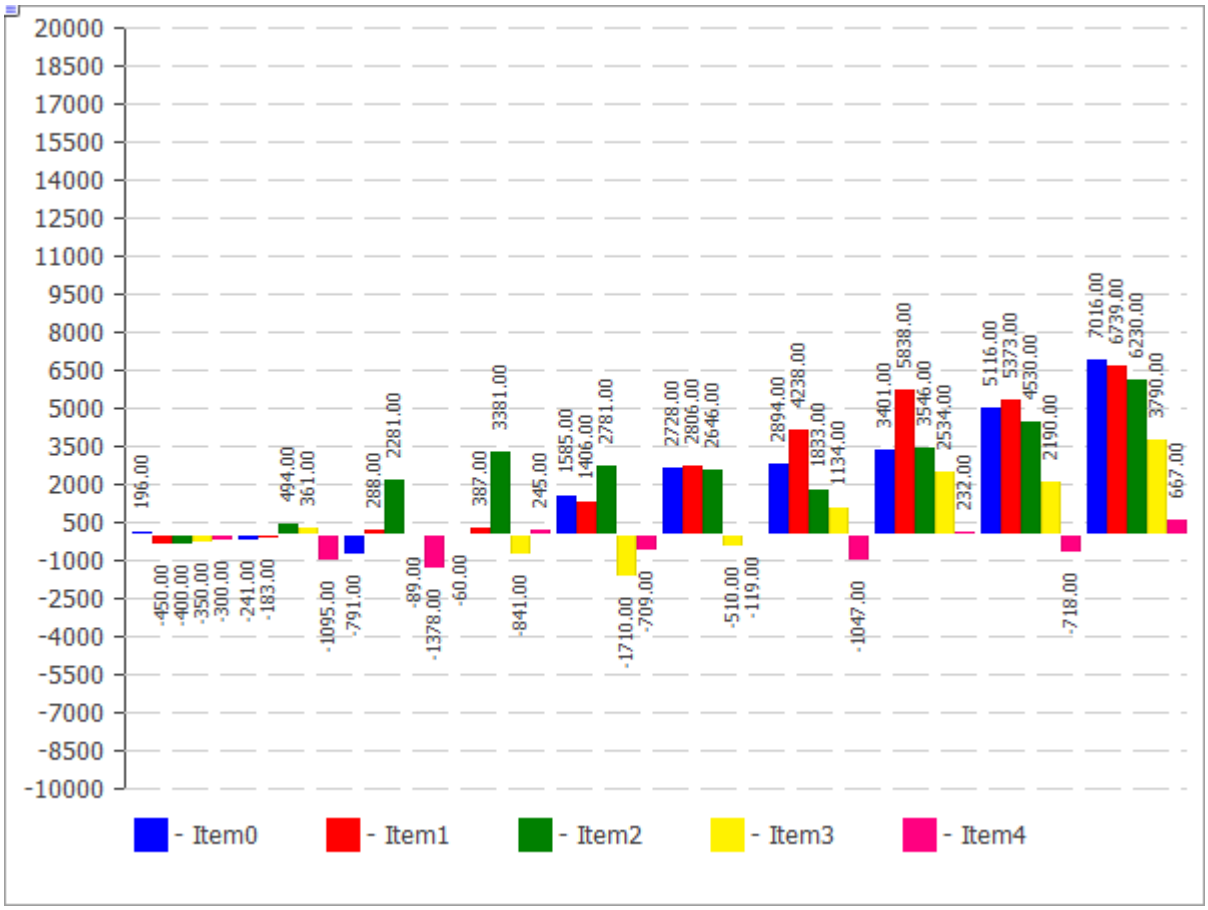
© 2000–2025, MetaQuotes Ltd.

CHistogramChart

ヒストグラムをプロットするためのクラス

説明

ヒストグラムのプロットを操作するためのすべてのメソッドは、このクラスで実装されています。列幅を設定したり、データ系列を使用した作業を設定したりすることができます。ヒストグラムの列をグラデーションで塗りつぶすためのメソッドが含まれているため、データをより明確に視覚化できます。



上の図のコードは下にあります。

宣言

```
class CHistogramChart : public CChartCanvas
```

タイトル

```
#include <Canvas\Charts\HistogramChart.mqh>
```

継承階層

CCanvas
  CChartCanvas
    CHistogramChart

© 2000–2025, MetaQuotes Ltd.

クラスメソッド

メソッド	アクション
Gradient	ヒストグラム列にグラデーション塗りを適用するかどうかを示すフラグを設定します。
BarGap	ヒストグラムの原点からのオフセットの値を設定します。
BarMinSize	ヒストグラムの列の最小幅を設定します。
BarBorder	各列の境界線を描画する必要があるかどうかを示すフラグを設定します。
Create	グラフィカルリソースを作成する仮想メソッド。
SeriesAdd	新しいデータ系列を追加します。
SeriesInsert	チャートにデータ系列を挿入します。
SeriesUpdate	チャートのデータ系列を更新します。
SeriesDelete	チャートからデータ系列を削除します。
ValueUpdate	指定された系列の要素値を更新します。
DrawData	指定された系列でヒストグラムを描画するための仮想メソッド。
DrawBar	ヒストグラムの列を塗りつぶされた矩形として描画します。
GradientBrush	グラデーション塗りのブラシを作成します。

クラスから継承されたメソッド CCanvas

CreateBitmap, CreateBitmap, CreateBitmapLabel, CreateBitmapLabel, Attach, Attach, Destroy, ChartObjectName, ResourceName, Width, Height, Update, Resize, Erase, PixelGet, PixelSet, LineVertical, LineHorizontal, Line, Polyline, Polygon, Rectangle, Triangle, Circle, Ellipse, Arc, Arc, Arc, Pie, Pie, FillRectangle, FillTriangle, FillPolygon, FillCircle, FillEllipse, Fill, Fill, PixelSetAA, LineAA, PolylineAA, PolygonAA, TriangleAA, CircleAA, EllipseAA, LineWu, PolylineWu, PolygonWu, TriangleWu, CircleWu, EllipseWu, LineThickVertical, LineThickHorizontal, LineThick, PolylineThick, PolygonThick, PolylineSmooth, PolygonSmooth, FontSet, FontNameSet, FontSizeSet, FontFlagsSet, FontAngleSet, FontGet, FontNameGet, FontSizeGet, FontFlagsGet, FontAngleGet, TextOut, TextWidth, TextHeight, TextSize, GetDefaultColor, TransparentLevelSet, LoadFromFile, LineStyleGet, LineStyleSet

クラスから継承されたメソッド CChartCanvas

ColorBackground, ColorBackground, ColorBorder, ColorBorder, ColorText, ColorText, ColorGrid, ColorGrid, MaxData, MaxData, MaxDescrLen, MaxDescrLen, AllowedShowFlags, ShowFlags, ShowFlags, IsShowLegend, IsShowScaleLeft, IsShowScaleRight, IsShowScaleTop, IsShowScaleBottom, IsShowGrid, IsShowDescriptors, IsShowPercent, ShowLegend, ShowScaleLeft,

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CCanvas

CreateBitmap, CreateBitmap, CreateBitmapLabel, CreateBitmapLabel, Attach, Attach, Destroy, ChartObjectName, ResourceName, Width, Height, Update, Resize, Erase, PixelGet, PixelSet, LineVertical, LineHorizontal, Line, Polyline, Polygon, Rectangle, Triangle, Circle, Ellipse, Arc, Arc, Arc, Pie, Pie, FillRectangle, FillTriangle, FillPolygon, FillCircle, FillEllipse, Fill, Fill, PixelSetAA, LineAA, PolylineAA, PolygonAA, TriangleAA, CircleAA, EllipseAA, LineWu, PolylineWu, PolygonWu, TriangleWu, CircleWu, EllipseWu, LineThickVertical, LineThickHorizontal, LineThick, PolylineThick, PolygonThick, PolylineSmooth, PolygonSmooth, FontSet, FontNameSet, FontSizeSet, FontFlagsSet, FontAngleSet, FontGet, FontNameGet, FontSizeGet, FontFlagsGet, FontAngleGet, TextOut, TextWidth, TextHeight, TextSize, GetDefaultColor, TransparentLevelSet, LoadFromFile, LineStyleGet, LineStyleSet

ShowScaleRight, ShowScaleTop, ShowScaleBottom, ShowGrid, ShowDescriptors, ShowValue, ShowPercent, LegendAlignment, Accumulative, VScaleMin, VScaleMin, VScaleMax, VScaleMax, NumGrid, NumGrid, VScaleParams, DataOffset, DataOffset, DataTotal, DescriptorUpdate, ColorUpdate

例

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                          HistogramChartSample.mq5 |
//|                  Copyright 2009-2017, MetaQuotes Software Corp. |
//|                                              http://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "2009-2017, MetaQuotes Software Corp."
#property link        "http://www.mql5.com"
#property description "Example of using histogram"
//---
#include <Canvas\Charts\HistogramChart.mqh>
//+------------------------------------------------------------------+
//| inputs                                                           |
//+------------------------------------------------------------------+
input bool Accumulative=true;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
int OnStart(void)
  {
   int k=100;
   double arr[10];
//--- チャートを作成する
   CHistogramChart chart;
   if(!chart.CreateBitmapLabel("SampleHistogramChart",10,10,600,450))
     {
      Print("Error creating histogram chart: ",GetLastError());
      return(-1);
     }
   if(Accumulative)
     {
      chart.Accumulative();
      chart.VScaleParams(20*k*10,-10*k*10,20);
     }
   else
      chart.VScaleParams(20*k,-10*k,20);
   chart.ShowValue(true);
   chart.ShowScaleTop(false);
   chart.ShowScaleBottom(false);
   chart.ShowScaleRight(false);
   chart.ShowLegend();
   for(int j=0;j<5;j++)
     {
      for(int i=0;i<10;i++)
        {
         k=-k;
         if(k>0)
            arr[i]=k*(i+10-j);
         else
            arr[i]=k*(i+10-j)/2;
        }
      chart.SeriesAdd(arr,"Item"+IntegerToString(j));
     }
//--- play with values
   while(!IsStopped())
     {
      int i=rand()%5;
      int j=rand()%10;
      k=rand()%3000-1000;
      chart.ValueUpdate(i,j,k);
      Sleep(200);
     }
```

© 2000–2025, MetaQuotes Ltd.

```
//--- finish
   chart.Destroy();
   return(0);
  }
```

© 2000–2025, MetaQuotes Ltd.

Gradient

ヒストグラム列にグラデーション塗りを適用するかどうかを示すフラグを設定します。

```
void  Gradient(
  const bool  flag=true,  // フラグ値
  )
```

パラメータ

flag=true

フラグの値：グラデーションの塗りが有効な場合はtrue、そうでない場合はfalse

© 2000-2025, MetaQuotes Ltd.

BarGap

ヒスト グラムの原 点 からのオフセット の値 を設 定 します。

```
void  BarGap(
  const uint  value,  // オフセット
  )
```

パラメータ

value

[in] ヒスト グラムのオフセット 値

© 2000–2025, MetaQuotes Ltd.

BarMinSize

ヒストグラムの列の最小幅を設定します。

```
void  BarMinSize(
  const uint  value,  // 最小幅
  )
```

パラメータ

value

[in] 最小幅

© 2000–2025, MetaQuotes Ltd.

BarBorder

各列の境界線を描画する必要があるかどうかを示すフラグを設定します。

```
void  BarBorder(
  const uint  value,  // フラグ
  )
```

パラメータ

value

[in]フラグ値：

- true — 境界が描画されます。
- false — 境界が描画されません。

© 2000–2025, MetaQuotes Ltd.

Create

グラフィカルリソースを作成する仮想メソッド。

```
virtual bool  Create(
  const string       name,    // 名前
  const int          width,   // 幅
  const int          height,  // 高さ
  ENUM_COLOR_FORMAT  clrfmt,  // 形式
  )
```

パラメータ

name

[in] グラフィカルリソース名の基底。リソース名は、作成中に擬似乱数文字列を追加することによって生成されます。

width

[in] ピクセル単位での幅（X軸に沿ったサイズ）

height

[in] ピクセル単位での高さ（Y軸に沿ったサイズ）

clrfmt

[in] 色処理メソッド。色処理メソッドの詳細については、ResourceCreate()関数の説明をご参照ください。

戻り値

成功した場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

SeriesAdd

新しいデータ系列を追加します。

```
bool  SeriesAdd(
  const double&  value[],  // 値
  const string   descr,    // ラベル
  const uint     clr,      // 色
  )
```

パラメータ

value[]

[in] データ系列

descr

[in] 系列のラベル

clr

[in] 系列の表示色

戻り値

成功した場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

SeriesInsert

チャートにデータ系列を挿入します。

```
bool  SeriesInsert(
  const uint    pos,        // インデックス
  const double&  value[],   // 値
  const string  descr,      // ラベル
  const uint    clr,        // 色
  )
```

パラメータ

pos

[in] 挿入に使われるインデックス

value[]

[in] データ系列

descr

[in] 系列のラベル

clr

[in] 系列の表示色

戻り値

成功した場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

SeriesUpdate

チャートのデータ系列を更新します。

```
bool  SeriesUpdate(
  const uint    pos,        // インデックス
  const double  &value[],   // 値
  const string  descr,      // ラベル
  const uint    clr,        // 色
  )
```

パラメータ

pos

[in] 系列のインデックス－0から始まったその追加の系列番号

&value[]

[in] データ系列の新しい値

descr

[in] 系列のラベル

clr

[in] 系列の表示色

戻り値

成功した場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

SeriesDelete

チャートからデータ系列を削除します。

```
bool  SeriesDelete(
  const uint  pos,  // インデックス
  )
```

パラメータ

pos

[in] 系列のインデックス－0から始まったその追加の系列番号

戻り値

成功した場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

ValueUpdate

指定された系列の指定された値を更新します。

```
bool  ValueUpdate(
  const uint   series,  // 系列のインデックス
  const uint   pos,     // 要素のインデックス
  double       value,   // 値
  )
```

パラメータ

series

[in] 系列のインデックス－0から始まったその追加の系列番号

pos

[in] 系列での要素のインデックス

value

[in] 新しい値

戻り値

成功した場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

DrawData

指定された系列でヒストグラムを描画するための仮想メソッド。

```
virtual void  DrawData(
  const uint  index,  // インデックス
  )
```

パラメータ

index

[in] 系列のインデックス－0から始まったその追加の系列番号

© 2000–2025, MetaQuotes Ltd.

DrawBar

ヒストグラムの列を塗りつぶされた矩形として描画します。

```
void  DrawBar(
  const int   x,    // X座標
  const int   y,    // Y座標
  const int   w,    // 幅
  const int   h,    // 高さ
  const uint  clr,  // 色
  )
```

パラメータ

x

[in] 長方形の左上の点のX座標

y

[in] 長方形の左上の点のY座標

w

[in] 長方形の幅

h

[in] 長方形の高さ

clr

[in] 長方形の色

© 2000–2025, MetaQuotes Ltd.

GradientBrush

グラデーション塗りのブラシを作成します。

```
void  GradientBrush(
  const int   size,      // サイズ
  const uint  fill_clr,  // 塗る色
  )
```

パラメータ

size

[in] ブラシの幅

fill_clr

[in] 塗る色

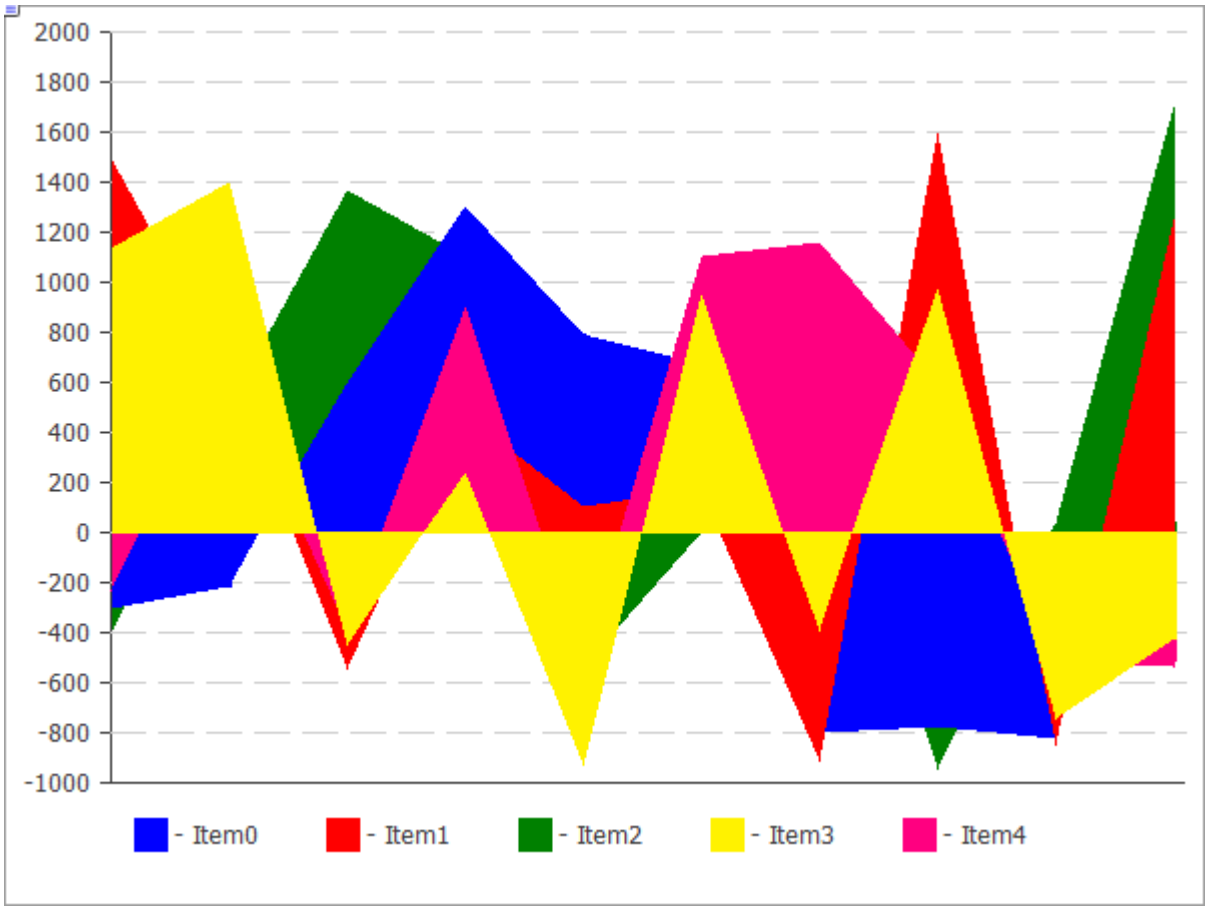
© 2000–2025, MetaQuotes Ltd.

CLineChart

曲線をプロットするためのクラス

説明

このクラスに含まれるメソッドは、チャートの曲線を操作するために設計されています。それは、プロットされた曲線によって制限された領域を満たす能力を特徴とします。



上の図のコードは下にあります。

宣言

```
class CLineChart : public CChartCanvas
```

タイトル

```
#include <Canvas\Charts\LineChart.mqh>
```

継承階層

CCanvas
  CChartCanvas
    CLineChart

クラスメソッド

© 2000–2025, MetaQuotes Ltd.

メソッド	アクション
Filled	データ系列で定義された曲線の下の領域を塗りつぶすためのフラグを設定します。
Create	グラフィカルリソースを作成します。
SeriesAdd	新しいデータ系列を追加します。
SeriesInsert	チャートにデータ系列を挿入します。
SeriesUpdate	チャートのデータ系列を更新します。
SeriesDelete	チャートからデータ系列を削除します。
ValueUpdate	指定された系列の指定された値を更新します。
DrawChart	曲線とそのすべての要素を描画するための仮想メソッド。
DrawData	指定された系列で曲線を描画するための仮想メソッド。
CalcArea	データ系列で定義された曲線の下の面積を計算します。

クラスから継承されたメソッド CCanvas

CreateBitmap, CreateBitmap, CreateBitmapLabel, CreateBitmapLabel, Attach, Attach, Destroy, ChartObjectName, ResourceName, Width, Height, Update, Resize, Erase, PixelGet, PixelSet, LineVertical, LineHorizontal, Line, Polyline, Polygon, Rectangle, Triangle, Circle, Ellipse, Arc, Arc, Arc, Pie, Pie, FillRectangle, FillTriangle, FillPolygon, FillCircle, FillEllipse, Fill, Fill, PixelSetAA, LineAA, PolylineAA, PolygonAA, TriangleAA, CircleAA, EllipseAA, LineWu, PolylineWu, PolygonWu, TriangleWu, CircleWu, EllipseWu, LineThickVertical, LineThickHorizontal, LineThick, PolylineThick, PolygonThick, PolylineSmooth, PolygonSmooth, FontSet, FontNameSet, FontSizeSet, FontFlagsSet, FontAngleSet, FontGet, FontNameGet, FontSizeGet, FontFlagsGet, FontAngleGet, TextOut, TextWidth, TextHeight, TextSize, GetDefaultColor, TransparentLevelSet, LoadFromFile, LineStyleGet, LineStyleSet

クラスから継承されたメソッド CChartCanvas

ColorBackground, ColorBackground, ColorBorder, ColorBorder, ColorText, ColorText, ColorGrid, ColorGrid, MaxData, MaxData, MaxDescrLen, MaxDescrLen, AllowedShowFlags, ShowFlags, ShowFlags, IsShowLegend, IsShowScaleLeft, IsShowScaleRight, IsShowScaleTop, IsShowScaleBottom, IsShowGrid, IsShowDescriptors, IsShowPercent, ShowLegend, ShowScaleLeft, ShowScaleRight, ShowScaleTop, ShowScaleBottom, ShowGrid, ShowDescriptors, ShowValue, ShowPercent, LegendAlignment, Accumulative, VScaleMin, VScaleMin, VScaleMax, VScaleMax, NumGrid, NumGrid, VScaleParams, DataOffset, DataOffset, DataTotal, DescriptorUpdate, ColorUpdate

例

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                LineChartSample.mq5 |
//|                  Copyright 2009-2017, MetaQuotes Software Corp. |
//|                                              http://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "2009-2017, MetaQuotes Software Corp."
#property link        "http://www.mql5.com"
#property description "Example of using line chart"
//---
#include <Canvas\Charts\LineChart.mqh>
//+------------------------------------------------------------------+
//| inputs                                                           |
//+------------------------------------------------------------------+
input bool Accumulative=false;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
int OnStart(void)
  {
   int k=100;
   double arr[10];
//--- チャートを作成する
   CLineChart chart;
//--- create chart
   if(!chart.CreateBitmapLabel("SampleHistogrammChart",10,10,600,450))
     {
      Print("Error creating line chart: ",GetLastError());
      return(-1);
     }
   if(Accumulative)
     {
      chart.Accumulative();
      chart.VScaleParams(20*k*10,-10*k*10,20);
     }
   else
      chart.VScaleParams(20*k,-10*k,15);
   chart.ShowScaleTop(false);
   chart.ShowScaleRight(false);
   chart.ShowLegend();
   chart.Filled();
   for(int j=0;j<5;j++)
     {
      for(int i=0;i<10;i++)
        {
         k=-k;
         if(k>0)
            arr[i]=k*(i+10-j);
         else
            arr[i]=k*(i+10-j)/2;
        }
      chart.SeriesAdd(arr,"Item"+IntegerToString(j));
     }
//--- play with values
   while(!IsStopped())
     {
      int i=rand()%5;
      int j=rand()%10;
      k=rand()%3000-1000;
      chart.ValueUpdate(i,j,k);
      Sleep(200);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- finish
   chart.Destroy();
   return(0);
  }
```

© 2000–2025, MetaQuotes Ltd.

Filled

データ系列で定義された曲線の下の領域を塗りつぶすことが必要かどうかを示すフラグを設定します。

```
void  Filled(
  const bool  flag=true,  // フラグ
  )
```

パラメータ

flag=true

[in] フラグ値：

- true — 曲線の下の領域を塗りつぶします
- false — 曲線の下の領域を塗りつぶしません

© 2000–2025, MetaQuotes Ltd.

Create

グラフィカルリソースを作成する仮想メソッド。

```
virtual bool  Create(
  const string       name,    // 名前
  const int          width,   // 幅
  const int          height,  // 高さ
  ENUM_COLOR_FORMAT  clrfmt,  // 形式
  )
```

パラメータ

name

[in] グラフィカルリソース名の基底。リソース名は、作成中に擬似乱数文字列を追加することによって生成されます。

width

[in] ピクセル単位での幅（X軸に沿ったサイズ）

height

[in] ピクセル単位での高さ（Y軸に沿ったサイズ）

clrfmt

[in] 色処理メソッド。色処理メソッドの詳細については、ResourceCreate()関数の説明をご参照ください。

戻り値

成功した場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

SeriesAdd

新しいデータ系列を追加します。

```
bool  SeriesAdd(
  const double&  value[],  // 値
  const string   descr,    // ラベル
  const uint     clr,      // 色
  )
```

パラメータ

value[]

[in] データ系列

descr

[in] 系列のラベル

clr

[in] 系列の表示色

戻り値

成功した場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

SeriesInsert

チャートにデータ系列を挿入します。

```
bool  SeriesInsert(
  const uint     pos,        // インデックス
  const double&  value[],    // 値
  const string   descr,      // ラベル
  const uint     clr,        // 色
  )
```

パラメータ

pos

[in] 挿入に使われるインデックス

value[]

[in] データ系列

descr

[in] 系列のラベル

clr

[in] 系列の表示色

戻り値

成功した場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

SeriesUpdate

チャートのデータ系列を更新します。

```
bool  SeriesUpdate(
  const uint     pos,        // インデックス
  const double&  value[],    // 値
  const string   descr,      // ラベル
  const uint     clr,        // 色
  )
```

パラメータ

pos

[in] 系列のインデックス－0から始まったその追加の系列番号

value[]

[in] データ系列の新しい値

descr

[in] 系列のラベル

clr

[in] 系列の表示色

戻り値

成功した場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

SeriesDelete

チャートからデータ系列を削除します。

```
bool  SeriesDelete(
  const uint  pos,  // インデックス
  )
```

パラメータ

pos

[in] 系列のインデックス－0から始まったその追加の系列番号

戻り値

成功した場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

ValueUpdate

指定された系列の指定された値を更新します。

```
bool  ValueUpdate(
  const uint   series,  // 系列のインデックス
  const uint   pos,     // 要素のインデックス
  double       value,   // 値
  )
```

パラメータ

series

[in] 系列のインデックス－0から始まったその追加の系列番号

pos

[in] 系列での要素のインデックス

value

[in] 新しい値

戻り値

成功した場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

DrawChart

曲線とそのすべての要素を描画するための仮想メソッド。

```
virtual void  DrawChart()
```

© 2000–2025, MetaQuotes Ltd.

DrawData

指定された系列で曲線を描画するための仮想メソッド。

```
virtual void  DrawData(
  const uint  index,  // インデックス
  )
```

パラメータ

index

[in] 系列のインデックス－0から始まったその追加の系列番号

© 2000–2025, MetaQuotes Ltd.

CalcArea

データ系列で定義された曲線の下の面積を計算します。

```
double  CalcArea(
  const uint  index,  // インデックス
  )
```

パラメータ

index

[in] 系列のインデックス－0から始まったその追加の系列番号

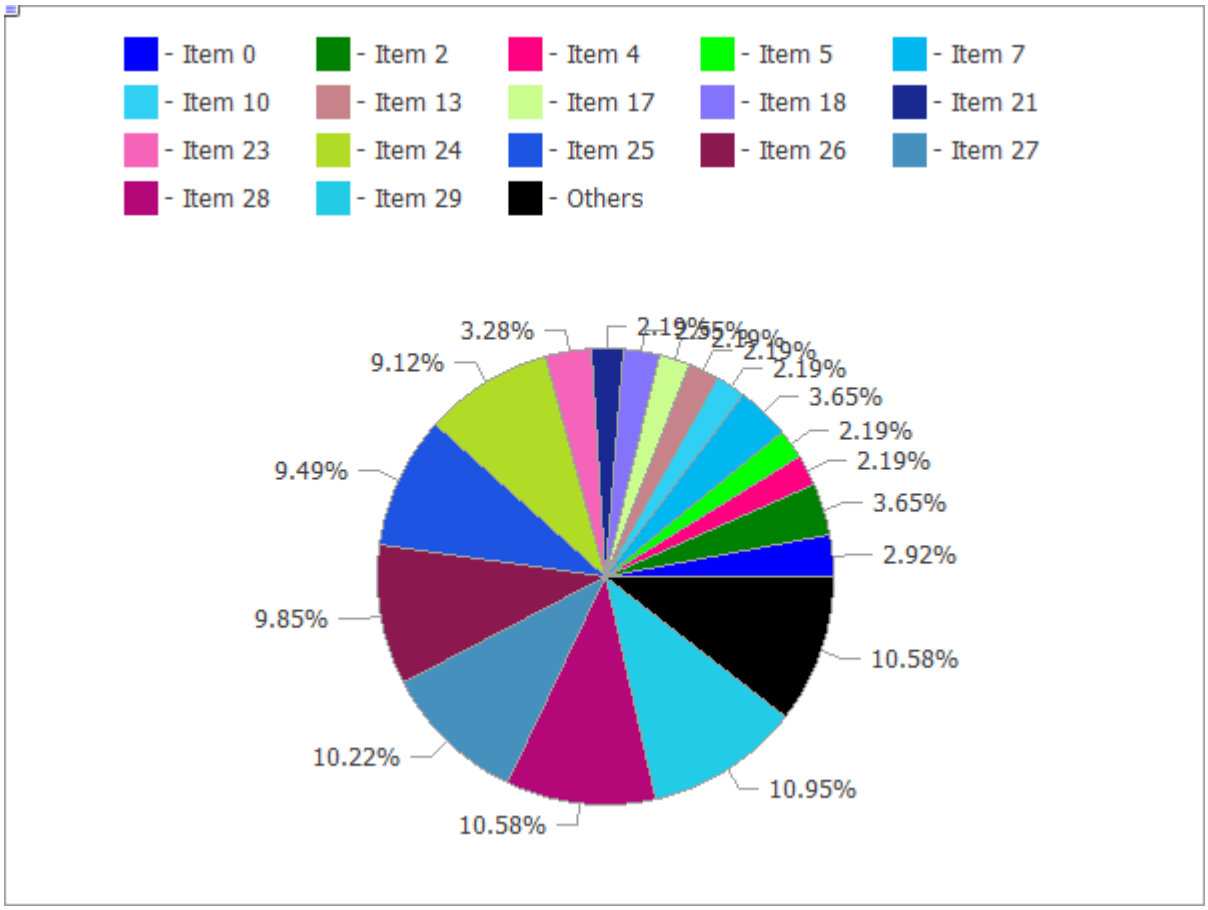
戻り値

データ系列によって定義される曲線によって制限された図形の面積

© 2000–2025, MetaQuotes Ltd.

CPieChart

円グラフをプロットするためのクラス



上の図のコードは下にあります。

説明

このクラスに含まれるメソッドは、グラフィックリソースの作成からラベルやセグメントのデザインまで、本格的な円グラフの操作のために設計されています。

宣言

```
class CPieChart : public CChartCanvas
```

タイトル

```
#include <Canvas\Charts\PieChart.mqh>
```

継承階層

CCanvas
  CChartCanvas
    CPieChart

クラスメソッド

© 2000–2025, MetaQuotes Ltd.

メソッド	アクション
Create	グラフィカルリソースを作成する仮想メソッド。
SeriesSet	円グラフに表示される系列の値を設定します。
ValueAdd	円グラフの最後に新しい値を追加します。
ValueInsert	円グラフの指定された位置に新しい値を挿入します。
ValueUpdate	円グラフの指定された位置での値を更新します。
ValueDelete	円グラフの指定された位置での値を削除します。
DrawChart	円グラフとそのすべての要素を描画するための仮想メソッド。
DrawPie	指定された値に対応する円グラフのセグメントを描画します。
LabelMake	セグメントラベルをその値と元のラベルに基づいて生成します。

クラスから継承されたメソッド CCanvas

CreateBitmap, CreateBitmap, CreateBitmapLabel, CreateBitmapLabel, Attach, Attach, Destroy, ChartObjectName, ResourceName, Width, Height, Update, Resize, Erase, PixelGet, PixelSet, LineVertical, LineHorizontal, Line, Polyline, Polygon, Rectangle, Triangle, Circle, Ellipse, Arc, Arc, Arc, Pie, Pie, FillRectangle, FillTriangle, FillPolygon, FillCircle, FillEllipse, Fill, Fill, PixelSetAA, LineAA, PolylineAA, PolygonAA, TriangleAA, CircleAA, EllipseAA, LineWu, PolylineWu, PolygonWu, TriangleWu, CircleWu, EllipseWu, LineThickVertical, LineThickHorizontal, LineThick, PolylineThick, PolygonThick, PolylineSmooth, PolygonSmooth, FontSet, FontNameSet, FontSizeSet, FontFlagsSet, FontAngleSet, FontGet, FontNameGet, FontSizeGet, FontFlagsGet, FontAngleGet, TextOut, TextWidth, TextHeight, TextSize, GetDefaultColor, TransparentLevelSet, LoadFromFile, LineStyleGet, LineStyleSet

クラスから継承されたメソッド CChartCanvas

ColorBackground, ColorBackground, ColorBorder, ColorBorder, ColorText, ColorText, ColorGrid, ColorGrid, MaxData, MaxData, MaxDescrLen, MaxDescrLen, AllowedShowFlags, ShowFlags, ShowFlags, IsShowLegend, IsShowScaleLeft, IsShowScaleRight, IsShowScaleTop, IsShowScaleBottom, IsShowGrid, IsShowDescriptors, IsShowPercent, ShowLegend, ShowScaleLeft, ShowScaleRight, ShowScaleTop, ShowScaleBottom, ShowGrid, ShowDescriptors, ShowValue, ShowPercent, LegendAlignment, Accumulative, VScaleMin, VScaleMin, VScaleMax, VScaleMax, NumGrid, NumGrid, VScaleParams, DataOffset, DataOffset, DataTotal, DescriptorUpdate, ColorUpdate

例

```
//+------------------------------------------------------------------+
//|                                                PieChartSample.mq5 |
```

© 2000–2025, MetaQuotes Ltd.

```
//|                  Copyright 2009-2017, MetaQuotes Software Corp. |
//|                                              http://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "2009-2017, MetaQuotes Software Corp."
#property link        "http://www.mql5.com"
#property description "Example of using pie chart"
//---
#include <Canvas\Charts\PieChart.mqh>
//+------------------------------------------------------------------+
//| 入力                                                             |
//+------------------------------------------------------------------+
input int      Width=600;
input int      Height=450;
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
int OnStart(void)
  {
//--- 確認
   if(Width<=0 || Height<=0)
     {
      Print("Too simple.");
      return(-1);
     }
//--- チャートを作成する
   CPieChart pie_chart;
   if(!pie_chart.CreateBitmapLabel("PieChart",10,10,Width,Height))
     {
      Print("Error creating pie chart: ",GetLastError());
      return(-1);
     }
   pie_chart.ShowPercent();
//--- 描画する
   for(uint i=0;i<30;i++)
     {
      pie_chart.ValueAdd(100*(i+1),"Item "+IntegerToString(i));
      Sleep(10);
     }
   Sleep(2000);
//--- 凡例を無効にする
   pie_chart.LegendAlignment(ALIGNMENT_LEFT);
   Sleep(2000);
//--- 凡例を無効にする
   pie_chart.LegendAlignment(ALIGNMENT_RIGHT);
   Sleep(2000);
//--- 凡例を無効にする
   pie_chart.LegendAlignment(ALIGNMENT_TOP);
   Sleep(2000);
//--- 凡例を無効にする
```

© 2000–2025, MetaQuotes Ltd.

```
   pie_chart.ShowLegend(false);
   Sleep(2000);
//--- パーセンテージを無効にする
   pie_chart.ShowPercent(false);
   Sleep(2000);
//--- 記述子を無効にする
   pie_chart.ShowDescriptors(false);
   Sleep(2000);
//--- 全てを有効にする
   pie_chart.ShowLegend();
   pie_chart.ShowValue();
   pie_chart.ShowDescriptors();
   Sleep(2000);
//--- またはこのように
   pie_chart.ShowFlags(FLAG_SHOW_LEGEND|FLAG_SHOW_DESCRIPTORS|FLAG_SHOW_PERCENT);
   uint total=pie_chart.DataTotal();
//--- 値と遊ぶ
   for(uint i=0;i<total && !IsStopped();i++)
     {
      pie_chart.ValueUpdate(i,100*(rand()%10+1));
      Sleep(1000);
     }
//--- 色と遊ぶ
   for(uint i=0;i<total && !IsStopped();i++)
     {
      pie_chart.ColorUpdate(i%total,RandomRGB());
      Sleep(1000);
     }
//--- 回転する
   while(!IsStopped())
     {
      pie_chart.DataOffset(pie_chart.DataOffset()+1);
      Sleep(200);
     }
//--- 終わる
   pie_chart.Destroy();
   return(0);
  }
//+------------------------------------------------------------------+
//| ランダムなRGB色                                                  |
//+------------------------------------------------------------------+
uint RandomRGB(void)
  {
   return(XRGB(rand()%255,rand()%255,rand()%255));
  }
```

© 2000–2025, MetaQuotes Ltd.

Create

グラフィカルリソースを作成する仮想メソッド。

```
virtual bool  Create(
  const string       name,    // 名前
  const int          width,   // 幅
  const int          height,  // 高さ
  ENUM_COLOR_FORMAT  clrfmt,  // 形式
  )
```

パラメータ

name

[in] グラフィカルリソース名の基底。リソース名は、作成中に擬似乱数文字列を追加することによって生成されます。

width

[in] ピクセル単位での幅（X軸に沿ったサイズ）

height

[in] ピクセル単位での高さ（Y軸に沿ったサイズ）

clrfmt

[in] 色処理メソッド。色処理メソッドの詳細については、ResourceCreate()関数の説明をご参照ください。

戻り値

成功した場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

SeriesSet

円グラフに表示される系列の値を設定します。

```
bool  SeriesSet(
  const double&  value[],  // 値
  const string&  text[],   // ラベル
  const uint&    clr[],    // 色
  )
```

パラメータ

value[]

[in] 値の配列

text[]

[in] 値のラベルの配列

clr[]

[in] 値の色の配列

戻り値

成功した場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

ValueAdd

円グラフの最後に新しい値を追加します。

```
bool  ValueAdd(
  const double  value,  // 値
  const string  descr,  >// ラベル
  const uint    clr,    // 色
  )
```

パラメータ

value

[in] 値

descr

[in] 値のラベル

clr

[in] 値の色

戻り値

成功した場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

ValueInsert

円グラフの指定された位置に新しい値を挿入します。

```
bool  ValueInsert(
  const uint    pos,    // インデックス
  const double  value,  // 値
  const string  descr,  >// ラベル
  const uint    clr,    // 色
  )
```

パラメータ

pos

[in] 挿入に使われるインデックス

value

[in] 値

descr

[in] 値のラベル

clr

[in] 値の色

戻り値

成功した場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

ValueUpdate

円グラフの指定された位置での値を更新します。

```
bool  ValueUpdate(
  const uint    pos,    // インデックス
  const double  value,  // 値
  const string  descr,  >// ラベル
  const uint    clr,    // 色
  )
```

パラメータ

pos

[in] 値のインデックス－0から始まったその追加の系列番号

value

[in] 値

descr

[in] 値のラベル

clr

[in] 値の色

戻り値

成功した場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

ValueDelete

円グラフの指定された位置での値を削除します。

```
bool  ValueDelete(
  const uint  pos,  // インデックス
  )
```

パラメータ

pos

[in] 値のインデックス－0から始まったその追加の系列番号

戻り値

成功した場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

DrawChart

円グラフとそのすべての要素を描画するための仮想メソッド。

```
virtual void  DrawChart()
```

© 2000–2025, MetaQuotes Ltd.

DrawPie

指定された値に対応する円グラフのセグメントを描画します。

```
void  DrawPie(
  double       fi3,    // 円弧の最初の境界を定義する円グラフ中心からの線の角度
  double       fi4,    // 円弧の第2の境界を定義する円グラフ中心からの線の角度
  int          idx,    // インデックス
  CPoint&      p[],  //
  const uint   clr,    //
  )
```

パラメータ

fi3

[in] アークの最初の境界を定義する角度（ラジアン単位）

fi4

[in] アークの2番目の境界を定義する角度（ラジアン単位）

idx

[in] セグメントが対応する値のインデックス

p[]

[in] セグメントをプロットするための参照点（x, y）の配列

clr

[in] セグメントの色

© 2000–2025, MetaQuotes Ltd.

LabelMake

セグメント ラベルをその値と元 のラベルに基 づいて生 成します。

```
string  LabelMake(
  const string  text,     // ラベル
  const double  value,    // 値
  const bool    to_left,  // フラグ
  )
```

パラメータ

text

[in] ラベル

value

[in] 値

to_left

[in] ラベルレイアウト の順 序 を定 義します。

- true — ラベルそして値 .
- false — 値 そしてラベル

戻り値

セグメント のラベル

© 2000–2025, MetaQuotes Ltd.

3Dグラフィックス

このセクションでは、3次元グラフィックス開発のクラスについて説明します。クラスは、DirectXを操作するための関数に基づいています。CCnavas3Dは、カメラと照明を管理するためのメソッドを含む基本クラスで、テクスチャ、シェーダー、頂点バッファー、インデックス、シェーダーのパラメータなどのグラフィックリソースのマネージャを備えています。

ライブラリの使用をはじめるにあたって、METATRADER5でDIRECTXを使用して3Dグラフィックスを作成する方法稿をお読みください。

さらに、ボックス、ユーザーデータの3次元表面、任意のグリッドなど、シーンベースオブジェクトのクラスがあります。

© 2000–2025, MetaQuotes Ltd.

CCanvas3D

CCanvas3Dは、チャート上の3Dオブジェクトを簡単に作成および視覚化するためのクラスです。

説明

CCanvas3Dは、アニメーション化された3Dグラフィックスの形で大量のデータの作成と視覚化を大幅に簡素化します。このクラスには、カメラと照明を管理するためのメソッドと、グラフィックリソースを作成するためのリソースマネージャ（テクスチャ、シェーダー、頂点バッファー、インデックス、シェーダーのパラメータ）の機能が含まれています。

さらに、ライブラリには、ボックス、ユーザデータの3次元表面、または任意のグリッドなど、シーンベースオブジェクトのクラスが含まれています。

機能が機能するには、ビデオカードがDX 11およびShader Model 5.0をサポートする必要があります。

Declaration

```
class CCanvas
```

タイトル

```
#include <Canvas\Canvas.mqh>
```

継承階層

CCanvas

CCanvas3D

グループ分けされたクラスメソッド

作成と解除	説明
Create	チャートオブジェクトに結合しないグラフィックリソースを作成します。
Attach	OBJ_BITMAP_LABEL オブジェクトからグラフィックオブジェクトを取得してCCanvasクラスのインスタンスに接続します。
ObjectAdd	後続のレンダリングのためにオブジェクトを3Dシーンに追加します。
Destroy	グラフィックリソースを破棄し、グラフィック3Dコンテキストを解放します。
照明	
AmbientColorSet	アンビエントオールラウンド照明の色と強度を設定します。
AmbientColorGet	アンビエントオールラウンド照明の色と強度を取得します。
LightDirectionSet	有向光源の方向を設定します。
LightDirectionGet	有向光源の方向を取得します。
LightColorSet	指向性光源の色と強度を設定します。

© 2000-2025, MetaQuotes Ltd.

作成と解除	説明
LightColorGet	指向性光源の色と強度を取得します。
カメラ	
ProjectionMatrixSet	3D座標投影行列を計算して2Dフレームに設定します。
ProjectionMatrixGet	3Dシーン投影行列を2Dフレームに取得します。
ViewMatrixSet	3Dシーン投影行列を設定します。
ViewMatrixGet	3Dシーン投影行列を取得します。
ViewPositionSet	3Dシーンに視点を設定します。
ViewRotationSet	3Dシーンでの視線の方向を設定します。
ViewTargetSet	視線が向けられるポイントの座標を設定します。
ViewUpDirectionSet	3D空間の上部フレームの境界線の方向を設定します。
レンダリング	
Render	後続の表示のために、フレーム内部バッファーのすべてのシーンオブジェクトをレンダリングします。
RenderBegin	新しいフレームをレンダリングするためのグラフィックコンテキストを準備します。
RenderEnd	レンダリングされたフレームを内部バッファーにコピーし、必要に応じてチャート画像を更新します。
リソースの取得	
DXContext	グラフィックコンテキストハンドルを取得します。
DXDispatcher	リソースディスパッチャハンドルを取得します。
InputScene	シーンパラメータのバッファへのポインタを取得します。

© 2000–2025, MetaQuotes Ltd.

AmbientColorGet

アンビエントオールラウンド照明の色と強度を取得します。

```
void  AmbientColorGet(
   DXColor  &ambient_color      // オールラウンドな照明の色と強度
   );
```

パラメータ

&ambient_color

[out] オールラウンドな照明の色と強度

戻り値

なし

注意事項

強度はDXColor構造体のアルファチャネルに保存されます。

© 2000–2025, MetaQuotes Ltd.

AmbientColorSet

アンビエントオールラウンド照明の色と強度を設定します。

```
void  AmbientColorSet(
   const DXColor  &ambient_color      // オールラウンドな照明の色と強度
   );
```

パラメータ

&ambient_color

[in] オールラウンドな照明の色と強度

戻り値

なし

注意事項

強度はDXColor構造体のアルファチャネルで設定されます。

© 2000–2025, MetaQuotes Ltd.

Attach

OBJ_BITMAP_LABELオブジェクトからグラフィックオブジェクトを取得してCCanvasクラスインスタンスに接続します。

```
bool  Attach(
   const long          chart_id,                          // チャート識別子
   const string        objname,                           // オブジェクト名
   ENUM_COLOR_FORMAT   clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // 色処理方法
   )
```

OBJ_BITMAP_LABELのためのグラフィックリソースを作成してCCanvasクラスインスタンスに接続します。

```
bool  Attach(
   const long          chart_id,                          // チャート識別子
   const string        objname,                           // オブジェクト名
   const int           width,                             // 画像幅(ピクセル単位)
   const int           height,                            // 画像の高さ(ピクセル単位)
   ENUM_COLOR_FORMAT   clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // 色処理方法
   )
```

パラメータ

chart_id

[in] チャート識別子。

objname

[in] グラフィックオブジェクトの名称

width

[in] リソースのフレーム幅

height

[in] フレームの高さ

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] 色処理方法色処理方法についてはResourceCreate()関数の説明で詳しく知ることが出来ます。

注意事項

成功の場合はtrue、グラフィックオブジェクトを追加できなかった場合はfalse

© 2000-2025, MetaQuotes Ltd.

Create

チャートオブジェクトに結合しないグラフィックリソースを作成します。

```
virtual bool  Create(
   const string       name,                                  // グラフィカルオブジェクト名
   const int          width,                                 // 幅
   const int          height,                                // 高さ
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // 色形式
   );
```

パラメータ

name

[in] グラフィックオブジェクト名

width

[in] フレーム幅

height

[in] フレームの高さ

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] 色処理方法色処理方法についてはResourceCreate() 関数の説明で詳しく知ることが出来ます。

注意事項

リソースが作成された場合はtrue、それ以外の場合はfalse

© 2000-2025, MetaQuotes Ltd.

Destroy

グラフィックリソースを破棄し、グラフィック3Dコンテキストを解放します。

```
virtual void  Destroy()
```

戻り値

なし

注意事項

グラフィックリソースがチャートオブジェクトに結合されている場合チャートオブジェクトが削除されます。

© 2000–2025, MetaQuotes Ltd.

DXContext

グラフィックコンテキスト ハンドルを取得します。

```
int  DXContext()
```

戻り値

グラフィックコンテキスト ハンドル

© 2000–2025, MetaQuotes Ltd.

DXDispatcher

リソースディスパッチャハンドルを取得します。

```
CDXDispatcher*  DXDispatcher()
```

戻り値

リソースディスパッチャハンドル

© 2000–2025, MetaQuotes Ltd.

InputScene

シーンパラメータのバッファへのポインタを取 得します。

```
CDXInput*  InputScene()
```

戻り値

シーンパラメータのバッファへのポインタ

© 2000–2025, MetaQuotes Ltd.

LightColorGet

指向性光源の色と強度を取得します。

```
void  LightColorGet(
   DXColor  &light_color      // 指向性照明の色と強度
   );
```

パラメータ

&light_color

[out] 指向性照明の色と強度

戻り値

なし

注意事項

強度はDXColor構造体のアルファチャネルに保存されます。

© 2000–2025, MetaQuotes Ltd.

LightColorSet

指向性光源の色と強度を設定します。

```
void  LightColorSet(
   const DXColor  &light_color      // 指向性照明の色と強度
   );
```

パラメータ

&light_color

[in] 指向性照明の色と強度

戻り値

なし

注意事項

強度はDXColor構造体のアルファチャネルで設定されます。

© 2000–2025, MetaQuotes Ltd.

LightDirectionGet

有向光源の方向を取得します。

```
void  LightDirectionGet(
   DXVector3  &light_direction      // 方向ベクトル
   );
```

パラメータ

&light_direction

[out] 方向ベクトル

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

LightDirectionSet

有向光源の方向を設定します。

```
void  LightDirectionSet(
   const DXVector3  &light_direction      // 方向ベクトル
   );
```

パラメータ

&light_direction

[in] 方向ベクトル

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

ObjectAdd

後続のレンダリングのためにオブジェクトを3Dシーンに追加します。

```
bool  ObjectAdd(
   CDXObject  *object       // オブジェクトへのポインタ
   );
```

パラメータ

object

[in] CDXObject抽象クラスから派生したクラスのインスタンスへのポインタ

戻り値

成功の場合はtrue、3Dグラフィックオブジェクトを追加できなかった場合はfalse

© 2000-2025, MetaQuotes Ltd.

ProjectionMatrixGet

3Dシーン投影行列を2Dフレームに取得します。

```
void  ProjectionMatrixGet(
   DXMatrix  &projection_matrix      // 投影行列
   );
```

パラメータ

&projection_matrix

[out] 投影行列

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

ProjectionMatrixSet

3D座標投影行列を計算して2Dフレームに設定します。

```
void  ProjectionMatrixSet(
   float  fov,               // 視野
   float  aspect_ratio,      // フレームアスペクト比
   float  z_near,            //
   float  z_far              //
   );
```

パラメータ

fov

[in] シーンの投影を作成するためのラジアン単位の視野幅

aspect_ratio

[in] 2Dフレームアスペクト比

z_near

[in] ニアクリッピングプレーンまでの距離

z_far

[in] ファークリッピングプレーンまでの距離

戻り値

なし

注意事項

2Dフレームには、指定された視野内にあり、ニアクリッピングプレーンとファークリッピングプレーンの間にある3Dオブジェクトの投影のみが表示されます。

© 2000–2025, MetaQuotes Ltd.

Render

後続の表示のために、フレーム内部バッファーのすべてのシーンオブジェクトをレンダリングします。

```
bool  Render(
   uint  flags,                  // フラグの組み合わせ
   uint  background_color=0      // 背景色
   );
```

パラメータ

flags

[in] レンダリングモードを設定するフラグの組み合わせ可能な値:
DX_CLEAR_COLOR - *背景色*を使用して画像バッファをクリアします、
DX_ CLEAR_ DEPTH – 深度バッファをクリアします

background_color=0

[in] 3Dシーンの背景色

戻り値

成功の場合はtrue、レンダリングできなかった場合はfalse

注意事項

Render()を呼び出しても、チャート上のシーンは更新されません。代わりに、画像の内部バッファーのみが更新されます。更新されたフレームをレンダリングするには、Update()メソッドを明示的に呼び出す必要があります。

Render()ではRenderBeginおよびRenderEnd()呼び出しがあります。

© 2000–2025, MetaQuotes Ltd.

RenderBegin

新しいフレームをレンダリングするためのグラフィックコンテキストを準備します。

```
virtual bool  RenderBegin(
   uint  flags,                  // フラグの組み合わせ
   uint  background_color=0      // 背景色
   );
```

パラメータ

flags

[in] レンダリングモードを設定するフラグの組み合わせ可能な値:
DX_CLEAR_COLOR - *背景色*を使用して画像バッファをクリアします、
DX_ CLEAR_ DEPTH – 深度バッファをクリアします

background_color=0

[in] 3Dシーンの背景色

戻り値

成功の場合はtrue、シェーダー入力を更新できなかった場合はfalse

© 2000–2025, MetaQuotes Ltd.

RenderEnd

レンダリングされたフレームを内部バッファーにコピーし、必要に応じてチャート画像を更新します。

```
virtual bool  RenderEnd(
   bool  redraw=false      // 更新フラグ
   );
```

パラメータ

redraw=false

[in] チャートの再描画の必要性を示すフラグ

戻り値

成功の場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

ViewMatrixGet

3Dシーン投影行列を取得します。

```
void  ViewMatrixGet(
   DXMatrix  &view_matrix      // ビュー行列
   );
```

パラメータ

&view_matrix

[out] 3D空間でカメラの位置と方向を設定するビュー行列

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

ViewMatrixSet

3Dシーン投影行列を設定します。

```
void  ViewMatrixSet(
   const DXMatrix  &view_matrix      // ビュー行列
   );
```

パラメータ

&view_matrix

[in] 3D空間でカメラの位置と方向を設定するビュー行列

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

ViewPositionSet

3Dシーンに視点を設定します。

```
void  ViewPositionSet(
   const DXVector3  &position      // 視点の位置
   );
```

パラメータ

&position

[in] 3Dシーンでの視点設定

戻り値

なし

注意事項

ViewPositionSet()を使用して視点位置を設定すると、ViewMatrixGet()で取得したビュー行列が変更されます。

© 2000–2025, MetaQuotes Ltd.

ViewRotationSet

3Dシーンでの視線の方向を設定します。

```
void  ViewRotationSet(
   const DXVector3  &rotation      // 回転角のベクトル
   );
```

パラメータ

&rotation

[in] 3Dシーンでの視線の方向を計算するためのベクトル設定オイラー角

戻り値

なし

注意事項

ViewRotationSet()を使用して視線方向を設定すると、ViewMatrixGet()で取得したビュー行列が変更されます。

© 2000–2025, MetaQuotes Ltd.

ViewTargetSet

視線が向けられるポイントの座標を設定します。

```
void  ViewTargetSet(
   const DXVector3  &target      // ターゲット座標
   );
```

パラメータ

&target

[in] 視線が向けられるポイントの座標

戻り値

なし

注意事項

視点を移動するときに、1つのシーンポイントの視線を修正するために使用します。

ViewRotationSet()を使用して新しいターゲット座標を設定すると、ViewMatrixGet()で取得したビュー行列が変更されます。

ViewTargetSet()はViewUpDirectionSet()と併用して、視線方向を定義します。

© 2000–2025, MetaQuotes Ltd.

ViewUpDirectionSet

3D空間の上部フレームの境界線の方向を設定します。

```
void  ViewUpDirectionSet(
   const DXVector3  &up_direction      // トップ方向
   );
```

パラメータ

&up_direction

[in] 3D空間でのフレームの上部の方向

戻り値

なし

注意事項

ViewRotationSet()を使用して新しい方向を設定すると、ViewMatrixGet()で取得したビュー行列が変更されます。

ViewUpDirectionSet()は>ViewTargetSet()と併用して、視線方向を定義します。

© 2000–2025, MetaQuotes Ltd.

CChart

CChart クラスは「Chart（チャート）」グラフィックオブジェクト のプロパティに簡 単 にアクセスするためのクラスです。

説 明

CChart クラスは「Chart（チャート）」オブジェクト プロパティへのアクセスを提 供 します。

宣 言

```
class CChart : public CObject
```

タイト ル

```
#include <Charts\Chart.mqh>
```

継 承 階 層

CObject
CChart

クラスメソッド

保 護 されたデータへのアクセス	
ChartID	チャート の識 別 子 を取 得 します。
一 般 的 なプロパティ	
Mode	「Mode」プロパティ（bar、candle または line）の値 を取 得 /設 定 します。
Foreground	「Foreground（前 景 ）」プロパティの値 を取 得 /設 定 します。
Shift	「Shift（シフト ）」プロパティの値 を取 得 /設 定 します。
ShiftSize	「ShiftSize（シフト サイズ）」プロパティの値 を（百 分 率 で）取 得 /設 定 します。
AutoScroll	「「AutoScroll（自 動 スクロール）」プロパティの値 を取 得 /設 定 します。
Scale	「Scale（スケール）」プロパティの値 を取 得 /設 定 します。
ScaleFix	「ScaleFix」プロパティ（チャート スケールが固 定 かどうか）の値 を取 得 /設 定 します。
ScaleFix_11	「ScaleFix_11」プロパティ（チャート スケールが 1:1 かどうか）の値 を取 得 /設 定 します。
FixedMax	「FixedMax」プロパティ（固 定 された最 高 値 ）の値 を取 得 /設 定 します。
FixedMin	「FixedMin」プロパティ（固 定 された最 安 値 ）の値 を取 得 /設 定 します。

© 2000–2025, MetaQuotes Ltd.

保護されたデータへのアクセス	
ScalePPB	「ScalePPB」プロパティ(「バーあたりのポイント」スケールかどうか)の値を取得/設定します。
PointsPerBar	「PointsPerBar」プロパティの値を(バーあたりのポイント単位で)取得/設定します。
表示プロパティ	
ShowOHLC	「ShowOHLC」プロパティの値を取得/設定します。
ShowLineBid	「ShowLineBid」プロパティの値を取得/設定します。
ShowLineAsk	「ShowLineAsk」プロパティの値を取得/設定します。
ShowLastLine	「ShowLastLine」プロパティの値を取得/設定します。
ShowPeriodSep	「ShowPeriodSep」プロパティ(期間の区切りの表示/非表示)の値を取得/設定します。
ShowGrid	「ShowGrid」プロパティの値を取得/設定します。
ShowVolumes	「ShowVolumes」プロパティ(未決済ポジションのボリュームとレベルの色)の値を取得/設定します。
ShowObjectDescr	「ShowObjectDescr」プロパティ(グラフィックオブジェクトの記述を表示)の値を取得/設定します。
ShowDateScale	「ShowDateScale」プロパティ(チャートの日付のスケール)の値を設定します。
ShowPriceScale	「ShowPriceScale」プロパティ(チャートの価格のスケール)の値を設定します。
色のプロパティ	
ColorBackground	「ColorBackground」プロパティ(チャートの背景色)の値を取得/設定します。
ColorForeground	「ColorForeground」プロパティ(軸、チャートのスケールと OHLC 文字列の色)の値を取得/設定します。
ColorGrid	「ColorGrid」プロパティ(グリッドの色)の値を取得/設定します。
ColorBarUp	「ColorBarUp」プロパティ(強気バーの色と影の色とローソク足のアウトラインの色)の値を取得/設定します。
ColorBarDown	「ColorBarDow」プロパティ(弱気バーの色と影の色とローソク足のアウトラインの色)の値を取得/設定します。
ColorCandleBull	「ColorCandleBull」プロパティ(強気ローソク足の色)の値を取得/設定します。
ColorCandleBear	「ColorCandleBear」プロパティ(弱気ローソク足の色)の値を取得/設定します。
ColorChartLine	「ColorChartLine」プロパティ(ラインチャートと同時線の色)の値を取得/設定します。

© 2000–2025, MetaQuotes Ltd.

保護されたデータへのアクセス	
ColorVolumes	「ColorVolumes」プロパティ(未決済ポジションのボリュームとレベルの色)の値を取得/設定します。
ColorLineBid	「ColorLineBid」プロパティ(買値線の色)の値を取得/設定します。
ColorLineAsk	「ColorLineAsk」プロパティ(買値線の色)の値を取得/設定します。
ColorLineLast	「ColorLineLast」プロパティの値を取得/設定します。(最後の約定価格線の色)
ColorStopLevels	「ColorStopLevels」プロパティの値を取得/設定します。(SL 及び TPレベルの色)
読み込み専用のプロパティ	
VisibleBars	表示されているチャートバーの総数を取得します。
WindowsTotal	チャート指標サブウィンドウを含んだチャートウィンドウの総数を取得します。
WindowIsVisible	指定されたチャートサブウィンドウの可視フラグを取得します。
WindowHandle	チャートのウィンドウハンドル(HWND)を取得します。
FirstVisibleBar	チャートに表示されている初めのバーの番号を取得します。
WidthInBars	バー単位でのウィンドウ幅を取得します。
WidthInPixels	ピクセル単位でのサブウィンドウ幅を取得します。
HeightInPixels	ピクセル単位でのサブウィンドウの高さを取得します。
PriceMin	指定されたサブウィンドウの安値を取得します。
PriceMax	指定されたサブウィンドウの高値を取得します。
プロパティ	
Attach	現在のチャートをクラスインスタンスに割り当てます。
FirstChart	クライアント端末の初めのチャートをクラスインスタンスに割り当てます。
NextChart	クライアント端末の次のチャートをクラスインスタンスに割り当てます。
Open	チャートを指定されたパラメータで開いてクラスインスタンスに割り当てます。
Detach	チャートをクラスインスタンスから切り離します。
Close	クラスインスタンスに割り当てたチャートを閉じます。
BringToTop	他のチャートの上に重ねてチャートを表示します。

© 2000-2025, MetaQuotes Ltd.

保護されたデータへのアクセス	
EventObjectCreate	チャート上の新しいオブジェクトの作成のイベントの通知を送信するためのフラグを設定します。
EventObjectDelete	チャート上のオブジェクトの削除のイベントの通知を送信するためのフラグを設定します。
指標	
IndicatorAdd	指定されたチャートサブウィンドウに指定されたハンドルを持つ指標を追加します。
IndicatorDelete	指定された名称を持つ指標を指定されたチャートサブウィンドウから削除します。
IndicatorsTotal	指定されたチャートサブウィンドウに適用された全ての指標の数を返します。
IndicatorName	指定されたチャートサブウィンドウ上の指標の短縮名を返します。
ナビゲート	
Navigate	チャートをナビゲートします。
MQL5 API へのアクセス	
Symbol	チャートのシンボルを取得します。
Period	チャートの期間を取得します。
Redraw	クラスインスタンスに割り当てたチャートを再描画します。
GetInteger	この関数は、対応するオブジェクトプロパティの値を返します。
SetInteger	整数型のプロパティに新しい値を設定します。
GetDouble	この関数は、対応するオブジェクトプロパティの値を返します。
SetDouble	double 型のプロパティに新しい値を設定します。
GetString	この関数は、対応するオブジェクトプロパティの値を返します。
SetString	string 型のプロパティに新しい値を設定します。
SetSymbolPeriod	クラスインスタンスに割り当てられたチャートの銘柄と期間を変更します。
ApplyTemplate	指定されたテンプレートをチャートに適用します。
ScreenShot	指定されたチャートのスクリーンショットを作成して .GIF ファイルに保存します。
WindowOnDropped	オブジェクト（エキスパートまたはスクリプト）ドロップポイントに対応するチャートサブウィンドウの番号を取得します。
PriceOnDropped	オブジェクト（エキスパートまたはスクリプト）ドロップポイントに対応する価格座標を取得します。

© 2000–2025, MetaQuotes Ltd.

保護されたデータへのアクセス	
TimeOnDropped	オブジェクト（エキスパートまたはスクリプト）ドロップポイントに対応する時間座標を取得します。
XOnDropped	オブジェクト（エキスパートまたはスクリプト）ドロップポイントに対応するX座標を取得します。
YOnDropped	オブジェクト（エキスパートまたはスクリプト）ドロップポイントに対応するY座標を取得します。
入出力	
virtual Save	オブジェクトパラメータをファイルに保存します。
virtual Load	オブジェクトパラメータをファイルから読み込みます。
virtual Type	グラフィックオブジェクトの型の識別子を取得します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

© 2000–2025, MetaQuotes Ltd.

ChartID

チャートの識別子を返します。

```
long  ChartID() const
```

戻り値

クラスインスタンスに割り当てたチャートの識別子（割り当てられたオブジェクトが存在しない場合は -1 ）

© 2000-2025, MetaQuotes Ltd.

Mode（Get メソッド）

「Mode」プロパティ（bar、candle または line）の値を取得します。

```
ENUM_CHART_MODE  Mode() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「Mode」プロパティの値（割り当てられたオブジェクトが存在しない場合は WRONG_VALUE）

Mode（Set メソッド）

「Mode」プロパティ（bar、candle または line）の新しい値を設定します。

```
bool  Mode(
   ENUM_CHART_MODE  mode       // 新しいチャートモード
  )
```

パラメータ

mode

[in] ENUM_CHART_MODE 列挙でのチャートモード（bar、candle または line）

戻り値

成功の場合は true、mode が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Foreground（Get メソッド）

「Foreground（前景）」プロパティの値を取得します。

```
bool  Foreground() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「Foreground（前景）」プロパティの値（割り当てられたオブジェクトが存在しない場合は false）

Foreground（Set メソッド）

「Foreground（前景）」プロパティの新しい値を設定します。

```
bool  Foreground(
   bool  foreground      // 新しいフラグ値
  )
```

パラメータ

foreground

[in]「Foreground（前景）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Shift（Get メソッド）

「Shift（シフト）」プロパティの値を取得します。

```
bool  Shift() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「Shift（シフト）」プロパティの値（割り当てられたオブジェクトが存在しない場合は false）

Shift（Set メソッド）

「Shift（シフト）」プロパティの新しい値を設定します。

```
bool  Shift(
   bool  shift       // 新しいフラグ値
  )
```

パラメータ

shift

[in]「Shift（シフト）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000-2025, MetaQuotes Ltd.

ShiftSize（Get メソッド）

「ShiftSize（シフト サイズ）」プロパティの値を（百分率で）取得します。

```
double  ShiftSize() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「ShiftSize（シフト サイズ）」プロパティの値（割り当てられたオブジェクトが存在しない場合は EMPTY_VALUE ）

ShiftSize（Set メソッド）

「Shift（シフト）」プロパティの新しい値を（百分率で）設定します。

```
bool  ShiftSize(
   double  shift_size      // 新しいプロパティ値
  )
```

パラメータ

shift_size

[in]「ShiftSize（シフト サイズ）」プロパティの新しい（百分率の）値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

AutoScroll（Get メソッド）

「「AutoScroll（自動スクロール）」プロパティの値を取得します。

```
bool  AutoScroll() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「「AutoScroll（自動スクロール）」プロパティの値（割り当てられたオブジェクトが存在しない場合は false ）

AutoScroll（Set メソッド）

「「AutoScroll（自動スクロール）」プロパティの新しい値を設定します。

```
bool  AutoScroll(
   bool  autoscroll      // 新しいフラグ値
  )
```

パラメータ

autoscroll

[in] 「「AutoScroll（自動スクロール）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000-2025, MetaQuotes Ltd.

Scale（Get メソッド）

「Scale（スケール）」プロパティの値を取得します。

```
int  Scale() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「Scale（スケール）」プロパティの値（割り当てられたオブジェクトが存在しない場合は 0）

Scale（Set メソッド）

「Scale（スケール）」プロパティの新しい値を設定します。

```
bool  Scale(
   int  scale      // 新しい値
  )
```

パラメータ

scale

[in]「Scale（スケール）」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

ScaleFix（Get メソッド）

「ScaleFix」プロパティ（チャート スケールが固 定 かどうか）の値 を取 得 します。

```
bool  ScaleFix() const
```

戻 り値

クラスインスタンスに割 り当 てられたオブジェクト の「ScaleFix」プロパティの値（割 り当 てられたオブジェクト が存 在 しない場 合 は false ）

ScaleFix（Set メソッド）

「ScaleFix」プロパティの新 しい値 を設 定 します。

```
bool  ScaleFix(
   bool  scale_fix      // 新しい値
  )
```

パラメータ

scale_fix

[in]「ScaleFix」プロパティの新 しい値

戻 り値

成 功 の場 合 は true、プロパティ値 が変 更 できなかった場 合 は false

© 2000–2025, MetaQuotes Ltd.

ScaleFix_11（Get メソッド）

「ScaleFix_11」プロパティ（チャートスケールが 1:1 かどうか）の値を取得します。

```
bool  ScaleFix_11() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「ScaleFix_11」プロパティの値（割り当てられたオブジェクトが存在しない場合は false）

ScaleFix_11（Set メソッド）

「ScaleFix_11」プロパティの新しい値を設定します。

```
bool  ScaleFix_11(
   string  scale_11      // 新しい値
  )
```

パラメータ

scale_11

[in]「ScaleFix_11」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000-2025, MetaQuotes Ltd.

FixedMax（Get メソッド）

「FixedMax」プロパティ（固定された最高値）の値を取得/設定します。

```
double  FixedMax() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「FixedMax」プロパティの値（割り当てられたオブジェクトが存在しない場合は EMPTY_VALUE ）

FixedMax（Set メソッド）

「FixedMax」プロパティの新しい値を設定します。

```
bool  FixedMax(
   double  max       // 新しい固定最高値
  )
```

パラメータ

max

[in]「FixedMax」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

FixedMin（Get メソッド）

「FixedMin」プロパティ（固定された最安値）の値を取得します。

```
double  FixedMin() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「FixedMin」プロパティの値（割り当てられたオブジェクトが存在しない場合は EMPTY_VALUE ）

FixedMin（Set メソッド）

「FixedMin」プロパティの新しい値を設定します。

```
bool  FixedMax(
   double  min      // 新しい固定最安値
  )
```

パラメータ

max

[in]「FixedMin」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

PointsPerBar（Get メソッド）

「PointsPerBar」プロパティの値を（ポイント パーバー単位で）取得します。

```
double  PointsPerBar() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「PointsPerBar」プロパティの値（割り当てられたオブジェクトが存在しない場合は EMPTY_VALUE ）

PointsPerBar（Set メソッド）

「PointsPerBar」プロパティの新しい値を設定します。

```
bool  PointsPerBar(
   double  ppb       // 新しいスケール(バーあたりのポイント単位)
  )
```

パラメータ

ppb

[in] 新しいスケール（バーあたりのポイント単位）

戻り値

成功の場合は true、スケールが変更できなかった場合は false

© 2000-2025, MetaQuotes Ltd.

ScalePPB（Get メソッド）

「ScalePPB」プロパティの値（「ポイント パーバー」スケールかどうか）を取得します。

```
bool  ScalePPB() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「ScalePP」プロパティの値（割り当てられたオブジェクトが存在しない場合は false）

ScalePPB（Set メソッド）

「ScalePPB」プロパティの新しい値を設定します。

```
bool  ScalePPB(
   bool  scale_ppb      // 新しいフラグ値
  )
```

パラメータ

scale_ppb

[in]「ScalePPB」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

ShowOHLC（Get メソッド）

「ShowOHLC」プロパティの値を取得します。

```
bool   ShowOHLC() const
```

戻り値

クラスインスタンスに割り当てられたオブジェクトの「ShowOHLC」プロパティの値（割り当てられたオブジェクトが存在しない場合は false）

ShowOHLC（Set メソッド）

「howOHLC」プロパティの新しい値を設定します。

```
bool   ShowOHLC(
   bool   show       // 新しい値
  )
```

パラメータ

show

[in]「ShowOHLC」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

ShowLineBid（Get メソッド）

「ShowLineBid」プロパティの値を取得します。

```
bool  ShowLineBid() const
```

戻り値

クラスインスタンスに割り当てられたチャートの「ShowLineBid」プロパティの値（割り当てられたチャートが存在しない場合は false）

ShowLineBid（Set メソッド）

「ShowLineBid」プロパティの新しい値を設定します。

```
bool  ShowLineBid(
   bool  show      // 新しい値
  )
```

パラメータ

show

[in]「ShowLineBi」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

ShowLineAsk（Get メソッド）

「ShowLineAsk」プロパティの値を取得します。

```
bool  ShowLineAsk() const
```

戻り値

クラスインスタンスに割り当てられたチャートの「ShowLineAsk」プロパティの値（割り当てられたチャートが存在しない場合は false）

ShowLineAsk（Set メソッド）

「ShowLineAsk」プロパティの新しい値を設定します。

```
bool  ShowLineAsk(
   bool  show      // 新しい値
  )
```

パラメータ

show

[in]「ShowLineAsk」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

ShowLastLine（Get メソッド）

「ShowLastLine」プロパティの値を取得します。

```
bool  ShowLastLine() const
```

戻り値

クラスインスタンスに割り当てられたチャートの「ShowLastLine」プロパティの値（割り当てられたチャートが存在しない場合は false）

ShowLastLine（Set メソッド）

「ShowLastLine」プロパティの新しい値を設定します。

```
bool  ShowLastLine(
   bool  show      // 新しいフラグ値
  )
```

パラメータ

show

[in]「ShowLastLine」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

ShowPeriodSep（Get メソッド）

「ShowPeriodSep」プロパティの値（期間の区切りを表示）を取得します。

```
bool  ShowPeriodSep() const
```

戻り値

クラスインスタンスに割り当てられたチャートの「ShowPeriodSep」プロパティの値（割り当てられたチャートが存在しない場合は false）

ShowPeriodSep（Set メソッド）

「ShowPeriodSep」プロパティの新しい値を設定します。

```
bool  ShowPeriodSep(
   bool  show      // 新しい値
  )
```

パラメータ

show

[in]「ShowPeriodSep」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

ShowGrid（Get メソッド）

「ShowGrid」プロパティの値を取得します。

```
bool  ShowGrid() const
```

戻り値

クラスインスタンスに割り当てられたチャートの「ShowGrid」プロパティの値（割り当てられたチャートが存在しない場合は false）

ShowGrid（Set メソッド）

「ShowGrid」プロパティの新しい値を設定します。

```
bool  ShowGrid(
   bool  show      // 新しい値
  )
```

パラメータ

show

[in]「ShowGrid」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000-2025, MetaQuotes Ltd.

ShowVolumes（Get メソッド）

「ShowVolumes」プロパティの値を取得します。

```
bool  ShowVolumes() const
```

戻り値

クラスインスタンスに割り当てられたチャートの「ShowVolumes」プロパティの値（割り当てられたチャートが存在しない場合は false）

ShowVolumes（Set メソッド）

「ShowVolumes」プロパティの新しい値を設定します。

```
bool  ShowVolumes(
   bool  show      // 新しい値
  )
```

パラメータ

show

[in]「ShowVolumes」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

ShowObjectDescr (Get Method)

Gets the value of "ShowObjectDescr" property (show description for graphic objects).

```
bool  ShowObjectDescr() const
```

戻り値

クラスインスタンスに割り当てられたチャートの「ShowObjectDescr」プロパティの値．（割り当てられたチャートが存在しない場合は false ）

ShowObjectDescr（Set メソッド）

「ShowObjectDescr」プロパティの新しい値を設定します。

```
bool  ShowObjectDescr(
   bool  show      // 新しい値
  )
```

パラメータ

show

[in]「ShowObjectDescr」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

ShowDateScale

「ShowDateScale」プロパティの新しい値を設定します。

```
bool  ShowDateScale(
   bool  show      // 新しい値
  )
```

パラメータ

show

[in]「ShowDateScale」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000-2025, MetaQuotes Ltd.

ShowPriceScale

「ShowPriceScale」プロパティの新しい値を設定します。

```
bool  ShowPriceScale(
   bool  show      // 新しい値
  )
```

パラメータ

show

[in]「ShowPriceScale」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000-2025, MetaQuotes Ltd.

ColorBackground（Get メソッド）

「ColorBackground」プロパティ（チャートの背景色）の値を取得します。

```
color  ColorBackground() const
```

戻り値

クラスインスタンスに割り当てられたチャートの「ColorBackground」プロパティの値（割り当てられたチャートが存在しない場合は CLR_NONE ）

ColorBackground（Set メソッド）

「ColorBackground」プロパティの新しい値を設定します。

```
bool  ColorBackground(
   color  new_color      // 新しい背景色
  )
```

パラメータ

new_color

[in] 新しい背景色

戻り値

成功の場合は true、色が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

ColorForeground（Get メソッド）

「ColorForeground」プロパティ（軸、チャートのスケールとOHLC文字列の色）の値を取得します。

```
color  ColorForeground() const
```

戻り値

クラスインスタンスに割り当てられたチャートの「ColorForeground」プロパティの値（割り当てられたチャートが存在しない場合は CLR_NONE ）

ColorForeground（Set メソッド）

「ColorForeground」プロパティ（軸、チャートのスケールとOHLC文字列の色）の新しい値を設定します。

```
bool  ColorForeground(
   color  new_color      // 新しい色
  )
```

パラメータ

new_color

[in] 軸、スケールとOHLC文字列の新しい色

戻り値

成功の場合は true、色が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

ColorGrid（Get メソッド）

「ColorGrid」プロパティ（グリッドの色）の値を取得します。

```
color  ColorGrid() const
```

戻り値

クラスインスタンスに割り当てられたチャートの「ColorGrid」プロパティの値（割り当てられたチャートが存在しない場合は CLR_NONE ）

ColorGrid（Set メソッド）

「ColorGrid」プロパティの新しい値を設定します。

```
bool  ColorGrid(
   color  new_color      // 新しいグリッドの色
  )
```

パラメータ

new_color

[in] 新しいグリッドの色

戻り値

成功の場合は true、色が変更できなかった場合は false

© 2000-2025, MetaQuotes Ltd.

ColorBarUp（Get メソッド）

「ColorBarUp」プロパティの（強気バーの色と影の色とローソク足のアウトラインの色）値を取得します。

```
color  ColorBarUp() const
```

戻り値

クラスインスタンスに割り当てられたチャートの「ColorBarUp」プロパティの値（割り当てられたチャートが存在しない場合は CLR_NONE ）

ColorBarUp（Set メソッド）

「ColorBarUp」プロパティの新しい値を設定します。

```
bool  ColorBarUp(
   color  new_color      // 強気バーの新しい色
  )
```

パラメータ

new_color

[in] 強気バーの新しい色

戻り値

成功の場合は true、色が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

ColorBarDown（Get メソッド）

「ColorBarDow」プロパティ（弱気バーの色と影の色とローソク足のアウトラインの色）の値を取得します。

```
color  ColorBarDown() const
```

戻り値

クラスインスタンスに割り当てられたチャートの「ColorBarDown」プロパティの値（割り当てられたチャートが存在しない場合は CLR_NONE）

ColorBarDown（Set メソッド）

「ColorBarDown」プロパティの新しい値を設定します。

```
bool  ColorBarDown(
   color  new_color      // 弱気バーの新しい色
  )
```

パラメータ

new_color

[in] 弱気バーの新しい色

戻り値

成功の場合は true、色が変更できなかった場合は false

© 2000-2025, MetaQuotes Ltd.

ColorCandleBull（Get メソッド）

「ColorCandleBull」（強気ローソク足の色）プロパティの値を取得します。

```
color  ColorCandleBull() const
```

戻り値

クラスインスタンスに割り当てられたチャートの「ColorCandleBull」プロパティの値（割り当てられたチャートが存在しない場合は CLR_NONE ）

ColorCandleBull（Set メソッド）

「ColorCandleBull」プロパティの新しい値を設定します。

```
bool  ColorCandleBull(
   color  new_color      // 強気ローソク足の新しい色
  )
```

パラメータ

new_color

[in] 強気ローソク足の新しい色

戻り値

成功の場合は true、色が変更できなかった場合は false

© 2000-2025, MetaQuotes Ltd.

ColorCandleBear（Get メソッド）

「ColorCandleBear」（弱気ローソク足の色）プロパティの値を取得します。

```
color  ColorCandleBear() const
```

戻り値

クラスインスタンスに割り当てられたチャートの「ColorCandleBear」プロパティの値（割り当てられたチャートが存在しない場合は CLR_NONE ）

ColorCandleBear（Set メソッド）

「ColorCandleBear」プロパティの新しい値を設定します。

```
bool  ColorCandleBear(
   color  new_color      // 弱気ローソク足の新しい色
  )
```

パラメータ

new_color

[in] 強気ローソク足の新しい色

戻り値

成功の場合は true、色が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

ColorChartLine（Get メソッド）

「ColorChartLine」プロパティ（ラインチャートと同時線の色）の値を取得します。

```
color  ColorChartLine() const
```

戻り値

クラスインスタンスに割り当てられたチャートの「ColorChartLine」プロパティの値（割り当てられたチャートが存在しない場合は CLR_NONE）

ColorChartLine（Set メソッド）

「ColorChartLine」プロパティの新しい値を設定します。

```
bool  ColorChartLine(
   color  new_color      // チャートラインの新しい色
  )
```

パラメータ

new_color

[in] チャートライン（同時線）の新しい色

戻り値

成功の場合は true、色が変更できなかった場合は false

© 2000-2025, MetaQuotes Ltd.

ColorVolumes（Get メソッド）

「ColorVolumes」プロパティ（未決済ポジションのボリュームとレベルの色）の値を取得します。

```
color  ColorVolumes() const
```

戻り値

クラスインスタンスに割り当てられたチャートの「ColorVolumes」プロパティの値（割り当てられたチャートが存在しない場合は CLR_NONE）

ColorVolumes（Set メソッド）

「ColorVolumes」プロパティの新しい値を設定します。

```
bool  ColorVolumes(
   color  new_color      // ボリューム(未決済ポジションレベル)の新しい色
  )
```

パラメータ

new_color

[in] ボリューム（未決済ポジションレベル）の新しい色

戻り値

成功の場合は true、色が変更できなかった場合は false

© 2000-2025, MetaQuotes Ltd.

ColorLineBid（Get メソッド）

「ColorLineBid」プロパティ（買値線の色）の値を取得/設定します。

```
color  ColorLineBid() const
```

戻り値

クラスインスタンスに割り当てられたチャートの「ColorLineBid」プロパティの値（割り当てられたチャートが存在しない場合は CLR_NONE）

ColorLineBid（Set メソッド）

「ColorLineBid」プロパティの新しい値を設定します。

```
bool  ColorLineBid(
   color  new_color      // 売値線の新しい色
  )
```

パラメータ

new_color

[in] 売値線の新しい色

戻り値

成功の場合は true、色が変更できなかった場合は false

© 2000-2025, MetaQuotes Ltd.

ColorLineAsk（Get メソッド）

「ColorLineAsk」プロパティ（買値線の色）の値を取得します。

```
color  ColorLineAsk() const
```

戻り値

クラスインスタンスに割り当てられたチャートの「 ColorLineAsk 」プロパティの値（割り当てられたチャートが存在しない場合は CLR_NONE ）

ColorLineAsk（Set メソッド）

「ColorLineAsk」プロパティの新しい値を設定します。

```
bool  ColorLineAsk(
   color  new_color      // 買値線の新しい色
  )
```

パラメータ

new_color

[in] 買値線の新しい色

戻り値

成功の場合は true、色が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

ColorLineLast（Get メソッド）

「ColorLineLast」プロパティ（直近約定価格線の色）の値を取得します。

```
color  ColorLineLast() const
```

戻り値

クラスインスタンスに割り当てられたチャートの「ColorLineLast」プロパティの値（割り当てられたチャートが存在しない場合は CLR_NONE）

ColorLineLast（Set メソッド）

「ColorLineLast」プロパティの新しい値を設定します。

```
bool  ColorLineLast(
   color  new_color      // 直近約定価格線の新しい色
  )
```

パラメータ

new_color

[in] 直近約定価格線の新しい色

戻り値

成功の場合は true、色が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

ColorStopLevels（Get メソッド）

「ColorStopLevels」プロパティ（SL 及び TPレベルの色）の値を取得します。

```
color  ColorStopLevels() const
```

戻り値

クラスインスタンスに割り当てられたチャートの「ColorStopLevels」プロパティの値（割り当てられたチャートが存在しない場合は CLR_NONE）

ColorStopLevels（Set メソッド）

「ColorStopLevels」プロパティの新しい値を設定します。

```
bool  ColorStopLevels(
   color  new_color      // SL 及び TP 価格レベルの新しい色
  )
```

パラメータ

new_color

[in] SL 及び TP 価格レベルの新しい色

戻り値

成功の場合は true、色が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

VisibleBars

表示されているチャートバーの総数を取得します。

```
int  VisibleBars() const
```

戻り値

クラスインスタンスに割り当てられたチャートバーの総数（割り当てられたチャートが存在しない場合は 0）

© 2000–2025, MetaQuotes Ltd.

WindowsTotal

チャート指標サブウィンドウを含んだチャートウィンドウの総数を取得します。

```
int  WindowsTotal() const
```

戻り値

クラスインスタンスに割り当てられたチャート指標サブウィンドウを含んだチャートウィンドウの総数（割り当てられたチャートが存在しない場合は 0）

© 2000–2025, MetaQuotes Ltd.

WindowIsVisible

指定されたチャート サブウィンドウの可視フラグを取得します。

```
bool  WindowIsVisible(
   int  num      // サブウィンドウ番号
   ) const
```

パラメータ

num

[in] サブウィンドウ番号（0 はメインウィンドウ）

戻り値

チャート インスタンスに割り当てられた指定されたチャート サブウィンドウの可視フラグ（割り当てられたチャートが存在しない場合は false ）

© 2000–2025, MetaQuotes Ltd.

WindowHandle

チャート のウィンド ウハンド ル（HW ND)を取 得します。

```
int  WindowHandle() const
```

戻り値

チャート インスタンスに割り当 てられたチャート のウィンド ウハンド ル（（割り当 てられたチャート が存 在しない場 合 は INVALID_HANDLE ）

© 2000–2025, MetaQuotes Ltd.

FirstVisibleBar

チャート上で最初に見えるバーの番号を取得します。

```
int  FirstVisibleBar() const
```

戻り値

チャートインスタンスに割り当てられたチャート上で最初に見えるバーの番号（割り当てられたオブジェクトが存在しない場合は -1 ）

© 2000-2025, MetaQuotes Ltd.

WidthInBars

バー単位でのウィンドウ幅を取得します。

```
int  WidthInBars() const
```

戻り値

チャートインスタンスに割り当てられたバー単位でのウィンドウ幅（割り当てられたチャートが存在しない場合は 0）

© 2000–2025, MetaQuotes Ltd.

WidthInPixels

ピクセル単位でのサブウィンドウ幅を取得します。

```
int  WidthInPixels() const
```

戻り値

チャートインスタンスに割り当てられたピクセル単位でのサブウィンドウ幅（割り当てられたチャートが存在しない場合は 0 ）

© 2000–2025, MetaQuotes Ltd.

HeightInPixels

ピクセル単位でのサブウィンドウの高さを取得します。

```
int  HeightInPixels(
   int  num      // サブウィンドウ番号
   ) const
```

パラメータ

num

[in] サブウィンドウ番号（0はメインウィンドウ）

戻り値

チャートインスタンスに割り当てられたピクセル単位でのサブウィンドウの高さ（割り当てられたチャートが存在しない場合は0）

© 2000-2025, MetaQuotes Ltd.

PriceMin

指定されたサブウィンドウの安値を取得します。

```
double  PriceMin(
   int  num      // サブウィンドウ番号
   ) const
```

パラメータ

num

[in] サブウィンドウ番号（0 はメインウィンドウ）

戻り値

クラスインスタンスに割り当てられたチャートの安値（割り当てられたオブジェクトが存在しない場合は EMPTY_VALUE ）

© 2000–2025, MetaQuotes Ltd.

PriceMax

指定されたサブウィンドウの高値を取得します。

```
double  PriceMax(
   int  num      // サブウィンドウ番号
   ) const
```

パラメータ

num

[in] サブウィンドウ番号（0 はメインウィンドウ）

戻り値

クラスインスタンスに割り当てられたチャートの高値（割り当てられたオブジェクトが存在しない場合は EMPTY_VALUE ）

© 2000–2025, MetaQuotes Ltd.

Attach

現在のチャートをクラスインスタンスに割り当てます。

```
void  Attach()
```

Attach

指定されたチャートをクラスインスタンスに割り当てます。

```
void  Attach(
   long  chart       // チャート識別子
  )
```

パラメータ

chart

[in] 割り当てるチャートの識別子

© 2000–2025, MetaQuotes Ltd.

FirstChart

クライアント端末の初めのチャートをクラスインスタンスに割り当てます。

```
void  FirstChart()
```

© 2000-2025, MetaQuotes Ltd.

NextChart

クライアント端末の次のチャートをクラスインスタンスに割り当てます。

```
void  NextChart()
```

© 2000-2025, MetaQuotes Ltd.

Open

チャートを指定されたパラメータで開いてクラスインスタンスに割り当てます。

```
long  Open(
   const string      symbol_name,      // 銘柄名
   ENUM_TIMEFRAMES   timeframe         // 期間
  )
```

パラメータ

symbol_name

[in] 銘柄名 NULL はエキスパートアドバイザーがに接続されている現材のチャートのシンボルを意味します。

timeframe

[in] チャートの時間軸（ENUM_TIMEFRAMES 列挙）0 は現在の時間軸を表します。

戻り値

チャート識別子

© 2000–2025, MetaQuotes Ltd.

Detach

チャートをクラスインスタンスから切り離します。

```
void  Detach()
```

© 2000-2025, MetaQuotes Ltd.

Close

クラスインスタンスに割り当てたチャートを閉じます。

```
void  Close()
```

© 2000-2025, MetaQuotes Ltd.

BringToTop

他のチャートの上にチャートを重ねて表示します。

```
bool  BringToTop() const
```

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

EventObjectCreate

チャート上の新規オブジェクト作成イベントの通知を全てのMQL5プログラムに送るフラグを設定します。

```
bool  EventObjectCreate(
   bool  flag      // フラグ
  )
```

パラメータ

flag

[in] 新しいフラグ値

戻り値

成功の場合はtrue、フラグが変更できなかった場合はfalse

© 2000-2025, MetaQuotes Ltd.

EventObjectDelete

チャート上のオブジェクト削除イベントの通知を全てのMQL5プログラムに送るフラグを設定します。

```
bool  EventObjectDelete(
   bool  flag      // フラグ
  )
```

パラメータ

flag

[in] 新しいフラグ値

戻り値

成功の場合はtrue、フラグが変更できなかった場合はfalse

© 2000-2025, MetaQuotes Ltd.

IndicatorAdd

指定されたチャートウィンドウに指定されたハンドルと指標を追加します。

```
bool  IndicatorAdd(
   int    sub_win          // サブウィンドウ番号
   int    handle           // 指標ハンドル
   );
```

パラメータ

sub_win

[in] チャートサブウィンドウ番号（0はメインチャートウィンドウ）存在しないウィンドウの番号が指定された場合、新規のウィンドウが作成されます。

handle

[in] 指標ハンドル

戻り値

成功の場合はtrue、それ以外の場合はfalse。エラー情報を取得するには、GetLastError()関数が呼ばれます。

参照

IndicatorDelete()、IndicatorsTotal()、IndicatorName()

© 2000–2025, MetaQuotes Ltd.

IndicatorDelete

指定された名称の指標を指定されたチャートウィンドウから削除します。

```
bool  IndicatorDelete(
   int            sub_win        // サブウィンドウ番号
   const string   name           // 指標短縮名
   );
```

パラメータ

sub_win

[in] チャートサブウィンドウの番号。(0 はメインチャートウィンドウ)

const name

[in] IndicatorSetString() 関数で INDICATOR_SHORTNAMEプロパティに設定した指標の短縮名指標の短縮名を取得するには IndicatorName() 関数が呼ばれます。

戻り値

指標の削除が成功した場合は true、その他の場合は false。エラーの詳細を取得するには GetLastError() 関数を使用します。

注意事項

同一の短縮名を持つ指標がチャートサブウィンドウに 2 つ存在する場合、行の最初のものは削除されます。

このチャート上の他の指標が削除される指標の値に基づいている場合は、そのような指標も削除されます。

iCustom() 及び IndicatorCreate() 関数を使用して指標を作成する際、指標の短縮名とファイル名を混同しないでください。指標の短縮名が明示的に設定されていない場合、指標のソースコードを含むファイルの名称は、コンパイル時に指定されます。

指標のチャートからの削除は、その演算部が端末のメモリから削除されることを意味するものではありません。指標ハンドルを解除するには IndicatorRelease() 関数が使用されます。

指標の短縮名は正しく形成されなければなりません。短縮名は IndicatorSetString() 関数を使用して INDICATOR_SHORTNAMEプロパティに書かれます。IndicatorDelete() 関数によってチャートから削除される指標は短縮名によって識別されるので、短縮名に指標の全ての入力パラメータの値を含むようお勧めします。

参照

IndicatorAdd()、IndicatorsTotal()、IndicatorName()、iCustom()、IndicatorCreate()、IndicatorSetString()

© 2000–2025, MetaQuotes Ltd.

IndicatorsTotal

指定されたチャートウィンドウに適用された全ての指標の数を返します。

```
int  IndicatorsTotal(
   int   sub_win     // サブウィンドウ番号
   );
```

パラメータ

sub_win

[in] チャートサブウィンドウの番号。(0 はメインチャートウィンドウ)

戻り値

指定されたチャートウィンドウの指標の数。エラーの詳細を取得するにはGetLastError() 関数を使用します。

注意事項

この関数は、チャートに接続された全ての指標を検索することが出来ます。チャートのウィンドウの数の合計は GetInteger() 関数を使用して CHART_WINDOWS_TOTAL プロパティから得られます。

参照

IndicatorAdd()、IndicatorDelete()、IndicatorsTotal()、iCustom()、IndicatorCreate()、IndicatorSetString()

© 2000–2025, MetaQuotes Ltd.

IndicatorName

指定されたチャートウィンドウの指標リスト内の番号で指標の短縮名を返します。

```
string  IndicatorName(
   int    sub_win      // サブウィンドウ番号
   int    index        // チャートサブウィンドウに追加された指標のリスト内の指標のインデックス
   );
```

パラメータ

sub_win

[in] チャートサブウィンドウの番号。(0 はメインチャートウィンドウ)

index

[in] 指標のリスト内の指標のインデックスインデックスの命数は、ゼロで始まります。すなわち、リストの最初の指標が 0 のインデックスを持ちます。リスト内の指標の数を取得するには IndicatorsTotal() 関数が使用されます。

戻り値

IndicatorSetString() 関数でINDICATOR_SHORTNAME プロパティに設定した指標の短縮名。エラーの詳細を取得するには GetLastError() 関数を使用します。

注意事項

iCustom() 及び IndicatorCreate() 関数を使用して指標を作成する際、指標の短縮名とファイル名を混同しないでください。指標の短縮名が明示的に設定されていない場合、コンパイル時に指標のソースコードを含むファイルの名称が指定されます。

指標の短縮名は正しく形成されなければなりません。短縮名は IndicatorSetString() 関数を使用してINDICATOR_SHORTNAMEプロパティに書かれます。IndicatorDelete() 関数によってチャートから削除される指標は短縮名によって識別されるので、短縮名に指標の全ての入力パラメータの値を含むようお勧めします。

参照

IndicatorAdd()、IndicatorDelete、IndicatorsTotal、iCustom()、IndicatorCreate()、IndicatorSetString()

© 2000–2025, MetaQuotes Ltd.

Navigate

チャートをナビゲートします。

```
bool  Navigate(
   ENUM_CHART_POSITION  position,     // 位置
   int                  shift=0       // シフト
  )
```

パラメータ

position

[in] ENUM_CHART_POSITION 列挙の値

shift=0

[in] シフトするバーの数

戻り値

成功の場合は true、チャートがナビゲートされなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Symbol

チャート のシンボルを取 得します。

```
string  Symbol() const
```

戻 り値

クラスインスタンスに割 り当 てられたチャート のシンボル(割 り当 てられたチャート が存 在 しない場 合 は 0)

© 2000-2025, MetaQuotes Ltd.

Period

チャートの期間を取得します。

```
ENUM_TIMEFRAMES  Period() const
```

戻り値

クラスインスタンスに割り当てられたチャートの期間（割り当てられたチャートが存在しない場合は 0）

© 2000–2025, MetaQuotes Ltd.

Redraw

クラスインスタンスに割り当てたチャートを再描画します。

```
void  Redraw()
```

© 2000-2025, MetaQuotes Ltd.

GetInteger

この関数は、対応するオブジェクトプロパティの値を返します。オブジェクトプロパティは整数型でなければなりません。この関数には 2 つのバージョンがあります。

1. すぐにプロパティ値を返します。

```
long  GetInteger(
   ENUM_CHART_PROPERTY_INTEGER  prop_id,          // プロパティ識別子
   int                          sub_window=0      // サブウィンドウ番号
   ) const
```

2. 成功した場合、プロパティの値は、参照によって渡された最後のパラメータである指定された整数型の変数に配置されます。

```
bool  GetInteger(
   ENUM_CHART_PROPERTY_INTEGER  prop_id,          // プロパティ識別子
   int                          sub_window,       // サブウィンドウ番号
   long&                        value             // プロパティ値を取得する
   ) const
```

パラメータ

prop_id

[in] プロパティ識別子（ENUM_CHART_PROPERTY_INTEGER 列挙）

sub_window

[in] チャートサブウィンドウの番号

value

[in] リクエストされたプロパティの値を受け取った整数型の変数

戻り値

クラスインスタンスに割り当てられたチャートプロパティの値 If there isn't any chart assigned, it returns -1.

2 番目の呼び出しバージョン例では、指定されたプロパティが維持され、その値がvalue 変数に置かれている場合には true, それ以外の場合は falseエラーの詳細を読むには GetLastError() を呼びます。

© 2000-2025, MetaQuotes Ltd.

SetInteger

整数型のプロパティに新しい値を設定します。

```
bool  SetInteger(
   ENUM_CHART_PROPERTY_INTEGER  prop_id,      // プロパティ識別子
   long                         value         // 新しい値
  )
```

パラメータ

prop_id

[in] プロパティ識別子（ENUM_CHART_PROPERTY_INTEGER 列挙）

value

[in] プロパティの新しい値

戻り値

成功の場合は true、整数型プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

GetDouble

この関数は、対応するオブジェクトプロパティの値を返します。オブジェクトプロパティは double 型でなければなりません。この関数には 2 つのバージョンがあります。

1. すぐにプロパティ値を返します。

```
double  GetDouble(
   ENUM_CHART_PROPERTY_DOUBLE  prop_id,          // プロパティ識別子
   int                         sub_window=0     // サブウィンドウ番号
   ) const
```

2. 成功した場合、プロパティの値は、参照によって渡された最後のパラメータである指定された double 型の変数に配置されます。

```
bool  GetDouble(
   ENUM_CHART_PROPERTY_DOUBLE  prop_id,         // プロパティ識別子
   int                         sub_window,      // サブウィンドウ番号
   double&                     value            // プロパティ値を取得する
   ) const
```

パラメータ

prop_id

[in] プロパティ識別子（ENUM_CHART_PROPERTY_DOUBLE 列挙）

sub_window

[in] チャートサブウィンドウの番号

value

[in] リクエストされたプロパティの値を受け取ったdouble 型の変数

戻り値

クラスインスタンスに割り当てられたチャートプロパティの値If there isn't any chart assigned, it returns EMPTY_VALUE.

2 番目の呼び出しバージョン例では、指定されたプロパティが維持され、その値がvalue 変数に置かれている場合には true，それ以外の場合は falseエラーの詳細を読むには GetLastError() を呼びます。

© 2000–2025, MetaQuotes Ltd.

SetDouble

double 型のプロパティに新しい値を設定します。

```
bool  SetDouble(
   ENUM_CHART_PROPERTY_DOUBLE  prop_id,      // プロパティ識別子
   double                      value         // 新しい値
  )
```

パラメータ

prop_id

[in] プロパティ識別子（ENUM_CHART_PROPERTY_DOUBLE 列挙）

value

[in] プロパティの新しい値

戻り値

成功の場合は true、double 型プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

GetString

この関数は、対応するオブジェクトプロパティの値を返します。オブジェクトプロパティは string 型でなければなりません。この関数には 2 つのバージョンがあります。

1. すぐにプロパティ値を返します。

```
string  GetString(
   ENUM_CHART_PROPERTY_STRING  prop_id      // プロパティ識別子
   ) const
```

2. 成功した場合、プロパティの値は、参照によって渡された最後のパラメータである指定された string 型の変数に配置されます。

```
bool  GetString(
   ENUM_CHART_PROPERTY_STRING  prop_id,     // プロパティ識別子
   string&                     value        // プロパティ値を取得する
   ) const
```

パラメータ

prop_id

[in] プロパティ識別子（ENUM_CHART_PROPERTY_STRING 列挙）

sub_window

[in] チャートサブウィンドウの番号

value

[in] リクエストされたプロパティの値を受け取った string t型の変数

戻り値

クラスインスタンスに割り当てられたチャートプロパティの値（割り当てられたチャートが存在しない場合は "" ）

2 番目の呼び出しバージョン例では、指定されたプロパティが維持され、その値がvalue 変数に置かれている場合には true，それ以外の場合は falseエラーの詳細を読むには GetLastError() を呼びます。

© 2000–2025, MetaQuotes Ltd.

SetString

string 型のプロパティに新しい値を設定します。

```
bool  SetString(
   ENUM_CHART_PROPERTY_STRING  prop_id,      // プロパティ識別子
   string                      value         // 新しいプロパティ値
  )
```

パラメータ

prop_id

[in] プロパティ識別子（ENUM_CHART_PROPERTY_STRING 列挙）

value

[in] プロパティの新しい値

戻り値

成功の場合は true、string 型プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

SetSymbolPeriod

クラスインスタンスに割り当てられたチャートの銘柄と期間を変更します。

```
bool  SetSymbolPeriod(
   const string      symbol_name,      // シンボル
   ENUM_TIMEFRAMES   timeframe         // 期間
  )
```

パラメータ

symbol_name

[in] 新しい銘柄名 NULL はエキスパートアドバイザーがに接続されている現材のチャートのシンボルを意味します。

timeframe

[in] 新しいチャートの時間軸（ENUM_TIMEFRAMES 列挙）。0 は現在の時間軸を表します。

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

ApplyTemplate

指定されたテンプレートをチャートに適用します。

```
bool  ApplyTemplate(
   const string  filename      // テンプレートファイル名
  )
```

パラメータ

filename

[in] テンプレートファイル名

戻り値

成功の場合は true、テンプレートが適用されなかった場合は false

© 2000–2025, MetaQuotes Ltd.

ScreenShot

指定されたチャートのスクリーンショットを作成して .GIF ファイルに保存します。

```
bool  ScreenShot(
   string            filename,                  // ファイル名
   int               width,                     // 幅
   int               height,                    // 高さ
   ENUM_ALIGN_MODE   align_mode=ALIGN_RIGHT     // 整列の種類
   ) const
```

パラメータ

filename

[in] スクリーンショットファイル名

幅

[in] ピクセル単位でのスクリーンショットの幅

height

[in] ピクセル単位でのスクリーンショットの高さ

align_mode=ALIGN_RIGHT

[in] スクリーンショットの幅が狭い場合の整列の種類

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

WindowOnDropped

オブジェクト（エキスパートまたはスクリプト）ドロップポイントに対応するチャートサブウィンドウの番号を取得します。

```
int  WindowOnDropped() const
```

戻り値

オブジェクト（エキスパートまたはスクリプト）ドロップポイントに対応するチャートサブウィンドウの番号0　はメインチャートウィンドウを意味します。

© 2000–2025, MetaQuotes Ltd.

PriceOnDropped

オブジェクト（エキスパートまたはスクリプト）ドロップポイントに対応する価格座標を取得します。

```
double  PriceOnDropped() const
```

戻り値

オブジェクト（エキスパートまたはスクリプト）ドロップポイントに対応する価格座標

© 2000–2025, MetaQuotes Ltd.

TimeOnDropped

オブジェクト（エキスパートまたはスクリプト）ドロップポイントに対応する時間座標を取得します。

```
datetime  TimeOnDropped() const
```

戻り値

オブジェクトドロップポイントに対応する時間座標

© 2000–2025, MetaQuotes Ltd.

XOnDropped

オブジェクト（エキスパートまたはスクリプト）ドロップポイントに対応するX座標を取得します。

```
int  XOnDropped() const
```

戻り値

オブジェクトドロップポイントに対応するX座標

© 2000–2025, MetaQuotes Ltd.

YOnDropped

オブジェクト（エキスパートまたはスクリプト）ドロップポイントに対応するＹ座標を取得します。

```
int  YOnDropped() const
```

戻り値

オブジェクトドロップポイントに対応するＹ座標

© 2000–2025, MetaQuotes Ltd.

Save

オブジェクト パラメータをファイルに保 存します。

```
virtual bool  Save(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] FileOpen(...) 関 数 ですでに開 かれたバイナリファイルのハンド ル

戻 り値

成 功 の場 合 は true、エラーが発 生 した場 合 は false

© 2000–2025, MetaQuotes Ltd.

Load

オブジェクト パラメータをファイルから読み込みます。

```
virtual bool  Load(
   int  file_handle      // ファイルハンドル
  )
```

パラメータ

file_handle

[in] FileOpen(...) 関数ですでに開かれたバイナリファイルのハンドル

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

Type

グラフィックオブジェクトの型の識別子を取得します。

```
virtual int  Type() const
```

戻り値

グラフィックオブジェクトの型の識別子（0x1111 は CChart）

© 2000-2025, MetaQuotes Ltd.

グラフィックス

グラフィックスライブラリにはカスタムチャートを簡単にプロットするためのクラスとグローバル関数が含まれています。 ライブラリは、軸や曲線の作成のための便利な既成のソリューションとともに、カスタムチャートの共通プロパティの変更への迅速なアクセス方法を提供します。

ライブラリの使用をはじめるにあたって、可視化の可能性　Rのプロットに似たMQL5のグラフィックス　ライブラリ稿をお読みください。

グラフィックライブラリは、端末作業ディレクトリのInclude¥Graphicsフォルダに置かれます。

クラス	説明
CAxis	座標軸で作業するためのクラス
ColorGenerator	デフォルト配色を指定するクラス
CCurve	曲線で作業するためのクラス
CGraphic	カスタムチャートを作成するための基本クラス

© 2000–2025, MetaQuotes Ltd.

GraphPlot

手早く曲線をプロットするための関数。

Y座標を使用して1つの曲線をプロットするためのバージョン。

```
string  GraphPlot(
   const double      &y[],                  // Y座標
   ENUM_CURVE_TYPE   type=CURVE_POINTS      // 曲線の種類
   )
```

注

Y配列のインデックスがカーブのX座標として使用されます。

X及びY座標を使用して1つの曲線をプロットするためのバージョン。

```
string  GraphPlot(
   const double      &x[],                  // X座標
   const double      &y[],                  // Y座標
   ENUM_CURVE_TYPE   type=CURVE_POINTS      // 曲線の種類
   )
```

X及びY座標を使用して2つの曲線をプロットするためのバージョン。

```
string  GraphPlot(
   const double      &x1[],                 // X座標
   const double      &y1[],                 // Y座標
   const double      &x2[],                 // X座標
   const double      &y2[],                 // Y座標
   ENUM_CURVE_TYPE   type=CURVE_POINTS      // 曲線の種類
   )
```

X及びY座標を使用して3つの曲線をプロットするためのバージョン。

```
string  GraphPlot(
   const double      &x1[],                 // X座標
   const double      &y1[],                 // Y座標
   const double      &x2[],                 // X座標
   const double      &y2[],                 // Y座標
   const double      &x3[],                 // X座標
   const double      &y3[],                 // Y座標
   ENUM_CURVE_TYPE   type=CURVE_POINTS      // 曲線の種類
   )
```

© 2000–2025, MetaQuotes Ltd.

CPoint2D点の座標を使用して1つの曲線をプロットするためのバージョン。

```
string  GraphPlot(
   const CPoint2D   &points[],            // 曲線座標
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // 曲線の種類
   )
```

CPoint2D点の座標を使用して2つの曲線をプロットするためのバージョン。

```
string  GraphPlot(
   const CPoint2D   &points1[],           // 曲線座標
   const CPoint2D   &points2[],           // 曲線座標
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // 曲線の種類
   )
```

CPoint2D点の座標を使用して3つの曲線をプロットするためのバージョン。

```
string  GraphPlot(
   const CPoint2D   &points1[],           // 曲線座標
   const CPoint2D   &points2[],           // 曲線座標
   const CPoint2D   &points3[],           // 曲線座標
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // 曲線の種類
   )
```

CurveFunctionへのポインタを使用して1つの曲線をプロットするためのバージョン。

```
string  GraphPlot(
   CurveFunction    function,             // 関数ポインタ
   const double     from,                 // 引数の初期値
   const double     to,                   // 引数の最終値
   const double     step,                 // 引数による増分
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // 曲線の種類
   )
```

CurveFunctionへのポインタを使用して2つの曲線をプロットするためのバージョン。

```
string  GraphPlot(
   CurveFunction    function1,            // 関数ポインタ
   CurveFunction    function2,            // 関数ポインタ
   const double     from,                 // 引数の初期値
   const double     to,                   // 引数の最終値
   const double     step,                 // 引数による増分
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // 曲線の種類
   )
```

© 2000–2025, MetaQuotes Ltd.

CurveFunctionへのポインタを使用して3つの曲線をプロットするためのバージョン。

```
string  GraphPlot(
   CurveFunction     function1,              // 関数ポインタ
   CurveFunction     function2,              // 関数ポインタ
   CurveFunction     function3,              // 関数ポインタ
   const double      from,                   // 引数の初期値
   const double      to,                     // 引数の最終値
   const double      step,                   // 引数による増分
   ENUM_CURVE_TYPE   type=CURVE_POINTS       // 曲線の種類
   )
```

パラメータ

&x[]
  [in] X座標。

&y[]
  [in] Y座標。

&x1[]
  [in] 1番目の曲線のX座標。

&y1[]
  [in] 1番目の曲線のY座標。

&x2[]
  [in] 2番目の曲線のX座標。

&y2[]
  [in] 2番目の曲線のY座標。

&x3[]
  [in] 3番目の曲線のX座標。

&y3[]
  [in] 3番目の曲線のY座標。

&points[]
  [in] 曲線の点の座標。

&points1[]
  [in] 1番目の曲線の点の座標。

&points2[]
  [in] 2番目の曲線の点の座標。

&points3[]
  [in] 3番目の曲線の点の座標。

© 2000-2025, MetaQuotes Ltd.

function

[in] CurveFunction関数へのポインタ。

function1

[in] 1番目の関数へのポインタ。

function2

[in] 2番目の関数へのポインタ。

function3

[in] 3番目の関数へのポインタ。

from

[in] 1番目のX座標に対応します。

to

[in] 最後のX座標に対応します。

step

[in] X座標を計算するためのパラメータ。

type=CURVE_POINTS

[in] 曲線の種類。

戻り値

グラフィカルリソース名。

© 2000-2025, MetaQuotes Ltd.

CAxis

CAxisは、座標軸を操作するための補助的なグラフィックスライブラリクラスです。.

説明

CAxisクラスは、座標軸のさまざまなパラメータを受け取って格納します。座標軸を動的に自動スケーリングする機能も実装されています。

宣言

```
class CAxis
```

タイトル

```
#include <Graphics\Axis.mqh>
```

クラスメソッド

メソッド	説明
AutoScale	自動スケーリングフラグを取得/設定します。
Min	最小軸値を取得/設定します。
Max	最大軸値を取得/設定します。
Step	軸によるステップ値を取得します。
Name	軸名を取得/設定します。
Color	軸の色を取得/設定します。
ValuesSize	軸番号のサイズを取得/設定します。
ValuesWidth	表示される軸番号の最大長を取得/設定します
ValuesFormat	軸番号のフォーマットを取得/設定します
ValuesDateTimeMode	日付を文字列に変換する形式を取得します。
ValuesFunctionFormat	軸上に値を表示するフォーマットを定義する関数へのポインタを取得します。
ValuesFunctionFormatCBData	軸値の変換に関する追加データを含むことがあるオブジェクトへのポインタを取得します。
NameSize	軸名のフォントサイズを取得/設定します。
ZeroLever	「ゼロ・レバー」値を取得/設定します。
DefaultStep	軸による初期ステップ値を取得/設定します。
MaxLabels	軸上の最大数を取得/設定します。
MinGrace	軸の最小値の「許容値」を取得/設定します。

© 2000–2025, MetaQuotes Ltd.

メソッド	説明
MaxGrace	軸の最大値の「許容値」を取得/設定します。
SelectAxisScale	軸を自動スケーリングします。

© 2000–2025, MetaQuotes Ltd.

AutoScale（Getメソッド）

自動スケールの必要性を定義するフラグを返します。

```
bool  AutoScale()
```

戻り値

フラグ値。

注

true — 自動スケールがオン。

false — 自動スケールがオフ。

AutoScale（Set メソッド）

自動スケールの必要性を定義するフラグを設定します。

```
void  AutoScale(
   const bool  auto      // フラグ値
   )
```

パラメータ

auto

[in]

注

true — 自動スケールがオン。

false — 自動スケールがオフ。

© 2000–2025, MetaQuotes Ltd.

Min（Getメソッド）

最小軸値を返します。

```
double  Min()
```

戻り値

最小軸値。

Min（Setメソッド）

最小軸値を設定します。

```
void  Min(
   const double  min       // 最小値
   )
```

パラメータ

min

[in] 最小値。

© 2000–2025, MetaQuotes Ltd.

Max（Getメソッド）

最大軸値を返します。

```
double  Max()
```

戻り値

最大軸値。

Max（Setメソッド）

最大軸値を設定します。

```
void  Max(
   const double  max      // 最大値
   )
```

パラメータ

max

[in] 最大軸値。

© 2000–2025, MetaQuotes Ltd.

Step（Getメソッド）

軸によるステップ値を返します。

```
double  Step()
```

戻り値

ステップ値。

© 2000–2025, MetaQuotes Ltd.

Name（Getメソッド）

軸の名前を返します。

```
string  Name()
```

戻り値

軸の名前。

Name（Setメソッド）

軸の名前を設定します。

```
void  Name(
   const string  name      // 軸の名前
   )
```

パラメータ

name

[in] 軸の名前。

© 2000–2025, MetaQuotes Ltd.

Color（Getメソッド）

軸の色を返します。

```
color  Color()
```

戻り値

軸の色。

Color（Set メソッド）

軸の色を設定します。

```
void  Color(
   const color  clr      // 軸の色
   )
```

パラメータ

clr

[in] 軸の色。

© 2000–2025, MetaQuotes Ltd.

ValuesSize（Getメソッド）

軸番号のサイズを返します。

```
int  ValuesSize()
```

戻り値

軸番号のサイズ。

ValuesSize（Setメソッド）

軸番号のサイズを設定します。

```
void  ValuesSize(
   const int  size      // 軸番号のサイズ
   )
```

パラメータ

size

[in] 軸番号のサイズ。

© 2000-2025, MetaQuotes Ltd.

ValuesWidth（Getメソッド）

軸番号を表示するために許容される最大長をピクセル単位で返します。

```
int  ValuesWidth()
```

戻り値

ピクセル単位での軸番号の長さ。

注

指定された数のピクセル単位の長さが許容される最大表示長を超えると、切り捨てられ、点で終わります。

ValuesWidth（Setメソッド）

軸番号を表示するために許容される最大長をピクセル単位で設定します。

```
void  ValuesWidth(
   const int  width      // ピクセル単位での最大許容長
   )
```

パラメータ

width

[in] 軸番号の最大許容の長さ。

注

指定された数のピクセル単位の長さが許容される最大表示長を超えると、切り捨てられ、点で終わります。

© 2000–2025, MetaQuotes Ltd.

ValuesFormat（Getメソッド）

軸番号の形式を返します。

```
string  ValuesFormat()
```

戻り値

軸番号の形式。

ValuesFormat（Setメソッド）

軸番号の形式を設定します。

```
void  ValuesFormat(
   const string  format      // 軸番号の形式
   )
```

パラメータ

format

[in] 軸番号の形式。

© 2000-2025, MetaQuotes Ltd.

ValuesDateTimeMode（取得メソッド）

日付を文字列に変換する形式を取得します。

```
int  ValuesDateTimeMode()
```

戻り値

日付を文字列に変換する形式

ValuesDateTimeMode（設定メソッド）

日付を文字列に変換する形式を設定します。

```
void  ValuesDateTimeMode(
   const int  mode      // 日付を文字列に変換する形式
  )
```

パラメータ

mode

[in] 変換形式

注意事項

日付を文字列に変換する形式の詳細はTimeToString()関数の記述でご覧ください。

© 2000-2025, MetaQuotes Ltd.

ValuesFunctionFormat（取得メソッド）

軸上に値を表示するフォーマットを定義する関数へのポインタを取得します。

```
DoubleToStringFunction  ValuesFunctionFormat()
```

戻り値

軸上に値を表示するフォーマットを定義する関数へのポインタ

ValuesFunctionFormat（設定メソッド）

軸上に値を表示するフォーマットを定義する関数へのポインタを設定します。

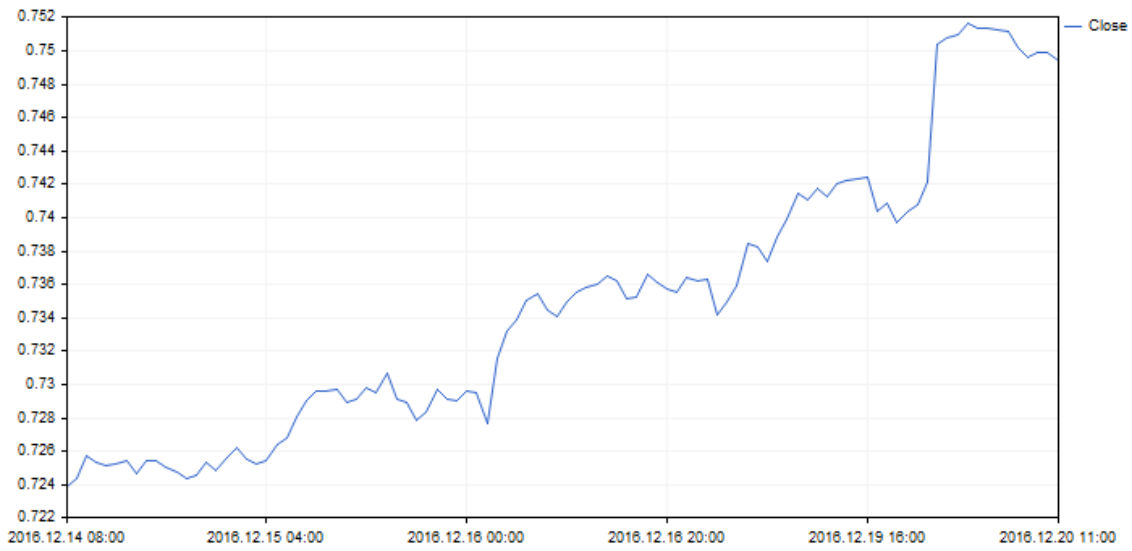
```
void  ValuesFunctionFormat(
   DoubleToStringFunction  func      // 数値を文字列に変換する関数
  )
```

パラメータ

func

[in] 数値を文字列に変換するカスタム関数

例：



X軸の値を表示する形式は次のコードを使用して変更されました。

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                DateAxisGraphic.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Graphics\Graphic.mqh>
//--- array for store values
double arrX[];
double arrY[];
//+------------------------------------------------------------------+
//| X軸に値を作成するカスタム関数                                    |
//+------------------------------------------------------------------+
string TimeFormat(double x,void *cbdata)
  {
   return(TimeToString((datetime)arrX[ArraySize(arrX)-(int)x-1]));
  }
//+------------------------------------------------------------------+
void OnStart()
  {
   MqlRates rates[];
   CopyRates(Symbol(),Period(),0,100,rates);
   ArraySetAsSeries(rates,true);
   int size=ArraySize(rates);
   ArrayResize(arrX,size);
   ArrayResize(arrY,size);
   for(int i=0; i<size;++i)
     {
      arrX[i]=(double)rates[i].time;
      arrY[i]=rates[i].close;
     }
//--- グラフィックを作成する
   CGraphic graphic;
   if(!graphic.Create(0,"DateAxisGraphic",0,30,30,780,380))
     {
      graphic.Attach(0,"DateAxisGraphic");
     }
//--- 曲線を作成する
   CCurve *curve=graphic.CurveAdd(arrY,CURVE_LINES);
//--- X軸を取得する
   CAxis *xAxis=graphic.XAxis();
//--- X軸プロパティを設定する
   xAxis.AutoScale(false);
   xAxis.Type(AXIS_TYPE_CUSTOM);
   xAxis.ValuesFunctionFormat(TimeFormat);
   xAxis.DefaultStep(20.0);
//--- プロット
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

© 2000–2025, MetaQuotes Ltd.

ValuesFunctionFormatCBData（取得メソッド）

軸値の変換に関する追加データを含むことがあるオブジェクトへのポインタを取得します。

```
void*  ValuesFunctionFormatCBData()
```

戻り値

軸値の変換に関する追加データを含むことがあるオブジェクトへのポインタ

ValuesFunctionFormatCBData（設定メソッド）

軸値の変換に関する追加データを含むことがあるオブジェクトへのポインタを設定します。

```
void  ValuesFunctionFormatCBData(
   void*  cbdata      // クラスオブジェクトへのポインタ
  )
```

パラメータ

cbdata

[in] 軸値の変換に関する追加データを含むことがあるオブジェクトへのポインタ

© 2000–2025, MetaQuotes Ltd.

NameSize（Getメソッド）

軸の名前のフォントサイズを返します。

```
int  NameSize()
```

戻り値

軸の名前のフォントサイズ。

NameSize（Setメソッド）

軸の名前のフォントサイズを設定します。

```
void  NameSize(
   const int  size      // 軸の名前のフォントサイズ
   )
```

パラメータ

size

[in] 軸の名前のフォントサイズ。

© 2000-2025, MetaQuotes Ltd.

ZeroLever（Getメソッド）

「ゼロ・レバー」値を戻します。

```
double  ZeroLever()
```

戻り値

「ゼロ・レバー」。

注

この値は、軸スケール範囲がいつゼロ値を含むように拡大されるべきかを定義するために使用されます。

ZeroLever（Setメソッド）

「ゼロ・レバー」値を設定します。

```
void  ZeroLever(
   const double  value      // 「ゼロ・レバー」値
   )
```

パラメータ

value

[in] 「ゼロ・レバー」値。

注

この値は、軸スケール範囲がいつゼロ値を含むように拡大されるべきかを定義するために使用されます。

© 2000–2025, MetaQuotes Ltd.

DefaultStep（Getメソッド）

軸による初期ステップ値を返します。

```
double  DefaultStep()
```

戻り値

軸によるステップ。

DefaultStep（Set メソッド）

軸による初期ステップ値を設定します。

```
void  DefaultStep(
   const double  value      // 軸によるステップ
   )
```

パラメータ

value

[in] 軸による初期ステップ値。

© 2000-2025, MetaQuotes Ltd.

MaxLabels（Getメソッド）

軸に表示される最大許容数値を返します。

```
double  MaxLabels()
```

戻り値

軸上の最大数値。

MaxLabels（Setメソッド）

軸に表示される最大許容数値を設定します。

```
void  MaxLabels(
   const double  value      // 最大値
   )
```

パラメータ

value

[in] 軸に表示される最大許容数。

© 2000-2025, MetaQuotes Ltd.

MinGrace（Getメソッド）

最小軸値に適応された「許容値」を返します。

```
double  MinGrace()
```

戻り値

最小軸値に適応された「許容」値。

注

この値は軸方向全長の一部として表されます。たとえば、軸の値が4.0～16.0の範囲内にあり、長さが12.0であるとします。MinGraceが0.1に等しい場合、軸長の10%（または1.2）が最小値から減算されます。その結果、軸は2.8から16.0までの区間をカバーします。

MinGrace（Setメソッド）

最小軸値に適応された「許容値」を設定します。

```
void  MinGrace(
   const double  value      // 「許容」値
   )
```

パラメータ

value

[in] 最小軸値に適応された「許容値」。

注

この値は軸方向全長の一部として表されます。たとえば、軸の値が4.0～16.0の範囲内にあり、長さが12.0であるとします。MinGraceが0.1に等しい場合、軸長の10%（または1.2）が最小値から減算されます。その結果、軸は2.8から16.0までの区間をカバーします。

© 2000-2025, MetaQuotes Ltd.

MaxGrace（Getメソッド）

最大軸値の「許容値」を返します。

```
double  MaxGrace()
```

戻り値

最大軸値の「許容値」。

注

この値は軸方向全長の一部として表されます。たとえば、軸の値が4.0～16.0の範囲内にあり、長さが12.0であるとします。MaxGraceが0.1に等しい場合、軸長の10%（または1.2）が最大値に加算されます。その結果、軸は4.0から17.2までの区間をカバーします。

MaxGrace（Setメソッド）

最大軸値の「許容値」を設定します。

```
void  MaxGrace(
   const double  value      // 「許容」値
   )
```

パラメータ

value

[in] 最大軸値に適応される「許容」値。

注

この値は軸方向全長の一部として表されます。たとえば、軸の値が4.0～16.0の範囲内にあり、長さが12.0であるとします。MinGraceが0.1に等しい場合、軸長の10%（または1.2）が最小値から減算されます。その結果、軸は2.8から16.0までの区間をカバーします。

© 2000-2025, MetaQuotes Ltd.

SelectAxisScale

軸を自動スケーリングします。

```
void  SelectAxisScale()
```

© 2000–2025, MetaQuotes Ltd.

CColorGenerator

CColorGeneratorは、カラーパレット作業するための補助的なグラフィックスライブラリクラスです。

説明

CColorGeneratorクラスには、デフォルトでカーブに使用される初期のカラーパレットが含まれています（ユーザーがカラーを指定していない場合）。

初期のパレットのすべての色が既に使用されている場合は、新しい色が自動的に生成され、パレットが補充されます。

宣言

```
class CColorGenerator
```

タイトル

```
#include <Graphics\ColorGenerator.mqh>
```

クラスメソッド

メソッド	説明
Next	パレットの次の色を返します。
Reset	生成器をリセットします。

© 2000–2025, MetaQuotes Ltd.

Next

パレットの次の色を返します。

```
uint  Next()
```

戻り値

色。

注

初期のパレットのすべての色が既に使用されている場合は、新しい色が自動的に生成され、パレットが補充されます。

© 2000–2025, MetaQuotes Ltd.

Reset

生成器をリセットします。

```
void  Reset()
```

© 2000–2025, MetaQuotes Ltd.

CCurve

CCurveクラスは、チャート上に生成される曲線のプロパティを使用するためのものです。

説明

CCurveクラスは、CGraphicクラスで作業するときに曲線の座標とさまざまなプロパティを設定、インストール、受信します。

曲線描画モードには、ドット、ライン、ヒストグラムの3つストレージクラスは 3 つあります。クラスの各プロットモードには個別のパラメータが実装されています。

宣言

```
class CCurve : public CObject
```

タイトル

```
#include <Graphics\Curve.mqh>
```

継承階層

CObject
  CCurve

クラスメソッド

メソッド	説明
Type	曲線タイプを取得します。
Name	曲線名を取得します。
Color	曲線の色を取得します。
XMax	X関数の最大値を取得します。
XMin	X関数の最小値を取得します。
YMax	Y関数の最大値を取得します。
YMin	Y関数の最小値を取得します。
Size	曲線を定義する点の数を取得します。
PointsSize	曲線を定義する点の線形サイズを取得/設定します。
PointsFill	曲線を定義する点を塗りつぶすためのフラグを取得/設定します。
PointsColor	点の塗りつぶしの色を取得/設定します。
GetX	曲線のすべての点のX値を配列に取得します。
GetY	曲線のすべての点のY値を配列に取得します。

© 2000–2025, MetaQuotes Ltd.

メソッド	説明
LinesStyle	曲線をプロットする線のスタイルを取得/設定します。
LinesIsSmooth	線を使用して描画するときのスムージングフラグを取得/設定します。
LinesSmoothTension	線を使用して描画するときのスムージングパラメータを取得/設定します。
LinesSmoothStep	線でプロットするときにスムージングする近似線の長さを取得/設定します。
LinesWidth	曲線をプロットする線の幅を取得/設定します。
HistogramWidth	ヒストグラムを使用してプロットするときの線の幅を取得/設定します。
CustomPlotCBData	カスタム曲線プロットモードで使用するオブジェクトへのポインタを取得/設定します。
CustomPlotFunction	カスタム曲線プロットモードを実装する関数へのポインタを取得/設定します。
PointsType	点線で曲線を描くときに使用する点の種類を示すフラグを取得/設定します。
StepsDimension	ステップ型曲線レンダリングで使用されるディメンションを示す値を取得/設定します。
TrendLineCoefficients	トレンドラインを配列に書き込むための比率を取得/設定します。
TrendLineColor	曲線のトレンドラインの色を取得/設定します。
TrendLineVisible	トレンドラインの可視性フラグを取得/設定します。
Update	曲線の座標を更新します。
Visible	チャート上に関数を表示するかどうかを定義するフラグを取得/設定します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Compare

© 2000-2025, MetaQuotes Ltd.

Type

曲線の種類を返します。

```
ENUM_CURVE_TYPE   Type()
```

戻り値

曲線の種類。

© 2000–2025, MetaQuotes Ltd.

Name

曲線の名前を返します。

```
string  Name()
```

戻り値

曲線の名前。

© 2000–2025, MetaQuotes Ltd.

Color

曲線の色を返します。

```
uint  Color()
```

戻り値

曲線の色。

© 2000–2025, MetaQuotes Ltd.

XMax

X関数の最大値を返します（実数のみ）。

```
double  XMax()
```

戻り値

すべての関数引数の中で最大の実数。

© 2000–2025, MetaQuotes Ltd.

XMin

X関数の最小値を返します（実数のみ）。

```
double  XMin()
```

戻り値

すべての関数引数の中で最小の実数。

© 2000–2025, MetaQuotes Ltd.

YMax

Y関数の最大値を返します（実数のみ）。

```
double  YMax()
```

戻り値

Y関数の最大値（実数のみ）。

© 2000–2025, MetaQuotes Ltd.

YMin

Y関数の最小値を返します（実数のみ）。

```
double  YMin()
```

戻り値

Y関数の最小値（実数のみ）。

© 2000–2025, MetaQuotes Ltd.

Size

曲線を定義する点の数を返します。

```
int  Size()
```

戻り値

曲線を定義する点の数。

© 2000–2025, MetaQuotes Ltd.

PointsSize（Getメソッド）

曲線を定義する点の線形サイズをピクセル単位で返します。

```
int  PointsSize()
```

戻り値

曲線を定義する点のピクセル単位でのサイズ。

PointsSize（Setメソッド）

曲線を定義する点の線形サイズをピクセル単位で設定します。

```
void  PointsSize(
   const int  size      // ピクセル単位での点のサイズ
   )
```

パラメータ

size

[in] 曲線を定義する点の線形のピクセル単位でのサイズ。

© 2000-2025, MetaQuotes Ltd.

PointsFill（Getメソッド）

曲線を定義する点を塗りつぶすためのフラグを返します。

```
bool  PointsFill ()
```

戻り値

フラグ値。

注

true — 塗りつぶす。

false — 塗りつぶさない。

PointsFill（Setメソッド）

曲線を定義する点を塗りつぶすためのフラグを設定します。

```
void  PointsFill(
   const bool  fill      // フラグ値
   )
```

パラメータ

fill

[in] フラグ値。

注

true — 塗りつぶす。

false — 塗りつぶさない。

© 2000–2025, MetaQuotes Ltd.

PointsColor（Getメソッド）

点を塗利つぶすための色を返します。

```
uint  PointsColor ()
```

戻り値

曲線を定義する点を塗りつぶすための色。

PointsColor（Setメソッド）

点を塗りつぶすための色を設定します。

```
void  PointsColor(
   const uint  clr      // 点を塗りつぶすための色
   )
```

パラメータ

clr

[in] 曲線を定義する点を塗りつぶすための色。

© 2000-2025, MetaQuotes Ltd.

GetX

すべての曲線の点のX座標を配列に書き込みます。

```
void  GetX(
   double  &x[]      // X座標
   )
```

パラメータ

&x[]

[out] X座標を書き入れるための配列。

© 2000–2025, MetaQuotes Ltd.

GetY

すべての曲線の点のY座標を配列に書き込みます。

```
void  GetY(
   double  &y[]      // Y座標
   )
```

パラメータ

&y[]

[out] Y座標を書き入れるための配列。

© 2000-2025, MetaQuotes Ltd.

LinesStyle（Getメソッド）

線を使用して曲線をプロットするときの線のスタイルを返します。

```
ENUM_LINE_STYLE  LinesStyle()
```

戻り値

線のスタイル。.

LinesStyle（Setメソッド）

線を使用して曲線をプロットするときの線のスタイルを設定します。

```
void  LinesStyle (
   ENUM_LINE_STYLE  style      // 線のスタイル
   )
```

パラメータ

style

[in] 線のスタイル。

© 2000-2025, MetaQuotes Ltd.

LinesIsSmooth（Getメソッド）

曲線を線でプロットするときにスムージングを実行するかどうかを定義するフラグを返します。

```
bool  LinesIsSmooth()
```

戻り値

フラグ値。

注

true — スムージングを行う。

false — スムージングを行わない。

LinesIsSmooth（Setメソッド）

曲線を線でプロットするときにスムージングを実行するかどうかを定義するフラグを設定します。

```
void  LinesIsSmooth(
   const bool  smooth      // フラグ値
   )
```

パラメータ

smooth

[in] フラグ値。

注

true — スムージングを行う。

false — スムージングを行わない。

© 2000–2025, MetaQuotes Ltd.

LinesSmoothTension（Getメソッド）

線を使用して描画するときの曲線スムージングパラメータを返します。

```
double  LinesSmoothTension()
```

戻り値

スムージングパラメータ値。

注

「tension（張力）」値は（0.0; 1.0）の範囲内にあります。

LinesSmoothTension（Setメソッド）

線を使用して描画するときの曲線スムージングパラメータを設定します。

```
void  LinesSmoothTension(
   const double  tension      // パラメータ値
   )
```

パラメータ

tension

[in] スムージングパラメータの値。.

注

「tension（張力）」値は（0.0; 1.0）の範囲内にあります。

© 2000-2025, MetaQuotes Ltd.

LinesSmoothStep（Getメソッド）

線でプロットするときのスムージングの近似線の長さを返します。

```
double  LinesSmoothStep()
```

戻り値

ピクセル単位での近似線の長さ。

LinesSmoothStep（Setメソッド）

線でプロットするときのスムージングの近似線の長さを設定します。

```
void  LinesSmoothStep(
   const double  step      // 線の長さ
   )
```

パラメータ

step

[in] 近似線の長さ。

© 2000-2025, MetaQuotes Ltd.

LinesEndStyle（設定メソッド）

線を使用して曲線をプロットする際に線の端をプロットするスタイルを示すフラグを取得します。

```
ENUM_LINE_END  LinesEndStyle()
```

戻り値

線を使用して曲線をプロットする際に線の端をプロットするスタイルを示すフラグの値

LinesEndStyle（取得メソッド）

線を使用して曲線をプロットする際に線の端をプロットするスタイルを示すフラグの値を設定します。

```
void  LinesEndStyle(
   ENUM_LINE_END  end_style      // フラグ値
  )
```

パラメータ

end_style

[in] 線を使用して曲線をプロットする際に線の端をプロットするスタイルを示すフラグの値

© 2000-2025, MetaQuotes Ltd.

LinesWidth（取得メソッド）

曲線をプロットする線の幅を取得します。

```
int  LinesWidth()
```

戻り値

線の幅

LinesWidth（設定メソッド）

曲線をプロットする線の幅を設定します。

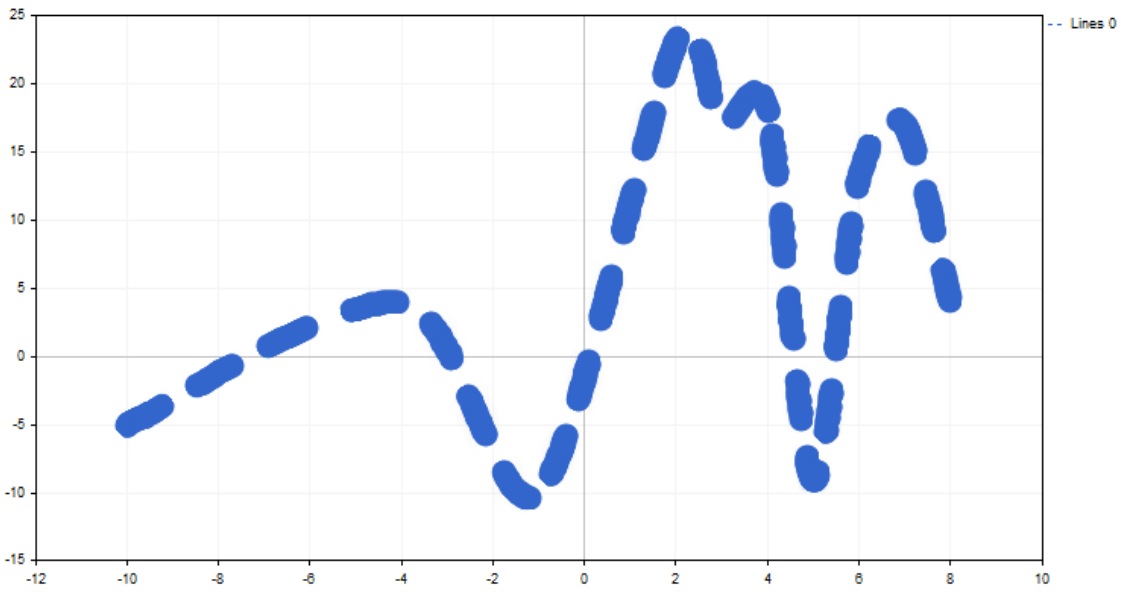
```
void  LinesWidth(
   const int  width      // 線の幅
  )
```

パラメータ

幅

[in] 曲線をプロットする線の幅

例：



線幅は次のコードを使用して変更されました。

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                CandleGraphic.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Graphics\Graphic.mqh>
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   double x[]= { -100,-40,-10,20,30,40,50,60,70,80,120 };
   double y[]= { -5,4,-10,23,17,18,-9,13,17,4,9 };
//--- グラフィックを作成する
   CGraphic graphic;
   if(!graphic.Create(0,"ThickLineGraphic",0,30,30,780,380))
     {
      graphic.Attach(0,"ThickLineGraphic");
     }
//--- 曲線を作成する
   CCurve *curve=graphic.CurveAdd(x,y,CURVE_LINES);
//--- 曲線のプロパティを設定する
   curve.LinesSmooth(true);
   curve.LinesStyle(STYLE_DASH);
   curve.LinesEndStyle(LINE_END_ROUND);
   curve.LinesWidth(10);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

© 2000–2025, MetaQuotes Ltd.

HistogramWidth（Getメソッド）

ヒストグラムを使用してプロットするときの列の幅を返します。

```
int  HistogramWidth()
```

戻り値

ピクセル単位での列の幅。

HistogramWidth（Setメソッド）

ヒストグラムを使用してプロットするときの列の幅を設定します。

```
void  HistogramWidth(
   const int  width      // 列の幅
   )
```

パラメータ

width

[in] ピクセル単位での列の幅。

© 2000–2025, MetaQuotes Ltd.

CustomPlotCBData（取得メソッド）

カスタム曲線プロットモードで使用するオブジェクトへのポインタを取得します。

```
void*  CustomPlotCBData()
```

戻り値

カスタム曲線プロットモードで使用するオブジェクトへのポインタ

CustomPlotCBData（設定メソッド）

カスタム曲線プロットモードで使用するオブジェクトへのポインタを設定します。

```
void  CustomPlotCBData(
   void*  cbdata      // オブジェクトへのポインタ
  )
```

パラメータ

cbdata

[in] カスタム曲線プロットモードで使用するオブジェクトへのポインタ

© 2000-2025, MetaQuotes Ltd.

CustomPlotFunction（取得メソッド）

カスタム曲線プロットモードを実装する関数へのポインタを取得します。

```
PlotFucntion  CustomPlotFunction()
```

戻り値

カスタム曲線プロットモードを実装する関数へのポインタ

CustomPlotFunction（設定メソッド）

カスタム曲線プロットモードを実装する関数へのポインタを設定します。

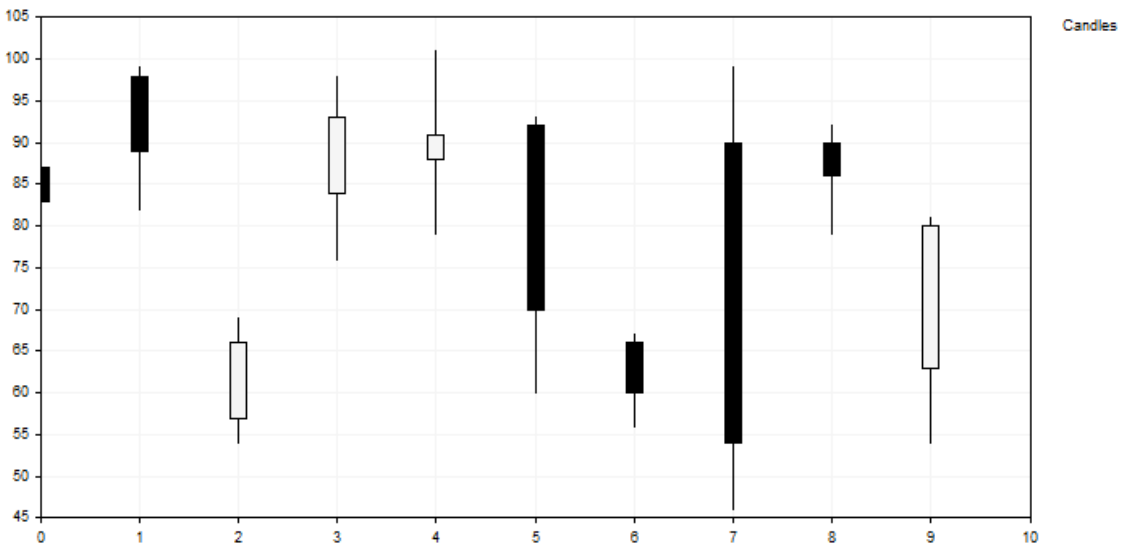
```
void  CustomPlotFunction(
   PlotFucntion  func      // 関数へのポインタ
  )
```

パラメータ

func

[in] カスタム曲線プロットモードを実装する関数へのポインタ

例：



バーで構成されるこの曲線は、次のコードを使用して作成されます。

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                CandleGraphic.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Graphics\Graphic.mqh>
//+------------------------------------------------------------------+
//| Class CCandle                                                    |
//| 使用法:ローソク足を表すクラス                                    |
//+------------------------------------------------------------------+
class CCandle: public CObject
  {
private:
   double            m_open;
   double            m_close;
   double            m_high;
   double            m_low;
   uint              m_clr_inc;
   uint              m_clr_dec;
   int               m_width;

public:
                     CCandle(const double open,const double close,const double
high,const double low,
                                                      const int width,const uint
clr_inc=0x000000,const uint clr_dec=0xF5F5F5);
                    ~CCandle(void);
   double            OpenValue(void)             const { return(m_open);      }
   double            CloseValue(void)            const { return(m_close);     }
   double            HigthValue(void)            const { return(m_high);      }
   double            LowValue(void)              const { return(m_low);       }
   uint              CandleColorIncrement(void) const { return(m_clr_inc);   }
   uint              CandleColorDecrement(void) const { return(m_clr_dec);   }
   int               CandleWidth(void)           const { return(m_width);     }
  };
//+------------------------------------------------------------------+
//| コンストラクタ                                                   |
//+------------------------------------------------------------------+
CCandle::CCandle(const double open,const double close,const double high,const double
low,
                                 const int width,const uint clr_inc=0x000000,const
uint clr_dec=0xF5F5F5):
                                 m_open(open),m_close(close),m_high(high),m_low(low)
,
                                 m_clr_inc(clr_inc),m_clr_dec(clr_dec),m_width(width
)
  {
  }
//+------------------------------------------------------------------+
//| デストラクタ                                                     |
//+------------------------------------------------------------------+
CCandle::~CCandle(void)
  {
  }
//+------------------------------------------------------------------+
//| Custom method for plot candles                                   |
//+------------------------------------------------------------------+
void PlotCandles(double &x[],double &y[],int size,CGraphic *graphic,CCanvas
*canvas,void *cbdata)
  {
//--- check obj
```

© 2000–2025, MetaQuotes Ltd.

```
   CArrayObj *candles=dynamic_cast<CArrayObj*>(cbdata);
   if(candles==NULL || candles.Total()!=size)
      return;
//--- plot candles
   for(int i=0; i<size; i++)
     {
      CCandle *candle=dynamic_cast<CCandle*>(candles.At(i));
      if(candle==NULL)
         return;
      //--- primary calculate
      int xc=graphic.ScaleX(x[i]);
      int width_2=candle.CandleWidth()/2;
      int open=graphic.ScaleY(candle.OpenValue());
      int close=graphic.ScaleY(candle.CloseValue());
      int high=graphic.ScaleY(candle.HigthValue());
      int low=graphic.ScaleY(candle.LowValue());
      uint clr=(open<=close) ?candle.CandleColorIncrement() :
candle.CandleColorDecrement();
      //--- ローソク足をプロットする
      canvas.LineVertical(xc,high,low,0x000000);
      //--- ローソクの実体をプロットする
      canvas.FillRectangle(xc+width_2,open,xc-width_2,close,clr);
      canvas.Rectangle(xc+width_2,open,xc-width_2,close,0x000000);
     }
  }
//+------------------------------------------------------------------+
//| スクリプトプログラム開始関数                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
   int count=10;
   int width=10;
   double x[];
   double y[];
   ArrayResize(x,count);
   ArrayResize(y,count);
   CArrayObj candles();
   double max=0;
   double min=0;
//--- 値を作成する
   for(int i=0; i<count; i++)
     {
      x[i] = i;
      y[i] = i;
      //--- 値を計算する
      double open=MathRound(50.0+(MathRand()/32767.0)*50.0);
      double close=MathRound(50.0+(MathRand()/32767.0)*50.0);
      double high=MathRound(MathMax(open,close)+(MathRand()/32767.0)*10.0);
      double low=MathRound(MathMin(open,close) -(MathRand()/32767.0)*10.0);
      //--- 最高値と最低値を見つける
      if(i==0 || max<high)
         max=high;
      if(i==0 || min>low)
         min=low;
      //--- ローソク足を作成する
      CCandle *candle=new CCandle(open,close,high,low,width);
      candles.Add(candle);
     }
//--- グラフィックを作成する
   CGraphic graphic;
   if(!graphic.Create(0,"CandleGraphic",0,30,30,780,380))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       graphic.Attach(0,"CandleGraphic");
      }
//--- 曲線を作成する
   CCurve *curve=graphic.CurveAdd(x,y,CURVE_CUSTOM,"Candles");
//--- 曲線のプロパティを設定する
   curve.CustomPlotFunction(PlotCandles);
   curve.CustomPlotCBData(GetPointer(candles));
//--- グラフィックのプロパティを設定する
   graphic.YAxis().Max((int)max);
   graphic.YAxis().Min((int)min);
//--- プロット
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

© 2000–2025, MetaQuotes Ltd.

PointsType（取得メソッド）

点線で曲線を描くときに使用する点の種類を示すフラグを取得します。

```
ENUM_POINT_TYPE  PointsType()
```

戻り値

点の種類を示すフラグの値

PointsType（設定メソッド）

点線で曲線を描くときに使用する点の種類を示すフラグを設定します。

```
void  PointsType(
   ENUM_POINT_TYPE  type      // フラグ値
  )
```

パラメータ

型

[in] 点線で曲線を描くときに使用する点の種類を示すフラグ

© 2000-2025, MetaQuotes Ltd.

StepsDimension（取得メソッド）

ステップ型曲線レンダリングで使用されるディメンションを示す値を取得します。

```
int  StepsDimension()
```

戻り値

ステップ型曲線レンダリングで使用されるディメンション

StepsDimension（設定メソッド）

ステップ型曲線レンダリングで使用されるディメンションを示す値を設定します。

```
void  StepsDimension(
   const int  dimension      // ディメンション
  )
```

パラメータ

dimension

[in] ディメンション（0または1）

注意事項

0 － x（水平線の後に垂直線が続きます）

1 － y（垂直線の後に水平線が続きます）

© 2000-2025, MetaQuotes Ltd.

TrendLineCoefficients（取得メソッド）

トレンドラインを配列に書き込むための比率を取得します。

```
double&  TrendLineCoefficients()
```

戻り値

トレンドラインの比率

TrendLineCoefficients（設定メソッド）

トレンドラインを配列に書き込むための比率を設定します。

```
void  TrendLineCoefficients(
   double&  coefficients[]      // 比率を各ための配列
  )
```

パラメータ

coefficients[]

[out] 比率を各ための配列<

© 2000-2025, MetaQuotes Ltd.

TrendLineColor（取得メソッド）

曲線のトレンドラインの色を取得します。

```
uint  TrendLineColor()
```

戻り値

トレンドラインの色

TrendLineColor（設定メソッド）

曲線のトレンドラインの色を設定します。

```
void  TrendLineColor(
   const uint  clr      // トレンドラインの色
  )
```

パラメータ

clr

[in] 線の色

© 2000-2025, MetaQuotes Ltd.

TrendLineVisible（取得メソッド）

トレンドラインの可視性フラグを取得します。

```
bool  TrendLineVisible()
```

戻り値

トレンドラインの可視性フラグの値

TrendLineVisible（設定メソッド）

トレンドラインの可視性フラグを設定します。

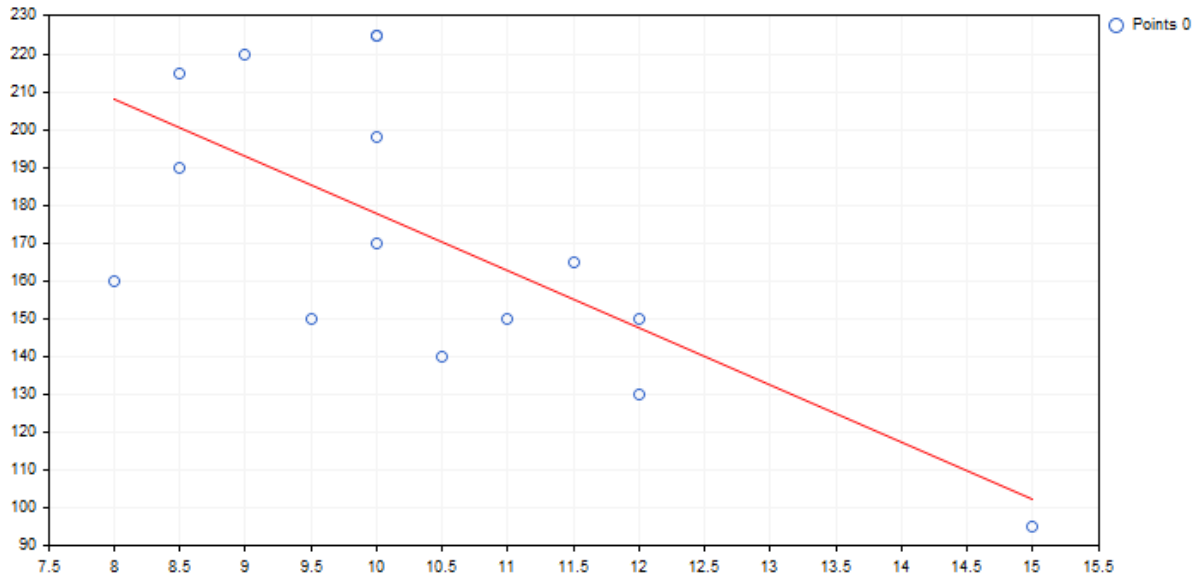
```
void  TrendLineVisible(
   const bool  visible      // フラグ値
  )
```

パラメータ

visible

[in] トレンドラインの可視性フラグの値

例：



以下は、上記トレンドラインのコードとチャート上のプロットです。

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                            TrendLineGraphic.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                            https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Graphics\Graphic.mqh>
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   double x[]
={12.0,11.5,11.0,12.0,10.5,10.0,9.0,8.5,10.0,8.5,10.0,8.0,9.5,10.0,15.0};
   double y[]
={130.0,165.0,150.0,150.0,140.0,198.0,220.0,215.0,225.0,190.0,170.0,160.0,150.0,225.
0,95.00};
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"TrendLineGraphic",0,30,30,780,380))
     {
      graphic.Attach(0,"TrendLineGraphic");
     }
//--- 曲線を作成する
   CCurve *curve=graphic.CurveAdd(x,y,CURVE_POINTS);
//--- 曲線のプロパティを設定する
   curve.TrendLineVisible(true);
   curve.TrendLineColor(ColorToARGB(clrRed));
//--- プロット
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

© 2000–2025, MetaQuotes Ltd.

Update

曲線の座標を更新します。

Y座標によって操作するバージョン。渡された配列インデックスは、ここではX座標として使用されます。

```
void  Update(
   const double&  y[]   // Y座標
  )
```

このバージョンでは、X座標とY座標が使用されます。

```
void  Update(
   const double&  x[],     // X座標
   const double&  y[]      // Y座標
  )
```

CPoint2Dポイントを使用するバージョン。.

```
void  Update(
   const CPoint2D&  points[]      // 曲線の座標
  )
```

CurveFunction関数へのポインタを使用するバージョン。

```
void  Update(
   CurveFunction  function,     // 曲線を記述する関数へのポインタ
   const double   from,         // 関数の引数の初期値
   const double   to,           // 関数の引数の最終値
   const double   step          // 引数の増分
  )
```

パラメータ

x[]

[in] X座標

y[]

[in] Y座標

points[]

[in] 曲線の座標

function

[in] 曲線を記述する関数へのポインタ

from

[in] 関数の引数の初期値

to

[in] 関数の引数の最終値

step

[in] 引数の増分

© 2000–2025, MetaQuotes Ltd.

© 2000–2025, MetaQuotes Ltd.

Visible（取得メソッド）

チャート上に関数を表示するかどうかを定義するフラグを取得します。

```
void  Visible(
   const bool  visible      //
  )
```

戻り値

チャート上に関数を表示するかどうかを定義するフラグの値

Visible（設定メソッド）

チャート上に関数を表示するかどうかを定義するフラグを設定します。

```
void  Visible(
   const bool  visible      // フラグ値
  )
```

パラメータ

visible

[in] チャート上に関数を表示するかどうかを定義するフラグの値

© 2000–2025, MetaQuotes Ltd.

CGraphic

CGraphicはカスタムチャートを作成するための基本クラスです。

説明

CGraphicクラスは、カスタムチャートを使用する際のさまざまな側面を提供します。

クラスは主要なチャート要素を格納し、パラメータを設定してプロットを実行します。

また、クラスはチャートの曲線を保存し、さまざまな表示オプションを提供します。

宣言

```
class CGraphic
```

タイトル

```
#include <Graphics\Graphic.mqh>
```

クラスメソッド

メソッド	説明
Create	チャートオブジェクトに結合するグラフィックリソースを作成します。
Destroy	チャートを削除してグラフィカルリソースを破壊します。
Update	実装された変更を表示します。
ChartObjectName	チャートに取り付けられたオブジェクトの名前を取得します。
ResourceName	グラフィカルリソースの名前を取得します。
XAxis	X軸へのポインタを取得します。
YAxis	Y軸へのポインタを取得します。
GapSize	チャート要素間のインデントのサイズを取得/設定します。
BackgroundColor	背景色を取得/設定します。
BackgroundMain	チャートヘッダを取得/設定します。
BackgroundMainSize	サブヘッダフォントサイズを取得/設定します。
BackgroundMainColor	チャートヘッダの色を取得/設定します。
BackgroundSub	サブヘッダを取得/設定します。
BackgroundSubSize	サブヘッダフォントサイズを取得/設定します。
BackgroundSubColor	チャートのサブヘッダの色を取得/設定します。

© 2000-2025, MetaQuotes Ltd.

メソッド	説明
GridLineColor	グリッドラインの色を取得/設定します。
GridBackgroundColor	グリッド背景色を取得/設定します。
GridCircleRadius	グリッドノードの点の半径を取得/設定します。
GridCircleColor	グリッドノードの点の色を取得/設定します。
GridHasCircle	グリッドノードの点描画フラグを取得/設定します。
GridAxisLineColor	実際のチャート軸の色の値を取得します。
HistoryNameWidth	曲線名を表示するための最大許容長さを取得/設定します。
HistoryNameSize	曲線名のフォントサイズを取得/設定します。
HistorySymbolSize	表記法シンボルのサイズを取得/設定します。
TextAdd	チャートにテキストを追加します。
LineAdd	チャートに線を追加します。
CurveAdd	曲線を作成してチャートに追加します。
CurvePlot	前に作成した曲線をインデックスでプロットします。
CurvePlotAll	前に作成した曲線をすべてプロットします。
CurveGetByIndex	指定されたインデックスを持つ曲線を取得します。
CurveGetByName	指定された名前を持つ曲線を取得します。
CurveRemoveByIndex	指定されたインデックスを持つ曲線を削除します。
CurveRemoveByName	指定された名前を持つ曲線を削除します。
CurvesTotal	指定されたチャートでの曲線の数を取得します。
MarksToAxisAdd	チャート軸に目盛りを追加します。
MajorMarkSize	チャート軸上のスケールの目盛のサイズを取得/設定します。
FontSet	現在フォントパラメータを設定します。
FontGet	現在フォントパラメータを取得します。
Attach	グラフィカルリソースを取得/設定してCGraphicGraphicクラスインスタンスにバインドします。
CalculateMaxMinValues	両方の軸の最小値と最大値を計算(再計算)します。
Height	チャートの高さをピクセル単位で取得します。
IndentDown	下の境界線からチャートインデントを取得/設定します。

© 2000–2025, MetaQuotes Ltd.

メソッド	説明
IndentLeft	左の境界線からチャートインデントを取得/設定します。
IndentRight	右の境界線からチャートインデントを取得/設定します。
IndentUp	上の境界線からチャートインデントを取得/設定します。
Redraw	チャートを再描画します。
ResetParameters	チャート再描画パラメータをリセットします。
ScaleX	値をX軸でスケーリングします。
ScaleY	値をY軸でスケーリングします。
SetDefaultParameters	チャートパラメータをデフォルト値に設定します。
Width	チャートの幅をピクセル単位で取得します。

© 2000–2025, MetaQuotes Ltd.

Create

チャート オブジェクト にバインド されたグラフィカルリソースを作 成します。

```
bool  Create(
   const long     chart,       // チャートI D
   const string   name,        // 名前
   const int      subwin,      // サブウィンドウインデックス
   const int      x1,          // x1座標
   const int      y1,          // y1座標
   const int      x2,          // x2座標
   const int      y2           // y1座標
   )
```

パラメータ

chart

[in] チャート ID。

name

[in] 名 前 。

subwin

[in] サブウィンド ウインデックス。

x1

[in] X1座 標 。

y1

[in] Y1座 標 。

x2

[in] X2座 標 。

y2

[in] Y2座 標 。

© 2000–2025, MetaQuotes Ltd.

Destroy

チャートを削除してグラフィカルリソースを破棄します。

```
void  Destroy()
```

© 2000–2025, MetaQuotes Ltd.

Update

変更の実装を表示します。

```
void  Update(
   const bool  redraw=true      // フラグ
   )
```

パラメータ

redraw=true

[in] フラグ値。

© 2000–2025, MetaQuotes Ltd.

ChartObjectName

チャート オブジェクト にバインド されたグラフィカルリソースの名 前 を取 得します。

```
string  ChartObjectName()
```

戻 り値

チャート オブジェクト にバインド されたグラフィカルリソースの名 前 。

© 2000-2025, MetaQuotes Ltd.

ResourceName

グラフィカルリソース名 を受 け取 ります。

```
string  ResourceName()
```

戻 り値

グラフィカルリソース名 。

© 2000–2025, MetaQuotes Ltd.

XAxis

X軸 へのポインタを返します。

```
CAxis  *XAxis()
```

戻り値

X軸 へのポインタ。

© 2000–2025, MetaQuotes Ltd.

YAxis

Y軸 へのポインタを返します。

```
CAxis  *YAxis()
```

戻り値

Y軸 へのポインタ。

© 2000–2025, MetaQuotes Ltd.

GapSize（Getメソッド）

チャートの要素間のインデントのサイズを返します。

```
int  GapSize()
```

戻り値

ピクセル単位でのインデントのサイズ。

GapSize（Setメソッド）

チャートの要素間のインデントのサイズを設定します。

```
void  GapSize(
   const int  size      // インデントサイズ
   )
```

パラメータ

size

[in] ピクセル単位でのインデントサイズ。

© 2000–2025, MetaQuotes Ltd.

BackgroundColor（Getメソッド）

背景色を返します。

```
color  BackgroundColor()
```

BackgroundColor（Setメソッド）

背景色を設定します。

```
void  BackgroundColor(
   const color  clr      // 背景色
   )
```

パラメータ

clr

[in] 背景色。

© 2000–2025, MetaQuotes Ltd.

BackgroundMain（Getメソッド）

チャート ヘッダーを返します。

```
string  BackgroundMain()
```

BackgroundMain（Setメソッド）

チャート のヘッダーテキスト を設 定します。

```
void  BackgroundMain(
   const string  main      // ヘッダーテキスト
   )
```

パラメータ

main

[in] チャート のヘッダーテキスト 。

© 2000–2025, MetaQuotes Ltd.

BackgroundMainSize（Getメソッド）

ヘッダーのサイズを返します。

```
int  BackgroundMainSize()
```

戻り値

ヘッダーフォント サイズ。

BackgroundMainSize（Setメソッド）

ヘッダーのサイズを設定します。

```
void  BackgroundMainSize(
   const int  size     // ヘッダーサイズ
   )
```

パラメータ

size

[in] ヘッダーフォント サイズ。

© 2000-2025, MetaQuotes Ltd.

BackgroundMainColor（Getメソッド）

ヘッダーの色 を返します。

```
color  BackgroundMainColor()
```

戻り値

ヘッダーの色 。

BackgroundMainColor（Setメソッド）

ヘッダーの色 を設 定します。

```
void  BackgroundMainColor(
   const color  clr      // ヘッダーの色
   )
```

パラメータ

clr

[in] ヘッダーの色 。

© 2000–2025, MetaQuotes Ltd.

BackgroundSub（Getメソッド）

サブヘッダーを返します。

```
string  BackgroundSub()
```

戻り値

サブヘッダーのテキスト。

BackgroundSub（Setメソッド）

サブヘッダーのテキストを設定します。

```
void  BackgroundSub（Setメソッド）(
   const string  sub       // サブヘッダーテキスト
   )
```

パラメータ

sub

[in] サブヘッダーテキスト。

© 2000-2025, MetaQuotes Ltd.

BackgroundSubSize（Getメソッド）

サブヘッダーフォント サイズを返します。

```
int BackgroundSubSize()
```

BackgroundSubSize（Getメソッド）

サブヘッダーフォント サイズを設 定します。

```
void  BackgroundSubSize(
   const int  size      // サブヘッダーフォントサイズ
   )
```

パラメータ

size

[in] サブヘッダーフォント サイズ。

© 2000–2025, MetaQuotes Ltd.

BackgroundSubColor（Getメソッド）

サブヘッダーの色 を返します。

```
color  BackgroundSubColor()
```

BackgroundSubColor（Setメソッド）

サブヘッダーの色 を設 定します。

```
void  BackgroundSubColor(
   const color  clr      // サブヘッダーの色
   )
```

パラメータ

clr

[in] サブヘッダーの色 。.

© 2000–2025, MetaQuotes Ltd.

GridLineColor（Getメソッド）

グリッド線の色を返します。

```
color  GridLineColor()
```

戻り値

グリッド線の色。

GridLineColor（Setメソッド）

グリッド線の色を設定します。

```
void  GridLineColor(
   const color  clr      // 線の色
   )
```

パラメータ

clr

[in] グリッド線の色。

© 2000–2025, MetaQuotes Ltd.

GridBackgroundColor（Getメソッド）

グリッドの背景色を返します。

```
color  GridBackgroundColor()
```

戻り値

グリッドの背景色。

GridBackgroundColor（Setメソッド）

グリッドの背景色を設定します。

```
void  GridBackgroundColor(
   const color  clr      // グリッドの背景色
   )
```

パラメータ

clr

[in] グリッドの背景色。

© 2000-2025, MetaQuotes Ltd.

GridCircleRadius（Getメソッド）

ネットワークノードのドットの半径を返します。

```
int  GridCircleRadius()
```

戻り値

ピクセル単位での点の半径。

GridCircleRadius（Setメソッド）

グリッドノードの点の半径を設定します。

```
void  GridCircleRadius(
   const int  r      // 半径
   )
```

パラメータ

r

[in] ピクセル単位での点の半径。

© 2000-2025, MetaQuotes Ltd.

GridCircleColor（Getメソッド）

ネットワークノードの点の色を返します。

```
color  GridCircleColor()
```

戻り値

点の色。

GridCircleColor（Setメソッド）

グリッドノードの点の色を設定します。

```
void  GridCircleColor(
   const color  clr      // 点の色
   )
```

パラメータ

clr

[in] 点の色。

© 2000-2025, MetaQuotes Ltd.

GridHasCircle（Getメソッド）

グリッドノードの点を表示するかどうかを定義するフラグを返します。

```
bool  GridHasCircle()
```

戻り値

フラグ値。

注

true — 点を表示する。

false — 点を表示しない。

GridHasCircle（Setメソッド）

グリッドノードの点を表示するかどうかを定義するフラグを設定します。

```
void  GridHasCircle(
   const bool  has
   )
```

パラメータ

has

[in] フラグ値。

注

true — 点を表示する。

false — 点を表示しない。

© 2000–2025, MetaQuotes Ltd.

GridAxisLineColor（取得メソッド）

実際のチャート軸の色の値を取得します。

```
uint  GridAxisLineColor()
```

戻り値

実際のチャート軸の色

GridAxisLineColor（設定メソッド）

実際のチャート軸の色の値を設定します。

```
void  GridAxisLineColor(
   const uint  clr      // チャート軸の色
  )
```

パラメータ

clr

[in] 実際のチャート軸の色

© 2000-2025, MetaQuotes Ltd.

HistoryNameWidth（Getメソッド）

曲線名を表示するために許容される最大長を返します。

```
int  HistoryNameWidth()
```

戻り値

ピクセル単位での最大長。

注

許容される最大表示長を超えた曲線名は、切り捨てられて点が追加されます。

HistoryNameWidth（Setメソッド）

曲線名を表示するために許容される最大長を設定します。

```
void  HistoryNameWidth(
   const int  width      // 最大長
   )
```

パラメータ

width

[in] ピクセル単位での最大長

注

許容される最大表示長を超えた曲線名は、切り捨てられて点が追加されます。

© 2000–2025, MetaQuotes Ltd.

HistoryNameSize（Getメソッド）

曲線名のフォントサイズを返します。

```
int  HistoryNameSize()
```

戻り値

曲線名のフォントサイズ。

HistoryNameSize（Setメソッド）

曲線名のフォントサイズを設定します。

```
void  HistoryNameSize（Setメソッド）(
   const int  size      // 名前のフォントサイズ
   )
```

パラメータ

size

[in] 名前のフォントサイズ。

© 2000-2025, MetaQuotes Ltd.

HistorySymbolSize（Getメソッド）

チャートの表記法シンボルのサイズを返します。

```
int  HistorySymbolSize()
```

戻り値

表記法シンボルのサイズ。

HistorySymbolSize（Setメソッド）

チャートの表記法シンボルのサイズを設定します。

```
void  HistorySymbolSize(
   const int  size      // シンボルサイズ
   )
```

パラメータ

size

[in] 表記法シンボルのサイズ。

© 2000-2025, MetaQuotes Ltd.

TextAdd

チャートにテキストを追加します。

X及びY座標での作業用のバージョン

```
void  TextAdd(
   const int     x,              // X座標
   const int     y,              // Y座標
   const string  text,           // テキスト
   const uint    clr,            // 色
   const uint    alignment=0     // 整列
   )
```

CPoint用のバージョン

```
void  TextAdd(
   const CPoint  &point,         // 点の座標
   const string  text,           // テキスト
   const uint    clr,            // 色
   const uint    alignment=0     // 整列
   )
```

パラメータ

x

[in] X座標。

y

[in] Y座標。

&point

[in] 点の座標。

text

[in] テキスト。

clr

[in] 色。

alignment=0

[in] 整列。

© 2000–2025, MetaQuotes Ltd.

LineAdd

チャートに線を追加します。

このバージョンではX及びY座標が使用されます。

```
void  LineAdd(
   const int   x1,         // x1座標
   const int   y1,         // y1座標
   const int   x2,         // x2座標
   const int   y2,         // y2座標
   const uint  clr,        // 色
   const uint  style       // スタイル
   )
```

CPoint用のバージョン

```
void  LineAdd2(
   const CPoint  &point1,     // 1番目の点の座標
   const CPoint  &point2,     // 2番目の点の座標
   const uint    clr,         // 色
   const uint    style        // スタイル
   )
```

パラメータ

x1
[in] X1座標。

y1
[in] Y1座標。

x2
[in] X2座標。

y2
[in] Y2座標。

&point1
[in] 1番目の点の座標。

&point2
[in] 2番目の点の座標。

clr
[in] 色。

style
[in] スタイル。

© 2000-2025, MetaQuotes Ltd.

CurveAdd

曲線を作成してチャートに追加します。

このバージョンではY座標が使用されます（曲線の色は自動的に設定されます）。

```
CCurve* CurveAdd(
   const double      &y[],          // Y座標
   ENUM_CURVE_TYPE   type,          // 曲線の種類
   const string      name=NULL      // 曲線名
   )
```

注

Y配列のインデックスがカーブのX座標として使用されます。

このバージョンではX及びY座標が使用されます（曲線の色は自動的に設定されます）。

```
CCurve*  CurveAdd(
   const double      &x[],          // X座標
   const double      &y[],          // Y座標
   ENUM_CURVE_TYPE   type,          // 曲線の種類
   const string      name=NULL      // 曲線名
   )
```

このバージョンはCPoint2Dの点との作業用です（曲線の色は自動的に設定されます）。

```
CCurve*  CurveAdd(
   const CPoint2D    &points[],     // 点の座標
   ENUM_CURVE_TYPE   type,          // 曲線の種類
   const string      name=NULL      // 曲線名
   )
```

このバージョンはCurveFunction関数へのポインタとの作業用です（曲線の色は自動的に設定されます）。

```
CCurve*  CurveAdd(
   CurveFunction     function,      // 関数ポインタ
   const double      from,          // 引数の初期値
   const double      to,            // 引数の最終値
   const double      step,          // 引数による増分
   ENUM_CURVE_TYPE   type,          // 曲線の種類
   const string      name=NULL      // 曲線名
   )
```

© 2000–2025, MetaQuotes Ltd.

下記はY座標で作業するためのバージョンです（曲線の色はユーザーによって設定されます）。

```
CCurve*  CurveAdd(
   const double      &y[],          // Y座標
   const uint        clr,           // 曲線の色
   ENUM_CURVE_TYPE   type,          // 曲線の種類
   const string      name=NULL      // 曲線名
   )
```

注

Y配列のインデックスがカーブのX座標として使用されます。

このバージョンではX及びY座標が使用されます（曲線の色はユーザーによって設定されます）。

```
CCurve*  CurveAdd(
   const double      &x[],          // X座標
   const double      &y[],          // Y座標
   const uint        clr,           // 曲線の色
   ENUM_CURVE_TYPE   type,          // 曲線の種類
   const string      name=NULL      // 曲線名
   )
```

このバージョンはCPoint2Dの点との作業用です（曲線の色はユーザーによって設定されます）。

```
CCurve*  CurveAdd(
   const CPoint2D    &points[],     // 点の座標
   const uint        clr,           // 曲線の色
   ENUM_CURVE_TYPE   type,          // 曲線の種類
   const string      name=NULL      // 曲線名
   )
```

このバージョンはCurveFunction関数へのポインタとの作業用です（曲線の色はユーザーによって設定されます）。

```
CCurve*  CurveAdd(
   CurveFunction     function,      // 関数ポインタ
   const double      from,          // 引数の初期値
   const double      to,            // 引数の最終値
   const double      step,          // 引数による増分
   const uint        clr,           // 曲線の色
   ENUM_CURVE_TYPE   type,          // 曲線の種類
   const string      name=NULL      // 曲線名
   )
```

© 2000–2025, MetaQuotes Ltd.

パラメータ

&x[]
[in] X座標。

&y[]
[in] Y座標。

&points[]
[in] 点の座標。

function
[in] 関数ポインタ。

from
[in] 引数の初期値。

to
[in] 引数の最終値。

step
[in] 引数の増分。

type
[in] 曲線の種類。

name=NULL
[in] 曲線名。

clr
[in] 曲線の色。

戻り値

作成されたカーブへのポインタ。

© 2000–2025, MetaQuotes Ltd.

CurvePlot

以前作成され指定されたインデックスを持つ曲線を表示します。

```
bool  CurvePlot(
   const int  index      // インデックス
   )
```

パラメータ

index

[in] 曲線のインデックス。

戻り値

成功した場合はtrue、そうでなければfalse。

© 2000–2025, MetaQuotes Ltd.

CurvePlotAll

以前チャートに追加された曲線を全部表示します。

```
bool  CurvePlotAll()
```

戻り値

成功した場合はtrue、そうでなければfalse。

© 2000-2025, MetaQuotes Ltd.

CurveGetByIndex

指定されたインデックスで曲線を取得します。

```
CCurve*  CurveGetByIndex(
   const int  index      // 曲線のインデックス
   )
```

パラメータ

index

[in] 曲線のインデックス。

戻り値

指定されたインデックスをもつ曲線へのポインタ。

© 2000–2025, MetaQuotes Ltd.

CurveGetByName

指定された名前で曲線を取得します。

```
CCurve*  CurveGetByName(
   const string  name      // 曲線名
   )
```

パラメータ

name

[in] 曲線名。

戻り値

指定された名前を持つ最初に見つかった曲線へのポインタ。

© 2000–2025, MetaQuotes Ltd.

CurvesTotal

指定されたチャートでの曲線の数を取得します。

```
int  CurvesTotal()
```

戻り値

曲線数

注意事項

現在のチャートのすべての曲線は、描画スタイルと表示スタイルに関係なく考慮されます。

© 2000-2025, MetaQuotes Ltd.

CurveRemoveByName

指定された名前を持つ曲線を削除します。

```
bool  CurveRemoveByName(
   const string  name      // 曲線名
  )
```

パラメータ

name

[in] 削除される曲線の名前

戻り値

成功の場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

CurveRemoveByIndex

指定されたインデックスを持つ曲線を削除します。

```
bool  CurveRemoveByIndex(
   const int  index      // 曲線のインデックス
  )
```

パラメータ

index

[in] 削除される曲線のインデックス

戻り値

成功の場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

MarksToAxisAdd

指定されたチャート軸に目盛りを追加します。

```
bool  MarksToAxisAdd(
   const double         &marks[],        // 目盛りの座標
   const int            mark_size,       // 目盛りのサイズ
   ENUM_MARK_POSITION   position,        // 目盛りの位置
   const int            dimension=0      // 大きさ
   )
```

パラメータ

&marks[]

[in] 目盛りの座標。

mark_size

[in] 目盛りサイズ。

position

[in] 目盛りの位置。

dimension=0

[in] 0 — X軸への追加。

1 — Y軸への追加。

戻り値

成功した場合はtrue、そうでなければfalse。

© 2000-2025, MetaQuotes Ltd.

MajorMarkSize（Getメソッド）

チャート軸の目盛りのサイズを返します。

```
int  MajorMarkSize()
```

MajorMarkSize（Setメソッド）

チャート軸の目盛りのサイズを返します。

```
void  MajorMarkSize(
   const int  size      // 目盛りサイズ
   )
```

パラメータ

size

[in] ピクセル単位での目盛りサイズ

© 2000–2025, MetaQuotes Ltd.

FontSet

現在のフォントパラメータを設定します。

```
bool  FontSet(
   const string  name,        // 名前
   const int     size,        // サイズ
   const uint    flags=0,     // フラグ
   const uint    angle=0      // 角度
   )
```

パラメータ

name

[in] 名前。

size

[in] サイズ。

flags=0

[in] フラグ。

angle=0

[in] 角度。

戻り値

成功した場合はtrue、そうでなければfalse。

© 2000–2025, MetaQuotes Ltd.

FontGet

現在のフォントパラメータを取得します。

```
void  FontGet(
   string  &name,      // 名前
   int     &size,      // サイズ
   uint    &flags,     // フラグ
   uint    &angle      // 角度
   )
```

パラメータ

&name

[out] 名前。

&size

[out] サイズ。

&flags

[out] フラグ。

&angle

[out] 角度。

© 2000–2025, MetaQuotes Ltd.

Attach

OBJ_BITMAP_LABELオブジェクトからグラフィカルリソースを取得してCGraphicクラスのインスタンスにバインドするためのバージョンです。

```
bool  Attach(
   const long     chart_id,      // チャート識別子
   const string   objname        // グラフィカルオブジェクト名
  )
```

OBJ_BITMAP_LABELオブジェクトのグラフィカルリソース作成してCGraphicクラスのインスタンスにバインドするためのバージョンです。

```
bool  Attach(
   const long     chart_id,      // チャート識別子
   const string   objname,       // グラフィカルオブジェクト名
   const int      width,         // 画像幅
   const int      height         // 画像の高さ
  )
```

パラメータ

chart_id

[in] チャート識別子。

objname

[in] グラフィックオブジェクトの名称

幅

[in] リソースの画像幅

height

[in] リソースの画像の高さ

戻り値

成功の場合はtrue、オブジェクトがバインドできなかった場合はfalse

© 2000-2025, MetaQuotes Ltd.

CalculateMaxMinValues

両方の軸の最小値と最大値を計算（再計算）します。

```
void  CalculateMaxMinValues()
```

© 2000–2025, MetaQuotes Ltd.

Height

チャート の高 さをピクセル単 位 で取 得 します。

```
int  Height()
```

戻 り値

ピクセル単 位 のチャート の高 さ

© 2000–2025, MetaQuotes Ltd.

IndentDown（取得メソッド）

下の境界線からチャートインデントを取得します。

```
int  IndentDown()
```

戻り値

ピクセル単位のインデント

IndentDown（設定メソッド）

下の境界線からチャートインデントを設定します。

```
void  IndentDown(
   const int  down      // インデントサイズ
  )
```

パラメータ

down

[in] ピクセル単位のインデントサイズ

© 2000-2025, MetaQuotes Ltd.

IndentLeft（取得メソッド）

左の境界線からチャートインデントを取得します。

```
int  IndentLeft()
```

戻り値

ピクセル単位のインデント

IndentLeft（設定メソッド）

左の境界線からチャートインデントを設定します。

```
void  IndentLeft(
   const int  left      // インデントサイズ
  )
```

パラメータ

left

[in] ピクセル単位のインデントサイズ

© 2000–2025, MetaQuotes Ltd.

IndentRight（取得メソッド）

右の境界線からチャートインデントを取得します。

```
int  IndentRight()
```

戻り値

ピクセル単位のインデント

IndentRight（設定メソッド）

右の境界線からチャートインデントを設定します。

```
void  IndentRight(
   const int  right      // インデントサイズ
  )
```

パラメータ

right

[in] ピクセル単位のインデントサイズ

© 2000–2025, MetaQuotes Ltd.

IndentUp（取得メソッド）

上の境界線からチャートインデントを取得します。

```
int  IndentUp()
```

戻り値

ピクセル単位のインデント

IndentUp（設定メソッド）

上の境界線からチャートインデントを設定します。

```
void  IndentUp(
   const int  up       // インデントサイズ
  )
```

パラメータ

up

[in] ピクセル単位のインデント値

© 2000–2025, MetaQuotes Ltd.

Redraw

チャートを再描画します。

```
bool  Redraw(
   const bool  rescale=false      // フラグ値
  )
```

パラメータ

rescale=false

[in] チャートを再スケーリングするかどうかを示すフラグ

戻り値

成功の場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

ResetParameters

チャート再描画パラメータをリセットします。

```
void  ResetParameters()
```

© 2000–2025, MetaQuotes Ltd.

ScaleX

値をX軸でスケーリングします。

```
virtual int  ScaleX(
   double  x      // X軸による値
  )
```

パラメータ

x

[in] X軸による実の値

戻り値

ピクセル単位での値

注意事項

実際の値はピクセルにスケーリングしてチャートに表示されます。

© 2000–2025, MetaQuotes Ltd.

ScaleY

値をY軸でスケーリングします。

```
virtual int  ScaleY(
   double  y      // Y軸による値
  )
```

パラメータ

y

[in] Y軸による実の値

戻り値

ピクセル単位での値

注意事項

実際の値はピクセルにスケーリングしてチャートに表示されます。

© 2000–2025, MetaQuotes Ltd.

SetDefaultParameters

チャート パラメータをデフォルト 値 に設 定 します。

```
void  SetDefaultParameters()
```

© 2000–2025, MetaQuotes Ltd.

Width

チャート の幅 をピクセル単 位 で取 得します。

```
int  Width()
```

戻 り値

ピクセル単 位 でのチャート の幅

© 2000–2025, MetaQuotes Ltd.

テクニカル指標と時系列

このセクションには、指標操作のクラスの操作に関する技術的な詳細と関連するMQL5　標準ライブラリコンポーネントの説明が含まれています。

指標操作のクラスの使用は、アプリケーション（スクリプト、エキスパートアドバイザー）の作成時に時間を節約します。

MQL5　　標準ライブラリのテクニカル指標及び時系列に関連したファイルは、端末の作業ディレクトリのInclude¥Indicators フォルダ内に配置されています。

クラス/グループ	説明
基本クラス	基本及び補助クラスのグループ
時系列クラス	時系列クラスのグループ
トレンド指標	トレンド指標クラスのグループ
オシレーター	オシレーター指標クラスのグループ
ボリューム指標	ボリューム指標クラスのグループ
ビルウィリアムズ指標	ビルウィリアムズ指標クラス
カスタム指標	カスタム指標クラス

© 2000–2025, MetaQuotes Ltd.

基本及び補助テクニカル指標と時系列

このセクションには、基本及び補助テクニカル指標のクラスの操作に関する技術的な詳細と関連するMQL5標準ライブラリコンポーネントの説明が含まれています。

クラス/グループ	説明
CSpreadBuffer	スプレッド履歴バッファクラス
CTimeBuffer	始値履歴バッファクラス
CTickVolumeBuffer	ティックボリューム履歴バッファクラス
CRealVolumeBuffer	出来高履歴バッファサイズ
CDoubleBuffer	double 型データバッファ基本クラス
COpenBuffer	バー始値バッファクラス
CHighBuffer	バー高値バッファクラス
CLowBuffer	バー安値バッファクラス
CCloseBuffer	バー終値バッファクラス
CIndicatorBuffer	テクニカル指標バッファクラス
CSeries	時系列データアクセスの基本クラス
CPriceSeries	価格データアクセスの基本クラス
CIndicator	テクニカル指標の基本クラス
CIndicators	テクニカル指標及び列のコレクション

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000–2025, MetaQuotes Ltd.

CSpreadBuffer

CSpreadBuffer はバーのスプレッド履歴への簡易化されたアクセスを可能にするクラスです。

説明

CSpreadBuffer クラスはバーのスプレッド履歴への簡易化されたアクセスを提供します。

宣言

```
class CSpreadBuffer: public CArrayInt
```

タイトル

```
#include <Indicators\TimeSeries.mqh>
```

継承階層

CObject
  CArray
    CArrayInt
      CSpreadBuffer

クラスメソッド

属性	
Size	バッファサイズを設定します。
設定	
SetSymbolPeriod	銘柄と期間を設定します。
データアクセスメソッド	
At	バッファ要素をインデックスで取得します。
データ更新メソッド	
virtual Refresh	バッファを更新します。
virtual RefreshCurrent	現在値を更新します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayInt

Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete,

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

© 2000–2025, MetaQuotes Ltd.

Size

バッファサイズを設 定します。

```
void  Size(
   const int  size      // 新しいサイズ
  )
```

パラメータ

size

[in] 新しいバッファサイズ

© 2000–2025, MetaQuotes Ltd.

SetSymbolPeriod

銘柄と期間を設定します。

```
void  SetSymbolPeriod(
   const string           symbol,     // シンボル
   const ENUM_TIMEFRAMES  period      // 期間
  )
```

パラメータ

symbol

[in] 新しいシンボル

period

[in] 新しいシンボル(ENUM_TIMEFRAMES 列挙)

© 2000-2025, MetaQuotes Ltd.

At

バッファ要素をインデックスで取得します。

```
int  At(
   const int  index      // インデックス
   ) const
```

パラメータ

index

[in] バッファ要素のインデックス

戻り値

指定されたインデックスを持つバッファ要素

© 2000–2025, MetaQuotes Ltd.

Refresh

バッファを更新します。

```
virtual bool  Refresh()
```

戻り値

成功の場合は true、バッファが更新できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

RefreshCurrent

バッファの現在の(ゼロ番目の)要素を更新します。

```
virtual bool  RefreshCurrent()
```

戻り値

成功の場合はtrue、バッファが更新できなかった場合はfalse

© 2000-2025, MetaQuotes Ltd.

CTimeBuffer

CTimeBuffer はバーの開始時刻履歴への簡易化されたアクセスを可能にするクラスです。

説明

CTimeBuffer クラスはバーの開始時刻履歴への簡易化されたアクセスを提供します。

宣言

```
class CTimeBuffer: public CArrayLong
```

タイトル

```
#include <Indicators\TimeSeries.mqh>
```

継承階層

CObject
  CArray
    CArrayLong
      CTimeBuffer

クラスメソッド

属性	
Size	バッファサイズを設定します。
設定	
SetSymbolPeriod	銘柄と期間を設定します。
データアクセスメソッド	
At	バッファ要素をインデックスで取得します。
データ更新メソッド	
virtual Refresh	バッファを更新します。
virtual RefreshCurrent	現在値を更新します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayLong

Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete,

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

© 2000–2025, MetaQuotes Ltd.

Size

バッファサイズを設定します。

```
void  Size(
   const int  size      // 新しいサイズ
  )
```

パラメータ

size

[in] 新しいバッファサイズ

© 2000–2025, MetaQuotes Ltd.

SetSymbolPeriod

銘柄と期間を設定します。

```
void  SetSymbolPeriod(
   const string           symbol,     // シンボル
   const ENUM_TIMEFRAMES  period      // 期間
  )
```

パラメータ

symbol

[in] 新しいシンボル

period

[in] 新しいシンボル(ENUM_TIMEFRAMES 列挙)

© 2000–2025, MetaQuotes Ltd.

At

バッファ要素をインデックスで取得します。

```
long  At(
   const int  index      // インデックス
   ) const
```

パラメータ

index

[in] バッファ要素のインデックス

戻り値

指定されたインデックスを持つバッファ要素

© 2000–2025, MetaQuotes Ltd.

Refresh

バッファを更新します。

```
virtual bool  Refresh()
```

戻り値

成功の場合は true、バッファが更新できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

RefreshCurrent

バッファの現在の(ゼロ番目の)要素を更新します。

```
virtual bool  RefreshCurrent()
```

戻り値

成功の場合はtrue、バッファが更新できなかった場合はfalse

© 2000-2025, MetaQuotes Ltd.

CTickVolumeBuffer

CTickVolumeBuffer はバーのティックボリューム履歴への簡易化されたアクセスを可能にするクラスです。

説明

CTickVolumeBuffer クラスはバーのティックボリューム履歴へのアクセスを提供します。

宣言

```
class CTickVolumeBuffer: public CArrayLong
```

タイトル

```
#include <Indicators\TimeSeries.mqh>
```

継承階層

CObject
  CArray
    CArrayLong
      CTickVolumeBuffer

クラスメソッド

属性	
Size	バッファサイズを設定します。
設定	
SetSymbolPeriod	銘柄と期間を設定します。
データアクセスメソッド	
At	バッファ要素をインデックスで取得します。
データ更新メソッド	
virtual Refresh	バッファを更新します。
virtual RefreshCurrent	現在値を更新します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayLong

Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete,

© 2000-2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

© 2000–2025, MetaQuotes Ltd.

Size

バッファサイズを設定します。

```
void  Size(
   const int  size      // 新しいサイズ
  )
```

パラメータ

size

[in] 新しいバッファサイズ

© 2000–2025, MetaQuotes Ltd.

SetSymbolPeriod

銘柄と期間を設定します。

```
void  SetSymbolPeriod(
   const string           symbol,     // シンボル
   const ENUM_TIMEFRAMES  period      // 期間
  )
```

パラメータ

symbol

[in] 新しいシンボル

period

[in] 新しいシンボル(ENUM_TIMEFRAMES 列挙)

© 2000-2025, MetaQuotes Ltd.

At

バッファ要素をインデックスで取得します。

```
long  At(
   const int  index      // インデックス
   ) const
```

パラメータ

index

[in] バッファ要素のインデックス

戻り値

指定されたインデックスを持つバッファ要素

© 2000-2025, MetaQuotes Ltd.

Refresh

バッファを更新します。

```
virtual bool  Refresh()
```

戻り値

成功の場合は true、バッファが更新できなかった場合は false

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

バッファの現在の(ゼロ番目の)要素を更新します。

```
virtual bool  RefreshCurrent()
```

戻り値

成功の場合はtrue、バッファが更新できなかった場合はfalse

© 2000-2025, MetaQuotes Ltd.

CRealVolumeBuffer

CRealVolumeBuffer はバーの出来高履歴への簡易化されたアクセスを可能にするクラスです。

説明

CTickVolumeBuffer クラスはバーの出来高履歴へのアクセスを提供します。

宣言

```
class CRealVolumeBuffer: public CArrayLong
```

タイトル

```
#include <Indicators\TimeSeries.mqh>
```

継承階層

CObject
  CArray
    CArrayLong
      CRealVolumeBuffer

クラスメソッド

属性	
Size	バッファサイズを設定します。
設定	
SetSymbolPeriod	銘柄と期間を設定します。
データアクセスメソッド	
At	バッファ要素をインデックスで取得します。
データ更新メソッド	
virtual Refresh	バッファを更新します。
virtual RefreshCurrent	現在値を更新します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayLong

Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete,

© 2000-2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

© 2000–2025, MetaQuotes Ltd.

Size

バッファサイズを設定します。

```
void  Size(
   const int  size      // 新しいサイズ
  )
```

パラメータ

size

[in] 新しいバッファサイズ

© 2000–2025, MetaQuotes Ltd.

SetSymbolPeriod

銘柄と期間を設定します。

```
void  SetSymbolPeriod(
   const string           symbol,     // シンボル
   const ENUM_TIMEFRAMES  period      // 期間
  )
```

パラメータ

symbol

[in] 新しいシンボル

period

[in] 新しいシンボル(ENUM_TIMEFRAMES 列挙)

© 2000-2025, MetaQuotes Ltd.

At

バッファ要素をインデックスで取得します。

```
long  At(
   const int  index      // インデックス
   ) const
```

パラメータ

index

[in] バッファ要素のインデックス

戻り値

指定されたインデックスを持つバッファ要素

© 2000-2025, MetaQuotes Ltd.

Refresh

バッファを更新します。

```
virtual bool  Refresh()
```

戻り値

成功の場合は true、バッファが更新できなかった場合は false

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

バッファの現在の(ゼロ番目の)要素を更新します。

```
virtual bool  RefreshCurrent()
```

戻り値

成功の場合はtrue、バッファが更新できなかった場合はfalse

© 2000-2025, MetaQuotes Ltd.

CDoubleBuffer

CDoubleBuffer は double 型のデータバッファへの簡易化されたアクセスを可能にするクラスです。

説明

CDoubleBuffer クラスは double 型のデータへのアクセスを提供します。

宣言

```
class CDoubleBuffer: public CArrayDouble
```

タイトル

```
#include <Indicators\TimeSeries.mqh>
```

継承階層

CObject
  CArray
    CArrayDouble
      CDoubleBuffer

直接子孫

CCloseBuffer, CHighBuffer, CIndicatorBuffer, CLowBuffer, COpenBuffer

クラスメソッド

属性	
Size	バッファサイズを設定します。
設定	
SetSymbolPeriod	銘柄と期間を設定します。
データアクセスメソッド	
At	バッファ要素を取得します。
データ更新メソッド	
virtual Refresh	バッファを更新します。
virtual RefreshCurrent	現在値を更新します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayDouble

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

© 2000–2025, MetaQuotes Ltd.

Size

バッファサイズを設定します。

```
void  Size(
   const int  size      // 新しいサイズ
  )
```

パラメータ

size

[in] 新しいバッファサイズ

© 2000–2025, MetaQuotes Ltd.

SetSymbolPeriod

銘柄と期間を設定します。

```
void  SetSymbolPeriod(
   const string           symbol,     // シンボル
   const ENUM_TIMEFRAMES  period      // 期間
  )
```

パラメータ

symbol

[in] 新しいシンボル

period

[in] 新しいシンボル(ENUM_TIMEFRAMES 列挙)

© 2000–2025, MetaQuotes Ltd.

At

バッファ要素をインデックスで取得します。

```
double  At(
   const int  index      // インデックス
   ) const
```

パラメータ

index

[in] バッファ要素のインデックス

戻り値

指定されたインデックスを持つバッファ要素

© 2000–2025, MetaQuotes Ltd.

Refresh

バッファを更新します。

```
virtual bool  Refresh()
```

戻り値

成功の場合は true、バッファが更新できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

RefreshCurrent

バッファの現在の(ゼロ番目の)要素を更新します。

```
virtual bool  RefreshCurrent()
```

戻り値

成功の場合はtrue、バッファが更新できなかった場合はfalse

© 2000-2025, MetaQuotes Ltd.

COpenBuffer

COpenBuffer はバーの始値履歴への簡易化されたアクセスを可能にするクラスです。

説明

COpenBuffer クラスはバーの始値履歴への簡易化されたアクセスを提供します。

宣言

```
class COpenBuffer: public CDoubleBuffer
```

タイトル

```
#include <Indicators\TimeSeries.mqh>
```

継承階層

CObject
  CArray
    CArrayDouble
      CDoubleBuffer
        COpenBuffer

クラスメソッド

データ更新メソッド	
virtual Refresh	バッファを更新します。
virtual RefreshCurrent	現在値を更新します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayDouble

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

クラスから継承されたメソッド CDoubleBuffer

Size, At, SetSymbolPeriod

© 2000–2025, MetaQuotes Ltd.

Refresh

バッファを更新します。

```
virtual bool  Refresh()
```

戻り値

成功の場合は true、バッファが更新できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

RefreshCurrent

バッファの現在の(ゼロ番目の)要素を更新します。

```
virtual bool  RefreshCurrent()
```

戻り値

成功の場合はtrue、バッファが更新できなかった場合はfalse

© 2000-2025, MetaQuotes Ltd.

CHighBuffer

CHighBuffer は足の高値履歴への簡易化されたアクセスを可能にするクラスです。

説明

CHighBuffer クラスは足の高値履歴への簡易化されたアクセスを提供します。

宣言

```
class CHighBuffer: public CDoubleBuffer
```

タイトル

```
#include <Indicators\TimeSeries.mqh>
```

継承階層

CObject
  CArray
    CArrayDouble
      CDoubleBuffer
        CHighBuffer

クラスメソッド

データ更新メソッド	
virtual Refresh	バッファを更新します。
virtual RefreshCurrent	現在値を更新します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayDouble

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

クラスから継承されたメソッド CDoubleBuffer

Size, At, SetSymbolPeriod

© 2000–2025, MetaQuotes Ltd.

Refresh

バッファを更新します。

```
virtual bool  Refresh()
```

戻り値

成功の場合は true、バッファが更新できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

RefreshCurrent

バッファの現在の(ゼロ番目の)要素を更新します。

```
virtual bool  RefreshCurrent()
```

戻り値

成功の場合はtrue、バッファが更新できなかった場合はfalse

© 2000-2025, MetaQuotes Ltd.

CLowBuffer

CLowBuffer is はバーの安値履歴への簡易化されたアクセスを可能にするクラスです。

説明

CLowBuffer クラスはバーの安値履歴への簡易化されたアクセスを提供します。

宣言

```
class CLowBuffer: public CDoubleBuffer
```

タイトル

```
#include <Indicators\TimeSeries.mqh>
```

継承階層

CObject
  CArray
    CArrayDouble
      CDoubleBuffer
        CLowBuffer

クラスメソッド

データ更新メソッド	
virtual Refresh	バッファを更新します。
virtual RefreshCurrent	現在値を更新します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayDouble

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

クラスから継承されたメソッド CDoubleBuffer

Size, At, SetSymbolPeriod

© 2000-2025, MetaQuotes Ltd.

Refresh

バッファを更新します。

```
virtual bool  Refresh()
```

戻り値

成功の場合は true、バッファが更新できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

RefreshCurrent

バッファの現在の(ゼロ番目の)要素を更新します。

```
virtual bool  RefreshCurrent()
```

戻り値

成功の場合はtrue、バッファが更新できなかった場合はfalse

© 2000-2025, MetaQuotes Ltd.

CCloseBuffer

CCloseBuffer はバーの終値履歴への簡易化されたアクセスを可能にするクラスです。

説明

CCloseBuffer クラスはバーの終値履歴への簡易化されたアクセスを提供します。

宣言

```
class CCloseBuffer: public CDoubleBuffer
```

タイトル

```
#include <Indicators\TimeSeries.mqh>
```

継承階層

CObject
  CArray
    CArrayDouble
      CDoubleBuffer
        CCloseBuffer

クラスメソッド

データ更新メソッド	
virtual Refresh	バッファを更新します。
virtual RefreshCurrent	現在値を更新します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayDouble

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

クラスから継承されたメソッド CDoubleBuffer

Size, At, SetSymbolPeriod

© 2000–2025, MetaQuotes Ltd.

Refresh

バッファを更新します。

```
virtual bool  Refresh()
```

戻り値

成功の場合は true、バッファが更新できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

RefreshCurrent

バッファの現在の(ゼロ番目の)要素を更新します。

```
virtual bool  RefreshCurrent()
```

戻り値

成功の場合はtrue、バッファが更新できなかった場合はfalse

© 2000-2025, MetaQuotes Ltd.

CIndicatorBuffer

CIndicatorBuffer は指標バッファのデータへの簡易化されたアクセスを可能にするクラスです。

説明

CIndicatorBuffer クラスはテクニカル指標のデータバッファへのアクセスを提供します。

宣言

```
class CIndicatorBuffer: public CDoubleBuffer
```

タイトル

```
#include <Indicators\Indicator.mqh>
```

継承階層

CObject
CArray
CArrayDouble
CDoubleBuffer
CIndicatorBuffer

クラスメソッド

属性	
Offset	バッファオフセットを取得/設定します。
Name	バッファ名を取得/設定します。
データアクセスメソッド	
At	バッファの要素を取得します。
データ更新メソッド	
Refresh	バッファを更新します。
RefreshCurrent	現在の値のみを更新します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayDouble

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update,

© 2000-2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

クラスから継承されたメソッド CDoubleBuffer

Size, At, SetSymbolPeriod

© 2000–2025, MetaQuotes Ltd.

Offset

バッファオフセットを取得します。

```
int  Offset() const
```

戻り値

バッファオフセット

Offset

バッファオフセットを設定します。

```
void  Offset(
   const int  offset      // オフセット
  )
```

パラメータ

offset

[in] 新しいバッファオフセット

© 2000–2025, MetaQuotes Ltd.

Name

バッファ名を取得します。

```
string  Name() const
```

戻り値

バッファ名

Name

バッファ名を設定します。

```
void  Name(
   const string  name      // 名称
  )
```

パラメータ

name

[in] 新しいバッファ名

© 2000–2025, MetaQuotes Ltd.

At

バッファ要素をインデックスで取得します。

```
double  At(
   int  index      // インデックス
   ) const
```

パラメータ

index

[in] バッファ要素のインデックス

戻り値

指定されたインデックスを持つバッファ要素

© 2000–2025, MetaQuotes Ltd.

Refresh

バッファを全体的に更新します。

```
bool  Refresh(
   const int  handle,     // ハンドル
   const int  num         // バッファ番号
  )
```

パラメータ

handle

[in] 指標ハンドル

num

[in] 指標バッファのインデックス

戻り値

成功の場合は true、バッファが更新できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

RefreshCurrent

現在の(ゼロ番目の)バッファ要素を更新します。

```
bool  RefreshCurrent(
   const int  handle,      // 指標ハンドル
   const int  num          // バッファ番号
  )
```

パラメータ

handle

[in] 指標ハンドル

num

[in] バッファ番号

戻り値

成功の場合は true、バッファが更新できなかった場合は false

© 2000-2025, MetaQuotes Ltd.

CSeries

CSeries は MQL5 標準ライブラリの時系列データへのアクセスの基本クラスです。

説明

CSeries クラスは派生クラスの時系列関数への容易化されたアクセスを提供します。

宣言

```
class CSeries: public CArrayObj
```

タイトル

```
#include <Indicators\Series.mqh>
```

継承階層

CObject
  CArray
    CArrayObj
      CSeries

直接子孫

CIndicator, CiRealVolume, CiSpread, CiTickVolume, CiTime, CPriceSeries

クラスメソッド

属性	
Name	時系列または指標の名称を取得します。
BuffersTotal	時系列または指標のバッファ数を取得します。
Timeframe	時系列または指標の時間軸フラグを取得します。
Symbol	時系列または指標のシンボルを取得します。
Period	時系列または指標の期間を取得します。
RefreshCurrent	現在データ更新のフラグを取得/設定します。
データアクセスメソッド	
virtual BufferResize	時系列または指標のバッファサイズを設定します。
データ更新メソッド	
virtual Refresh	時系列または指標のデータを更新します。
PeriodDescription	期間を文字列として取得します。

© 2000-2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

© 2000–2025, MetaQuotes Ltd.

Name

時系列または指標の名称を取得します。

```
string  Name() const
```

戻り値

時系列または指標の名称

© 2000–2025, MetaQuotes Ltd.

BuffersTotal

時系列または指標のバッファ数を取得します。

```
int  BuffersTotal() const
```

戻り値

時系列または指標のバッファ数

注意事項

時系列のバッファ数は 1 です。

© 2000–2025, MetaQuotes Ltd.

Timeframe

時系列または指標の時間軸フラグを取得します。

```
int  Timeframe() const
```

戻り値

時系列または指標の時間軸フラグ

注意事項

時間軸の可視性フラグです。

© 2000-2025, MetaQuotes Ltd.

Symbol

時系列または指標のシンボルを取得します。

```
string  Symbol() const
```

戻り値

時系列または指標のシンボル

© 2000–2025, MetaQuotes Ltd.

Period

時系列または指標の期間を取得します。

```
ENUM_TIMEFRAMES  Period() const
```

戻り値

時系列または指標の期間(ENUM_TIMEFRAMES 列挙の値)

© 2000–2025, MetaQuotes Ltd.

RefreshCurrent

時系列または指標の現在値を更新するフラグを設定します。

```
string  RefreshCurrent(
   const bool  flag      // 新しいフラグ
  )
```

パラメータ

flag

[in] 新しいフラグ

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

BufferSize

時系列バッファまたは指標バッファで使用可能なデータの量を返します。

```
int  BufferSize() const
```

戻り値

時系列バッファまたは指標バッファで使用可能なデータの量

© 2000-2025, MetaQuotes Ltd.

BufferResize

時系列または指標のバッファサイズを設定します。

```
virtual bool  BufferResize(
   const int  size      // 新しいサイズ
  )
```

パラメータ

size

[in] 新しいバッファサイズ

戻り値

成功の場合は true、それ以外の場合は false

注意事項

全ての時系列や指標バッファのサイズは同じです。

© 2000–2025, MetaQuotes Ltd.

Refresh

時系列または指標のデータを更新します。

```
virtual void  Refresh(
   const int  flags      // フラグ
  )
```

パラメータ

flags

[in] 更新する時間軸（フラグ）

© 2000–2025, MetaQuotes Ltd.

PeriodDescription

指定された ENUM_TIMEFRAMES 列挙を文字列として取得します。

```
string  PeriodDescription(
   const int  val=0      // 値
  )
```

パラメータ

val=0

[in] 変換する値

戻り値

指定された ENUM_TIMEFRAMES 列挙の文字列表現

注意事項

値が指定されていないまたはゼロに等しい場合、時系列や指標の時間軸が返されます。

© 2000–2025, MetaQuotes Ltd.

CPriceSeries

CPriceSeries は価格データアクセスの基本クラスです。

説明

CSeries クラスは派生クラスの MQL5 価格データ関数への容易化されたアクセスを提供します。

宣言

```
class CPriceSeries: public CSeries
```

タイトル

```
#include <Indicators\TimeSeries.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CPriceSeries

直接子孫

CiClose, CiHigh, CiLow, CiOpen

クラスメソッド

作成メソッド	
virtual BufferResize	バッファサイズを設定します。
データアクセスメソッド	
virtual GetData	指定されたバッファ要素をインデックスで取得します。
データ更新メソッド	
virtual Refresh	時系列データを更新します。
データ検索メソッド	
virtual MinIndex	指定された範囲内の最小要素のインデックスを取得します。
virtual MinValue	指定された範囲内の最小要素の値とインデックスを取得します。
virtual MaxIndex	指定された範囲内の最大要素のインデックスを取得します。
virtual MaxValue	指定された範囲内の最大要素の値とインデックスを取得します。

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000–2025, MetaQuotes Ltd.

BufferResize

新しいバッファサイズを設定します。

```
virtual void  BufferResize(
   const int  size      // 新しいサイズ
  )
```

パラメータ

size

[in] 新しいバッファサイズ

© 2000–2025, MetaQuotes Ltd.

GetData

指定されたバッファ要素をインデックスで取得します。

```
double  GetData(
   const int  index      // インデックス
   ) const
```

パラメータ

index

[in] バッファ要素のインデックス

戻り値

指定されたバッファ要素をインデックスまたは EMPTY_VALUE

© 2000-2025, MetaQuotes Ltd.

Refresh

時系列データを更新します。

```
virtual void  Refresh(
   const int  flags=OBJ_ALL_PERIODS      // 時間軸フラグ
  )
```

パラメータ

flags=OBJ_ALL_PERIODS

[in] 更新する時間軸（フラグ）

© 2000–2025, MetaQuotes Ltd.

MinIndex

指定された範囲内の最小要素のインデックスを取得します。

```
virtual int  MinIndex(
   const int  start,      // 開始インデックス
   const int  count       // スキャンする要素数
   ) const
```

パラメータ

start

[in] 開始インデックス

count

[in] 扱う要素の数

戻り値

指定された範囲内の最小要素のインデックス、またはエラーの場合は -1

© 2000-2025, MetaQuotes Ltd.

MinValue

指定された範囲内の最小要素の値とインデックスを取得します。

```
virtual double  MinValue(
   const int    start,      // 開始インデックス
   const int    count,      // スキャンする要素数
   int&         index       // インデックスを受け取る変数への参照
   ) const
```

パラメータ

start

[in] 開始インデックス

count

[in] 扱う要素の数

index

[out] 整数型の変数への参照

戻り値

指定されたバッファの指定された範囲内の最大要素の値、またはエラーの場合は EMPTY_VALUE

注意事項

最小バッファ要素のインデックスは index 変数に格納されます。

© 2000–2025, MetaQuotes Ltd.

MaxIndex

指定された範囲内の最大要素のインデックスを取得します。

```
virtual int  MaxIndex(
   const int  start,      // 開始インデックス
   const int  count       // スキャンする要素数
   ) const
```

パラメータ

start

[in] 開始インデックス

count

[in] 扱う要素の数

戻り値

指定された範囲内の最大要素のインデックス、またはエラーの場合は -1

© 2000-2025, MetaQuotes Ltd.

MaxValue

指定された範囲内の最大要素の値とインデックスを取得します。

```
virtual double  MaxValue(
   const int    start,      // 開始インデックス
   const int    count,      // スキャンする要素数
   int&         index       // インデックスを受け取る変数への参照
   ) const
```

パラメータ

start

[in] 開始インデックス

count

[in] 扱う要素の数

index

[out] 整数型の変数への参照

戻り値

指定されたバッファの指定された範囲内の最大要素の値、またはエラーの場合は EMPTY_VALUE

注意事項

最大バッファ要素のインデックスは index 変数に格納されます。

© 2000–2025, MetaQuotes Ltd.

CIndicator

CIndicator は MQL5 標準ライブラリに含まれている全てのテクニカル指標の基本クラスです。

説明

CIndicator クラスは派生クラスの MQL5 API テクニカル指標関数への容易化されたアクセスを提供します。

宣言

```
class CIndicator: public CSeries
```

タイトル

```
#include <Indicators\Indicator.mqh>
```

継承階層

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator

直接子孫

CiAC, CiAD, CiADX, CiADXWilder, CiAlligator, CiAMA, CiAO, CiATR, CiBands, CiBearsPower, CiBullsPower, CiBWMFI, CiCCI, CiChaikin, CiCustom, CiDEMA, CiDeMarker, CiEnvelopes, CiForce, CiFractals, CiFrAMA, CiGator, CiIchimoku, CiMA, CiMACD, CiMFI, CiMomentum, CiOBV, CiOsMA, CiRSI, CiRVI, CiSAR, CiStdDev, CiStochastic, CiTEMA, CiTriX, CiVIDyA, CiVolumes, CiWPR

クラスメソッド

属性	
Handle	指標ハンドルを取得します。
Status	指標の状態を取得します。
FullRelease	値指標のハンドルを解放するフラグを設定します。
作成	
Create	指標を作成します。
BufferResize	新しいバッファサイズを設定します。
データアクセスメソッド	
GetData	指標バッファからデータを複製します。
データ更新メソッド	
Refresh	指標のデータを更新します。

© 2000-2025, MetaQuotes Ltd.

属性	
最安値と最高値の検索	
Minimum	配列の指定された範囲内の最小要素のインデックスを取得します。
MinValue	配列の指定された範囲内の最小要素の値とインデックスを取得します。
Maximum	配列の指定された範囲内の最大要素のインデックスを取得します。
MaxValue	配列の指定された範囲内の最大要素の値とインデックスを取得します。
列挙の変換	
MethodDescription	ENUM_MA_METHOD 列挙の値を文字列として取得します。
PriceDescription	ENUM_APPLIED_PRICE 列挙の値を文字列として取得します。
VolumeDescription	ENUM_APPLIED_VOLUME 列挙の値を文字列として取得します。
チャートの操作	
AddToChart	チャートに指標を追加します。
DeleteFromChart	チャートから指標を削除します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

派生クラス

- CiAC
- CiAD
- CiADX
- CiADXWilder
- CiAlligator
- CiAMA

© 2000–2025, MetaQuotes Ltd.

- CiAO
- CiATR
- CiBands
- CiBearsPower
- CiBullsPower
- CiBWMFI
- CiCCI
- CiChaikin
- CiDEMA
-) CiDeMarker
- CiEnvelopes
- CiForce
- CiFractals
- CiFrAMA
- CiGator
- CiIchimoku
- CiMA
- CiMACD
- CiMFI
- CiMomentum
- CiOBV
- CiOsMA
- CiRSI
- CiRVI
- CiSAR
- CiStdDev
- CiStochastic
- CiTEMA
- CiTriX
- CiVIDyA
- CiVolumes
- CiWPR

© 2000–2025, MetaQuotes Ltd.

Handle

指標ハンドルを取得します。

```
int  Handle() const
```

戻り値

指標ハンドル

© 2000–2025, MetaQuotes Ltd.

Status

指標の状態を取得します。

```
string  Status() const
```

戻り値

指標作成の状態

© 2000–2025, MetaQuotes Ltd.

FullRelease

値指標のハンドルを解放するフラグを設定します。

```
void  FullRelease(
   const bool  flag=true      // フラグ
  )
```

パラメータ

flag

[in] ハンドル解放フラグの新しい値

© 2000–2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。

```
bool  Create(
   const string           symbol,          // シンボル
   const ENUM_TIMEFRAMES  period,          // 期間
   const ENUM_INDICATOR   type,            // 型
   const int              num_params,      // パラメータ数
   const MqlParam&        params[]         // パラメータ配列への参照
  )
```

パラメータ

symbol

[in] 銘柄名

period

[in] 期間（ENUM_TIMEFRAMES 配列）

型

[in] 指標の種類（ENUM_INDICATOR 列挙）

num_params

[in] 指標パラメータ数

params

[in] 指標パラメータ配列への参照

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

BufferResize

指標バッファサイズを設定します。

```
virtual bool  BufferResize(
   const int  size      // サイズ
  )
```

パラメータ

size

[in] 新しいバッファサイズ

戻り値

成功の場合は true、それ以外の場合は false

注意事項

全ての指標バッファのサイズは同じです。

© 2000–2025, MetaQuotes Ltd.

BarsCalculated

指標の計算されたデータ数を返します。

```
int  BarsCalculated() const;
```

戻り値

指標バッファで計算されたデータの量。エラーの場合（データはまだ計算されていない）-1 。

© 2000-2025, MetaQuotes Ltd.

GetData

指標の指定されたバッファから指定された要素を取得します。最新データの操作を保障するために、この関数が呼ばれる前に、Refresh() が呼び出されるべきです。

```
double  GetData(
   const int  buffer_num,     // バッファ番号
   const int  index           // 要素インデックス
   ) const
```

パラメータ

buffer_num

[in] バッファ番号

index

[in] 要素のインデックス

戻り値

成功の場合は要素の数値、エラーの場合は EMPTY_VALUE

GetData

指標バッファから開始ポジションと要素数によってデータを取得します。

```
int  GetData(
   const int      start_pos,      // ポジション
   const int      count,          // 必要な要素数
   const int      buffer_num,     // バッファ番号
   double&        buffer[]        // データを受け取る配列
   ) const
```

パラメータ

start_pos

[in] 指標バッファの開始位置

count

[in] 必要な要素数

buffer_num

[in] 指標バッファの番号

buffer

[in] データを受け取る配列への参照

戻り値

成功の場合は指定された指標バッファから受け取られた要素の数、エラーの場合は -1

GetData

指標バッファから開始時刻と要素数によってデータを取得します。

© 2000-2025, MetaQuotes Ltd.

```
int  GetData(
   const datetime  start_time,      // 開始時刻
   const int       count,           // 必要な要素数
   const int       buffer_num,      // バッファ番号
   double&         buffer[]         // データを受け取る配列
   ) const
```

パラメータ

start_time

[in] 開始時刻

count

[in] 必要な要素数

buffer_num

[in] 指標バッファの番号

buffer

[in] データを受け取る配列への参照

戻り値

成功の場合は指定された指標バッファから受け取られた要素の数、エラーの場合は -1

GetData

指標バッファから開始時刻、終了時刻と要素数によってデータを取得します。

```
int  GetData(
   const datetime  start_time,      // 開始時刻
   const datetime  stop_time,       // 終了時刻
   const int       buffer_num,      // バッファ番号
   double&         buffer[]         // データを受け取る配列
   ) const
```

パラメータ

start_time

[in] 開始時刻

stop_time

[in] 終了時刻

buffer_num

[in] 指標バッファの番号

buffer

[in] データを受け取る配列への参照

戻り値

成功の場合は指定された指標バッファから受け取られた要素の数、エラーの場合は -1

© 2000-2025, MetaQuotes Ltd.

© 2000–2025, MetaQuotes Ltd.

Refresh

指標のデータを更新します。GetData() の呼び出し前にこのメソッドを呼び出すことが推奨されます。

```
virtual void  Refresh(
   int  flags=OBJ_ALL_PERIODS      // フラグ
  )
```

パラメータ

flags=OBJ_ALL_PERIODS

[in] 時間軸更新フラグ

© 2000–2025, MetaQuotes Ltd.

Minimum

配列の指定された範囲内の最小要素のインデックスを返します。

```
int  Minimum(
   const int  buffer_num,      // バッファ番号
   const int  start,           // 開始インデックス
   const int  count            // 処理する要素数
   ) const
```

パラメータ

buffer_num

[in] 値を検索するバッファの番号

start

[in] 検索の開始インデックス

count

[in] 検索する要素数

戻り値

配列の指定された範囲内の最小要素のインデックス

© 2000-2025, MetaQuotes Ltd.

MinValue

指定されたバッファの指定された範囲内の最小要素の値とインデックスを取得します。

```
double  MinValue(
   const int    buffer_num,      // バッファ番号
   const int    start,           // 開始インデックス
   const int    count,           // 処理する要素数
   int&         index            // 参照
   ) const
```

パラメータ

buffer_num

[in] 値を検索するバッファの番号

start

[in] 開始インデックス

count

[in] 扱う要素の数

index

[out] 最小要素のインデックスを受け取る int 型の変数への参照

戻り値

指定されたバッファの指定された範囲内の最小要素の値

注意事項

最小バッファ要素のインデックスは参照で渡された index 変数に格納されます。

© 2000-2025, MetaQuotes Ltd.

Maximum

配列の指定された範囲内の最大要素のインデックスを返します。

```
int  Maximum(
   const int  buffer_num,      // バッファ番号
   const int  start,           // 開始インデックス
   const int  count            // 処理する要素数
   ) const
```

パラメータ

buffer_num

[in] 値を検索するバッファの番号

start

[in] 検索の開始インデックス

count

[in] 検索する要素数

戻り値

指定されたバッファの指定された範囲内の最大要素のインデックス

© 2000–2025, MetaQuotes Ltd.

MaxValue

配列の指定された範囲内の最大要素の値とインデックスを返します。

```
double  MaxValue(
   const int    buffer_num,      // バッファ番号
   const int    start,           // 開始インデックス
   const int    count,           // 処理する要素数
   int&         index            // 参照
   ) const
```

パラメータ

buffer_num

[in] 値を検索するバッファの番号

start

[in] 開始インデックス

count

[in] 扱う要素の数

index

[out] 最大要素のインデックスを受け取る int 型の変数への参照

戻り値

指定されたバッファの指定された範囲内の最大要素の値

注意事項

最大バッファ要素のインデックスは参照で渡された index 変数に格納されます。

© 2000-2025, MetaQuotes Ltd.

MethodDescription

この関数は ENUM_MA_METHOD 列挙の値を文字列として返します。

```
string  MethodDescription(
   const int  val      // 値
   ) const
```

パラメータ

val

[in] ENUM_MA_METHOD 列挙の値

戻り値

文字列としての ENUM_MA_METHOD 列挙値

© 2000–2025, MetaQuotes Ltd.

PriceDescription

この関数は ENUM_APPLIED_PRICE 列挙の値を文字列として返します。

```
string  PriceDescription(
   const int  val      // 値
   ) const
```

パラメータ

val

[in] ENUM_APPLIED_PRICE 列挙の値

戻り値

文字列としての ENUM_APPLIED_PRICE 列挙の値

© 2000–2025, MetaQuotes Ltd.

VolumeDescription

この関数は ENUM_APPLIED_VOLUME 列挙の値を文字列として返します。

```
string  VolumeDescription(
   const int  val      // 値
   ) const
```

パラメータ

val

[in] ENUM_APPLIED_VOLUME 列挙の値

戻り値

文字列としての ENUM_APPLIED_VOLUME 列挙の値

© 2000–2025, MetaQuotes Ltd.

AddToChart

チャートに指標を追加します。

```
bool  AddToChart(
   const long  chart,      // チャート識別子
   const int   subwin      // チャートサブウィンドウ
  )
```

パラメータ

chart

[in] チャート識別子。

subwin

[in] チャートサブウィンドウ

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

DeleteFromChart

チャートから指標を削除します。

```
bool  DeleteFromChart(
   const long  chart,      // チャート識別子
   const int   subwin      // チャートサブウィンドウ
  )
```

パラメータ

chart

[in] チャート識別子。

subwin

[in] チャートサブウィンドウ

戻り値

成功の場合は true、エラーが発生した場合は false

© 2000–2025, MetaQuotes Ltd.

CIndicators

CIndicators は時系列及びテクニカル指標クラスのインスタンスをまとめるクラスです。

説明

CIndicators クラスはテクニカル指標インスタンスの作成、格納及び管理（データの同期化、ハンドルとメモリの管理）を提供します。

宣言

```
class CIndicators: public CArrayObj
```

タイトル

```
#include <Indicators\Indicators.mqh>
```

クラスメソッド

作成メソッド	
Create	テクニカル指標を作成します。
データ更新メソッド	
Refresh	コレクションのテクニカル指標を更新します。

© 2000-2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。

```
CIndicator*  Create(
   const string            symbol,      // 銘柄名
   const ENUM_TIMEFRAMES   period,      // 期間
   const ENUM_INDICATOR    type,        // 指標の種類
   const int               count,       // パラメータ数
   const MqlParam&         params       // パラメータ配列への参照
  )
```

パラメータ

symbol

[in] 銘柄名

period

[in] 期間（ENUM_TIMEFRAMES 配列）

型

[in] 指標の種類（ENUM_INDICATOR）

count

[in] 指標のパラメータ数

params

[in] 指標パラメータ配列への参照

戻り値

成功の場合は作成された指標への参照、指標が作成できなかった場合は NULL

© 2000–2025, MetaQuotes Ltd.

Refresh

コレクション内の全てのテクニカル指標を更新します。

```
int  Refresh()
```

戻り値

更新された時間軸のフラグ（オブジェクト可視性フラグとして形成）

© 2000–2025, MetaQuotes Ltd.

時系列クラス

このセクションには、MQL5 標準ライブラリの時系列クラスの技術的な詳細と、その全ての主要コンポーネントの説明が含まれています。

クラス	説明
CiSpread	スプレッド履歴へのアクセスを提供します。
CiTime	バーの開始時刻履歴へのアクセスを提供します。
CiTickVolume	バーのティックボリューム履歴へのアクセスを提供します。
CiRealVolume	バーの出来高履歴へのアクセスを提供します。
CiOpen	バーの始値履歴へのアクセスを提供します。
CiHigh	足の高値履歴へのアクセスを提供します。
CiLow	バーの安値履歴へのアクセスを提供します。
CiClose	バーの終値履歴へのアクセスを提供します。

© 2000–2025, MetaQuotes Ltd.

CiSpread

CiSpread はバーのスプレッド履歴へのアクセスを意図されたクラスです。

説明

CiSpread クラスはスプレッド履歴へのアクセスを提供します。

宣言

```
class CiSpread: public CSeries
```

タイトル

```
#include <Indicators\TimeSeries.mqh>
```

継承階層

COBject
CArray
CArrayObj
CSeries
CiSpread

クラスメソッド

作成メソッド	
Create	時系列を作成します。
BufferResize	バッファサイズを設定します。
データアクセスメソッド	
GetData	データを取得します。
データ更新メソッド	
Refresh	データを更新します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000–2025, MetaQuotes Ltd.

Create

スプレッド履歴にアクセスするために指定されたパラメータを持つ時系列を作成します。

```
bool  Create(
   string           symbol,      // シンボル
   ENUM_TIMEFRAMES  period       // 期間
  )
```

パラメータ

symbol

[in] シンボル

period

[in] Timeframe (ENUM_TIMEFRAMES enumeration).

戻り値

成功の場合は true、時系列が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

BufferResize

シリーズの新しいサイズを設定します。

```
virtual void  BufferResize(
   int  size      // 新しいサイズ
  )
```

パラメータ

size

[in] 新しいバッファサイズ

© 2000–2025, MetaQuotes Ltd.

GetData

インデックスによって時系列の要素を取得します。

```
int  GetData(
   int  index      // インデックス
   ) const
```

パラメータ

index

[in] 必要な要素のインデックス

戻り値

時系列バッファの要素、または 0

GetData

開始位置と要素数によって時系列から要素を取得します。

```
int  GetData(
   int   start_pos,     // 開始位置
   int   count,         // 取得する要素
   int&  buffer         // データを受け取る配列
   ) const
```

パラメータ

start_pos

[in] 時系列の開始位置

count

[in] 必要な要素数

buffer

[in] データを受け取る配列への参照

戻り値

成功した場合 >=0、エラーの場合 -1

GetData

開始時刻と要素数によって時系列から要素を取得します。

```
int  GetData(
   datetime  start_time,     // 開始時刻
   int       count,          // 要素数
   int&      buffer          // データを受け取る配列
   ) const
```

パラメータ

start_time

© 2000-2025, MetaQuotes Ltd.

[in] 開始時刻

count

[in] 必要な要素数

buffer

[in] データを受け取る配列への参照

戻り値

成功した場合 >=0、エラーの場合 -1

GetData

開始時刻と終了時刻によって時系列から要素を取得します。

```
int  GetData(
   datetime  start_time,     // 開始時刻
   datetime  stop_time,      // 終了時刻
   int&      buffer          // データを受け取る配列
   ) const
```

パラメータ

start_time

[in] 開始時刻

stop_time

[in] 終了時刻

buffer

[in] データを受け取る配列への参照

戻り値

成功した場合 >=0、エラーの場合 -1

© 2000-2025, MetaQuotes Ltd.

Refresh

時系列のデータを更新します。

```
virtual void  Refresh(
   int  flags      // フラグ
  )
```

パラメータ

flags

[in] 時間軸フラグ

© 2000-2025, MetaQuotes Ltd.

CiTime

CiTime はバーの開始時刻履歴へのアクセスに意図されたクラスです。

説明

CTimeBuffer クラスはバーの開始時刻履歴への簡易化されたアクセスを提供します。

宣言

```
class CiTime: public CSeries
```

タイトル

```
#include <Indicators\TimeSeries.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CiTime

クラスメソッド

作成メソッド	
Create	時系列を作成します。
BufferResize	バッファサイズを設定します。
データアクセスメソッド	
GetData	データを取得します。
データ更新メソッド	
Refresh	データを更新します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000–2025, MetaQuotes Ltd.

Create

履歴内のバーの開始時刻にアクセスするために指定されたパラメータを持つ時系列を作成します。

```
bool  Create(
   string           symbol,      // シンボル
   ENUM_TIMEFRAMES  period       // 期間
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

戻り値

成功の場合は true、時系列が作成できなかった場合は false

© 2000-2025, MetaQuotes Ltd.

BufferResize

シリーズの新しいサイズを設定します。

```
virtual void  BufferResize(
   int  size      // 新しいサイズ
  )
```

パラメータ

size

[in] 新しいバッファサイズ

© 2000-2025, MetaQuotes Ltd.

GetData

インデックスによって時系列の要素を取得します。

```
datetime  GetData(
   int  index      // インデックス
   ) const
```

パラメータ

index

[in] 必要な要素のインデックス

戻り値

時系列バッファの要素、または 0

GetData

開始位置と要素数によって時系列から要素を取得します。

```
int  GetData(
   int   start_pos,     // 開始位置
   int   count,         // 取得する要素
   long& buffer         // データを受け取る配列
   ) const
```

パラメータ

start_pos

[in] 時系列の開始位置

count

[in] 必要な要素数

buffer

[in] データを受け取る配列への参照

戻り値

成功した場合 >=0、エラーの場合 -1

GetData

開始時刻と要素数によって時系列から要素を取得します。

```
int  GetData(
   datetime  start_time,     // 開始時刻
   int       count,          // 要素数
   long&     buffer          // データを受け取る配列
   ) const
```

パラメータ

start_time

© 2000-2025, MetaQuotes Ltd.

[in] 開始時刻

count

[in] 必要な要素数

buffer

[in] データを受け取る配列への参照

戻り値

成功した場合 >=0、エラーの場合 -1

GetData

開始時刻と終了時刻によって時系列から要素を取得します。

```
int  GetData(
   datetime  start_time,     // 開始時刻
   datetime  stop_time,      // 終了時刻
   long&     buffer          // データを受け取る配列
   ) const
```

パラメータ

start_time

[in] 開始時刻

stop_time

[in] 終了時刻

buffer

[in] データを受け取る配列への参照

戻り値

成功した場合 >=0、エラーの場合 -1

© 2000-2025, MetaQuotes Ltd.

Refresh

時系列のデータを更新します。

```
virtual void  Refresh(
   int  flags      // フラグ
  )
```

パラメータ

flags

[in] 時間軸フラグ

© 2000–2025, MetaQuotes Ltd.

CiTickVolume

CiTickVolume はバーのティックボリューム履歴へのアクセスに意図されたクラスです。

説明

CiTickVolume クラスはバーのティックボリューム履歴へのアクセスを提供します。

宣言

```
class CiTickVolume: public CSeries
```

タイトル

```
#include <Indicators\TimeSeries.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CiTickVolume

クラスメソッド

作成メソッド	
Create	時系列を作成します。
BufferResize	バッファサイズを設定します。
データアクセスメソッド	
GetData	データを取得します。
データ更新メソッド	
Refresh	データを更新します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000–2025, MetaQuotes Ltd.

Create

履歴内のバーのティックボリュームにアクセスするために指定されたパラメータを持つ時系列を作成します。

```
bool  Create(
   string           symbol,      // シンボル
   ENUM_TIMEFRAMES  period       // 期間
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

戻り値

成功の場合は true、時系列が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

BufferResize

シリーズの新しいサイズを設定します。

```
virtual void  BufferResize(
   int  size      // 新しいサイズ
  )
```

パラメータ

size

[in] 新しいバッファサイズ

© 2000–2025, MetaQuotes Ltd.

GetData

インデックスによって時系列の要素を取得します。

```
long  GetData(
   int  index      // インデックス
   ) const
```

パラメータ

index

[in] 必要な要素のインデックス

戻り値

時系列バッファの要素、または 0

GetData

開始位置と要素数によって時系列から要素を取得します。

```
int  GetData(
   int   start_pos,     // 開始位置
   int   count,         // 取得する要素
   long& buffer         // データを受け取る配列
   ) const
```

パラメータ

start_pos

[in] 時系列の開始位置

count

[in] 必要な要素数

buffer

[in] データを受け取る配列への参照

戻り値

成功した場合 >=0、エラーの場合 -1

GetData

開始時刻と要素数によって時系列から要素を取得します。

```
int  GetData(
   datetime  start_time,     // 開始時刻
   int       count,          // 要素数
   long&     buffer          // データを受け取る配列
   ) const
```

パラメータ

start_time

© 2000-2025, MetaQuotes Ltd.

[in] 開始時刻

count

[in] 必要な要素数

buffer

[in] データを受け取る配列への参照

戻り値

成功した場合 >=0、エラーの場合 -1

GetData

開始時刻と終了時刻によって時系列から要素を取得します。

```
int  GetData(
   datetime  start_time,     // 開始時刻
   datetime  stop_time,      // 終了時刻
   long&     buffer          // データを受け取る配列
   ) const
```

パラメータ

start_time

[in] 開始時刻

stop_time

[in] 終了時刻

buffer

[in] データを受け取る配列への参照

戻り値

成功した場合 >=0、エラーの場合 -1

© 2000-2025, MetaQuotes Ltd.

Refresh

時系列のデータを更新します。

```
virtual void  Refresh(
   int  flags      // フラグ
  )
```

パラメータ

flags

[in] 時間軸フラグ

© 2000-2025, MetaQuotes Ltd.

CiRealVolume

CiRealVolume はバーの出来高履歴へのアクセスに意図されたクラスです。

説明

CiRealVolume クラスはバーの出来高履歴へのアクセスを提供します。

宣言

```
class CiRealVolume: public CSeries
```

タイトル

```
#include <Indicators\TimeSeries.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CiRealVolume

クラスメソッド

作成メソッド	
Create	時系列を作成します。
BufferResize	バッファサイズを設定します。
データアクセスメソッド	
GetData	データを取得します。
データ更新メソッド	
Refresh	データを更新します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000–2025, MetaQuotes Ltd.

Create

履歴内のバーの出来高にアクセスするために指定されたパラメータを持つ時系列を作成します。

```
bool  Create(
   string           symbol,     // シンボル
   ENUM_TIMEFRAMES  period      // 期間
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

戻り値

成功の場合は true、時系列が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

BufferResize

シリーズの新しいサイズを設定します。

```
virtual void  BufferResize(
   int  size      // 新しいサイズ
  )
```

パラメータ

size

[in] 新しいバッファサイズ

© 2000-2025, MetaQuotes Ltd.

GetData

インデックスによって時系列の要素を取得します。

```
datetime  GetData(
   int  index      // インデックス
   ) const
```

パラメータ

index

[in] 必要な要素のインデックス

戻り値

時系列バッファの要素、または 0

GetData

開始位置と要素数によって時系列から要素を取得します。

```
int  GetData(
   int   start_pos,     // 開始位置
   int   count,         // 取得する要素
   long& buffer         // データを受け取る配列
   ) const
```

パラメータ

start_pos

[in] 時系列の開始位置

count

[in] 必要な要素数

buffer

[in] データを受け取る配列への参照

戻り値

成功した場合 >=0、エラーの場合 -1

GetData

開始時刻と要素数によって時系列から要素を取得します。

```
int  GetData(
   datetime  start_time,     // 開始時刻
   int       count,          // 要素数
   long&     buffer          // データを受け取る配列
   ) const
```

パラメータ

start_time

© 2000-2025, MetaQuotes Ltd.

[in] 開始時刻

count

[in] 必要な要素数

buffer

[in] データを受け取る配列への参照

戻り値

成功した場合 >=0、エラーの場合 -1

GetData

開始時刻と終了時刻によって時系列から要素を取得します。

```
int  GetData(
   datetime  start_time,     // 開始時刻
   datetime  stop_time,      // 終了時刻
   long&     buffer          // データを受け取る配列
   ) const
```

パラメータ

start_time

[in] 開始時刻

stop_time

[in] 終了時刻

buffer

[in] データを受け取る配列への参照

戻り値

成功した場合 >=0、エラーの場合 -1

© 2000-2025, MetaQuotes Ltd.

Refresh

時系列のデータを更新します。

```
virtual void  Refresh(
   int  flags      // フラグ
  )
```

パラメータ

flags

[in] 時間軸フラグ

© 2000-2025, MetaQuotes Ltd.

CiOpen

CiOpen はバーの始値履歴への簡易化されたアクセスに意図されたクラスです。

説明

CiOpen クラスはバーの始値履歴へのアクセスを提供します。

宣言

```
class CiOpen: public CPriceSeries
```

タイトル

```
#include <Indicators\TimeSeries.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CPriceSeries
CiOpen

クラスメソッド

作成メソッド	
Create	時系列を作成します。
データアクセスメソッド	
GetData	データを取得します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CPriceSeries

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

© 2000–2025, MetaQuotes Ltd.

Create

履歴内のバーの始値にアクセスするために指定されたパラメータを持つ時系列を作成します。

```
bool  Create(
   string           symbol,      // シンボル
   ENUM_TIMEFRAMES  period       // 期間
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

戻り値

成功の場合は true、時系列が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

GetData

開始位置と要素数によって時系列から要素を取得します。

```
int  GetData(
   int    start_pos,      // 開始位置
   int    count,          // 取得する要素
   double& buffer         // データを受け取る配列
   ) const
```

パラメータ

start_pos

[in] 時系列の開始位置

count

[in] 必要な要素数

buffer

[in] データを受け取る配列への参照

戻り値

成功した場合 >=0 、エラーの場合 -1

GetData

開始時刻と要素数によって時系列から要素を取得します。

```
int  GetData(
   datetime  start_time,      // 開始時刻
   int       count,           // 要素数
   double&     buffer         // データを受け取る配列
   ) const
```

パラメータ

start_time

[in] 開始時刻

count

[in] 必要な要素数

buffer

[in] データを受け取る配列への参照

戻り値

成功した場合 >=0 、エラーの場合 -1

GetData

開始時刻と終了時刻によって時系列から要素を取得します。

© 2000-2025, MetaQuotes Ltd.

```
int  GetData(
   datetime  start_time,      // 開始時刻
   datetime  stop_time,       // 終了時刻
   double&   buffer           // データを受け取る配列
   ) const
```

パラメータ

start_time

[in] 開始時刻

stop_time

[in] 終了時刻

buffer

[in] データを受け取る配列への参照

戻り値

成功した場合 >=0、エラーの場合 -1

© 2000-2025, MetaQuotes Ltd.

CiHigh

CiHigh は足の高値履歴への簡易化されたアクセスに意図されたクラスです。

説明

CiHigh クラスは足の高値履歴へのアクセスを提供します。

宣言

```
class CiHigh: public CPriceSeries
```

タイトル

```
#include <Indicators\TimeSeries.mqh>
```

継承階層

CObject
  CArray
    CArrayObj
      CSeries
        CPriceSeries
          CiHigh

クラスメソッド

作成メソッド	
Create	時系列を作成します。
データアクセスメソッド	
GetData	データを取得します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CPriceSeries

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

© 2000–2025, MetaQuotes Ltd.

Create

履歴内の足の高値にアクセスするために指定されたパラメータを持つ時系列を作成します。

```
bool  Create(
   string           symbol,     // シンボル
   ENUM_TIMEFRAMES  period      // 期間
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

戻り値

成功の場合は true、時系列が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

GetData

開始位置と要素数によって時系列から要素を取得します。

```
int  GetData(
   int    start_pos,      // 開始位置
   int    count,          // 取得する要素
   double& buffer         // データを受け取る配列
   ) const
```

パラメータ

start_pos

[in] 時系列の開始位置

count

[in] 必要な要素数

buffer

[in] データを受け取る配列への参照

戻り値

成功した場合 >=0 、エラーの場合 -1

GetData

開始時刻と要素数によって時系列から要素を取得します。

```
int  GetData(
   datetime  start_time,      // 開始時刻
   int       count,           // 要素数
   double&     buffer         // データを受け取る配列
   ) const
```

パラメータ

start_time

[in] 開始時刻

count

[in] 必要な要素数

buffer

[in] データを受け取る配列への参照

戻り値

成功した場合 >=0 、エラーの場合 -1

GetData

開始時刻と終了時刻によって時系列から要素を取得します。

© 2000-2025, MetaQuotes Ltd.

```
int  GetData(
   datetime  start_time,      // 開始時刻
   datetime  stop_time,       // 終了時刻
   double&   buffer           // データを受け取る配列
   ) const
```

パラメータ

start_time

[in] 開始時刻

stop_time

[in] 終了時刻

buffer

[in] データを受け取る配列への参照

戻り値

成功した場合 >=0、エラーの場合 -1

© 2000-2025, MetaQuotes Ltd.

CiLow

CiLow はバーの安値履歴への簡易化されたアクセスに意図されたクラスです。

説明

CiLow クラスはバーの安値履歴へのアクセスを提供します。

宣言

```
class CiLow: public CPriceSeries
```

タイトル

```
#include <Indicators\TimeSeries.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CPriceSeries
CiLow

クラスメソッド

作成メソッド	
Create	時系列を作成します。
データアクセスメソッド	
GetData	データを取得します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CPriceSeries

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

© 2000–2025, MetaQuotes Ltd.

Create

履歴内のバーの安値にアクセスするために指定されたパラメータを持つ時系列を作成します。

```
bool  Create(
   string           symbol,     // シンボル
   ENUM_TIMEFRAMES  period      // 期間
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

戻り値

成功の場合は true、時系列が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

GetData

開始位置と要素数によって時系列から要素を取得します。

```
int  GetData(
   int    start_pos,      // 開始位置
   int    count,          // 取得する要素
   double& buffer         // データを受け取る配列
   ) const
```

パラメータ

start_pos

[in] 時系列の開始位置

count

[in] 必要な要素数

buffer

[in] データを受け取る配列への参照

戻り値

成功した場合 >=0 、エラーの場合 -1

GetData

開始時刻と要素数によって時系列から要素を取得します。

```
int  GetData(
   datetime  start_time,      // 開始時刻
   int       count,           // 要素数
   double&     buffer         // データを受け取る配列
   ) const
```

パラメータ

start_time

[in] 開始時刻

count

[in] 必要な要素数

buffer

[in] データを受け取る配列への参照

戻り値

成功した場合 >=0 、エラーの場合 -1

GetData

開始時刻と終了時刻によって時系列から要素を取得します。

© 2000-2025, MetaQuotes Ltd.

```
int  GetData(
   datetime  start_time,      // 開始時刻
   datetime  stop_time,       // 終了時刻
   double&   buffer           // データを受け取る配列
   ) const
```

パラメータ

start_time

[in] 開始時刻

stop_time

[in] 終了時刻

buffer

[in] データを受け取る配列への参照

戻り値

成功した場合 >=0、エラーの場合 -1

© 2000-2025, MetaQuotes Ltd.

CiClose

CiCloseBuffer はバーの終値履歴への簡易化されたアクセスに意図されたクラスです。

説明

CiCloseBuffer クラスはバーの終値履歴への簡易化されたアクセスを提供します。

宣言

```
class CiClose: public CPriceSeries
```

タイトル

```
#include <Indicators\TimeSeries.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CPriceSeries
CiClose

クラスメソッド

作成メソッド	
Create	時系列を作成します。
データアクセスメソッド	
GetData	データを取得します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CPriceSeries

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

© 2000–2025, MetaQuotes Ltd.

Create

履歴内のバーの終値にアクセスするために指定されたパラメータを持つ時系列を作成します。

```
bool  Create(
   string           symbol,     // シンボル
   ENUM_TIMEFRAMES  period      // 期間
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

戻り値

成功の場合は true、時系列が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

GetData

開始位置と要素数によって時系列から要素を取得します。

```
int  GetData(
   int    start_pos,      // 開始位置
   int    count,          // 取得する要素
   double& buffer         // データを受け取る配列
   ) const
```

パラメータ

start_pos

[in] 時系列の開始位置

count

[in] 必要な要素数

buffer

[in] データを受け取る配列への参照

戻り値

成功した場合 >=0 、エラーの場合 -1

GetData

開始時刻と要素数によって時系列から要素を取得します。

```
int  GetData(
   datetime  start_time,      // 開始時刻
   int       count,           // 要素数
   double&     buffer         // データを受け取る配列
   ) const
```

パラメータ

start_time

[in] 開始時刻

count

[in] 必要な要素数

buffer

[in] データを受け取る配列への参照

戻り値

成功した場合 >=0 、エラーの場合 -1

GetData

開始時刻と終了時刻によって時系列から要素を取得します。

© 2000-2025, MetaQuotes Ltd.

```
int  GetData(
   datetime  start_time,      // 開始時刻
   datetime  stop_time,       // 終了時刻
   double&   buffer           // データを受け取る配列
   ) const
```

パラメータ

start_time

[in] 開始時刻

stop_time

[in] 終了時刻

buffer

[in] データを受け取る配列への参照

戻り値

成功した場合 >=0、エラーの場合 -1

© 2000-2025, MetaQuotes Ltd.

トレンド指標クラス

このセクションには、MQL5 標準ライブラリのトレンド指標クラスの技術的な詳細と、その全ての主要コンポーネントの説明が含まれています。

クラス/グループ	説明
CiADX	平均方向性指数（Average Directional Index）。
CiADXWilder	ウェルズ·ワイルダーの平均方向性指数（Average Directional Index by Welles Wilder）
CiBands	ボリンジャーバンド®（Bollinger Bands®）。
CiEnvelopes	エンベロープ（Envelopes）。
CiIchimoku	一目均衡表（Ichimoku Kinko Hyo）。
CiMA	移動平均（Moving Average）。
CiSAR	パラボリック停止・リバースシステム（Parabolic Stop And Reverse System）
CiStdDev	標準偏差（Standard Deviation）。
CiDEMA	2重指数移動平均（Double Exponential Moving Average）。
CiTEMA	3重指数移動平均（Triple Exponential Moving Average）。
CiFrAMA	フラクタル適応型移動平均（Fractal Adaptive Moving Average）。
CiAMA	適応型移動平均（Adaptive Moving Average）。
CiVIDyA	可変インデックス動的平均（Variable Index Dynamic Average）。

© 2000–2025, MetaQuotes Ltd.

CiADX

CiADX は平均方向性指数（Average Directional Index）テクニカル指標の使用に意図されたクラスです。

説明

CiADX は、平均方向性指数指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiADX: public CIndicator
```

タイトル

```
#include <Indicators\Trend.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CIndicator
CiADX

クラスメソッド

属性	
MaPeriod	平均期間を返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Main	中正線のバッファ要素を返します。
Plus	+DI 線のバッファ要素を返します。
Minus	-DI 線のバッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

MaPeriod

平均期間を返します。

```
int  MaPeriod() const
```

戻り値

指標の作成時に定義された平均期間

© 2000-2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string           symbol,         // シンボル
   ENUM_TIMEFRAMES  period,         // 期間
   int              ma_period       // 平均期間
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

ma_period

[in] 平均期間

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000-2025, MetaQuotes Ltd.

Plus

指定されたインデックスを持つ +DI 線のバッファ要素を返します。

```
double   Plus(
   int   index       // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

指定されたインデックスを持つ +DI 線のバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000-2025, MetaQuotes Ltd.

Minus

指定されたインデックスを持つ -DI 線のバッファ要素を返します。

```
double  Minus(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

The buffer element of the -DI line of the specified index, or EMPTY_VALUE if there isn't any correct data.

© 2000-2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

オブジェクトの型の識別子（IND_ADX は CiADX）

© 2000–2025, MetaQuotes Ltd.

CiADXWilder

CiADXWilder はウェルズ·ワイルダーの ADX テクニカル指標（Average Directional Index by Welles Wilder）の使用に意図されたクラスです。

説明

CiADXWilder は、ウェルズ·ワイルダーの ADX 指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiADXWilder: public CIndicator
```

タイトル

```
#include <Indicators\Trend.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CIndicator
CiADXWilder

クラスメソッド

属性	
MaPeriod	平均期間を返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Main	中正線のバッファ要素を返します。
Plus	+DI 線のバッファ要素を返します。
Minus	-DI 線のバッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

© 2000-2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

MaPeriod

平均期間を返します。

```
int  MaPeriod() const
```

戻り値

指標の作成時に定義された平均期間

© 2000-2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string            symbol,         // シンボル
   ENUM_TIMEFRAMES   period,         // 期間
   int               ma_period       // 平均期間
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

ma_period

[in] 平均期間

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000-2025, MetaQuotes Ltd.

Plus

指定されたインデックスを持つ +DI 線のバッファ要素を返します。

```
double   Plus(
   int   index       // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

指定されたインデックスを持つ +DI 線のバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000-2025, MetaQuotes Ltd.

Minus

指定されたインデックスを持つ -DI 線のバッファ要素を返します。

```
double  Minus(
   int  index       // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

指定されたインデックスを持つ -DI 線のバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000-2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

オブジェクトの型の識別子（IND_ADXW は CiADXWilder）

© 2000–2025, MetaQuotes Ltd.

CiBands

CiBands はボリンジャーバンド®（Bollinger Bands®）テクニカル指標の使用に意図されたクラスです。

説明

CiBands クラスは、ボリンジャーバンド指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiBands: public CIndicator
```

タイトル

```
#include <Indicators\Trend.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CIndicator
CiBands

クラスメソッド

属性	
MaPeriod	平均期間を返します。
MaShift	水平方向のシフトを返します。
Deviation	偏差値を返します。
Applied	適用する型またはハンドルを返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Base	基本線のバッファ要素を返します。
Upper	上部線のバッファ要素を返します。
Lower	下部線のバッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

MaPeriod

平均期間を返します。

```
int  MaPeriod() const
```

戻り値

指標の作成時に定義された平均期間

© 2000-2025, MetaQuotes Ltd.

MaShift

指標の水平方向のシフトを返します。

```
int  MaShift() const
```

戻り値

指標の作成時に定義された水平方向のシフトの値

© 2000-2025, MetaQuotes Ltd.

Deviation

偏差値を返します。

```
double  Deviation() const
```

戻り値

指標の作成時に定義された偏差値

© 2000–2025, MetaQuotes Ltd.

Applied

適用する型またはハンドルを返します。

```
int  Applied() const
```

戻り値

指標の作成時に定義された、適用する型またはハンドル

© 2000-2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string            symbol,        // シンボル
   ENUM_TIMEFRAMES   period,        // 期間
   int               ma_period,     // 平均期間
   int               ma_shift,      // シフト
   double            deviation,     // 偏差値
   int               applied        // 適用された価格またはハンドル
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

ma_period

[in] 平均期間

ma_shift

[in] 指標の水平方向のシフト

偏差

[in] 偏差値

applied

[in] 適用するボリュームの種類

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Base

指定されたインデックスを持つ基本線のバッファ要素を返します。

```
double  Base(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

指定されたインデックスを持つ基本線のバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000-2025, MetaQuotes Ltd.

Upper

指定されたインデックスを持つ上部線のバッファ要素を返します。

```
double  Upper(
   int  index       // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

指定されたインデックスを持つ上部線のバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000-2025, MetaQuotes Ltd.

Lower

指定されたインデックスを持つ下部線のバッファ要素を返します。

```
double  Lower(
   int  index       // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

指定されたインデックスを持つ下部線のバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000-2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

オブジェクトの型の識別子（IND_BANDS は CiBands）

© 2000-2025, MetaQuotes Ltd.

CiEnvelopes

CiEnvelopes はエンベロープ（Envelopes）テクニカル指標の使用に意図されたクラスです。

説明

CiEnvelopes クラスは、エンベロープ指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiEnvelopes: public CIndicator
```

タイトル

```
#include <Indicators\Trend.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CIndicator
CiEnvelopes

クラスメソッド

属性	
MaPeriod	平均期間を返します。
MaShift	水平方向のシフトを返します。
MaMethod	平均化の方法を返します。
Deviation	偏差値を返します。
Applied	適用する型またはハンドルを返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Upper	上部線のバッファ要素を返します。
Lower	下部線のバッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

© 2000-2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

MaPeriod

平均期間を返します。

```
int  MaPeriod() const
```

戻り値

指標の作成時に定義された平均期間

© 2000-2025, MetaQuotes Ltd.

MaShift

指標の水平方向のシフトを返します。

```
int  MaShift() const
```

戻り値

指標の作成時に定義された水平方向のシフトの値

© 2000-2025, MetaQuotes Ltd.

MaMethod

平均化の方法を返します。

```
ENUM_MA_METHOD  MaMethod() const
```

戻り値

指標の作成時に定義された平均化の方法

© 2000-2025, MetaQuotes Ltd.

Deviation

偏差値を返します。

```
double  Deviation() const
```

戻り値

指標の作成時に定義された偏差値

© 2000–2025, MetaQuotes Ltd.

Applied

適用する型またはハンドルを返します。

```
int  Applied() const
```

戻り値

指標の作成時に定義された、適用する型またはハンドル

© 2000-2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string            symbol,        // シンボル
   ENUM_TIMEFRAMES   period,        // 期間
   int               ma_period,     // 平均期間
   int               ma_shift,      // 水平シフト
   ENUM_MA_METHOD    ma_method,     // 平均化の方法
   int               applied,       // 適用する価格型またはハンドル
   double            deviation      // 偏差値
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

ma_period

[in] 平均期間

ma_shift

[in] 水平方向のシフト

ma_method

[in] 平均化の方法（ENUM_MA_METHOD 列挙）

applied

[in] 適用する価格型またはハンドル

偏差

[in] 偏差値

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Upper

指定されたインデックスを持つ上部線のバッファ要素を返します。

```
double  Upper(
   int  index       // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

指定されたインデックスを持つ上部線のバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000-2025, MetaQuotes Ltd.

Lower

指定されたインデックスを持つ下部線のバッファ要素を返します。

```
double  Lower(
   int  index       // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

指定されたインデックスを持つ下部線のバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000–2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

オブジェクトの型の識別子（IND_ENVELOPES は CiEnvelopes）

© 2000–2025, MetaQuotes Ltd.

CiIchimoku

CiIchimoku は一目均衡表テクニカル指標の使用に意図されたクラスです。

説明

CiIchimoku クラスは、一目均衡表指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiIchimoku: public CIndicator
```

タイトル

```
#include <Indicators\Trend.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CIndicator
CiIchimoku

クラスメソッド

属性	
TenkanSenPeriod	転換線の期間を返します。
KijunSenPeriod	基準線の期間を返します。
SenkouSpanBPeriod	先行スパンBの期間を返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
TenkanSen	転換線のバッファ要素を返します。
KijunSen	基準線のバッファ要素を返します。
SenkouSpanA	先行スパンA線のバッファ要素を返します。
SenkouSpanB	先行スパンB線のバッファ要素を返します。
ChinkouSpan	遅行スパン線のバッファ要素を返します。
入出力	

© 2000-2025, MetaQuotes Ltd.

属性	
virtual Type	オブジェクトの型の識別子を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

TenkanSenPeriod

転換線の期間を返します。

```
int  TenkanSenPeriod() const
```

戻り値

指標の作成時に定義された転換線の期間

© 2000-2025, MetaQuotes Ltd.

KijunSenPeriod

基準線の期間を返します。

```
int  KijunSenPeriod() const
```

戻り値

指標の作成時に定義された基準線の期間

© 2000-2025, MetaQuotes Ltd.

SenkouSpanBPeriod

先行スパンBの期間を返します。

```
int  SenkouSpanBPeriod() const
```

戻り値

指標の作成時に定義された先行スパンBの期間

© 2000-2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string           symbol,           // シンボル
   ENUM_TIMEFRAMES  period,           // 期間
   int              tenkan_sen,       // 転換線の期間
   int              kijun_sen,        // 基準線の期間
   int              senkou_span_b     // 先行スパンBの期間
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

tenkan_sen

[in] 転換線の期間

kijun_sen

[in] 基準線の期間

senkou_span_b

[in] 先行スパンBの期間

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

TenkanSen

指定されたインデックスを持つ転換線のバッファ要素を返します。

```
double  TenkanSen(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

指定されたインデックスを持つ転換線のバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000-2025, MetaQuotes Ltd.

KijunSen

指定されたインデックスを持つ基準線のバッファ要素を返します。

```
double  KijunSen(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

指定されたインデックスを持つ基準線のバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000-2025, MetaQuotes Ltd.

SenkouSpanA

指定されたインデックスを持つ先行スパンA線のバッファ要素を返します。

```
double  SenkouSpanA(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

指定されたインデックスを持つ先行スパンA線のバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000-2025, MetaQuotes Ltd.

SenkouSpanB

指定されたインデックスを持つ先行スパンB線のバッファ要素を返します。

```
double  SenkouSpanB(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

指定されたインデックスを持つ先行スパン B 線のバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000-2025, MetaQuotes Ltd.

ChinkouSpan

指定されたインデックスを持つ遅行スパン線のバッファ要素を返します。

```
double  ChinkouSpan(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

指定されたインデックスを持つ遅行スパン線のバッファ要素、またはデータが不在な場合は EMPTY_VALUE

© 2000-2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

オブジェクトの型の識別子（IND_ICHIMOKU は CiIchimoku）

© 2000-2025, MetaQuotes Ltd.

CiMA

CiMA は移動平均（Moving Average）テクニカル指標の使用に意図されたクラスです。

説明

CiMA クラスは、移動平均指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiMA: public CIndicator
```

タイトル

```
#include <Indicators\Trend.mqh>
```

継承階層

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiMA

クラスメソッド

属性	
MaPeriod	平均期間を返します。
MaShift	水平方向のシフトを返します。
MaMethod	平均化の方法を返します。
Applied	適用する型またはハンドルを返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Main	バッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

MaPeriod

平均期間を返します。

```
int  MaPeriod() const
```

戻り値

指標の作成時に定義された平均期間

© 2000-2025, MetaQuotes Ltd.

MaShift

指標の水平方向のシフトを返します。

```
int  MaShift() const
```

戻り値

指標の作成時に定義された水平方向のシフトの値

© 2000-2025, MetaQuotes Ltd.

MaMethod

平均化の方法を返します。

```
ENUM_MA_METHOD  MaMethod() const
```

戻り値

指標の作成時に定義された平均化の方法（ENUM_MA_METHOD 列挙の値）を返します。

© 2000–2025, MetaQuotes Ltd.

Applied

適用する型またはハンドルを返します。

```
int  Applied() const
```

戻り値

指標の作成時に定義された、適用する型またはハンドル

© 2000-2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string            string,        // シンボル
   ENUM_TIMEFRAMES   period,        // 期間
   int               ma_period,     // 平均期間
   int               ma_shift,      // 水平シフト
   ENUM_MA_METHOD    ma_method,     // 平均化の方法
   int               applied        // 適用するハンドルの価格型
  )
```

パラメータ

string

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

ma_period

[in] 平均期間

ma_shift

[in] 水平方向のシフト

ma_method

[in] 平均化の方法（ENUM_MA_METHOD 列挙）

applied

[in] 適用する価格型またはハンドル

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000–2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

オブジェクトの型の識別子（IND_MA は CiMA）

© 2000–2025, MetaQuotes Ltd.

CiSAR

CiSAR はパラボリック SAR(Parabolic Stop and Reverse system、放物線の停留と反転)テクニカル指標の使用に意図されたクラスです。

説明

CiSAR クラスは、パラボリック SAR指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiSAR: public CIndicator
```

タイトル

```
#include <Indicators\Trend.mqh>
```

継承階層

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiSAR

クラスメソッド

属性	
SarStep	価格増分ステップを返します。
Maximum	ステップの最大値を返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Main	バッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

SarStep

価格増分ステップ(加速因子)を返します。

```
double  SarStep() const
```

戻り値

指標の作成時に定義された価格増分ステップ

© 2000-2025, MetaQuotes Ltd.

Maximum

ステップの最大値を返します。

```
double  Maximum() const
```

戻り値

指標の作成時に定義されたステップの最大値

© 2000–2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string           symbol,      // シンボル
   ENUM_TIMEFRAMES  period,      // 期間
   double           step,        // ステップ
   double           maximum      // 係数
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

step

[in] 速度の増加のためのステップを返します。

maximum

[in] 最高ストップレベル

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000–2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

オブジェクトの型の識別子（IND_SAR は CiSAR）

© 2000–2025, MetaQuotes Ltd.

CiStdDev

CiStdDev は標準偏差（Standard Deviation）テクニカル指標の使用に意図されたクラスです。

説明

CiStdDev クラスは、標準偏差指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiStdDev: public CIndicator
```

タイトル

```
#include <Indicators\Trend.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CIndicator
CiStdDev

クラスメソッド

属性	
MaPeriod	平均期間を返します。
MaShift	水平方向のシフトを返します。
MaMethod	平均化の方法を返します。
Applied	適用する型またはハンドルを返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Main	バッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

© 2000-2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

MaPeriod

平均期間を返します。

```
int  MaPeriod() const
```

戻り値

指標の作成時に定義された平均期間

© 2000-2025, MetaQuotes Ltd.

MaShift

指標の水平方向のシフトを返します。

```
int  MaShift() const
```

戻り値

指標の作成時に定義された水平方向のシフトの値

© 2000-2025, MetaQuotes Ltd.

MaMethod

平均化の方法を返します。

```
ENUM_MA_METHOD  MaMethod() const
```

戻り値

指標の作成時に定義された平均化の方法（ENUM_MA_METHOD 列挙値）を返します。

© 2000–2025, MetaQuotes Ltd.

Applied

適用する型またはハンドルを返します。

```
int  Applied() const
```

戻り値

指標の作成時に定義された、適用する型またはハンドル

© 2000–2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string            symbol,         // シンボル
   ENUM_TIMEFRAMES   period,         // 期間
   int               ma_period,      // 平均期間
   int               ma_shift,       // 水平シフト
   ENUM_MA_METHOD    ma_method,      // 平均化の方法
   int               applied         // 適用する価格型またはハンドル
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

ma_period

[in] 平均期間

ma_shift

[in] 水平方向のシフト

ma_method

[in] 平均化の方法（ENUM_MA_METHOD 列挙）

applied

[in] 適用する価格型またはハンドル

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000-2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

オブジェクトの型の識別子（IND_STDDEV は CiStdDev）

© 2000-2025, MetaQuotes Ltd.

CiDEMA

CiDEMA は2 重指数移動平均（Double Exponential Moving Average）テクニカル指標の使用に意図されたクラスです。

説明

CiDEMA クラスは、2 重指数移動平均指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiDEMA: public CIndicator
```

タイトル

```
#include <Indicators\Trend.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CIndicator
CiDEMA

クラスメソッド

属性	
MaPeriod	平均期間を返します。
IndShift	水平方向のシフトを返します。
Applied	適用する型またはハンドルを返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Main	バッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

MaPeriod

平均期間を返します。

```
int  MaPeriod() const
```

戻り値

指標の作成時に定義された平均期間

© 2000–2025, MetaQuotes Ltd.

IndShift

指標の水平方向のシフトを返します。

```
int  IndShift() const
```

戻り値

指標の作成時に定義された水平方向のシフトの値

© 2000-2025, MetaQuotes Ltd.

Applied

適用する型またはハンドルを返します。

```
int  Applied() const
```

戻り値

指標の作成時に定義された、適用する型またはハンドル

© 2000–2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string           string,        // シンボル
   ENUM_TIMEFRAMES  period,        // 期間
   int              ma_period,     // 平均期間
   int              ind_shift,     // シフト
   int              applied        // 適用するハンドルの価格型
  )
```

パラメータ

string

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

ma_period

[in] 平均期間

ind_shift

[in] 水平方向のシフト

applied

[in] 適用する価格型またはハンドル

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000–2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

オブジェクトの型の識別子（IND_DEMA は CiDEMA）

© 2000–2025, MetaQuotes Ltd.

CiTEMA

CiTEMA は3 重指数移動平均（Triple Exponential Moving Average）テクニカル指標の使用に意図されたクラスです。

説明

CiTEMA クラスは、3 重指数移動平均指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiTEMA: public CIndicator
```

タイトル

```
#include <Indicators\Trend.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CIndicator
CiTEMA

クラスメソッド

属性	
MaPeriod	平均期間を返します。
IndShift	水平方向のシフトを返します。
Applied	適用する型またはハンドルを返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Main	バッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

MaPeriod

平均期間を返します。

```
int  MaPeriod() const
```

戻り値

指標の作成時に定義された平均期間

© 2000–2025, MetaQuotes Ltd.

IndShift

指標の水平方向のシフトを返します。

```
int  IndShift() const
```

戻り値

指標の作成時に定義された水平方向のシフトの値

© 2000–2025, MetaQuotes Ltd.

Applied

適用する型またはハンドルを返します。

```
int  Applied() const
```

戻り値

指標の作成時に定義された、適用する型またはハンドル

© 2000–2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string           symbol,         // シンボル
   ENUM_TIMEFRAMES  period,         // 期間
   int              ma_period,      // 平均期間
   int              ma_shift,       // オフセット
   int              applied         // 適用するハンドルの価格型
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

ma_period

[in] 平均期間

ma_shift

[in] 水平方向のシフト

applied

[in] 適用する価格型またはハンドル

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000–2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

オブジェクトの型の識別子（IND_TEMA は CiTEMA）

© 2000–2025, MetaQuotes Ltd.

CiFrAMA

CiFrAMA はフラクタル適応型移動平均（Fractal Adaptive Moving Average）テクニカル指標の使用に意図されたクラスです。

説明

CiFrAMA クラスは、フラクタル適応型移動平均指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiFrAMA: public CIndicator
```

タイトル

```
#include <Indicators\Trend.mqh>
```

継承階層

COBject
CArray
CArrayObj
CSeries
CIndicator
CiFrAMA

クラスメソッド

属性	
MaPeriod	平均期間を返します。
IndShift	水平方向のシフトを返します。
Applied	適用する型またはハンドルを返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Main	バッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

MaPeriod

平均期間を返します。

```
int  MaPeriod() const
```

戻り値

指標の作成時に定義された平均期間

© 2000–2025, MetaQuotes Ltd.

IndShift

指標の水平方向のシフトを返します。

```
int  IndShift() const
```

戻り値

指標の作成時に定義された水平方向のシフトの値

© 2000–2025, MetaQuotes Ltd.

Applied

適用する型またはハンドルを返します。

```
int  Applied() const
```

戻り値

指標の作成時に定義された、適用する型またはハンドル

© 2000–2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string           symbol,         // シンボル
   ENUM_TIMEFRAMES  period,         // 期間
   int              ma_period,      // 平均期間
   int              ma_shift,       // オフセット
   int              applied         // 適用するハンドルの価格型
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

ma_period

[in] 平均期間

ma_shift

[in] 水平方向のシフト

applied

[in] 適用する価格型またはハンドル

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000–2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

オブジェクトの型の識別子（IND_FRAMA は CiFrAMA）

© 2000–2025, MetaQuotes Ltd.

CiAMA

CiAMA は適応型移動平均（Adaptive Moving Average）テクニカル指標の使用に意図されたクラスです。

説明

CiAMA クラスは、適応型移動平均指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiAMA: public CIndicator
```

タイトル

```
#include <Indicators\Trend.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CIndicator
CiAMA

クラスメソッド

属性	
MaPeriod	平均期間を返します。
FastEmaPeriod	高速 EMA の平均期間を返します。
SlowEmaPeriod	低速 EMA の平均期間を返します。
IndShift	水平方向のシフトを返します。
Applied	適用する型またはハンドルを返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Main	バッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

MaPeriod

平均期間を返します。

```
int  MaPeriod() const
```

戻り値

指標の作成時に定義された平均期間

© 2000–2025, MetaQuotes Ltd.

FastEmaPeriod

高速 EMA の平均期間を返します。

```
int  FastEmaPeriod() const
```

戻り値

指標の作成時に定義された高速 EMA の平均期間

© 2000–2025, MetaQuotes Ltd.

SlowEmaPeriod

低速 EMA の平均期間を返します。

```
int  SlowEmaPeriod() const
```

戻り値

指標の作成時に定義された低速 EMA の平均期間

© 2000–2025, MetaQuotes Ltd.

IndShift

指標の水平方向のシフトを返します。

```
int  IndShift() const
```

戻り値

指標の作成時に定義された水平方向のシフトの値

© 2000–2025, MetaQuotes Ltd.

Applied

適用する型またはハンドルを返します。

```
int  Applied() const
```

戻り値

指標の作成時に定義された、適用する型またはハンドル

© 2000–2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string           string,              // シンボル
   ENUM_TIMEFRAMES  period,              // 期間
   int              ma_period,           // 平均期間
   int              fast_ema_period,     // 高速 EMA 期間
   int              slow_ema_period,     // 低速 EMA 期間
   int              ind_shift,           // シフト
   int              applied              // 適用する価格型またはハンドル
  )
```

パラメータ

string

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

ma_period

[in] 平均期間

fast_ema_period

[in] 高速 EMA 平均期間

slow_ema_period

[in] 低速 EMA 平均期間

ind_shift

[in] 水平方向のシフト

applied

[in] 適用する価格型またはハンドル

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000–2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

オブジェクトの型の識別子（IND_AMA は CiAMA）

© 2000–2025, MetaQuotes Ltd.

CiVIDyA

CiVIDyA は可変インデックス動的平均（Variable Index Dynamic Average）テクニカル指標の使用に意図されたクラスです。

説明

CiVIDyA クラスは、可変インデックス動的平均指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiVIDyA: public CIndicator
```

タイトル

```
#include <Indicators\Trend.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CIndicator
CiVIDyA

クラスメソッド

属性	
CmoPeriod	モメンタムの期間を返します。
EmaPeriod	EMA の平均期間を返します。
IndShift	水平方向のシフトを返します。
Applied	適用する型またはハンドルを返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Main	バッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

CmoPeriod

モメンタムの期間を返します。

```
int  CmoPeriod() const
```

戻り値

指標の作成時に定義されたモメンタムの期間を返します。

© 2000–2025, MetaQuotes Ltd.

EmaPeriod

EMA の平均期間を返します。

```
int  EmaPeriod() const
```

戻り値

指標の作成時に定義されたEMA の平均期間

© 2000–2025, MetaQuotes Ltd.

IndShift

指標の水平方向のシフトを返します。

```
int  IndShift() const
```

戻り値

指標の作成時に定義された水平方向のシフトの値

© 2000–2025, MetaQuotes Ltd.

Applied

適用する型またはハンドルを返します。

```
int  Applied() const
```

戻り値

指標の作成時に定義された、適用する型またはハンドル

© 2000–2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string           symbol,          // シンボル
   ENUM_TIMEFRAMES  period,          // 期間
   int              cmo_period,      // モメンタム期間
   int              ema_period,      // 平均期間
   int              ind_shift,       // シフト
   int              applied          // 適用する価格型またはハンドル
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

cmo_period

[in] モメンタム期間

ema_period

[in] 平均期間

ind_shift

[in] 水平方向のシフト

applied

[in] 適用する価格型またはハンドル

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000-2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

オブジェクトの型の識別子（IND_VIDYA は CiVIDyA）

© 2000–2025, MetaQuotes Ltd.

オシレーター

このセクションには、MQL5 標準ライブラリのオシレータークラスの技術的な詳細と、その全ての主要コンポーネントの説明が含まれています。

クラス/グループ	説明
CiATR	ATR(Average True Range)。
CiBearsPower	ベアパワー(Bears Power)。
CiBullsPower	ブルパワー(Bulls Power)。
CiCCI	コモディティチャンネルインデックス(Commodity Channel Index)。
CiChaikin	チャイキンオシレーター(Chaikin Oscillator)。
) CiDeMarker	デマーカー(DeMarker)。
CiForce	勢力指数(Force Index)。
CiMACD	移動平均収束拡散法(Moving Averages Convergence-Divergence)
CiMomentum	モメンタム(Momentum)。
CiOsMA	移動平均オシレーター(Moving Average of Oscillator)(MACDヒストグラム)。
CiRSI	相対力指数(Relative Strength Index)。
CiRVI	相対活力指数(Relative Vigor Index)。
CiStochastic	ストキャスティックス(Stochastic Oscillator)。
CiWPR	ウィリアムパーセントレンジ(Williams' Percent Range)。
CiTriX	3 重指数移動平均オシレーター(Triple Exponential Moving Averages Oscillator)。

© 2000-2025, MetaQuotes Ltd.

CiATR

CiATR は ATR(Average True Range) テクニカル指標の使用に意図されたクラスです。

説明

CiATR クラスは、ATR テクニカル指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiATR: public CIndicator
```

タイトル

```
#include <Indicators\Oscilators.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CIndicator
CiATR

クラスメソッド

属性	
MaPeriod	平均期間を返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Main	バッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

MaPeriod

平均期間を返します。

```
int  MaPeriod() const
```

戻り値

指標の作成時に定義された平均期間

© 2000–2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string           symbol,         // シンボル
   ENUM_TIMEFRAMES  period,         // 期間
   int              ma_period       // 平均期間
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

ma_period

[in] 平均期間

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000–2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

Indicator type (IND_ATR for CiATR).

© 2000–2025, MetaQuotes Ltd.

CiBearsPower

CiBearsPower はベアパワー(BearsPower)テクニカル指標の使用に意図されたクラスです。

説明

CiBearsPower クラスは、ベアパワー指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiBearsPower: public CIndicator
```

タイトル

```
#include <Indicators\Oscilators.mqh>
```

継承階層

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiBearsPower

クラスメソッド

属性	
MaPeriod	平均期間を返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Main	バッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

© 2000-2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

MaPeriod

平均期間を返します。

```
int  MaPeriod() const
```

戻り値

指標の作成時に定義された平均期間

© 2000–2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string            symbol,         // シンボル
   ENUM_TIMEFRAMES   period,         // 期間
   int               ma_period       // 平均期間
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

ma_period

[in] 平均期間

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000–2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

Indicator type (IND_BEARS for CiBearsPower).

© 2000–2025, MetaQuotes Ltd.

CiBullsPower

CiBullsPower はブルパワー(Bulls Power)テクニカル指標の使用に意図されたクラスです。

説明

CiBullsPower クラスは、ブルパワー指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiBullsPower: public CIndicator
```

タイトル

```
#include <Indicators\Oscilators.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CIndicator
CiBullsPower

クラスメソッド

属性	
MaPeriod	平均期間を返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Main	バッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

MaPeriod

平均期間を返します。

```
int  MaPeriod() const
```

戻り値

指標の作成時に定義された平均期間

© 2000–2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string            symbol,         // シンボル
   ENUM_TIMEFRAMES   period,         // 期間
   int               ma_period       // 平均期間
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

ma_period

[in] 平均期間

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000–2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

Indicator type (IND_BULLS for CiBullsPower).

© 2000–2025, MetaQuotes Ltd.

CiCCI

CiCCI は CCI（商品チャンネル指数 Commodity Channel Index ）テクニカル指標の使用に意図されたクラスです。

説明

CiCCI クラスは、CCI テクニカル指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiCCI: public CIndicator
```

タイトル

```
#include <Indicators\Oscilators.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CIndicator
CiCCI

クラスメソッド

属性	
MaPeriod	平均期間を返します。
Applied	適用する型またはハンドルを返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Main	バッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

MaPeriod

平均期間を返します。

```
int  MaPeriod() const
```

戻り値

指標の作成時に定義された平均期間

© 2000–2025, MetaQuotes Ltd.

Applied

適用する型またはハンドルを返します。

```
int  Applied() const
```

戻り値

指標の作成時に定義された、適用する型またはハンドル

© 2000–2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string           symbol,         // シンボル
   ENUM_TIMEFRAMES  period,         // 期間
   int              ma_period,      // 平均期間
   int              applied         // 適用する価格型またはハンドル
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

ma_period

[in] 平均期間

applied

[in] 適用する価格型またはハンドル

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000–2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

Indicator type (IND_CCI for CiCCI).

© 2000–2025, MetaQuotes Ltd.

CiChaikin

CiChaikin はチャイキンオシレーター(Chaikin Oscillator)テクニカル指標の使用に意図されたクラスです。

説明

CiChaikin クラスは、チャイキンオシレーター指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiChaikin: public CIndicator
```

タイトル

```
#include <Indicators\Oscilators.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CIndicator
CiChaikin

クラスメソッド

属性	
FastMaPeriod	高速 MA の平均期間を返します。
SlowMaPeriod	低速 MA の平均期間を返します。
MaMethod	平均化の方法を返します。
Applied	適用する型またはハンドルを返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Main	バッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

© 2000-2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

FastMaPeriod

高速 EMA の平均期間を返します。

```
int  FastMaPeriod() const
```

戻り値

指標の作成時に定義された高速 EMA の平均期間

© 2000–2025, MetaQuotes Ltd.

SlowMaPeriod

低速 EMA の平均期間を返します。

```
int  SlowMaPeriod() const
```

戻り値

指標の作成時に定義された低速 EMA の平均期間

© 2000–2025, MetaQuotes Ltd.

MaMethod

平均化の方法を返します。

```
ENUM_MA_METHOD  MaMethod() const
```

戻り値

指標の作成時に定義された平均化の方法

© 2000–2025, MetaQuotes Ltd.

Applied

適用するボリュームの種類を返します。

```
ENUM_APPLIED_VOLUME  Applied() const
```

戻り値

指標の作成時に定義された、適用するボリュームの種類

© 2000–2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string               symbol,              // シンボル
   ENUM_TIMEFRAMES      period,              // 期間
   int                  fast_ma_period,      // 高速 EMA 期間
   int                  slow_ma_period,      // 低速 EMA 期間
   ENUM_MA_METHOD       ma_method,           // 平均期間
   ENUM_APPLIED_VOLUME  applied              // 適用するボリュームの種類
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

fast_ma_period

[in] 高速 EMA の期間

slow_ma_period

[in] 低速 EMA の期間

ma_method

[in] 平均化の方法（ENUM_MA_METHOD 列挙）

applied

[in] 適用するボリュームの種類（ENUM_APPLIED_VOLUME 列挙）

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000–2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

Indicator type (IND_CHAIKIN for CiChaikin).

© 2000–2025, MetaQuotes Ltd.

) CiDeMarker

CiDeMarker はデマーカー(DeMarker) テクニカル指標の使用に意図されたクラスです。

説明

CiDeMarker クラスは、デマーカー指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiDeMarker: public CIndicator
```

タイトル

```
#include <Indicators\Oscilators.mqh>
```

継承階層

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiDeMarker

クラスメソッド

属性	
MaPeriod	平均期間を返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Main	バッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

© 2000-2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

MaPeriod

平均期間を返します。

```
int  MaPeriod() const
```

戻り値

指標の作成時に定義された平均期間

© 2000–2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string           symbol,        // シンボル
   ENUM_TIMEFRAMES  period,        // 期間
   int              ma_period      // 平均期間
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

ma_period

[in] 平均期間

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000-2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

Indicator type (IND_DEMARKER for CiDeMarker).

© 2000-2025, MetaQuotes Ltd.

CiForce

CiForce は勢力指数（Force Index）テクニカル指標の使用に意図されたクラスです。

説明

CiForce クラスは、勢力指数指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiForce: public CIndicator
```

タイトル

```
#include <Indicators\Oscilators.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CIndicator
CiForce

クラスメソッド

属性	
MaPeriod	平均期間を返します。
MaMethod	平均化の方法を返します。
Applied	適用する型またはハンドルを返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Main	バッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

MaPeriod

平均期間を返します。

```
int  MaPeriod() const
```

戻り値

指標の作成時に定義された平均期間

© 2000–2025, MetaQuotes Ltd.

MaMethod

平均化の方法を返します。

```
ENUM_MA_METHOD  MaMethod() const
```

戻り値

指標の作成時に定義された平均化の方法

© 2000–2025, MetaQuotes Ltd.

Applied

適用するボリュームの種類を返します。

```
ENUM_APPLIED_VOLUME  Applied() const
```

戻り値

指標の作成時に定義された、適用するボリュームの種類

© 2000–2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string                symbol,          // シンボル
   ENUM_TIMEFRAMES       period,          // 期間
   int                   ma_period,       // 平均期間
   ENUM_MA_METHOD        ma_method,       // 平均化の方法
   ENUM_APPLIED_VOLUME   applied          // 適用するボリュームの種類
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

ma_period

[in] 平均期間

ma_method

[in] 平均化の方法（ENUM_MA_METHOD 列挙）

applied

[in] 適用するボリュームの種類（ENUM_APPLIED_VOLUME 列挙）

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000–2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

Indicator type (IND_FORCE for CiForce).

© 2000–2025, MetaQuotes Ltd.

CiMACD

CiMACD は移動平均収束拡散法（ Moving Averages Convergence-Divergence ）テクニカル指標の使用に意図されたクラスです。

説明

CiMACD クラスは、移動平均収束拡散法テクニカル指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiMACD: public CIndicator
```

タイトル

```
#include <Indicators\Oscilators.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CIndicator
CiMACD

クラスメソッド

属性	
FastEmaPeriod	高速 EMA の平均期間を返します。
SlowEmaPeriod	低速 EMA の平均期間を返します。
SignalPeriod	シグナル線の平均期間を返します。
Applied	適用する型またはハンドルを返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Main	中正線のバッファ要素を返します。
Signal	シグナル線のバッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

© 2000-2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

FastEmaPeriod

高速 EMA の平均期間を返します。

```
int  FastEmaPeriod() const
```

戻り値

指標の作成時に定義された高速 EMA の平均期間

© 2000–2025, MetaQuotes Ltd.

SlowEmaPeriod

低速 EMA の平均期間を返します。

```
int  SlowEmaPeriod() const
```

戻り値

指標の作成時に定義された低速 EMA の平均期間

© 2000–2025, MetaQuotes Ltd.

SignalPeriod

シグナル線の平均期間を返します。

```
int  SignalPeriod() const
```

戻り値

指標の作成時に定義されたシグナル線の平均期間

© 2000-2025, MetaQuotes Ltd.

Applied

適用する型またはハンドルを返します。

```
int  Applied() const
```

戻り値

指標の作成時に定義された、適用する型またはハンドル

© 2000–2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string           symbol,              // シンボル
   ENUM_TIMEFRAMES  period,              // 期間
   int              fast_ema_period,     // 高速 EMA 期間
   int              slow_ema_period,     // 低速 EMA 期間
   int              signal_period,       // シグナル期間
   int              applied              // 適用する価格型またはハンドル
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

fast_ema_period

[in] 高速 EMA 期間

slow_ema_period

[in] 低速 EMA 期間

signal_period

[in] シグナル線の期間

applied

[in] 適用する価格型またはハンドル

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000-2025, MetaQuotes Ltd.

Signal

指定されたインデックスを持つシグナル線のバッファ要素を返します。

```
double  Signal(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

指定されたインデックスを持つシグナル線のバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000-2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

Indicator type (IND_MACD for CiMACD).

© 2000-2025, MetaQuotes Ltd.

CiMomentum

CiMomentum はモメンタム(Momentum)テクニカル指標の使用に意図されたクラスです。

説明

CiMomentum クラスは、モメンタム指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiMomentum: public CIndicator
```

タイトル

```
#include <Indicators\Oscilators.mqh>
```

継承階層

COBject
CArray
CArrayObj
CSeries
CIndicator
CiMomentum

クラスメソッド

属性	
MaPeriod	平均期間を返します。
Applied	適用する型またはハンドルを返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Main	バッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

MaPeriod

平均期間を返します。

```
int  MaPeriod() const
```

戻り値

指標の作成時に定義された平均期間

© 2000–2025, MetaQuotes Ltd.

Applied

適用する型またはハンドルを返します。

```
int  Applied() const
```

戻り値

指標の作成時に定義された、適用する型またはハンドル

© 2000–2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string            symbol,         // シンボル
   ENUM_TIMEFRAMES   period,         // 期間
   int               ma_period,      // 平均期間
   int               applied         // 適用する価格型またはハンドル
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

ma_period

[in] 平均期間

applied

[in] 適用する価格型またはハンドル

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000–2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

Indicator type (IND_MOMENTUM for CiMomentum).

© 2000–2025, MetaQuotes Ltd.

CiOsMA

CiOsMA は移動平均オシレーター(Moving Average of Oscillator、MACD ヒストグラム)テクニカル指標の使用に意図されたクラスです。

説明

CiOsMA クラスは、移動平均オシレーター(MACD ヒストグラム)指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiOsMA: public CIndicator
```

タイトル

```
#include <Indicators\Oscilators.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CIndicator
CiOsMA

クラスメソッド

属性	
FastEmaPeriod	高速 EMA の平均期間を返します。
SlowEmaPeriod	低速 EMA の平均期間を返します。
SignalPeriod	シグナル線の平均期間を返します。
Applied	適用する型またはハンドルを返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Main	バッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

FastEmaPeriod

高速 EMA の平均期間を返します。

```
int  FastEmaPeriod() const
```

戻り値

指標の作成時に定義された高速 EMA の平均期間

© 2000–2025, MetaQuotes Ltd.

SlowEmaPeriod

低速 EMA の平均期間を返します。

```
int  SlowEmaPeriod() const
```

戻り値

指標の作成時に定義された低速 EMA の平均期間

© 2000–2025, MetaQuotes Ltd.

SignalPeriod

シグナル線の平均期間を返します。

```
int  SignalPeriod() const
```

戻り値

指標の作成時に定義されたシグナル線の平均期間

© 2000–2025, MetaQuotes Ltd.

Applied

適用する型またはハンドルを返します。

```
int  Applied() const
```

戻り値

指標の作成時に定義された、適用する型またはハンドル

© 2000–2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string           symbol,              // シンボル
   ENUM_TIMEFRAMES  period,              // 期間
   int              fast_ema_period,     // 高速 EMA 期間
   int              slow_ema_period,     // 低速 EMA 期間
   int              signal_period,       // シグナル線の期間
   int              applied              // 適用する価格型またはハンドル
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

fast_ema_period

[in] 高速 EMA 期間

slow_ema_period

[in] 低速 EMA 期間

signal_period

[in] シグナル線の期間

applied

[in] 適用する価格型またはハンドル

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000–2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

Indicator type (IND_OSMA for CiOsMA).

© 2000–2025, MetaQuotes Ltd.

CiRSI

CiRSI は相対力指数（Relative Strength Index）テクニカル指標の使用に意図されたクラスです。

説明

CiRSI クラスは、相対力指数指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiRSI: public CIndicator
```

タイトル

```
#include <Indicators\Oscilators.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CIndicator
CiRSI

クラスメソッド

属性	
MaPeriod	平均期間を返します。
Applied	適用する型またはハンドルを返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Main	バッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

MaPeriod

平均期間を返します。

```
int  MaPeriod() const
```

戻り値

指標の作成時に定義された平均期間

© 2000–2025, MetaQuotes Ltd.

Applied

適用する型またはハンドルを返します。

```
int  Applied() const
```

戻り値

指標の作成時に定義された、適用する型またはハンドル

© 2000–2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string            symbol,         // シンボル
   ENUM_TIMEFRAMES   period,         // 期間
   int               ma_period,      // 平均期間
   int               applied         // 適用する価格型またはハンドル
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

ma_period

[in] 平均期間

applied

[in] 適用する価格型またはハンドル

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000-2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

オブジェクトの型の識別子（IND_RSI は CiRSI）

© 2000–2025, MetaQuotes Ltd.

CiRVI

CiRVI は相対活力指数（Relative Vigor Index）テクニカル指標の使用に意図されたクラスです。

説明

CiRVI クラスは、相対活力指数指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiRVI: public CIndicator
```

タイトル

```
#include <Indicators\Oscilators.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CIndicator
CiRVI

クラスメソッド

属性	
MaPeriod	平均期間を返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Main	基本線のバッファ要素を返します。
Signal	シグナル線のバッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

MaPeriod

平均期間を返します。

```
int  MaPeriod() const
```

戻り値

指標の作成時に定義された平均期間

© 2000–2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string            symbol,         // シンボル
   ENUM_TIMEFRAMES   period,         // 期間
   int               ma_period       // 平均期間
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

ma_period

[in] 平均期間

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000–2025, MetaQuotes Ltd.

Signal

指定されたインデックスを持つシグナル線のバッファ要素を返します。

```
double  Signal(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

指定されたインデックスを持つシグナル線のバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000–2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

Indicator type (IND_RVI for CiRVI).

© 2000–2025, MetaQuotes Ltd.

CiStochastic

CiStochastic はスト キャスティックス(Stochastic Oscillator)テクニカル指 標 の使 用 に意 図 されたクラスです。

説 明

CiStochastic クラスは、スト キャスティックスーテクニカル指 標 の作 成 、設 定 及 びデータアクセスを提 供 します。

宣 言

```
class CiStochastic: public CIndicator
```

タイト ル

```
#include <Indicators\Oscilators.mqh>
```

継 承 階 層

CObject
CArray
CArrayObj
CSeries
CIndicator
CiStochastic

クラスメソッド

属 性	
Kperiod	%K 線 の平 均 期 間 を返 します。
Dperiod	%D 線 の平 均 期 間 を返 します。
Slowing	スローイング期 間 を返 します。
MaMethod	平 均 化 の方 法 を返 します。
PriceField	適 用 する価 格 の種 類 (安 /高 または終 /終)
作 成 メソッド	
Create	指 標 を作 成 します。
データアクセスメソッド	
Main	基 本 線 のバッファ要 素 を返 します。
Signal	シグナル線 のバッファ要 素 を返 します。
入 出 力	
virtual Type	オブジェクト の型 の識 別 子 を返 します。

© 2000-2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

Kperiod

%K 線の平均期間を返します。

```
int  Kperiod() const
```

戻り値

指標の作成時に定義された %K 線の平均期間

© 2000–2025, MetaQuotes Ltd.

Dperiod

%D 線の平均期間を返します。

```
int  Dperiod() const
```

戻り値

指標の作成時に定義された %D 線の平均期間

© 2000–2025, MetaQuotes Ltd.

Slowing

スローイング期間を返します。

```
int  Slowing() const
```

戻り値

指標の作成時に定義されたスローイング期間

© 2000–2025, MetaQuotes Ltd.

MaMethod

平均化の方法を返します。

```
ENUM_MA_METHOD  MaMethod() const
```

戻り値

指標の作成時に定義された平均化の方法

© 2000–2025, MetaQuotes Ltd.

PriceField

適用する価格の種類（安/高または終/終）を返します。

```
ENUM_STO_PRICE  PriceField() const
```

戻り値

指標の作成時に定義された、適用する価格の種類

© 2000-2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string           symbol,          // シンボル
   ENUM_TIMEFRAMES  period,          // 期間
   int              Kperiod,         // %K の平均期間
   int              Dperiod,         // %D の平均期間
   int              slowing,         // スローイング期間
   ENUM_MA_METHOD   ma_method,       // 平均化の方法
   ENUM_STO_PRICE   price_field      // 適応する価格の型
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

Kperiod

[in] %K 線の平均期間

Dperiod

[in] %D 線の平均期間

slowing

[in] スローイング期間

ma_method

[in] 平均化の方法（ENUM_MA_METHOD 列挙）

price_field

[in] 適用する価格の種類（低/高または終/始）（ENUM_STO_PRICE 列挙）

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000–2025, MetaQuotes Ltd.

Signal

指定されたインデックスを持つシグナル線のバッファ要素を返します。

```
double  Signal(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

Signal

指定されたインデックスを持つシグナル線のバッファ要素を返します。

```
double  Signal(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

指定されたインデックスを持つシグナル線のバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000–2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

Indicator type (IND_STOCHASTIC for CiStochastic).

© 2000–2025, MetaQuotes Ltd.

CiTriX

CiTriX は3 重指数移動平均（ Triple Exponential Moving Averages Oscillator ）テクニカル指標の使用に意図されたクラスです。

説明

CiTriX クラスは、3 重指数移動平均指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiTriX: public CIndicator
```

タイトル

```
#include <Indicators\Oscilators.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CIndicator
CiTriX

クラスメソッド

属性	
MaPeriod	平均期間を返します。
Applied	適用する型またはハンドルを返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Main	バッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

MaPeriod

平均期間を返します。

```
int  MaPeriod() const
```

戻り値

指標の作成時に定義された平均期間

© 2000–2025, MetaQuotes Ltd.

Applied

適用する型またはハンドルを返します。

```
int  Applied() const
```

戻り値

指標の作成時に定義された、適用する型またはハンドル

© 2000–2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string            symbol,         // シンボル
   ENUM_TIMEFRAMES   period,         // 期間
   int               ma_period,      // 平均期間
   int               applied         // 価格の型かハンドル
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

ma_period

[in] 平均期間

applied

[in] 適用する価格型またはハンドル

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000–2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

Indicator type (IND_TRIX for CiTriX).

© 2000–2025, MetaQuotes Ltd.

CiWPR

CiWPR はウィリアムパーセントレンジ(Williams' Percent Range)テクニカル指標の使用に意図されたクラスです。

説明

CiWPR クラスは、ウィリアムパーセントレンジ テクニカル指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiWPR: public CIndicator
```

タイトル

```
#include <Indicators\Oscilators.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CIndicator
CiWPR

クラスメソッド

属性	
CalcPeriod	計算期間を返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Main	バッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

© 2000-2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

CalcPeriod

計算の期間を返します。

```
int  CalcPeriod() const
```

戻り値

指標の作成時に定義された計算期間を返します。

© 2000–2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string           symbol,          // シンボル
   ENUM_TIMEFRAMES  period,          // 期間
   int              calc_period      // 計算期間
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

calc_period

[in] 計算期間

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000–2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

Indicator type (IND_WPR for CiWPR).

© 2000–2025, MetaQuotes Ltd.

ボリューム指標

このセクションには、MQL5 標準ライブラリのボリューム指標クラスの技術的な詳細と、その全ての主要コンポーネントの説明が含まれています。

クラス/グループ	説明
CiAD	蓄積/配信（Accumulation/Distribution）。
CiMFI	マネーフローインデックス（Money Flow Index）。
CiOBV	オンバランスボリューム（On Balance Volume）。
CiVolumes	ボリューム。

© 2000–2025, MetaQuotes Ltd.

CiAD

CiAD はAccumulation（蓄積）/ Distribution（分配）テクニカル指標の使用に意図されたクラスです。

説明

CiAD クラスは、A/D 指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiAD: public CIndicator
```

タイトル

```
#include <Indicators\Volumes.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CIndicator
CiAD

クラスメソッド

属性	
Applied	適用するボリュームの種類を返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Main	バッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

Applied

適用するボリュームの種類を返します。

```
ENUM_APPLIED_VOLUME  Applied() const
```

戻り値

指標の作成時に定義された、適用するボリュームの種類

© 2000–2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string                symbol,      // シンボル
   ENUM_TIMEFRAMES       period,      // 期間
   ENUM_APPLIED_VOLUME   applied      // 適用するボリュームの種類
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

applied

[in] 適用するボリュームの種類（ENUM_APPLIED_VOLUME 列挙）

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000-2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

オブジェクトの型の識別子（IND_ADはCiAD）

© 2000–2025, MetaQuotes Ltd.

CiMFI

CiMFI は MFI(Money Flow Index) テクニカル指標の使用に意図されたクラスです。

説明

CiMFI クラスは、MFI 指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiMFI: public CIndicator
```

タイトル

```
#include <Indicators\Volumes.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CIndicator
CiMFI

クラスメソッド

属性	
MaPeriod	平均期間を返します。
Applied	適用するボリュームの種類を返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Main	バッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

MaPeriod

平均期間を返します。

```
int  MaPeriod() const
```

戻り値

指標の作成時に定義された平均期間

© 2000–2025, MetaQuotes Ltd.

Applied

適用するボリュームの種類を返します。

```
ENUM_APPLIED_VOLUME  Applied() const
```

戻り値

指標の作成時に定義された、適用するボリュームの種類

© 2000–2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string                symbol,          // シンボル
   ENUM_TIMEFRAMES       period,          // 期間
   int                   ma_period,       // 平均期間
   ENUM_APPLIED_VOLUME   applied          // 適用するボリュームの種類
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

ma_period

[in] 平均期間

applied

[in] 適用するボリュームの種類（ENUM_APPLIED_VOLUME 列挙）

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000–2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

オブジェクトの型の識別子（IND_MFI は CiMFI）

© 2000–2025, MetaQuotes Ltd.

CiOBV

CiOBV はオンバランスボリューム(On Balance Volume)テクニカル指標の使用に意図されたクラスです。

説明

CiOBV クラスは、オンバランスボリューム指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiOBV: public CIndicator
```

タイトル

```
#include <Indicators\Volumes.mqh>
```

継承階層

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiOBV

クラスメソッド

属性	
Applied	適用するボリュームの種類を返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Main	バッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

Applied

適用するボリュームの種類を返します。

```
ENUM_APPLIED_VOLUME  Applied() const
```

戻り値

指標の作成時に定義された、適用するボリュームの種類

© 2000–2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string                symbol,      // シンボル
   ENUM_TIMEFRAMES       period,      // 期間
   ENUM_APPLIED_VOLUME   applied      // 適用するボリュームの種類
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

applied

[in] 適用するボリュームの種類（ENUM_APPLIED_VOLUME 列挙）

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000–2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

オブジェクトの型の識別子（IND_OBV は CiOBV）

© 2000–2025, MetaQuotes Ltd.

CiVolumes

CiVolumes はボリューム(Volumes)テクニカル指標の使用に意図されたクラスです。

説明

CiVolumes クラスは、ボリューム指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiVolumes: public CIndicator
```

タイトル

```
#include <Indicators\Volumes.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CIndicator
CiVolumes

クラスメソッド

属性	
Applied	適用するボリュームの種類を返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Main	バッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

Applied

適用するボリュームの種類を返します。

```
ENUM_APPLIED_VOLUME  Applied() const
```

戻り値

指標の作成時に定義された、適用するボリュームの種類

© 2000–2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string                symbol,      // シンボル
   ENUM_TIMEFRAMES       period,      // 期間
   ENUM_APPLIED_VOLUME   applied      // 適用するボリュームの種類
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

applied

[in] 適用するボリュームの種類（ENUM_APPLIED_VOLUME 列挙）

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000-2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

オブジェクトの型の識別子（IND_VOLUMES は CiVolumes）

© 2000–2025, MetaQuotes Ltd.

ビルウィリアムズ指標

このセクションには、MQL5　標準ライブラリのビルウィリアムズ指標クラスの技術的な詳細と、その全ての主要コンポーネントの説明が含まれています。

クラス/グループ	説明
CiAC	ACオシレーター。
CiAlligator	アリゲーター。
CiAO	オーサムオシレーター。
CiFractals	フラクタル。
CiGator	ゲーターオシレーター。
CiBWMFI	マーケットファシリテーションインデックス。

© 2000–2025, MetaQuotes Ltd.

CiAC

CiAC はACオシレーターテクニカル指標の使用に意図されたクラスです。

説明

CiAC は、ACオシレーターテクニカル指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiAC: public CIndicator
```

タイトル

```
#include <Indicators\BillWilliams.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CIndicator
CiAC

クラスメソッド

作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Main	バッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string           symbol,      // シンボル
   ENUM_TIMEFRAMES  period       // 期間
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000–2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

Indicator type (IND_AC for CiAC).

© 2000–2025, MetaQuotes Ltd.

CiAlligator

CiAlligator はアリゲータテクニカル指標の使用に意図されたクラスです。

説明

CiAlligator クラスは、アリゲータオシレーターテクニカル指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiAlligator: public CIndicator
```

タイトル

```
#include <Indicators\BillWilliams.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CIndicator
CiAlligator

クラスメソッド

属性	
JawPeriod	顎線の平均期間を返します。
JawShift	顎線の水平方向のシフトを返します。
TeethPeriod	歯線の平均期間を返します。
TeethShift	歯線の水平方向のシフトを返します。
LipsPeriod	口線の平均期間を返します。
LipsShift	口線の水平方向のシフトを返します。
MaMethod	平均化の方法を返します。
Applied	適用する型またはハンドルを返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Jaw	顎線バッファのバッファ要素を返します。
Teeth	歯線バッファのバッファ要素を返します。

© 2000–2025, MetaQuotes Ltd.

属性	
Lips	口線バッファのバッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

JawPeriod

顎線の平均期間を返します。

```
int  JawPeriod() const
```

戻り値

指標の作成時に定義された顎線の平均期間

© 2000–2025, MetaQuotes Ltd.

JawShift

顎線の水平方向のシフトを返します。

```
int  JawShift() const
```

戻り値

指標の作成時に定義された顎線の水平方向のシフト

© 2000–2025, MetaQuotes Ltd.

TeethPeriod

歯線の平均期間を返します。

```
int  TeethPeriod() const
```

戻り値

指標の作成時に定義された歯線の平均期間

© 2000–2025, MetaQuotes Ltd.

TeethShift

歯線の水平方向のシフトを返します。

```
int  TeethShift() const
```

戻り値

指標の作成時に定義された歯線の水平方向のシフト

© 2000–2025, MetaQuotes Ltd.

LipsPeriod

口線の平均期間を返します。

```
int  LipsPeriod() const
```

戻り値

指標の作成時に定義された口線の平均期間

© 2000–2025, MetaQuotes Ltd.

LipsShift

口線の水平方向のシフトを返します。

```
int  LipsShift() const
```

戻り値

指標の作成時に定義された口線の水平方向のシフト

© 2000-2025, MetaQuotes Ltd.

MaMethod

平均化の方法を返します。

```
ENUM_MA_METHOD  MaMethod() const
```

戻り値

指標の作成時に定義された平均化の方法

© 2000–2025, MetaQuotes Ltd.

Applied

適用する型またはハンドルを返します。

```
int  Applied() const
```

戻り値

指標の作成時に定義された、適用する型またはハンドル

© 2000-2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string           symbol,           // シンボル
   ENUM_TIMEFRAMES  period,           // 期間
   int              jaw_period,       // 顎線の期間
   int              jaw_shift,        // 顎線のシフト
   int              teeth_period,     // 歯線の期間
   int              teeth_shift,      // 歯線のシフト
   int              lips_period,      // 口線の期間
   int              lips_shift,       // 口線のシフト
   ENUM_MA_METHOD   ma_method,        // 平均化の方法
   int              applied           // 適用するボリュームの種類
  )
```

パラメータ

symbol
[in] シンボル

period
[in] 時間軸（ENUM_TIMEFRAMES 列挙）

jaw_period
[in] 顎線の期間

jaw_shift
[in] 顎線のシフト

teeth_period
[in] 歯線の期間

teeth_shift
[in] 歯線のシフト

lips_period
[in] 口線の期間

lips_shift
[in] 口線のシフト

ma_method
[in] 移動平均メソッド（ENUM_MA_METHOD 列挙）

applied
[in] 適用するボリュームの種類

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Jaw

指定されたインデックスを持つ顎線のバッファ要素を返します。

```
double  Jaw(
   int  index       // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

指定されたインデックスを持つ顎線のバッファ要素、または正当なデータが不在な場合はEMPTY_VALUE

© 2000-2025, MetaQuotes Ltd.

Teeth

指定されたインデックスを持つ歯 線のバッファ要素を返します。

```
double  Teeth(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

指定されたインデックスを持つ歯線のバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000-2025, MetaQuotes Ltd.

Lips

指定されたインデックスを持つ口線のバッファ要素を返します。

```
double  Lips(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

指定されたインデックスを持つ口線のバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000-2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

Indicator type (IND_ALLIGATOR for CiAlligator).

© 2000–2025, MetaQuotes Ltd.

CiAO

CiAO はオーサムオシレーターテクニカル指標の使用に意図されたクラスです。

説明

CiAO クラスは、オーサムオシレーターテクニカル指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiAO: public CIndicator
```

タイトル

```
#include <Indicators\BillWilliams.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CIndicator
CiAO

クラスメソッド

作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Main	バッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string           symbol,      // シンボル
   ENUM_TIMEFRAMES  period       // 期間
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000–2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

Indicator type (IND_AO for CiAO).

© 2000–2025, MetaQuotes Ltd.

CiFractals

CiFractals はフラクタルテクニカル指標の使用に意図されたクラスです。

説明

CiFractals クラスは、フラクタル指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiFractals: public CIndicator
```

タイトル

```
#include <Indicators\BillWilliams.mqh>
```

継承階層

COBject
CArray
CArrayObj
CSeries
CIndicator
CiFractals

クラスメソッド

作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Upper	上部バッファのバッファ要素を返します。
Lower	下部バッファのバッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear,

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string           symbol,      // シンボル
   ENUM_TIMEFRAMES  period       // 期間
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Upper

指定されたインデックスを持つ上部バッファのバッファ要素を返します。

```
double  Upper(
   int  index       // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

指定されたインデックスを持つ上部バッファのバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000–2025, MetaQuotes Ltd.

Lower

指定されたインデックスを持つ下部バッファのバッファ要素を返します。

```
double  Lower(
   int  index       // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

指定されたインデックスを持つ下部バッファのバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000–2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

Indicator type (IND_FRACTALS for CiFractals).

© 2000–2025, MetaQuotes Ltd.

CiGator

CiGator はゲーターオシレーターテクニカル指標の使用に意図されたクラスです。

説明

CiGator クラスは、ゲーターオシレーターテクニカル指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiGator: public CIndicator
```

タイトル

```
#include <Indicators\BillWilliams.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CIndicator
CiGator

クラスメソッド

属性	
JawPeriod	顎線の平均期間を返します。
JawShift	顎線の水平方向のシフトを返します。
TeethPeriod	歯線の平均期間を返します。
TeethShift	歯線の水平方向のシフトを返します。
LipsPeriod	口線の平均期間を返します。
LipsShift	口線の水平方向のシフトを返します。
MaMethod	平均化の方法を返します。
Applied	適用する型またはハンドルを返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Upper	上部バッファのバッファ要素を返します。
Lower	下部バッファのバッファ要素を返します。

© 2000-2025, MetaQuotes Ltd.

属性	
入出力	
virtual Type	オブジェクトの型の識別子を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

JawPeriod

顎線の平均期間を返します。

```
int  JawPeriod() const
```

戻り値

指標の作成時に定義された顎線の平均期間を返します。

© 2000–2025, MetaQuotes Ltd.

JawShift

顎線の水平方向のシフトを返します。

```
int  JawShift() const
```

戻り値

指標の作成時に定義された顎線の水平方向のシフト

© 2000–2025, MetaQuotes Ltd.

TeethPeriod

歯線の平均期間を返します。

```
int  TeethPeriod() const
```

戻り値

指標の作成時に定義された歯線の平均期間

© 2000–2025, MetaQuotes Ltd.

TeethShift

歯線の水平方向のシフトを返します。

```
int  TeethShift() const
```

戻り値

指標の作成時に定義された歯線の水平方向のシフト

© 2000–2025, MetaQuotes Ltd.

LipsPeriod

口線の平均期間を返します。

```
int  LipsPeriod() const
```

戻り値

指標の作成時に定義された口線の平均期間

© 2000–2025, MetaQuotes Ltd.

LipsShift

口線の水平方向のシフトを返します。

```
int  LipsShift() const
```

戻り値

指標の作成時に定義された口線の水平方向のシフト

© 2000–2025, MetaQuotes Ltd.

MaMethod

平均化の方法を返します。

```
ENUM_MA_METHOD  MaMethod() const
```

戻り値

指標の作成時に定義された平均化の方法

© 2000–2025, MetaQuotes Ltd.

Applied

適用する型またはハンドルを返します。

```
int  Applied() const
```

戻り値

指標の作成時に定義された、適用する型またはハンドル

© 2000–2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string           symbol,           // シンボル
   ENUM_TIMEFRAMES  period,           // 期間
   int              jaw_period,       // 顎線の期間
   int              jaw_shift,        // 顎線のシフト
   int              teeth_period,     // 歯線の期間
   int              teeth_shift,      // 歯線のシフト
   int              lips_period,      // 口線の期間
   int              lips_shift,       // 口線のシフト
   ENUM_MA_METHOD   ma_method,        // 平均化の方法
   int              applied           // 適用するボリュームの種類
  )
```

パラメータ

symbol
[in] シンボル

period
[in] 時間軸（ENUM_TIMEFRAMES 列挙）

jaw_period
[in] 顎線の期間

jaw_shift
[in] 顎線のシフト

teeth_period
[in] 歯線の期間

teeth_shift
[in] 歯線のシフト

lips_period
[in] 口線の期間

lips_shift
[in] 口線のシフト

ma_method
[in] 平均化の方法（ENUM_MA_METHOD 列挙）

applied
[in] 適用するボリュームの種類

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Upper

指定されたインデックスを持つ上部バッファのバッファ要素を返します。

```
double  Upper(
   int  index       // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

指定されたインデックスを持つ上部バッファのバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000–2025, MetaQuotes Ltd.

Lower

指定されたインデックスを持つ下部バッファのバッファ要素を返します。

```
double  Upper(
   int  index       // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

指定されたインデックスを持つ下部バッファのバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000–2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

Indicator type (IND_GATOR for CiGator).

© 2000–2025, MetaQuotes Ltd.

CiBWMFI

CiBWMFI は BW MFI（ビルウィリアムズのマーケットファシリテーションインデックス）テクニカル指標の使用に意図されたクラスです。

説明

CiBWMFI クラスは、BW MFI（ビルウィリアムズのマーケットファシリテーションインデックス）テクニカル指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiBWMFI: public CIndicator
```

タイトル

```
#include <Indicators\BillWilliams.mqh>
```

継承階層

CObject
CArray
CArrayObj
CSeries
CIndicator
CiBWMFI

クラスメソッド

属性	
Applied	適用するボリュームの種類を返します。
作成メソッド	
Create	指標を作成します。
データアクセスメソッド	
Main	バッファ要素を返します。
入出力	
virtual Type	オブジェクトの型の識別子を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Compare

クラスから継承されたメソッド CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

クラスから継承されたメソッド CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

クラスから継承されたメソッド CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

Applied

適用するボリュームの種類を返します。

```
ENUM_APPLIED_VOLUME  Applied() const
```

戻り値

指標の作成時に定義された、適用するボリュームの種類

© 2000–2025, MetaQuotes Ltd.

Create

指定されたパラメータで指標を作成します。指標値の更新と取得には Refresh() 及び GetData() が使用されます。

```
bool  Create(
   string                symbol,      // シンボル
   ENUM_TIMEFRAMES       period,      // 期間
   ENUM_APPLIED_VOLUME   applied      // 適用するボリュームの種類
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

applied

[in] 適用するボリュームの種類（ENUM_APPLIED_VOLUME 列挙）

戻り値

成功の場合は true、指標が作成できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

Main

指定されたインデックスを持つバッファ要素を返します。

```
double  Main(
   int  index      // インデックス
  )
```

パラメータ

index

[in] 要素のインデックス

戻り値

成功の場合は指定されたインデックスを持つバッファ要素、または正当なデータが不在な場合は EMPTY_VALUE

© 2000–2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

Indicator type (IND_BWMFI for CiBWMFI).

© 2000–2025, MetaQuotes Ltd.

CiCustom

CiCustom はカスタムテクニカル指標の使用に意図されたクラスです。

説明

CiCustom クラスは、カスタム指標の作成、設定及びデータアクセスを提供します。

宣言

```
class CiCustom: public CIndicator
```

タイトル

```
#include <Indicators\Custom.mqh>
```

クラスメソッド

属性	
NumBuffers	バッファ数を設定します。
NumParams	パラメータ数を取得します。
ParamType	指定されたパラメータの型を取得します。
ParamLong	指定された整数型パラメータの値を取得します。
ParamDouble	指定された double 型パラメータの値を取得します。
ParamString	指定された string 型パラメータの値を取得します。
入出力	
virtual Type	オブジェクトの型の認識しを取得します。

© 2000-2025, MetaQuotes Ltd.

NumBuffers

バッファ数を設定します。

```
bool  NumBuffers(
   int  buffers      // バッファ数
  )
```

戻り値

成功の場合は true、バッファが設定されなかった場合は false

© 2000–2025, MetaQuotes Ltd.

NumParams

パラメータ数を取得します。

```
int  NumParams() const
```

戻り値

指標作成に使用されるパラメータ数

© 2000–2025, MetaQuotes Ltd.

ParamType

指定されたインデックスのパラメータの型を取得します。

```
ENUM_DATATYPE  ParamType(
   int  index      // パラメータインデックス
   ) const
```

パラメータ

index

[in] パラメータインデックス

戻り値

指標作成に使用される、指定されたインデックスのパラメータの型（ENUM_DATATYPE 列挙値）

注意事項

パラメータインデックスが不正の場合 WRONG_VALUE が返されます。

© 2000–2025, MetaQuotes Ltd.

ParamLong

指定された long 型パラメータの値を取得します。

```
long  ParamLong(
   int  index      // インデックス
   ) const
```

パラメータ

index

[in] パラメータインデックス

戻り値

指標作成に使用される、指定された long 型パラメータの値

注意事項

値が不正の場合 0 が返されます。

© 2000–2025, MetaQuotes Ltd.

ParamDouble

指定された double 型パラメータの値を取得します。

```
double  ParamDouble(
   int  index      // インデックス
   ) const
```

パラメータ

index

[in] パラメータインデックス

戻り値

指標作成に使用される、指定された double 型パラメータの値

注意事項

値が不正の場合 EMPTY_VALUE が返されます。

© 2000–2025, MetaQuotes Ltd.

ParamString

指定された string 型パラメータの値を取得します。

```
string  ParamString(
   int  index      // インデックス
   ) const
```

パラメータ

index

[in] パラメータインデックス

戻り値

指標作成に使用される、指定された string 型パラメータの値

注意事項

値が不正の場合、空の文字列が返されます。

© 2000-2025, MetaQuotes Ltd.

Type

オブジェクトの型の識別子を返します。

```
virtual int  Type() const
```

戻り値

Indicator type (IND_CUSTOM for CiCustom).

© 2000–2025, MetaQuotes Ltd.

取引クラス

このセクションには、取引クラスの操作に関する技術的な詳細と関連するMQL5 標準ライブラリコンポーネントの説明が含まれています。

これらのクラスの使用は、カスタムプログラム(エキスパートアドバイザ)の作成時に時間を節約します。

MQL5 標準ライブラリの取引クラスに関連したファイルは、端末ディレクトリの Include¥Trade フォルダ内に配置されています。

クラス/グループ	説明
CAccountInfo	取引口座プロパティ操作に使用されるクラス
CSymbolInfo	金融商品のプロパティ操作に使用されるクラス
COrderInfo	未決注文のプロパティ操作に使用されるクラス
CHistoryOrderInfo	履歴注文のプロパティ操作に使用されるクラス
CPositionInfo	ポジションのプロパティ操作に使用されるクラス
CDealInfo	履歴注文のプロパティ操作に使用されるクラス
CTrade	取引操作実行に使用されるクラス
CTerminalInfo	MQL5 プログラム環境取得に使用されるクラス

© 2000–2025, MetaQuotes Ltd.

CAccountInfo

CAccountInfo は現在開いている取引口座プロパティに簡単にアクセスするためのクラスです。

説明

CAccountInfo クラスは現在開いている取引口座プロパティへの容易化したアクセスを提供します。

宣言

```
class CAccountInfo : public CObject
```

タイトル

```
#include <Trade\AccountInfo.mqh>
```

継承階層

CObject
  CAccountInfo

グループ分けされたクラスメソッド

整数型プロパティへのアクセス	
Login	口座番号を取得します。
TradeMode	取引モードを取得します。
TradeModeDescription	取引モードを文字列として取得します。
Leverage	与えられたレバレッジの額を取得します。
StopoutMode	口座ストップアウトのモードを取得します。
StopoutModeDescription	口座ストップアウトのモードの説明を取得します。
TradeAllowed	取引許可のフラグを取得します。
TradeExpert	自動売買許可のフラグを取得します。
LimitOrders	未決注文の最大限を取得します。
MarginMode	証拠金計算モードを取得します。
MarginModeDescription	文字列として証拠金計算モードを取得します。
double 型プロパティへのアクセス	
Balance	口座の残高を取得します。
Credit	与えられたクレジットの額を取得します。
Profit	口座の現在の利益の額を取得します。

© 2000-2025, MetaQuotes Ltd.

整数型プロパティへのアクセス	
Equity	口座の現在の株式の額を取得します。
Margin	予約された証拠金の額を取得します。
FreeMargin	余剰証拠金の額を取得します。
MarginLevel	証拠金のレベルを取得します。
MarginCall	預金のための証拠金のレベルを取得します。
MarginStopOut	ストップアウトのための証拠金のレベルを取得します。
テキストプロパティへのアクセス	
Name	クライアント名を取得します。
Server	取引サーバ名を取得します。
Currency	預金通貨名を取得します。
Company	口座を提供してる会社名を取得します。
MQL5 API 関数へのアクセス	
InfoInteger	指定された整数型プロパティの値を取得します。
InfoDouble	指定された double 型プロパティの値を取得します。
InfoString	指定された string 型プロパティの値を取得します。
追加メソッド	
OrderProfitCheck	渡されたパラメータに基づいて評価利益を取得します。
MarginCheck	取引処理を実行するのに必要な証拠金の量を取得します。
FreeMarginCheck	取引操作の実行後に残った余剰証拠金の量を取得します。
MaxLotCheck	取引操作のボリュームの最大限を取得します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000–2025, MetaQuotes Ltd.

Login

口座番号を取得します。

```
long  Login() const
```

戻り値

口座番号

© 2000–2025, MetaQuotes Ltd.

TradeMode

取引モードを取得します。

```
ENUM_ACCOUNT_TRADE_MODE  TradeMode() const
```

戻り値

取引モード（ENUM_ACCOUNT_TRADE_MODE 列挙値）

© 2000–2025, MetaQuotes Ltd.

TradeModeDescription

取引モードを文字列として取得します。

```
string  TradeModeDescription() const
```

戻り値

文字列としての取引モード

© 2000–2025, MetaQuotes Ltd.

Leverage

与えられたレバレッジの額を取得します。

```
long  Leverage() const
```

戻り値

与えられたレバレッジの額

© 2000–2025, MetaQuotes Ltd.

StopoutMode

口座ストップアウトのモードを取得します。

```
ENUM_ACCOUNT_STOPOUT_MODE  StopoutMode() const
```

戻り値

口座ストップアウトのモード（ENUM_ACCOUNT_STOPOUT_MODE 列挙の値）

© 2000–2025, MetaQuotes Ltd.

StopoutModeDescription

最低限の証拠金レベルを設定するモードを文字列として取得します。

```
string  StopoutModeDescription() const
```

戻り値

文字列としての最低限の証拠金レベルを設定するモード

© 2000–2025, MetaQuotes Ltd.

MarginMode

証拠金計算モードを取得します。

```
ENUM_ACCOUNT_MARGIN_MODE  MarginMode() const
```

戻り値

ENUM_ACCOUNT_MARGIN_MODE列挙からの証拠金計算モード。

© 2000–2025, MetaQuotes Ltd.

MarginModeDescription

文字列として証拠金計算モードを取得します。

```
string  MarginModeDescription() const
```

戻り値

文字列としての証拠金計算モード。

© 2000–2025, MetaQuotes Ltd.

TradeAllowed

取引許可のフラグを取得します。

```
bool  TradeAllowed() const
```

戻り値

取引許可のフラグ

© 2000-2025, MetaQuotes Ltd.

TradeExpert

自動売買許可のフラグを取得します。

```
bool  TradeExpert() const
```

戻り値

自動売買許可のフラグ

© 2000–2025, MetaQuotes Ltd.

LimitOrders

未決注文の最大限を取得します。

```
int  LimitOrders() const
```

戻り値

未決注文の最大限

注意事項

0 – 制限なし

© 2000–2025, MetaQuotes Ltd.

Balance

口座の残高を取得します。

```
double  Balance() const
```

戻り値

口座の預金通貨での口座の残高

© 2000–2025, MetaQuotes Ltd.

Credit

与えられたクレジットの額を取得します。

```
double  Credit() const
```

戻り値

口座の預金通貨での与えられたクレジットの額

© 2000–2025, MetaQuotes Ltd.

Profit

口座の現在の利益の額を取得します。

```
double  Profit() const
```

戻り値

口座の預金通貨での現在の利益額

© 2000–2025, MetaQuotes Ltd.

Equity

口座の現在の株式の額を取得します。

```
double  Equity() const
```

戻り値

口座の預金通貨での口座の現在の株式の額

© 2000–2025, MetaQuotes Ltd.

Margin

予約された証拠金の額を取得します。

```
double  Margin() const
```

戻り値

口座の預金通貨での予約された証拠金の額

© 2000–2025, MetaQuotes Ltd.

FreeMargin

余剰証拠金の額を取得します。

```
double  FreeMargin() const
```

戻り値

口座の預金通貨での余剰証拠金の額

© 2000–2025, MetaQuotes Ltd.

MarginLevel

証拠金のレベルを取得します。

```
double  MarginLevel() const
```

戻り値

証拠金のレベル

© 2000–2025, MetaQuotes Ltd.

MarginCall

預金のための証拠金のレベルを取得します。

```
double  MarginCall() const
```

戻り値

預金のための証拠金のレベル.

© 2000-2025, MetaQuotes Ltd.

MarginStopOut

ストップアウトのための証拠金のレベルを取得します。

```
double  MarginStopOut() const
```

戻り値

ストップアウトのための証拠金のレベル

© 2000–2025, MetaQuotes Ltd.

Name

クライアント名を取得します。

```
string  Name() const
```

戻り値

クライアント名

© 2000–2025, MetaQuotes Ltd.

Server

取引サーバ名を取得します。

```
string  Server() const
```

戻り値

取引サーバ名

© 2000–2025, MetaQuotes Ltd.

Currency

預金通貨名を取得します。

```
string  Currency() const
```

戻り値

預金通貨名

© 2000–2025, MetaQuotes Ltd.

Company

口座を提供している会社名を取得します。

```
string  Company() const
```

戻り値

口座を提供している会社の名称

© 2000–2025, MetaQuotes Ltd.

InfoInteger

指定された整数型プロパティの値を取得します。

```
long  InfoInteger(
   ENUM_ACCOUNT_INFO_INTEGER  prop_id     // プロパティ識別子
   ) const
```

パラメータ

prop_id

[in] プロパティの識別子。可能な値はf ENUM_ACCOUNT_INFO_INTEGER 列挙のいずれかです

戻り値

long型の値

© 2000–2025, MetaQuotes Ltd.

InfoDouble

指定された double 型プロパティの値を取得します。

```
double  InfoDouble(
   ENUM_ACCOUNT_INFO_DOUBLE  prop_id     // プロパティ識別子
   ) const
```

パラメータ

prop_id

[in] プロパティの識別子。可能な値は ENUM_ACCOUNT_INFO_DOUBLE 列挙のいずれかです。

戻り値

double 型の値

© 2000–2025, MetaQuotes Ltd.

InfoString

指定された string 型プロパティの値を取得します。

```
string  InfoString(
   ENUM_ACCOUNT_INFO_STRING  prop_id     // プロパティ識別子
   ) const
```

パラメータ

prop_id

[in] プロパティの識別子。可能な値は ENUM_ACCOUNT_INFO_STRING 列挙のいずれかです。

戻り値

string 型の値

© 2000–2025, MetaQuotes Ltd.

OrderProfitCheck

この関数は、渡されたパラメータに基づいて、現在の市況での現在の口座の利益を計算します。これは、取引操作の結果の事前評価のために使用されます。値は口座の預金通貨で返されます。

```
double  OrderProfitCheck(
   const string        symbol,              // シンボル
   ENUM_ORDER_TYPE     trade_operation,     // 操作の種類(ORDER_TYPE_BUY または ORDER_TYPE_SELL)
   double              volume,              // ボリューム
   double              price_open,          // 始値
   double              price_close          // 終値
   ) const
```

パラメータ

symbol

[in] 取引操作のシンボル

trade_operation

[in] 取引操作の型（ENUM_ORDER_TYPE 列挙）

ボリューム

[in] 取引操作のボリューム

price_open

[in] 始値

price_close

[in] 終値

戻り値

成功の場合は利益の額、エラーの場合は EMPTY_VALUE

© 2000-2025, MetaQuotes Ltd.

MarginCheck

取引処理を実行するのに必要な証拠金の量を取得します。

```
double  MarginCheck(
   const string         symbol,              // シンボル
   ENUM_ORDER_TYPE      trade_operation,     // 操作
   double               volume,              // ボリューム
   double               price                // 価格
   ) const
```

パラメータ

symbol

[in] 取引操作のシンボル

trade_operation

[in] 取引操作の型（ENUM_ORDER_TYPE 列挙）

ボリューム

[in] 取引操作のボリューム

価格

[in] 取引操作の価格

戻り値

取引処理を実行するのに必要な証拠金の量

© 2000–2025, MetaQuotes Ltd.

FreeMarginCheck

取引操作の実行後に残った余剰証拠金の量を取得します。

```
double  FreeMarginCheck(
   const string         symbol,              // シンボル
   ENUM_ORDER_TYPE      trade_operation,     // 操作
   double               volume,              // ボリューム
   double               price                // 価格
   ) const
```

パラメータ

symbol

[in] 取引操作のシンボル

trade_operation

[in] 取引操作の型（ENUM_ORDER_TYPE 列挙）

ボリューム

[in] 取引操作のボリューム

価格

[in] 取引操作の価格

戻り値

取引操作の実行後に残った余剰証拠金の量

© 2000–2025, MetaQuotes Ltd.

MaxLotCheck

取引操作のボリュームの最大限を取得します。

```
double  MaxLotCheck(
   const string         symbol,              // シンボル
   ENUM_ORDER_TYPE      trade_operation,     // 操作
   double               price,               // 価格
   double               percent=100          // 取引操作に使用された証拠金の百分率での割合
( 0~100%)
   ) const
```

パラメータ

symbol

[in] 取引操作のシンボル

trade_operation

[in] 取引操作の型（ENUM_ORDER_TYPE 列挙）

価格

[in] 取引操作の価格

percent=100

[in] 取引操作に使用された証拠金の百分率での割合（0~100%）

戻り値

取引操作のボリュームの最大限

© 2000-2025, MetaQuotes Ltd.

CSymbolInfo

CSymbolInfo はシンボルのプロパティに簡単にアクセスするためのクラスです。

説明

CSymbolInfo クラスはシンボルプロパティへの簡易化されたアクセスを提供します。

宣言

```
class CSymbolInfo : public CObject
```

タイトル

```
#include <Trade\SymbolInfo.mqh>
```

継承階層

CObject

CSymbolInfo

グループ分けされたクラスメソッド

Controlling	
Refresh	シンボルデータを更新します。
RefreshRates	シンボルの相場を更新します。
プロパティ	
Name	銘柄名を取得/設定します。
Select	「気配値表示」シンボルフラグを取得/設定します。
IsSynchronized	シンボルのサーバとの同期をチェックします。
ボリューム。	
Volume	直近約定ボリュームを取得します。
VolumeHigh	一日の最大ボリュームを取得します。
VolumeLow	一日の最小ボリュームを取得します。
Miscellaneous	
Time	最終の相場の時刻を取得します。
Spread	スプレッドの額を取得します(ポイント単位)。
SpreadFloat	変動スプレッドのフラグを取得します。
TicksBookDepth	ティック保存の数を取得します。
Levels	

© 2000-2025, MetaQuotes Ltd.

Controlling	
StopsLevel	受注のための最小限のインデントを取得します（ポイント単位）。
FreezeLevel	取引凍結の操作の距離を取得します（ポイント単位）。
売値	
Bid	現在の売値を取得します。
BidHigh	一日の最高売値を取得します。
BidLow	一日の最低売値を取得します。
買値	
Ask	現在の買値を取得します。
AskHigh	一日の最高買値を取得します。
AskLow	一日の最低買値を取得します。
Prices	
Last	現在の直近約定値を取得します。
LastHigh	一日の最高終値を取得します。
LastLow	一日の最低終値を取得します。
取引モード	
TradeCalcMode	契約コスト計算のモードを取得します。
TradeCalcModeDescription	契約コスト計算のモードを文字列として取得します。
TradeMode	注文実行の種類を取得します。
TradeModeDescription	注文実行の種類を文字列として取得します。
TradeExecution	約定決済のモードを取得します。
TradeExecutionDescription	約定決済のモードを文字列として取得します。
スワップ	
SwapMode	スワップ計算モデルを取得します。
SwapModeDescription	スワップ計算モデルを文字列として取得します。
SwapRollover3days	トリップスワップ料金の日を取得します。
SwapRollover3daysDescription	トリップスワップ料金の日を文字列として取得します。
証拠金とフラグ	
MarginInitial	初期証拠金の値を取得します。
MarginMaintenance	維持証拠金の値を取得します。
MarginLong	ロング（買い）ポジションの証拠金の料金を取得します。

© 2000–2025, MetaQuotes Ltd.

Controlling	
MarginShort	ショート（売り）ポジションの証拠金の料金を取得します。
MarginLimit	リミット注文の証拠金の料金を取得します。
MarginStop	ストップ注文の証拠金の料金を取得します。
MarginStopLimit	指値つきストップ注文の証拠金の料金を取得します。
TradeTimeFlags	許可された注文期限のモードのフラグを取得します。
TradeFillFlags	許可された注文充填のモードのフラグを取得します。
量子化	
Digits	小数点以下の桁数を取得します。
Point	1 ポイントの値を取得します。
TickValue	ティックのコスト（価格の最小限の変更）を取得します。
TickValueProfit	収益性の高いポジションのために計算されたティックの価格を取得します。
TickValueLoss	収益性の低いポジションのために計算されたティックの価格を取得します。
TickSize	価格の最小限の変更を取得します。
契約サイズ	
ContractSize	取引契約の量を取得します。
LotsMin	約定の決済に必要な最小ボリュームを取得します。
LotsMax	注文の決済に必要な最大ボリュームを取得します。
LotsStep	約定の決済に必要なボリューム変化の最小ステップを取得します。
LotsLimit	1 シンボルのポジション及び未決注文（両方向）の最大限ボリュームを取得します。
スワップサイズ	
SwapLong	買いポジションスワップの値を取得します。
SwapShort	売りポジションスワップの値を取得します。
テキストプロパティ	
CurrencyBase	シンボルの基本通貨の名称を取得します。
CurrencyProfit	利益の通貨の名称を取得します。
CurrencyMargin	証拠金の通貨の名称を取得します。
Bank	現在の相場の元の名称を取得します。
Description	シンボルの文字列での説明を取得します。

© 2000–2025, MetaQuotes Ltd.

Controlling	
Path	シンボルツリーでのパスを取得します。
シンボルプロパティ	
SessionDeals	現在のセッションでの約定の数を取得します。
SessionBuyOrders	現時点での買い注文の数を取得します。
SessionSellOrders	現時点での売り注文の数を取得します。
SessionTurnover	現在のセッションの売上高の要約を取得します。
SessionInterest	現在のセッションの未決済の建玉の概要を取得します。
SessionBuyOrdersVolume	現在の買い注文のボリュームを取得します。
SessionSellOrdersVolume	現在の売り注文のボリュームを取得します。
SessionOpen	現在のセッションの始値を取得します。
SessionClose	現在のセッションの終値を取得します。
SessionAW	現在のセッションの加重平均価格を取得します。
SessionPriceSettlement	現在のセッションの決済価格を取得します。
SessionPriceLimitMin	現在のセッションの最小値を取得します。
SessionPriceLimitMax	現在のセッションの最大値を取得します。
MQL5 API 関数へのアクセス	
InfoInteger	指定された整数型プロパティの値を取得します。
InfoDouble	指定された double 型プロパティの値を取得します。
InfoString	指定された string 型プロパティの値を取得します。
サービス関数	
NormalizePrice	シンボルプロパティを使用して正規化された価格の値を返します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000–2025, MetaQuotes Ltd.

Refresh

シンボルデータを更新します。

```
void  Refresh()
```

戻り値

なし

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

RefreshRates

シンボルの相場を更新します。

```
bool  RefreshRates()
```

戻り値

成功の場合は true、相場が更新できなかった場合は false

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Name

銘柄名を取得します。

```
string  Name() const
```

戻り値

銘柄名

Name

銘柄名を設定します。

```
bool  Name(string name)
```

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

Select

「気配値表示」シンボルフラグを取得します。

```
bool  Select() const
```

戻り値

「気配値表示」シンボルフラグ

Select

「気配値表示」シンボルフラグを設定します。

```
bool  Select()
```

戻り値

成功の場合は true、フラグが変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

IsSynchronized

シンボルのサーバとの同 期 をチェックします。

```
bool  IsSynchronized() const
```

戻 り値

シンボルがサーバと同 期 されている場 合 は true、それ以 外 の場 合 は false

注 意 事 項

シンボルは Name メソッド で選 択 されるべきです。

© 2000–2025, MetaQuotes Ltd.

Volume

直近約定ボリュームを取得します。

```
long  Volume() const
```

戻り値

直近約定ボリューム

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

VolumeHigh

一日の最大ボリュームを取得します。

```
long  VolumeHigh() const
```

戻り値

一日の最大ボリューム

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

VolumeLow

一日の最小ボリュームを取得します。

```
long  VolumeLow() const
```

戻り値

一日の最小ボリューム

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Time

最終の相場の時刻を取得します。

```
datetime  Time() const
```

戻り値

最終の相場の時刻

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Spread

スプレッドの額を取得します（ポイント単位）。

```
int  Spread() const
```

戻り値

スプレッドの額を取得します（ポイント単位）。

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

SpreadFloat

変動スプレッドのフラグを取得します。

```
bool  SpreadFloat() const
```

戻り値

変動スプレッドのフラグ

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TicksBookDepth

ティック保存の数を取得します。

```
int  TicksBookDepth() const
```

戻り値

ティック保存の数

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

StopsLevel

注文の最小限のストップレベルを取得します（ポイント単位）。

```
int  StopsLevel() const
```

戻り値

注文の最小限のストップレベル

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

FreezeLevel

凍結レベルを取得します（ポイント単位）。

```
int  FreezeLevel() const
```

戻り値

凍結レベルの距離（ポイント単位）

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Bid

現在の売値を取得します。

```
double  Bid() const
```

戻り値

現在の売値

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000-2025, MetaQuotes Ltd.

BidHigh

一日の最高売値を取得します。

```
double  BidHigh() const
```

戻り値

一日の最高売値.

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000-2025, MetaQuotes Ltd.

BidLow

一日の最低売値を取得します。

```
double  BidLow() const
```

戻り値

一日の最低売値

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000-2025, MetaQuotes Ltd.

Ask

現在の買値を取得します。

```
double  Ask() const
```

戻り値

現在の買値

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000-2025, MetaQuotes Ltd.

AskHigh

一日の最高買値を取得します。

```
double  AskHigh() const
```

戻り値

一日の最高買値

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000-2025, MetaQuotes Ltd.

AskLow

一日の最低買値を取得します。

```
double  AskLow() const
```

戻り値

一日の最低買値.

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000-2025, MetaQuotes Ltd.

Last

現在の直近約定値を取得します。

```
double  Last() const
```

戻り値

現在の直近約定値

© 2000-2025, MetaQuotes Ltd.

LastHigh

一日の最高終値を取得します。

```
double  LastHigh() const
```

戻り値

一日の最高終値

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000-2025, MetaQuotes Ltd.

LastLow

一日の最低終値を取得します。

```
double  LastLow() const
```

戻り値

一日の最低終値

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TradeCalcMode

契約コスト計算のモードを取得します。

```
ENUM_SYMBOL_CALC_MODE  TradeCalcMode() const
```

戻り値

契約コスト計算のモード（ENUM_SYMBOL_CALC_MODE 列挙値）

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TradeCalcModeDescription

契約コスト計算のモードを文字列として取得します。

```
string  TradeCalcModeDescription() const
```

戻り値

文字列としての契約コスト計算のモード

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TradeMode

注文実行の種類を取得します。

```
ENUM_SYMBOL_TRADE_MODE  TradeMode() const
```

戻り値

注文実行の種類（ENUM_SYMBOL_TRADE_MODE 列挙値）

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TradeModeDescription

取引モードを文字列として取得します。

```
string  TradeModeDescription() const
```

戻り値

文字列としての取引モード

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000-2025, MetaQuotes Ltd.

TradeExecution

取引モードを取得します。

```
ENUM_SYMBOL_TRADE_EXECUTION  TradeExecution() const
```

戻り値

取引モード（ENUM_SYMBOL_TRADE_EXECUTION 列挙値）

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TradeExecutionDescription

取引モードを文字列として取得します。

```
string  TradeExecutionDescription() const
```

戻り値

文字列としての取引モード

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

SwapMode

スワップ計算モデルを取得します。

```
ENUM_SYMBOL_SWAP_MODE  SwapMode() const
```

戻り値

スワップ計算モデルを取得します。（ENUM_SYMBOL_SWAP_MODE 列挙値）

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

SwapModeDescription

スワップ計算モデルを文字列として取得します。

```
string  SwapModeDescription() const
```

戻り値

文字列としてのスワップ計算モデル

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

SwapRollover3days

スワップロールオーバー日を取得します。

```
ENUM_DAY_OF_WEEK  SwapRollover3days() const
```

戻り値

スワップロールオーバー日（ENUM_DAY_OF_WEEK 列挙値）

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

SwapRollover3daysDescription

スワップロールオーバー日を文字列として取得します。

```
string  SwapRollover3daysDescription() const
```

戻り値

文字列としてのスワップロールオーバー日

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

MarginInitial

初期証拠金の値を取得します。

```
double  MarginInitial()
```

戻り値

初期証拠金の値

注意事項

1 ロットから課金される、シンボルの証拠金通貨での証拠金量を返します。これは、クライアントが市場に入った時に、資産をチェックするために使用されます。

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

MarginMaintenance

維持証拠金の値を取得します。

```
double  MarginMaintenance()
```

戻り値

維持証拠金の値

注意事項

1　ロットから課金される、シンボルの証拠金通貨での証拠金量を返します。これは、クライアントの口座状態が変わった時に、資産をチェックするために使用されます。維持証拠金が 0 に等しい場合、当初証拠金が使用されます。

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

MarginLong

ロング（買い）ポジションの証拠金の料金を取得します。

```
double  MarginLong() const
```

戻り値

ロング（買い）ポジションの証拠金の料金

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

MarginShort

ショート（売り）ポジションの証拠金の料金を取得します。

```
double  MarginShort() const
```

戻り値

ショート（売り）ポジションの証拠金の料金

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

MarginLimit

リミット注文の証拠金の料金を取得します。

```
double  MarginLimit() const
```

戻り値

リミット注文の証拠金の料金

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

MarginStop

ストップ注文の証拠金の料金を取得します。

```
double  MarginStop() const
```

戻り値

ストップ注文の証拠金の料金

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

MarginStopLimit

指値つきストップ注文の証拠金の料金を取得します。

```
double  MarginStopLimit() const
```

戻り値

指値つきストップ注文の証拠金の料金

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TradeTimeFlags

許可された注文期限のモードのフラグを取得します。

```
int  TradeTimeFlags() const
```

戻り値

許可された注文期限のモード

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000-2025, MetaQuotes Ltd.

TradeFillFlags

許可された注文充填のモードのフラグを取得します。

```
int  TradeFillFlags() const
```

戻り値

許可された注文充填のモード

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000-2025, MetaQuotes Ltd.

Digits

小数点以下の桁数を取得します。

```
int  Digits() const
```

戻り値

小数点以下の桁数を取得します。

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Point

1 ポイントの値を取得します。

```
double  Point() const
```

戻り値

1 ポイントの値

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TickValue

ティックのコスト（価格の最小限の変更）を取得します。

```
double  TickValue() const
```

戻り値

ティックのコスト（価格の最小限の変更）

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TickValueProfit

収益性の高いポジションのために計算されたティックの価格を取得します。

```
double  TickValueProfit() const
```

戻り値

収益性の高いポジションのために計算されたティックの価格

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TickValueLoss

収益性の低いポジションのために計算されたティックの価格を取得します。

```
double  TickValueLoss() const
```

戻り値

収益性の低いポジションのために計算されたティックの価格

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TickSize

価格の最小限の変更を取得します。

```
double  TickSize() const
```

戻り値

価格の最小限の変更

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

ContractSize

取引契約の量を取得します。

```
double  ContractSize() const
```

戻り値

取引契約の量

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

LotsMin

約定の決済に必要な最小ボリュームを取得します。

```
double  LotsMin() const
```

戻り値

約定の決済に必要な最小ボリューム

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

LotsMax

約定の決済に必要な最大ボリュームを取得します。

```
double  LotsMax() const
```

戻り値

約定の決済に必要な最大ボリューム

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

LotsStep

約定の決済に必要なボリューム変化の最小ステップを取得します。

```
double  LotsStep() const
```

戻り値

約定の決済に必要なボリューム変化の最小ステップ

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

LotsLimit

シンボルのポジション及び未決注文（両方向）の最大限ボリュームを取得します。

```
double  LotsLimit() const
```

戻り値

シンボルのポジション及び未決注文（両方向）の最大限ボリューム

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

SwapLong

買いポジションスワップの値を取得します。

```
double  SwapLong() const
```

戻り値

買いポジションスワップの値

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000-2025, MetaQuotes Ltd.

SwapShort

売りポジションスワップの値を取得します。

```
double  SwapShort() const
```

戻り値

売りポジションスワップの値

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

CurrencyBase

シンボルの基本通貨の名称を取得します。

```
string  CurrencyBase() const
```

戻り値

シンボルの基本通貨の名称

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

CurrencyProfit

利益の通貨の名称を取得します。

```
string  CurrencyProfit() const
```

戻り値

利益の通貨の名称

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

CurrencyMargin

証拠金の通貨の名称を取得します。

```
string  CurrencyMargin() const
```

戻り値

証拠金の通貨の名称

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Bank

現在の相場の元の名称を取得します。

```
string  Bank() const
```

戻り値

現在の相場の元の名称

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Description

シンボルの文字列での説明を取得します。

```
string  Description() const
```

戻り値

シンボルの文字列での説明

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Path

シンボルツリーでのパスを取得します。

```
string  Path() const
```

戻り値

シンボルツリーでのパスを取得します。

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000-2025, MetaQuotes Ltd.

SessionDeals

現在のセッションでの約定の数を取得します。

```
long  SessionDeals() const
```

戻り値

現在のセッションでの約定の数

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

SessionBuyOrders

現時点での買い注文の数を取得します。

```
long  SessionBuyOrders() const
```

戻り値

現時点での買い注文の数

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

SessionSellOrders

現時点での売り注文の数を取得します。

```
long  SessionSellOrders() const
```

戻り値

現時点での売り注文の数

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000-2025, MetaQuotes Ltd.

SessionTurnover

現在のセッションの売上高の要約を取得します。

```
double  SessionTurnover() const
```

戻り値

現在のセッションの売上高の要約

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

SessionInterest

現在のセッションの未決済の建玉の概要を取得します。

```
double  SessionInterest() const
```

戻り値

現在のセッションの未決済の建玉の概要

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

SessionBuyOrdersVolume

現在の買い注文のボリュームを取得します。

```
double  SessionBuyOrdersVolume() const
```

戻り値

現在の買い注文のボリューム

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000-2025, MetaQuotes Ltd.

SessionSellOrdersVolume

現在の売り注文のボリュームを取得します。

```
double  SessionSellOrdersVolume() const
```

戻り値

現在の売り注文のボリューム

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000-2025, MetaQuotes Ltd.

SessionOpen

現在のセッションの始値を取得します。

```
double  SessionOpen() const
```

戻り値

現在のセッションの始値

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

SessionClose

現在のセッションの終値を取得します。

```
double  SessionClose() const
```

戻り値

現在のセッションの終値

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

SessionAW

現在のセッションの加重平均価格を取得します。

```
double  SessionAW() const
```

戻り値

現在のセッションの加重平均価格

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

SessionPriceSettlement

現在のセッションの決済価格を取得します。

```
double  SessionPriceSettlement() const
```

戻り値

現在のセッションの決済価格

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

SessionPriceLimitMin

現在のセッションの最小値を取得します。

```
double  SessionPriceLimitMin() const
```

戻り値

現在のセッションの最小値

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

SessionPriceLimitMax

現在のセッションの最大値を取得します。

```
double  SessionPriceLimitMax() const
```

戻り値

現在のセッションの最大値

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000-2025, MetaQuotes Ltd.

InfoInteger

指定された整数型プロパティの値を取得します。

```
bool  InfoInteger(
   ENUM_SYMBOL_INFO_INTEGER  prop_id,      // プロパティ識別子
   long&                     var           // 変数への参照
   ) const
```

パラメータ

prop_id

[in] 整数型プロパティの識別子（ENUM_SYMBOL_INFO_INTEGER 列挙値）

var

[out] 結果を受け取る long 型変数への参照

戻り値

成功の場合は true、プロパティ値が取得できなかった場合は false

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

InfoDouble

指定された double 型プロパティの値を取得します。

```
bool  InfoDouble(
   ENUM_SYMBOL_INFO_DOUBLE  prop_id,      // プロパティ識別子
   double&                  var           // 変数への参照
   ) const
```

パラメータ

prop_id

[in] double 型プロパティの識別子（ENUM_SYMBOL_INFO_DOUBLE 列挙値）

var

[out] 結果を受け取る double 型変数への参照

戻り値

成功の場合は true、プロパティ値が取得できなかった場合は false

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

InfoString

指定された string 型プロパティの値を取得します。

```
bool  InfoString(
   ENUM_SYMBOL_INFO_STRING  prop_id,      // プロパティ識別子
   string&                  var           // 変数への参照
   ) const
```

パラメータ

prop_id

[in] テキストプロパティの識別子

var

[out] 結果を受け取る string 型変数への参照

戻り値

成功の場合は true、プロパティ値が取得できなかった場合は false

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

NormalizePrice

シンボルプロパティを使用して正規化された価格の値を返します。

```
double  NormalizePrice(
   double  price   // 価格
   ) const
```

パラメータ

価格

[in] 価格

戻り値

正規化された価格

注意事項

シンボルは Name メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

COrderInfo

COrderInfo は未決注文のプロパティに簡単にアクセスするためのクラスです。

説明

COrderInfo クラスは未決注文のプロパティへの簡易化されたアクセスを提供します。

宣言

```
class COrderInfo : public CObject
```

タイトル

```
#include <Trade\OrderInfo.mqh>
```

継承階層

CObject
  COrderInfo

グループ分けされたクラスメソッド

整数型プロパティへのアクセス	
Ticket	事前にアクセスに選択された注文のチケットを取得します。
TimeSetup	発注時刻を取得します。
TimeSetupMsc	01.01.1970から経過したミリ秒数で表された発注時刻
OrderType	注文の種類を取得します。
OrderTypeDescription	注文の種類を文字列として取得します。
State	注文状態を取得します。
StateDescription	注文状態を文字列として取得します。
TimeExpiration	注文期限を取得します。
TimeDone	注文実行または取り消しの時刻を取得します。
TimeDoneMsc	01.01.1970　から経ったミリ秒で表された注文の実行/取り消し時刻を受け取ります。
TypeFilling	余りによって注文実行の種類を取得します。
TypeFillingDescription	余りによって注文実行の種類を文字列として取得します。
TypeTime	期限時の注文の種類を取得します。
TypeTimeDescription	期限時の注文の種類を文字列として取得します。
Magic	注文を出したエキスパートアドバイザーの識別子を取得します。

© 2000-2025, MetaQuotes Ltd.

整数型プロパティへのアクセス	
PositionId	ポジションの識別子を取得します。
double 型プロパティへのアクセス	
VolumeInitial	注文の初期ボリュームを取得します。
VolumeCurrent	満たされていない注文ボリュームを取得します。
PriceOpen	注文の価格を取得します。
StopLoss	注文の決済逆指値を取得します。
TakeProfit	注文の決済指値を取得します。
PriceCurrent	注文シンボルの現価格を取得します。
PriceStopLimit	逆指値注文の設定価格を取得します。
テキストプロパティへのアクセス	
Symbol	注文の銘柄名を取得します。
Comment	注文コメントを取得します。
MQL5 API 関数へのアクセス	
InfoInteger	指定された整数型プロパティの値を取得します。
InfoDouble	指定された double 型プロパティの値を取得します。
InfoString	指定された string 型プロパティの値を取得します。
State	
StoreState	注文パラメータを保存します。
CheckState	現在のパラメータを保存されたパラメータに対してチェックします。
Selection	
Select	注文をチケットによって選択しプロパティにアクセスします。
SelectByIndex	注文をインデックスによって選択しプロパティにアクセスします。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000–2025, MetaQuotes Ltd.

Ticket

事前にSelectメソッドでアクセスのために選択された注文のチケットを取得します。

```
ulong  Ticket() const
```

戻り値

成功の場合は注文チケット、それ以外の場合はULONG_MAX

注意事項

注文はSelect（チケットによる選択）またはSelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TimeSetup

発注時刻を取得します。

```
datetime  TimeSetup() const
```

戻り値

発注時刻

注意事項

注文は Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TimeSetupMsc

01.01.1970から経過したミリ秒数で表された発注時刻を受け取ります。

```
ulong  TimeSetupMsc() const
```

戻り値

01.01.1970から経過したミリ秒数で表された注文時刻

注意事項

注文は主に Select（チケットによる選択）または SelectByIndex （インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

OrderType

注文の種類を取得します。

```
ENUM_ORDER_TYPE  OrderType()
```

戻り値

注文の種類（ENUM_ORDER_TYPE 列挙値）

注意事項

注文は Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TypeDescription

注文の種類を文字列として取得します。

```
string  TypeDescription() const
```

戻り値

文字列としての注文の種類

注意事項

注文は Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

State

注文状態を取得します。

```
ENUM_ORDER_STATE  State() const
```

戻り値

注文状態（ENUM_ORDER_STATE 列挙値）

注意事項

注文は Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

StateDescription

注文状態を文字列として取得します。

```
string  StateDescription() const
```

戻り値

文字列としての注文状態

注意事項

注文は Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TimeExpiration

注文期限を取得します。

```
datetime  TimeExpiration() const
```

戻り値

注文時に設定された注文期限

注意事項

注文は Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TimeDone

注文実行または取り消しの時刻を取得します。

```
datetime  TimeDone() const
```

戻り値

注文実行または取り消しの時刻

注意事項

注文は Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TimeDoneMsc

01.01.1970 から経ったミリ秒で表された注文の実行/取り消し時刻を受け取ります。

```
ulong  TimeDoneMsc() const
```

戻り値

01.01.1970 から経ったミリ秒で表された注文の実行/取り消し時刻

注意事項

注文は主に Select（チケットによる選択）または SelectByIndex （インデックスによる選択）メソッドで選択されるべきです。

© 2000-2025, MetaQuotes Ltd.

TypeFilling

注文充填の種類を取得します。

```
ENUM_ORDER_TYPE_FILLING  TypeFilling() const
```

戻り値

注文充填の種類（ENUM_ORDER_TYPE_FILLING 列挙値）

注意事項

注文は Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TypeFillingDescription

注文充填の種類を文字列としてを取得します。

```
string  TypeFillingDescription() const
```

戻り値

文字列としての注文充填の種類

注意事項

注文は Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TypeTime

期限時の注文の種類を取得します。

```
ENUM_ORDER_TYPE_TIME  TypeTime() const
```

戻り値

期限時の注文の種類

注意事項

注文は Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TypeTimeDescription

期限時の注文の種類を文字列として取得します。

```
string  TypeTimeDescription() const
```

戻り値

文字列としての期限時の注文の種類

注意事項

注文は Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Magic

注文を出したエキスパートアドバイザーの識別子を取得します。

```
long  Magic() const
```

戻り値

注文を出したエキスパートアドバイザーの識別子

注意事項

注文は Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

PositionId

ポジションの識別子を取得します。

```
long  PositionId() const
```

戻り値

注文が発生したポジションの識別子

注意事項

注文は Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

VolumeInitial

注文の初期ボリュームを取得します。

```
double  VolumeInitial() const
```

戻り値

注文の初期ボリューム

注意事項

注文は Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

VolumeCurrent

満たされていない注文ボリュームを取得します。

```
double  VolumeCurrent() const
```

戻り値

満たされていない注文ボリューム

注意事項

注文は Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

PriceOpen

注文の価格を取得します。

```
double  PriceOpen() const
```

戻り値

注文の価格

注意事項

注文は Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

StopLoss

注文の決済逆指値を取得します。

```
double  StopLoss() const
```

戻り値

注文の決済逆指値

注意事項

注文は Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TakeProfit

注文の決済指値を取得します。

```
double  TakeProfit() const
```

戻り値

注文の決済指値

注意事項

注文は Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

PriceCurrent

注文シンボルの現価格を取得します。

```
double  PriceCurrent() const
```

戻り値

注文シンボルの現価格

注意事項

注文は Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

PriceStopLimit

逆指値注文の設定価格を取得します。

```
double  PriceStopLimit() const
```

戻り値

逆指値注文の設定価格

注意事項

注文は Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Symbol

注文の銘柄名を取得します。

```
string  Symbol() const
```

戻り値

注文の銘柄名

注意事項

注文は Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Comment

注文コメントを取得します。

```
string  Comment() const
```

戻り値

注文コメント

注意事項

注文は Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

InfoInteger

指定された整数型プロパティの値を取得します。

```
bool  InfoInteger(
   ENUM_ORDER_PROPERTY_INTEGER  prop_id,      // プロパティ識別子
   long&                        var           // 変数への参照
   ) const
```

パラメータ

prop_id

[in] 整数型プロパティの識別子（ENUM_ORDER_PROPERTY_INTEGER 列挙値）

var

[out] 結果を受け取る long 型変数への参照

戻り値

成功の場合は true、プロパティ値が取得できなかった場合は false

注意事項

注文は Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

InfoDouble

指定された double 型プロパティの値を取得します。

```
bool  InfoDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  prop_id,      // プロパティ識別子
   double&                     var          // 変数への参照
   ) const
```

パラメータ

prop_id

[in] double 型プロパティの識別子（ENUM_ORDER_PROPERTY_DOUBLE 列挙値）

var

[out] 結果を受け取る double 型変数への参照

戻り値

成功の場合は true、プロパティ値が取得できなかった場合は false

注意事項

注文は Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

InfoString

指定された string 型プロパティの値を取得します。

```
bool  InfoString(
   ENUM_ORDER_PROPERTY_STRING  prop_id,      // プロパティ識別子
   string&                     var           // 変数への参照
   ) const
```

パラメータ

prop_id

[in] テキストプロパティの識別子

var

[out] 結果を受け取る string 型変数への参照

戻り値

成功の場合は true、プロパティ値が取得できなかった場合は false

注意事項

注文は Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

StoreState

注文パラメータを保存します。

```
void  StoreState()
```

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

CheckState

現在のパラメータを保存されたパラメータに対してチェックします。

```
bool  CheckState()
```

戻り値

注文パラメータが最後の StoreState() メソッドの呼び出し以来変更された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Select

注文をチケットによって選択しプロパティにアクセスします。

```
bool  Select(
   ulong  ticket   // 注文チケット
  )
```

戻り値

成功の場合は true、注文が選択できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

SelectByIndex

注文をインデックスで選択しプロパティにアクセスします。

```
bool  SelectByIndex(
   int  index   // 注文インデックス
  )
```

戻り値

成功の場合は true、注文が選択できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

CHistoryOrderInfo

CHistoryOrderInfo は履歴注文のプロパティに簡単にアクセスするためのクラスです。

説明

CHistoryOrderInfo クラスは履歴注文のプロパティへの簡易化されたアクセスを提供します。

宣言

```
class CHistoryOrderInfo : public CObject
```

タイトル

```
#include <Trade\HistoryOrderInfo.mqh>
```

継承階層

CObject

CHistoryOrderInfo

グループ分けされたクラスメソッド

整数型プロパティへのアクセス	
TimeSetup	発注時刻を取得します。
TimeSetupMsc	01.01.1970から経過したミリ秒数で表された発注時刻を受け取ります。
OrderType	注文の種類を取得します。
OrderTypeDescription	注文の種類を文字列として取得します。
State	注文状態を取得します。
StateDescription	注文状態を文字列として取得します。
TimeExpiration	注文期限を取得します。
TimeDone	注文実行または取り消しの時刻を取得します。
TimeDoneMsc	01.01.1970　から経ったミリ秒で表された注文の実行/取り消し時刻を受け取ります。
TypeFilling	余りによって注文実行の種類を取得します。
TypeFillingDescription	余りによって注文実行の種類を文字列として取得します。
TypeTime	期限時の注文の種類を取得します。
TypeTimeDescription	期限時の注文の種類を文字列として取得します。
Magic	注文を出したエキスパートアドバイザーの識別子を取得します。
PositionId	ポジションの識別子を取得します。

© 2000-2025, MetaQuotes Ltd.

整数型プロパティへのアクセス	
double 型プロパティへのアクセス	
VolumeInitial	注文の初期ボリュームを取得します。
VolumeCurrent	満たされていない注文ボリュームを取得します。
PriceOpen	注文の価格を取得します。
StopLoss	注文の決済逆指値を取得します。
TakeProfit	注文の決済指値を取得します。
PriceCurrent	注文シンボルの現価格を取得します。
PriceStopLimit	逆指値注文の設定価格を取得します。
テキストプロパティへのアクセス	
Symbol	注文シンボルを取得します。
Comment	注文コメントを取得します。
MQL5 API 関数へのアクセス	
InfoInteger	指定された整数型プロパティの値を取得します。
InfoDouble	指定された double 型プロパティの値を取得します。
InfoString	指定された string 型プロパティの値を取得します。
Selection	
Ticket	チケットを取得して注文を選択します。
SelectByIndex	インデックスによって注文を選択します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000–2025, MetaQuotes Ltd.

TimeSetup

発注時刻を取得します。

```
datetime  TimeSetup() const
```

戻り値

発注時刻

注意事項

履歴注文は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TimeSetupMsc

01.01.1970から経過したミリ秒数で表された発注時刻を受け取ります。

```
ulong  TimeSetupMsc() const
```

戻り値

01.01.1970から経過したミリ秒数で表された発注時刻

注意事項

履歴注文は主に Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000-2025, MetaQuotes Ltd.

OrderType

注文の種類を取得します。

```
ENUM_ORDER_TYPE  OrderType() const
```

戻り値

注文の種類（ENUM_ORDER_TYPE 列挙値）

注意事項

履歴注文は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TypeDescription

注文の種類を文字列として取得します。

```
string  TypeDescription() const
```

戻り値

文字列としての注文の種類

注意事項

履歴注文は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

State

注文状態を取得します。

```
ENUM_ORDER_STATE  State() const
```

戻り値

注文状態（ENUM_ORDER_STATE 列挙値）

注意事項

履歴注文は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

StateDescription

注文状態を文字列として取得します。

```
string  StateDescription() const
```

戻り値

文字列としての注文状態

注意事項

履歴注文は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TimeExpiration

注文期限を取得します。

```
datetime  TimeExpiration() const
```

戻り値

注文時に設定された注文期限

注意事項

履歴注文は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TimeDone

注文実行または取り消しの時刻を取得します。

```
datetime  TimeDone() const
```

戻り値

注文実行または取り消しの時刻

注意事項

履歴注文は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TimeDoneMsc

01.01.1970 から経ったミリ秒で表された注文の実行/取り消し時刻を受け取ります。

```
ulong  TimeDoneMsc() const
```

戻り値

01.01.1970 から経ったミリ秒で表された注文の実行/取り消し時刻

注意事項

履歴注文は主に Ticket(チケットによる選択)または SelectByIndex(インデックスによる選択)メソッドで選択されるべきです。

© 2000-2025, MetaQuotes Ltd.

TypeFilling

余りによって注文実行の種類を取得します。

```
ENUM_ORDER_TYPE_FILLING  TypeFilling() const
```

戻り値

余りによる注文実行の種類（ENUM_ORDER_TYPE_FILLING 列挙値）

注意事項

履歴注文は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TypeFillingDescription

余りによって注文実行の種類を文字列として取得します。

```
string  TypeFillingDescription() const
```

戻り値

文字列としての余りによる注文実行の種類

注意事項

履歴注文は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TypeTime

期限時の注文の種類を取得します。

```
ENUM_ORDER_TYPE_TIME  TypeTime() const
```

戻り値

期限時に注文の種類（ENUM_ORDER_TYPE_TIME 列挙値）

注意事項

履歴注文は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TypeTimeDescription

期限時の注文の種類を文字列として取得します。

```
string  TypeTimeDescription() const
```

戻り値

文字列としての期限時の注文の種類

注意事項

履歴注文は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Magic

注文を出したエキスパートアドバイザーの識別子を取得します。

```
long  Magic() const
```

戻り値

注文を出したエキスパートアドバイザーの識別子

注意事項

履歴注文は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

PositionId

ポジションの識別子を取得します。

```
long  PositionId() const
```

戻り値

注文が発生したポジションの識別子

注意事項

履歴注文は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

VolumeInitial

注文の初期ボリュームを取得します。

```
double  VolumeInitial() const
```

戻り値

注文の初期ボリューム

注意事項

履歴注文は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

VolumeCurrent

満たされていない注文ボリュームを取得します。

```
double  VolumeCurrent() const
```

戻り値

満たされていない注文ボリューム

注意事項

履歴注文は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

PriceOpen

注文の価格を取得します。

```
double  PriceOpen() const
```

戻り値

注文の価格

注意事項

履歴注文は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

StopLoss

注文の決済逆指値を取得します。

```
double  StopLoss() const
```

戻り値

注文の決済逆指値

注意事項

履歴注文は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TakeProfit

注文の決済指値を取得します。

```
double  TakeProfit() const
```

戻り値

注文の決済指値

注意事項

履歴注文は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

PriceCurrent

注文シンボルの現在の価格を取得します。

```
double  PriceCurrent() const
```

戻り値

注文シンボルの現在の価格

注意事項

履歴注文は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

PriceStopLimit

注文の逆指値を取得します。

```
double  PriceStopLimit() const
```

戻り値

注文の逆指値

注意事項

履歴注文は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Symbol

注文の銘柄名を取得します。

```
string  Symbol() const
```

戻り値

注文の銘柄名

注意事項

履歴注文は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Comment

注文コメントを取得します。

```
string  Comment() const
```

戻り値

注文コメント

注意事項

履歴注文は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

InfoInteger

指定された整数型プロパティの値を取得します。

```
bool  InfoInteger(
   ENUM_ORDER_PROPERTY_INTEGER  prop_id,      // プロパティ識別子
   long&                        var           // 変数への参照
   ) const
```

パラメータ

prop_id

[in] 整数型プロパティの識別子（ENUM_ORDER_PROPERTY_INTEGER 列挙値）

var

[out] 結果を受け取る long 型変数への参照

戻り値

成功の場合は true、プロパティ値が取得できなかった場合は false

注意事項

履歴注文は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

InfoDouble

指定された double 型プロパティの値を取得します。

```
bool  InfoDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  prop_id,      // プロパティ識別子
   double&                     var          // 変数への参照
   ) const
```

パラメータ

prop_id

[in] double 型プロパティの識別子（ENUM_ORDER_PROPERTY_DOUBLE 列挙値）

var

[out] 結果を受け取る double 型変数への参照

戻り値

成功の場合は true、プロパティ値が取得できなかった場合は false

注意事項

履歴注文は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

InfoString

指定された string 型プロパティの値を取得します。

```
bool  InfoString(
   ENUM_ORDER_PROPERTY_STRING  prop_id,      // プロパティ識別子
   string&                     var           // 変数への参照
   ) const
```

パラメータ

prop_id

[in] テキストプロパティの識別子（ENUM_ORDER_PROPERTY_STRING列挙値）

var

[out] 結果を受け取る string 型変数への参照

戻り値

成功の場合は true、プロパティ値が取得できなかった場合は false

注意事項

履歴注文は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Ticket（Get メソッド）

注文チケットを取得します。

```
ulong  Ticket() const
```

戻り値

注文チケット。

Ticket（Set メソッド）

操作を続けるために注文を選択します。

```
void  Ticket(
   ulong  ticket      // 注文チケット
  )
```

パラメータ

ticket

[in] 注文チケット

© 2000–2025, MetaQuotes Ltd.

SelectByIndex

注文をインデックスで選択しプロパティにアクセスします。

```
bool  SelectByIndex(
   int  index   // 注文インデックス
  )
```

戻り値

成功の場合は true、注文が選択できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

CPositionInfo

CPositionInfo はオープンポジションのプロパティに簡単にアクセスするためのクラスです。

説明

CPositionInfo クラスはオープンポジションプロパティへの容易化したアクセスを提供します。

宣言

```
class CPositionInfo : public CObject
```

タイトル

```
#include <Trade\PositionInfo.mqh>
```

継承階層

CObject

CPositionInfo

グループ分けされたクラスメソッド

整数型プロパティへのアクセス	
Time	注文が出された時刻を取得します。
TimeMsc	01.01.1970 からから経過したミリ秒数で表された注文が出された時刻を受け取ります。
TimeUpdate	01.01.1970 から経過した秒数で表されたポジション変更の時刻を受け取ります。
TimeUpdateMsc	01.01.1970 からから経過したミリ秒数で表されたポジション変更の時刻を受け取ります。
PositionType	ポジションの種類を取得します。
TypeDescription	ポジションの種類を文字列として取得します。
Magic	注文を出したエキスパートアドバイザーの識別子を取得します。
Identifier	ポジションの識別子を取得します。
double 型プロパティへのアクセス	
Volume	ポジションのボリュームを取得します。
PriceOpen	ポジション価格を取得します。
StopLoss	ポジションの決済逆指値を取得します。
TakeProfit	ポジションの決済指値を取得します。

© 2000-2025, MetaQuotes Ltd.

整数型プロパティへのアクセス	
PriceCurrent	ポジションシンボルの現価格を取得します。
Commission	ポジションの手数料を取得します。
Swap	ポジションのスワップを取得します。
Profit	ポジションの現在利益を取得します。
テキストプロパティへのアクセス	
Symbol	ポジション銘柄名を取得します。
Comment	ポジションコメントを取得します。
MQL5 API 関数へのアクセス	
InfoInteger	指定された整数型プロパティの値を取得します。
InfoDouble	指定された double 型プロパティの値を取得します。
InfoString	指定された string 型プロパティの値を取得します。
Selection	
Select	ポジションを選択します。
SelectByIndex	ポジションをインデックスによって選択します。
SelectByMagic	商品名と今後の作業のマジックナンバーでポジションを選択します。
SelectByTicket	チケットでポジションを選択します。
State	
StoreState	ポジションパラメータを保存します。
CheckState	現在のパラメータを保存されたパラメータに対してチェックします。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000–2025, MetaQuotes Ltd.

Time

注文が出された時刻を取得します。

```
datetime  Time() const
```

戻り値

注文が出された時刻

注意事項

ポジションは Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TimeMsc

01.01.1970 から経ったミリ秒で表されたポジション開始時刻を取得します。

```
ulong  TimeMsc() const
```

戻り値

01.01.1970 から経ったミリ秒で表されたポジション開始時刻

注意事項

ポジションは主に Select（シンボルによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000-2025, MetaQuotes Ltd.

TimeUpdate

01.01.1970 から経過した秒数で表されたポジション変更の時刻を受け取ります。

```
datetime  TimeUpdate() const
```

戻り値

01.01.1970 から経過した秒数で表されたポジション変更の時刻

注意事項

ポジションは主に Select（シンボルによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000-2025, MetaQuotes Ltd.

TimeUpdateMsc

01.01.1970 からから経過したミリ秒数で表されたポジション変更の時刻を受け取ります。

```
ulong  TimeUpdateMsc() const
```

戻り値

01.01.1970 からから経過したミリ秒数で表されたポジション変更の時刻

注意事項

ポジションは主に Select（シンボルによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000-2025, MetaQuotes Ltd.

PositionType

ポジションの種類を取得します。

```
ENUM_POSITION_TYPE  PositionType() const
```

戻り値

ポジションの種類（ENUM_POSITION_TYPE 列挙値）

注意事項

ポジションは Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TypeDescription

ポジションの種類を文字列として取得します。

```
string  TypeDescription() const
```

戻り値

文字列としてのポジションの種類

注意事項

ポジションは Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Magic

注文を出したエキスパートアドバイザーの識別子を取得します。

```
long  Magic() const
```

戻り値

注文を出したエキスパートアドバイザーの識別子

注意事項

ポジションは Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Identifier

ポジションの識別子を取得します。

```
long  Identifier() const
```

戻り値

ポジションの識別子

注意事項

ポジションは Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Volume

ポジションのボリュームを取得します。

```
double  Volume() const
```

戻り値

ポジションのボリューム

注意事項

ポジションは Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000-2025, MetaQuotes Ltd.

PriceOpen

ポジション価格を取得します。

```
double  PriceOpen() const
```

戻り値

ポジション価格

注意事項

ポジションは Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

StopLoss

ポジションの決済逆指値を取得します。

```
double  StopLoss() const
```

戻り値

ポジションの決済逆指値

注意事項

ポジションは Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TakeProfit

ポジションの決済指値を取得します。

```
double  TakeProfit() const
```

戻り値

ポジションの決済指値

注意事項

ポジションは Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

PriceCurrent

ポジションシンボルの現価格を取得します。

```
double  PriceCurrent() const
```

戻り値

ポジションシンボルの現価格

© 2000–2025, MetaQuotes Ltd.

Commission

ポジションの手数料を取得します。

```
double  Commission() const
```

戻り値

口座の預金通貨でのポジションの手数料

注意事項

ポジションは Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Swap

ポジションのスワップを取得します。

```
double  Swap() const
```

戻り値

口座の預金通貨でのポジションのスワップ

注意事項

ポジションは Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Profit

ポジションの現在利益を取得します。

```
double  Profit() const
```

戻り値

口座の預金通貨でのポジションの現在利益

注意事項

ポジションは Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Symbol

ポジション銘柄名を取得します。

```
string  Symbol() const
```

戻り値

ポジション銘柄名

注意事項

ポジションは Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Comment

ポジションコメントを取得します。

```
string  Comment() const
```

戻り値

ポジションコメント

注意事項

ポジションは Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

InfoInteger

指定された整数型プロパティの値を取得します。

```
bool  InfoInteger(
   ENUM_POSITION_PROPERTY_INTEGER  prop_id,      // プロパティ識別子
   long&                           var           // 変数への参照
   ) const
```

パラメータ

prop_id

[in] 整数型プロパティの識別子（ENUM_POSITION_PROPERTY_INTEGER 列挙値）

var

[out] 結果を受け取る long 型変数への参照

戻り値

成功の場合は true、プロパティ値が取得できなかった場合は false

注意事項

ポジションは Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

InfoDouble

指定された double 型プロパティの値を取得します。

```
bool  InfoDouble(
   ENUM_POSITION_PROPERTY_DOUBLE  prop_id,      // プロパティ識別子
   double&                        var          // 変数への参照
   ) const
```

パラメータ

prop_id

[in] double 型プロパティの識別子（ENUM_POSITION_PROPERTY_DOUBLE 列挙値）

var

[in] 結果を受け取る double 型変数への参照

戻り値

成功の場合は true、プロパティ値が取得できなかった場合は false

注意事項

ポジションは Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

InfoString

指定された string 型プロパティの値を取得します。

```
bool  InfoString(
   ENUM_POSITION_PROPERTY_STRING  prop_id,      // プロパティ識別子
   string&                        var           // 変数への参照
   ) const
```

パラメータ

prop_id

[in] テキストプロパティの識別子（ENUM_POSITION_PROPERTY_STRING 列挙値）

var

[out] 結果を受け取る string 型変数への参照

戻り値

成功の場合は true、プロパティ値が取得できなかった場合は false

注意事項

ポジションは Select（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Select

操作するポジションを選択します。

```
bool  Select(
   const string  symbol      // シンボル
  )
```

パラメータ

symbol

[in] ポジション選択に使用されるシンボル

© 2000–2025, MetaQuotes Ltd.

SelectByIndex

ポジションをインデックスで選択しプロパティにアクセスします。

```
bool  SelectByIndex(
   int  index   // ポジションインデックス
  );
```

戻り値

成功の場合は true、ポジションが選択できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

SelectByMagic

商品名と今後の作業のマジックナンバーでポジションを選択します。

```
bool  SelectByMagic(
   const string  symbol, // 商品名
   const ulong   magic   // マジックナンバー
   );
```

パラメータ

symbol

[in] 商品名。

magic

[in] ポジションのマジックナンバー。

戻り値

成功した場合trueを返し、ポジションが選択されていない場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

SelectByTicket

今後の作業の為のチケットによってポジションを選択します。

```
bool  SelectByTicket(
   ulong  ticket      // ポジションチケット
   )
```

パラメータ

ticket

[in] ポジションチケット。

戻り値

成功した場合trueを返し、ポジションが選択されていない場合はfalseを返します。

© 2000–2025, MetaQuotes Ltd.

StoreState

ポジションパラメータを保 存します。

```
void  StoreState()
```

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

CheckState

現在のパラメータを保存されたパラメータに対してチェックします。

```
bool  CheckState()
```

戻り値

ポジションパラメータが最後の StoreState() メソッドの呼び出し以来変更された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

CDealInfo

CDealInfo は約定のプロパティに簡単にアクセスするためのクラスです。

説明

CDealInfo クラスは約定のプロパティへの簡易化されたアクセスを提供します。

宣言

```
class CDealInfo : public CObject
```

タイトル

```
#include <Trade\DealInfo.mqh>
```

継承階層

CObject
  CDealInfo

グループ分けされたクラスメソッド

整数型プロパティへのアクセス	
Order	約定に使用された注文を取得します。
Time	約定の時刻を取得します。
TimeMsc	01.01.1970 から経過したミリ秒数で表された約定の時刻を受け取ります。
DealType	約定の種類を取得します。
TypeDescription	約定の種類を文字列として取得します。
Entry	約定の方向を取得します。
EntryDescription	約定の方向を文字列として取得します。
Magic	約定に使用されたエキスパートアドバイザーの識別子を取得します。
PositionId	約定が発生したポジションの識別子を取得します。
double 型プロパティへのアクセス	
Volume	約定ボリュームを取得します。
Price	約定価格を取得します。
Commision	約定の手数料を取得します。
Swap	ポジション決済時のスワップを取得します。
Profit	約定の金銭的な結果を取得します。

© 2000–2025, MetaQuotes Ltd.

整数型プロパティへのアクセス	
テキストプロパティへのアクセス	
Symbol	約定の銘柄名を取得します。
Comment	約定のコメントを取得します。
MQL5 API 関数へのアクセス	
InfoInteger	指定された整数型プロパティの値を取得します。
InfoDouble	指定された double 型プロパティの値を取得します。
InfoString	指定された string 型プロパティの値を取得します。
Selection	
Ticket	チケットを取得して約定を選択します。
SelectByIndex	約定をインデックスで選択します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000–2025, MetaQuotes Ltd.

Order

約定に使用された注文を取得します。

```
long  Order() const
```

戻り値

約定に使用された注文

注意事項

約定は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。</t4>

© 2000–2025, MetaQuotes Ltd.

Time

約定の時刻を取得します。

```
datetime  Time() const
```

戻り値

約定の時刻

注意事項

約定は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TimeMsc

01.01.1970 から経過したミリ秒数で表された約定時刻を受け取ります。

```
ulong  TimeMsc() const
```

戻り値

01.01.1970 から経過したミリ秒数で表された注文の約定時刻

注意事項

約定は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000-2025, MetaQuotes Ltd.

DealType

約定の種類を取得します。

```
ENUM_DEAL_TYPE  DealType() const
```

戻り値

注文の種類（ENUM_DEAL_TYPE 列挙値）

注意事項

約定は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

TypeDescription

約定の種類を文字列として取得します。

```
string  TypeDescription() const
```

戻り値

文字列としての約定の種類

注意事項

約定は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Entry

約定の方向を取得します。

```
ENUM_DEAL_ENTRY  Entry() const
```

戻り値

注文の方向（ENUM_DEAL_ENTRY 列挙値）

注意事項

約定は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

EntryDescription

約定の方向を文字列として取得します。

```
string  EntryDescription() const
```

戻り値

文字列としての約定の方向

注意事項

約定は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Magic

約定に使用されたエキスパートアドバイザーの識別子を取得します。

```
long  Magic() const
```

戻り値

約定に使用されたエキスパートアドバイザーの識別子

注意事項

約定は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

PositionId

約定が発生したポジションの識別子を取得します。

```
long  PositionId() const
```

戻り値

約定が発生したポジションの識別子

注意事項

約定は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Volume

約定のボリュームを取得します。

```
double  Volume() const
```

戻り値

約定ボリューム

注意事項

約定は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Price

約定価格を取得します。

```
double  Price() const
```

戻り値

約定価格

注意事項

約定は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Commission

約定の手数料を取得します。

```
double  Commission() const
```

戻り値

約定の手数料

注意事項

約定は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Swap

ポジション決済時のスワップを取得します。

```
double  Swap() const
```

戻り値

ポジション決済時のスワップ

注意事項

約定は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Profit

約定の金銭的な結果を取得します。

```
double  Profit() const
```

戻り値

口座の預金通貨での約定の金銭的な結果

注意事項

約定は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Symbol

約定の銘柄名を取得します。

```
string  Symbol() const
```

戻り値

約定の銘柄名

注意事項

約定は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Comment

約定コメントを取得します。

```
string  Comment() const
```

戻り値

約定コメント

注意事項

約定は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

InfoInteger

指定された整数型プロパティの値を取得します。

```
bool  InfoInteger(
   ENUM_DEAL_PROPERTY_INTEGER  prop_id,      // プロパティ識別子
   long&                       var           // 変数への参照
   ) const
```

パラメータ

prop_id

[in] 整数型プロパティの識別子（ENUM_DEAL_PROPERTY_INTEGER 列挙値）

var

[out] 結果を受け取る long 型変数への参照

戻り値

成功の場合は true、プロパティ値が取得できなかった場合は false

注意事項

約定は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

InfoDouble

指定された double 型プロパティの値を取得します。

```
bool  InfoDouble(
   ENUM_DEAL_PROPERTY_DOUBLE  prop_id,      // プロパティ識別子
   double&                    var           // 変数への参照
   ) const
```

パラメータ

prop_id

[in] double 型プロパティの識別子（ENUM_DEAL_PROPERTY_DOUBLE 列挙値）

var

[out] 結果を受け取る double 型変数への参照

戻り値

成功の場合は true、プロパティ値が取得できなかった場合は false

注意事項

約定は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

InfoString

指定された string 型プロパティの値を取得します。

```
bool  InfoString(
   ENUM_DEAL_PROPERTY_STRING  prop_id,      // プロパティ識別子
   string&                    var           // 変数への参照
   ) const
```

パラメータ

prop_id

[in] テキストプロパティの識別子（ENUM_DEAL_PROPERTY_STRING 列挙値）

var

[out] 結果を受け取る string 型変数への参照

戻り値

成功の場合は true、プロパティ値が取得できなかった場合は false

注意事項

約定は Ticket（チケットによる選択）または SelectByIndex（インデックスによる選択）メソッドで選択されるべきです。

© 2000–2025, MetaQuotes Ltd.

Ticket（Get メソッド）

約定チケットを取得します。

```
ulong  Ticket() const
```

戻り値

約定チケット。

Ticket（Set メソッド）

操作するポジションを選択します。

```
void  Ticket(
   ulong  ticket      // チケット
  )
```

パラメータ

ticket

[in] 約定チケット

© 2000-2025, MetaQuotes Ltd.

SelectByIndex

約定をインデックスで選択しプロパティにアクセスします。

```
bool  SelectByIndex(
   int  index   // 注文インデックス
  )
```

戻り値

成功の場合は true、注文が選択できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

CTrade

CTrade は取引関数に簡単にアクセスするためのクラスです。

説明

CTrade クラスは取引関数への簡易化されたアクセスを提供します。

宣言

```
class CTrade : public CObject
```

タイトル

```
#include <Trade\Trade.mqh>
```

継承階層

COBject

CTrade

直接子孫

CExpertTrade

グループ分けされたクラスメソッド

パラメータの設定	
LogLevel	ログレベルを設定します。
SetExpertMagicNumber	エキスパートアドバイザーIDを設定します。
SetDeviationInPoints	許可された偏差を設定します。
SetTypeFilling	注文の充填タイプを設定します。
SetTypeFillingBySymbol	指定された銘柄の設定によって注文の履行タイプを設定します。
SetAsyncMode	取引操作のための非同期モードを設定します。
SetMarginMode	現在の口座の設定に応じて、証拠金計算モードを設定します。
注文の操作	
OrderOpen	パラメータを使用して未決注文を出します。
OrderModify	未決注文のパラメータを変更します。
OrderDelete	未決注文を削除します。
ポジションの操作	
PositionOpen	指定されたパラメータで注文を出します。
PositionModify	ポジションパラメータを変更します。

© 2000–2025, MetaQuotes Ltd.

パラメータの設定	
PositionClose	ポジションを決済します。
PositionClosePartial	指定された銘柄のポジションまたは指定されたチケットを持つポジションを部分的に決済します。
PositionCloseBy	反対ポジションで指定されたチケットのポジションを決済します。
追加メソッド	
Buy	指定されたパラメータで買い注文を出します。
Sell	指定されたパラメータで売り注文を出します。
BuyLimit	指定されたパラメータで買い指値注文を出します。
BuyStop	指定されたパラメータで買い逆指注文を出します。
SellLimit	指定されたパラメータで売り指値注文を出します。
SellStop	指定されたパラメータで売り逆指注文を出します。
直近のリクエストのパラメータへのアクセス	
Request	直近のリクエストの構造体のコピーを取得します。
RequestAction	取引操作の種類を取得します。
RequestActionDescription	取引操作の種類を文字列として取得します。
RequestMagic	エキスパートアドバイザーのマジックナンバーを取得します。
RequestOrder	直近のリクエストで使用された注文チケットを取得します。
RequestSymbol	直近のリクエストで使用された銘柄名を取得します。
RequestVolume	直近のリクエストで使用された取引高を(ロット単位で)取得します。
RequestPrice	直近のリクエストで使用された価格を取得します。
RequestStopLimit	直近のリクエストで使用された未決済の指値つきストップ注文の価格を取得します。
RequestSL	直近のリクエストで使用された決済逆指注文の価格を取得します。
RequestTP	直近のリクエストで使用された決済指注文の価格を取得します。
RequestDeviation	直近のリクエストで使用された注文の価格偏差値を取得します。
RequestType	直近のリクエストで使用された注文の種類を取得します。
RequestTypeDescription	直近のリクエストで使用された注文の種類を文字列として取得します。
RequestTypeFilling	直近のリクエストで使用された注文充填の種類を取得します。
RequestTypeFillingDescription	直近のリクエストで使用された注文充填の種類を文字列として取得します。

© 2000–2025, MetaQuotes Ltd.

パラメータの設定	
RequestTypeTime	直近のリクエストで使用された注文の期限を取得します。
RequestTypeTimeDescription	直近のリクエストで使用された注文の期限を文字列として取得します。
RequestExpiration	直近のリクエストで使用された注文の期限を取得します。
RequestComment	直近のリクエストで使用された注文の期限を文字列として取得します。
RequestPosition	ポジションチケットを取得します。
RequestPositionBy	反対ポジションのチケットを取得します。
直近のリクエストのチェック結果へのアクセス	
CheckResult	直近のリクエストチェック結果の構造体を取得します。
CheckResultRetcode	リクエストの正当さをチェックしながら満たされたMqlTradeCheckResult型のretcodeフィールドの値を取得します。
CheckResultRetcodeDescription	リクエストの正当さをチェックしながら満たされたMqlTradeCheckResult型のretcodeフィールドの文字列としての説明を取得します。
CheckResultBalance	リクエストの正当さをチェックしながら満たされたMqlTradeCheckResult型のbalanceフィールドの値を取得します。
CheckResultEquity	リクエストの正当さをチェックしながら満たされたMqlTradeCheckResult型のequityフィールドの値を取得します。
CheckResultProfit	リクエストの正当さをチェックしながら満たされたMqlTradeCheckResult型のprofitフィールドの値を取得します。
CheckResultMargin	リクエストの正当さをチェックしながら満たされたMqlTradeCheckResult型のmarginフィールドの値を取得します。
CheckResultMarginFree	リクエストの正当さをチェックしながら満たされたMqlTradeCheckResult型のmargin_freeフィールドの値を取得します。
CheckResultMarginLevel	リクエストの正当さをチェックしながら満たされたMqlTradeCheckResult型のmargin_levelフィールドの値を取得します。
CheckResultComment	リクエストの正当さをチェックしながら満たされたMqlTradeCheckResult型のcommentフィールドの値を取得します。
直近のリクエストの実行結果へのアクセス	
Result	直近のリクエスト結果の構造体を取得します。
ResultRetcode	リクエスト結果のコードを取得します。

© 2000–2025, MetaQuotes Ltd.

パラメータの設定	
ResultRetcodeDescription	リクエスト結果のコードをテキストとして取得します。
ResultDeal	約定チケットを取得します。
ResultOrder	注文チケットを取得します。
ResultVolume	約定または注文のボリュームを取得します。
ResultPrice	ブローカーによって確認された価格を取得します。
ResultBid	現在の売値を取得します。
ResultAsk	現在の買値を取得します。
ResultComment	ブローカーコメントを取得します。
補助メソッド	
PrintRequest	直近のリクエストのパラメータを操作ログにプリントします。
PrintResult	直近のリクエストの結果を操作ログにプリントします。
FormatRequest	直近のリクエストのパラメータのフォーマットされた文字列を準備します。
FormatRequestResult	直近のリクエストの結果のフォーマットされた文字列を準備します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

LogLevel

メッセージのログレベルを設定します。

```
void  LogLevel(
   ENUM_LOG_LEVELS  log_level      // ログレベル
  )
```

パラメータ

log_level

[in] ログレベル

戻り値

なし

注意事項

LOG_LEVEL_NO 及びそれ以下の値はメッセージを非表示にします。(このモードは最適化で自動設定されます。) LOG_LEVEL_ERRORS はエラーメッセージのみを表示します。(これが初期値です。) LOG_LEVEL_ALL 及びそれ以上の値は全てのメッセージを表示します。(個のモードはテストで自動設定されます。)

ENUM_LOG_LEVELS

識別子	説明	値
LOG_LEVEL_NO	メッセージが非表示です。	0
LOG_LEVEL_ERRORS	エラーメッセージのみが表示されます。	1
LOG_LEVEL_ALL	全てのメッセージが表示されます。	2

© 2000-2025, MetaQuotes Ltd.

SetExpertMagicNumber

エキスパートアドバイザーID(識別子)を設定します。

```
void  SetExpertMagicNumber(
   ulong  magic      // 識別子
  )
```

パラメータ

magic

[in] 新しいエキスパートアドバイザーID

戻り値

なし

© 2000-2025, MetaQuotes Ltd.

SetDeviationInPoints

許可された偏差を設定します。

```
void  SetDeviationInPoints(
   ulong  deviation      // 偏差値
  )
```

パラメータ

偏差

[in] 許可された偏差値

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

SetTypeFilling

注文充填の種類を設定します。

```
void  SetTypeFilling(
   ENUM_ORDER_TYPE_FILLING  filling      // 注文充填の種類
  )
```

パラメータ

filling

[in] 注文充填の種類（ENUM_ORDER_TYPE_FILLING 列挙値）

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

SetTypeFillingBySymbol

指定された銘柄の設定によって注文の履行タイプを設定します。

```
bool  SetTypeFillingBySymbol(
   const string   symbol      // 銘柄名
   )
```

パラメータ

symbol

[in] SYMBOL_FILLING_MODEが許可された履行ポリシーを含む銘柄

戻り値

成功の場合はtrue、履行ポリシーを定義できなかった場合はfalse

注意事項

銘柄でSYMBOL_FILLING_FOK及びSYMBOL_FILLING_IOC履行ポリシーが同時に許可されている場合は、注文にORDER_FILLING_FOKが設定されます。

© 2000–2025, MetaQuotes Ltd.

SetAsyncMode

取引操作のための非同期モードを設定します。

```
void  SetAsyncMode(
   bool  mode       // 非同期モードフラグ
 )
```

パラメータ

mode

[in] 非同期モードフラグ

戻り値

なし

注意事項

このモードは（送信されたリクエストへの取引サーバの返答を待たない）非同期的な取引操作に使用されます（OrderSendAsync を参照）。

© 2000–2025, MetaQuotes Ltd.

SetMarginMode

現在の口座設定に応じて、証拠金計算モードを設定します。

```
void  SetMarginMode()
```

戻り値

Нет.

注意事項

証拠金計算モードは、ENUM_ACCOUNT_MARGIN_MODEで設定されます。

© 2000–2025, MetaQuotes Ltd.

OrderOpen

パラメータを使用して未決注文を出します。

```
bool  OrderOpen(
   const string           symbol,           // シンボル
   ENUM_ORDER_TYPE        order_type,       // 注文の種類
   double                 volume,           // 注文のボリューム
   double                 limit_price,      // ストップリミット価格
   double                 price,            // 実行価格
   double                 sl,               // 決済逆指値
   double                 tp,               // 決済指値
   ENUM_ORDER_TYPE_TIME   type_time,        // 期限によっての種類
   datetime               expiration,       // 期限
   const string           comment=""        // コメント
  )
```

パラメータ

symbol

[in] 取引商品の名称

order_type

[in] 取引操作の型（ENUM_ORDER_TYPE 列挙の値）

ボリューム

[in] リクエストされた注文ボリューム

limit_price

[in] 指値つきストップ注文が出される価格

価格

[in] 注文実行価格

sl

[in] 決済逆指のトリガ値

tp

[in] 決済指のトリガ値

type_time

[in] 実行による注文の種類（ENUM_ORDER_TYPE_TIME 列挙値）

expiration

[in] 未決注文の期限

comment=""

[in] 注文コメント

戻り値

基本構造体のチェックが成功した場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

注意事項

OrderSend(...) の正常な完成は、取引操作実行の成功を意味するものではありません。取引リクエストの結果（取引サーバのリターンコード）を ResultRetcode() 及び ResultOrder() から戻された値を使用してチェックすることが必要です。

© 2000–2025, MetaQuotes Ltd.

OrderModify

未決注文のパラメータを変更します。

```
bool  OrderModify(
   ulong                ticket,          // 注文チケット
   double               price,           // 実行価格
   double               sl,              // 決済逆指値
   double               tp,              // 決済指値
   ENUM_ORDER_TYPE_TIME type_time,       // 期限によっての種類
   datetime             expiration,      // 期限
   double               stoplimit        // リミット注文価格
  )
```

パラメータ

ticket

[in] 注文チケット

価格

[in] 新しい注文実行価格（変更が必要でない場合は以前の値）

sl

[in] 決済逆指のトリガとなる新しい価格（変更が必要でない場合は以前の値）

tp

[in] 決済指値のトリガとなる新しい価格（変更が必要でない場合は以前の値）

type_time

[in] 期限による注文の新しい種類（変更が必要でない場合は以前の値）。ENUM_ORDER_TYPE_TIME 列挙値。

expiration

[in] 未決注文の新しい期限（変更が必要でない場合は以前の値）

stoplimit

[in] 価格が price の値になった時にリミット注文設定するのに使用される新しい価格ストップリミット注文のみで指定されます。

戻り値

基本構造体のチェックが成功した場合は true、それ以外の場合は false

注意事項

OrderModify(...) メソッドの正常な完成は、取引操作実行の成功を意味するものではありません。ResultRetcode() を使用しての取引リクエストの結果（取引サーバのリターンコード）のチェックが必要です。

© 2000–2025, MetaQuotes Ltd.

OrderDelete

Deletes the pending order.

```
bool  OrderDelete(
   ulong  ticket      // 注文チケット
  )
```

パラメータ

ticket

[in] 注文チケット

戻り値

基本構造体のチェックが成功した場合は true、それ以外の場合は false

注意事項

OrderDelete(...) [ResultRetcode()] メソッドの正常な完成は、取引操作実行の成功を意味するものではありません。を使用しての取引リクエストの結果（取引サーバのリターンコード）のチェックが必要です。

© 2000–2025, MetaQuotes Ltd.

PositionOpen

指定されたパラメータで注文を出します。

```
bool  PositionOpen(
   const string      symbol,          // シンボル
   ENUM_ORDER_TYPE   order_type,      // 注文を出すための注文の種類
   double            volume,          // ポジションボリューム
   double            price,           // 実行価格
   double            sl,              // 決済逆指値
   double            tp,              // 決済指値
   const string      comment=""       >// コメント
  )
```

パラメータ

symbol

[in] 注文を出すのに使用される金融製品の名称

order_type

[in] 注文を出すのに使用される注文の種類（ENUM_ORDER_TYPE 列挙の値）

ボリューム

[in] リクエストされたポジションボリューム

価格

[in] ポジションを取ラなければならない価格

sl

[in] 決済逆指のトリガ値

tp

[in] 決済指のトリガ値

comment=""

[in] ポジションコメント

戻り値

基本構造体のチェックが成功した場合は true、それ以外の場合は false

注意事項

PositionOpen(...) メソッドの正常な完成は、取引操作実行の成功を意味するものではありません。取引リクエストの結果（取引サーバのリターンコード）を ResultRetcode() 及び ResultDeal() から戻された値を使用してチェックすることが必要です。

© 2000–2025, MetaQuotes Ltd.

PositionModify

指定されたシンボルでポジションパラメータを変更します。

```
bool  PositionModify(
   const string  symbol,      // シンボル
   double        sl,          // 決済逆指値
   double        tp           // 決済指値
  )
```

指定されたチケットでポジションパラメータを変更します。

```
bool  PositionModify(
   const ulong   ticket,      // position ticket
   double        sl,          // Stop Loss price
   double        tp           // Take Profit price
   )
```

パラメータ

symbol

[in] ポジション変更に使用される金融製品の名称

ticket

[in] 変更が予定されているポジションチケット。

sl

[in] 決済逆指のトリガとなる新しい価格（変更が必要でない場合は以前の値）

tp

[in] 決済指値のトリガとなる新しい価格（変更が必要でない場合は以前の値）

戻り値

基本構造体のチェックが成功した場合は true、それ以外の場合は false

注意事項

PositionModify(...)メソッドの正常な完了は、常に取引実行の成功を意味するものではありません。ResultRetcode()メソッドの呼び出しで取引リクエスト（取引サーバーのリターンコード）の実行結果を確認する必要があります。

ポジションの『ネッティング』計算時（ACCOUNT_MARGIN_MODE_RETAIL_NETTINGとACCOUNT_MARGIN_MODE_EXCHANGE）各シンボルにつき常に一つのポジションのみ（1つ以上の取引の結果である）保有することができます。ポジションと『ツールボックス』パネルの『取引』タブに表示される有効な未決注文と混同しないようにしてください。

ポジションに制限がない場合（ACCOUNT_MARGIN_MODE_RETAIL_HEDGING）各シンボルごとに同時に複数のポジションを保有することができます。この場合、PositionModifyは最小チケットのポジションを変更します。

© 2000–2025, MetaQuotes Ltd.

PositionClose

指定されたシンボルでポジションを決済します。

```
bool  PositionClose(
   const string  symbol,                  // シンボル
   ulong         deviation=ULONG_MAX      // 偏差値
  )
```

Closes a position with the specified ticket.

```
bool  PositionClose(
   const ulong   ticket,                  // Position ticket
   ulong         deviation=ULONG_MAX      // Deviation
   )
```

パラメータ

symbol

[in] ポジション決済に使用される金融製品の名称

ticket

[in] 決済するポジションのチケット。

deviation=ULONG_MAX

[in] 現在価格からの最大偏差値（ポイント単位）

戻り値

基本構造体のチェックが成功した場合はtrue、それ以外の場合はfalse

注意事項

PositionClose(...)メソッドの正常な完了は、常に取引実行の成功を意味するものではありません。ResultRetcode()メソッドの呼び出しで取引リクエスト（取引サーバーのリターンコード）の実行結果を確認する必要があります。

ポジションの『ネッティング』計算時（ACCOUNT_MARGIN_MODE_RETAIL_NETTINGとACCOUNT_MARGIN_MODE_EXCHANGE）各シンボルにつき常に一つのポジションのみ（1つ以上の取引の結果である）保有することができます。ポジションと『ツールボックス』パネルの『取引』タブに表示される有効な未決注文と混同しないようにしてください。

ポジションに制限がない場合（ACCOUNT_MARGIN_MODE_RETAIL_HEDGING）各シンボルごとに同時に複数のポジションを保有することができます。この場合、PositionCloseは、最小チケットのポジションを決済します。

© 2000–2025, MetaQuotes Ltd.

PositionClosePartial

「ヘッジ」会計の場合に、指定された銘柄のポジションを部分的に決済します。

```
bool  PositionClosePartial(
   const string  symbol,                  // 銘柄
   const double  volume,                  // 数量
   ulong         deviation=ULONG_MAX      // 偏差
   )
```

「ヘッジ」会計の場合に、指定された銘柄のポジションを部分的に決済します。

```
bool  PositionClosePartial(
   const ulong   ticket,                  // ポジションチケット
   const double  volume,                  // 数量
   ulong         deviation=ULONG_MAX      // 偏差
   )
```

パラメータ

symbol

[in] ポジションが部分的に決済される金融製品の名称。ポジションの部分的決済のために(チケットではなく)銘柄が指定されると、銘柄の指定されたマジックナンバー(エキスパートアドバイザーID)を持つ初めのポジションが選択されます。したがって、指定されたポジションチケットでPositionClosePartial() を使用する方が良い場合があります。

volume

[in] ポジションを減らす数量。値が部分的に決済されたポジションの数量を超えると、ポジションは完全に決済されます。反対方向のポジションは開かれません。

ticket

[in] 決済されたポジションチケット

deviation=ULONG_MAX

[in] 現在価格からの最大偏差値(ポイント単位)

戻り値

構造体の基本チェックが成功した場合はtrue、失敗した場合はfalse

注意事項

PositionClosePartial(...) メソッドの正常な完成は、取引操作実行の成功を意味するものではありません。 ResultRetcode() を使用しての取引リクエストの結果(取引サーバのリターンコード)のチェックが必要です。

「ネッティング」システム(ACCOUNT_MARGIN_MODE_RETAIL_NETTING及びACCOUNT_MARGIN_MODE_EXCHANGE)では、任意の時点で存在できる各銘柄のポジションは1つです。これは1つまたは複数の 約定の結果です。現在の未決注文 はクライアント端末の「ツールボックス」の「取引」タブに表示されているポジションと混乱されてはなりません。

ポジション表示(ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)では、各銘柄で複数のポジションを同時に開くことができます。この場合、PositionCloseはチケットが最も少ないポジションを決済します。

© 2000-2025, MetaQuotes Ltd.

© 2000–2025, MetaQuotes Ltd.

PositionCloseBy

反対ポジションで指定されたチケットのポジションを決済します。

```
bool  PositionCloseBy(
   const ulong   ticket,         // ポジションチケット
   const ulong   ticket_by       // 反対ポジションのチケット
   )
```

パラメータ

ticket

[in] 決済するポジションのチケット。

ticket_by

[in] 決済の為に使用される反対ポジションのチケット。

戻り値

構造の基本的なテストに成功した場合true、それ以外の場合はfalse。

注意事項

PositionCloseBy(...)メソッドの正常な完了は、常に取引実行の成功を意味するものではありません。ResultRetcode()メソッドの呼び出しで取引リクエスト（取引サーバーのリターンコード）の実行結果を確認する必要があります。

© 2000–2025, MetaQuotes Ltd.

Buy

指定されたパラメータで買い注文を出します。

```
bool  Buy(
   double         volume,           // ポジションボリューム
   const string   symbol=NULL,      // シンボル
   double         price=0.0,        // 価格
   double         sl=0.0,           // 決済逆指値
   double         tp=0.0,           // 決済指値
   const string   comment=""        // コメント
  )
```

パラメータ

ボリューム

[in] ポジションボリューム

symbol=NULL

[in] ポジションシンボル(シンボルが指定されていない場合は、現在のシンボルが使用されます)。

price=0.0

[in] 価格(価格が指定されていない場合は、現在の買値が使用されます)。

sl=0.0

[in] 決済逆指値

tp=0.0

[in] 決済指値

comment=""

[in] コメント

戻り値

構造体のチェックが成功した場合は true、それ以外の場合は false

注意事項

Buy(...) メソッドの正常な完成は、取引操作実行の成功を意味するものではありません。取引リクエストの結果(取引サーバの リターンコード)を ResultRetcode() 及び ResultDeal() から戻された値を使用してチェックすることが必要です。

© 2000-2025, MetaQuotes Ltd.

Sell

指定されたパラメータで売り注文を出します。

```
bool  Sell(
   double         volume,           // ポジションボリューム
   const string   symbol=NULL,      // シンボル
   double         price=0.0,        // 価格
   double         sl=0.0,           // 決済逆指値
   double         tp=0.0,           // 決済指値
   const string   comment=""        // コメント
  )
```

パラメータ

ボリューム

[in] ポジションボリューム

symbol=NULL

[in] ポジションシンボル(シンボルが指定されていない場合は、現在のシンボルが使用されます)。

price=0.0

[in] 価格価格が指定されていない場合は、現在の市場売値が使用されます。

sl=0.0

[in] 決済逆指値

tp=0.0

[in] 決済指値

comment=""

[in] コメント

戻り値

構造体のチェックが成功した場合は true、それ以外の場合は false

注意事項

Sell(...) メソッドの正常な完成は、取引操作実行の成功を意味するものではありません。取引リクエストの結果(取引サーバの リターンコード)を ResultRetcode() 及び ResultDeal() から戻された値を使用してチェックすることが必要です。

© 2000-2025, MetaQuotes Ltd.

BuyLimit

指定されたパラメータで買い指値注文（現在の市場価格より低い値段での買い）をします。

```
bool  BuyLimit(
   double                volume,                    // 注文ボリューム
   double                price,                     // 注文価格
   const string          symbol=NULL,               // シンボル
   double                sl=0.0,                    // 決済逆指値
   double                tp=0.0,                    // 決済指値
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,  // 注文の寿命
   datetime              expiration=0,              // 注文期限
   const string          comment=""                 // コメント
  )
```

パラメータ

ボリューム

[in] 注文ボリューム

価格

[in] 注文価格

symbol=NULL

[in] 注文シンボル（シンボルが指定されていない場合は、現在のシンボルが使用されます）。

sl=0.0

[in] 決済逆指値

tp=0.0

[in] 決済指値

type_time=ORDER_TIME_GTC

[in] 注文の寿命（ENUM_ORDER_TYPE_TIME 列挙値）

expiration=0

[in] 注文期限（ type_time=ORDER_TIME_SPECIFIED の時のみ使用されます）

comment=""

[in] 注文コメント

戻り値

構造体のチェックが成功した場合は true、それ以外の場合は false

注意事項

BuyLimit(...) メソッドの正常な完成は、取引操作実行の成功を意味するものではありません。取引リクエストの結果（取引サーバの リターンコード）を ResultRetcode() 及び ResultOrder() から戻された値を使用してチェックすることが必要です。

© 2000-2025, MetaQuotes Ltd.

BuyStop

指定されたパラメータで買い逆指注文（市場価格より高い価格での買い）をします。

```
bool  BuyStop(
   double                volume,                    // 注文ボリューム
   double                price,                     // 注文価格
   const string          symbol=NULL,               // シンボル
   double                sl=0.0,                    // 決済逆指値
   double                tp=0.0,                    // 決済指値
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,  // 注文の寿命
   datetime              expiration=0,              // 注文期限
   const string          comment=""                 // コメント
  )
```

パラメータ

ボリューム

[in] 注文ボリューム

価格

[in] 注文価格

symbol=NULL

[in] 注文シンボル（シンボルが指定されていない場合は、現在のシンボルが使用されます）。

sl=0.0

[in] 決済逆指値

tp=0.0

[in] 決済指値

type_time=ORDER_TIME_GTC

[in] 注文の寿命（ENUM_ORDER_TYPE_TIME 列挙値）

expiration=0

[in] 注文期限（type_time=ORDER_TIME_SPECIFIED の時のみ使用されます）

comment=""

[in] 注文コメント

戻り値

構造体のチェックが成功した場合は true、それ以外の場合は false

注意事項

BuyStop(...) メソッドの正常な完成は、取引操作実行の成功を意味するものではありません。取引リクエストの結果（取引サーバの リターンコード）を ResultRetcode() 及び ResultOrder() から戻された値を使用してチェックすることが必要です。

© 2000-2025, MetaQuotes Ltd.

SellLimit

指定されたパラメータで売り指値注文（現在の市場価格より高い値段での売り）をします。

```
bool  SellLimit(
   double                volume,                    // 注文ボリューム
   double                price,                     // 注文価格
   const string          symbol=NULL,               // シンボル
   double                sl=0.0,                    // 決済逆指値
   double                tp=0.0,                    // 決済指値
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,  // 注文の寿命
   datetime              expiration=0,              // 注文期限
   const string          comment=""                 // コメント
  )
```

パラメータ

ボリューム

[in] 注文ボリューム

価格

[in] 注文価格

symbol=NULL

[in] 注文シンボル（シンボルが指定されていない場合は、現在のシンボルが使用されます）。

sl=0.0

[in] 決済逆指値

tp=0.0

[in] 決済指値

type_time=ORDER_TIME_GTC

[in] 注文の寿命（ENUM_ORDER_TYPE_TIME 列挙値）

expiration=0

[in] 注文期限（type_time=ORDER_TIME_SPECIFIED の時のみ使用されます）

comment=""

[in] 注文コメント

戻り値

構造体のチェックが成功した場合は true、それ以外の場合は false

注意事項

SellLimit(...) メソッドの正常な完成は、取引操作実行の成功を意味するものではありません。 取引リクエストの結果（取引サーバの リターンコード）を ResultRetcode() 及び ResultOrder() から戻された値を使用してチェックすることが必要です。

© 2000-2025, MetaQuotes Ltd.

SellStop

指定されたパラメータで売り逆指注文（市場価格より低い価格での売り）をします。

```
bool  SellStop(
   double                volume,                    // 注文ボリューム
   double                price,                     // 注文価格
   const string          symbol=NULL,               // シンボル
   double                sl=0.0,                    // 決済逆指値
   double                tp=0.0,                    // 決済指値
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,  // 注文の寿命
   datetime              expiration=0,              // 注文期限
   const string          comment=""                 // コメント
  )
```

パラメータ

ボリューム

[in] 注文ボリューム

価格

[in] 注文価格

symbol=NULL

[in] 注文シンボル（シンボルが指定されていない場合は、現在のシンボルが使用されます）。

sl=0.0

[in] 決済逆指値

tp=0.0

[in] 決済指値

type_time=ORDER_TIME_GTC

[in] 注文の寿命（ENUM_ORDER_TYPE_TIME 列挙値）

expiration=0

[in] 注文期限（type_time=ORDER_TIME_SPECIFIED の時のみ使用されます）

comment=""

[in] 注文コメント

戻り値

構造体のチェックが成功した場合は true、それ以外の場合は false

注意事項

SellStop(...) メソッドの正常な完成は、取引操作実行の成功を意味するものではありません。取引リクエストの結果（取引サーバの リターンコード）を ResultRetcode() 及び ResultOrder() から戻された値を使用してチェックすることが必要です。

© 2000-2025, MetaQuotes Ltd.

Request

直近のリクエスト構造体のコピーを取得します。

```
void  Request(
   MqlTradeRequest&  request      // コピー元の構造体
   ) const
```

パラメータ

request

[out] MqlTradeRequest 型の構造体への参照

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

RequestAction

取引操作の種類を取得します。

```
ENUM_TRADE_REQUEST_ACTIONS  RequestAction() const
```

戻り値

直近のリクエストで使用された取引操作の種類

© 2000–2025, MetaQuotes Ltd.

RequestActionDescription

取引操作の種類を文字列として取得します。

```
string  RequestActionDescription() const
```

戻り値

文字列としての取引操作の種類

© 2000–2025, MetaQuotes Ltd.

RequestMagic

エキスパート アド バイザーのマジックナンバーを取 得します。

```
ulong  RequestMagic() const
```

戻り値

直 近 のリクエスト で使 用 されたエキスパート アド バイザーのマジックナンバー（ID）

© 2000–2025, MetaQuotes Ltd.

RequestOrder

直近のリクエストで使用された注文チケットを取得します。

```
ulong  RequestOrder() const
```

戻り値

直近のリクエストで使用された注文チケット

© 2000–2025, MetaQuotes Ltd.

RequestSymbol

直近のリクエストで使用された銘柄名を取得します。

```
string  RequestSymbol() const
```

戻り値

直近のリクエストで使用された銘柄名

© 2000–2025, MetaQuotes Ltd.

RequestVolume

直近のリクエストで使用された(ロット単位の)取引高を取得します。

```
double  RequestVolume() const
```

戻り値

直近のリクエストで使用された(ロット単位の)取引高

© 2000–2025, MetaQuotes Ltd.

RequestPrice

直近のリクエストで使用された価格を取得します。

```
double  RequestPrice() const
```

戻り値

直近のリクエストで使用された価格

© 2000–2025, MetaQuotes Ltd.

RequestStopLimit

直近のリクエストで使用された未決済の指値つきストップ注文の価格を取得します。

```
double  RequestStoplimit() const
```

戻り値

直近のリクエストで使用された未決済の指値つきストップ注文の価格

© 2000–2025, MetaQuotes Ltd.

RequestSL

直近のリクエストで使用された決済逆指注文の価格を取得します。

```
double  RequestSL() const
```

戻り値

直近のリクエストで使用された決済逆指注文の価格

© 2000–2025, MetaQuotes Ltd.

RequestTP

直近のリクエストで使用された決済指注文の価格を取得します。

```
double  RequestTP() const
```

戻り値

直近のリクエストで使用された決済指注文の価格

© 2000–2025, MetaQuotes Ltd.

RequestDeviation

直近のリクエストで使用された注文の価格偏差値を取得します。

```
ulong  RequestDeviation() const
```

戻り値

直近のリクエストで使用された注文の価格偏差値

© 2000-2025, MetaQuotes Ltd.

RequestType

直近のリクエストで使用された注文の種類を取得します。

```
ENUM_ORDER_TYPE  RequestType() const
```

戻り値

直近のリクエストで使用された注文の種類（ENUM_ORDER_TYPE 列挙値）

© 2000–2025, MetaQuotes Ltd.

RequestTypeDescription

直近のリクエストで使用された注文の種類を文字列として取得します。

```
string  RequestTypeDescription() const
```

戻り値

文字列としての直近のリクエストで使用された注文の種類

© 2000–2025, MetaQuotes Ltd.

RequestTypeFilling

直近のリクエストで使用された注文充填の種類を取得します。

```
ENUM_ORDER_TYPE_FILLING  RequestTypeFilling() const
```

戻り値

直近のリクエストで使用された注文充填の種類（ENUM_ORDER_TYPE_FILLING 列挙値）

© 2000–2025, MetaQuotes Ltd.

RequestTypeFillingDescription

直近のリクエストで使用された注文充填の種類を文字列として取得します。

```
string  RequestTypeFillingDescription() const
```

戻り値

文字列としての直近のリクエストで使用された注文充填の種類

© 2000–2025, MetaQuotes Ltd.

RequestTypeTime

直近のリクエストで使用された注文の期限を取得します。

```
ENUM_ORDER_TYPE_TIME  RequestTypeTime() const
```

戻り値

直近のリクエストで使用された注文の期限（ENUM_ORDER_TYPE_TIME 列挙値）

© 2000–2025, MetaQuotes Ltd.

RequestTypeTimeDescription

直近のリクエストで使用された注文の期限を文字列として取得します。

```
string  RequestTypeTimeDescription() const
```

戻り値

文字列としての直近のリクエストで使用された注文の期限

© 2000–2025, MetaQuotes Ltd.

RequestExpiration

直近のリクエストで使用された注文の期限を取得します。

```
datetime  RequestExpiration() const
```

戻り値

直近のリクエストで使用された注文の期限

© 2000–2025, MetaQuotes Ltd.

RequestComment

直近のリクエストで使用された注文のコメントを取得します。

```
string  RequestComment() const
```

戻り値

直近のリクエストで使用された注文のコメント

© 2000–2025, MetaQuotes Ltd.

RequestPosition

ポジションチケットを取得します。

```
ulong  RequestPosition() const
```

戻り値

最後のリクエストで使用されたポジションチケット。

© 2000–2025, MetaQuotes Ltd.

RequestPositionBy

反対ポジションのチケットを取得します。

```
ulong  RequestPositionBy() const
```

戻り値

最後のリクエストで使用された反対ポジションのチケット。

© 2000–2025, MetaQuotes Ltd.

Result

直近のリクエスト結果の構造体を取得します。

```
void  Result(
   MqlTradeResult&  result      // 参照
   ) const
```

パラメータ

result

[out] MqlTradeResult 型の構造体への参照

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

ResultRetcode

リクエスト結果のコードを取得します。

```
uint  ResultRetcode() const
```

戻り値

リクエスト結果のコード

© 2000–2025, MetaQuotes Ltd.

ResultRetcodeDescription

リクエスト結果のコードをテキストとして取得します。

```
string  ResultRetcodeDescription() const
```

戻り値

直近のリクエストのコードのテキストとしての結果

© 2000–2025, MetaQuotes Ltd.

ResultDeal

約定チケットを取得します。

```
ulong  ResultDeal() const
```

戻り値

約定が実行された場合は約定チケット

© 2000–2025, MetaQuotes Ltd.

ResultOrder

注文チケットを取得します。

```
ulong  ResultOrder() const
```

戻り値

注文された場合は注文チケット

© 2000–2025, MetaQuotes Ltd.

ResultVolume

約定または注文のボリュームを取得します。

```
double  ResultVolume() const
```

戻り値

約定または注文のボリューム

© 2000-2025, MetaQuotes Ltd.

ResultPrice

ブローカーによって確認された価格を取得します。

```
double  ResultPrice() const
```

戻り値

ブローカーによって確認された価格

© 2000–2025, MetaQuotes Ltd.

ResultBid

現在の売値を取得します。

```
double  ResultBid() const
```

戻り値

現在の売値

© 2000–2025, MetaQuotes Ltd.

ResultAsk

現在の買値を取得します。

```
double  ResultAsk() const
```

戻り値

現在の買値

© 2000–2025, MetaQuotes Ltd.

ResultComment

ブローカーコメントを取得します。

```
string  ResultComment() const
```

戻り値

操作のブローカーコメント

© 2000-2025, MetaQuotes Ltd.

CheckResult

直近のリクエストチェック結果の構造体を取得します。

```
void  CheckResult(
   MqlTradeCheckResult&  check_result      // 参照
   ) const
```

パラメータ

check_result

[out] MqlTradeCheckResult 型のターゲット構造体への参照

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

CheckResultRetcode

リクエストの正当さをチェックしながら満たされたMqlTradeCheckResult 型の retcode フィールド の値を取得します。

```
uint  CheckResultRetcode() const
```

戻り値

リクエストの正当さをチェックしながら満たされた MqlTradeCheckResult 型の retcode（エラーコード）フィールド の値を取得します。

© 2000–2025, MetaQuotes Ltd.

CheckResultRetcodeDescription

リクエストの正当さをチェックしながら満たされた MqlTradeCheckResult 型の retcode フィールドの文字列としての説明を取得します。

```
string  ResultRetcodeDescription() const
```

戻り値

リクエストの正当さをチェックしながら満たされた MqlTradeCheckResult 型の retcode（エラーコード）フィールドの文字列としての説明を取得します。

© 2000–2025, MetaQuotes Ltd.

CheckResultBalance

リクエストの正当さをチェックしながら満たされた MqlTradeCheckResult 型の balance フィールドの値を取得します。

```
double  CheckResultBalance() const
```

戻り値

リクエストの正当さをチェックしながら満たされた MqlTradeCheckResult 型の balance フィールド（取引操作実行の後の残高値）の値

© 2000–2025, MetaQuotes Ltd.

CheckResultEquity

リクエストの正当さをチェックしながら満たされた MqlTradeCheckResult 型の equity フィールド の値を取得します。

```
double  CheckResultEquity() const
```

戻り値

リクエストの正当さをチェックしながら満たされた MqlTradeCheckResult 型の equity フィールド（取引操作実行の後の株式の値）の値を取得します。

© 2000–2025, MetaQuotes Ltd.

CheckResultProfit

リクエストの正当さをチェックしながら満たされた MqlTradeCheckResult 型の profit フィールド の値を取得します。

```
double  CheckResultProfit() const
```

戻り値

リクエストの正当さをチェックしながら満たされた MqlTradeCheckResult 型の profit フィールド（取引操作実行の後の利益の値）の値を取得します。

© 2000–2025, MetaQuotes Ltd.

CheckResultMargin

リクエストの正当さをチェックしながら満たされた MqlTradeCheckResult 型の margin フィールドの値を取得します。

```
double  CheckResultMargin() const
```

戻り値

リクエストの正当さをチェックしながら満たされた MqlTradeCheckResult 型の margin フィールド（取引操作に必要な証拠金）の値を取得します。

© 2000–2025, MetaQuotes Ltd.

CheckResultMarginFree

リクエストの正当さをチェックしながら満たされた MqlTradeCheckResult 型の margin_free フィールドの値を取得します。

```
double CheckResultMarginFree() const
```

戻り値

リクエストの正当さをチェックしながら満たされた MqlTradeCheckResult 型の margin_free フィールド（取引操作実行の後の余剰証拠金）の値を取得します。

© 2000–2025, MetaQuotes Ltd.

CheckResultMarginLevel

リクエストの正当さをチェックしながら満たされた MqlTradeCheckResult 型の margin_level フィールドの値を取得します。

```
double   CheckResultMarginLevel() const
```

戻り値

リクエストの正当さをチェックしながら満たされたMqlTradeCheckResult 型の margin_level（取引操作実行の後で設定される証拠金レベル）フィールドの値を取得します。

© 2000–2025, MetaQuotes Ltd.

CheckResultComment

リクエストの正当さをチェックしながら満たされた MqlTradeCheckResult 型の comment フィールド の値を取得します。

```
string  CheckResultComment() const
```

戻り値

リクエストの正当さをチェックしながら満たされた MqlTradeCheckResult 型の comment フィールド（レプライコードへのコメントとエラーの記述）の値を取得します。

© 2000–2025, MetaQuotes Ltd.

PrintRequest

直近のリクエストのパラメータを操作ログにプリントします。

```
void  PrintRequest() const
```

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

PrintResult

直近のリクエストの結果を操作ログにプリントします。

```
void  PrintResult() const
```

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

FormatRequest

直近のリクエストのパラメータのフォーマットされた文字列を準備します。

```
string  FormatRequest(
   string&                  str,          // ターゲット文字列
   const MqlTradeRequest&   request      // リクエスト
   ) const
```

パラメータ

str

[in] 参照で渡されたターゲット文字れ越

request

[in] 直近のリクエストのパラメータを持つMqlTradeRequest 型の構造体

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

FormatRequestResult

直近のリクエストの結果のフォーマットされた文字列を準備します。

```
string  FormatRequestResult(
   string&                  str,          // 文字列
   const MqlTradeRequest&   request,      // リクエスト構造体
   const MqlTradeResult&    result        // リクエスト構造体
   ) const
```

パラメータ

str

[in] 参照で渡されたターゲット文字れ越

request

[in] 直近のリクエストのパラメータを持つ MqlTradeRequest 型の構造体

result

[in] 直近のリクエストの結果を持つ MqlTradeResult 型の構造体

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

CTerminalInfo

CTerminalInfo はMQL5 プログラム環境のプロパティに簡単にアクセスするためのクラスです。

説明

CTerminalInfo クラスはMQL5 プログラム環境のプロパティへの簡易化されたアクセスを提供します。

宣言

```
class CTerminalInfo : public CObject
```

タイトル

```
#include <Trade\TerminalInfo.mqh>
```

継承階層

CObject

CTerminalInfo

グループ分けされたクラスメソッド

整数型プロパティの新しい値の設定	
Build	クライアント端末のビルド番号を取得します。
IsConnected	取引サーバへの接続に関する情報を取得します。
IsDLLsAllowed	DLLの使用の許可に関する情報を取得します。
IsTradeAllowed	取引許可に関する情報を取得します。
IsEmailEnabled	端末の設定で指定されたSMTP　サーバへの電子メールの送信及びログインの許可に関する情報を取得します。
IsFtpEnabled	端末の設定で指定された FTP サーバへの取引レポート送信及びログインの許可に関する情報を取得します。
MaxBars	チャート上のバーの最大数についての情報を取得します。
CodePage	クライアント端末の言語のコードページについての情報を取得します。
CPUCores	CPU コアについての情報を取得します。
MemoryPhysical	物理メモリについての情報を取得します。(MB 単位)。
MemoryTotal	端末/エージェントプロセスによって使用可能な合計メモリに関する情報を取得します。(MB 単位)。
MemoryAvailable	端末/エージェントプロセスによって使用可能な空きメモリに関する情報を取得します。(MB 単位)。

© 2000-2025, MetaQuotes Ltd.

整数型プロパティの新しい値の設定	
MemoryUsed	端末/エージェントプロセスに使用されているメモリに関する情報を取得します。(MB単位)。
IsX64	クライアント端末の種類に関する情報(32/64ビット)を取得します。
OpenCLSupport	ビデオカードがサポートしている OpenCL のバージョンに関する情報を取得します。
DiskSpace	ディスクの空き領域についての情報を取得します。(MB単位)。
string型のプロパティへのアクセスメソッド	
Language	クライアント端末の言語を取得します。
Name	クライアント端末の名称を取得します。
Company	クライアント端末の会社を取得します。
Path	クライアント端末のフォルダを取得します。
DataPath	クライアント端末のデータフォルダを取得します。
CommonDataPath	コンピュータにインストールされている全てのクライアント端末の共通データフォルダを取得します。
MQL5 API関数へのアクセス	
InfoInteger	整数型パラメータの値を取得します。
InfoString	string型パラメータの値を取得します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

Build

クライアント端末のビルド番号を取得します。

```
int  CBuild() const
```

戻り値

クライアント端末のビルド番号

注意事項

ビルド番号の取得には TerminalInfoInteger() 関数（TERMINAL_BUILD プロパティ）が使用されます。

© 2000–2025, MetaQuotes Ltd.

IsConnected

取引サーバへの接続に関する情報を取得します。

```
bool  IsConnected() const
```

戻り値

取引サーバに接続されている場合は true、それ以外の場合は false

注意事項

取引状態の取得には TerminalInfoInteger() 関数（TERMINAL_CONNECTED プロパティ）が使用されます。

© 2000–2025, MetaQuotes Ltd.

IsDLLsAllowed

DLLの使用の許可に関する情報を取得します。

```
bool  IsDLLsAllowed() const
```

戻り値

DLLの使用が許可されている場合はtrue、それ以外の場合はfalse

注意事項

DLLの使用許可についての情報の取得には TerminalInfoInteger() 関数（TERMINAL_DLLS_ALLOWED プロパティ）が使用されます。

© 2000–2025, MetaQuotes Ltd.

IsTradeAllowed

取引許可に関する情報を取得します。

```
bool  IsTradeAllowed() const
```

戻り値

取引が許可されている場合は true、それ以外の場合は false

注意事項

取引許可に関する情報の取得には TerminalInfoInteger() 関数（TERMINAL_TRADE_ALLOWED プロパティ）が使用されます。

© 2000–2025, MetaQuotes Ltd.

IsEmailEnabled

端末の設定で指定されたSMTP サーバへの電子メールの送信及びログインの許可に関する情報を取得します。

```
bool  IsEmailEnabled() const
```

戻り値

電子メールの送信が許可されている場合は true、それ以外の場合は false

注意事項

電子メールの送信許可の情報取得には TerminalInfoInteger() 関数（TERMINAL_EMAIL_ENABLED プロパティ）が使用されます。

© 2000–2025, MetaQuotes Ltd.

IsFtpEnabled

端末の設定で指定された FTP サーバへの取引レポート送信及びログインの許可に関する情報を取得します。

```
bool  IsFtpEnabled() const
```

戻り値

FTP サーバへの取引レポート送信が許可されている場合は true、それ以外の場合は false

注意事項

取引レポート送信許可についての情報の取得には TerminalInfoInteger() 関数（TERMINAL_FTP_ENABLED プロパティ）が使用されます。

© 2000–2025, MetaQuotes Ltd.

MaxBars

クライアント端末設定で指定された、チャート上のバーの最大数を取得します。

```
int  MaxBars() const
```

戻り値

チャート上のバーの最大数

注意事項

チャート上のバーの最大数の取得には TerminalInfoInteger() 関数（TERMINAL_MAXBARS プロパティ）が使用されます。

© 2000–2025, MetaQuotes Ltd.

CodePage

クライアント端末の言語のコードページについての情報を取得します。

```
int  CodePage() const
```

戻り値

クライアント端末の言語のコードページ

注意事項

コードページの取得には TerminalInfoInteger() 関数（TERMINAL_CODEPAGE プロパティ）が使用されます。

© 2000–2025, MetaQuotes Ltd.

CPUCores

システム内のCPUコアの数についての情報を取得します。

```
int  CPUCores() const
```

戻り値

システム内のCPUコアの数

注意事項

CPUコア数の取得には TerminalInfoInteger() 関数（TERMINAL_CPU_CORES プロパティ）が使用されます。

© 2000–2025, MetaQuotes Ltd.

MemoryPhysical

物理メモリについての情報を取得します。(MB単位)。

```
int  MemoryPhysical() const
```

戻り値

物理メモリ(MB単位)

注意事項

物理メモリについての情報の取得には TerminalInfoInteger() 関数(TERMINAL_MEMORY_PHYSICAL プロパティ)が使用されます。

© 2000-2025, MetaQuotes Ltd.

MemoryTotal

クライアント端末/エージェントによって使用可能な合計メモリに関する情報を取得します。(MB単位)。

```
int  MemoryTotal() const
```

戻り値

端末/エージェントによって使用可能な合計メモリ(MB単位)

注意事項

合計メモリ情報の取得には TerminalInfoInteger() 関数(TERMINAL_MEMORY_TOTAL プロパティ)が使用されます。

© 2000-2025, MetaQuotes Ltd.

MemoryAvailable

クライアント端末/エージェントによって使用可能な空きメモリに関する情報を取得します。(MB単位)。

```
int  MemoryTotal() const
```

戻り値

端末/エージェントによって使用可能な空きメモリ(MB単位)

注意事項

空きメモリの取得には TerminalInfoInteger() 関数(TERMINAL_MEMORY_TOTAL プロパティ)が使用されます。

© 2000-2025, MetaQuotes Ltd.

MemoryUsed

クライアント端末/エージェントプロセスに使用されているメモリに関する情報を取得します。(MB単位)。

```
int  MemoryUsed() const
```

戻り値

クライアント端末/エージェントプロセスに使用されているメモリ(MB単位)

注意事項

メモリ情報の取得には TerminalInfoInteger() 関数(TERMINAL_MEMORY_USED プロパティ)が使用されます。

© 2000-2025, MetaQuotes Ltd.

IsX64

クライアント端末の種類に関する情報を取得します。

```
bool  IsX64() const
```

戻り値

64ビット版が使用されている場合はtrue、それ以外の場合はfalse

注意事項

クライアント端末の種類を取得するには TerminalInfoInteger() 関数（TERMINAL_X64 プロパティ）が使用されます。

© 2000–2025, MetaQuotes Ltd.

OpenCLSupport

ビデオカードがサポートしている OpenCL のバージョンに関する情報を取得します。

```
int  OpenCLSupport() const
```

戻り値

戻り値の形式は 0x00010002 = "1.2" です。0 は OpenCL のサポートがないことを示します。

注意事項

OpenCL のバージョンの取得には TerminalInfoInteger() 関数（TERMINAL_OPENCL_SUPPORT プロパティ）が使用されます。

© 2000–2025, MetaQuotes Ltd.

DiskSpace

クライアント端末/エージェントによって使用可能なディスクの空き領域についての情報を取得します。(MB単位)。

```
int  MDiskSpace() const
```

戻り値

クライアント端末/エージェント(MQL5¥Files フォルダに保存されたファイル)に使用可能なディスクの空き領域

注意事項

ディスクの空き領域の取得には TerminalInfoInteger() 関数(TERMINAL_DISK_SPACE プロパティ)が使用されます。

© 2000-2025, MetaQuotes Ltd.

Language

クライアント端末の言語についての情報を取得します。

```
string  Language() const
```

戻り値

クライアント端末で使用されている言語

注意事項

言語の取得には TerminalInfoString()関数（TERMINAL_LANGUAGE プロパティ）が使用されます。

© 2000–2025, MetaQuotes Ltd.

Name

クライアント端末の名称を取得します。

```
string  Name() const
```

戻り値

クライアント端末の名称.

注意事項

クライアント端末の名称の取得には TerminalInfoString() 関数（TERMINAL_NAME プロパティ）が使用されます。

© 2000–2025, MetaQuotes Ltd.

Company

ブローカー名についての情報を取得します。

```
string  Company() const
```

戻り値

ブローカー名

注意事項

ブローカー名の取得には TerminalInfoString() 関数（TERMINAL_COMPANY プロパティ）が使用されます。

© 2000–2025, MetaQuotes Ltd.

Path

クライアント端末フォルダについての情報を取得します。

```
string  Path() const
```

戻り値

クライアント端末フォルダ

注意事項

クライアント端末フォルダの取得には TerminalInfoString() （TERMINAL_PATH プロパティ）が使用されます。

© 2000–2025, MetaQuotes Ltd.

DataPath

端末データフォルダについての情報を取得します。

```
string  DataPath() const
```

戻り値

クライアント端末データフォルダ

注意事項

端末データフォルダの取得には TerminalInfoString() 関数（TERMINAL_DATA_PATH プロパティ）が使用されます。

© 2000–2025, MetaQuotes Ltd.

CommonDataPath

コンピュータにインストールされている全てのクライアント端末の共通データフォルダを取得します。

```
string  CommonDataPath() const
```

戻り値

共通データフォルダ

注意事項

共通データフォルダの取得には TerminalInfoString() 関数（COMMON_DATA_PATH プロパティ）が使用されます。

© 2000–2025, MetaQuotes Ltd.

InfoInteger

MQL5 プログラム環境の対応するプロパティの値を返します。

```
int  TerminalInfoInteger(
   int  property_id      // プロパティ識別子
   );
```

パラメータ

property_id

[in] プロパティの識別子。ENUM_TERMINAL_INFO_INTEGER 列挙のいずれかです。

戻り値

int 型の値

注意事項

プロパティ値の取得には TerminalInfoInteger() 関数が使用されます。

© 2000–2025, MetaQuotes Ltd.

InfoString

MQL5 プログラム環境の対応するプロパティの値を返します。プロパティは文字列型でなければなりません。

```
string  TerminalInfoString(
   int  property_id      // プロパティ識別子
   );
```

パラメータ

property_id

[in] プロパティの識別子。多分 ENUM_TERMINAL_INFO_STRING 列挙のいずれかです。

戻り値

string 型の値

注意事項

プロパティ値の取得には TerminalInfoString() 関数が使用されます。

© 2000–2025, MetaQuotes Ltd.

取引ストラテジークラス

このセクションには、取引ストラテジーを作成とテストするクラスの操作に関する技術的な詳細と関連するMQL5 標準ライブラリコンポーネントの説明が含まれています。

これらのクラスの使用は、取引ストラテジーの作成時に時間を節約します。

MQL5 標準ライブラリの取引ストラテジーに関連したファイルは、端末ディレクトリの Include¥Expertフォルダ内に配置されています。

基本クラス	説明
CExpertBase	全ての取引ストラテジークラスの基本クラス
CExpert	エキスパートアドバイザーの基本クラス
CExpertSignal	売買シグナルの基本クラス
CExpertTrailing	トレール注文の基本クラス
CExpertMoney	資金管理の基本クラス

売買シグナルクラス	説明
CSignalAC	ＡＣオシレーター指標のマーケットモデルに基づいたシグナルモジュール
CSignalAMA	適応型移動平均指標のマーケットモデルに基づいたシグナルモジュール
CSignalAO	オーサムオシレーター指標のマーケットモデルに基づいたシグナルモジュール
CSignalBearsPower	ベアパワーオシレーター指標のマーケットモデルに基づいたシグナルモジュール
CSignalBullsPower	ブルパワーオシレーター指標のマーケットモデルに基づいたシグナルモジュール
CSignalCCI	CCI(コモディティチャンネルインデックス)指標のマーケットモデルに基づいたシグナルモジュール
CSignalDeM	デマーカー指標のマーケットモデルに基づいたシグナルモジュール
CSignalDEMA	2 重指数移動平均指標のマーケットモデルに基づいたシグナルモジュール
CSignalEnvelopes	エンベロープ指標のマーケットモデルに基づいたシグナルモジュール
CSignalFrAMA	フラクタル適応型移動平均指標のマーケットモデルに基づいたシグナルモジュール
CSignalITF	時間によるシグナルフィルタのモジュール
CSignalMACD	MACD オシレーターのマーケットモデルに基づいたシグナルモジュール
CSignalMA	移動平均指標のマーケットモデルに基づいたシグナルモジュール

© 2000–2025, MetaQuotes Ltd.

売買シグナルクラス	説明
CSignalSAR	パラボリックSAR指標のマーケットモデルに基づいたシグナルモジュール
CSignalRSI	相対力指数オシレーターのマーケットモデルに基づいたシグナルモジュール
CSignalRVI	相対活力指数オシレーターのマーケットモデルに基づいたシグナルモジュール
CSignalStoch	ストキャスティックス一指標のマーケットモデルに基づいたシグナルモジュール
CSignalTRIX	3重指数平均指標のマーケットモデルに基づいたシグナルモジュール
CSignalTEMA	3 重指数移動平均指標のマーケットモデルに基づいたシグナルモジュール
CSignalWPR	ウィリアムパーセントレンジ指標のマーケットモデルに基づいたシグナルモジュール

トレール注文のクラス	説明
CTrailingFixedPips	このクラスは、固定点に基づいたトレール注文のアルゴリズムを実装します。
CTrailingMA	このクラスは、移動平均指標の値に基づいたトレール注文のアルゴリズムを実装します。
CTrailingNone	スタブクラスで、トレール注文のアルゴリズムを使用しません。
CTrailingPSAR	このクラスは、パラボリック SAR 指標の値に基づいたトレール注文のアルゴリズムを実装します。

資金管理のクラス	説明
CMoneyFixedLot	事前に定義された固定ロットサイズでの取引に基づくアルゴリズムを持つクラス
CMoneyFixedMargin	事前に定義された固定証拠金での取引に基づくアルゴリズムを持つクラス
CMoneyFixedRisk	事前に定義されたリスクでの取引に基づくアルゴリズムを持つクラス
CMoneyNone	事前に定義された最小限のロットサイズでの取引に基づくアルゴリズムを持つクラス
CMoneySizeOptimized	前の取引の結果に応じて変更されたロットサイズでの取引に基づくアルゴリズムを持つクラス

© 2000–2025, MetaQuotes Ltd.

エキスパートアドバイザーの基本クラス

このセクションには、取引ストラテジーを作成とテストするクラスの操作に関する技術的な詳細と関連するMQL5標準ライブラリコンポーネントの説明が含まれています。

これらのクラスの使用は、取引ストラテジーの作成時に時間を節約します。

MQL5標準ライブラリの取引ストラテジーに関連したファイルは、端末ディレクトリのInclude¥Expertフォルダ内に配置されています。

Class	説明
CExpertBase	全ての取引ストラテジークラスの基本クラス
CExpert	エキスパートアドバイザーの基本クラス
CExpertSignal	売買シグナルの基本クラス
CExpertTrailing	トレール注文の基本クラス
CExpertMoney	資金管理の基本クラス

© 2000–2025, MetaQuotes Ltd.

CExpertBase

CExpertBase は CExpert クラス及び全ての取引ストラテジークラスの基本クラスです。

説明

CExpertBase は全てのエキスパートアドバイザーオブジェクトに共通なデータとメソッドを提供します。

宣言

```
class CExpertBase : public CObject
```

タイトル

```
#include <Expert\ExpertBase.mqh>
```

継承階層

CObject

CExpertBase

直接子孫

CExpert, CExpertMoney, CExpertSignal, CExpertTrailing

クラスメソッド

Public メソッド

初期化	
virtual Init	クラスインスタンス初期化メソッド。
virtual ValidationSettings	設定をチェックします。
パラメータ	
Symbol	シンボルを設定します。
Period	時間軸を設定します。
Magic	エキスパートアドバイザー IDを設定します。
指標と時系列	
virtual SetPriceSeries	外部の時系列・価格列へのポインタを設定します。
virtual SetOtherSeries	外部の時系列（価格列以外）へのポインタを設定します。
virtual InitIndicators	指標と時系列を初期化します。
Protected データへのアクセス	
InitPhase	オブジェクト初期化の現在の段階を取得します。
TrendType	トレンドタイプを設定します。

© 2000–2025, MetaQuotes Ltd.

初期化	
UsedSeries	使用される時系列のビットマスクを取得します。
EveryTick	「Every tick（ティックごと）」フラグを設定します。
時系列へのアクセス	
Open	「Open（始値）」時系列の要素をインデックスで取得します。
High	「High（高値）」時系列の要素をインデックスで取得します。
Low	「Low（安値）」時系列の要素をインデックスで取得します。
Close	「Close（終値）」時系列の要素をインデックスで取得します。
Spread	「Spread（スプレッド）」時系列の要素をインデックスで取得します。
Time	「Time（時刻）」時系列の要素をインデックスで取得します。
TickVolume	「TickVolume（ティックボリューム）」時系列の要素をインデックスで取得します。
RealVolume	「RealVolume（実のボリューム）」時系列の要素をインデックスで取得します。

Protected メソッド

時系列の初期化	
InitOpen	「Open（始値）」時系列の初期化メソッド。
InitHigh	「High（高値）」時系列の初期化メソッド。
InitLow	「Low（安値）」時系列の初期化メソッド。
InitClose	「Close（終値）」時系列の初期化メソッド。
InitSpread	「Spread（スプレッド）」時系列の初期化メソッド。
InitTime	「Time（時刻）」時系列の初期化メソッド。
InitTickVolume	「TickVolume（ティックボリューム）」時系列の初期化メソッド。
InitRealVolume	「RealVolume（実のボリューム）」時系列の初期化メソッド。
サービスメソッド	
virtual PriceLevelUnit	価格レベル単位を取得します。
virtual StartIndex	分析される最初のバーのインデックスを取得します。
virtual CompareMagic	エキスパートアドバイザーIDを指定された値と比較します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000–2025, MetaQuotes Ltd.

InitPhase

オブジェクト初期化の現在の段階を取得します。

```
ENUM_INIT_PHASE  InitPhase()
```

戻り値

オブジェクト初期化の現在の段階

注意事項

オブジェクト初期化はいくつかの段階によって成り立ちます。

1. 初期化の開始

。 -開始 -コンストラクタ終了の後
-終了 -Init(...) メソッドの正常な終了の後
-可能 -Init(...) メソッドの呼び出し
-不可能 -ValidationSettings() メソッド及び他の初期化メソッドの呼び出し

2. パラメータの設定この段階では、指標の作成に使用される全てのオブジェクトのパラメータが設定される必要があります。

-開始 -Init(...) メソッドの正常な終了の後
-終了 -ValidationSettings() メソッドの正常な終了の後
-可能 -Symbol(...) 及び Period(...) メソッドの呼び出し
-不可能 -Init(...)、SetPriceSeries(...)、SetOtherSeries(...) 及び InitIndicators(...) メソッドの呼び出し

3. パラメータのチェック

-開始 -ValidationSettings() メソッドの正常な終了の後
-終了 -InitIndicators(...) メソッドの正常な終了の後
-可能 -Symbol(...)、Period(...) 及び InitIndicators(...) メソッドの呼び出し
-不可能 -他の初期化メソッドの呼び出し

4. 初期化の終了

-開始 -InitIndicators(...) メソッドの正常な終了の後
-不可能 -初期化メソッドの呼び出し

© 2000-2025, MetaQuotes Ltd.

TrendType

トレンドタイプを設定します。

```
void  TrendType(
   M_TYPE_TREND    value          // 新しい値
  )
```

パラメータ

value

[in] トレンドタイプの新しい値

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

UsedSeries

使用される時系列のビットマスクを取得します。

```
int  UsedSeries()
```

戻り値

ビットマスクとしての使用される時系列のリスト

注意事項

ビットが設定されている場合には対応する時系列が使用され、設定されていない場合には対応する時系列が使用されません。

ビットと時系列の対象

ビット 0 - Open 時系列
ビット 1 - High 時系列
ビット 2 - Low 時系列
ビット 3 - Close 時系列
ビット 4 - Spread 時系列
ビット 5 - Time 時系列
ビット 6 - TickVolume 時系列
ビット 7 - RealVolume 時系列

© 2000-2025, MetaQuotes Ltd.

EveryTick

「Every tick（ティックごと）」フラグを設定します。

```
void  EveryTick(
   bool    value         // フラグ
  )
```

パラメータ

value

[in] フラグの新しい値

戻り値

なし

注意事項

フラグが設定されていない場合、処理メソッドは、操作中の時系列及びシンボルの新しいバーの作成時のみに呼び出されます。

© 2000–2025, MetaQuotes Ltd.

Open

「Open(始値)」時系列の要素をインデックスで取得します。

```
double  Open(
   int     ind          // インデックス
  )
```

パラメータ

ind

[in] 要素のインデックス

戻り値

成功した場合は指定されたインデックスを持つ「Open(始値)」時系列の要素の数値、その他の場合は EMPTY_VALUE

注意事項

EMPTY_VALUE は 2 つの場合に返されます。

1. 時系列が使用されていない(対応するビットが設定されていない)
2. 要素のインデックスが範囲外

© 2000-2025, MetaQuotes Ltd.

High

「High（高値）」時系列の要素をインデックスで取得します。

```
double  High(
   int     ind          // インデックス
  )
```

パラメータ

ind

[in] 要素のインデックス

戻り値

成功した場合は指定されたインデックスを持つ「High（高値）」時系列の要素の数値、その他の場合は EMPTY_VALUE

注意事項

EMPTY_VALUE は 2 つの場合に返されます。

1. 時系列が使用されていない（対応するビットが設定されていない）
2. 要素のインデックスが範囲外

© 2000–2025, MetaQuotes Ltd.

Low

「Low（安値）」時系列の要素をインデックスで取得します。

```
double  Low(
   int     ind          // インデックス
  )
```

パラメータ

ind

[in] 要素のインデックス

戻り値

成功した場合は指定されたインデックスを持つ「Low（安値）」時系列の要素の数値、その他の場合は EMPTY_VALUE

注意事項

EMPTY_VALUE は 2 つの場合に返されます。

1. 時系列が使用されていない（対応するビットが設定されていない）
2. 要素のインデックスが範囲外

© 2000–2025, MetaQuotes Ltd.

Close

「Close（終値）」時系列の要素をインデックスで取得します。

```
double  Close(
   int     ind          // インデックス
  )
```

パラメータ

ind

[in] 要素のインデックス

戻り値

成功した場合は指定されたインデックスを持つ「Close（終値）」時系列の要素の数値、その他の場合は EMPTY_VALUE

注意事項

EMPTY_VALUE は 2 つの場合に返されます。

1. 時系列が使用されていない（対応するビットが設定されていない）
2. 要素のインデックスが範囲外

© 2000–2025, MetaQuotes Ltd.

Spread

「Spread（スプレッド）」時系列の要素をインデックスで取得します。

```
double  Spread(
   int     ind          // インデックス
  )
```

パラメータ

ind

[in] 要素のインデックス

戻り値

成功した場合は指定されたインデックスを持つ「Spread（スプレッド）」時系列の要素の数値、その他の場合は EMPTY_VALUE

注意事項

EMPTY_VALUE は 2 つの場合に返されます。

1. 時系列が使用されていない（対応するビットが設定されていない）
2. 要素のインデックスが範囲外

© 2000-2025, MetaQuotes Ltd.

Time

「Time(時刻)」時系列の要素をインデックスで取得します。

```
datetime  Time(
   int    ind          // インデックス
  )
```

パラメータ

ind

[in] 要素のインデックス

戻り値

成功した場合は指定されたインデックスを持つ「Time(時刻)」時系列の要素の数値、その他の場合は EMPTY_VALUE

注意事項

EMPTY_VALUE は 2 つの場合に返されます。

1. 時系列が使用されていない(対応するビットが設定されていない)
2. 要素のインデックスが範囲外

© 2000–2025, MetaQuotes Ltd.

TickVolume

「TickVolume（ティックボリューム）」時系列の要素をインデックスで取得します。

```
long  TickVolume(
   int    ind          // インデックス
  )
```

パラメータ

ind

[in] 要素のインデックス

戻り値

成功した場合は指定されたインデックスを持つ「TickVolume（ティックボリューム）」時系列の要素の数値、その他の場合は EMPTY_VALUE

注意事項

EMPTY_VALUE は 2 つの場合に返されます。

1. 時系列が使用されていない（対応するビットが設定されていない）
2. 要素のインデックスが範囲外

© 2000–2025, MetaQuotes Ltd.

RealVolume

「RealVolume（実のボリューム）」時系列の要素をインデックスで取得します。

```
long  RealVolume(
   int    ind          // インデックス
  )
```

パラメータ

ind

[in] 要素のインデックス

戻り値

成功した場合は指定されたインデックスを持つ「RealVolume（実のボリューム）」時系列の要素の数値、その他の場合は EMPTY_VALUE

注意事項

EMPTY_VALUE は 2 つの場合に返されます。

1. 時系列が使用されていない（対応するビットが設定されていない）
2. 要素のインデックスが範囲外

© 2000–2025, MetaQuotes Ltd.

Init

オブジェクトを初期化します。

```
bool  Init(
   CSymbolInfo       symbol,    // シンボル
   ENUM_TIMEFRAMES   period,    // 時間軸
   double            point      // ポイント
  )
```

パラメータ

symbol

[in] シンボル情報へのアクセスに使用されるCSymbolInfo 型のオブジェクトへのポインタ

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

point

[in] 2/4桁ポイントの「重み」

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Symbol

シンボルを設定します。

```
bool  Symbol(
   string    name      // シンボル
  )
```

パラメータ

name

[in] シンボル

戻り値

成功の場合は true、それ以外の場合は false

注意事項

オブジェクトが初期化時に定義されたのとは異なるシンボルを使用する場合、そのシンボルの設定が必要です。

© 2000–2025, MetaQuotes Ltd.

Period

時間軸を設定します。

```
bool  Period(
   ENUM_TIMEFRAMES  value     // 時間軸
  )
```

パラメータ

value

[in] 時間軸

戻り値

成功の場合は true、それ以外の場合は false

注意事項

オブジェクトが初期化時に定義されたのとは異なる時間軸を使用する場合、その時間軸の設定が必要です。

© 2000–2025, MetaQuotes Ltd.

Magic

エキスパート アド バイザー IDを設 定します。

```
void  Magic(
   ulong  value     // マジック
  )
```

パラメータ

value

[in] エキスパート アド バイザー ID

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

ValidationSettings

設定をチェックします。

```
virtual bool  ValidationSettings()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

SetPriceSeries

外部の価格列へのポインタを設定します。

```
virtual bool  SetPriceSeries(
   CiOpen*     open,         // ポインタ
   CiHigh*     high,         // ポインタ
   CiLow*      low,          // ポインタ
   CiClose*    close         // ポインタ
  )
```

パラメータ

open

[in] Open 時系列へのポインタ

high

[in] High 時系列へのポインタ

low

[in] Low 時系列へのポインタ

close

[in] Close 時系列へのポインタ

戻り値

成功の場合は true、それ以外の場合は false

注意事項

オブジェクトが初期化時に定義されたのとは異なるシンボルまたは時間軸の時系列を使用する場合には、その外部時系列（価格列）へのポインタの設定が必要です。

© 2000–2025, MetaQuotes Ltd.

SetOtherSeries

外部の価格列以外の列へのポインタを設定します。

```
virtual bool  SetOtherSeries(
   CiSpread*       spread,        // ポインタ
   CiTime*         time,          // ポインタ
   CiTickVolume*   tick_volume,   // ポインタ
   CiRealVolume*   real_volume    // ポインタ
  )
```

パラメータ

spread

[in] Spread 時系列へのポインタ

time

[in] Time 時系列へのポインタ

tick_volume

[in] TickVolume 時系列へのポインタ

real_volume

[in] RealVolume 時系列へのポインタ

戻り値

成功の場合は true、それ以外の場合は false

注意事項

オブジェクトが初期化時に定義されたのとは異なるシンボルまたは時間軸の時系列を使用する場合には、その外部時系列（価格列以外）へのポインタの設定が必要です。

© 2000–2025, MetaQuotes Ltd.

InitIndicators

全ての指標と時系列を初期化解除します。

```
virtual bool  InitIndicators(
   CIndicators*  indicators=NULL    // ポインタ
  )
```

パラメータ

指標

[in] 指標と時系列の集合へのポインタ

戻り値

成功の場合は true、それ以外の場合は false

注意事項

時系列は、オブジェクトが、初期化時に定義されたのとは異なるシンボルまたは時間軸を使用する場合にのみ初期化されます。

© 2000–2025, MetaQuotes Ltd.

InitOpen

「Open(始値)」時系列を初期化します。

```
bool  InitOpen(
   CIndicators*  indicators    // ポインタ
  )
```

パラメータ

指標

[in] 指標と時系列の集合へのポインタ

戻り値

成功の場合は true、それ以外の場合は false

注意事項

「Open(始値)」時系列は、エキスパートアドバイザーが、初期化時に定義されたのとは異なるシンボルまたは時間軸を使用する場合にのみ初期化され、使用されます。

© 2000-2025, MetaQuotes Ltd.

InitHigh

「High(終値)」時系列を初期化します。

```
bool  InitHigh(
   CIndicators*  indicators    // ポインタ
  )
```

パラメータ

指標

[in] 指標と時系列の集合へのポインタ

戻り値

成功の場合は true、それ以外の場合は false

注意事項

「High(終値)」時系列は、エキスパートアドバイザーが、初期化時に定義されたのとは異なるシンボルまたは時間軸を使用する場合にのみ初期化され、使用されます。

© 2000-2025, MetaQuotes Ltd.

InitLow

「Low（安値）」時系列を初期化します。

```
bool  InitLow(
   CIndicators*  indicators    // ポインタ
  )
```

パラメータ

指標

[in] 指標と時系列の集合へのポインタ

戻り値

成功の場合は true、それ以外の場合は false

注意事項

「Low（安値）」時系列は、エキスパートアドバイザーが、初期化時に定義されたのとは異なるシンボルまたは時間軸を使用する場合にのみ初期化され、使用されます。

© 2000–2025, MetaQuotes Ltd.

InitClose

「Close(終値)」時系列を初期化します。

```
bool  InitClose(
   CIndicators*  indicators    // ポインタ
  )
```

パラメータ

指標

[in] 指標と時系列の集合へのポインタ

戻り値

成功の場合は true、それ以外の場合は false

注意事項

「Close(終値)」時系列は、エキスパートアドバイザーが、初期化時に定義されたのとは異なるシンボルまたは時間軸を使用する場合にのみ初期化され、使用されます。

© 2000-2025, MetaQuotes Ltd.

InitSpread

「Spread（スプレッド）」時系列を初期化します。

```
bool  InitSpread(
   CIndicators*  indicators    // ポインタ
  )
```

パラメータ

指標

[in] 指標と時系列の集合へのポインタ

戻り値

成功の場合は true、それ以外の場合は false

注意事項

「Spread（スプレッド）」時系列は、エキスパートアドバイザーが、初期化時に定義されたのとは異なるシンボルまたは時間軸を使用する場合にのみ初期化され、使用されます。

© 2000-2025, MetaQuotes Ltd.

InitTime

「Time(時刻)」時系列を初期化します。

```
bool  InitTime(
   CIndicators*  indicators    // ポインタ
  )
```

パラメータ

指標

[in] 指標と時系列の集合へのポインタ

戻り値

成功の場合は true、それ以外の場合は false

注意事項

「Time(時刻)」時系列は、エキスパートアドバイザーが、初期化時に定義されたのとは異なるシンボルまたは時間軸を使用する場合にのみ初期化され、使用されます。

© 2000–2025, MetaQuotes Ltd.

InitTickVolume

「TickVolume（ティックボリューム）」時系列を初期化します。

```
bool  InitTickVolume(
   CIndicators*  indicators    // ポインタ
  )
```

パラメータ

指標

[in] 指標と時系列の集合へのポインタ

戻り値

成功の場合は true、それ以外の場合は false

注意事項

「TickVolume（ティックボリューム）」 時系列は、エキスパートアドバイザーが、初期化時に定義されたのとは異なるシンボルまたは時間軸を使用する場合にのみ初期化され、使用されます。

© 2000–2025, MetaQuotes Ltd.

InitRealVolume

「RealVolume（実のボリューム）」時系列を初期化します。

```
bool  InitRealVolume(
   CIndicators*  indicators    // ポインタ
  )
```

パラメータ

指標

[in] 指標と時系列の集合へのポインタ

戻り値

成功の場合は true、それ以外の場合は false

注意事項

「RealVolume（実のボリューム）」 時系列は、エキスパートアドバイザーが、初期化時に定義されたのとは異なるシンボルまたは時間軸を使用する場合にのみ初期化され、使用されます。

© 2000–2025, MetaQuotes Ltd.

PriceLevelUnit

価格レベル単位を取得します。

```
virtual double  PriceLevelUnit()
```

戻り値

価格レベル単位

注意事項

基本クラスのメソッドは2/4桁ポイントの「重み」を返します。

© 2000–2025, MetaQuotes Ltd.

StartIndex

分析される最初のバーのインデックスを取得します。

```
virtual int  StartIndex()
```

戻り値

分析される最初のバーのインデックス

注意事項

このメソッドは、現在足を分析するフラグがtrueの場合は（分析が現在足で始まる）0を返します。もしこのフラグが設定されていない場合は、1が返されます（分析が最後の完成したバーで始まる）。

© 2000–2025, MetaQuotes Ltd.

CompareMagic

要素のインデックスが範囲外

```
virtual bool  CompareMagic(
   ulong  magic    // 比較に使用される値
  )
```

パラメータ

magic

[in] 比較に使用される値

戻り値

同一の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CExpert

CExpert は取引ストラテジーの基本クラスです。このクラスには、時系列や指標操作のアルゴリズムと、取引ストラテジーのための仮想メソッドのセットが内蔵されています。

使用法

1ストラテジーのアルゴリズムを準備します。
2. CExpert クラスから継承した自分自身のクラスを作成します。
3. 仮想メソッドを自分自身のアルゴリズムでオーバーライドします。

説明

CExpert クラスは取引ストラテジーを実装するための仮想メソッドのセットです。

宣言

```
class CExpert : public CExpertBase
```

タイトル

```
#include <Expert\Expert.mqh>
```

継承階層

CObject
CExpertBase
CExpert

クラスメソッド

初期化	
Init	クラスインスタンス初期化メソッド
virtual InitSignal	売買シグナルオブジェクトの初期化
virtual InitTrailing	トレール注文オブジェクトの初期化
virtual InitMoney	資金管理オブジェクトの初期化
virtual InitTrade	取引オブジェクトの初期化
virtual ValidationSettings	設定のチェック
virtual InitIndicators	指標と時系列の初期化
virtual InitParameters	パラメータ初期化メソッド
virtual Deinit	クラスインスタンス初期化解除メソッド
virtual DeinitSignal	売買シグナルオブジェクトの初期化解除
virtual DeinitTrailing	トレール注文オブジェクトの初期化解除
virtual DeinitMoney	資金管理オブジェクトの初期化解除

© 2000–2025, MetaQuotes Ltd.

初期化	
virtual DeinitTrade	取引オブジェクトの初期化解除
virtual DeinitIndicators	指標と時系列の初期化解除
パラメータ	
Magic	エキスパートアドバイザーIDの設定
MaxOrders	注文最高額の取得/設定
OnTickProcess	「OnTick」イベント続行のフラグの設定
OnTradeProcess	「OnTrade」イベント続行のフラグの設定
OnTimerProcess	「OnTimer」イベント続行のフラグの設定
OnChartEventProcess	「OnChartEvent」イベント続行のフラグの設定
OnBookEventProcess	「OnBookEvent」イベント続行のフラグの設定
イベント処理メソッド	
OnTick	OnTick イベントハンドラ
OnTrade	OnTrade イベントハンドラ
OnTimer	OnTimer イベントハンドラ
OnChartEvent	OnChartEvent イベントハンドラ
OnBookEvent	OnBookEvent イベントハンドラ
更新メソッド	
Refresh	全てのデータの更新
Processing	
処理	主要な処理アルゴリズム
マーケットエントリーメソッド	
CheckOpen	注文を出す条件のチェック
CheckOpenLong	ロング(買い)注文を出す条件のチェック
CheckOpenShort	ショート(売り)注文を出す条件のチェック
OpenLong	ロング(買い)注文を出す
OpenShort	ショート(売り)注文を出す
マーケットイグジットメソッド	
CheckClose	現在のポジションを決済する条件のチェック
CheckCloseLong	ロング(買い)ポジションを決済する条件のチェック
CheckCloseShort	ショート(売り)ポジションを決済する条件のチェック

© 2000–2025, MetaQuotes Ltd.

初期化	
CloseAll	未決済ポジションの決済と全ての注文の削除
Close	未決済ポジションの決済
CloseLong	ロング(買い)ポジションの決済
CloseShort	ショート(売り)ポジションの決済
ポジション反転のメソッド	
CheckReverse	未決済ポジションを反転する条件のチェック
CheckReverseLong	ロング(買い)ポジションを反転する条件のチェック
CheckReverseShort	ショート(売り)ポジションを反転する条件のチェック
ReverseLong	ロング(買い)ポジションの反転操作
ReverseShort	ショート(売り)ポジションの反転操作
ポジション/注文トレーリングメソッド	
CheckTrailingStop	ポジションパラメータ変更の条件のチェック
CheckTrailingStopLong	ロング(買い)ポジションのトレール注文条件のチェック
CheckTrailingStopShort	ショート(売り)ポジションのトレール注文条件のチェック
TrailingStopLong	ロング(買い)ポジションのトレール注文
TrailingStopShort	ショート(売り)ポジションのトレール注文
CheckTrailingOrderLong	買い逆指値/指値注文のトレール注文条件のチェック
CheckTrailingOrderShort	売り逆指値/指値注文のトレール注文条件のチェック
TrailingOrderLong	買い逆指値/指値注文のトレール注文
TrailingOrderShort	売り逆指値/指値注文のトレール注文
注文削除メソッド	
CheckDeleteOrderLong	買い未決注文削除の条件のチェック
CheckDeleteOrderShort	売り未決注文削除の条件のチェック
DeleteOrders	全ての注文の削除
DeleteOrder	逆指値/指値注文の削除
DeleteOrderLong	買い逆指値/指値注文の削除
DeleteOrderShort	売り逆指値/指値注文の削除
取引高メソッド	
LotOpenLong	買いの取引高の取得
LotOpenShort	売りの取引高の取得

© 2000–2025, MetaQuotes Ltd.

初期化	
LotReverse	ポジション反転の取引高の取得
取引履歴メソッド	
PrepareHistoryDate	取引履歴追跡の開始日の設定
HistoryPoint	取引履歴のチェックポイントの作成（ポジション数、注文、注文、と履歴注文の保存）
CheckTradeState	保存された状態と現在の状態の比較と対応するイベントハンドラの呼び出し
イベントフラグ	
WaitEvent	イベント待機フラグの設定
NoWaitEvent	イベント待機フラグのリセット
取引イベント処理メソッド	
TradeEventPositionStopTake	「決済逆指及び決済指トリガ」イベントハンドラ
TradeEventOrderTriggered	「未決注文トリガ」イベントハンドラ
TradeEventPositionOpened	「注文を出す」イベントハンドラ
TradeEventPositionVolumeChanged	「ポジションボリューム変更」イベントハンドラ
TradeEventPositionModified	「ポジション変更」イベントハンドラ
TradeEventPositionClosed	「ポジション決済」イベントハンドラ
TradeEventOrderPlaced	「未決注文を出す」イベントハンドラ
TradeEventOrderModified	「未決注文変更」イベントハンドラ
TradeEventOrderDeleted	「未決注文削除」イベントハンドラ
TradeEventNotIdentified	未知イベントのハンドラ
サービスメソッド	
TimeframeAdd	時間軸の追跡への追加
TimeframesFlags	新しいバーを持つ時間軸を示すフラグの取得
SelectPosition	Selects a position to work with

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

クラスから継承されたメソッド CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries

© 2000–2025, MetaQuotes Ltd.

Init

クラスインスタンスの初期化メソッドです。

```
bool  Init(
   string              symbol,         // シンボル
   ENUM_TIMEFRAMES     period,         // 時間軸
   bool                every_tick,     // フラグ
   ulong               magic           // マジック
  )
```

パラメータ

symbol

[in] シンボル

period

[in] 時間軸（ENUM_TIMEFRAMES 列挙）

every_tick

[in] フラグ

magic

[in] エキスパートアドバイザー ID（マジックナンバー）

戻り値

成功の場合は true、それ以外の場合は false

注意事項

every_tick が true の場合は、Processing() メソッドが操作中のシンボルのティックごとに呼び出されます。
その他の場合は Processing() は操作中のシンボルの新しいバーの作成時のみに呼び出されます。

© 2000–2025, MetaQuotes Ltd.

Magic

エキスパート アド バイザー ID（マジック）を設 定 します。

```
void  Magic(
   ulong  value     // 新しい値
  )
```

パラメータ

value

[in] エキスパート アド バイザー IDの新 しい値

戻 り値

なし

注 意 事 項

エキスパート アド バイザー ID（マジック）を取 引 、シグナル、トレーリングの各 クラスに設 定 します。

実 装

```
//+------------------------------------------------------------------+
//| オブジェクトと依存オブジェクトのマジックナンバーを設定する              |
//| 入力: value - マジックナンバーの新しい値                            |
//| 出力:なし                                                         |
//| メモ:なし                                                         |
//+------------------------------------------------------------------+
void CExpert::Magic(ulong value)
  {
   if(m_trade!=NULL)     m_trade.SetExpertMagicNumber(value);
   if(m_signal!=NULL)    m_signal.Magic(value);
   if(m_money!=NULL)     m_money.Magic(value);
   if(m_trailing!=NULL) m_trailing.Magic(value);
//---
   CExpertBase::Magic(value);
  }
```

© 2000–2025, MetaQuotes Ltd.

InitSignal

売買シグナルオブジェクトを初期化します。

```
virtual bool  InitSignal(
   CExpertSignal*      signal=NULL,         // ポインタ
  )
```

パラメータ

signal

[in] CExpertSignal クラスオブジェクトまたは派生オブジェクトへのポインタ

戻り値

成功の場合は true、それ以外の場合は false

注意事項

signal が NULL の場合は、CExpertSignal クラスが使用され何の操作も発生しません。

© 2000–2025, MetaQuotes Ltd.

InitTrailing

トレール注文オブジェクトを初期化します。

```
virtual bool  InitTrailing(
   CExpertTrailing*     trailing=NULL,         // ポインタ
  )
```

パラメータ

trailing

[in] CExpertTrailing クラスオブジェクトまたは派生オブジェクトへのポインタ

戻り値

成功の場合は true、それ以外の場合は false

注意事項

trailing が NULL の場合は、ExpertTrailing クラスが使用され何の操作も発生しません。

© 2000–2025, MetaQuotes Ltd.

InitMoney

資金管理オブジェクトの初期化です。

```
virtual bool  InitMoney(
   CExpertMoney*      money=NULL,         // ポインタ
  )
```

パラメータ

money

[in] CExpertMoney クラスオブジェクトまたは派生オブジェクトへのポインタ

戻り値

成功の場合は true、それ以外の場合は false

注意事項

money が NULL の場合は、CExpertMoney クラスが使用され、最小ロットが使用されます。

© 2000–2025, MetaQuotes Ltd.

InitTrade

取引オブジェクトを初期化します。

```
virtual bool  InitTrade(
   ulong             magic,        // マジック
   CExpertTrade*     trade=NULL    // ポインタ
  )
```

パラメータ

magic

[in] エキスパートアドバイザーID（取引リクエストに使用されます）

trade

[in] CExpertTrade オブジェクトへのポインタ

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Deinit

クラスインスタンスの初期化解除メソッドです。

```
virtual void Deinit()
```

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

OnTickProcess

OnTick イベント続行のフラグを設定します。

```
void  OnTickOProcess(
   bool     value        // フラグ
  )
```

パラメータ

value

[in] OnTick イベント続行のフラグ

戻り値

なし

注意事項

フラグが true の場合は、OnTick イベントが続行され、デフォルトでは、フラグが true に設定されます。

© 2000–2025, MetaQuotes Ltd.

OnTradeProcess

OnTrade イベント続行のフラグを設定します。

```
void  OnTradeProcess(
   bool     value        // フラグ
  )
```

パラメータ

value

[in] OnTrade イベント続行のフラグ

戻り値

なし

注意事項

フラグが true の場合は、OnTrade イベントが続行され、デフォルトでは、フラグが false に設定されます。

© 2000–2025, MetaQuotes Ltd.

OnTimerProcess

OnTimer イベント続行のフラグを設定します。

```
void  OnTimerProcess(
   bool     value        // フラグ
  )
```

パラメータ

value

[in] OnTimer イベント続行のフラグ

戻り値

なし

注意事項

フラグが true の場合は、OnTimer イベントが続行され、デフォルトでは、フラグが false に設定されます。

© 2000–2025, MetaQuotes Ltd.

OnChartEventProcess

OnChartEvent イベント続行のフラグを設定します。.

```
void  OnChartEventProcess(
   bool     value        // フラグ
  )
```

パラメータ

value

[in] OnChartEvent イベント続行のフラグ

戻り値

なし

注意事項

フラグが true の場合は、OnChartEvent イベントが続行され、デフォルトでは、フラグが false に設定されます。

© 2000–2025, MetaQuotes Ltd.

OnBookEventProcess

OnBookEvent イベント続行のフラグを設定します。

```
void  OnChartEventProcess(
   bool     value        // フラグ
  )
```

パラメータ

value

[in] OnBookEvent イベント続行のフラグ

戻り値

なし

注意事項

フラグが true の場合は、OnBookEvent イベントが続行され、デフォルトでは、フラグが false に設定されます。

© 2000–2025, MetaQuotes Ltd.

MaxOrders（Get メソッド）

許可された注文最大量を取得します。

```
int  MaxOrders()
```

戻り値

許可された注文最大量

MaxOrders（Set メソッド）

許可された注文最大量を設定します。

```
void  MaxOrders(
   int      max_orders         // 新しい値
  )
```

パラメータ

max_orders

[in] 許可された注文最大量の新しい値

戻り値

なし

注意事項

デフォルトでは、許可された注文最大量は 1 です。

© 2000–2025, MetaQuotes Ltd.

Signal

売買シグナルオブジェクトへのポインタを取得します。

```
CExpertSignal*  Signal() const
```

戻り値

売買シグナルオブジェクトへのポインタ

© 2000–2025, MetaQuotes Ltd.

ValidationSettings

設定をチェックします。

```
virtual bool  ValidationSettings()
```

戻り値

成功の場合は true、それ以外の場合は false

注意事項

エキスパートアドバイザーオブジェクトの設定をチェックします。

© 2000–2025, MetaQuotes Ltd.

InitIndicators

全ての指標と時系列を初期化します。

```
virtual bool  InitIndicators(
   CIndicators*  indicators=NULL    // ポインタ
  )
```

パラメータ

指標

[in] 指標と時系列の集合へのポインタ

戻り値

成功の場合は true、それ以外の場合は false

注意事項

オブジェクトが初期化時に定義されたのとは異なるシンボルまたは時間枠を使用する場合、時系列が初期化されます。売買シグナルオブジェクトと資金管理オブジェクトの InitIndicators() 仮想メソッドが順番に呼び出されます。

© 2000–2025, MetaQuotes Ltd.

OnTick

OnTick イベントのハンドラです。

```
virtual void  OnTick()
```

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

OnTrade

OnTrade イベントのハンドラです。

```
virtual void  OnTrade()
```

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

OnTimer

OnTimer イベントのハンドラです。

```
virtual void  OnTimer()
```

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

OnChartEvent

OnChartEvent イベントのハンドラです。

```
virtual void  OnChartEvent(
   const int      id,        // Event ID
   const long&    lparam,    // long パラメータ
   const double   dparam,    // double パラメータ
   const string   sparam     // string パラメータ
  )
```

パラメータ

id

[in] イベント識別子

lparam

[in] long 型のイベントパラメータ

dparam

[in] double 型のイベントパラメータ

sparam

[in] string 型のイベントパラメータ

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

OnBookEvent

OnBookEvent イベントのハンドラです。

```
virtual void  OnBookEvent(
   const string&     symbol        // シンボル
)
```

パラメータ

symbol

[in] OnBookEvent イベントのシンボル

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

InitParameters

エキスパートアドバイザーのパラメータを初期化します。

```
virtual bool  InitParameters()
```

戻り値

成功の場合は true、それ以外の場合は false

注意事項

CExpert 基本クラスの InitParameters() は何の操作も行わずに常に true を返します。

© 2000–2025, MetaQuotes Ltd.

DeinitTrade

取引オブジェクトを初期化解除します。

```
virtual void  DeinitTrade()
```

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

DeinitSignal

シグナルオブジェクトを初期化解除します。

```
virtual void  DeinitSignal()
```

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

DeinitTrailing

トレーリングオブジェクトを初期化解除します。

```
virtual void  DeinitTrailing()
```

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

DeinitMoney

資金管理オブジェクトを初期化解除します。

```
virtual void  DeinitMoney()
```

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

DeinitIndicators

全ての指標と時系列を初期化解除します。

```
virtual void  DeinitIndicators()
```

戻り値

なし

注意事項

補助オブジェクトの全ての指標と時系列も初期化解除されます。

© 2000–2025, MetaQuotes Ltd.

Refresh

全てのデータを更新します。

```
virtual bool  Refresh()
```

戻り値

さらなるティック処理が必要な場合は true、それ以外の場合は false

注意事項

ティック処理の必要性を決定します。必要な場合は、全ての相場と時系列と指標データを更新し、true を返します。

実装

```
//+------------------------------------------------------------------+
//|  処理のためにデータをリフレッシュするg                                  |
//|  入力:なし                                                        |
//|  出力:成功の場合は true、それ以外の場合は false                        |
//|  メモ:なし                                                        |
//+------------------------------------------------------------------+
bool CExpert::Refresh()
  {
   MqlDateTime time;
//--- レートリフレッシュをする
   if(!m_symbol.RefreshRates()) return(false);
//--- チェックの処理が必要
   TimeToStruct(m_symbol.Time(),time);
   if(m_period_flags!=WRONG_VALUE && m_period_flags!=0)
      if((m_period_flags & TimeframesFlags(time))==0) return(false);
   m_last_tick_time=time;
//--- 指標をフレッシュする
   m_indicators.Refresh();
//--- ok
   return(true);
  }
```

© 2000–2025, MetaQuotes Ltd.

Processing

主要な処理アルゴリズムです。

```
virtual bool  Processing()
```

戻り値

取引操作が実行された場合は true、それ以外の場合は false

注意事項

下記のステップが実行されます。

1シンボルの未決済ポジションの存在を確認します。未決済ポジションがない場合、ステップ 2～4 は抜かされます。
2. ポジション反転の条件をチェックします（CheckReverse() メソッド）。ポジションが反転された場合、終了します。
3. ポジション決済の条件をチェックします（CheckClose() メソッド）。ポジションが決済された場合、ステップ 4 が抜かされます。
4. ポジションパラメータ変更の条件をチェックします（CheckTrailingStop() メソッド）。ポジションパラメータが変更された場合、終了します。
5. シンボルの未決注文の存在を確認します。未決注文がない場合、ステップ 9 に行きます。
6. 注文削除の条件のチェックします（買い未決注文と売り未決注文に CheckDeleteOrderLong() と CheckDeleteOrderShort() のそれぞれが使用されます）。注文が削除された場合、ステップ 9 に行きます。
7. 未決注文パラメータ変更の条件をチェックします（買い注文と売り注文にCheckTrailingOrderLong() と CheckTrailingOrderShort() のそれぞれが使用されます）。パラメータが変更された場合、終了します。
8. 終了します。
9. 注文を出す条件をチェックします（CheckOpen() メソッド）。

自分独自のアルゴリズムを実装したい場合は、派生クラスの Processing() メソッドをオーバーライドする必要があります。

実装

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| メイン関数                                                       |
//| 入力:なし                                                        |
//| 出力:取引操作が処理された場合は true、それ以外の場合は false |
//| メモ:なし                                                        |
//+------------------------------------------------------------------+
bool CExpert::Processing()
  {
//--- ポジションかどうかをチェック
   if(m_position.Select(m_symbol.Name()))
     {
      //--- ポジションが利用可能
      //--- ポジション反転の可能性を見る
      if(CheckReverse()) return(true);
      //--- ポジション決済か未決注文削除の可能性を見る
      if(!CheckClose())
        {
         //--- ポジション変更の可能性を見る
         if(CheckTrailingStop()) return(true);
         //--- 操作なしで返る
         return(false);
        }
     }
//--- 未決注文が出されたかをチェック
   int total=OrdersTotal();
   if(total!=0)
     {
      for(int i=total-1;i>=0;i--)
        {
         m_order.SelectByIndex(i);
         if(m_order.Symbol()!=m_symbol.Name()) continue;
         if(m_order.OrderType()==ORDER_TYPE_BUY_LIMIT || m_order.OrderType()
==ORDER_TYPE_BUY_STOP)
           {
            //--- 未決買い注文を削除できるかをチェック
            if(CheckDeleteOrderLong()) return(true);
            //--- 未決買い注文を変更できるかをチェック
            if(CheckTrailingOrderLong()) return(true);
           }
         else
           {
            //--- 未決売り注文を削除できるかをチェック
            if(CheckDeleteOrderShort()) return(true);
            //--- 未決売り注文を変更できるかをチェック
            if(CheckTrailingOrderShort()) return(true);
           }
         //--- 操作なしで返る
         return(false);
        }
     }
//--- 注文を出すか未決注文を設定する可能性をチェックする
   if(CheckOpen()) return(true);
//--- 操作なしで返る
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

SelectPosition

Selects a position to work with.

```
void  SelectPosition()
```

Return Value

No.

Implementation

```
//+------------------------------------------------------------------+
//| Position select                                                  |
//| INPUT:  no.                                                      |
//| OUTPUT: no.                                                      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::SelectPosition(void)
  {
   bool res=false;
//---
   if(m_margin_mode==ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)
      res=m_position.SelectByMagic(m_symbol.Name(),m_magic);
   else
      res=m_position.Select(m_symbol.Name());
//---
   return(res);
  }
```

© 2000–2025, MetaQuotes Ltd.

CheckOpen

注文を出す条件をチェックします。

```
virtual bool  CheckOpen()
```

戻り値

取引操作が実行された場合はtrue、それ以外の場合はfalse

注意事項

ロング（CheckOpenLong()）及びショート（CheckOpenShort()）注文を出す条件をチェックします。

実装

```
//+------------------------------------------------------------------+
//|  注文または逆指値/指値注文確定をチェックする                      |
//|  入力:なし                                                       |
//|  出力:取引操作が処理された場合はtrue、それ以外の場合はfalse      |
//|  メモ:なし                                                       |
//+------------------------------------------------------------------+
bool CExpert::CheckOpen()
  {
   if(CheckOpenLong())  return(true);
   if(CheckOpenShort()) return(true);
//--- 操作なしで返る
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckOpenLong

ロング（買い）注文を出す条件をチェックします。

```
virtual bool  CheckOpenLong()
```

戻り値

取引操作が実行された場合は true、それ以外の場合は false

注意事項

ロング（買い）注文を出す条件をチェックし（Signal オブジェクトの CheckOpenLong() メソッド）、必要な場合はポジションを取ります（OpenLong() メソッド）。

実装

```
//+------------------------------------------------------------------+
//|  買い注文または逆指値/指値注文確定をチェックする                  |
//|  入力:なし                                                       |
//|  出力:取引操作が処理された場合は true、それ以外の場合は false    |
//|  メモ:なし                                                       |
//+------------------------------------------------------------------+
bool CExpert::CheckOpenLong()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- check signal for long enter operations
   if(m_signal.CheckOpenLong(price,sl,tp,expiration))
     {
      if(!m_trade.SetOrderExpiration(expiration))
        {
         m_expiration=expiration;
        }
      return(OpenLong(price,sl,tp));
     }
//--- 操作なしで返る
   return(false);
  }
```

© 2000–2025, MetaQuotes Ltd.

CheckOpenShort

ショート（売り）注文を出す条件をチェックします。

```
virtual bool  CheckOpenShort()
```

戻り値

取引操作が実行された場合は true、それ以外の場合は false

注意事項

ショート（売り）注文を出す条件をチェックし（Signal オブジェクトの CheckOpenShort() メソッド）、必要な場合はポジションを取ります（OpenShort() メソッド）。

実装

```
//+------------------------------------------------------------------+
//| 売り注文または逆指値/指値注文確定をチェックする                  |
//| 入力:なし                                                        |
//| 出力:取引操作が処理された場合は true、それ以外の場合は false    |
//| メモ:なし                                                        |
//+------------------------------------------------------------------+
bool CExpert::CheckOpenShort()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- ショートエンター操作をチェックする
   if(m_signal.CheckOpenShort(price,sl,tp,expiration))
     {
      if(!m_trade.SetOrderExpiration(expiration))
        {
         m_expiration=expiration;
        }
      return(OpenShort(price,sl,tp));
     }
//--- 操作なしで返る
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

OpenLong

ロング（買い）ポジションを取ります。

```
virtual bool  OpenLong(
   double    price,    // 価格
   double    sl,       // 決済逆指値
   double    tp        // 決済指値
  )
```

パラメータ

価格

[in] 価格.

sl

[in] 決済逆指値

tp

[in] 決済指値

戻り値

取引操作が実行された場合はtrue、それ以外の場合はfalse

注意事項

取引高を取得し（LotOpenLong(...) メソッド）、ボリュームが 0 でない場合は、ロング（買い）ポジションが取られます（取引オブジェクトの Buy() メソッド）。

実装

```
//+------------------------------------------------------------------+
//| ロング注文またはリミット/ストップ注文確定をチェックする                |
//| 入力:  price - 価格                                               |
//|          sl    - 決済逆指値                                       |
//|          tp    - 決済指値                                         |
//| 出力:取引操作が処理された場合は true、 それ以外の場合は false      |
//| メモ:なし                                                         |
//+------------------------------------------------------------------+
bool CExpert::OpenLong(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE) return(false);
//--- 未決済ロットの取得
   double lot=LotOpenLong(price,sl);
//--- 未決済ロットのチェック
   if(lot==0.0) return(false);
//---
   return(m_trade.Buy(lot,price,sl,tp));
  }
```

© 2000-2025, MetaQuotes Ltd.

OpenShort

ショート（売り）ポジションを取ります。

```
virtual bool  OpenShort(
   double    price,     // 価格
   double    sl,        // 決済逆指値
   double    tp         // 決済指値
  )
```

パラメータ

価格

[in] 価格.

sl

[in] 決済逆指値

tp

[in] 決済指値

戻り値

取引操作が実行された場合は true、それ以外の場合は false

注意事項

取引高を取得し（LotOpenShort() メソッド）、ボリュームが 0 でない場合は、ショート（売り）ポジションが取られます（取引オブジェクトの Sell() メソッド）。

実装

```
//+------------------------------------------------------------------+
//| ショート注文またはリミット/ストップ注文確定をチェックする                |
//| 入力:price - 価格                                                |
//|          sl     - 決済逆指値,                                    |
//|          tp     - 決済指値                                       |
//| 出力:取引操作が処理された場合は true、それ以外の場合は false     |
//| メモ:なし                                                        |
//+------------------------------------------------------------------+
bool CExpert::OpenShort(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE) return(false);
//--- 未決済ロットの取得
   double lot=LotOpenShort(price,sl);
//--- 未決済ロットのチェック
   if(lot==0.0) return(false);
//---
   return(m_trade.Sell(lot,price,sl,tp));
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckReverse

未決済ポジションを反転する条件をチェックします。

```
virtual bool  CheckReverse()
```

戻り値

取引操作が実行された場合は true、それ以外の場合は false

注意事項

ロング（CheckReverseLong()）及びショート（CheckReverseShort()）ポジションを反転する条件をチェックします。

実装

```
//+------------------------------------------------------------------+
//| ポジション反転をチェックする                                     |
//| 入力:なし                                                        |
//| 出力:取引操作が処理された場合は true、それ以外の場合は false    |
//| メモ:なし                                                        |
//+------------------------------------------------------------------+
bool CExpert::CheckReverse()
  {
   if(m_position.PositionType()==POSITION_TYPE_BUY)
     {
      //--- ロングポジションの反転をチェックする
      if(CheckReverseLong())  return(true);
     }
   else
      //--- ショートポジションの反転をチェックする
      if(CheckReverseShort()) return(true);
//--- 操作なしで返る
   return(false);
  }
```

© 2000–2025, MetaQuotes Ltd.

CheckReverseLong

ロング（買い）ポジションを反転する条件をチェックします。

```
virtual bool  CheckReverseLong()
```

戻り値

取引操作が実行された場合は true、それ以外の場合は false

注意事項

ロング（買い）ポジションを反転する条件をチェックし（Signal オブジェクトの CheckReverseLong() メソッド）、必要な場合は反転操作をします。（ReverseLong(...) メソッド）。

実装

```
//+------------------------------------------------------------------+
//| ロングポジション反転をチェックする                                |
//| 入力:なし                                                        |
//| 出力:取引操作が処理された場合は true、それ以外の場合は false     |
//| メモ:なし                                                        |
//+------------------------------------------------------------------+
bool CExpert::CheckReverseLong()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- ロングポジションの反転をチェックする
   if(m_signal.CheckReverseLong(price,sl,tp,expiration))
return(ReverseLong(price,sl,tp));
//--- 操作なしで返る
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckReverseShort

ショート（売り）ポジションを反転する条件をチェックします。

```
virtual bool  CheckReverseLong()
```

戻り値

取引操作が実行された場合はtrue、それ以外の場合はfalse

注意事項

ショート（売り）ポジションを反転する条件をチェックし（Signal オブジェクトの CheckReverseShort() メソッド）、必要な場合は反転操作をします。（ReverseShort() メソッド）。

実装

```
//+------------------------------------------------------------------+
//| ショートポジション反転をチェックする                                |
//| 入力:なし                                                        |
//| 出力:取引操作が処理された場合は true、それ以外の場合は false      |
//| メモ:なし                                                        |
//+------------------------------------------------------------------+
bool CExpert::CheckReverseShort()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- ショートポジションの反転をチェックする
   if(m_signal.CheckReverseShort(price,sl,tp,expiration))
return(ReverseShort(price,sl,tp));
//--- 操作なしで返る
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

ReverseLong

ロング（買い）ポジションの反転操作を行います。

```
virtual bool  ReverseLong(
   double    price,    // 価格
   double    sl,       // 決済逆指値
   double    tp        // 決済指値
  )
```

パラメータ

価格

[in] 価格

sl

[in] 決済逆指値

tp

[in] 決済指値

戻り値

取引操作が実行された場合はtrue、それ以外の場合はfalse

注意事項

ポジション反転ボリュームを取得し（LotReverse() メソッド）、取引高が0でない場合は、ロング（買い）ポジション反転の取引操作を行います（取引オブジェクトのSell() メソッド）。

実装

```
//+------------------------------------------------------------------+
//| ロングポジションの反転                                           |
//| 入力:price - 価格                                                |
//|          sl    - 決済逆指値                                      |
//|          tp    - 決済指値                                        |
//| 出力:取引操作が処理された場合はtrue、それ以外の場合はfalse       |
//| メモ:なし                                                        |
//+------------------------------------------------------------------+
bool CExpert::ReverseLong(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE) return(false);
//--- 反転ロットの取得
   double lot=LotReverse(sl);
//--- ロットをチェック
   if(lot==0.0) return(false);
//---
   return(m_trade.Sell(lot,price,sl,tp));
  }
```

© 2000-2025, MetaQuotes Ltd.

ReverseShort

ショート（売り）ポジションの反転操作を行います。

```
virtual bool  ReverseShort(
   double    price,    // 価格
   double    sl,       // 決済逆指値
   double    tp        // 決済指値
  )
```

パラメータ

価格

[in] 価格.

sl

[in] 決済逆指値

tp

[in] 決済指値

戻り値

取引操作が実行された場合はtrue、それ以外の場合はfalse

注意事項

ポジション反転ボリュームを取得し（LotReverse(...) メソッド）、取引高が 0 でない場合は、ショート（売り）ポジション反転の取引操作を行います（取引オブジェクトの Buy() メソッド）。

実装

```
//+------------------------------------------------------------------+
//| ショートポジションの反転                                         |
//| 入力:price - price,                                              |
//|          sl    - 決済逆指値                                      |
//|          tp    - 決済指値                                        |
//| 出力:取引操作が処理された場合は true、それ以外の場合は false     |
//| メモ:なし                                                        |
//+------------------------------------------------------------------+
bool CExpert::ReverseShort(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE) return(false);
//--- 反転ロットの取得
   double lot=LotReverse(sl);
//--- ロットのチェック
   if(lot==0.0) return(false);
//---
   return(m_trade.Buy(lot,price,sl,tp));
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckClose

ポジションを決済する条件をチェックします。

```
virtual bool  CheckClose()
```

戻り値

取引操作が実行された場合は true、それ以外の場合は false

注意事項

1. エキスパートアドバイザーのストップアウト条件（資金管理オブジェクトの CheckClose() メソッド）がチェックされます。条件が満たされている場合は、ポジションが決済されて全ての注文が削除され（CloseAll(...)）終了されます。あ
2. ロング（買い）またはショート（売り）ポジション決済の条件をチェックし（CheckCloseLong() または CheckCloseShort() メソッド）、ポジションが決済されたら、全ての注文を削除します。（DeleteOrders() メソッド。

実装

```
//+------------------------------------------------------------------+
//|  ポジションの決済または逆指値/指値注文削除をチェックする                |
//|  入力:なし                                                       |
//|  出力:取引操作が処理された場合は true、 それ以外の場合は false      |
//|  メモ:なし                                                       |
//+------------------------------------------------------------------+
bool CExpert::CheckClose()
  {
   double lot;
//--- ポジションは呼び出しの前に選択されなければならない
   if((lot=m_money.CheckClose(GetPointer(m_position)))!=0.0)
      return(CloseAll(lot));
//--- ポジションの種類をチェックする
   if(m_position.PositionType()==POSITION_TYPE_BUY)
     {
      //--- ロングポジションを決済するか未決済の買注文を削除できるかを見る
      if(CheckCloseLong())
        {
         DeleteOrders();
         return(true);
        }
     }
   else
     {
      //--- ショートポジションを決済するか未決済の売注文を削除できるかを見る
      if(CheckCloseShort())
        {
         DeleteOrders();
         return(true);
        }
     }
//--- 操作なしで返る
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckCloseLong

ロング（買い）ポジションを決済する条件をチェックします。

```
virtual bool  CheckCloseLong()
```

戻り値

取引操作が実行された場合はtrue、それ以外の場合はfalse

注意事項

ロング（買い）ポジション決済の条件をチェックし（Signalオブジェクトの CheckCloseLong() メソッド）、満たされている場合はポジションを決済します（CloseLong(...) メソッド）。

実装

```
//+------------------------------------------------------------------+
//| ロング(買い)ポジションの決済または逆指値/指値注文削除をチェックする          |
//| 入力:なし                                                          |
//| 出力:取引操作が処理された場合は true、 それ以外の場合は false       |
//| メモ:なし                                                          |
//+------------------------------------------------------------------+
bool CExpert::CheckCloseLong()
  {
   double price=EMPTY_VALUE;
//--- ロング決済操作をチェックする
   if(m_signal.CheckCloseLong(price))
      return(CloseLong(price));
//--- 操作なしで返る
   return(false);
  }
```

© 2000–2025, MetaQuotes Ltd.

CheckCloseShort

ショート（売り）ポジションを決済する条件のチェックをします。

```
virtual bool  CheckCloseShort()
```

戻り値

取引操作が実行された場合はtrue、それ以外の場合はfalse

注意事項

ショート（売り）ポジション決済の条件をチェックし（Signal オブジェクトの CheckCloseShort() メソッド）、満たされている場合はポジションを決済します。（CloseShort() メソッド）。

実装

```
//+------------------------------------------------------------------+
//| ショートポジションの決済または逆指値/指値注文削除をチェックする          |
//| 入力:なし                                                        |
//| 出力:取引操作が処理された場合は true、 それ以外の場合は false      |
//| メモ:なし                                                        |
//+------------------------------------------------------------------+
bool CExpert::CheckCloseShort()
  {
   double price=EMPTY_VALUE;
//--- ショート決済操作をチェックする
   if(m_signal.CheckCloseShort(price))
      return(CloseShort(price));
//--- 操作なしで返る
   return(false);
  }
```

© 2000–2025, MetaQuotes Ltd.

CloseAll

部分的または完全なポジションの決済を行います。

```
virtual bool  CloseAll(
   double    lot    // ロット
  )
```

パラメータ

lot

[in] ポジション減少に使用されるロット数

戻り値

取引操作が実行された場合はtrue、それ以外の場合はfalse

注意事項

「ネッティング」モードでは、ポジションはCExpertTrade::BuyまたはCExpertTrade::Sellメソッドを用いて決済されます。「ヘッジ」モードでは、CTrade::　PositionCloseメソッドが使われ、これはネッティングモードを持つ口座でも使用可能です。DeleteOrders()メソッドは、注文を削除するために使用されます。

実装

```
//+------------------------------------------------------------------+
//|  ポジション決済と注文削除                                        |
//|  入力:決済ボリューム(ロット単位)                                 |
//|  出力:取引操作が処理された場合はtrue、それ以外の場合はfalse      |
//|  メモ:なし                                                       |
//+------------------------------------------------------------------+
bool CExpert::CloseAll(double lot)
  {
   bool result;
//--- 決済操作をチェックする
   if(m_position.PositionType()==POSITION_TYPE_BUY) result=m_trade.Sell(lot,0,0,0);
   else                                             result=m_trade.Buy(lot,0,0,0);
   result|=DeleteOrders();
//---
   return(result);
  }
```

© 2000-2025, MetaQuotes Ltd.

Close

未決済ポジションを決済します。

```
virtual bool  Close()
```

戻り値

取引操作が実行された場合はtrue、それ以外の場合はfalse

注意事項

未決済ポジションを決済します（CTradeクラスオブジェクトのPositionClose()メソッド）。

実装

```
//+------------------------------------------------------------------+
//|  ポジション決済                                                  |
//|  入力:なし                                                       |
//|  出力:取引操作が処理された場合はtrue、それ以外の場合はfalse      |
//|  メモ:なし                                                       |
//+------------------------------------------------------------------+
bool CExpert::Close()
  {
   return(m_trade.PositionClose(m_symbol.Name()));
  }
```

© 2000–2025, MetaQuotes Ltd.

CloseLong

ロング（買い）ポジションを決済します。

```
virtual bool  CloseLong(
   double    price    // 価格
  )
```

パラメータ

価格

[in] 価格.

戻り値

取引操作が実行された場合はtrue、それ以外の場合はfalse

注意事項

ロングポジションを決済します（CTradeクラスオブジェクトのSell(...)メソッド）。

実装

```
//+------------------------------------------------------------------+
//| ロングポジション決済                                             |
//| 入力:price - 決済価格                                            |
//| 出力:取引操作が処理された場合はtrue、それ以外の場合はfalse       |
//| メモ:なし                                                        |
//+------------------------------------------------------------------+
bool CExpert::CloseLong(double price)
  {
   if(price==EMPTY_VALUE) return(false);
//---
   return(m_trade.Sell(m_position.Volume(),price,0,0));
  }
```

© 2000-2025, MetaQuotes Ltd.

CloseShort

ショート（売り）ポジションを決済します。

```
virtual bool  CloseShort(
   double    price    // 価格
  )
```

パラメータ

価格

[in] 価格.

戻り値

取引操作が実行された場合はtrue、それ以外の場合はfalse

注意事項

ショートポジションを決済します（CTradeクラスオブジェクトのBuy(...)メソッド）。

実装

```
//+------------------------------------------------------------------+
//| ショートポジション決済                                           |
//| 入力:price - 決済価格                                            |
//| 出力:取引操作が処理された場合はtrue、それ以外の場合はfalse       |
//| メモ:なし                                                        |
//+------------------------------------------------------------------+
bool CExpert::CloseShort(double price)
  {
   if(price==EMPTY_VALUE) return(false);
//---
   return(m_trade.Buy(m_position.Volume(),price,0,0));
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStop

未決済ポジションのトレール注文条件をチェックします。

```
virtual bool  CheckTrailingStop()
```

戻り値

取引操作が実行された場合は true、それ以外の場合は false

注意事項

未決済ポジションのトレール注文条件をチェックします。（ロングとショートポジションにそれぞれ CheckTrailingStopLong() または CheckTrailingStopShort() が使用されます）。

実装

```
//+------------------------------------------------------------------+
//| トレールストップ/プロフィット注文をチェックする                  |
//| 入力:なし                                                        |
//| 出力:取引操作が処理された場合は true、それ以外の場合は false    |
//| メモ:なし                                                        |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingStop()
  {
//--- ポジションは呼び出しの前に選択されなければならない
   if(m_position.PositionType()==POSITION_TYPE_BUY)
     {
      //--- ロングポジション変更が可能かをチェックする
      if(CheckTrailingStopLong()) return(true);
     }
   else
     {
      //--- ショートポジション変更が可能かをチェックする
      if(CheckTrailingStopShort()) return(true);
     }
//--- 操作なしで返る
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopLong

未決済のロング(買い)ポジションのトレール注文条件をチェックします。

```
virtual bool  CheckTrailingStopLong()
```

戻り値

取引操作が実行された場合はtrue、それ以外の場合はfalse

注意事項

未決済のロング(買い)ポジションのトレール注文条件をチェックします。(Expert Trailing オブジェクトのCheckTrailingStopLong(...) メソッド)。条件が満たされている場合は、ポジションパラメータが変更されます(TrailingStopLong(...) メソッド)。

実装

```
//+------------------------------------------------------------------+
//| ロングトレールストップ/プロフィット注文をチェックする                |
//| 入力:なし                                                        |
//| 出力:取引操作が処理された場合は true、それ以外の場合は false      |
//| メモ:なし                                                        |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingStopLong()
  {
   double sl=EMPTY_VALUE;
   double tp=EMPTY_VALUE;
//--- ロングトレイルストップ操作をチェックする
   if(m_trailing.CheckTrailingStopLong(GetPointer(m_position),sl,tp))
     {
      if(sl==EMPTY_VALUE) sl=m_position.StopLoss();
      if(tp==EMPTY_VALUE) tp=m_position.TakeProfit();
      //--- ロングトレイルストップ操作
      return(TrailingStopLong(sl,tp));
     }
//--- 操作なしで返る
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopShort

未決済のショート（売り）ポジションのトレール注文条件をチェックします。

```
virtual bool  CheckTrailingStopShort()
```

戻り値

取引操作が実行された場合はtrue、それ以外の場合はfalse

注意事項

未決済のショート（売り）ポジションのトレール注文条件をチェックします。（Expert Trailing オブジェクトのCheckTrailingStopShort(...) メソッド）。条件が満たされている場合は、ポジションパラメータが変更されます（TrailingStopShort(...) メソッド）。

実装

```
//+------------------------------------------------------------------+
//| ショートトレールストップ/プロフィット注文をチェックする                          |
//| 入力:なし                                                          |
//| 出力:取引操作が処理された場合は true、それ以外の場合は false          |
//| メモ:なし                                                          |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingStopShort()
  {
   double sl=EMPTY_VALUE;
   double tp=EMPTY_VALUE;
//--- ショートトレイルストップ操作をチェックする
   if(m_trailing.CheckTrailingStopShort(GetPointer(m_position),sl,tp))
     {
      if(sl==EMPTY_VALUE) sl=m_position.StopLoss();
      if(tp==EMPTY_VALUE) tp=m_position.TakeProfit();
      //--- ショートトレイルストップ操作
      return(TrailingStopShort(sl,tp));
     }
//--- 操作なしで返る
   return(false);
  }
```

© 2000–2025, MetaQuotes Ltd.

TrailingStopLong

未決済のロング（買い）ポジションのパラメータを変更します。

```
virtual bool  TrailingStopLong(
   double    sl,    // 決済逆指値
   double    tp,    // 決済指値
  )
```

パラメータ

sl

[in] 決済逆指値

tp

[in] 決済指値

戻り値

取引操作が実行された場合はtrue、それ以外の場合はfalse

注意事項

この関数は未決済のロング（買い）ポジションのパラメータを変更します（CTrade クラスオブジェクトのPositionModify(...)メソッド）。

実装

```
//+------------------------------------------------------------------+
//| トレイルストップ/プロフィットロングポジション                   |
//| 入力:sl - 新しい決済逆指値                                       |
//|         tp - 新しい決済逆指値                                    |
//| 出力:取引操作が処理された場合は true、それ以外の場合は false    |
//| メモ:なし                                                        |
//+------------------------------------------------------------------+
bool CExpert::TrailingStopLong(double sl,double tp)
  {
   return(m_trade.PositionModify(m_symbol.Name(),sl,tp));
  }
```

© 2000–2025, MetaQuotes Ltd.

TrailingStopShort

未決済のショート（売り）ポジションのパラメータを変更します。

```
virtual bool  TrailingStopLong(
   double    sl,    // 決済逆指値
   double    tp,    // 決済指値
  )
```

パラメータ

sl

[in] 決済逆指値

tp

[in] 決済指値

戻り値

取引操作が実行された場合はtrue、それ以外の場合はfalse

注意事項

この関数は未決済のショート（売り）ポジションのパラメータを変更します（CTrade クラスオブジェクトのPositionModify(...)メソッド）。

実装

```
//+------------------------------------------------------------------+
//| トレイルストップ/プロフィットショートポジション                  |
//| 入力:sl - 新しい決済逆指値                                       |
//|         tp - 新しい決済指値                                      |
//| 出力:取引操作が処理された場合は true、それ以外の場合は false    |
//| メモ:なし                                                        |
//+------------------------------------------------------------------+
bool CExpert::TrailingStopShort(double sl,double tp)
  {
   return(m_trade.PositionModify(m_symbol.Name(),sl,tp));
  }
```

© 2000–2025, MetaQuotes Ltd.

CheckTrailingOrderLong

買い逆指値/指値注文のトレール注文条件をチェックします。

```
virtual bool  CheckTrailingOrderLong()
```

戻り値

取引操作が実行された場合はtrue、それ以外の場合はfalse

注意事項

買い逆指値/指値注文のトレール注文条件をチェックし（Trade Signals オブジェクトのCheckTrailingOrderLong() メソッド）、必要な場合は注文パラメータを変更します。（TrailingOrderLong(...) メソッド）。

実装

```
//+------------------------------------------------------------------+
//| ロング(買い)トレイル逆指値/指値注文をチェックする                  |
//| 入力:なし                                                        |
//| 出力:取引操作が処理された場合は true、それ以外の場合は false      |
//| メモ:なし                                                        |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingOrderLong()
  {
   double price;
//--- ロング注文の変更が可能かをチェックする
   if(m_signal.CheckTrailingOrderLong(GetPointer(m_order),price))
      return(TrailingOrderLong(m_order.PriceOpen()-price));
//--- 操作なしで返る
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckTrailingOrderShort

売り逆指値/指値注文のトレール注文条件をチェックします。

```
virtual bool  CheckTrailingOrderShort()
```

戻り値

取引操作が実行された場合はtrue、それ以外の場合はfalse

注意事項

売り逆指値/指値注文のトレール注文条件をチェックし（Trade Signals オブジェクトのCheckTrailingOrderShort() メソッド）、必要な場合は注文パラメータを変更します。（TrailingOrderShort() メソッド）。

実装

```
//+------------------------------------------------------------------+
//| ショートトレイル逆指値/指値注文をチェックする                    |
//| 入力:なし                                                        |
//| 出力:取引操作が処理された場合はtrue、それ以外の場合はfalse      |
//| メモ:なし                                                        |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingOrderShort()
  {
   double price;
//--- ショート注文の変更が可能かをチェックする
   if(m_signal.CheckTrailingOrderShort(GetPointer(m_order),price))
      return(TrailingOrderShort(m_order.PriceOpen()-price));
//--- 操作なしで返る
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

TrailingOrderLong

買い逆指値/指値注文のパラメータを変更します。

```
virtual bool  TrailingOrderLong(
   double    delta    // デルタ(差)
  )
```

パラメータ

delta

[in] 価格データ

戻り値

取引操作が実行された場合はtrue、それ以外の場合はfalse

注意事項

買い逆指値/指値注文のパラメータを変更します(CTradeクラスオブジェクトのOrderModify(...)メソッド)。

実装

```
//+------------------------------------------------------------------+
//| トレイリング買いリミット/ストップ注文                                |
//| 入力:delta - 価格変更                                             |
//| 出力:取引操作が処理された場合はtrue、それ以外の場合はfalse          |
//| メモ:なし                                                         |
//+------------------------------------------------------------------+
bool CExpert::TrailingOrderLong(double delta)
  {
   ulong  ticket=m_order.Ticket();
   double price =m_order.PriceOpen()-delta;
   double sl    =m_order.StopLoss()-delta;
   double tp    =m_order.TakeProfit()-delta;
//--- ロング注文の変更
   return(m_trade.OrderModify(ticket,price,sl,tp,m_order.TypeTime(),m_order.TimeExpi
ration()));
  }
```

© 2000-2025, MetaQuotes Ltd.

TrailingOrderShort

売り逆指値/指値注文のパラメータを変更します。

```
virtual bool  TrailingOrderShort(
   double    delta    // デルタ(差)
  )
```

パラメータ

delta

[in] 価格データ

戻り値

取引操作が実行された場合はtrue、それ以外の場合はfalse

注意事項

売り逆指値/指値注文のパラメータを変更します(CTradeクラスオブジェクトのOrderModify(...)メソッド)。

実装

```
//+------------------------------------------------------------------+
//| トレイリング売りリミット/ストップ注文                                |
//| 入力:delta - price change.                                       |
//| 出力:取引操作が処理された場合はtrue、それ以外の場合はfalse          |
//| メモ:なし                                                        |
//+------------------------------------------------------------------+
bool CExpert::TrailingOrderShort(double delta)
  {
   ulong  ticket=m_order.Ticket();
   double price =m_order.PriceOpen()-delta;
   double sl    =m_order.StopLoss()-delta;
   double tp    =m_order.TakeProfit()-delta;
//--- modifying the short order
   return(m_trade.OrderModify(ticket,price,sl,tp,m_order.TypeTime(),m_order.TimeExpi
ration()));
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckDeleteOrderLong

買い決注文削除の条件をチェックします。

```
virtual bool  CheckDeleteOrderLong()
```

戻り値

取引操作が実行された場合はtrue、それ以外の場合はfalse

注意事項

注文の期限をチェックします。買い逆指値/指値注文削除の条件をチェックし（Signal　クラスオブジェクトのCheckCloseLong(...)）、満たされている場合はポジションを決済します（DeleteOrderLong()メソッド）。

実装

```
//+------------------------------------------------------------------+
//|  ロング(買い)逆指値/指値注文削除をチェックする                    |
//|  入力:なし                                                       |
//|  出力:取引操作が処理された場合はtrue、それ以外の場合はfalse      |
//|  メモ:なし                                                       |
//+------------------------------------------------------------------+
bool CExpert::CheckDeleteOrderLong()
  {
   double price;
//--- ロング注文の削除をチェックする
   if(m_expiration!=0 && TimeCurrent()>m_expiration)
     {
      m_expiration=0;
      return(DeleteOrderLong());
     }
   if(m_signal.CheckCloseLong(price))
      return(DeleteOrderLong());
//--- 操作なしで返る
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckDeleteOrderShort

売り決注文削除の条件をチェックします。

```
virtual bool  CheckDeleteOrderShort()
```

戻り値

取引操作が実行された場合は true、それ以外の場合は false

注意事項

注文の期限をチェックします。売り決注文削除の条件をチェックし（Signal オブジェクトの CheckCloseShort(...) メソッド）、満たされている場合はポジションを決済します。（DeleteOrderShort() メソッド）。

実装

```
//+------------------------------------------------------------------+
//| ショート逆指値/指値注文削除をチェックする                        |
//| 入力:なし                                                        |
//| 出力:取引操作が処理された場合は true、それ以外の場合は false    |
//| メモ:なし                                                        |
//+------------------------------------------------------------------+
bool CExpert::CheckDeleteOrderShort()
  {
   double price;
//--- ショート注文の削除をチェックする
   if(m_expiration!=0 && TimeCurrent()>m_expiration)
     {
      m_expiration=0;
      return(DeleteOrderShort());
     }
   if(m_signal.CheckCloseShort(price))
      return(DeleteOrderShort());
//--- 操作なしで返る
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteOrders

全ての注文を削除します。

```
virtual bool  DeleteOrders()
```

戻り値

取引操作が実行された場合はtrue、それ以外の場合はfalse

注意事項

全ての注文を削除します。（DeleteOrder()）

実装

```
//+------------------------------------------------------------------+
//|  リミット/ストップ注文をすべて削除する                                  |
//|  入力:なし                                                        |
//|  出力:取引操作が処理された場合はtrue、それ以外の場合はfalse              |
//|  メモ:なし                                                        |
//+------------------------------------------------------------------+
bool CExpert::DeleteOrders()
  {
   bool result=false;
   int  total=OrdersTotal();
//---
   for(int i=total-1;i>=0;i--)
     {
      if(m_order.Select(OrderGetTicket(i)))
        {
         if(m_order.Symbol()!=m_symbol.Name()) continue;
         result|=DeleteOrder();
        }
     }
//---
   return(result);
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteOrder

逆指値/指値注文を削除します。

```
virtual bool  DeleteOrder()
```

戻り値

取引操作が実行された場合はtrue、それ以外の場合はfalse

注意事項

逆指値/指値注文を削除します(CTradeクラスオブジェクトのOrderDelete(...)メソッド)。

実装

```
//+------------------------------------------------------------------+
//| リミット/ストップ注文を削除する                                  |
//| 入力:なし                                                        |
//| 出力:取引操作が処理された場合はtrue、それ以外の場合はfalse      |
//| メモ:なし                                                        |
//+------------------------------------------------------------------+
bool CExpert::DeleteOrder()
  {
   return(m_trade.OrderDelete(m_order.Ticket()));
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteOrderLong

買い逆指値/指値注文を削除します。

```
virtual bool  DeleteOrderLong()
```

戻り値

取引操作が実行された場合はtrue、それ以外の場合はfalse

注意事項

買い逆指値/指値注文を削除します（CTradeクラスオブジェクトのOrderDelete(...)メソッド）。

実装

```
//+------------------------------------------------------------------+
//|  ロングリミット/ストップ注文を削除する                                  |
//|  入力:なし                                                        |
//|  出力:取引操作が処理された場合はtrue、それ以外の場合はfalse              |
//|  メモ:なし                                                        |
//+------------------------------------------------------------------+
bool CExpert::DeleteOrderLong()
  {
   return(m_trade.OrderDelete(m_order.Ticket()));
  }
```

© 2000–2025, MetaQuotes Ltd.

DeleteOrderShort

売り逆指値/指値注文を削除します。

```
virtual bool  DeleteOrderShort()
```

戻り値

取引操作が実行された場合は true、それ以外の場合は false

注意事項

売り逆指値/指値注文を削除します（CTrade クラスオブジェクトの OrderDelete(...) メソッド）。

実装

```
//+------------------------------------------------------------------+
//| ショートリミット/ストップ注文を削除する                                  |
//| 入力:なし                                                        |
//| 出力:取引操作が処理された場合は true、 それ以外の場合は false     |
//| メモ:なし                                                        |
//+------------------------------------------------------------------+
bool CExpert::DeleteOrderShort()
  {
   return(m_trade.OrderDelete(m_order.Ticket()));
  }
```

© 2000-2025, MetaQuotes Ltd.

LotOpenLong

買いの取引高を取得します。

```
double  LotOpenLong(
   double    price,   // 価格
   double    sl       // 決済逆指値
  )
```

パラメータ

価格

[in] 価格

sl

[in] 決済逆指値

戻り値

買いの取引高

注意事項

買いの取引高を取得します。(資金管理オブジェクトの CheckOpenLong(...) メソッド)。

実装

```
//+------------------------------------------------------------------+
//| ロングポジションのロット取得の方法                               |
//| 入力:price - 価格                                                |
//|          sl    - 決済逆指値                                      |
//| OUTPUT:  未決済ロット                                            |
//| REMARK:  なし                                                    |
//+------------------------------------------------------------------+
double CExpert::LotOpenLong(double price,double sl)
  {
   return(m_money.CheckOpenLong(price,sl));
  }
```

© 2000-2025, MetaQuotes Ltd.

LotOpenShort

売りの取引高を取得します。

```
double  LotOpenShort(
   double    price,   // 価格
   double    sl       // 決済逆指値
  )
```

パラメータ

価格

[in] 価格

sl

[in] 決済逆指値

戻り値

ロット単位での売りの取引高

注意事項

売りの取引高を取得します。(資金管理オブジェクトの CheckOpenShort(...) メソッド)。

実装

```
//+------------------------------------------------------------------+
//| ショートポジションのロット取得の方法                             |
//| 入力:price - 価格                                                |
//|          sl    - 決済逆指値                                      |
//| OUTPUT:  未決済ロット                                            |
//| REMARK:  なし                                                    |
//+------------------------------------------------------------------+
double CExpert::LotOpenShort(double price,double sl)
  {
   return(m_money.CheckOpenShort(price,sl));
  }
```

© 2000-2025, MetaQuotes Ltd.

LotReverse

ポジション反転の取引高を取得します。

```
double  LotReverse(
   double    sl       // 決済逆指値
  )
```

パラメータ

sl

[in] 決済逆指値

戻り値

ロット単位でのポジション反転の取引高

注意事項

ポジション反転の取引高を取得します。(資金管理オブジェクトの CheckReverse(...) メソッド)。

実装

```
//+------------------------------------------------------------------+
//| ポジション反転のロット取得の方法                                 |
//| 入力:sl - 決済逆指値                                             |
//| 出力:未決済ロット                                                |
//| メモ:なし                                                        |
//+------------------------------------------------------------------+
double CExpert::LotReverse(double sl)
  {
   return(m_money.CheckReverse(GetPointer(m_position),sl));
  }
```

© 2000-2025, MetaQuotes Ltd.

PrepareHistoryDate

取引履歴追跡の開始日を設定します。

```
void  PrepareHistoryDate()
```

注意事項

取引履歴の追跡期間は月の最初から設定されています（ただし一日以上）。

© 2000–2025, MetaQuotes Ltd.

HistoryPoint

取引履歴のチェックポイントを作成します。(ポジション、注文、注文の数と履歴注文が保存されます)。

```
void  HistoryPoint(
   bool   from_check_trade=false   // フラグ
  )
```

パラメータ

from_check_trade=false

[in] 再帰を避けるためのフラグ

注意事項

ポジション、注文、注文の数と履歴注文が保存されます。

© 2000-2025, MetaQuotes Ltd.

CheckTradeState

保存された状態と現在の状態の比較して、対応するイベントハンドラを呼び出します。

```
bool  CheckTradeState()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

ポジション、注文、注文及び履歴注文の数を HistoryPoint() メソッドで保存されたものと比較してチェックします。取引履歴が変更された場合、対応する仮想イベントハンドラが呼ばれます。

© 2000–2025, MetaQuotes Ltd.

WaitEvent

イベント待機フラグを設定します。

```
void  WaitEvent(
   ENUM_TRADE_EVENTS    event          // フラグ
  )
```

パラメータ

event

[in] 設定するイベントのフラグ（ENUM_TRADE_EVENTS 列挙）

戻り値

なし

イベントフラグ

```
//--- 予期されたイベントのフラグ
enum ENUM_TRADE_EVENTS
  {
   TRADE_EVENT_NO_EVENT                =0,         // イベントが予期されない
   TRADE_EVENT_POSITION_OPEN           =0x1,       // 「注文を出す」イベントが予期されるフラグ
   TRADE_EVENT_POSITION_VOLUME_CHANGE=0x2,         // 「ポジションボリュームの変更」イベントが予期されるフラグ
   TRADE_EVENT_POSITION_MODIFY         =0x4,       // 「ポジションのストップ注文の変更」イベントが予期されるフラグ
   TRADE_EVENT_POSITION_CLOSE          =0x8,       // 「ポジション決済」イベントが予期されるフラグ
   TRADE_EVENT_POSITION_STOP_TAKE      =0x10,      // 「ストップ注文トリガ」イベントが予期されるフラグ
   TRADE_EVENT_ORDER_PLACE             =0x20,      // 「未決注文を出す」イベントが予期されるフラグ
   TRADE_EVENT_ORDER_MODIFY            =0x40,      // 「未決注文変更」イベントが予期されるフラグ
   TRADE_EVENT_ORDER_DELETE            =0x80,      // 「未決注文削除」イベントが予期されるフラグ
   TRADE_EVENT_ORDER_TRIGGER           =0x100      // 「未決注文トリガ」イベントが予期されるフラグ
  };
```

© 2000–2025, MetaQuotes Ltd.

NoWaitEvent

イベント待機フラグをリセットします。

```
void  NoWaitEvent(
   ENUM_TRADE_EVENTS     event          // フラグ
  )
```

パラメータ

event

[in] リセットするイベントのフラグ（ENUM_TRADE_EVENTS 列挙）

戻り値

なし

イベントフラグ

```
//--- 予期されたイベントのフラグ
enum ENUM_TRADE_EVENTS
  {
   TRADE_EVENT_NO_EVENT                =0,         // イベントが予期されない
   TRADE_EVENT_POSITION_OPEN           =0x1,       // 「注文を出す」イベントが予期されるフラグ
   TRADE_EVENT_POSITION_VOLUME_CHANGE=0x2,         // 「ポジションボリュームの変更」イベントが予期されるフラグ
   TRADE_EVENT_POSITION_MODIFY         =0x4,       // 「ポジションのストップ注文の変更」イベントが予期されるフラグ
   TRADE_EVENT_POSITION_CLOSE          =0x8,       // 「ポジション決済」イベントが予期されるフラグ
   TRADE_EVENT_POSITION_STOP_TAKE      =0x10,      // 「ストップ注文トリガ」イベントが予期されるフラグ
   TRADE_EVENT_ORDER_PLACE             =0x20,      // 「未決注文を出す」イベントが予期されるフラグ
   TRADE_EVENT_ORDER_MODIFY            =0x40,      // 「未決注文変更」イベントが予期されるフラグ
   TRADE_EVENT_ORDER_DELETE            =0x80,      // 「未決注文削除」イベントが予期されるフラグ
   TRADE_EVENT_ORDER_TRIGGER           =0x100      // 「未決注文トリガ」イベントが予期されるフラグ
  };
```

© 2000–2025, MetaQuotes Ltd.

TradeEventPositionStopTake

「決済逆指及び決済指トリガ」イベントハンドラです。

```
virtual bool  TradeEventPositionStopTake()
```

戻り値

CExpert クラスメソッドは、何の操作も行わずに常に true を返します。

© 2000–2025, MetaQuotes Ltd.

TradeEventOrderTriggered

「未決注文トリガ」イベントハンドラです。

```
virtual bool  TradeEventOrderTriggered()
```

戻り値

CExpert クラスメソッドは、何の操作も行わずに常に true を返します。

© 2000–2025, MetaQuotes Ltd.

TradeEventPositionOpened

「注文を出す」イベントハンドラです。

```
virtual bool  TradeEventPositionOpened()
```

戻り値

CExpert クラスメソッドは、何の操作も行わずに常に true を返します。

© 2000–2025, MetaQuotes Ltd.

TradeEventPositionVolumeChanged

「ポジションボリューム変更」イベントハンドラです。

```
virtual bool  TradeEventPositionVolumeChanged()
```

戻り値

CExpert クラスメソッドは、何の操作も行わずに常に true を返します。

© 2000–2025, MetaQuotes Ltd.

TradeEventPositionModified

「ポジション変更」イベントハンドラです。

```
virtual bool  TradeEventPositionModified()
```

戻り値

CExpert クラスメソッドは、何の操作も行わずに常に true を返します。

© 2000–2025, MetaQuotes Ltd.

TradeEventPositionClosed

「ポジション決済」イベントハンドラです。

```
virtual bool  TradeEventPositionClosed()
```

戻り値

CExpert クラスメソッドは、何の操作も行わずに常に true を返します。

© 2000–2025, MetaQuotes Ltd.

TradeEventOrderPlaced

「未決注文を出す」イベントハンドラです。

```
virtual bool  TradeEventOrderPlaced()
```

戻り値

CExpert クラスメソッドは、何の操作も行わずに常に true を返します。

© 2000–2025, MetaQuotes Ltd.

TradeEventOrderModified

「未決注文変更」イベントハンドラです。

```
virtual bool  TradeEventOrderModified()
```

戻り値

CExpert クラスメソッドは、何の操作も行わずに常に true を返します。

© 2000-2025, MetaQuotes Ltd.

TradeEventOrderDeleted

「未決注文削除」イベントハンドラです。

```
virtual bool  TradeEventOrderDeleted()
```

戻り値

CExpert クラスメソッドは、何の操作も行わずに常に true を返します。

© 2000-2025, MetaQuotes Ltd.

TradeEventNotIdentified

未知イベントのハンドラです。

```
virtual bool  TradeEventNotIdentified()
```

戻り値

CExpert クラスメソッドは、何の操作も行わずに常に true を返します。

注意事項

複数の取引イベントが到着した場合、イベントを識別することが困難であるのでご注意ください。

© 2000-2025, MetaQuotes Ltd.

TimeframeAdd

時間軸を追跡に追加します。

```
void  TimeframeAdd(
   ENUM_TIMEFRAMES    period          // 時間軸
  )
```

パラメータ

period

[in] Timeframe (ENUM_TIMEFRAMES enumeration).

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

TimeframesFlags

このメソッドは、新しいバーを持つ時間軸を示すフラグを返します。

```
int  TimeframesFlags(
   MqlDateTime&     time          // 時刻変数
  )
```

パラメータ

time

[in] 新しい時刻を示す、参照で渡された MqlDateTime 型の変数

戻り値

新しいバーを持つ時間軸を示すフラグ

© 2000-2025, MetaQuotes Ltd.

CExpertSignal

CExpertSignal は売買シグナルの基本クラスです。CheckReverseLong() 及び CheckReverseShort() メソッドの他には何の操作もせず、インターフェースを提供します。

使用法

1売買シグナルのアルゴリズムを準備します。
2. CExpertSignal クラスから継承した自分自身の売買シグナルクラスを作成します。
3. 仮想メソッドを自分自身のアルゴリズムでオーバーライドします。

売買シグナルクラスの例は Expert¥Signal¥ フォルダに見られます。

説明

CExpertSignal は売買シグナルアルゴリズム実装の基本クラスです。

宣言

```
class CExpertSignal : public CExpertBase
```

タイトル

```
#include <Expert\ExpertSignal.mqh>
```

継承階層

CObject
  CExpertBase
    CExpertSignal

直接子孫

CSignalAC, CSignalAMA, CSignalAO, CSignalBearsPower, CSignalBullsPower, CSignalCCI, CSignalDeM, CSignalDEMA, CSignalEnvelopes, CSignalFrAMA, CSignalRSI, CSignalRVI, CSignalSAR, CSignalStoch, CSignalTEMA, CSignalTriX, CSignalWPR

クラスメソッド

初期化	
virtual InitIndicators	指標と時系列を初期化します。
virtual ValidationSettings	設定のチェック
virtual AddFilter	組み合わされたシグナルにフィルタを追加します。
Protected データへのアクセス	
BasePrice	基本価格レベルを設定します。
UsedSeries	使用される時系列のフラグを取得します。
パラメータ設定	

© 2000–2025, MetaQuotes Ltd.

初期化	
Weight	「Weight」パラメータの値を設定します。
PatternsUsage	「PatternsUsage」パラメータの値を設定します。
General	「General」パラメータの値を設定します。
Ignore	「Ignore」パラメータの値を設定します。
Invert	「Invert」パラメータの値を設定します。
ThresholdOpen	「ThresholdOpen」パラメータの値を設定します。
ThresholdClose	「ThresholdClose」パラメータの値を設定します。
PriceLevel	「PriceLevel」パラメータの値を設定します。
StopLevel	「StopLevel」パラメータの値を設定します。
TakeLevel	「TakeLevel」パラメータの値を設定します。
Expiration	「Expiration」パラメータの値を設定します。
Magic	「Magic」パラメータの値を設定します。
取引条件のチェック	
virtual CheckOpenLong	ロング（買い）注文を出す条件のチェック
virtual CheckCloseLong	ロング（買い）ポジションを決済する条件のチェック
virtual CheckOpenShort	ショート（売り）注文を出す条件のチェック
virtual CheckCloseShort	ショート（売り）ポジションを決済する条件のチェック
virtual CheckReverseLong	ロング（買い）ポジションを反転する条件のチェック
virtual CheckReverseShort	ショート（売り）ポジションを反転する条件のチェック
取引パラメータの設定	
virtual OpenLongParams	ロング（買い）注文を出す際のパラメータの設定
virtual OpenShortParams	ショート（売り）注文を出す際のパラメータの設定
virtual CloseLongParams	ロング（買い）ポジション決済のパラメータの設定
virtual CloseShortParams	ショート（売り）ポジション決済のパラメータの設定
注文トレイル条件のチェック	
virtual CheckTrailingOrderLong	買い未決注文のパラメータを変更する条件のチェック
virtual CheckTrailingOrderShort	売り未決注文のパラメータを変更する条件のチェック
成行注文結成をチェックする方法	

© 2000–2025, MetaQuotes Ltd.

初期化	
virtual LongCondition	買い条件をチェックした結果を取得します。
virtual ShortCondition	売り条件をチェックした結果を取得します。
virtual Direction	価格の「加重」方向を取得します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

クラスから継承されたメソッド CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries

© 2000–2025, MetaQuotes Ltd.

BasePrice

基本価格レベルを設定します。

```
void  BasePrice(
   double    value          // 新しい値
  )
```

パラメータ

value

[in] 基本価格レベルの新しい値

戻り値

なし

© 2000-2025, MetaQuotes Ltd.

UsedSeries

使用される時系列のフラグを取得します。

```
int  UsedSeries()
```

戻り値

銘柄/時間軸が操作中の銘柄/時間軸に対応する場合は使用される時系列のフラグ、その他の場合は 0

© 2000-2025, MetaQuotes Ltd.

Weight

「Weight」パラメータの新しい値を設定します。

```
void  Weight(
   double    value          // 新しい値
  )
```

パラメータ

value

[in]「Weight」の新しい値

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

PatternUsage

「PatternsUsage」パラメータの新しい値を設定します。

```
void  PatternUsage(
   double    value          // 新しい値
  )
```

パラメータ

value

[in]「PatternsUsage」の新しい値

戻り値

なし

© 2000-2025, MetaQuotes Ltd.

General

「General」パラメータの新しい値を設定します。

```
void  General(
   int     value          // 新しい値
  )
```

パラメータ

value

[in]「General」の新しい値

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

Ignore

「Ignore」パラメータの新しい値を設定します。

```
void  Ignore(
   long     value          // 新しい値
  )
```

パラメータ

value

[in]「Ignore」の新しい値

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

Invert

「Invert」パラメータの新しい値を設定します。

```
void  Invert(
   long     value          // 新しい値
  )
```

パラメータ

value

[in]「Invert」の新しい値

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

ThresholdOpen

「ThresholdOpen」パラメータの値を設定します。

```
void  ThresholdOpen(
   long     value          // 新しい値
  )
```

パラメータ

value

[in]「ThresholdOpen」の新しい値

戻り値

なし

注意事項

「ThresholdOpen」パラメータの範囲は 0~100 です。このパラメータは注文を出す「投票」をとっている時に使用されます。

© 2000-2025, MetaQuotes Ltd.

ThresholdClose

「ThresholdClose」パラメータの新しい値を設定します。

```
void  ThresholdOpen(
   long     value          // 新しい値
  )
```

パラメータ

value

[in]「ThresholdClose」の新しい値

戻り値

なし

注意事項

「ThresholdClose」パラメータの範囲は 0~100 です。このパラメータはポジション決済の「投票」をとっている時に使用されます。

© 2000-2025, MetaQuotes Ltd.

PriceLevel

「PriceLevel」パラメータの新しい値を設定します。

```
void  PriceLevel(
   double    value          // 新しい値
  )
```

パラメータ

value

[in]「PriceLevel」の新しい値

戻り値

なし

注意事項

「PriceLevel」の値は価格レベル単位で定義されています。価格レベル単位の数値は PriceLevelUnit() メソッドで返されます。「PriceLevel」は基本価格と相対して始値を定義するのに使用されます。

© 2000-2025, MetaQuotes Ltd.

StopLevel

「StopLevel」パラメータの新しい値を設定します。

```
void  StopLevel(
   double    value          // 新しい値
  )
```

パラメータ

value

[in]「StopLevel」の新しい値

戻り値

なし

注意事項

「StopLevel」の値は価格レベル単位で定義されています。価格レベル単位の数値は PriceLevelUnit() メソッドで返されます。「StopLevel」は基本価格と相対して決済逆指値を定義するのに使用されます。

© 2000–2025, MetaQuotes Ltd.

TakeLevel

「TakeLevel」パラメータの新しい値を設定します。

```
void  TakeLevel(
   double    value          // 新しい値
  )
```

パラメータ

value

[in]「TakeLevel」の新しい値

戻り値

なし

注意事項

「PriceLevel」の値は価格レベル単位で定義されています。価格レベル単位の数値は PriceLevelUnit() メソッドで返されます。「TakeLevel」は基本価格と相対して決済指値を定義するのに使用されます。

© 2000–2025, MetaQuotes Ltd.

Expiration

「Expiration」パラメータの値を設定します。

```
void  Expiration(
   int     value          // 新しい値
  )
```

パラメータ

value

[in]「Expiration」の新しい値

戻り値

なし

注意事項

「Expiration」パラメータの値はバー単位で定義されています。個のパラメータは、未決注文の取引の際に、未決注文の期限として使われます。

© 2000–2025, MetaQuotes Ltd.

Magic

「Magic」パラメータの値を設定します。

```
void  Magic(
   int     value          // 新しい値
  )
```

パラメータ

value

[in]「Magic」(エキスパートアドバイザー ID) の新しい値

戻り値

なし

© 2000-2025, MetaQuotes Ltd.

ValidationSettings

設定をチェックします。

```
virtual bool  ValidationSettings()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

InitIndicators

全ての指標と時系列を初期化解除します。

```
virtual bool  InitIndicators(
   CIndicators*  indicators    // ポインタ
  )
```

パラメータ

指標

[in] 指標と時系列の集合へのポインタ

戻り値

成功の場合は true、それ以外の場合は false

注意事項

時系列は、オブジェクトが、初期化時に定義されたのとは異なるシンボルまたは時間軸を使用する場合にのみ初期化されます。

© 2000–2025, MetaQuotes Ltd.

AddFilter

組み合わされたシグナルにフィルタを追加します。

```
virtual bool  AddFilter(
   CExpertSignal*  filter    // ポインタ
  )
```

パラメータ

指標

[in] フィルタオブジェクトへのポインタ

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CheckOpenLong

ロング（買い）注文を出す条件をチェックします。

```
virtual bool  CheckOpenLong(
   double&    price,          // 価格
   double&    sl,             // 決済逆指値
   double&    tp,             // 決済指値
   datetime&  expiration      // 期限
  )
```

パラメータ

価格

[in][out] 参照で渡された値の変数

sl

[in][out] 参照で渡された決済逆指値の変数

tp

[in][out] 参照で渡された決済指値の変数

expiration

[in][out] 参照で渡された期限の変数

戻り値

条件が満たされている場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CheckOpenShort

ショート（売り）注文を出す条件をチェックします。

```
virtual bool  CheckOpenShort(
   double&    price,          // 価格
   double&    sl,             // 決済逆指値
   double&    tp,             // 決済指値
   datetime&  expiration      // 期限
  )
```

パラメータ

価格

[in][out] 参照で渡された値の変数

sl

[in][out] 参照で渡された決済逆指値の変数

tp

[in][out] 参照で渡された決済指値の変数

expiration

[in][out] 参照で渡された期限の変数

戻り値

条件が満たされている場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OpenLongParams

ロング（買い）注文を出す際のパラメータを設定します。

```
virtual bool  OpenLongParams(
   double&    price,          // 価格
   double&    sl,             // 決済逆指値
   double&    tp,             // 決済指値
   datetime&  expiration      // 期限
  )
```

パラメータ

価格

[in][out] 参照で渡された値の変数

sl

[in][out] 参照で渡された決済逆指値の変数

tp

[in][out] 参照で渡された決済指値の変数

expiration

[in][out] 参照で渡された期限の変数

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OpenShortParams

ショート（売り）注文を出す際のパラメータを設定します。

```
virtual bool  OpenShortParams(
   double&    price,          // 価格
   double&    sl,             // 決済逆指値
   double&    tp,             // 決済指値
   datetime&  expiration      // 期限
  )
```

パラメータ

価格

[in][out] 参照で渡された値の変数

sl

[in][out] 参照で渡された決済逆指値の変数

tp

[in][out] 参照で渡された決済指値の変数

expiration

[in][out] 参照で渡された期限の変数

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CheckCloseLong

ロング（買い）ポジションを決済する条件をチェックします。

```
virtual bool  CheckCloseLong(
   double&  price      // 価格
  )
```

パラメータ

価格

[in][out] 参照で渡された終値の変数

戻り値

条件が満たされている場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CheckCloseShort

ショート（売り）ポジションを決済する条件のチェックをします。

```
virtual bool  CheckCloseShort(
   double&  price      // 価格
  )
```

パラメータ

価格

[in][out] 参照で渡された終値の変数

戻り値

条件が満たされている場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CloseLongParams

ロング（買い）ポジション決済のパラメータを設定します。

```
virtual bool  CloseLongParams(
   double&    price          // 価格
  )
```

パラメータ

価格

[in][out] 参照で渡された終値の変数

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CloseShortParams

ショート（売り）ポジション決済のパラメータを設定します。

```
virtual bool  CloseShortParams(
   double&    price          // 価格
  )
```

パラメータ

価格

[in][out] 参照で渡された終値の変数

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CheckReverseLong

ロング（買い）ポジションを反転する条件をチェックします。

```
virtual bool  CheckReverseLong(
   double&    price,          // 価格
   double&    sl,             // 決済逆指値
   double&    tp,             // 決済指値
   datetime&  expiration      // 期限
  )
```

パラメータ

価格

[in][out] 参照で渡された値の変数

sl

[in][out] 参照で渡された決済逆指値の変数

tp

[in][out] 参照で渡された決済指値の変数

expiration

[in][out] 参照で渡された期限の変数

戻り値

条件が満たされている場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CheckReverseShort

ショート（売り）ポジションを反転する条件をチェックします。

```
virtual bool  CheckReverseShort(
   double&    price,          // 価格
   double&    sl,             // 決済逆指値
   double&    tp,             // 決済指値
   datetime&  expiration      // 期限
  )
```

パラメータ

価格

[in][out] 参照で渡された反転の値の変数

sl

[in][out] 参照で渡された決済逆指値の変数

tp

[in][out] 参照で渡された決済指値の変数

expiration

[in][out] 参照で渡された期限の変数

戻り値

条件が満たされている場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CheckTrailingOrderLong

買い未決注文のパラメータを変更する条件をチェックします。

```
virtual bool  CheckTrailingOrderLong(
   COrderInfo*     order,          // 注文
   double&         price           // 価格
  )
```

パラメータ

order

[in] COrderInfo クラスオブジェクトへのポインタ

価格

[in][out] 決済逆指値の変数

戻り値

条件が満たされている場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CheckTrailingOrderShort

売り未決注文のパラメータを変更する条件をチェックします。

```
virtual bool  CheckTrailingOrderShort(
   COrderInfo*     order,          // 注文
   double&         price           // 価格
  )
```

パラメータ

order

[in] COrderInfo クラスオブジェクトへのポインタ

価格

[in][out] 決済逆指値の変数

戻り値

条件が満たされている場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

LongCondition

ロング（買い）注文を出す条件をチェックします。

```
virtual int  LongCondition()
```

戻り値

条件が満たされた場合には、（シグナルの「強さ」に依存して）1～100　の値。ロングポジションを取るためのシグナルがない場合には 0。

注意事項

基本クラスの LongCondition() メソッドは、ロング注文を出す条件のチェックを実装せず、常に 0 を返します。

© 2000–2025, MetaQuotes Ltd.

ShortCondition

ショート（売り）注文を出す条件をチェックします。

```
virtual int  ShortCondition()
```

戻り値

条件が満たされた場合には、（シグナルの「強さ」に依存して）1～100 の値。If there isn't a signal to open short position, it returns 0.

注意事項

基本クラスの ShortCondition() メソッドは、ショート注文を出す条件のチェックを実装せず、常に 0 を返します。

© 2000-2025, MetaQuotes Ltd.

Direction

価格の「加重」方向を取得します。

```
virtual double  Direction()
```

戻り値

上昇の際は正、下降の際は負の値絶対値は、シグナルの「強さ」に依存しています。

注意事項

フィルタが使用される場合、結果はフィルタに依存します。

© 2000–2025, MetaQuotes Ltd.

CExpertTrailing

CExpertTrailing はトレイルアルゴリズムの基本クラスです、インターフェースのみを提供します。

使用法

1Prepare an algorithm for trailing;
2. CExpertTrailing クラスから継承した自分自身のトレイルクラスを作成します。仮想メソッドを自分自身のアルゴリズムでオーバーライドします。

トレイルクラスの例は Expert¥Trailing¥ フォルダに見られます。

説明

CExpertTrailing はトレイルアルゴリズム実装の基本クラスです。

宣言

```
class CExpertTrailing : public CExpertBase
```

タイトル

```
#include <Expert\ExpertTrailing.mqh>
```

継承階層

CObject
  CExpertBase
    CExpertTrailing

直接子孫

CTrailingFixedPips, CTrailingMA, CTrailingNone, CTrailingPSAR

クラスメソッド

トレール注文条件のチェック	
virtual CheckTrailingStopLong	ロング（買い）ポジションのパラメータを変更する条件をチェックします。
virtual CheckTrailingStopShort	ショート（売り）ポジションのパラメータを変更する条件をチェックします。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

クラスから継承されたメソッド CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, ValidationSettings, SetPriceSeries, SetOtherSeries, InitIndicators

© 2000–2025, MetaQuotes Ltd.

CheckTrailingStopLong

ロング（買い）ポジションのパラメータを変更する条件をチェックします。

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,      // ポインタ
   double&         sl,            // 決済逆指
   double&         tp             // 決済指値
  )
```

パラメータ

position

[in] CPositionInfo クラスオブジェクトへのポインタ

sl

[in][out] 参照で渡された決済逆指値の変数

tp

[in][out] 参照で渡された決済指値の変数

戻り値

条件が満たされている場合は true、それ以外の場合は false

注意事項

基本クラスの CheckTrailingStopLong(...) メソッドは常に true を返します。

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopShort

ショート（売り）ポジションのパラメータを変更する条件をチェックします。

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,      // ポインタ
   double&         sl,            // 決済逆指
   double&         tp             // 決済指値
  )
```

パラメータ

position

[in] CPositionInfo クラスオブジェクトへのポインタ

sl

[in][out] 参照で渡された決済逆指値の変数

tp

[in][out] 参照で渡された決済指値の変数

戻り値

条件が満たされている場合は true、それ以外の場合は false

注意事項

基本クラスの CheckTrailingStopShort(...) メソッドは常に true を返します。

© 2000–2025, MetaQuotes Ltd.

CExpertMoney

CExpertMoney は資金及びリスク管理アルゴリズムの基本クラスです。

説明

CExpertMoney は資金及びリスク管理クラス実装の基本クラスです。

宣言

```
class CExpertMoney : public CObject
```

タイトル

```
#include <Expert\ExpertMoney.mqh>
```

継承階層

CObject
  CExpertBase
    CExpertMoney

直接子孫

CMoneyFixedLot, CMoneyFixedMargin, CMoneyFixedRisk, CMoneyNone, CMoneySizeOptimized

クラスメソッド

Protected データへのアクセス	
Percent	「リスクパーセント」パラメータの値を設定します。
初期化	
virtual ValidationSettings	設定のチェック
取引条件をチェックします。	
virtual CheckOpenLong	ロング（買い）ポジションのボリュームを取得します。
virtual CheckOpenShort	ショート（売り）ポジションのボリュームを取得します。
virtual CheckReverse	ポジション反転のボリュームを取得します。
virtual CheckClose	未決済ポジションを決済する条件をチェックします。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

クラスから継承されたメソッド CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

© 2000–2025, MetaQuotes Ltd.

Percent

「リスクパーセント」パラメータの値を設定します。

```
void  Percent(
   double  percent      // リスクパーセント
  )
```

パラメータ

percent

[in] リスクパーセント

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

ValidationSettings

設定をチェックします。

```
virtual bool  ValidationSettings()
```

戻り値

成功の場合は true、それ以外の場合は false

注意事項

基本クラスの ValidationSettings() メソッドは常に true を返します。

© 2000–2025, MetaQuotes Ltd.

CheckOpenLong

ロング（買い）ポジションのボリュームを取得します。

```
virtual double  CheckOpenLong(
   double  price,     // 価格
   double  sl         // 決済逆指値
  )
```

パラメータ

価格

[in] ロング（買い）ポジションの始値

sl

[in] 決済逆指値

戻り値

ロング（買い）ポジションの取引高

© 2000–2025, MetaQuotes Ltd.

CheckOpenShort

ショート（売り）ポジションのボリュームを取得します。

```
virtual double  CheckOpenShort(
   double  price,     // 価格
   double  sl         // 決済逆指値
  )
```

パラメータ

価格

[in] ショート（売り）ポジションの始値

sl

[in] 決済逆指値

戻り値

ショート（売り）ポジションの取引高

© 2000–2025, MetaQuotes Ltd.

CheckReverse

ポジション反転のボリュームを取得します。

```
virtual double  CheckReverse(
   CPositionInfo*  position,     // ポインタ
   double          sl            // 決済逆指値
  )
```

パラメータ

position

[in] CPositionInfo クラスオブジェクトへのポインタ

sl

[in] 決済逆指値

戻り値

ポジション反転のボリューム

© 2000-2025, MetaQuotes Ltd.

CheckClose

未決済ポジションを決済する条件をチェックします。

```
virtual double  CheckClose()
```

戻り値

条件が満たされている場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

売買シグナルのモジュール

クライアント端末の標準パッケージは「MQL5 Wizard」のための売買シグナルの既製モジュールのセットを含んでいます。MQL5 Wizard でのエキスパートアドバイザー作成の際に、売買シグナルモジュールの(64 までの)組み合わせを使用することが出来ます。取引操作の最終決定は、含まれたモジュールの全てから取得されたシグナルの複雑な分析に基づいて行われます。取引意思決定の仕組みの詳細な説明は下記で与えられています。

標準パッケージは以下のシグナルモジュールを含みます。

- AC オシレーター(Accelerator Oscillator)指標のシグナル
- 適応型移動平均(Adaptive Moving Average)指標のシグナル
- オーサムオシレーター(Awesome Oscillator)指標のシグナル
- ベアパワー(Bears Power)オシレーター指標のシグナル
- ブルパワー(Bulls Power)オシレーター指標のシグナル
- CCI(Commodity Channel Index、コモディティチャンネルインデックス)指標のシグナル
- デマーカー(DeMarker)オシレーター指標のシグナル
- 2 重指数移動平均(Double Exponential Moving Average)指標のシグナル
- エンベロープ(Envelopes)指標のシグナル
- フラクタル適応型移動平均(Fractal Adaptive Moving Average)指標のシグナル
- 日中時間フィルタのシグナル
- MACD オシレーターのシグナル
- 移動平均(Moving Average)指標のシグナル
- パラボリック SAR(Parabolic SAR) 指標のシグナル
- 相対力指数(Relative Strength Index)オシレーターのシグナル
- 相対力指数(Relative Vigor Index)オシレーターのシグナル
- ストキャスティックス(Stochastic)のシグナル
- 3 重指数平均(Triple Exponential Average)オシレーターのシグナル
- 3 重指数移動平均(Triple Exponential Moving Average)指標のシグナル
- ウィリアムパーセントレンジ(Williams Percent Range)オシレーターのシグナル

シグナルモジュールに基づいて取引決定を下すメカニズム

取引意思決定のメカニズムの基本原理は、以下のリストとして表すことが出来ます。

- シグナルの各モジュールは、マーケットモジュールのセット(価格及び指標の値の特定の組み合わせ)を有します。
- 各マーケットモデルは、1～100の範囲の意義を持っています。意義が高いほど、モデルは強くなります。
- それぞれのモデルは、価格の動きの方向の予測を生成します。
- モジュールの予測は、埋め込まれたモデルの検索の結果で、-100～100の範囲内の数として出力されます。符号は予測された動きの方向を決定します。(負の符号は価格の下降、正の符号は価格の上昇を意味します。)。絶対値が見つかった最良モデルの強度に対応しています。
- 各モジュールの予測は、設定で指定された 0~1 の範囲を持つ係数(「重さ」)とともに最後の「投票」に送信されます。

© 2000-2025, MetaQuotes Ltd.

- 投票の結果は、符号が予測移動方向を決定する-100～100　の範囲内の数であり、絶対値は、信号の強さを特徴付けます。これは、全てのシグナルモジュールの重み付きの予測の算術平均として計算されます。

生成されたエキスパートアドバイザーにはそれぞれ 2 つの調整可能設定があります。それは 0～100 の範囲を持つ、注文を出す及び決済する際のしきい値レベルです（ThresholdOpen　とThresholdClose）。最終のシグナル強度がしきい値レベルを超えた場合、シグナルの符号に対応する取引操作が行われます。

例

ThresholdOpen=20　及び　ThresholdClose=90　のしきい値を持つエキスパートアドバイザーを見てみましょう。「重さ」0.4 のMA シグナルモジュールと「重さ」0.8 の ストキャスティックス シグナルモジュールの 2 つが取引操作の決定に参加するとします。収得された売買シグナルの 2 つのバージョン体を分析してみましょう。

バージョン1

価格が上昇中のMAを上向きに交差しました。これはMA　モジュールに実装されているマーケットモデルの　1 つに対応します。このモデルは価格上昇を暗示します。その意義は、100　に等しいです。同時に、ストキャスティックス――が拒否して価格との乖離を形成しました。これはストキャスティックスモジュールに実装されているモデルの 1 つに対応します。このもでるは価格下降を暗示します。このモデルの「重み」は 80 です。

ここで「投票」の結果を計算しましょう。MA モジュールから取得された値は 0.4 * 100 = 40 として算出されます。ストキャスティックスモジュールから取得された値は 0.8 *（-80）= -64 として算出されます。最終的な値は、これら 2 つの値の算術平均として（4 - 64）/2 = -30 で計算されます　。投票の結果は 12 に等しい相対的な強さを持つ売りシグナルです。20　　のしきい値には届きません。従って、個の取引操作は行われません。

バージョン 2

価格が上昇中のMAを下向きに交差しました。これはMA　モジュールに実装されているモデルの　1　つに対応し、価格上昇を暗示します。その意義は、10 に等しいです。同時に、ストキャスティックス――が拒否して価格との乖離を形成しました。これはストキャスティックスモジュールに実装されているモデルの　1　つに対応します。このもでるは価格下降を暗示します。このモデルの「重み」は 80 です。

ここで「投票」の結果を計算しましょう。MA モジュールから取得された値は 0.4 * 10 = 4 として算出されます。ストキャスティックスモジュールから取得された値は 0.8 *（-80）= -64 として算出されます。最終値は、これら 2 つの値の算術平均として（4 - 64）/2 = -30 で計算されます　。投票の結果は 30 に等しい相対的な強さを持つ売りシグナルです。これは20　のしきい値を超えます。よって、ショート（売り）注文を出すシグナルが結果となります。

© 2000-2025, MetaQuotes Ltd.



a) 価格のストキャスティックスーの発散（変種1及び2）。

b) 価格がMAを上向きに交差しました（変種 1）。

c) 価格がMAを下向きに交差しました（変種 2）。

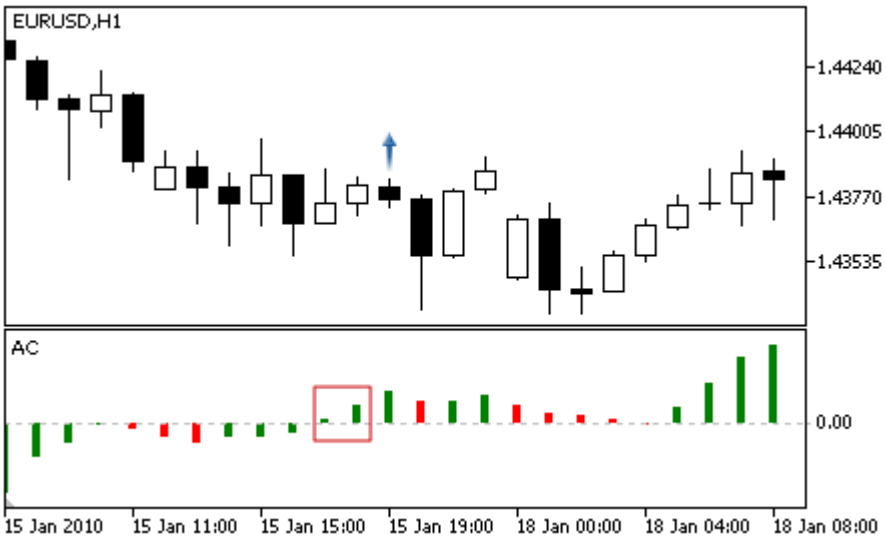
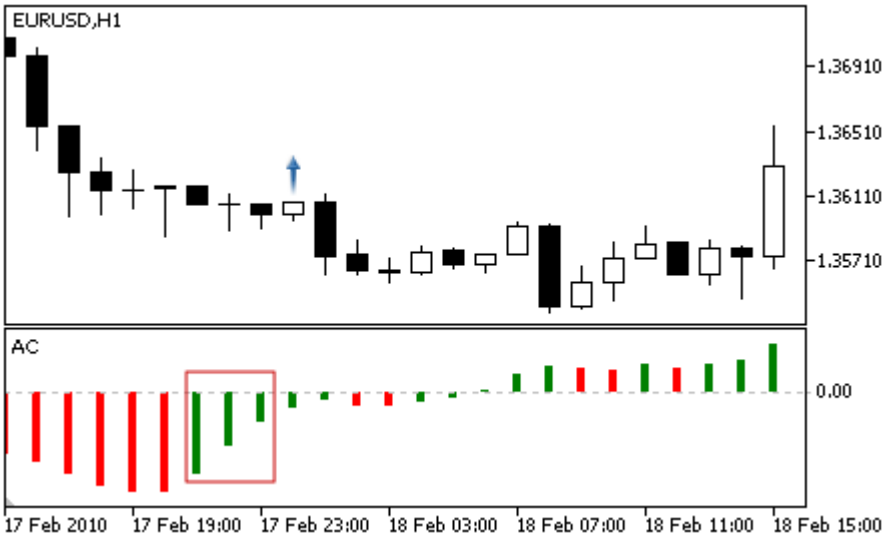
© 2000–2025, MetaQuotes Ltd.

AC オシレーター(Accelerator Oscillator) 指標のシグナル

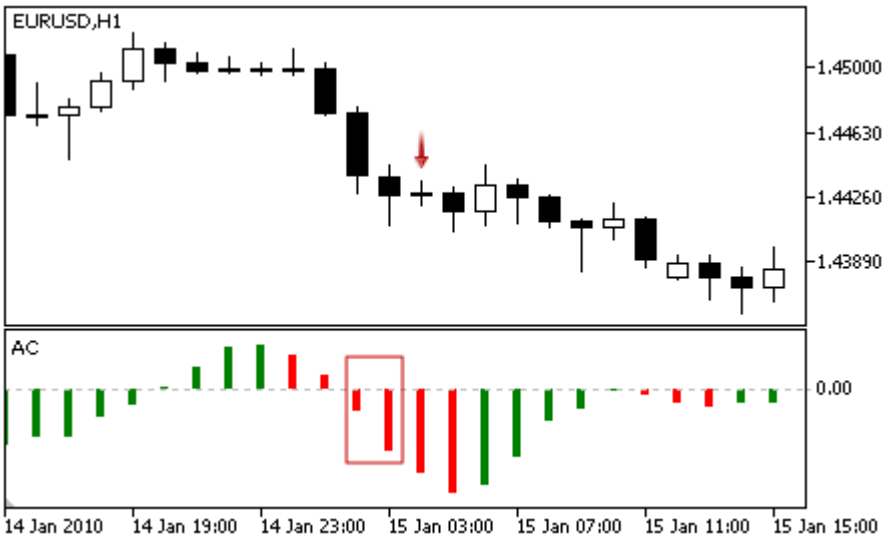
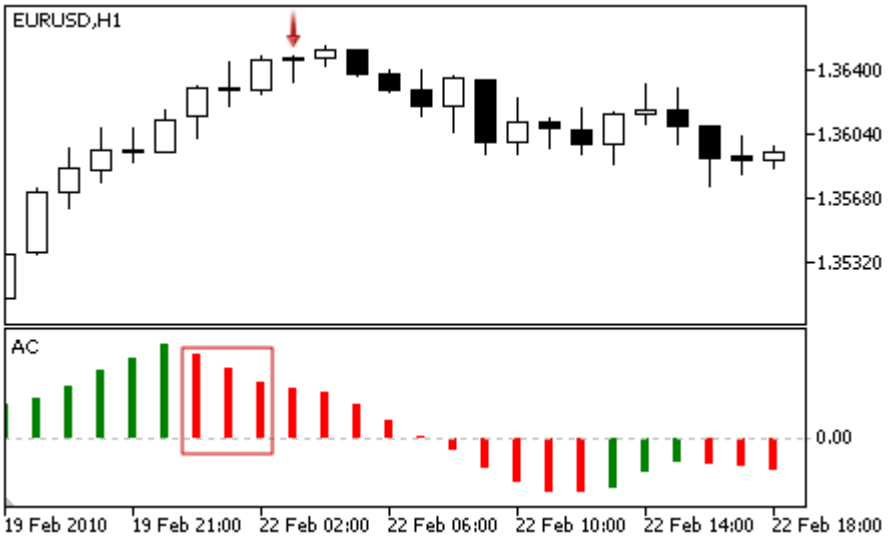
このモジュールはACオシレーター指標のマーケットモデルに基づいています。モジュールから得られるシグナルに基づいての取引意思決定の仕組みは別のセクションで説明されています。

シグナル生成の条件

下記は、モジュールがエキスパートアドバイザーへのシグナルを受け渡す条件についてを説明します。

シグナルの種類	条件の説明
買いシグナル	• 指標値が正で、分析されているバー及び1つ前のバーで上昇します。  • 指標値が負で、分析されているバー及び1つ前のバーで上昇します。 
売りシグナル	• 指標値が負で、分析されているバー及び1つ前のバーで下降します。

© 2000–2025, MetaQuotes Ltd.

シグナルの種類	条件の説明
	 • 指標値が負で、分析されているバー及び1 つ前のバーで下降します。 
買いに異議なし	分析されているバーで指標値が増加します。
売りに異議なし	分析されているバーで指標値が減少します。

注意事項

エキスパートアドバイザーの動作モード（「Every tick」（ティックごと）または「Open prices only」（始値のみ））に応じて、分析バーは（インデックス 0 の）現在足、または（インデックス 1 の）直近に形成されたバーのいずれかです。

調整可能のパラメータ

このモジュールは、以下の調整可能なパラメータを持っています。

パラメータ	説明
Weight	0～1 のモジュールのシグナルの重み

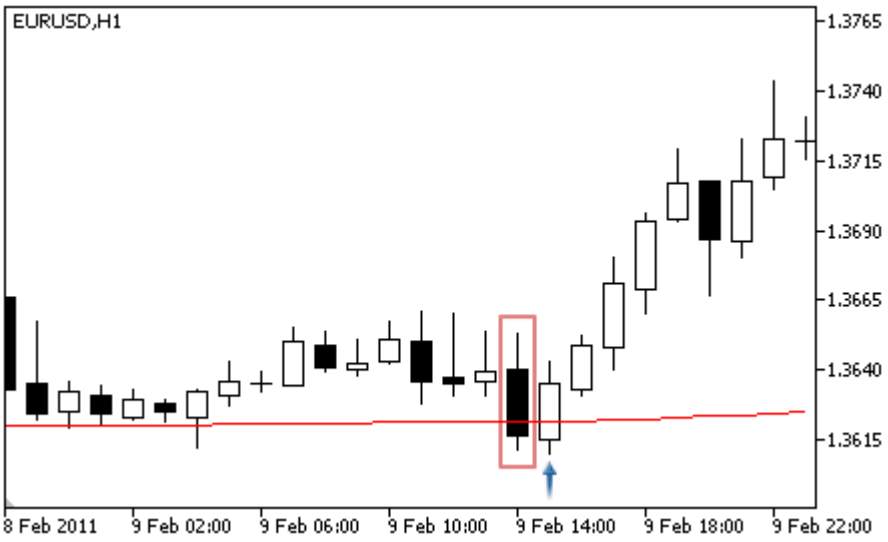
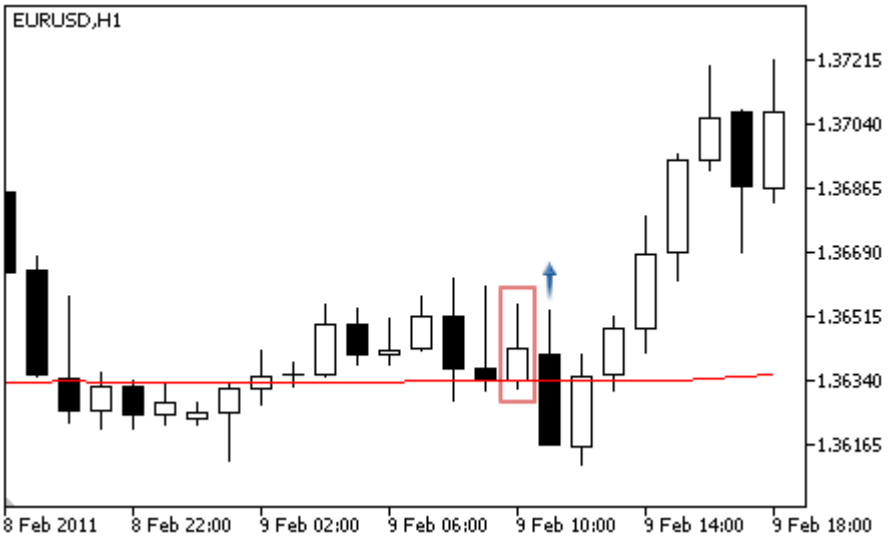
© 2000–2025, MetaQuotes Ltd.

適応型移動平均（Adaptive Moving Average）指標のシグナル

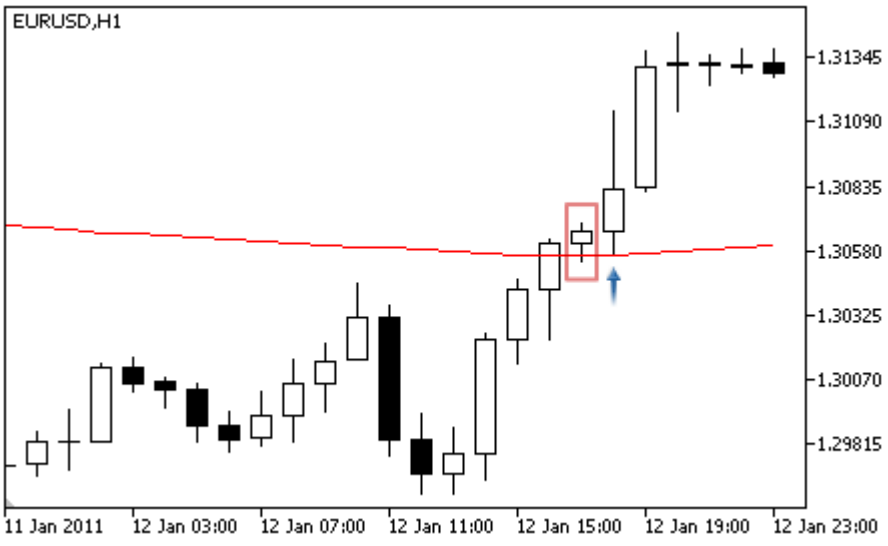
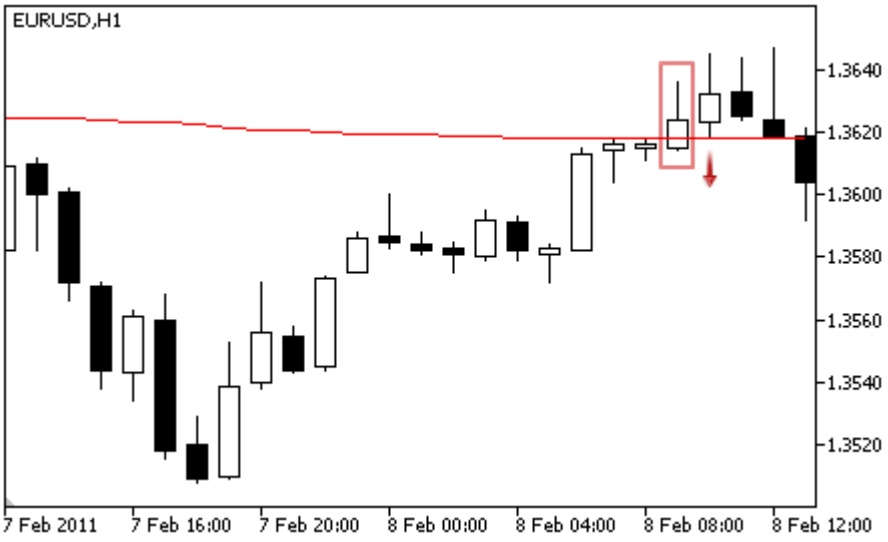
このモジュールは適応型移動平均指標のマーケットモデルに基づいています。モジュールから得られるシグナルに基づいての取引意思決定の仕組みは別のセクションで説明されています。

シグナル生成の条件

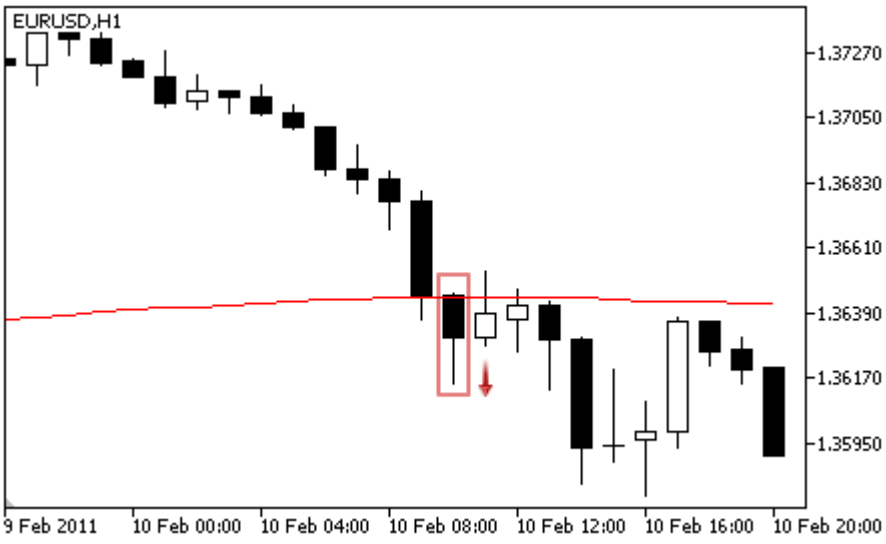
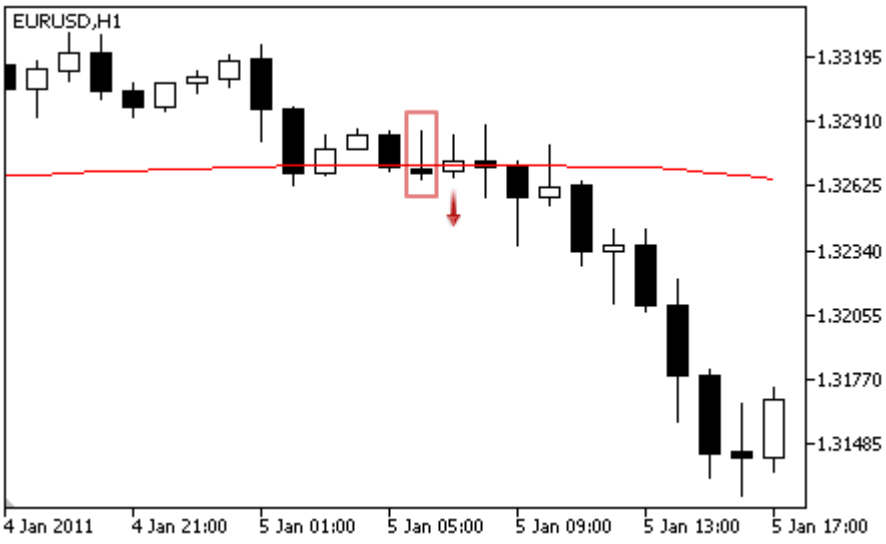
下記は、モジュールがエキスパートアドバイザーへのシグナルを受け渡す条件についてを説明します。

シグナルの種類	条件の説明
買いシグナル	• 価格が下向きに指標を交差（分析されているバーの始値が指標を上回り、終値が指標を下回る）し、指標が上昇しています（弱いシグナル）。  • 移動平均交差。 価格が上向きに指標を交差（分析されているバーの始値が指標を下回り、終値が指標を上回る）し、指標が上昇しています（強いシグナル）。  • バーの下部の影が指標と交差（分析されているバーの始値と終値が指標を上回り、安値が指標を下回る）し、指標が上昇しています（弱いシグナル）。

© 2000–2025, MetaQuotes Ltd.

シグナルの種類	条件の説明
	
売りシグナル	• 価格が上向きに指標と交差（分析されているバーの始値が指標を下回り、終値が指標を上回る）し、指標が下降しています（弱いシグナル）。  • 移動平均交差。価格が下向きに指標を交差（分析されているバーの始値が指標を上回り、終値が指標を下回る）し、指標が下降しています（強いシグナル）。

© 2000–2025, MetaQuotes Ltd.

シグナルの種類	条件の説明
	 • バーの上部の影が指標と交差（分析されているバーの始値と終値が指標を下回り、高値が指標を上回る）し、指標が下降しています（弱いシグナル）。 
買いに異議なし	価格が指標を上回ります。
売りに異議なし	価格が指標を下回ります。

注意事項

エキスパートアドバイザーの動作モード（「Every tick」（ティックごと）または「Open prices only」（始値のみ））に応じて、分析バーは（インデックス 0 の）現在足、または（インデックス 1 の）直近に形成されたバーのいずれかです。

調整可能のパラメータ

このモジュールは、以下の調整可能なパラメータを持っています。

© 2000–2025, MetaQuotes Ltd.

パラメータ	説明
Weight	0～1 のモジュールのシグナルの重み
PeriodMA	指標の平均化の期間
Shift	時間軸に沿っての指標のシフト（バー単位）
Method	平均化の方法
Applied	指標計算に使用される価格列***

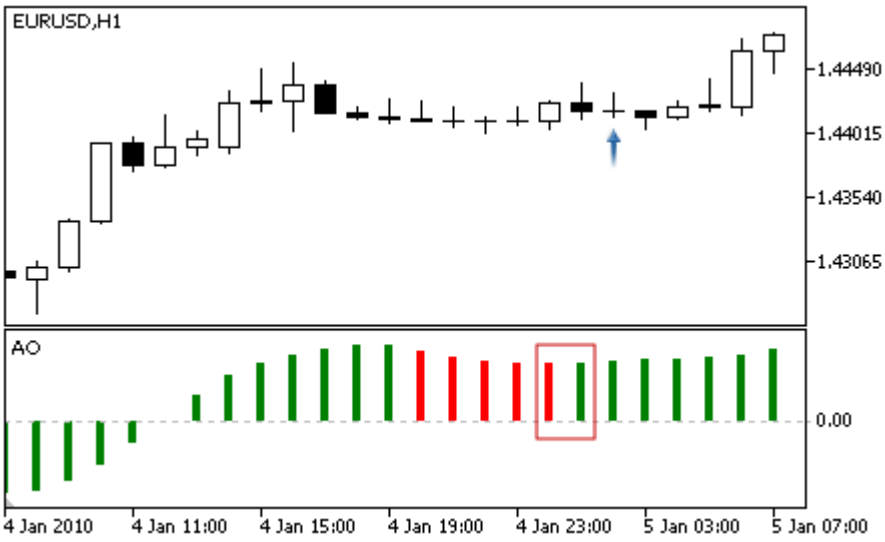
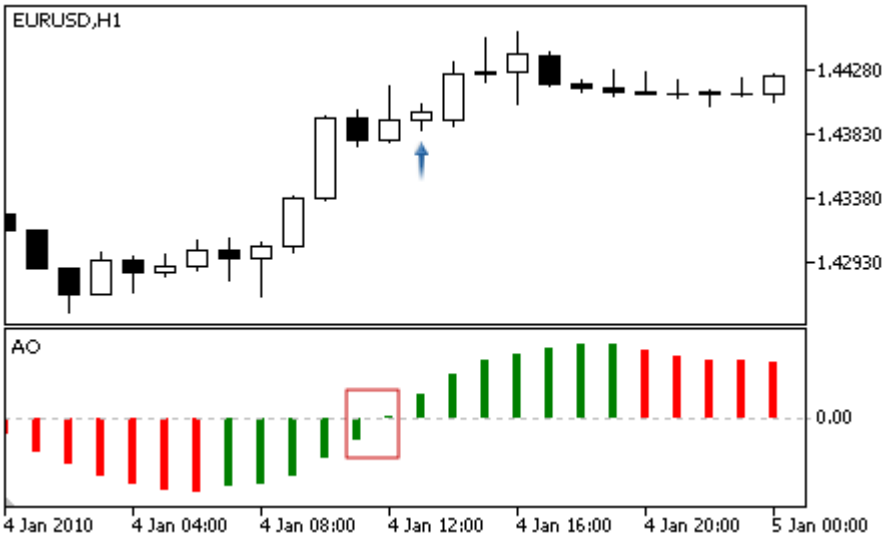
© 2000-2025, MetaQuotes Ltd.

オーサムオシレーター(Awesome Oscillator)指標のシグナル

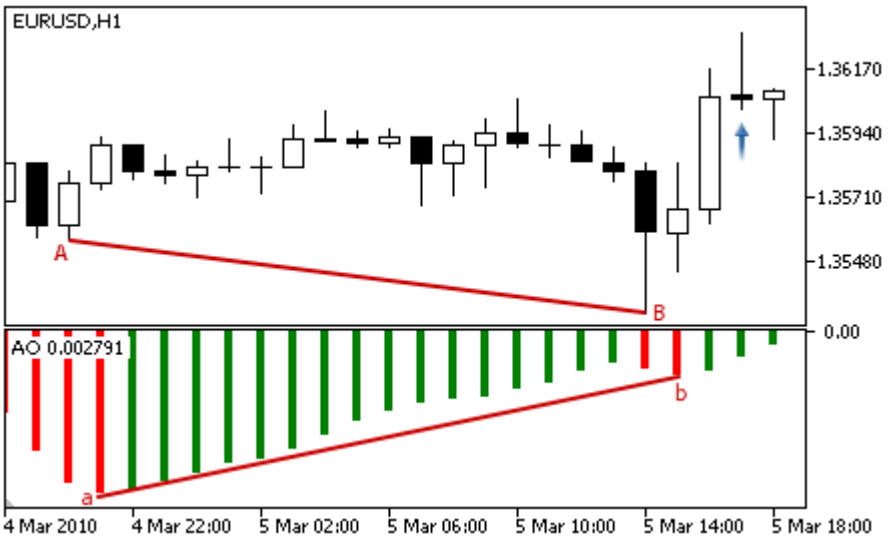
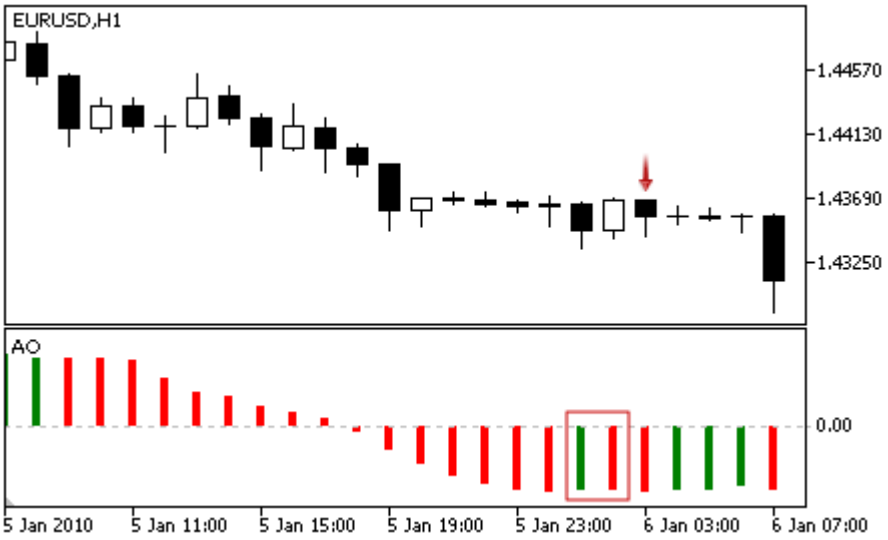
このシグナルモジュールはオーサムオシレーター指標のマーケットモデルに基づいています。モジュールから得られるシグナルに基づいての取引意思決定の仕組みは別のセクションで説明されています。

シグナル生成の条件

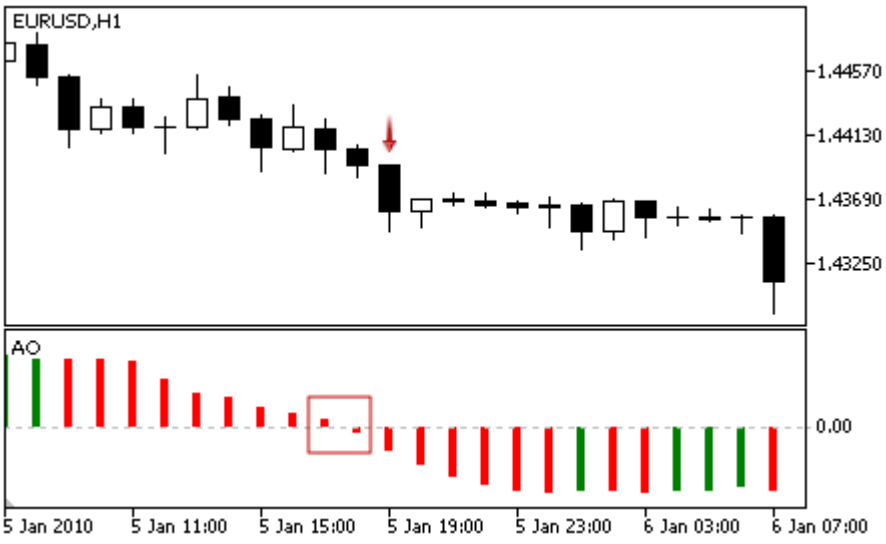
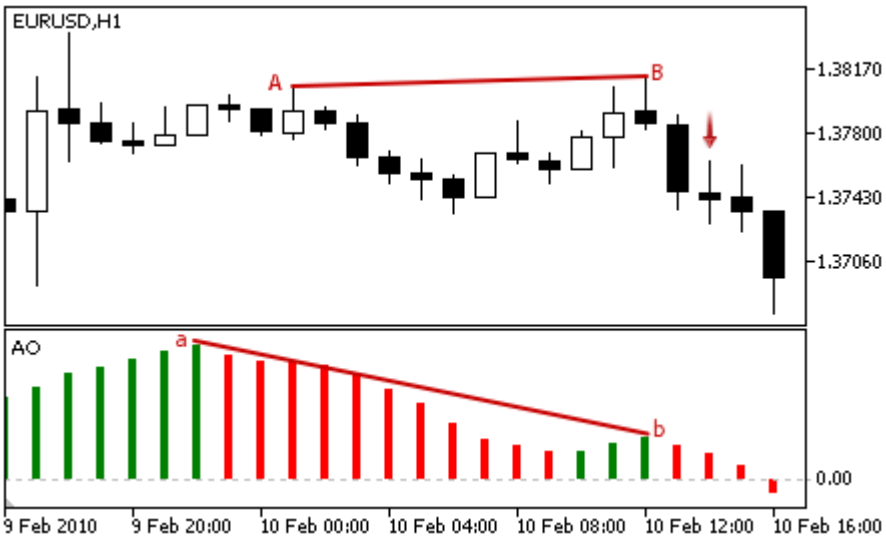
下記は、モジュールがエキスパートアドバイザーへのシグナルを受け渡す条件についてを説明します。

シグナルの種類	条件の説明
買いシグナル	• ソーサー – 分析されているバーでの指標の値が上昇、前のバーでは下降し、両方の値が正です。  • ゼロ線の交差 – 分析されているバーでの指標値が正で、1つ前のバーでは負です。  • 発散 – 指標の最初の分析底が1つ前のものよりも浅くなり、対応する価格の谷は、1つ前のものよりも深いです。更に、指標はゼロレベルを超えて上昇してはいけません。

© 2000–2025, MetaQuotes Ltd.

シグナルの種類	条件の説明
	
売りシグナル	• ソーサー — 分析されているバーでの指標の値が下降、前のバーでは上昇し、両方の値が負です。  • ゼロ線の交差 — 分析されているバーでの指標値が負で、1 つ前のバーでは正です。

© 2000–2025, MetaQuotes Ltd.

シグナルの種類	条件の説明
	 • 発散 – 指標の最初の分析された天井が1つ前のものよりも低く、対応する価格の天井は、1つ前のものよりも高いです。更に、指標はゼロレベルを超えて下降してはいけません。 
買いに異議なし	分析されているバーで指標値が増加します。
売りに異議なし	分析されているバーで指標値が減少します。

注意事項

エキスパートアドバイザーの動作モード(「Every tick」(ティックごと)または「Open prices only」(始値のみ))に応じて、分析バーは(インデックス 0 の)現在足、または(インデックス 1 の)直近に形成されたバーのいずれかです。

調整可能のパラメータ

このモジュールは、以下の調整可能なパラメータを持っています。

© 2000–2025, MetaQuotes Ltd.

パラメータ	説 明
Weight	0～1 のモジュールのシグナルの重 み

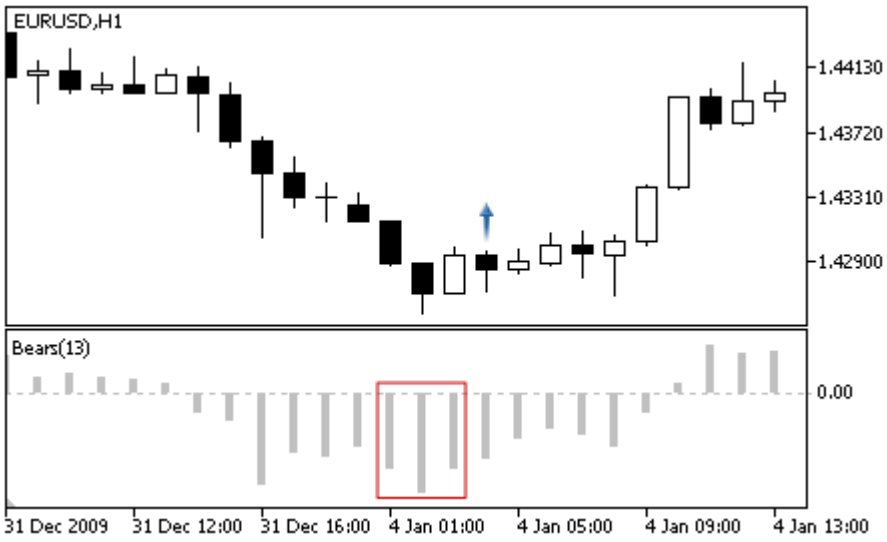
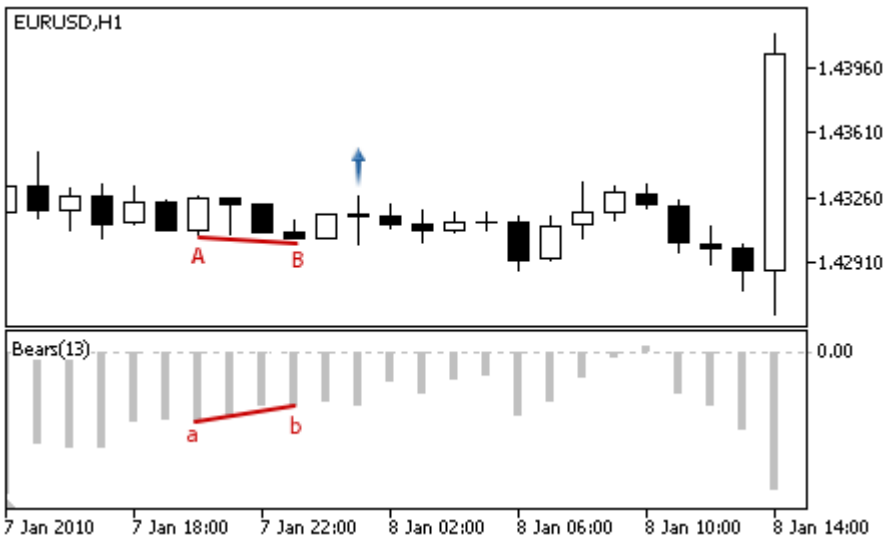
© 2000-2025, MetaQuotes Ltd.

ベアパワー(Bears Power)オシレーター指標のシグナル

このシグナルモジュールはベアパワーオシレーター指標のマーケットモデルに基づいています。モジュールから得られるシグナルに基づいての取引意思決定の仕組みは別のセクションで説明されています。

シグナル生成の条件

下記は、モジュールがエキスパートアドバイザーへのシグナルを受け渡す条件についてを説明します。

シグナルの種類	条件の説明
買いシグナル	• 反転 – オシレーターが上向きで分析されたバーでの値が負です。  • 発散 – 指標の最初の分析底が 1 つ前のものより高く、対応する価格の底は、1つ前のものより低いです。更に、指標はゼロレベルを超えて上昇してはいけません。 
売りシグナル	売りシグナルはありません。
買いに異議なし	指標値が負です。
売りに異議なし	シグナルはありません。

© 2000–2025, MetaQuotes Ltd.

注意事項

エキスパートアドバイザーの動作モード（「Every tick」（ティックごと）または「Open prices only」（始値のみ））に応じて、分析バーは（インデックス 0 の）現在足、または（インデックス 1 の）直近に形成されたバーのいずれかです。

調整可能のパラメータ

このモジュールは、以下の調整可能なパラメータを持っています。

パラメータ	説明
Weight	0～1 のモジュールのシグナルの重み
PeriodBears	指標計算の期間

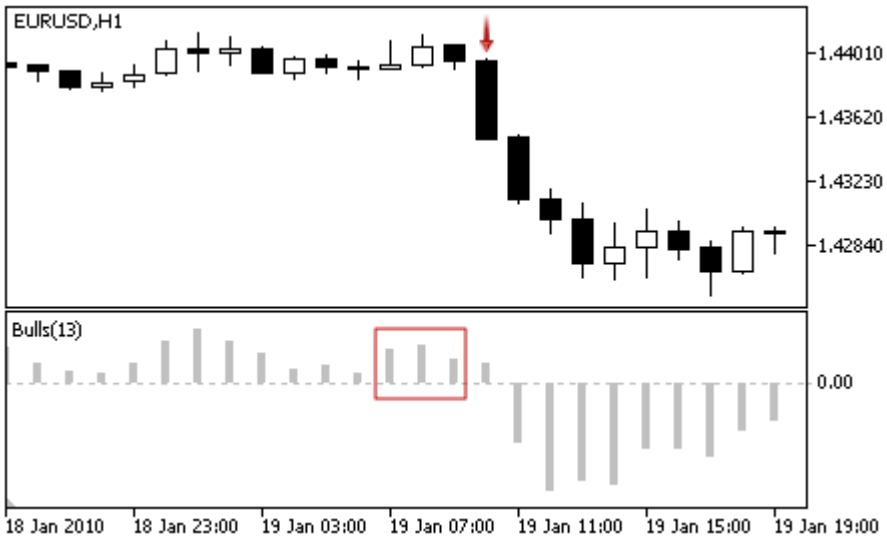
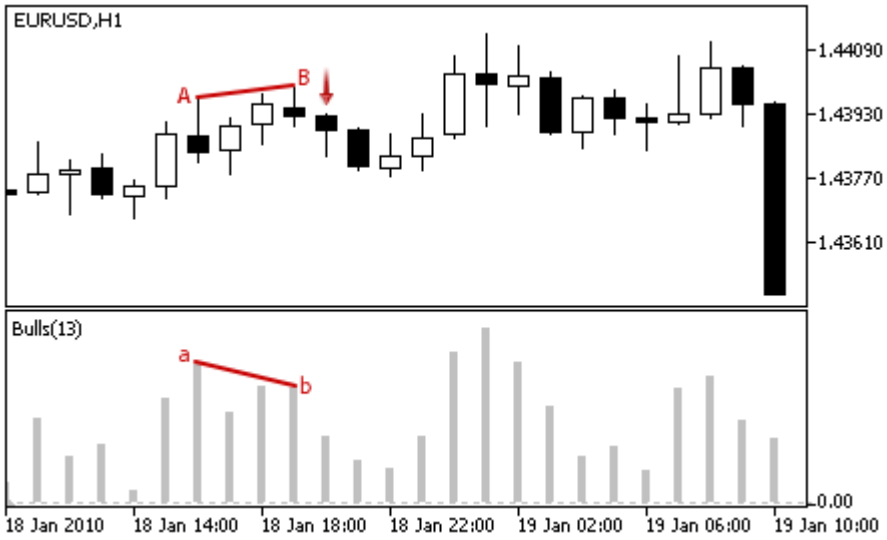
© 2000-2025, MetaQuotes Ltd.

ブルパワー(Bulls Power)オシレーター指標のシグナル

このシグナルモジュールはブルパワーオシレーター指標のマーケットモデルに基づいています。モジュールから得られるシグナルに基づいての取引意思決定の仕組みは別のセクションで説明されています。

シグナル生成の条件

下記は、モジュールがエキスパートアドバイザーへのシグナルを受け渡す条件についてを説明します。

シグナルの種類	条件の説明
買いシグナル	買いシグナルはありません。
売りシグナル	• 反転 – オシレーターが下向きで分析されたバーでの値が正です。  • 発散 – 指標の最初の分析された天井が1つ前のものよりも低く、対応する価格の天井は、1つ前のものよりも高いです。更に、指標はゼロレベルを超えて下降してはいけません。 
買いに異議なし	シグナルはありません。
売りに異議なし	指標値が正です。

© 2000–2025, MetaQuotes Ltd.

注意事項

エキスパートアドバイザーの動作モード（「Every tick」（ティックごと）または「Open prices only」（始値のみ））に応じて、分析バーは（インデックス 0 の）現在足、または（インデックス 1 の）直近に形成されたバーのいずれかです。

調整可能のパラメータ

このモジュールは、以下の調整可能なパラメータを持っています。

パラメータ	説明
Weight	0～1 のモジュールのシグナルの重み
PeriodBulls	指標計算の期間

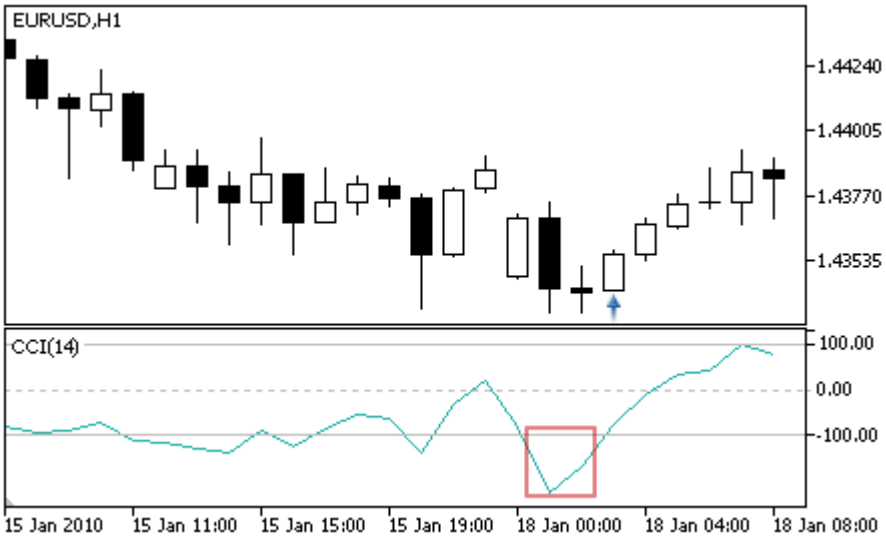
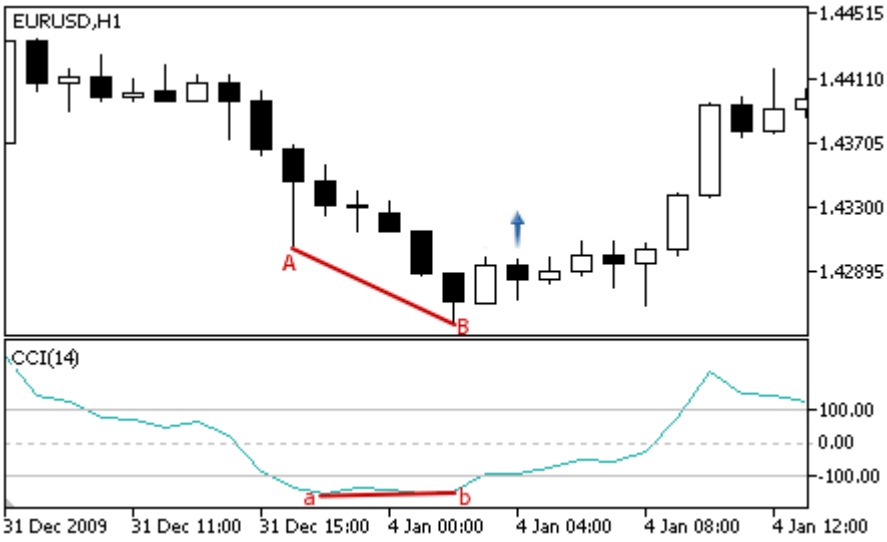
© 2000-2025, MetaQuotes Ltd.

CCI(Commodity Channel Index、コモディティチャンネルインデックス)指標のシグナル

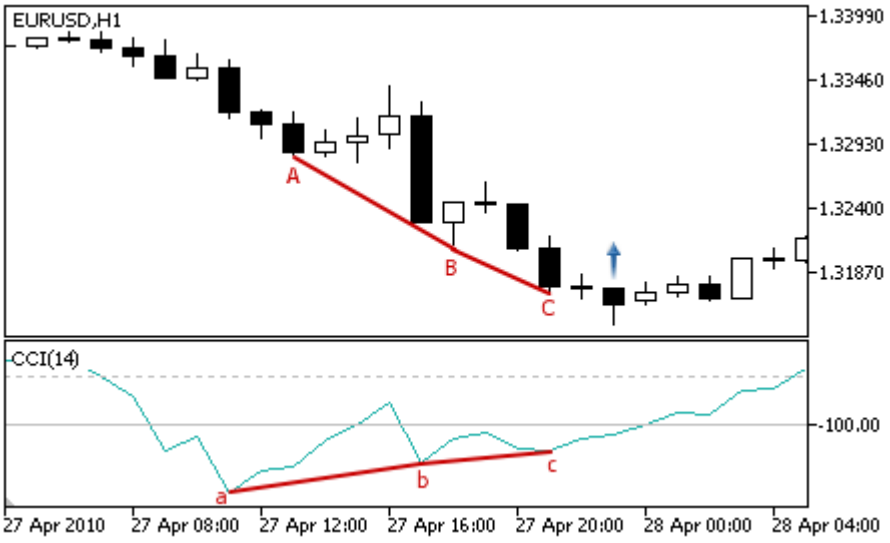
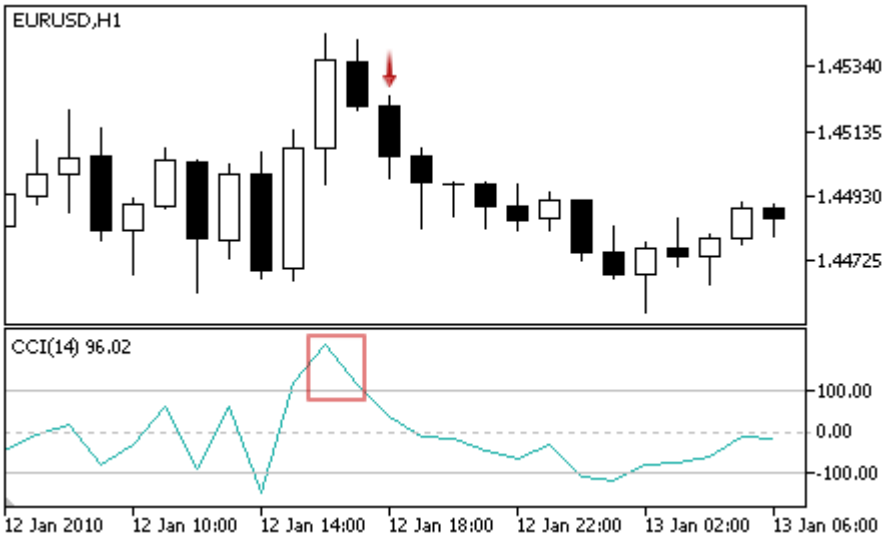
このシグナルモジュールはCCI(コモディティチャンネルインデックス)オシレーター指標のマーケットモデルに基づいています。モジュールから得られるシグナルに基づいての取引意思決定の仕組みは別のセクションで説明されています。

シグナル生成の条件

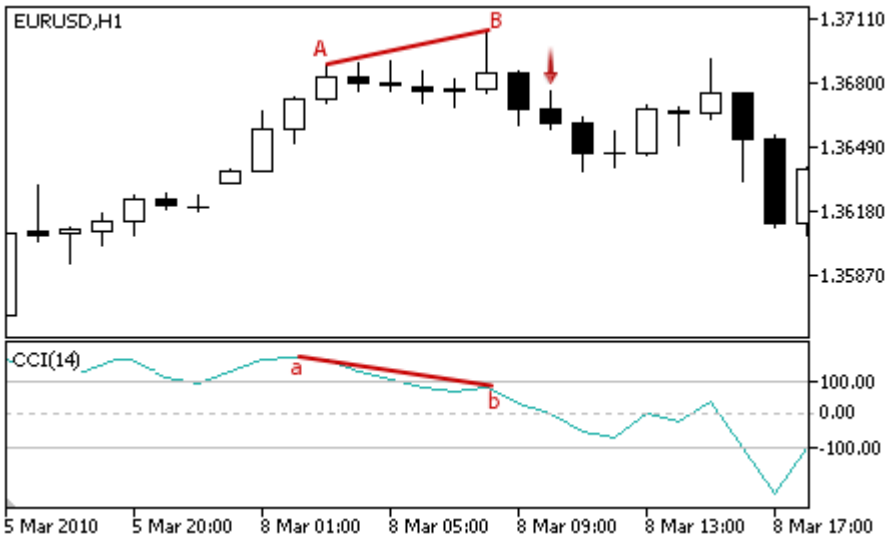
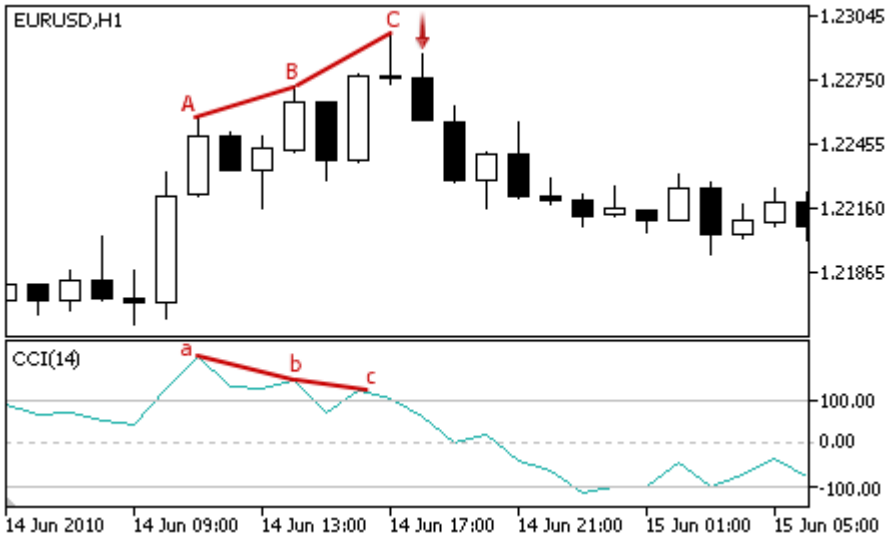
下記は、モジュールがエキスパートアドバイザーへのシグナルを受け渡す条件についてを説明します。

シグナルの種類	条件の説明
買いシグナル	• 売りすぎレベルを越した反転 – 指標が上向きで分析されたバーでの値が売りすぎレベルを越しています(初期値は -100)。  • 発散 – 指標の最初の分析底が1つ前のものより高く、対応する価格の底は、1つ前のものより低いです。 

© 2000–2025, MetaQuotes Ltd.

シグナルの種類	条件の説明
	• **二重発散** – オシレーターが3つの続いた底を形成し、それぞれが 1 つ前のものより高くなっています。価格は、3つの対応する底を形成し、それぞれが前のものよりも低いです。 
売りシグナル	• **買いすぎレベルを越した反転** – オシレーターが下向きで分析されたバーでの値が買いすぎレベルを越しています（初期値は 100）。  • **発散** – オシレーターの最初の分析された天井が1つ前のものよりも低く、対応する価格の天井は、1つ前のものよりも高いです。

© 2000–2025, MetaQuotes Ltd.

シグナルの種類	条件の説明
	 • 二重発散 – オシレーターが3つの続いた天井を形成し、それぞれが 1 つ前のものより低くなっています。価格は、3つの対応する天井を形成し、それぞれが前のものよりも高いです. 
買いに異議なし	分析されているバーでオシレーター値が増加します。
売りに異議なし	分析されているバーでオシレーター値が減少します。

注意事項

エキスパートアドバイザーの動作モード(「Every tick」(ティックごと)または「Open prices only」(始値のみ))に応じて、分析バーは(インデックス 0 の)現在足、または(インデックス 1 の)直近に形成されたバーのいずれかです。

調整可能のパラメータ

このモジュールは、以下の調整可能なパラメータを持っています。

© 2000–2025, MetaQuotes Ltd.

パラメータ	説 明
Weight	0～1 のモジュールのシグナルの重 み
PeriodCCI	指 標 計 算 の期 間
Applied	オシレーター計 算 に使 用 される価 格 列***

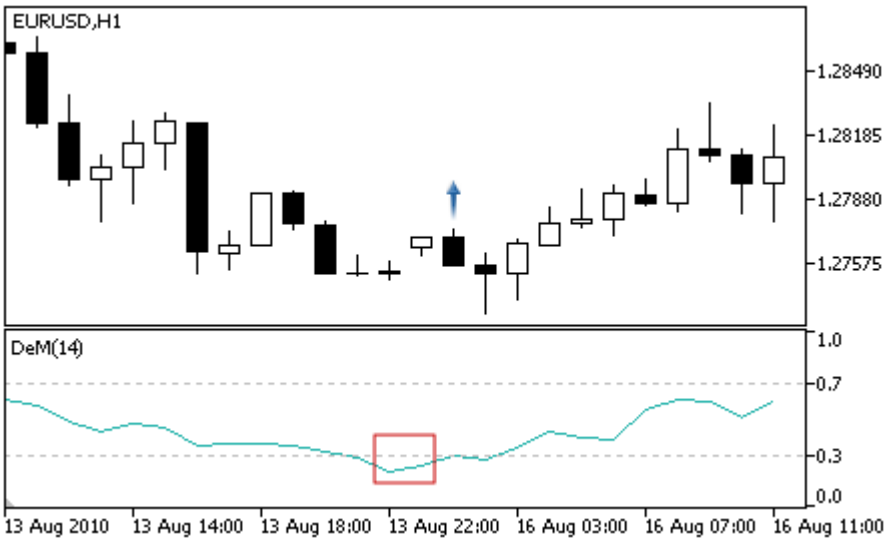
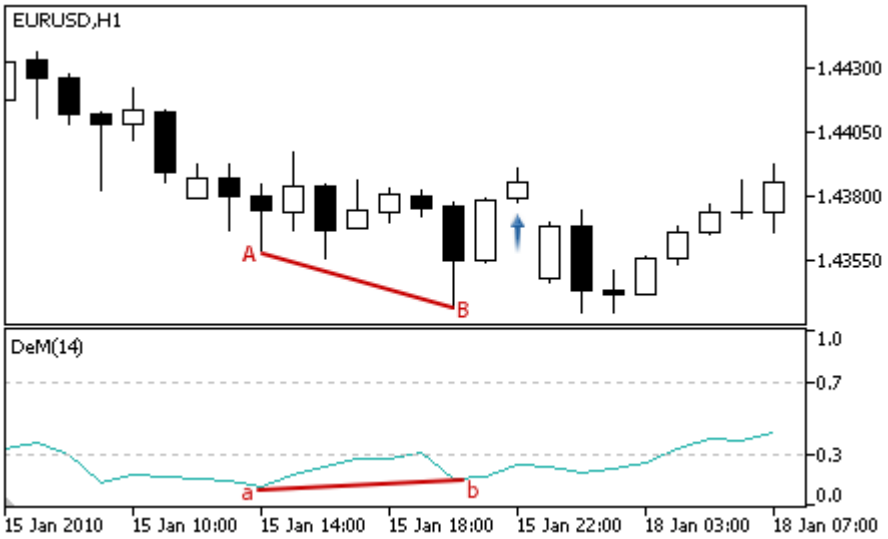
© 2000–2025, MetaQuotes Ltd.

デマーカー(DeMarker)オシレーター指標のシグナル

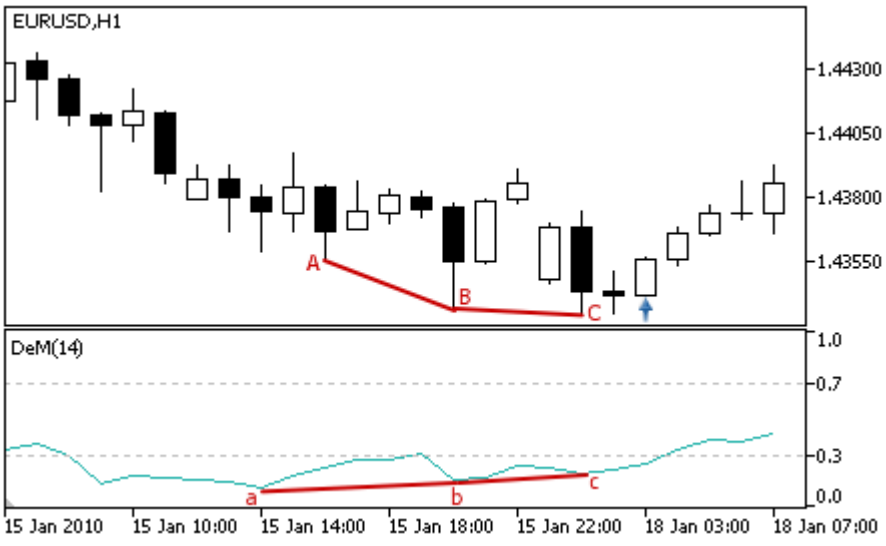
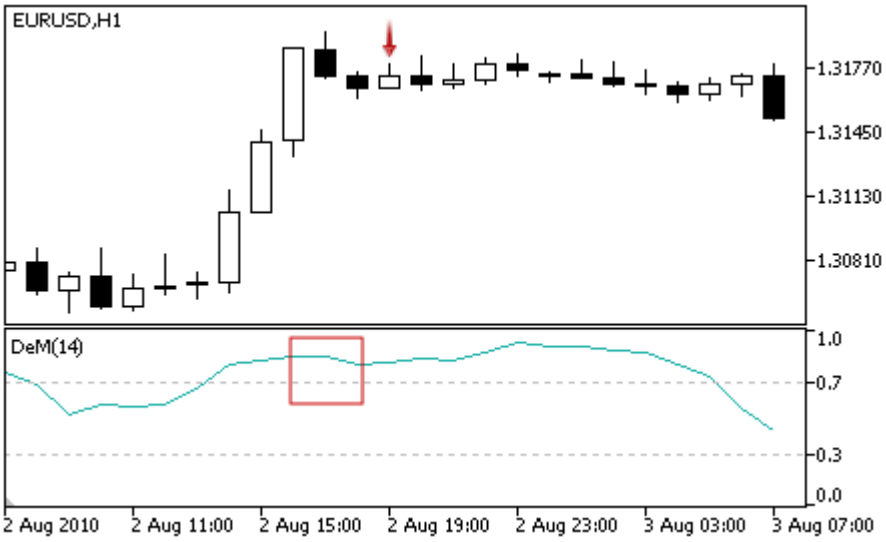
このシグナルモジュールは　デマーカーオシレーターのマーケットモデルに基づいています。モジュールから得られるシグナルに基づいての取引意思決定の仕組みは別のセクションで説明されています。

シグナル生成の条件

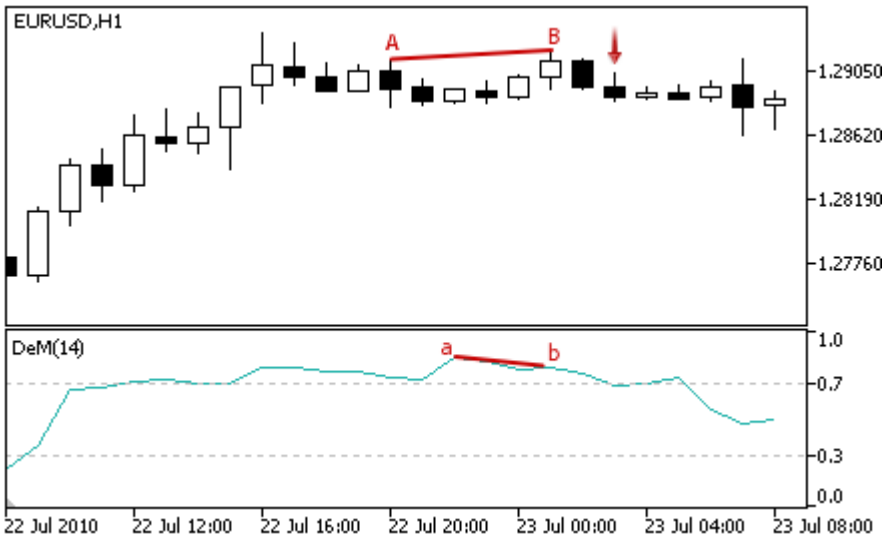
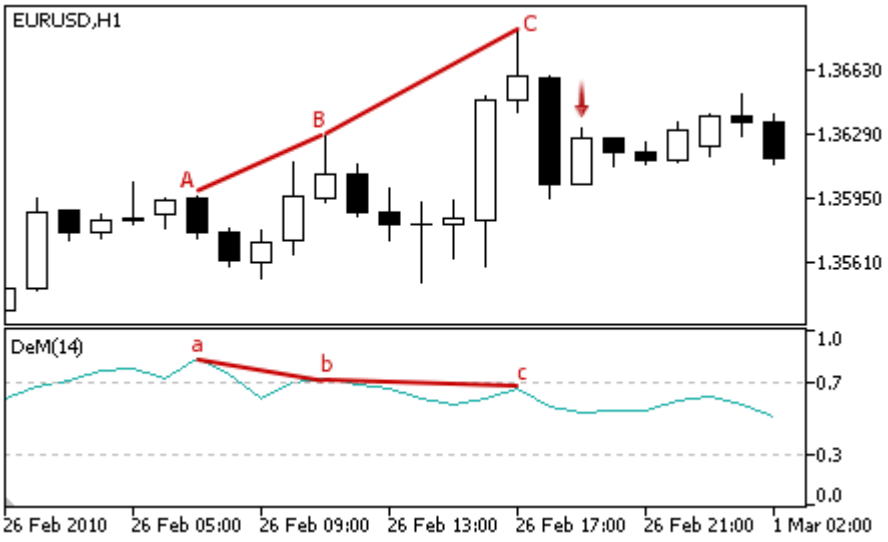
下記は、モジュールがエキスパートアドバイザーへのシグナルを受け渡す条件についてを説明します。

シグナルの種類	条件の説明
買いシグナル	• **売りすぎレベルを越した反転** – オシレーターが上向きで分析されたバーでの値が売りすぎレベルを越しています(初期値は0.3)。  • **発散** – 指標の最初の分析底が1つ前のものより高く、対応する価格の底は、1つ前のものより低いです。  • **二重発散** –オシレーターーーーが3つの続いた底を形成し、それぞれが1つ前のものより高くなっています。価格は、3つの対応する底を形成し、それぞれが前のものよりも低いです。

© 2000–2025, MetaQuotes Ltd.

シグナルの種類	条件の説明
	
売りシグナル	• 買いすぎレベルを越した反転 – オシレーターが下向きで分析されたバーでの値が買いすぎレベルを越しています（初期値は 0.7）。  • 発散 – オシレーターの最初の分析された天井が1つ前のものよりも低く、対応する価格の天井は、1つ前のものよりも高いです。

© 2000–2025, MetaQuotes Ltd.

シグナルの種類	条件の説明
	 • 二重発散 – オシレーターが3つの続いた天井を形成し、それぞれが 1 つ前のものより低くなっています。価格は、3つの対応する天井を形成し、それぞれが前のものよりも高いです. 
買いに異議なし	分析されているバーでオシレーター値が増加します。
売りに異議なし	分析されているバーでオシレーター値が減少します。

注意事項

エキスパートアドバイザーの動作モード(「Every tick」(ティックごと)または「Open prices only」(始値のみ))に応じて、分析バーは(インデックス 0 の)現在足、または(インデックス 1 の)直近に形成されたバーのいずれかです。

調整可能のパラメータ

このモジュールは、以下の調整可能なパラメータを持っています。

© 2000–2025, MetaQuotes Ltd.

パラメータ	説 明
Weight	0～1 のモジュールのシグナルの重 み
PeriodDeM	指 標 計 算 の期 間

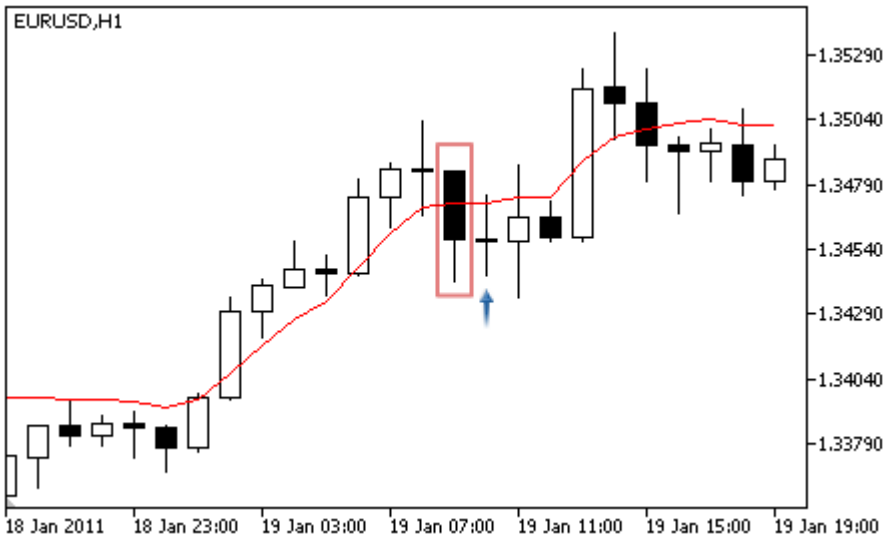
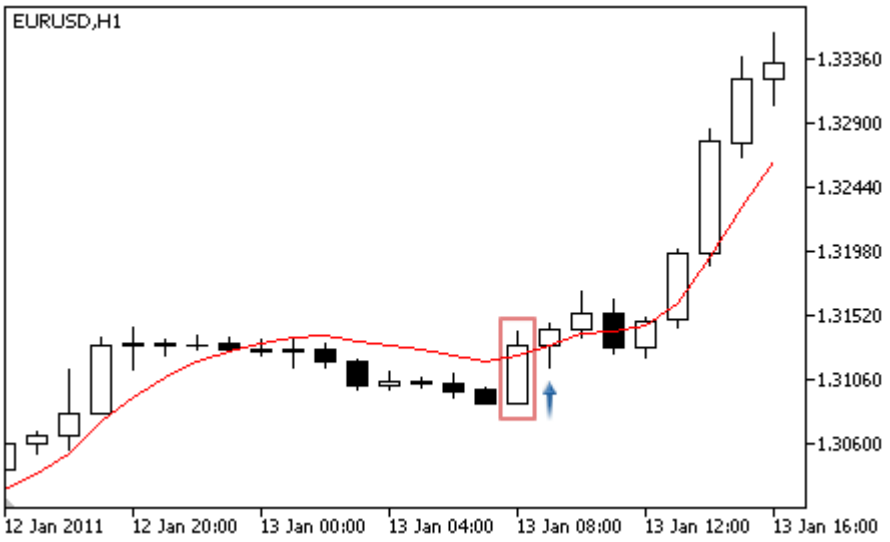
© 2000-2025, MetaQuotes Ltd.

2 重指数移動平均(Double Exponential Moving Average)指標のシグナル

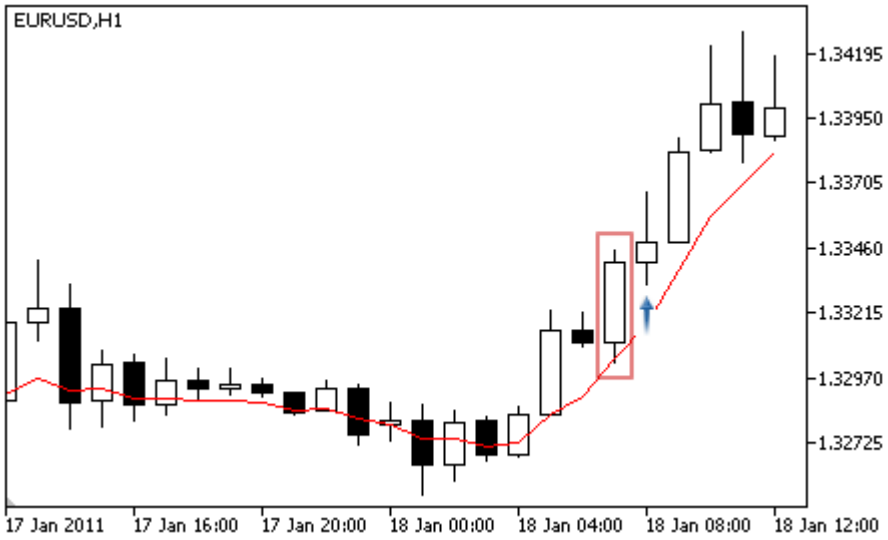
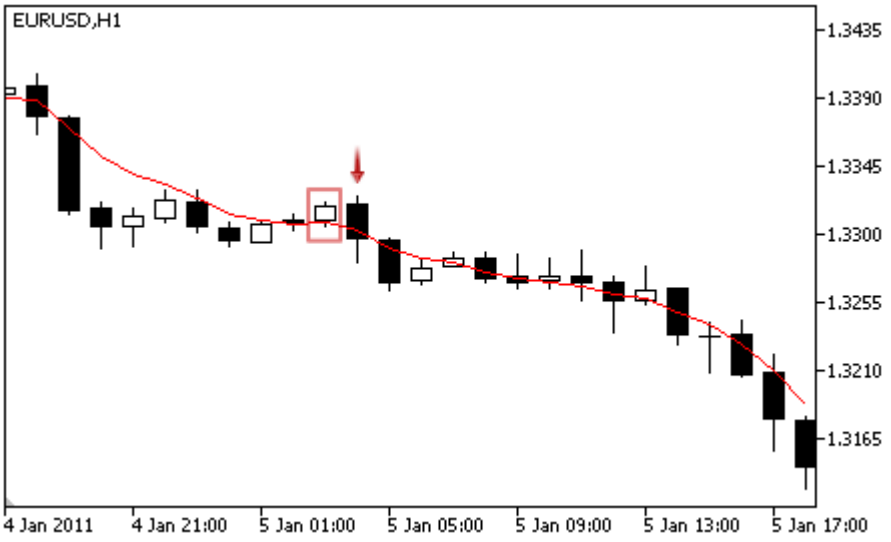
このモジュールは2 重指数移動平均指標のマーケットモデルに基づいています。モジュールから得られるシグナルに基づいての取引意思決定の仕組みは別のセクションで説明されています。

シグナル生成の条件

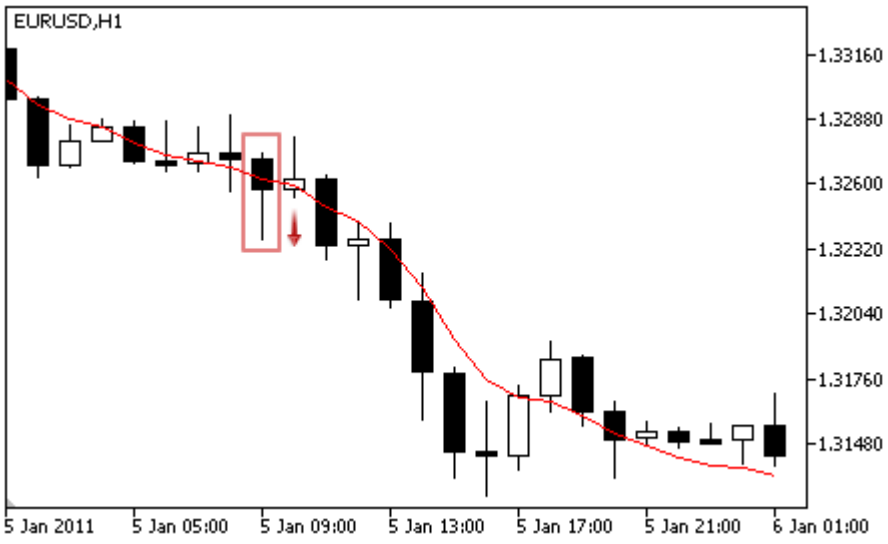
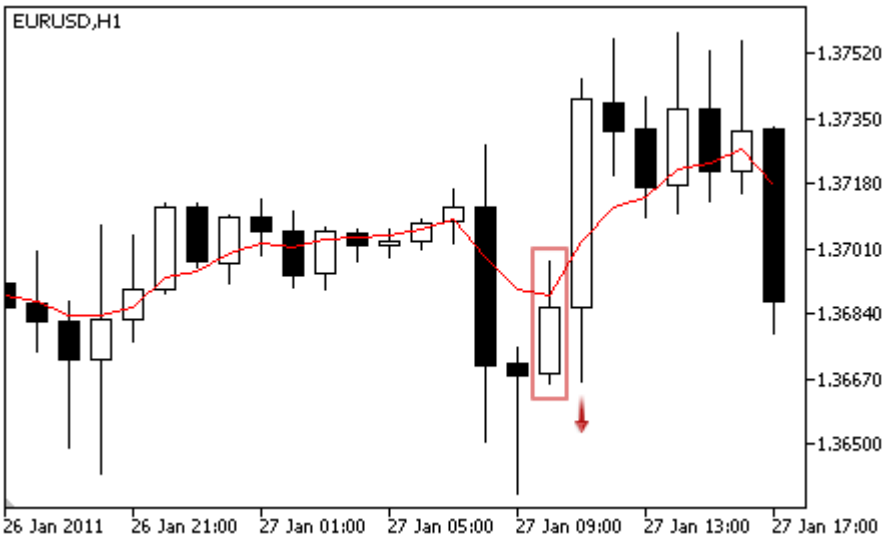
下記は、モジュールがエキスパートアドバイザーへのシグナルを受け渡す条件についてを説明します。

シグナルの種類	条件の説明
買いシグナル	• 価格が下向きに指標を交差(分析されているバーの始値が指標を上回り、終値が指標を下回る)し、指標が上昇しています(弱いシグナル)。  • 移動平均交差。 価格が上向きに指標を交差(分析されているバーの始値が指標を下回り、終値が指標を上回る)し、指標が上昇しています(強いシグナル)。 

© 2000–2025, MetaQuotes Ltd.

シグナルの種類	条件の説明
	• バーの下部の影が指標と交差（分析されているバーの始値と終値が指標を上回り、安値が指標を下回る）し、指標が上昇しています（弱いシグナル）。 
売りシグナル	• 価格が上向きに指標と交差（分析されているバーの始値が指標を下回り、終値が指標を上回る）し、指標が下降しています（弱いシグナル）。  • 移動平均交差。価格が下向きに指標を交差（分析されているバーの始値が指標を上回り、終値が指標を下回る）し、指標が下降しています（強いシグナル）。

© 2000–2025, MetaQuotes Ltd.

シグナルの種類	条件の説明
	 • バーの上部の影が指標と交差（分析されているバーの始値と終値が指標を下回り、高値が指標を上回る）し、指標が下降しています（弱いシグナル）。 
買いに異議なし	価格は指標を上回ります。
売りに異議なし	価格が指標を下回ります。

注意事項

エキスパートアドバイザーの動作モード（「Every tick」（ティックごと）または「Open prices only」（始値のみ））に応じて、分析バーは（インデックス 0 の）現在足、または（インデックス 1 の）直近に形成されたバーのいずれかです。

調整可能のパラメータ

このモジュールは、以下の調整可能なパラメータを持っています。

© 2000–2025, MetaQuotes Ltd.

パラメータ	説明
Weight	0～1 のモジュールのシグナルの重み
PeriodMA	指標の平均化の期間
Shift	時間軸に沿っての指標のシフト（バー単位）
Method	平均化の方法
Applied	指標計算に使用される価格列***

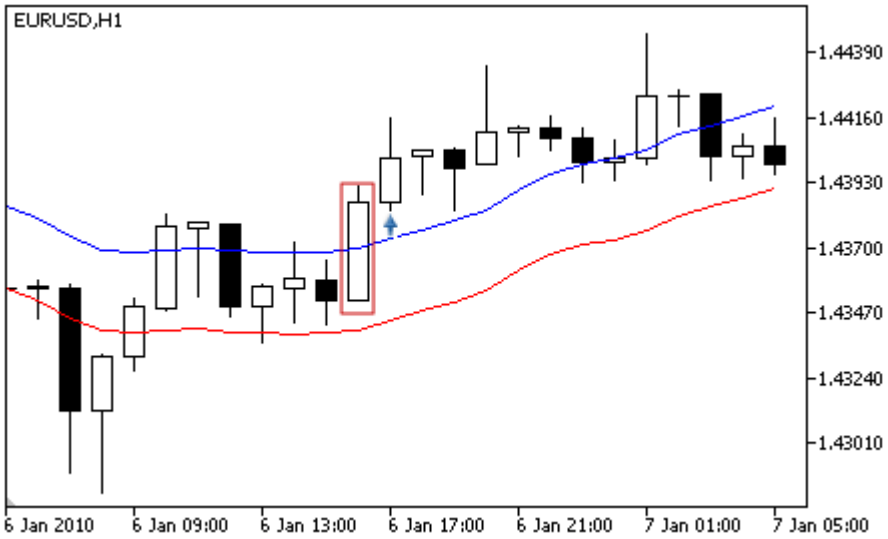
© 2000–2025, MetaQuotes Ltd.

エンベロープ(Envelopes)指標のシグナル

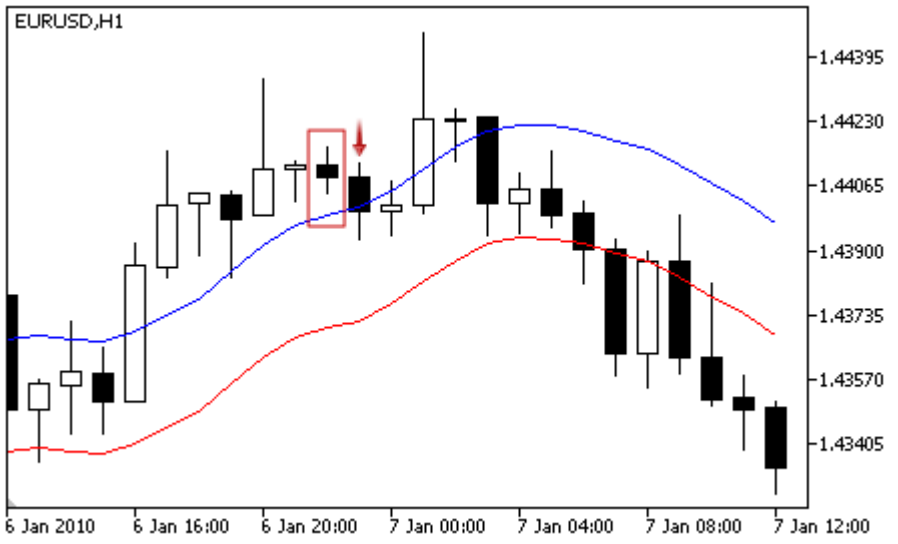
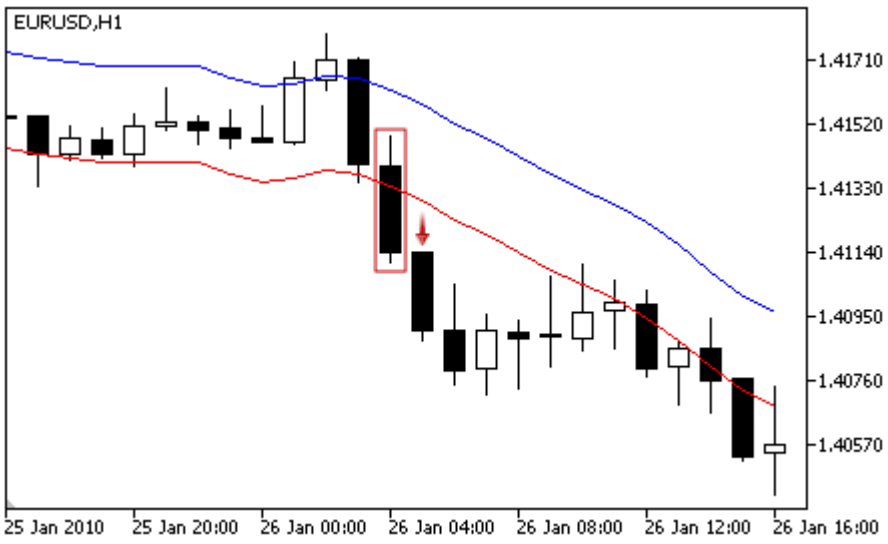
このシグナルモジュールはEnvelopes(エンベロープ)指標のマーケットモデルに基づいています。モジュールから得られるシグナルに基づいての取引意思決定の仕組みは別のセクションで説明されています。

シグナル生成の条件

下記は、モジュールがエキスパートアドバイザーへのシグナルを受け渡す条件についてを説明します。

シグナルの種類	条件の説明
買いシグナル	• 価格は分析されているバーで指標の下の線の近くにあります。  • 分析されているバーでは、価格が指標の上部の線を交差します。 
売りシグナル	• 分析されているバーでは、価格が指標の上部の線を交差します。

© 2000–2025, MetaQuotes Ltd.

シグナルの種類	条件の説明
	 • 分析されているバーでは、価格が指標の下部の線を交差します。 
買いに異議なし	シグナルはありません。
売りに異議なし	シグナルはありません。

注意事項

エキスパートアドバイザーの動作モード（「Every tick」（ティックごと）または「Open prices only」（始値のみ））に応じて、分析バーは（インデックス 0 の）現在足、または（インデックス 1 の）直近に形成されたバーのいずれかです。

調整可能のパラメータ

このモジュールは、以下の調整可能なパラメータを持っています。

© 2000–2025, MetaQuotes Ltd.

パラメータ	説明
Weight	0～1 のモジュールのシグナルの重み
PeriodMA	指標計算の期間
Shift	時間軸に沿っての指標のシフト（バー単位）
Method	平均化の方法
Applied	指標計算に使用される価格列***
Deviation	中心線（MA）からのエンベロープの境界の百分率での偏差

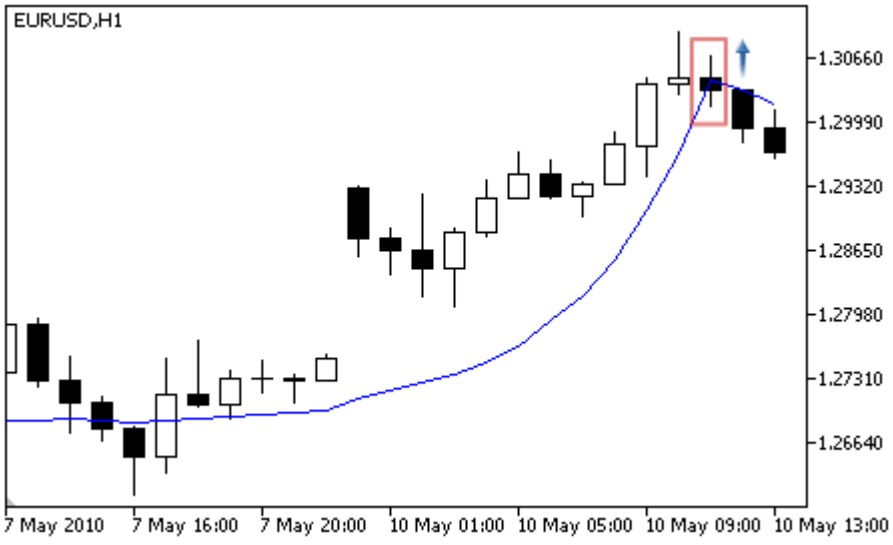
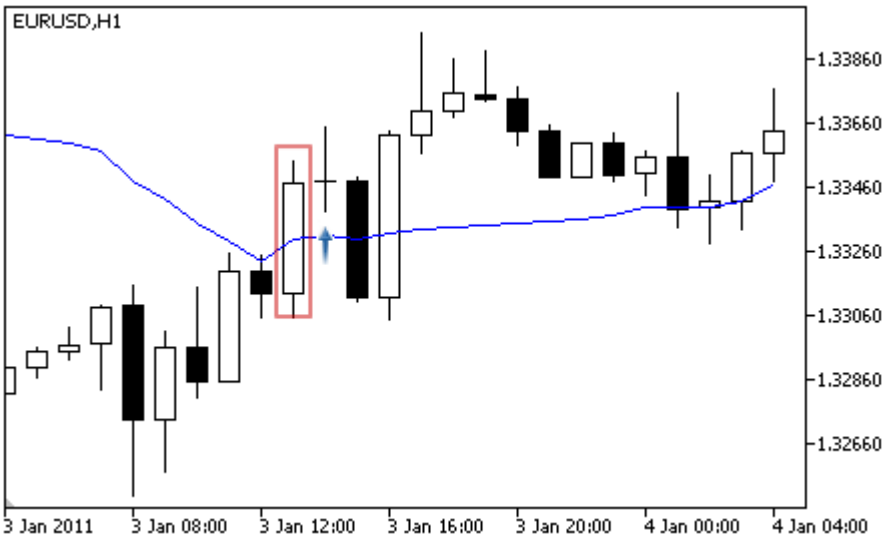
© 2000-2025, MetaQuotes Ltd.

フラクタル適応型移動平均（Fractal Adaptive Moving Average）指標のシグナル

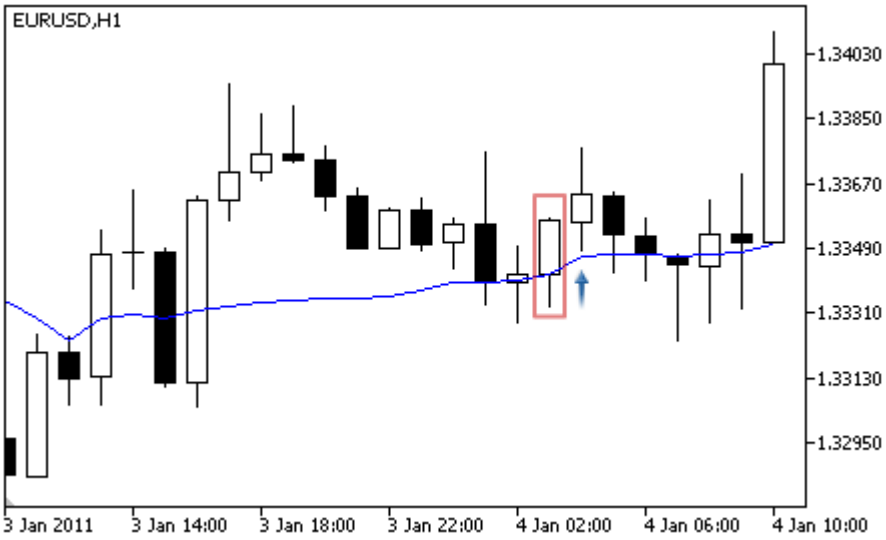
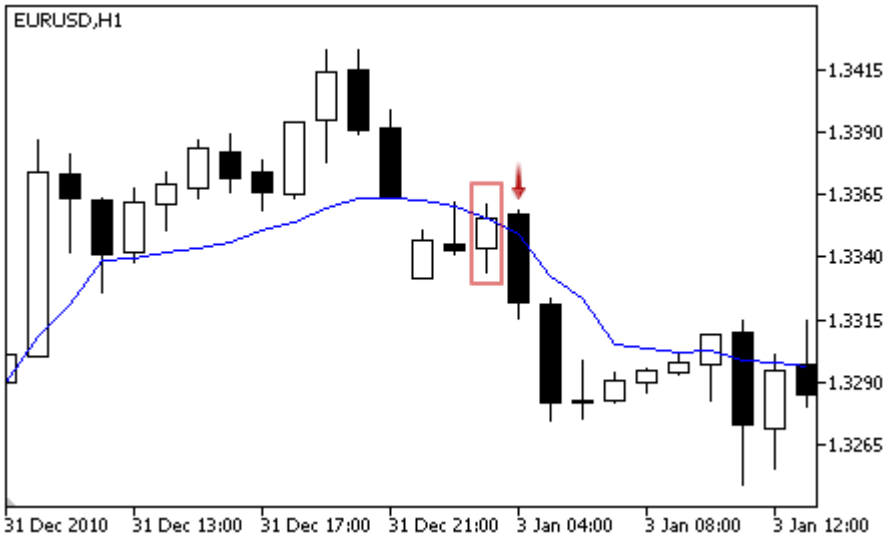
このシグナルモジュールはフラクタル適応型移動平均指標のマーケットモデルに基づいています。モジュールから得られるシグナルに基づいての取引意思決定の仕組みは別のセクションで説明されています。

シグナル生成の条件

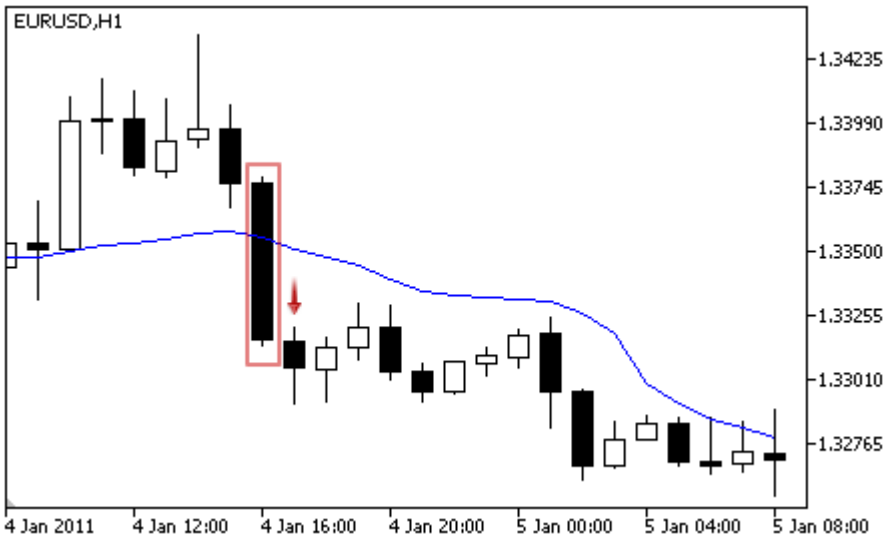
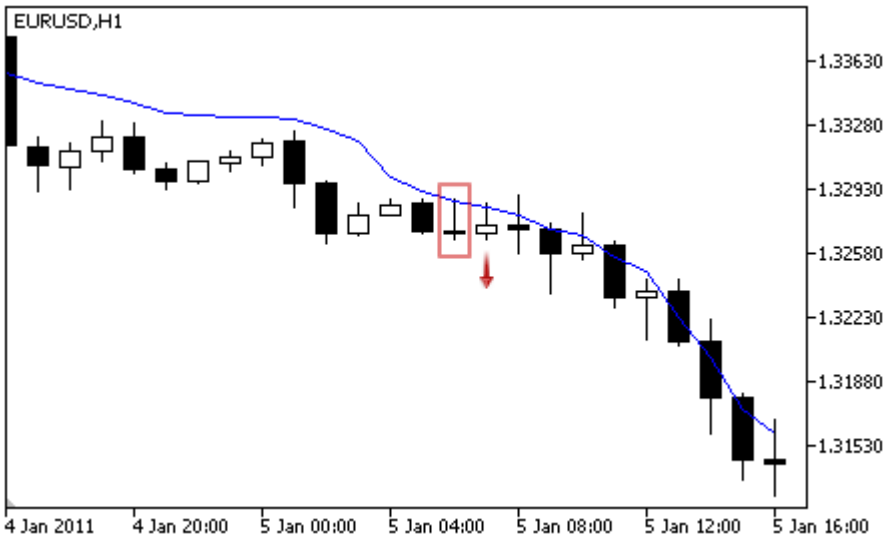
下記は、モジュールがエキスパートアドバイザーへのシグナルを受け渡す条件についてを説明します。

シグナルの種類	条件の説明
買いシグナル	• 価格が下向きに指標を交差（分析されているバーの始値が指標を上回り、終値が指標を下回る）し、指標が上昇しています（弱いシグナル）。  • 移動平均交差。 価格が上向きに指標を交差（分析されているバーの始値が指標を下回り、終値が指標を上回る）し、指標が上昇しています（強いシグナル）。 

© 2000–2025, MetaQuotes Ltd.

シグナルの種類	条件の説明
	• バーの下部の影が指標と交差（分析されているバーの始値と終値が指標を上回り、安値が指標を下回る）し、指標が上昇しています（弱いシグナル）。 
売りシグナル	• 価格が上向きに指標と交差（分析されているバーの始値が指標を下回り、終値が指標を上回る）し、指標が下降しています（弱いシグナル）。  • 移動平均交差。価格が下向きに指標を交差（分析されているバーの始値が指標を上回り、終値が指標を下回る）し、指標が下降しています（強いシグナル）。

© 2000–2025, MetaQuotes Ltd.

シグナルの種類	条件の説明
	 • バーの上部の影が指標と交差(分析されているバーの始値と終値が指標を下回り、高値が指標を上回る)し、指標が下降しています(弱いシグナル)。 
買いに異議なし	価格が指標を上回ります。
売りに異議なし	価格が指標を下回ります。

注意事項

エキスパートアドバイザーの動作モード(「Every tick」(ティックごと)または「Open prices only」(始値のみ))に応じて、分析バーは(インデックス 0 の)現在足、または(インデックス 1 の)直近に形成されたバーのいずれかです。

調整可能のパラメータ

このモジュールは、以下の調整可能なパラメータを持っています。

© 2000-2025, MetaQuotes Ltd.

パラメータ	説明
Weight	0～1 のモジュールのシグナルの重み
PeriodMA	指標の平均化の期間
Shift	時間軸に沿っての指標のシフト（バー単位）
Method	平均化の方法
Applied	指標計算に使用される価格列***

© 2000–2025, MetaQuotes Ltd.

日中時間フィルタのシグナル

このモジュールは、市場モデルの効率が時間的に変化するという仮定に基づいています。このモジュールを使用すると、他のモジュールから受信したシグナルを曜日と時間でフィルタすることが出来ます。これで、不利な時間帯を遮断によって生成されたシグナルの品質を増加することができします。モジュールから得られるシグナルに基づいての取引意思決定の仕組みは別のセクションで説明されています。

シグナル生成の条件

下記は、モジュールがエキスパートアドバイザーへのシグナルを受け渡す条件についてを説明します。

シグナルの種類	条件の説明
買いシグナル	シグナルはありません。
売りシグナル	シグナルはありません。
買いに異議なし	現在日付と時刻が指定されたパラメータを満たします。
売りに異議なし	現在日付と時刻が指定されたパラメータを満たします。

調整可能のパラメータ

このモジュールは、以下の調整可能なパラメータを持っています。

パラメータ	説明
Weight	0～1 のモジュールのシグナルの重み
GoodHourOfDay	一日で売買シグナルが有効になる時間（0~23）値が -1 の場合、シグナルは一日を通して有効です。
BadHoursOfDay	ビットフィールド。このフィールドの各ビットが時間に対応します。（0 ビット – 0 時、... 23 ビット – 23 時）。ビット値が 0 の場合、売買シグナルは対応する時間で有効にされます。ビット値が 1 の場合、売買シグナルは対応する時間で無効にされます。指定された数は、2進数として表されるビットマスクとして使用されます。 無効にされた時間は有効なものよりも高い優先度を持っています。
GoodDayOfWeek	売買シグナルが有効な曜日（0~6 で6が日曜日）。値が -1 の場合、シグナルは一週間を通して有効です。
BadDaysOfWeek	ビットフィールド。このフィールドの各ビットが曜日に対応します。（0 ビット – 日曜、... 23 ビット – 土曜）。ビット値が 0 の場合、売買シグナルは対応する曜日で有効にされます。ビット値が 1 の場合、売買シグナルは対応する曜日で無効にされます。指定された数は、2進数として表されるビットマスクとして使用されます。 無効にされた曜日は有効なものよりも高い優先度を持っています。

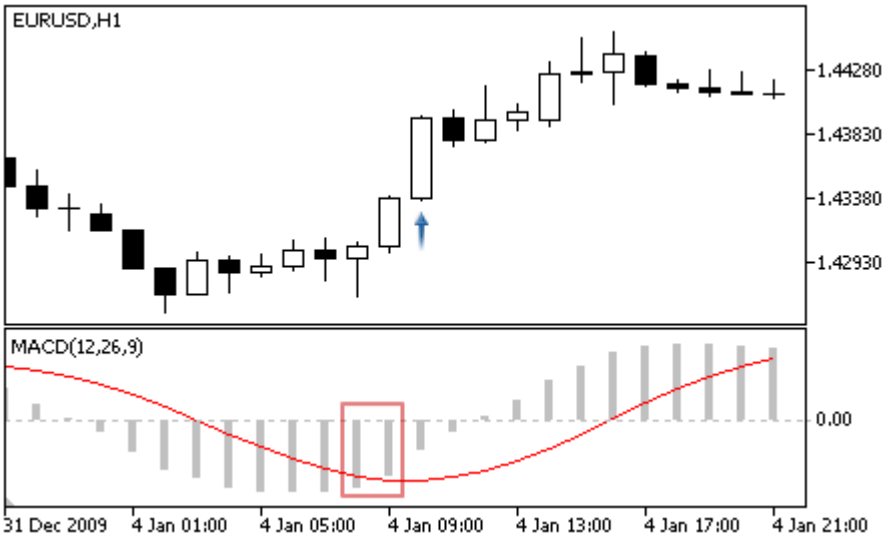
© 2000–2025, MetaQuotes Ltd.

MACD オシレーターのシグナル

このシグナルモジュールは MACDオシレーターのマーケット モデルに基づいています。モジュールから得られるシグナルに基づいての取引意思決定の仕組みは別のセクションで説明されています。

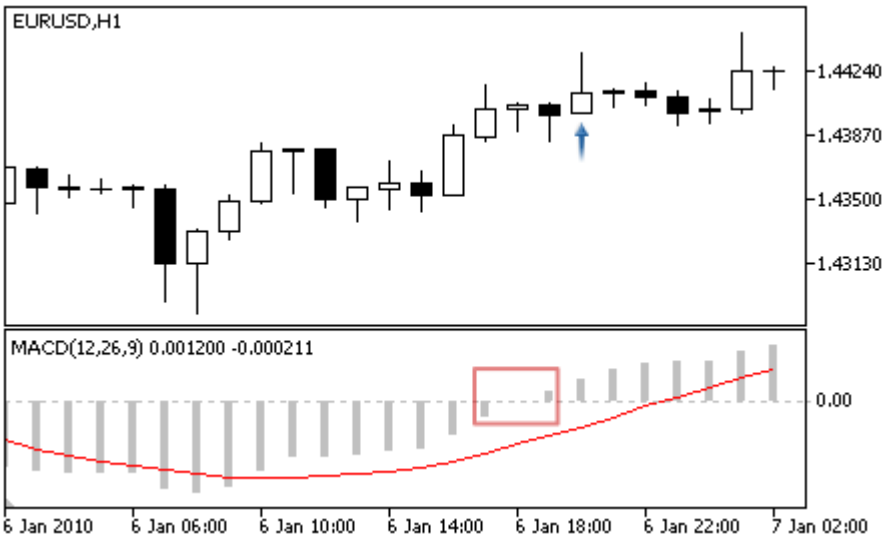
シグナル生成の条件

下記は、モジュールがエキスパート アドバイザーへのシグナルを受け渡す条件についてを説明します。

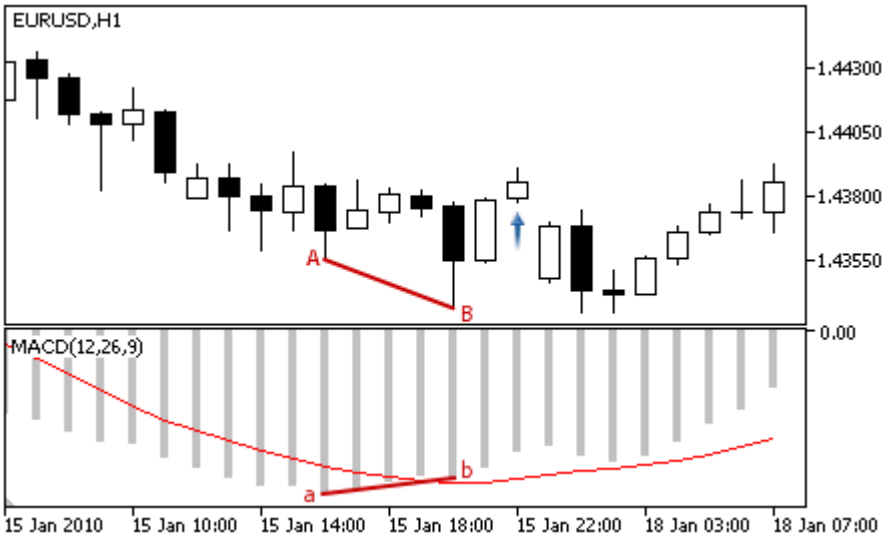
シグナルの種類	条件の説明
買いシグナル	• **反転** — オシレーターが上向きです（オシレーターは分析されているバーで上昇し 1 つ前のバーで下降します。）。  • **中心線とシグナル線の交差** — 分析されているバーでは中心線がシグナル線を上回り、1 つ前のバーではシグナル線を下回ります。  • **ゼロレベルの交差** — 分析されているバーでは中心線がゼロレベルを上回り、1 つ前のバーでは下回ります。

© 2000–2025, MetaQuotes Ltd.

シグナルの種類	条件の説明

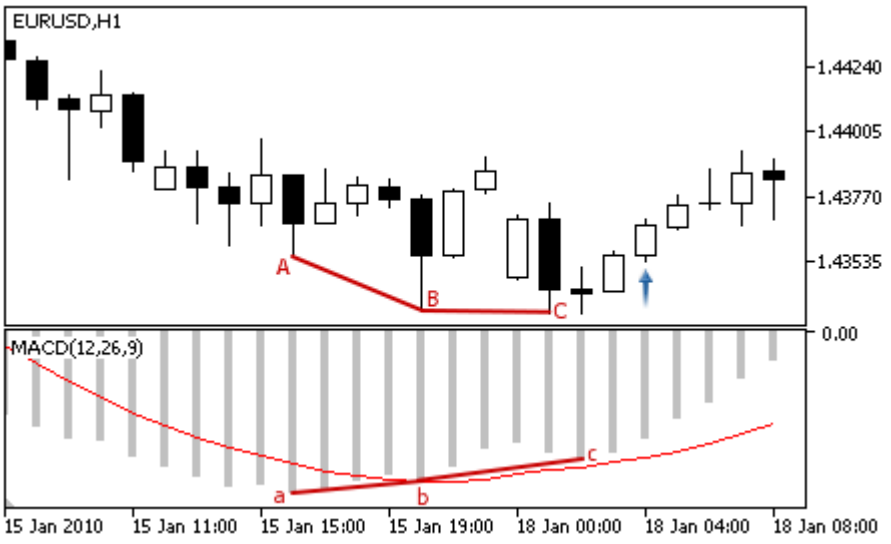
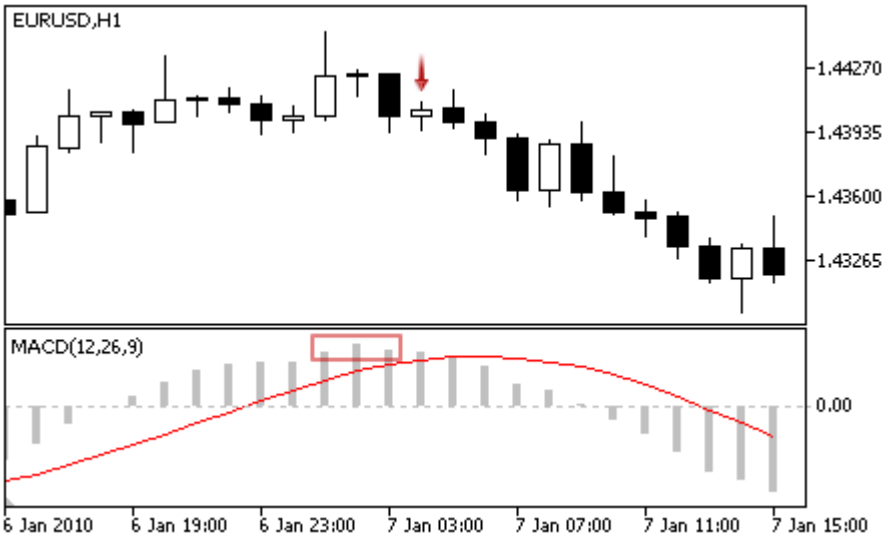


- 発散 – 指標の最初の分析底が 1 つ前のものより高く、対応する価格の底は、1つ前のものより低いです。

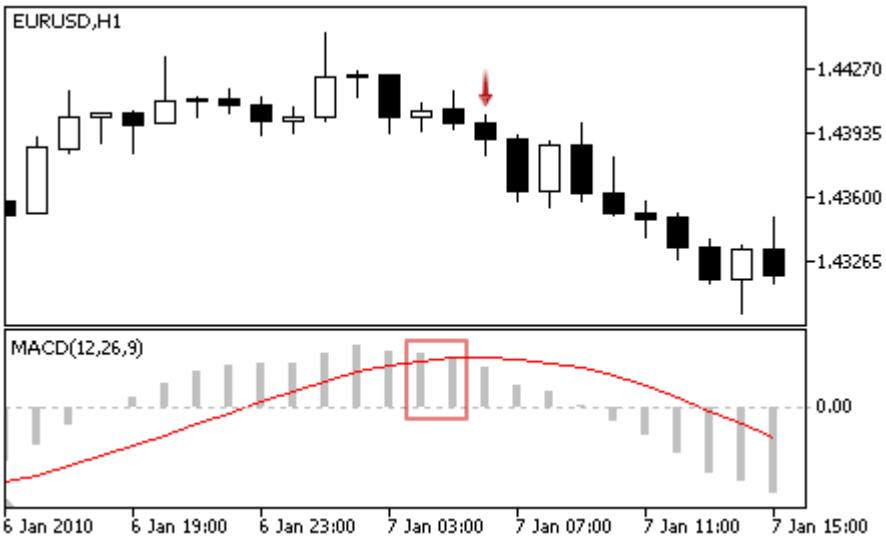
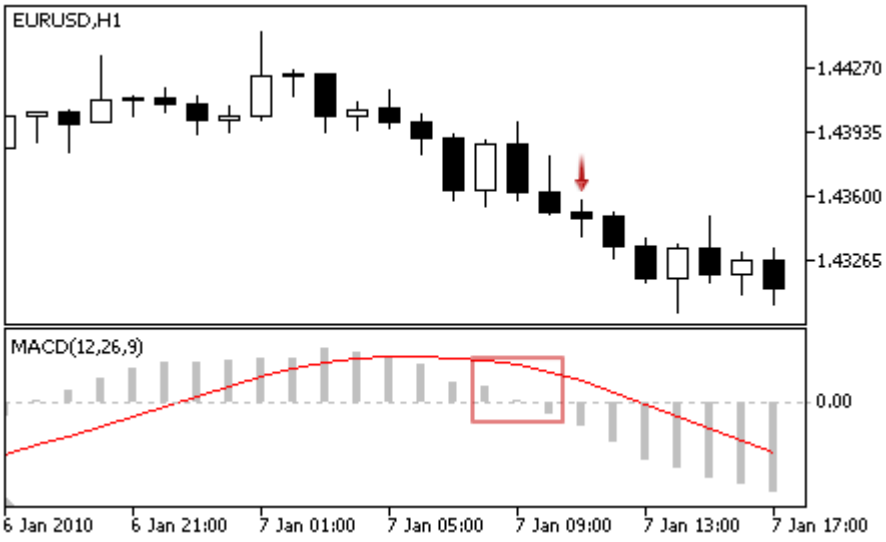


- 二重発散 –オシレーターーーーが3つの続いた底を形成し、それぞれが 1 つ前のものより高くなっています。価格は、3つの対応する底を形成し、それぞれが前のものよりも低いです。

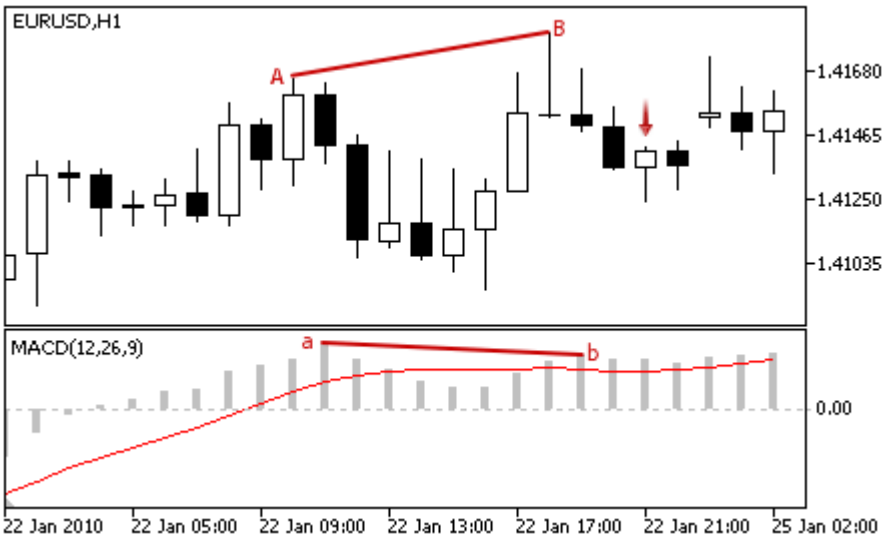
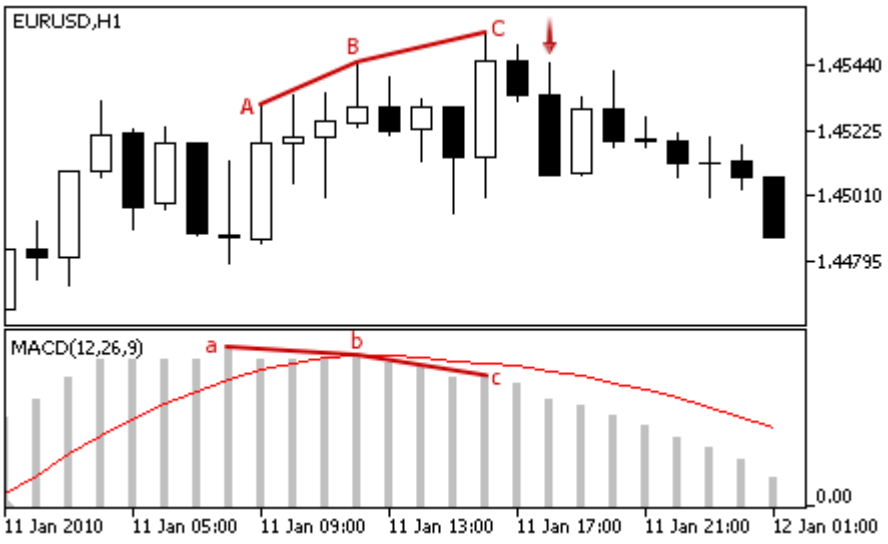
© 2000–2025, MetaQuotes Ltd.

シグナルの種類	条件の説明
	
売りシグナル	• **反転** — オシレーターが下向きです（オシレーターは分析されているバーで下降し 1 つ前のバーで上昇します。）  • **中心線とシグナル線の交差** — 分析されているバーでは中心線がシグナル線を下回り、1 つ前のバーではシグナル線を上回ります。

© 2000–2025, MetaQuotes Ltd.

シグナルの種類	条件の説明
	 • **ゼロレベルの交差** – 分析されているバーでは中心線がゼロレベルを下回り、1つ前のバーでは上回ります。  • **発散** – オシレーターの最初の分析された天井が1つ前のものよりも低く、対応する価格の天井は、1つ前のものよりも高いです。

© 2000–2025, MetaQuotes Ltd.

シグナルの種類	条件の説明
	 • 二重発散 – オシレーターが3つの続いた天井を形成し、それぞれが 1 つ前のものより低くなっています。価格は、3つの対応する天井を形成し、それぞれが前のものよりも高いです. 
買いに異議なし	分析されているバーでオシレーター値が増加します。
売りに異議なし	分析されているバーでオシレーター値が減少します。

注意事項

エキスパートアドバイザーの動作モード(「Every tick」(ティックごと)または「Open prices only」(始値のみ))に応じて、分析バーは(インデックス 0 の)現在足、または(インデックス 1 の)直近に形成されたバーのいずれかです。

調整可能のパラメータ

このモジュールは、以下の調整可能なパラメータを持っています。

© 2000–2025, MetaQuotes Ltd.

パラメータ	説明
Weight	0～1 のモジュールのシグナルの重み
PeriodFast	高速 EMA 期間の計算
PeriodSlow	低速 EMA 期間の計算
PeriodSignal	平滑化期間
Applied	オシレーター計算に使用される価格列***

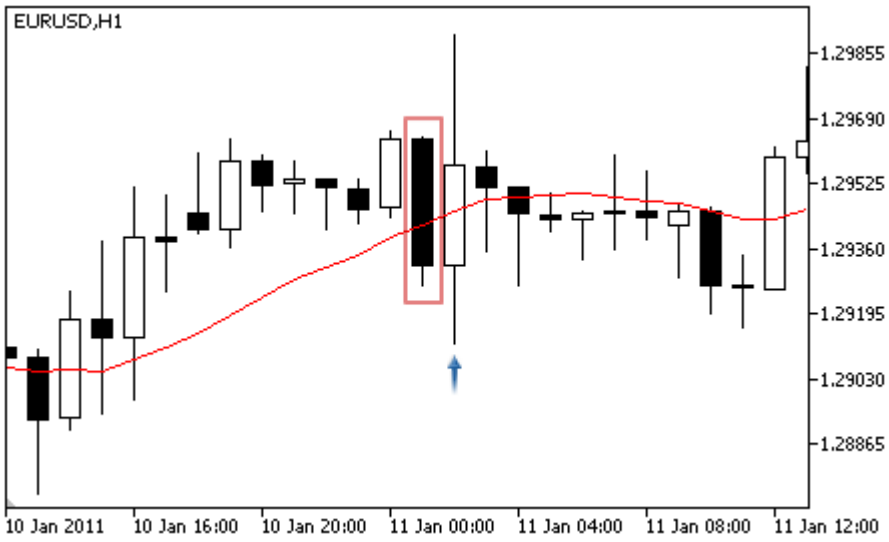
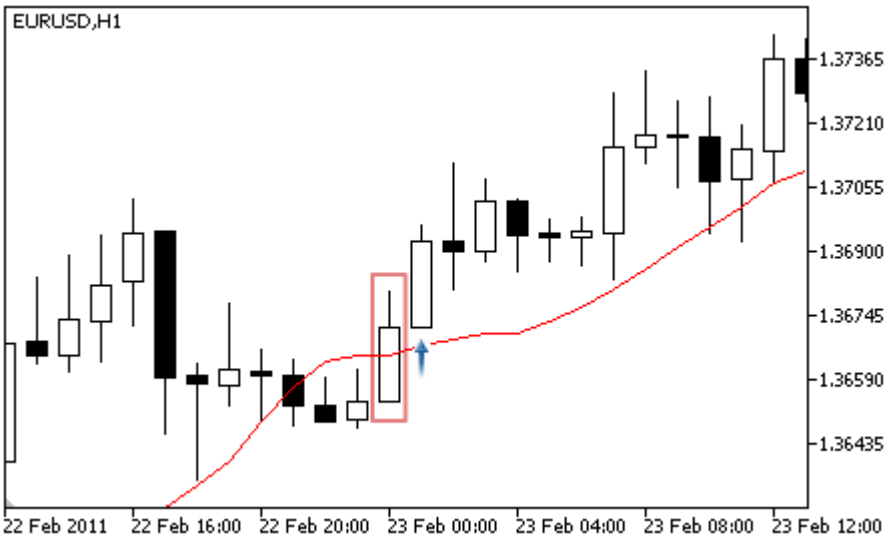
© 2000–2025, MetaQuotes Ltd.

移動平均（Moving Average）指標のシグナル

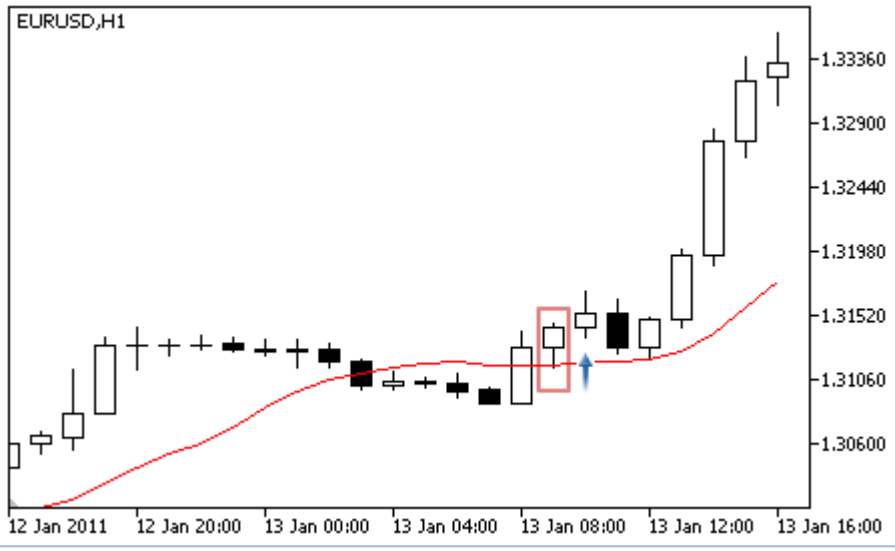
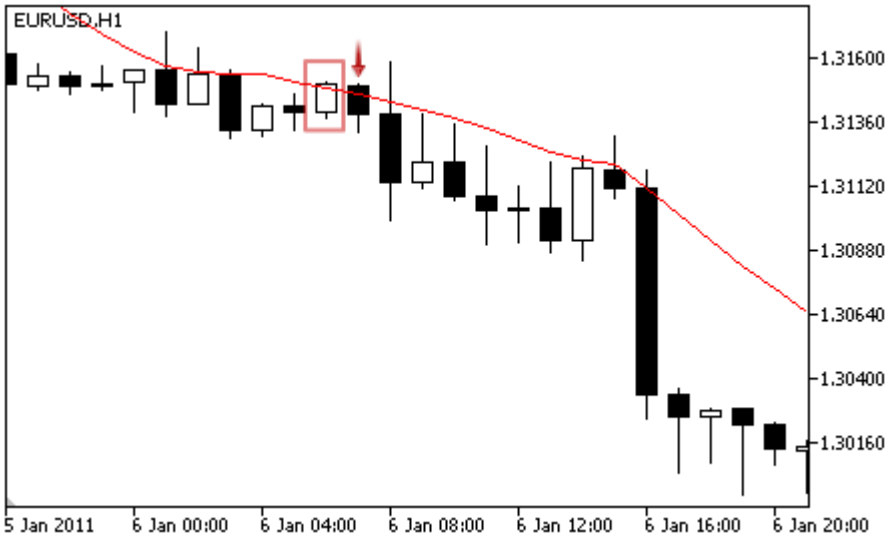
このシグナルモジュールは移動平均指標のマーケットモデルに基づいています。モジュールから得られるシグナルに基づいての取引意思決定の仕組みは別のセクションで説明されています。

シグナル生成の条件

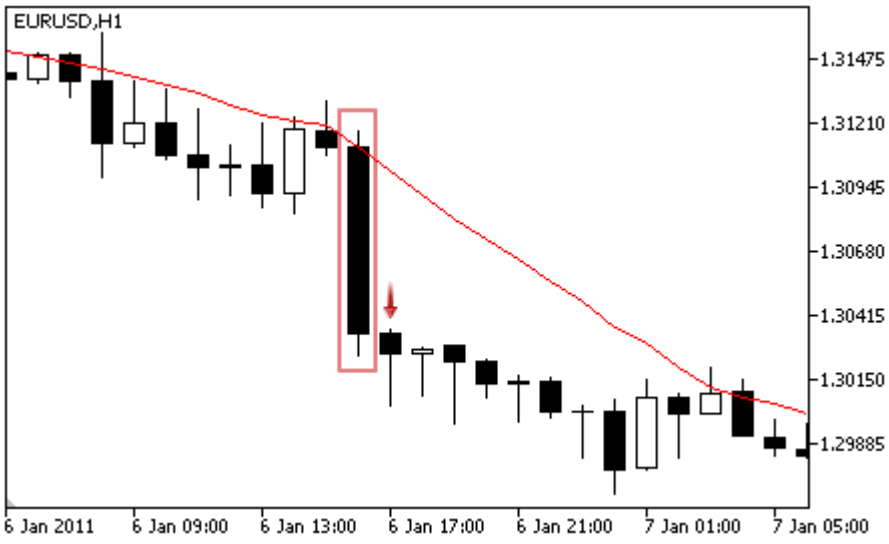
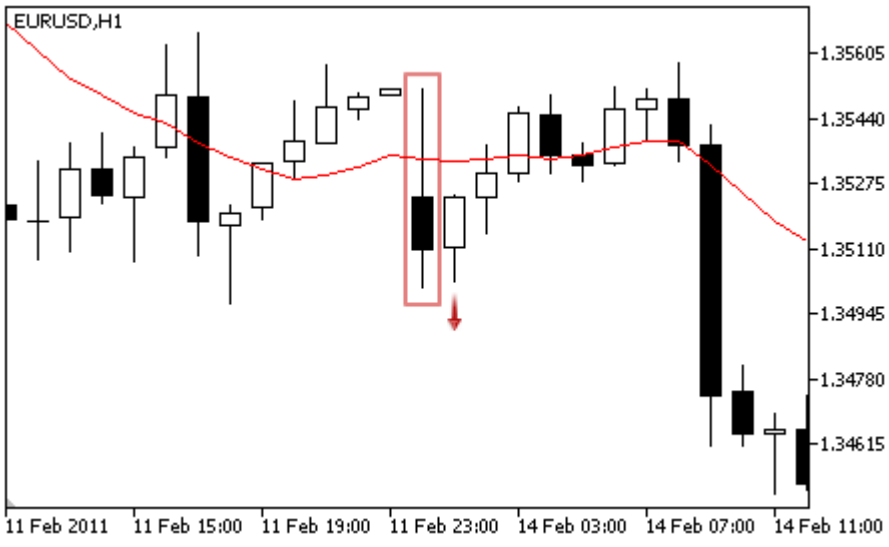
下記は、モジュールがエキスパートアドバイザーへのシグナルを受け渡す条件についてを説明します。

シグナルの種類	条件の説明
買いシグナル	• 価格が下向きに指標を交差（分析されているバーの始値が指標を上回り、終値が指標を下回る）し、指標が上昇しています（弱いシグナル）。  • 移動平均交差。　価格が上向きに指標を交差（分析されているバーの始値が指標を下回り、終値が指標を上回る）し、指標が上昇しています（強いシグナル）。  • バーの下部の影が指標と交差（分析されているバーの始値と終値が指標を上回り、安値が指標を下回る）し、指標が上昇しています（弱いシグナル）。

© 2000–2025, MetaQuotes Ltd.

シグナルの種類	条件の説明
	
売りシグナル	• 価格が上向きに指標と交差（分析されているバーの始値が指標を下回り、終値が指標を上回る）し、指標が下降しています（弱いシグナル）。  • **移動平均交差**。価格が下向きに指標を交差（分析されているバーの始値が指標を上回り、終値が指標を下回る）し、指標が下降しています（強いシグナル）。

© 2000–2025, MetaQuotes Ltd.

シグナルの種類	条件の説明
	 • バーの上部の影が指標と交差(分析されているバーの始値と終値が指標を下回り、高値が指標を上回る)し、指標が下降しています(弱いシグナル)。 
買いに異議なし	価格が指標を上回ります。
売りに異議なし	価格が指標を下回ります。

注意事項

エキスパートアドバイザーの動作モード(「Every tick」(ティックごと)または「Open prices only」(始値のみ))に応じて、分析バーは(インデックス 0 の)現在足、または(インデックス 1 の)直近に形成されたバーのいずれかです。

調整可能のパラメータ

このモジュールは、以下の調整可能なパラメータを持っています。

© 2000-2025, MetaQuotes Ltd.

パラメータ	説明
Weight	0～1 のモジュールのシグナルの重み
PeriodMA	指標の平均化の期間
Shift	時間軸に沿っての指標のシフト（バー単位）
Method	平均化の方法
Applied	指標計算に使用される価格列***

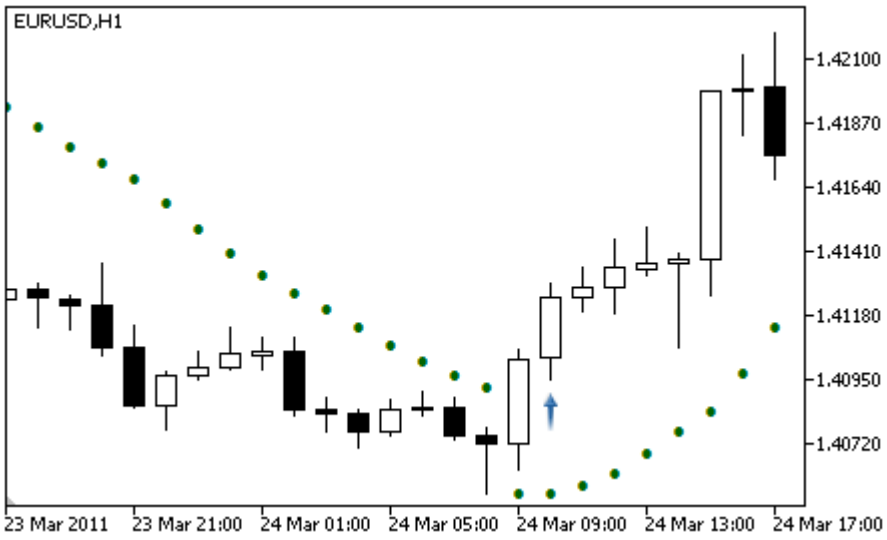
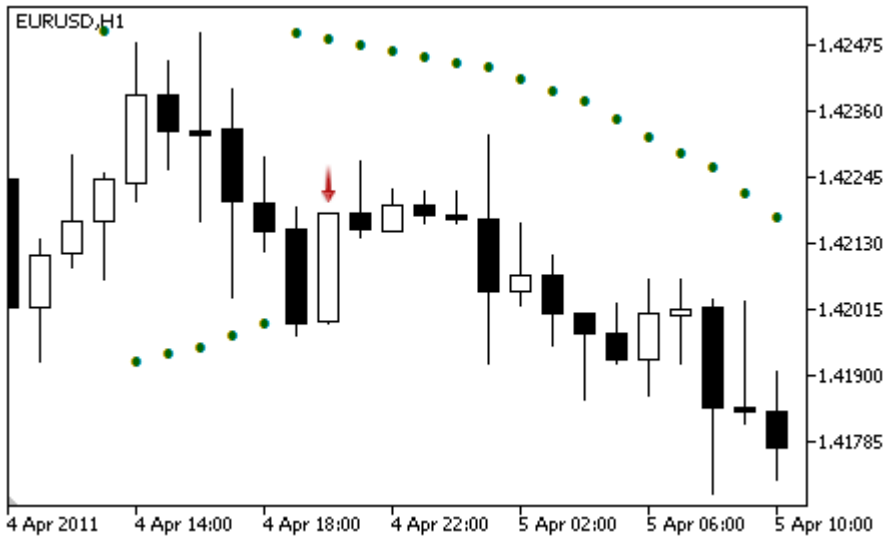
© 2000–2025, MetaQuotes Ltd.

パラボリック SAR 指標のシグナル

このシグナルモジュールはパラボリック SAR 指標のマーケットモデルに基づいています。モジュールから得られるシグナルに基づいての取引意思決定の仕組みは別のセクションで説明されています。

シグナル生成の条件

下記は、モジュールがエキスパートアドバイザーへのシグナルを受け渡す条件についてを説明します。

シグナルの種類	条件の説明
買いシグナル	反転 — 分析されているバーでは指標が価格を下回り、1 つ前のバーでは価格を上回ります。 
売りシグナル	反転 — 分析されているバーでは指標が価格を上回り、1 つ前のバーでは価格を下回ります。 
買いに異議なし	価格が指標を上回ります。
売りに異議なし	価格が指標を下回ります。

注意事項

© 2000–2025, MetaQuotes Ltd.

エキスパートアドバイザーの動作モード（「Every tick」（ティックごと）または「Open prices only」（始値のみ））に応じて、分析バーは（インデックス 0 の）現在足、または（インデックス 1 の）直近に形成されたバーのいずれかです。

調整可能のパラメータ

このモジュールは、以下の調整可能なパラメータを持っています。

パラメータ	説明
Weight	0～1 のモジュールのシグナルの重み
Step	価格増分ステップ
Maximum	価格と指標の収束の最大速度

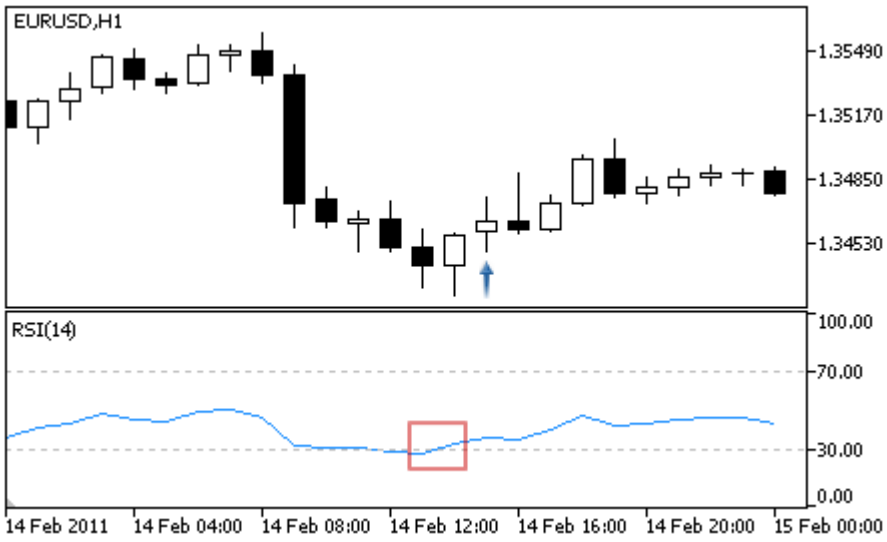
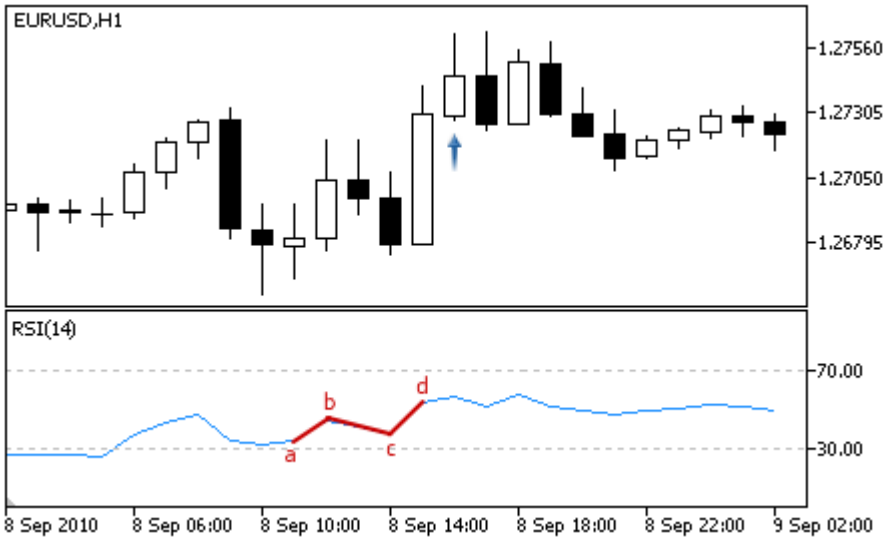
© 2000–2025, MetaQuotes Ltd.

相対力指数（Relative Strength Index）オシレーターのシグナル

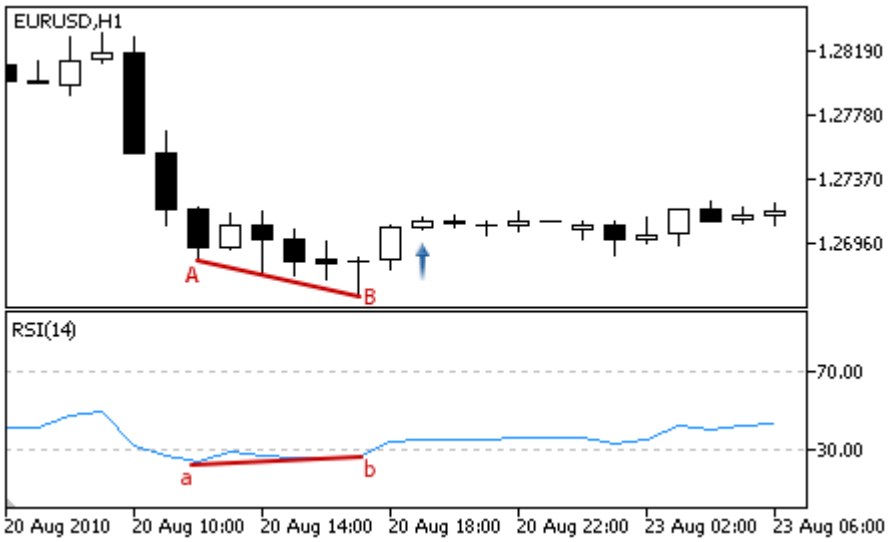
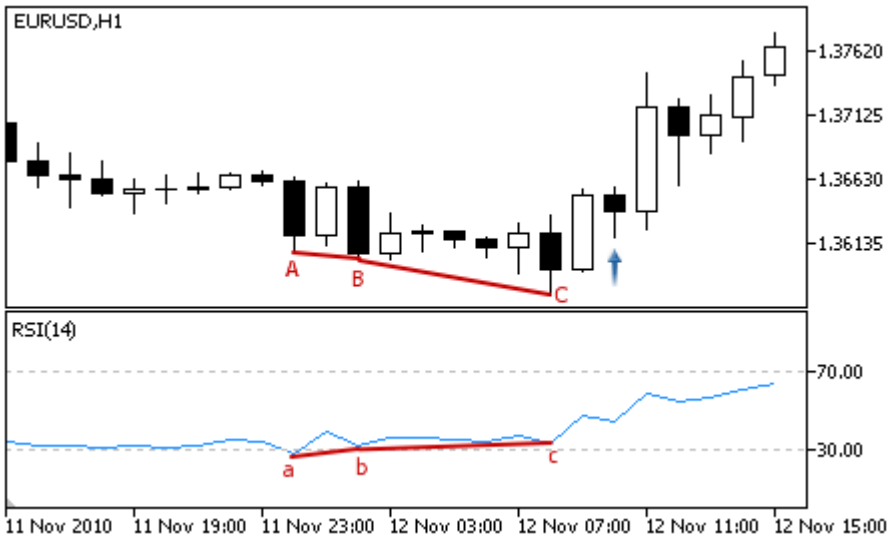
このシグナルモジュールは相対力指数オシレーター指標のマーケットモデルに基づいています。モジュールから得られるシグナルに基づいての取引意思決定の仕組みは別のセクションで説明されています。

シグナル生成の条件

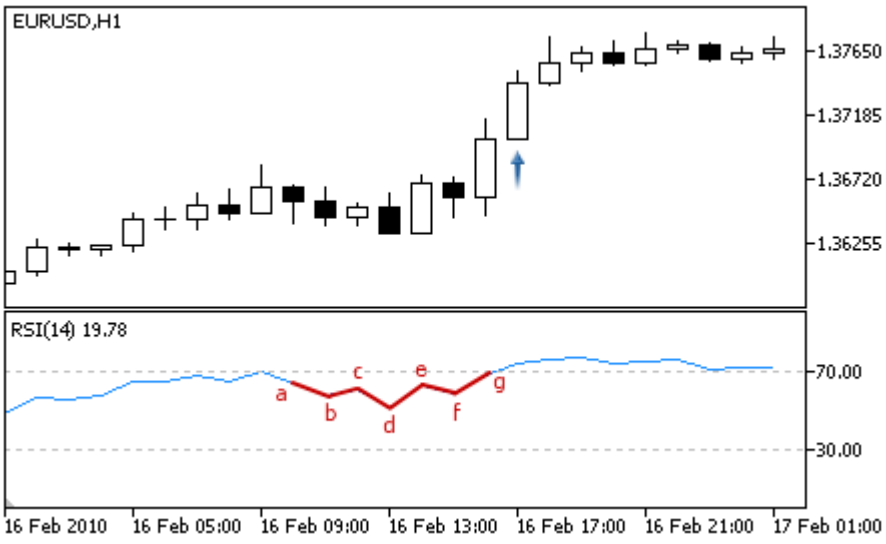
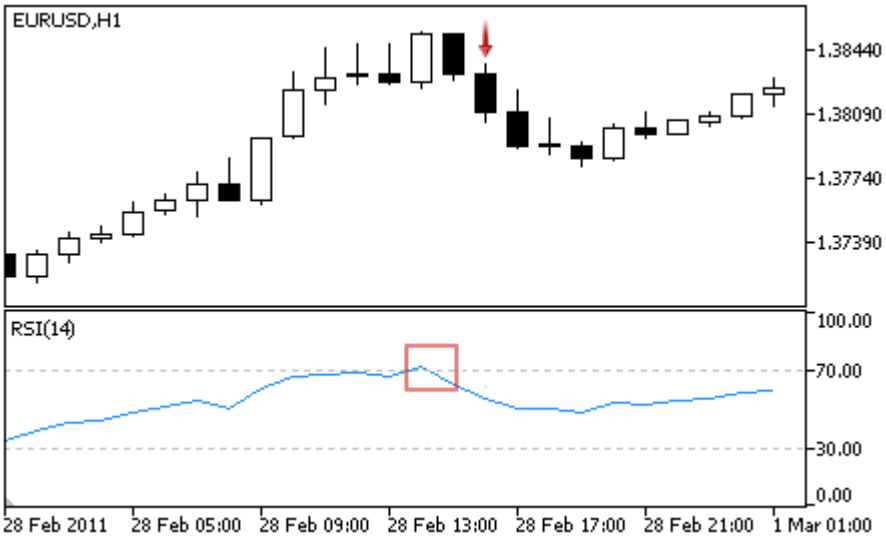
下記は、モジュールがエキスパートアドバイザーへのシグナルを受け渡す条件についてを説明します。

シグナルの種類	条件の説明
買いシグナル	• 売りすぎレベルを越した反転 – オシレーターが上向きで分析されたバーでの値が売りすぎレベルを越しています（初期値は 30）。  • 失敗したスイング – オシレーターが分析されたバーで前の天井よりも高くなります。  • 発散 – 指標の最初の分析底が 1 つ前のものより低く、対応する価格の底は、1つ前のものより低いです。

© 2000–2025, MetaQuotes Ltd.

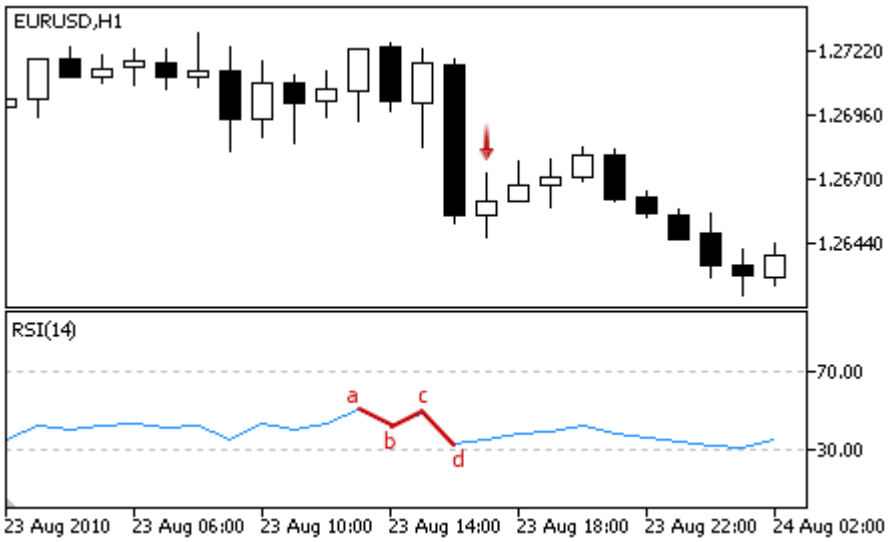
シグナルの種類	条件の説明
	 • 二重発散 –オシレーターがが3つの続いた底を形成し、それぞれが 1 つ前のものより高くなっています。価格は、3つの対応する底を形成し、それぞれが前のものよりも低いです。  • ヘッドアンドショルダー – オシレーターが3つの続いた底を形成し、真ん中の底があと 2 つより低いです。

© 2000–2025, MetaQuotes Ltd.

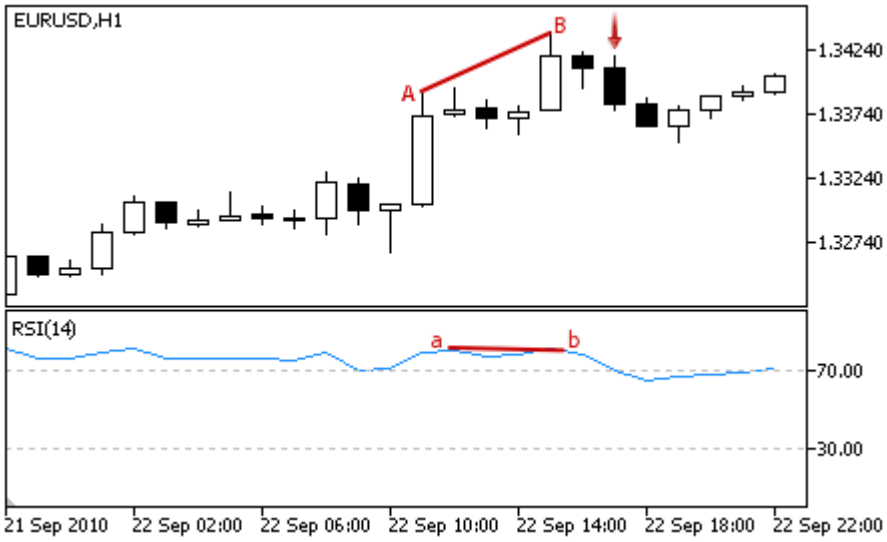
シグナルの種類	条件の説明
	
売りシグナル	• **買いすぎレベルを越した反転** — 買いすぎレベルを越した反転 — オシレーターが下向きで分析されたバーでの値が買いすぎレベルを越しています（初期値は 70）。  • **失敗したスイング** — オシレーターが分析されたバーで前の底よりも低くなります。

© 2000–2025, MetaQuotes Ltd.

シグナルの種類	条件の説明

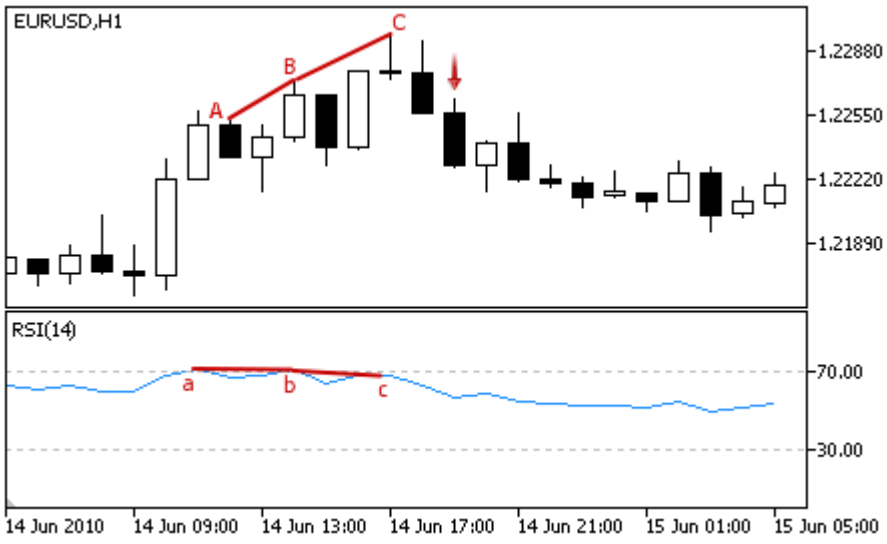
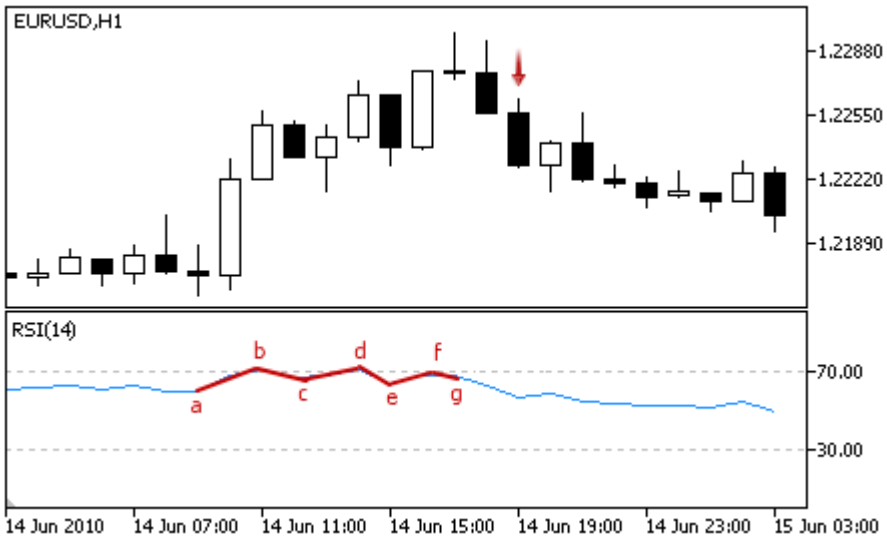


- 発散 – オシレーターの最初の分析された天井が1つ前のものよりも低く、対応する価格の天井は、1つ前のものよりも高いです。



- 二重発散 – オシレーターが3つの続いた天井を形成し、それぞれが 1 つ前のものより低くなっています。価格は、3つの対応する天井を形成し、それぞれが前のものよりも高いです.

© 2000–2025, MetaQuotes Ltd.

シグナルの種類	条件の説明
	 • ヘッドアンドショルダー – オシレーターが3つの続いた天井を形成し、真ん中の天井があと2つより高いです。 
買いに異議なし	分析されているバーでオシレーター値が増加します。
売りに異議なし	分析されているバーでオシレーター値が減少します。

注意事項

エキスパートアドバイザーの動作モード（「Every tick」（ティックごと）または「Open prices only」（始値のみ））に応じて、分析バーは（インデックス 0 の）現在足、または（インデックス 1 の）直近に形成されたバーのいずれかです。

調整可能のパラメータ

このモジュールは、以下の調整可能なパラメータを持っています。

© 2000–2025, MetaQuotes Ltd.

パラメータ	説 明
Weight	0～1 のモジュールのシグナルの重 み
PeriodRSI	指 標 計 算 の期 間
Applied	オシレーター計 算 に使 用 される価 格 列***

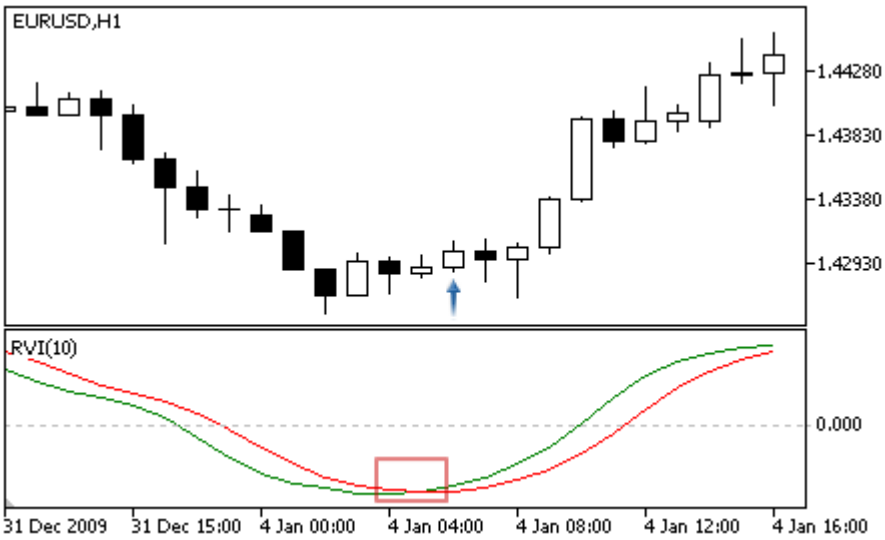
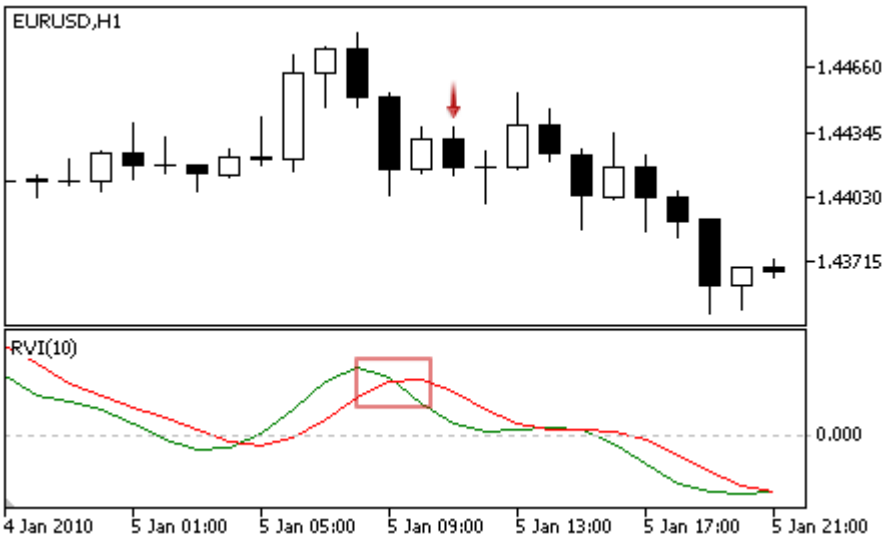
© 2000–2025, MetaQuotes Ltd.

相対力指数（Relative Vigor Index）オシレーターのシグナル

このシグナルモジュールは相対活力指数オシレーターのマーケットモデルに基づいています。モジュールから得られるシグナルに基づいての取引意思決定の仕組みは別のセクションで説明されています。

シグナル生成の条件

下記は、モジュールがエキスパートアドバイザーへのシグナルを受け渡す条件についてを説明します。

シグナルの種類	条件の説明
買いシグナル	**中心線とシグナル線の交差** — 分析されているバーでは中心線がシグナル線を上回り、1つ前のバーではシグナル線を下回ります。 
売りシグナル	**中心線とシグナル線の交差** — 分析されているバーでは中心線がシグナル線を下回り、1つ前のバーではシグナル線を上回ります。 
買いに異議なし	分析されているバーでオシレーター値が増加します。
売りに異議なし	分析されているバーでオシレーター値が減少します。

注意事項

© 2000–2025, MetaQuotes Ltd.

エキスパートアドバイザーの動作モード(「Every tick」(ティックごと)または「Open prices only」(始値のみ))に応じて、分析バーは(インデックス 0 の)現在足、または(インデックス 1 の)直近に形成されたバーのいずれかです。

調整可能のパラメータ

このモジュールは、以下の調整可能なパラメータを持っています。

パラメータ	説明
Weight	0～1 のモジュールのシグナルの重み
PeriodRVI	指標計算の期間

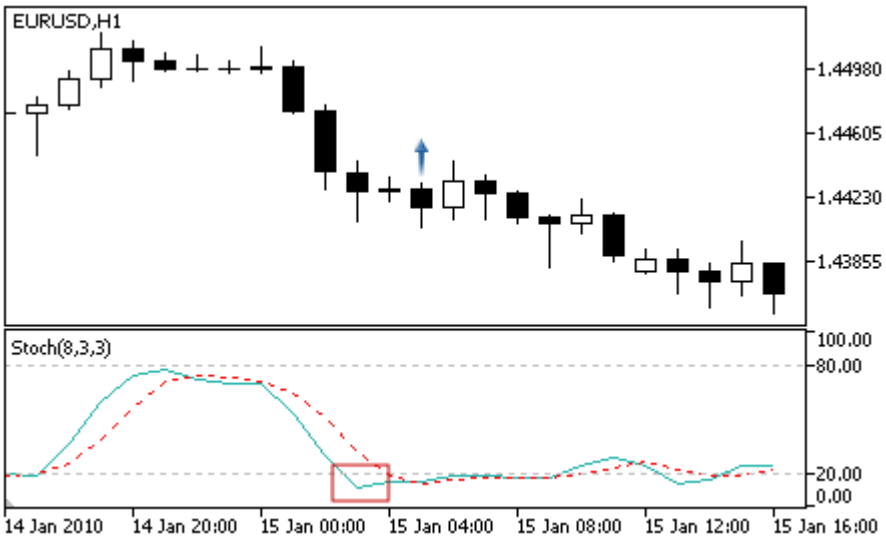
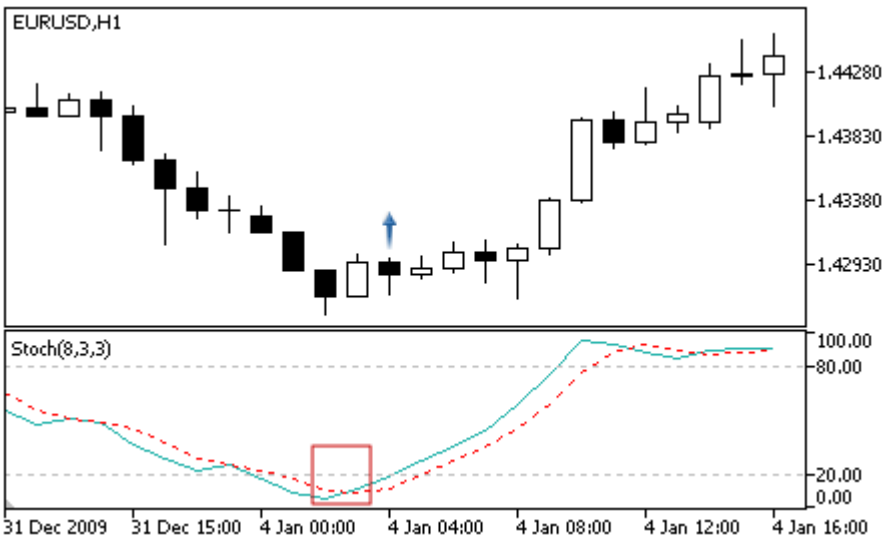
© 2000-2025, MetaQuotes Ltd.

ストキャスティックス(Stochastic)のシグナル

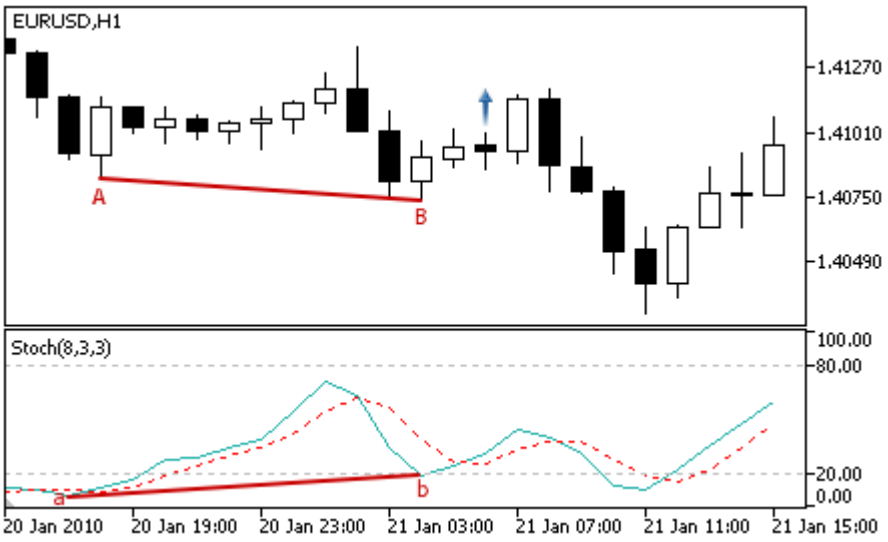
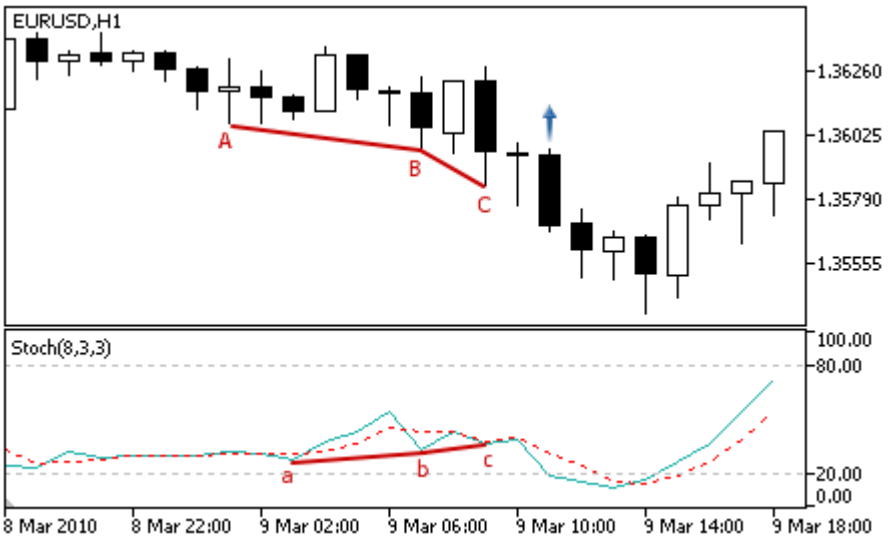
このシグナルモジュールはストキャスティックスオシレーターのマーケットモデルに基づいています。モジュールから得られるシグナルに基づいての取引意思決定の仕組みは別のセクションで説明されています。

シグナル生成の条件

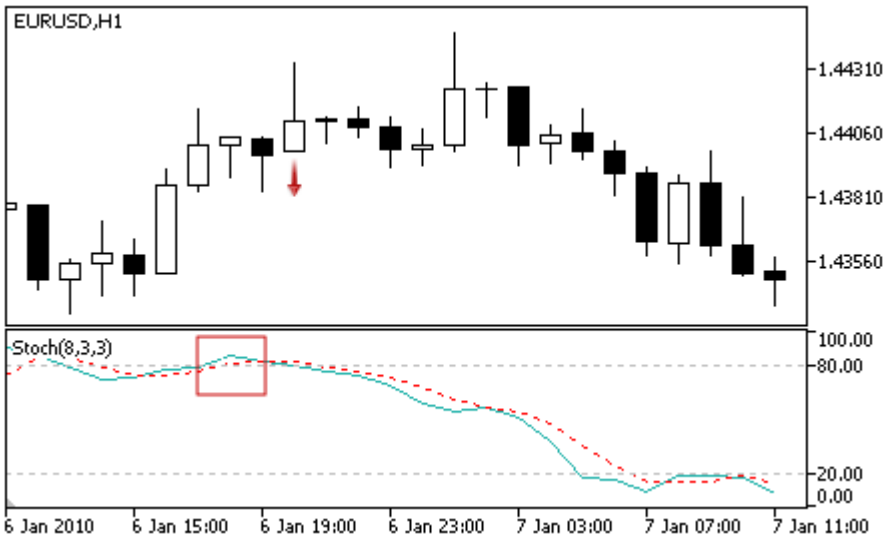
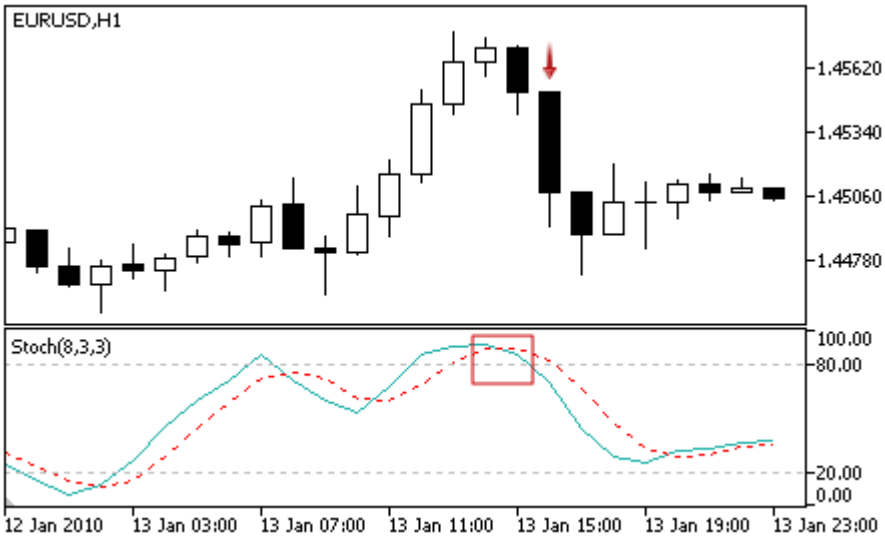
下記は、モジュールがエキスパートアドバイザーへのシグナルを受け渡す条件についてを説明します。

シグナルの種類	条件の説明
買いシグナル	• **反転** – オシレーターが上向きです(オシレーターは分析されているバーで上昇し1つ前のバーで下降します。)。  • **中心線とシグナル線の交差** – 分析されているバーでは中心線がシグナル線を上回り、1つ前のバーではシグナル線を下回ります。  • **発散** – 指標の最初の分析底が1つ前のものより高く、対応する価格の底は、1つ前のものより低いです。

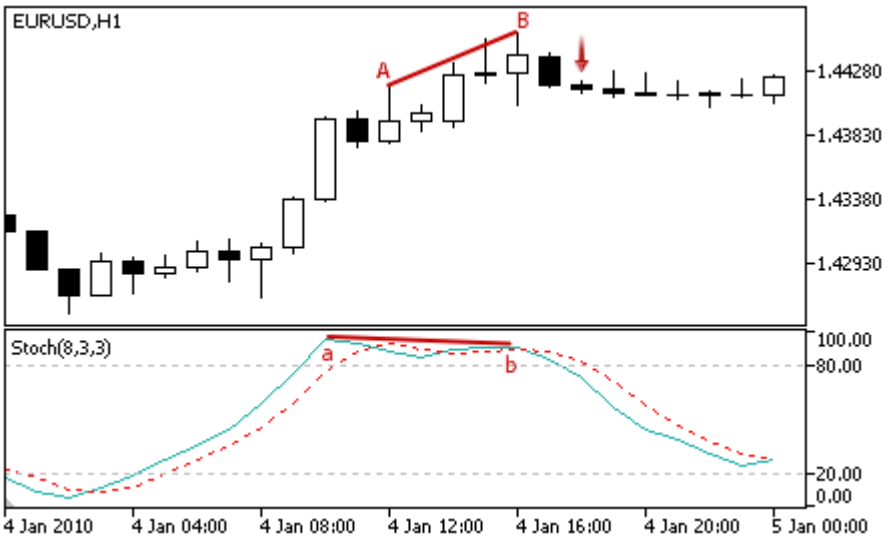
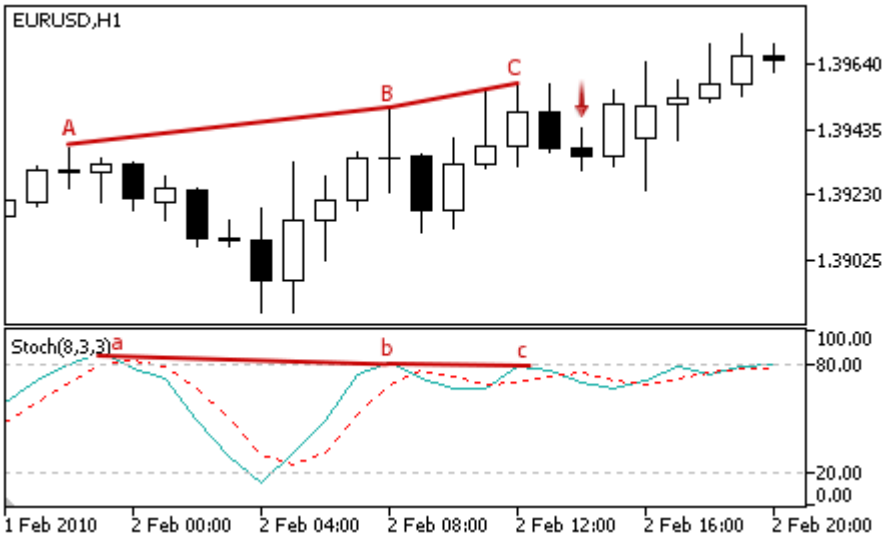
© 2000–2025, MetaQuotes Ltd.

シグナルの種類	条件の説明
	 • 二重発散 −オシレーター———が3つの続いた底を形成し、それぞれが 1 つ前のものより高くなっています。価格は、3つの対応する底を形成し、それぞれが前のものよりも低いです。 
売りシグナル	• 反転 − オシレーター———が下向きです（オシレーター———は分析されているバーで下降し 1 つ前のバーで上昇します。）

© 2000−2025, MetaQuotes Ltd.

シグナルの種類	条件の説明
	 • **中心線とシグナル線の交差** – 分析されているバーでは中心線がシグナル線を下回り、1つ前のバーではシグナル線を上回ります。  • **発散** – オシレーターの最初の分析された天井が1つ前のものよりも低く、対応する価格の天井は、1つ前のものよりも高いです。

© 2000–2025, MetaQuotes Ltd.

シグナルの種類	条件の説明
	 • 二重発散 – オシレーターが3つの続いた天井を形成し、それぞれが 1 つ前のものより低くなっています。価格は、3つの対応する天井を形成し、それぞれが前のものよりも高いです. 
買いに異議なし	分析されているバーでオシレーター値が増加します。
売りに異議なし	分析されているバーでオシレーター値が減少します。

注意事項

エキスパートアドバイザーの動作モード(「Every tick」(ティックごと)または「Open prices only」(始値のみ))に応じて、分析バーは(インデックス 0 の)現在足、または(インデックス 1 の)直近に形成されたバーのいずれかです。

調整可能のパラメータ

このモジュールは、以下の調整可能なパラメータを持っています。

© 2000–2025, MetaQuotes Ltd.

パラメータ	説明
Weight	0～1 のモジュールのシグナルの重み
PeriodK	オシレーターの正中線の平均期間
PeriodD	オシレーターの正中線の平均期間
PeriodSlow	低速期間
Applied	オシレーター計算に使用される価格列***

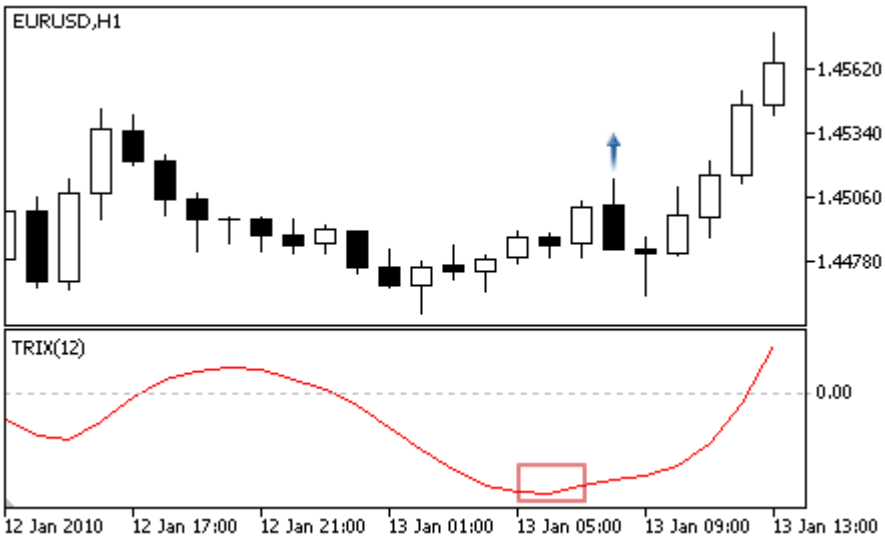
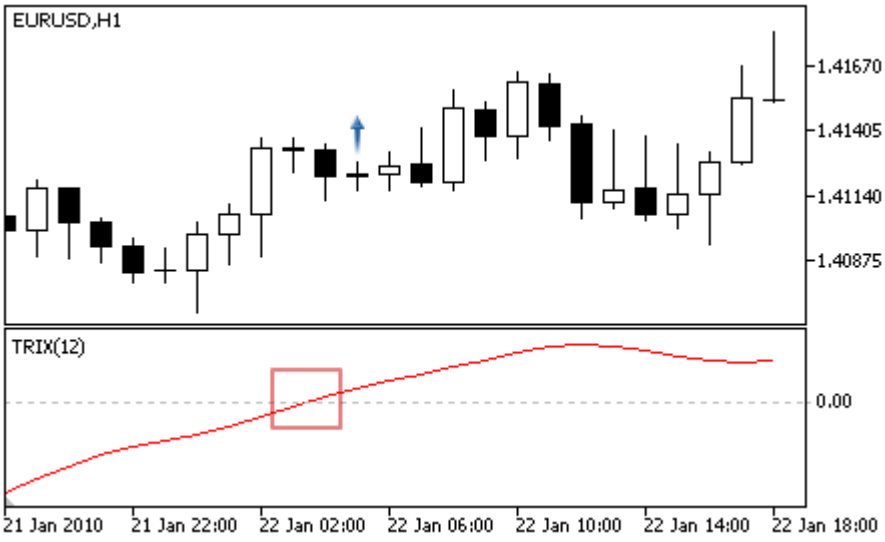
© 2000–2025, MetaQuotes Ltd.

3 重指数平均（ Triple Exponential Average ）オシレーターのシグナル

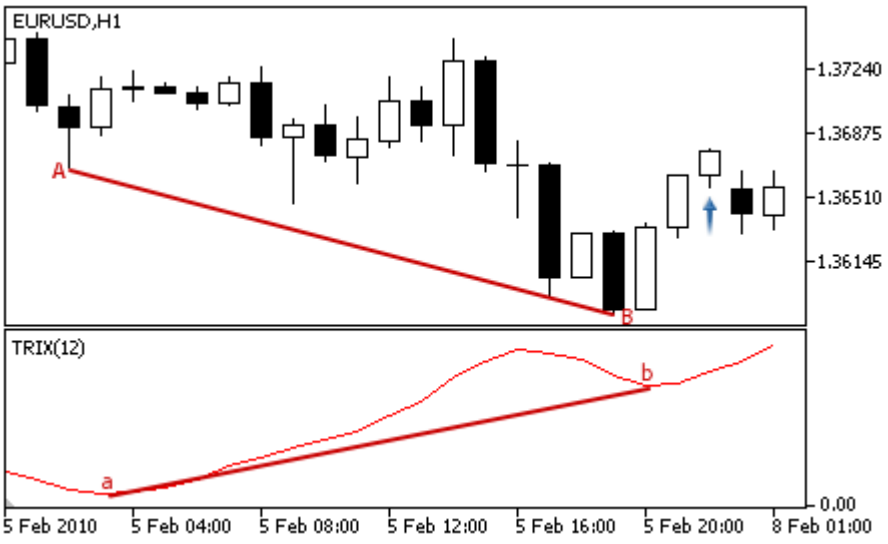
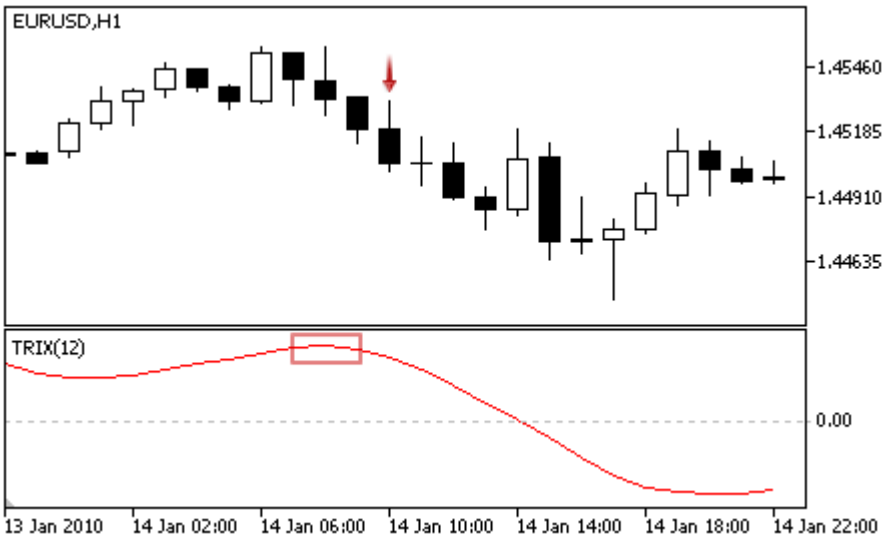
このシグナルモジュールは3 重指数平均オシレーター指標のマーケットモデルに基づいています。モジュールから得られるシグナルに基づいての取引意思決定の仕組みは別のセクションで説明されています。

シグナル生成の条件

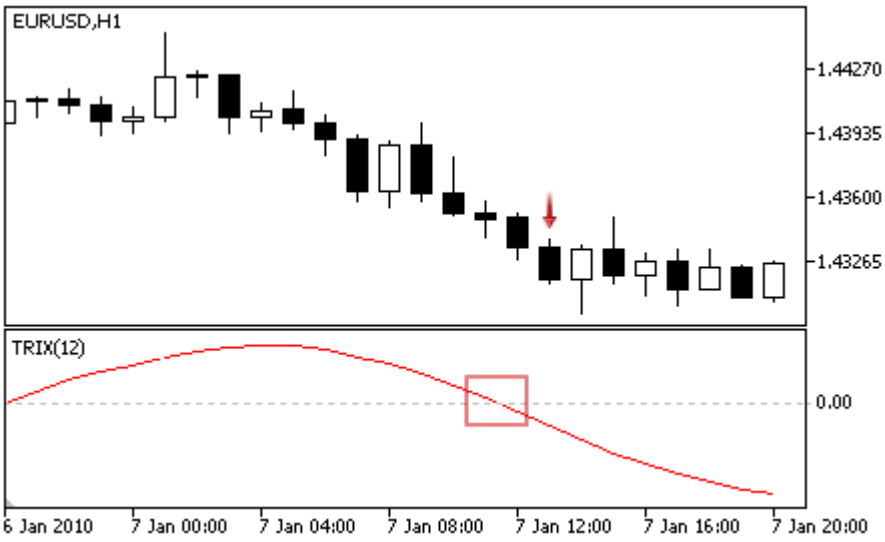
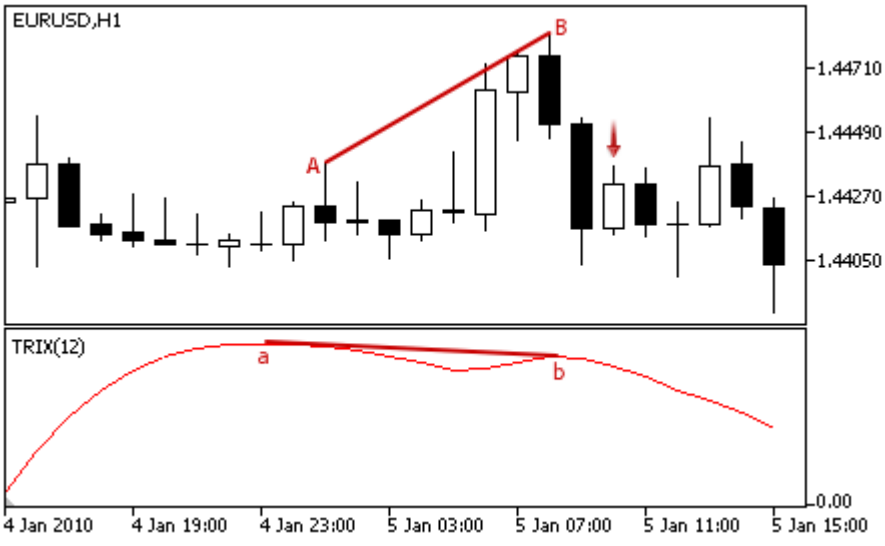
下記は、モジュールがエキスパートアドバイザーへのシグナルを受け渡す条件についてを説明します。

シグナルの種類	条件の説明
買いシグナル	• 反転 – オシレーターが上向きです（オシレーターは分析されているバーで上昇し 1 つ前のバーで下降します。）。  • ゼロレベルの交差 – 分析されているバーでは中心線がゼロレベルを上回り、1 つ前のバーでは下回ります。  • 発散 – 指標の最初の分析底が 1 つ前のものより高く、対応する価格の底は、1つ前のものより低いです。

© 2000–2025, MetaQuotes Ltd.

シグナルの種類	条件の説明
	
売りシグナル	• **反転** — オシレーターーーが下向きです（オシレーターーーは分析されているバーで下降し 1 つ前のバーで上昇します。）  • **ゼロレベルの交差** — 分析されているバーでは中心線がゼロレベルを下回り、1 つ前のバーでは上回ります。

© 2000–2025, MetaQuotes Ltd.

シグナルの種類	条件の説明
	 • 発散 – オシレーターの最初の分析された天井が1つ前のものよりも低く、対応する価格の天井は、1つ前のものよりも高いです。 
買いに異議なし	分析されているバーでオシレーター値が増加します。
売りに異議なし	分析されているバーでオシレーター値が減少します。

注意事項

エキスパートアドバイザーの動作モード（「Every tick」（ティックごと）または「Open prices only」（始値のみ））に応じて、分析バーは（インデックス 0 の）現在足、または（インデックス 1 の）直近に形成されたバーのいずれかです。

調整可能のパラメータ

このモジュールは、以下の調整可能なパラメータを持っています。

© 2000–2025, MetaQuotes Ltd.

パラメータ	説 明
Weight	0～1 のモジュールのシグナルの重 み
PeriodTriX	指 標 計 算 の期 間
Applied	オシレーター計 算 に使 用 される価 格 列 ***

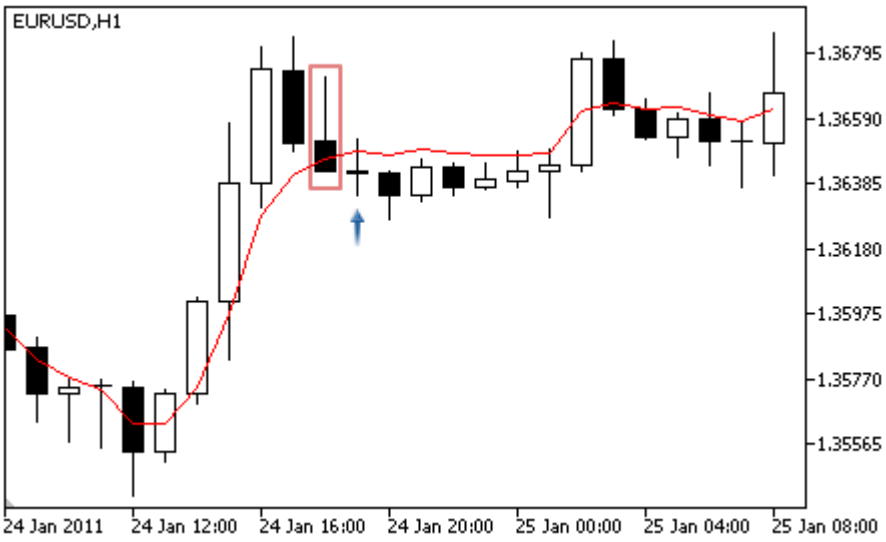
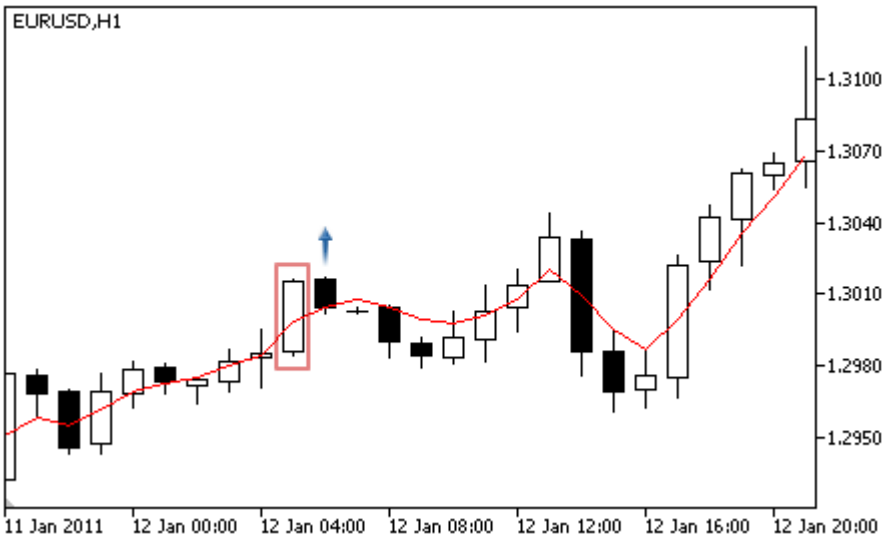
© 2000–2025, MetaQuotes Ltd.

3 重指数移動平均(Triple Exponential Moving Average)指標のシグナル

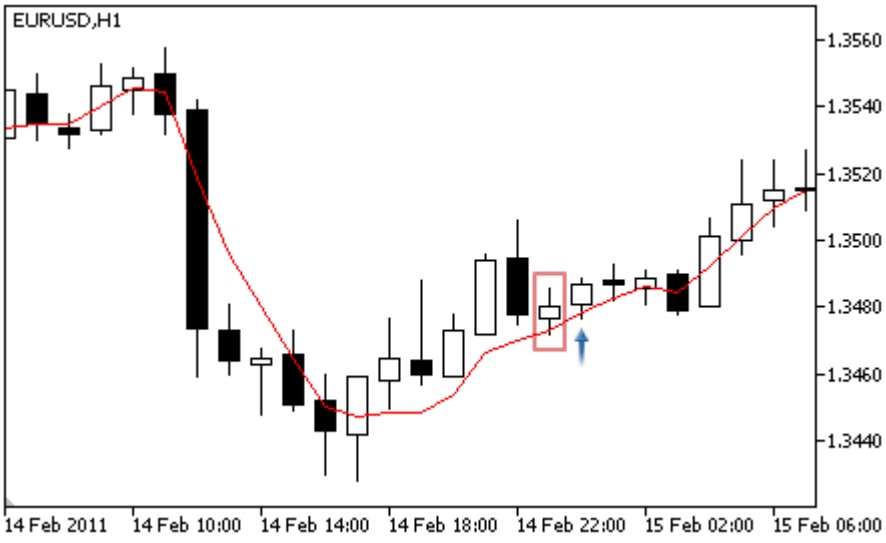
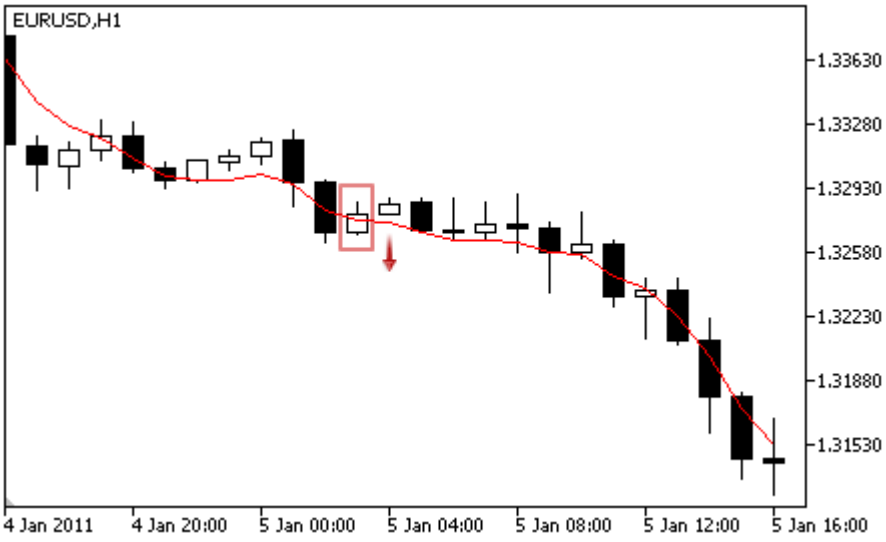
このシグナルモジュールは3 重指数移動平均指標指標のマーケットモデルに基づいています。モジュールから得られるシグナルに基づいての取引意思決定の仕組みは別のセクションで説明されています。

シグナル生成の条件

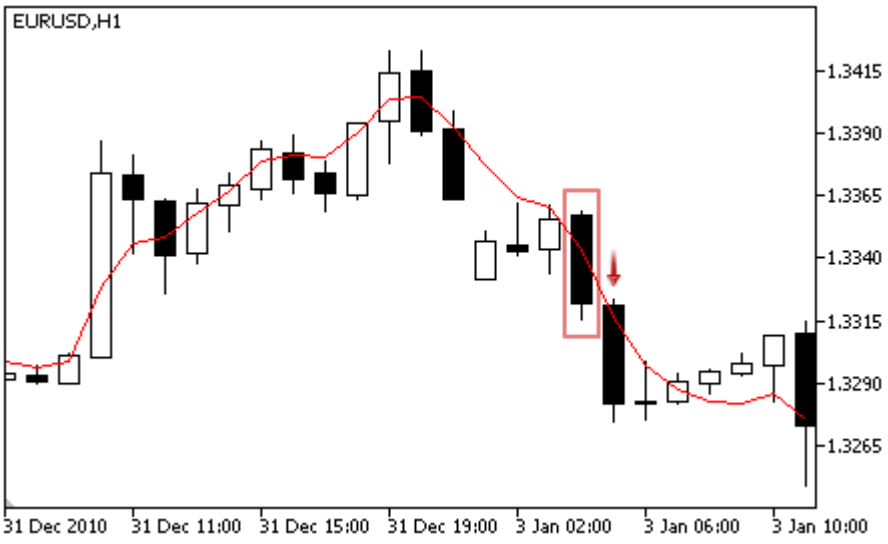
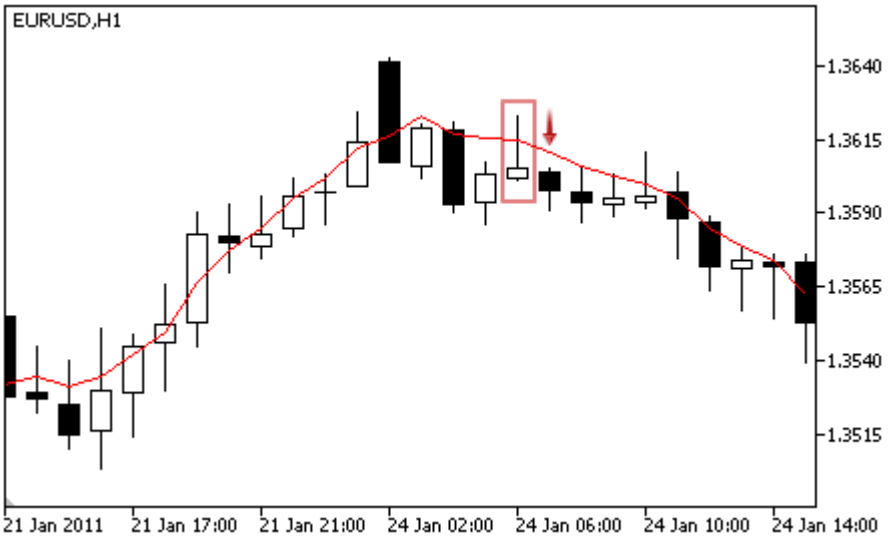
下記は、モジュールがエキスパートアドバイザーへのシグナルを受け渡す条件についてを説明します。

シグナルの種類	条件の説明
買いシグナル	• 価格が下向きに指標を交差(分析されているバーの始値が指標を上回り、終値が指標を下回る)し、指標が上昇しています(弱いシグナル)。  • 移動平均交差。 価格が上向きに指標を交差(分析されているバーの始値が指標を下回り、終値が指標を上回る)し、指標が上昇しています(強いシグナル)。 

© 2000–2025, MetaQuotes Ltd.

シグナルの種類	条件の説明
	• バーの下部の影が指標と交差（分析されているバーの始値と終値が指標を上回り、安値が指標を下回る）し、指標が上昇しています（弱いシグナル）。 
売りシグナル	• 価格が上向きに指標と交差（分析されているバーの始値が指標を下回り、終値が指標を上回る）し、指標が下降しています（弱いシグナル）。  • 移動平均交差。価格が下向きに指標を交差（分析されているバーの始値が指標を上回り、終値が指標を下回る）し、指標が下降しています（強いシグナル）。

© 2000–2025, MetaQuotes Ltd.

シグナルの種類	条件の説明
	 • バーの上部の影が指標と交差(分析されているバーの始値と終値が指標を下回り、高値が指標を上回る)し、指標が下降しています(弱いシグナル)。 
買いに異議なし	価格が指標を上回ります。
売りに異議なし	価格が指標を下回ります。

注意事項

エキスパートアドバイザーの動作モード(「Every tick」(ティックごと)または「Open prices only」(始値のみ))に応じて、分析バーは(インデックス 0 の)現在足、または(インデックス 1 の)直近に形成されたバーのいずれかです。

調整可能のパラメータ

このモジュールは、以下の調整可能なパラメータを持っています。

© 2000-2025, MetaQuotes Ltd.

パラメータ	説明
Weight	0～1 のモジュールのシグナルの重み
PeriodMA	指標の平均化の期間
Shift	時間軸に沿っての指標のシフト（バー単位）
Method	平均化の方法
Applied	指標計算に使用される価格列***

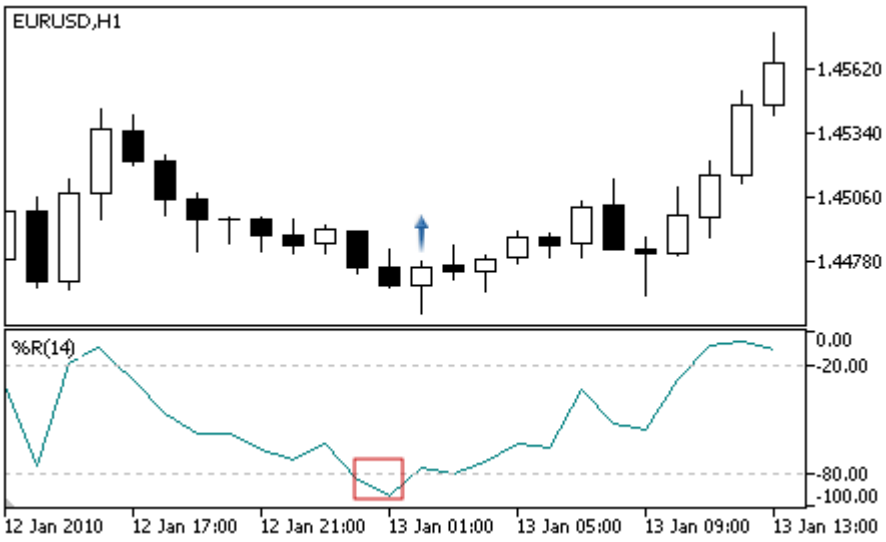
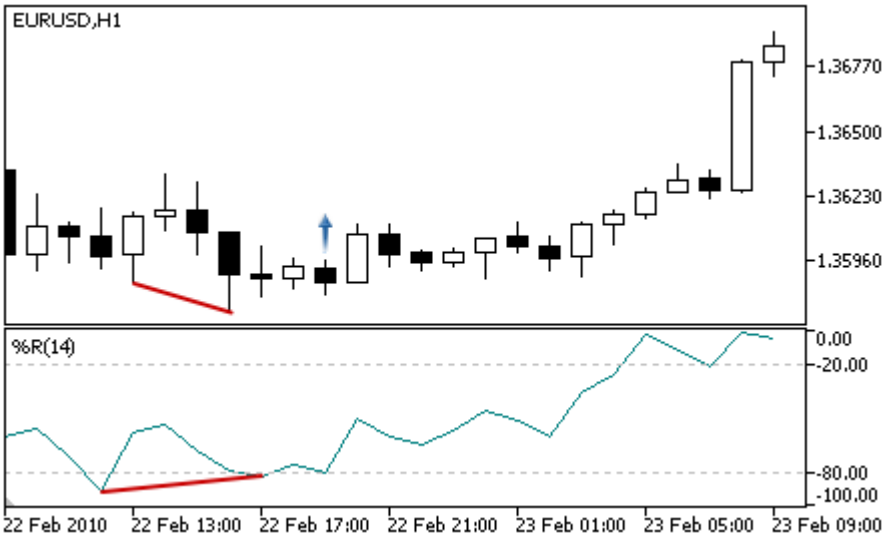
© 2000–2025, MetaQuotes Ltd.

ウィリアムパーセントレンジ（Williams Percent Range）オシレーターのシグナル

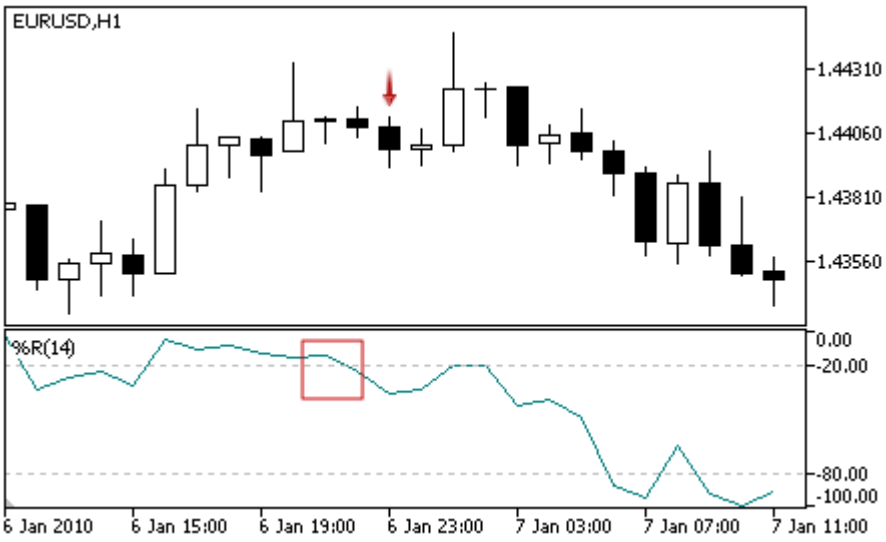
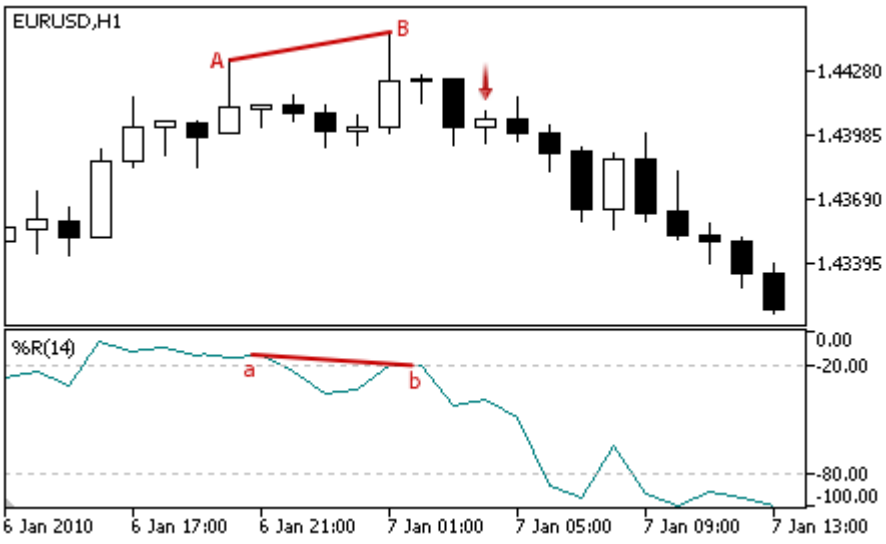
このシグナルモジュールはウィリアムパーセントレンジオシレーター指標のマーケットモデルに基づいています。モジュールから得られるシグナルに基づいての取引意思決定の仕組みは別のセクションで説明されています。

シグナル生成の条件

下記は、モジュールがエキスパートアドバイザーへのシグナルを受け渡す条件についてを説明します。

シグナルの種類	条件の説明
買いシグナル	• **売りすぎレベルを越した反転** — オシレーターが上向きで分析されたバーでの値が売りすぎレベルを越しています（初期値は -20）。  • **発散** — 指標の最初の分析底が 1 つ前のものより高く、対応する価格の底は、1つ前のものより低いです。 

© 2000–2025, MetaQuotes Ltd.

シグナルの種類	条件の説明
売りシグナル	• **買いすぎレベルを越した反転** – 買いすぎレベルを越した反転 – オシレーターが下向きで分析されたバーでの値が買いすぎレベルを越しています（初期値は -80）。  • **発散** – オシレーターの最初の分析された天井が1つ前のものよりも低く、対応する価格の天井は、1つ前のものよりも高いです。 
買いに異議なし	分析されているバーでオシレーター値が増加します。
売りに異議なし	分析されているバーでオシレーター値が減少します。

注意事項

エキスパートアドバイザーの動作モード（「Every tick」（ティックごと）または「Open prices only」（始値のみ））に応じて、分析バーは（インデックス 0 の）現在足、または（インデックス 1 の）直近に形成されたバーのいずれかです。

ウィリアムパーセントレンジ は反転したスケールを持つことを忘れてはいけません。-100 が最高値で最安値が 0 です。

© 2000–2025, MetaQuotes Ltd.

調整可能のパラメータ

このモジュールは、以下の調整可能なパラメータを持っています。

パラメータ	説明
Weight	0～1 のモジュールのシグナルの重み
PeriodWPR	指標計算の期間

© 2000-2025, MetaQuotes Ltd.

トレール注文のクラス

このセクションには、トレール注文のクラスの操作に関する技術的な詳細と関連するMQL5 標準ライブラリコンポーネントの説明が含まれています。

これらのクラスの使用は、取引ストラテジーの作成時（及びテスト時）に時間を節約します。

MQL5 標準ライブラリのトレール注文に関連したファイルは、端末ディレクトリの Include¥Expert¥Trailing フォルダ内に配置されています

Class	説明
CTrailingFixedPips	このクラスは、固定点に基づいたトレール注文のアルゴリズムを実装します。
CTrailingMA	このクラスは、移動平均指標の値に基づいたトレール注文のアルゴリズムを実装します。
CTrailingNone	スタブクラスで、トレール注文のアルゴリズムを使用しません。
CTrailingPSAR	このクラスは、パラボリック SAR 指標の値に基づいたトレール注文のアルゴリズムを実装します。

© 2000–2025, MetaQuotes Ltd.

CTrailingFixedPips

CTrailingFixedPips は固定点に基づいたトレール注文のアルゴリズムを実装するクラスです。

ポジションが決済逆指値の価格を持っている場合は、現在の価格への最小限の決済逆指値の距離をチェックします。値が決済逆指値より低い場合、決済逆指値を新しく設定することが推奨されます。この場合、もし決済指値があれば、その値も新しく設定されることが推奨されます。

エキスパートアドバイザーが every_tick=false フラグで初期化された場合、新しい操作（売買、トレールなど）は全て新しいバーでのみ行われます。この場合、決済指値が使用可能です。これは、新しいバーが完成する前に、決済指値でのポジション決済を可能にします。

説明

CTrailingFixedPips は、固定点に基づいたトレール注文のアルゴリズムを実装します。

宣言

```
class CTrailingFixedPips: public CExpertTrailing
```

タイトル

```
#include <Expert\Trailing\CTrailingFixedPips.mqh>
```

継承階層

CObject
  CExpertBase
    CExpertTrailing
      CTrailingFixedPips

クラスメソッド

初期化	
StopLevel	決済逆指値レベルの値を設定します。
ProfitLevel	決済指値レベルの値を設定します。
virtual ValidationSettings	設定のチェック
トレールの方法のチェック	
virtual CheckTrailingStopLong	ロング（買い）ポジションのトレール注文の条件をチェックします。
virtual CheckTrailingStopShort	ショート（売り）ポジションのトレール注文の条件をチェックします。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

クラスから継承されたメソッド CExpertBase

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

© 2000–2025, MetaQuotes Ltd.

StopLevel

決済逆指値レベルの値を設定します（ポイント単位）。

```
void  StopLevel(
   int  stop_level      // 決済逆指値レベル
  )
```

パラメータ

stop_loss

[in] 決済逆指値レベルの値（従来の2/4桁のポイント単位）

注意事項

決済逆指レベルが 0 に等しい場合、トレール注文は使用されません。

© 2000–2025, MetaQuotes Ltd.

ProfitLevel

決済指値レベルの値を設定します（ポイント単位）。

```
void  ProfitLevel(
   int  profit_level       // 決済指値レベル
  )
```

パラメータ

profit_level

[in] 決済指値レベルの値（従来の2/4桁のポイント単位）

注意事項

決済指値レベルが0に等しい場合、トレール注文は使用されません。

© 2000–2025, MetaQuotes Ltd.

ValidationSettings

設定をチェックします。

```
virtual bool  ValidationSettings()
```

戻り値

成功の場合は true、それ以外の場合は false

注意事項

この関数は決済指値及び決済逆指値レベルをチェックします。正しい値は 0 及び現在の終値からストップ注文を配置する場合の最少のインデントを超える値です。

© 2000–2025, MetaQuotes Ltd.

CheckTrailingStopLong

ロング（買い）ポジションのトレール注文の条件をチェックします。

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,      // CPositionInfo オブジェクトポインタ
   double&         sl,            // 決済逆指値
   double&         tp             // 決済指値
  )
```

パラメータ

position

[in] CPositionInfo オブジェクトへのポインタ

sl

[in][out] 決済逆指値の変数

tp

[in][out] 決済指値の変数

戻り値

条件が満たされている場合は true、それ以外の場合は false

注意事項

決済逆指レベルが 0 に等しい場合、トレール注文は使用されません。ポジションが決済逆指値の価格を持っている場合は、その値は基本価格であるとみなされます。それ以外の場合は、始値が基本価格であるとみなされます。

現在の売値が基本価格 + 決済指レベルより高い場合、新しい決済指レベルの設定が推奨されます。この場合、もし決済指値があれば、その値も新しく、売値 + 決済指値で設定されることが推奨されます。

© 2000–2025, MetaQuotes Ltd.

CheckTrailingStopShort

ショート（売り）ポジションのトレール注文の条件をチェックします。

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,      // CPositionInfo オブジェクトポインタ
   double&         sl,            // 決済逆指値
   double&         tp             // 決済指値
  )
```

パラメータ

position

[in] CPositionInfo オブジェクトへのポインタ

sl

[in][out] 決済逆指値の変数

tp

[in][out] 決済指値の変数

戻り値

条件が満たされている場合は true、それ以外の場合は false

注意事項

決済逆指レベルが 0 に等しい場合、トレール注文は使用されません。ポジションが決済逆指値の価格を持っている場合は、その値は基本価格であるとみなされます。それ以外の場合は、始値が基本価格であるとみなされます。

現在の買値が基本価格 - 決済指レベルより低い場合、新しい決済指レベルの設定が推奨されます。この場合、もし決済指値があれば、その値も新しく、買値 - 決済指値で設定されることが推奨されます。

© 2000-2025, MetaQuotes Ltd.

CTrailingMA

CTrailingMA は、移動平均指標の値に基づいたトレール注文のアルゴリズムを実装するクラスです。

説明

CTrailingMA クラスは 1 つ前の（完成した）バーでの移動平均指標の値に基づいたトレール注文のアルゴリズムを実装します。

宣言

```
class CTrailingMA: public CExpertTrailing
```

タイトル

```
   #include <Expert\Trailing\TrailingMA.mqh>
```

継承階層

COBject
CExpertBase
CExpertTrailing
CTrailingMA

クラスメソッド

初期化	
Period	移動平均の期間を設定します。
Shift	移動平均のシフトを設定します。
Method	低速移動平均の平滑化の方法を設定します。
Applied	移動平均の適用された価格を設定します。
virtual InitIndicators	指標と時系列を初期化します。
virtual ValidationSettings	設定のチェック
トレールの方法のチェック	
virtual CheckTrailingStopLong	ロング（買い）ポジションのトレール注文の条件をチェックします。
virtual CheckTrailingStopShort	ショート（売り）ポジションのトレール注文の条件をチェックします。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

クラスから継承されたメソッド CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries

© 2000–2025, MetaQuotes Ltd.

Period

移動平均の期間を設定します。

```
void  Period(
   int  period      // 平滑化期間
  )
```

パラメータ

period

[in] 移動平均の期間

© 2000–2025, MetaQuotes Ltd.

Shift

移動平均のシフトを設定します。

```
void  Shift(
   int  shift      // シフト
  )
```

パラメータ

shift

[in] 移動平均のシフト

© 2000–2025, MetaQuotes Ltd.

Method

移動平均の平滑化の方法を設定します。

```
void  Method(
   ENUM_MA_METHOD  method      // 平滑化の方法
  )
```

パラメータ

method

[in] 移動平均指標の平滑化の方法

© 2000–2025, MetaQuotes Ltd.

Applied

移動平均の適用された価格を設定します。

```
void  Applied(
   ENUM_APPLIED_PRICE  applied      // 適用された価格
  )
```

パラメータ

applied

[in] 移動平均の適用された価格

© 2000–2025, MetaQuotes Ltd.

InitIndicators

指標と時系列を初期化します。

```
virtual bool  InitIndicators(
   CIndicators*  indicators      // CIndicators コレクションへのポインタ
  )
```

パラメータ

指標

[in] 指標と時系列コレクションへのポインタ（CExpert クラスメンバ）。

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

ValidationSettings

設定をチェックします。

```
virtual bool  ValidationSettings()
```

戻り値

成功の場合は true、それ以外の場合は false

注意事項

この関数は、移動平均の期間をチェックします。正しい値は正です。

© 2000–2025, MetaQuotes Ltd.

CheckTrailingStopLong

ロング（買い）ポジションのトレーリング注文の条件をチェックします。

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,      // CPositionInfo オブジェクトポインタ
   double&         sl,            // 決済逆指値
   double&         tp             // 決済指値
  )
```

パラメータ

position

[in] CPositionInfo オブジェクトへのポインタ

sl

[in][out] 決済逆指値の変数

tp

[in][out] 決済指値の変数

戻り値

条件が満たされている場合は true、それ以外の場合は false

注意事項

まず、現在の価格に最も近い決済逆指値の可能な最大値を計算し、1つ前の（完了した）バーでの移動平均指標の値を使用して、決済逆指値を計算します。

Iポジションが決済逆指値の価格を持っている場合は、その値は基本価格であるとみなされます。それ以外の場合は、ポジションの始値が基本価格であるとみなされます。

算出された決済逆指値が基本価格よりも高く決済逆指値の可能な最大値より低い場合には、新しい決済逆指値の設定が推奨されます。

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopShort

ショート（売り）ポジションのトレール注文の条件をチェックします。

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,      // CPositionInfo オブジェクトポインタ
   double&         sl,            // 決済逆指値
   double&         tp             // 決済指値
  )
```

パラメータ

position

[in] CPositionInfo オブジェクトへのポインタ

sl

[in][out] 決済逆指値の変数

tp

[in][out] 決済指値の変数

戻り値

条件が満たされている場合は true、それ以外の場合は false

注意事項

First it calculates the minimal allowed Stop Loss price closest to the current price and calculates Stop Loss price using the values of moving average indicator of the previous (completed) bar.

Iポジションが決済逆指値の価格を持っている場合は、その値は基本価格であるとみなされます。それ以外の場合は、ポジションの始値が基本価格であるとみなされます。

算出された決済逆指値が基本価格よりも高く決済逆指値の可能な最小値より低い場合には、新しい決済逆指値の設定が推奨されます。

© 2000–2025, MetaQuotes Ltd.

CTrailingNone

CTrailingNone はスタブクラスです。このクラスは、ストラテジーがトレール注文を使用しない場合に Trailng オブジェクトの初期化時に使用されるべきです。

説明

CTrailingNone クラスは、トレール注文アルゴリズムを実装しません。トレール注文条件をチェックするメソッドは常に false を返します。

宣言

```
class CTrailingNone: public CExpertTrailing
```

タイトル

```
   #include <Expert\Trailing\TrailingNone.mqh>
```

継承階層

CObject
  CExpertBase
    CExpertTrailing
      CTrailingNone

クラスメソッド

トレールの方法のチェック	
virtual CheckTrailingStopLong	ロング（買い）ポジションのトレール注文の条件をチェックするスタブメソッドです。
virtual CheckTrailingStopShort	ショート（売り）ポジションのトレール注文の条件をチェックするスタブメソッドです。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

クラスから継承されたメソッド CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, ValidationSettings, SetPriceSeries, SetOtherSeries, InitIndicators

クラスから継承されたメソッド CExpertTrailing

CheckTrailingStopLong, CheckTrailingStopShort

© 2000–2025, MetaQuotes Ltd.

CheckTrailingStopLong

ロング（買い）ポジションのトレーリング注文の条件をチェックします。

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,      // CPositionInfo オブジェクトポインタ
   double&         sl,            // 決済逆指値
   double&         tp             // 決済指値
  )
```

パラメータ

position

[in] CPositionInfo オブジェクトへのポインタ

sl

[in][out] 決済逆指値の変数

tp

[in][out] 決済指値の変数

戻り値

条件が満たされている場合は true、それ以外の場合は false

注意事項

この関数は常に false を返します。

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopShort

ショート（売り）ポジションのトレール注文の条件をチェックします。

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,      // CPositionInfo オブジェクトポインタ
   double&         sl,            // 決済逆指値
   double&         tp             // 決済指値
  )
```

パラメータ

position

[in] CPositionInfo オブジェクトへのポインタ

sl

[in][out] 決済逆指値の変数

tp

[in][out] 決済指値の変数

戻り値

条件が満たされている場合は true、それ以外の場合は false

注意事項

この関数は常に false を返します。

© 2000–2025, MetaQuotes Ltd.

CTrailingPSAR

CTrailingPSAR は、パラボリック SAR 指標の値に基づいたトレール注文のアルゴリズムを実装するクラスです。

説明

CTrailingPSAR クラスは、一つ前の(完成した)パラボリック SAR 指標の値に基づいたトレール注文のアルゴリズムを実装します。

宣言

```
class CTrailingPSAR: public CExpertTrailing
```

タイトル

```
   #include <Expert\Trailing\TrailingParabolicSAR.mqh>
```

継承階層

CObject
  CExpertBase
    CExpertTrailing
      CTrailingPSAR

クラスメソッド

初期化	
Step	パラボリック SAR 指標のステップ値を設定します。
Maximum	パラボリック SAR 指標の最大値を設定します。
virtual InitIndicators	指標と時系列を初期化します。
トレールの方法のチェック	
virtual CheckTrailingStopLong	ロング(買い)ポジションのトレール注文の条件をチェックします。
virtual CheckTrailingStopShort	ショート(売り)ポジションのトレール注文の条件をチェックします。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

クラスから継承されたメソッド CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, ValidationSettings, SetPriceSeries, SetOtherSeries

© 2000–2025, MetaQuotes Ltd.

© 2000–2025, MetaQuotes Ltd.

Step

パラボリック SAR 指標のステップ値を設定します。

```
void  Step(
   double  step      // ステップ
  )
```

パラメータ

step

[in] パラボリック SAR 指標のステップ値

© 2000–2025, MetaQuotes Ltd.

Maximum

パラボリック SAR 指標の最大値を設定します。

```
void  Maximum(
   double  maximum      // 最大値
  )
```

パラメータ

maximum

[in] パラボリック SAR 指標の最大値

© 2000–2025, MetaQuotes Ltd.

InitIndicators

指標と時系列を初期化します。

```
virtual bool  InitIndicators(
   CIndicators*  indicators      // CIndicators コレクションへのポインタ
  )
```

パラメータ

指標

[in] 指標と時系列コレクションへのポインタ（CExpert クラスメンバ）。

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CheckTrailingStopLong

ロング（買い）ポジションのトレーリング注文の条件をチェックします。

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,      // ポインタ
   double&         sl,            // リンク
   double&         tp             // リンク
  )
```

パラメータ

position

[in] CPositionInfo オブジェクトへのポインタ

sl

[in][out] 決済逆指値の変数

tp

[in][out] 決済指値の変数

戻り値

条件が満たされている場合は true、それ以外の場合は false

注意事項

まず、現在の価格に最も近い決済逆指値の可能な最大値を計算し、1 つ前の（完了した）バーでのパラボリック SAR 指標の値を使用して、決済逆指値を計算します。

ポジションが決済逆指値の価格を持っている場合は、その値は基本価格であるとみなされます。それ以外の場合は、始値が基本価格であるとみなされます。

算出された決済逆指値が基本価格よりも高く決済逆指値の可能な最大値より低い場合には、新しい決済逆指値の設定が推奨されます。

© 2000–2025, MetaQuotes Ltd.

CheckTrailingStopShort

ショート（売り）ポジションのトレール注文の条件をチェックします。

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,      // ポインタ
   double&         sl,            // リンク
   double&         tp             // リンク
  )
```

パラメータ

position

[in] CPositionInfo オブジェクトへのポインタ

sl

[in][out] 決済逆指値の変数

tp

[in][out] 決済指値の変数

戻り値

条件が満たされている場合は true、それ以外の場合は false

注意事項

First it calculates the minimal allowed Stop Loss price closest to the current price and calculates Stop Loss price using the values of Parabolic SAR indicator of the previous (completed) bar.

ポジションが決済逆指値の価格を持っている場合は、その値は基本価格であるとみなされます。それ以外の場合は、始値が基本価格であるとみなされます。

算出された決済逆指値が基本価格よりも高く決済逆指値の可能な最小値より低い場合には、新しい決済逆指値の設定が推奨されます。

© 2000–2025, MetaQuotes Ltd.

資金管理のクラス

このセクションには、資金及びリスク管理のクラスの操作に関する技術的な詳細と関連するMQL5 標準ライブラリコンポーネントの説明が含まれています。

これらのクラスの使用は、取引ストラテジーの作成時（及びテスト時）に時間を節約します。

MQL5 標準ライブラリの資金及びリスク管理に関連したファイルは、端末ディレクトリの Include¥Expert¥Money¥ フォルダ内に配置されています。

Class	説明
CMoneyFixedLot	このクラスは、事前に定義された固定ロットサイズでの取引に基づく資金管理アルゴリズムを実装します。
CMoneyFixedMargin	このクラスは、事前に定義された固定証拠金での取引に基づく資金管理アルゴリズムを実装します。
CMoneyFixedRisk	このクラスは、事前に定義されたリスクでの取引に基づく資金管理アルゴリズムを実装します。<77/84/88% >
CMoneyNone	このクラスは、事前に定義された最小限のロットサイズでの取引に基づく資金管理アルゴリズムを実装します。
CMoneySizeOptimized	このクラスは、前の取引の結果に応じて変更されたロットサイズでの取引に基づく資金管理アルゴリズムを実装します。

© 2000–2025, MetaQuotes Ltd.

CMoneyFixedLot

CMoneyFixedLot は、事前に定義された固定ロットサイズでの取引に基づく資金管理アルゴリズムを実装するクラスです。

説明

CMoneyFixedLot クラスは、事前に定義された固定ロットサイズでの取引に基づく資金管理アルゴリズムを実装します。

宣言

```
class CMoneyFixedLot: public CExpertMoney
```

タイトル

```
#include <Expert\Money\MoneyFixedLot.mqh>
```

継承階層

CObject
  CExpertBase
    CExpertMoney
      CMoneyFixedLot

クラスメソッド

初期化	
Lots	取引高を設定します。
virtual ValidationSettings	設定のチェック
資金及びリスク管理のメソッド	
virtual CheckOpenLong	ロング（買い）ポジションの取引高を取得します。
virtual CheckOpenShort	ショート（売り）ポジションの取引高を取得します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

クラスから継承されたメソッド CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

クラスから継承されたメソッド CExpertMoney

Percent, CheckReverse, CheckClose

© 2000-2025, MetaQuotes Ltd.

© 2000–2025, MetaQuotes Ltd.

Lots

取引高を設定します。(ロット単位)。

```
void  Lots(
   double  lots      // ロット
  )
```

パラメータ

lots

[in] 取引高(ロット単位)

© 2000-2025, MetaQuotes Ltd.

ValidationSettings

設定をチェックします。

```
virtual bool  ValidationSettings()
```

戻り値

成功の場合は true、それ以外の場合は false

注意事項

指定された取引高の正しさをチェックします。

© 2000–2025, MetaQuotes Ltd.

CheckOpenLong

ロング（買い）ポジションの取引高を取得します。

```
virtual double  CheckOpenLong(
   double  price,     // 価格
   double  sl         // 決済逆指値
  )
```

パラメータ

価格

[in] 価格

sl

[in] 決済逆指値

戻り値

ロング（買い）ポジションの取引高

注意事項

この関数は常に Lots メソッドで定義された固定取引高を返します。

© 2000–2025, MetaQuotes Ltd.

CheckOpenShort

ショート（売り）ポジションの取引高を取得します。

```
virtual double  CheckOpenShort(
   double  price,     // 価格
   double  sl         // 決済逆指値
  )
```

パラメータ

価格

[in] 価格

sl

[in] 決済逆指値

戻り値

ショート（売り）ポジションの取引高

注意事項

この関数は常に Lots メソッドで定義された固定取引高を返します。

© 2000–2025, MetaQuotes Ltd.

CMoneyFixedMargin

CMoneyFixedMargin は、事前に定義された固定証拠金での取引に基づく資金管理アルゴリズムを持つクラスです。

説明

CMoneyFixedMargin は、事前に定義された固定証拠金での取引に基づく資金管理アルゴリズムを実装します。

宣言

```
class CMoneyFixedMargin: public CExpertMoney
```

タイトル

```
#include <Expert\Money\MoneyFixedMargin.mqh>
```

継承階層

CObject
  CExpertBase
    CExpertMoney
      CMoneyFixedMargin

クラスメソッド

資金及びリスク管理のメソッド	
virtual CheckOpenLong	ロング（買い）ポジションの取引高を取得します。
virtual CheckOpenShort	ショート（売り）ポジションの取引高を取得します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

クラスから継承されたメソッド CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

クラスから継承されたメソッド CExpertMoney

Percent, ValidationSettings, CheckReverse, CheckClose

© 2000–2025, MetaQuotes Ltd.

CheckOpenLong

ロング（買い）ポジションの取引高を取得します。

```
virtual double  CheckOpenLong(
   double  price,     // 価格
   double  sl         // 決済逆指値
  )
```

パラメータ

価格

[in] 価格

sl

[in] 決済逆指値

戻り値

ロング（買い）ポジションの取引高

注意事項

この関数はロング（買い）ポジションの取引高を返し、固定証拠金を使用します。証拠金は CExpertMoney 基本クラスの Percent パラメータで定義されています。

© 2000–2025, MetaQuotes Ltd.

CheckOpenShort

ショート（売り）ポジションの取引高を取得します。

```
virtual double  CheckOpenShort(
   double  price,     // 価格
   double  sl         // 決済逆指値
  )
```

パラメータ

価格

[in] 価格

sl

[in] 決済逆指値

戻り値

ショート（売り）ポジションの取引高

注意事項

この関数はショート（売り）ポジションの取引高を返し、固定証拠金を使用します。証拠金はCExpertMoney基本クラスのPercentパラメータで定義されています。

© 2000–2025, MetaQuotes Ltd.

CMoneyFixedRisk

CMoneyFixedRisk は、事前に定義されたリスクでの取引に基づく資金管理アルゴリズムを持つクラスです。

説明

CMoneyFixedRisk は、事前に定義されたリスクでの取引に基づく資金管理アルゴリズムを実装します。

宣言

```
class CMoneyFixedRisk: public CExpertMoney
```

タイトル

```
#include <Expert\Money\MoneyFixedRisk.mqh>
```

継承階層

CObject
  CExpertBase
    CExpertMoney
      CMoneyFixedRisk

クラスメソッド

資金及びリスク管理のメソッド	
virtual CheckOpenLong	ロング（買い）ポジションの取引高を取得します。
virtual CheckOpenShort	ショート（売り）ポジションの取引高を取得します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

クラスから継承されたメソッド CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

クラスから継承されたメソッド CExpertMoney

Percent, ValidationSettings, CheckReverse

© 2000–2025, MetaQuotes Ltd.

CheckOpenLong

ロング（買い）ポジションの取引高を取得します。

```
virtual double  CheckOpenLong(
   double  price,     // 価格
   double  sl         // 決済逆指値
  )
```

パラメータ

価格

[in] 価格

sl

[in] 決済逆指値

戻り値

ロング（買い）ポジションの取引高

注意事項

この関数はロング（買い）ポジションの取引高を返し、固定リスクを使用します。リスクは CExpertMoney 基本クラスの Percent パラメータで定義されています。

© 2000–2025, MetaQuotes Ltd.

CheckOpenShort

ショート（売り）ポジションの取引高を取得します。

```
virtual double  CheckOpenShort(
   double  price,     // 価格
   double  sl         // 決済逆指値
  )
```

パラメータ

価格

[in] 価格

sl

[in] 決済逆指値

戻り値

ショート（売り）ポジションの取引高

注意事項

この関数はショート（売り）ポジションの取引高を返し、固定リスクを使用します。リスクは CExpertMoney 基本クラスの Percent パラメータで定義されています。

© 2000–2025, MetaQuotes Ltd.

CMoneyNone

CMoneyNone は、事前に定義された最小限のロットサイズでの取引に基づくアルゴリズムを持つクラスです。

説明

CMoneyNone クラスは、最小限のロットサイズでの取引を実装します。

宣言

```
class CMoneyNone: public CExpertMoney
```

タイトル

```
#include <Expert\Money\MoneyNone.mqh>
```

継承階層

CObject
  CExpertBase
    CExpertMoney
      CMoneyNone

クラスメソッド

初期化	
virtual ValidationSettings	設定のチェック
資金及びリスク管理のメソッド	
virtual CheckOpenLong	ロング（買い）ポジションの取引高を取得します。
virtual CheckOpenShort	ショート（売り）ポジションの取引高を取得します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

クラスから継承されたメソッド CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

クラスから継承されたメソッド CExpertMoney

Percent, CheckReverse, CheckClose

© 2000–2025, MetaQuotes Ltd.

ValidationSettings

設定をチェックします。

```
virtual bool  ValidationSettings()
```

戻り値

成功の場合は true、それ以外の場合は false

注意事項

この関数は常に true を返します。

© 2000–2025, MetaQuotes Ltd.

CheckOpenLong

ロング（買い）ポジションの取引高を取得します。

```
virtual double  CheckOpenLong(
   double  price,     // 価格
   double  sl         // 決済逆指値
  )
```

パラメータ

価格

[in] 価格

sl

[in] 決済逆指値

戻り値

ロング（買い）ポジションの取引高

注意事項

この関数は常に最小限のロットを返します。

© 2000–2025, MetaQuotes Ltd.

CheckOpenShort

ロング（買い）ポジションの取引高を取得します。

```
virtual double  CheckOpenShort(
   double  price,     // 価格
   double  sl         // 決済逆指値
  )
```

パラメータ

価格

[in] 価格

sl

[in] 決済逆指値

戻り値

ショート（売り）ポジションの取引高

注意事項

この関数は常に最小限のロットを返します。

© 2000–2025, MetaQuotes Ltd.

CMoneySizeOptimized

CMoneySizeOptimized は、前の約定の結果に応じて変更されたロットサイズでの取引に基づく資金管理アルゴリズムを持つクラスです。

説明

CMoneySizeOptimized は、前の約定の結果に応じて変更されたロットサイズでの取引に基づく資金管理アルゴリズムを実装します。

宣言

```
class CMoneySizeOptimized: public CExpertMoney
```

タイトル

```
#include <Expert\Money\MoneySizeOptimized.mqh>
```

継承階層

CObject
  CExpertBase
    CExpertMoney
      CMoneySizeOptimized

クラスメソッド

初期化	
DecreaseFactor	減少率の値を設定します。
virtual ValidationSettings	設定のチェック
資金及びリスク管理のメソッド	
virtual CheckOpenLong	ロング（買い）ポジションの取引高を取得します。
virtual CheckOpenShort	ショート（売り）ポジションの取引高を取得します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

クラスから継承されたメソッド CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

クラスから継承されたメソッド CExpertMoney

Percent, CheckReverse, CheckClose

© 2000–2025, MetaQuotes Ltd.

© 2000–2025, MetaQuotes Ltd.

DecreaseFactor

減少率の値を設定します。

```
void  DecreaseFactor(
   double  decrease_factor      // 減少率
  )
```

パラメータ

decrease_factor

[in] 減少率

注意事項

DecreaseFactor は、連続した損失取引の場合の（前のポジションのボリュームと比較した）ボリュームの減少係数を定義します。

© 2000–2025, MetaQuotes Ltd.

ValidationSettings

設定をチェックします。

```
virtual bool  ValidationSettings()
```

戻り値

成功の場合は true、それ以外の場合は false

注意事項

減少率の値が負であル場合は false、それ以外の場合は true を返します。

© 2000–2025, MetaQuotes Ltd.

CheckOpenLong

ロング（買い）ポジションの取引高を取得します。

```
virtual double  CheckOpenLong(
   double  price,     // 価格
   double  sl         // 決済逆指値
  )
```

パラメータ

価格

[in] 価格

sl

[in] 決済逆指値

戻り値

ロング（買い）ポジションの取引高

注意事項

この関数はロング（買い）ポジションの取引高を返します。ボリュームは前の注文の結果に依存します。

© 2000–2025, MetaQuotes Ltd.

CheckOpenShort

ショート（売り）ポジションの取引高を取得します。

```
virtual double  CheckOpenShort(
   double  price,     // 価格
   double  sl         // 決済逆指値
  )
```

パラメータ

価格

[in] 価格

sl

[in] 決済逆指値

戻り値

ロング（買い）ポジションの取引高

注意事項

この関数はショート（売り）ポジションの取引高を返します。ボリュームは前の注文の結果に依存します。

© 2000–2025, MetaQuotes Ltd.

コントロールパネル及びダイアログ作成のクラス

このセクションには、グラフィックオブジェクトのコントロールパネル作成クラスの操作に関する技術的な詳細と関連するMQL5 標準ライブラリコンポーネントの説明が含まれています。

これらのクラスの使用は、MQL5 プログラム（エキスパートアドバイザー及び指標）でのコントロールパネルの作成時に時間を節約します。

MQL5 標準ライブラリはのコントロールに関連したファイルは、クライアント端末データフォルダーであるMQL5¥Include¥Controlsに配置されます。

次の記事でクラスの使用例をご覧ください。

- 任意の複雑さのレベルのグラフィカルなパネルを作成する方法
- パネルを改善してみましょう(CAPPDIALOG / CWNDCLIENTからの継承、背景の色の変更、透明性の追加)
- エキスパートアドバイザとインディケータに素早く制御パネルを追加する方法
- MQL5にて独自のグラフィカルパネルを作成する
- MQL5におけるトレーディング用コントロールパネルの作成

これらのクラスの操作をデモするエキスパートアドバイザーは MQL5¥Expert¥Examples¥Controls で見られます。

補助構造体	説明
CRect	矩形領域の構造体
CDateTime	日付と時刻操作に使用される構造体

基本クラス	説明
CWnd	全てのコントロールの基本クラス
CWndObj	コントロール及びダイアログの基本クラス
CWndContainer	（依存コントロールを含む）複雑なコントロールの基本クラス

シンプルコントロール	説明
CLabel	「Text label」グラフィックオブジェクトに基づいたコントロール
CBmpButton	「Bitmap Label（ビットマップラベル）」グラフィックオブジェクトに基づいたコントロール
CButton	「Button」グラフィックオブジェクトに基づいたコントロール
CEdit	「Edit field」グラフィックオブジェクトに基づいたコントロール

© 2000–2025, MetaQuotes Ltd.

シンプルコントロール	説明
CPanel	「Rectangle label」グラフィックオブジェクトに基づいたコントロール
CPicture	「Bitmap Label（ビットマップラベル）」グラフィックオブジェクトに基づいたコントロール

複雑なコントロール	説明
CScroll	スクロールバーの基本クラス
CScrollV	垂直方向のスクロールバー
CScrollH	水平方向のスクロールバー
CWndClient	スクロールバー付きのクライアント領域の基本クラス
CListView	ListView
CComboBox	ComboBox
CCheckBox	CheckBox
CCheckGroup	CheckGroup
CRadioButton	RadioButton
CRadioGroup	RadioGroup
CSpinEdit	SpinEdit
CDialog	ダイアログ
CAppDialog	アプリケーションダイアログ

© 2000–2025, MetaQuotes Ltd.

CRect

CRect はチャートの矩形領域のクラスです。

説明

CRect は、デカルト座標で矩形の左上と右下の角の両方の座標で定義される、領域のクラスです。

宣言

```
class CRect
```

タイトル

```
#include <Controls\Rect.mqh>
```

クラスメソッド

プロパティ	
Left	左上の隅の X 座標を取得/設定します。
Top	左上の隅の Y 座標を取得/設定します。
Right	右下の隅の X 座標を取得/設定します。
Bottom	右下の隅の Y 座標を取得/設定します。
Width	幅を取得/設定します。
Height	高さを取得/設定します。
SetBound	CRect クラスの座標を使用して領域の新しい座標を設定します。
Move	CRect クラスの新しい座標を設定します。
Shift	CRect 座標の相対的なシフトを行います。
Contains	点が CRect クラス領域の内側に位置するかどうかをチェックします。
追加メソッド	
Format	領域の座標を文字列として取得します。

© 2000-2025, MetaQuotes Ltd.

Left（Get メソッド）

左上の隅のX座標を取得します。

```
int  Left()
```

戻り値

左上の隅のX座標

Left（Set メソッド）

左上の隅のX座標を設定します。

```
void  Left(
   const int  x      // 新しい x 座標
  )
```

パラメータ

x

[in] 左上の隅の新しいX座標

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

Top（Get メソッド）

左上の隅のＹ座標を取得します。

```
int  Top()
```

戻り値

左上の隅のＹ座標

Top（Set メソッド）

左上の隅のＹ座標を設定します。

```
void  Top(
   const int  y      // y 座標
  )
```

パラメータ

y

[in] 左上の隅の新しいＹ座標

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

Right（Get メソッド）

右下の隅のX座標を取得します。

```
int  Right()
```

戻り値

右下の隅のX座標

Right（Set メソッド））

右下の隅のY座標を設定します。

```
void  Right(
   const int  x      // x 座標
  )
```

パラメータ

x

[in] 右下の隅の新しいX座標

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

Bottom（Get メソッド）

右下の隅のＹ座標を取得します。

```
int  Bottom()
```

戻り値

右下の隅のＹ座標

Bottom（Set メソッド）

右下の隅のＹ座標を設定します。

```
void  Bottom(
   const int  y      // y 座標
  )
```

パラメータ

y

[in] 右下の隅の新しいＹ座標

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

Width（Get メソッド）

領域の幅を取得します。

```
int  Width()
```

戻り値

Width of the area.

Width（Set メソッド）

領域の幅を設定します。

```
virtual bool  Width(
   const int  w      // 幅
  )
```

パラメータ

w

[in] 新しい幅

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Height（Get メソッド）

領域の高さを取得します。

```
int  Height()
```

戻り値

領域の高さ

Height（Set メソッド）

領域の高さを設定します。

```
virtual bool  Height(
   const int  h      // 高さ
  )
```

パラメータ

h

[in] 新しい高さ

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

SetBound

CRect クラスの座標を使用して領域の新しい座標を設定します。

```
void  SetBound(
   const & CRect  rect      // CRect クラス
  )
```

戻り値

なし

SetBound

領域の新しい座標を設定します。

```
void  SetBound(
   const int  l      // 左
   const int  t      // 上
   const int  r      // 右
   const int  b      // 下
  )
```

パラメータ

l

[in] 左上の隅の X 座標

t

[in] 左上の隅の Y 座標

r

[in] 右下の隅の X 座標

b

[in] 右下の隅の Y 座標

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

Move

CRect クラスの新しい座標を設定します。

```
void  Move(
   const int  x,      // X 座標
   const int  y       // Y 座標
  )
```

パラメータ

x

[in] 新しい X 座標

y

[in] 新しい Y 座標

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

Shift

CRect クラスの座標の相対的なシフトを行います。

```
void  Shift(
   const int  dx,      // X のデルタ(差)
   const int  dy       // Y のデルタ(差)
  )
```

パラメータ

dx

[in] X のデルタ(差)

dy

[in] Y のデルタ(差)

戻り値

なし

© 2000-2025, MetaQuotes Ltd.

Contains

点が CRect クラス領域の内側に位置するかどうかをチェックします。

```
bool  Contains(
   const int  x,      // X 座標
   const int  y       // Y 座標
  )
```

パラメータ

x

[in] X 座標

y

[in] Y 座標

戻り値

点が領域の内側（境界線を含む）に位置する場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Format

領域の座標を文字列として取得します。

```
string  Format(
   string &  fmt,     // フォーマット
   ) const
```

パラメータ

fmt

[in] フォーマットされた文字列

戻り値

領域の座標の文字列

© 2000–2025, MetaQuotes Ltd.

CDateTime

CDateTime は日付と時刻の操作に使用される構造体です。

説明

CDateTime 構造体は MqlDateTime から派生し、コントロールの日付と時刻の操作に使用されます。

宣言

```
struct CDateTime
```

タイトル

```
#include <Tools\DateTime.mqh>
```

クラスメソッド

プロパティ	
MonthName	月の名前を取得します。
ShortMonthName	月の短縮名を取得します。
DayName	曜日名を取得します。
ShortDayName	曜日の短縮名を取得します。
DaysInMonth	月の日数を取得します。
Get/Set methods	
DateTime	日付と時刻を取得/設定します。
Date	日付を設定します。
Time	時刻を設定します。
Sec	秒を設定します。
Min	分を設定します。
Hour	時間を設定します。
Day	月の日を設定します。
Mon	月を設定します。
Year	年を設定します。
追加メソッド	
SecDec	指定された秒数を減算します。
SecInc	指定された秒数を加算します。
MinDec	指定された分数を減算します。

© 2000-2025, MetaQuotes Ltd.

プロパティ	
MinInc	指定された分数を加算します。
HourDec	指定された時間数を減算します。
HourInc	指定された時間数を加算します。
DayDec	指定された日数を減算します。
DayInc	指定された日数を加算します。
MonDec	指定された月数を減算します。
MonInc	指定された月数を加算します。
YearDec	指定された年数を減算します。
YearInc	指定された年数を加算します。

© 2000–2025, MetaQuotes Ltd.

MonthName

月の名前を取得します。

```
string  MonthName()  const
```

Gets month name by index.

```
string  MonthName(
   const int       num          // 月のインデックス
   )  const
```

パラメータ

num

[in] 月のインデックス（１～１２）

戻り値

月の名前

© 2000-2025, MetaQuotes Ltd.

ShortMonthName

月の短縮名を取得します。

```
string  ShortMonthName()  const
```

Gets short month name by index.

```
string  ShortMonthName(
   const int       num          // 月のインデックス
   )  const
```

パラメータ

num

[in] 月のインデックス(1～12)

戻り値

月の短縮名

© 2000-2025, MetaQuotes Ltd.

DayName

曜日名を取得します。

```
string  DayName()  const
```

インデックスによって曜日名を取得します。

```
string  DayName(
   const int       num          // 曜日のインデックス
   )  const
```

パラメータ

num

[in] 曜日のインデックス(0～ 6)

戻り値

曜日の名前

© 2000–2025, MetaQuotes Ltd.

ShortDayName

曜日の短縮名を取得します。

```
string  ShortDayName()  const
```

インデックスによって曜日の短縮名を取得します。

```
string  ShortDayName(
   const int       num          // 曜日のインデックス
   )  const
```

パラメータ

num

[in] 曜日のインデックス(0～ 6)

戻り値

曜日の短縮名

© 2000-2025, MetaQuotes Ltd.

DaysInMonth

月の日数を取得します。

```
int  DaysInMonth()  const
```

戻り値

月の日数

© 2000–2025, MetaQuotes Ltd.

DateTime（Get メソッド）

日付と時刻を取得します。

```
datetime  DateTime()
```

戻り値

datetime 型の値

DateTime（ datetime 型の Set メソッド）

datetime 型の日付と時刻を設定します。

```
void  DateTime(
   const datetime      value          // 日付と時刻
  )
```

パラメータ

value

[in] datetime 型の値

戻り値

なし

DateTime（ MqlDateTime 型の Set メソッド）

MqlDateTime 型の日付と時刻を設定します。

```
void  DateTime(
   const MqlDateTime   &value         // 日付と時刻
  )
```

パラメータ

value

[in] MqlDateTime 型の日付を設定します。

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

Date(datetime 型の Set メソッド)

datetime 型の日付を設定します。

```
void  Date(
   const datetime      value          // 日付
  )
```

パラメータ

value

[in] datetime 型の値

戻り値

なし

Date(MqlDateTime 型の Set メソッド)

MqlDateTime 型の日付を設定します。

```
void  Date(
   const MqlDateTime   &value         // 日付
  )
```

パラメータ

value

[in] MqlDateTime 型の日付を設定します。

戻り値

なし

© 2000-2025, MetaQuotes Ltd.

Time(datetime 型の Set メソッド)

datetime 型の時刻を設定します。

```
void  Time(
   const datetime      value          // 時刻
  )
```

パラメータ

value

[in] datetime 型の値

戻り値

なし

Time(MqlDateTime 型の Set メソッド)

Sets time with MqlDateTime type.

```
void  Time(
   const MqlDateTime   &value         // 時刻
  )
```

パラメータ

value

[in] MqlDateTime 型の日付を設定します。

戻り値

なし

© 2000-2025, MetaQuotes Ltd.

Sec

秒を設定します。

```
void  Sec(
   const int   value        // 秒
  )
```

パラメータ

value

[in] 秒

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

Min

分を設定します。

```
void  Min(
   const int    value        // 分数
  )
```

パラメータ

value

[in] 分数

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

Hour

時間を設定します。

```
void  Hour(
   const int    value        // 時間
  )
```

パラメータ

value

[in] 時間

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

Day

月の日を設定します。

```
void  Day(
   const int   value        // 日
  )
```

パラメータ

value

[in] 月の日

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

Mon

月を設定します。

```
void  Mon(
   const int   value        // 月
  )
```

パラメータ

value

[in] 月

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

Year

年を設定します。

```
void  Year(
   const int   value        // 年
  )
```

パラメータ

value

[in] 年

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

SecDec

指定された秒数を減算します。

```
void  SecDec(
   int   delta=1        // 秒数
  )
```

パラメータ

delta

[in] 減算する秒数

戻り値

なし

© 2000-2025, MetaQuotes Ltd.

SecInc

指定された秒数を加算します。

```
void  SecInc(
   int   delta=1        // 秒数
  )
```

パラメータ

delta

[in] 加算する秒数

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

MinDec

指定された分数を減算します。

```
void  MinDec(
   int   delta=1        // 分数
  )
```

パラメータ

delta

[in] 減算する分数

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

MinInc

指定された分数を加算します。

```
void  MinInc(
   int   delta=1        // 分数
  )
```

パラメータ

delta

[in] 加算する分数

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

HourDec

指定された時間数を減算します。

```
void  HourDec(
   int   delta=1        // 時間数
  )
```

パラメータ

delta

[in] 減算する時間数

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

HourInc

指定された時間数を加算します。

```
void  HourInc(
   int   delta=1        // 時間数
  )
```

パラメータ

delta

[in] 加算する時間数

戻り値

なし

© 2000-2025, MetaQuotes Ltd.

DayDec

指定された日数を減算します。

```
void  DayDec(
   int   delta=1        // 日数
  )
```

パラメータ

delta

[in] 減算する日数

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

DayInc

指定された日数を加算します。

```
void  DayInc(
   int   delta=1        // 日数
  )
```

パラメータ

delta

[in] 加算する日数

戻り値

なし

© 2000-2025, MetaQuotes Ltd.

MonDec

指定された月数を減算します。

```
void  MonDec(
   int   delta=1        // 月数
  )
```

パラメータ

delta

[in] 減算する月数

戻り値

なし

© 2000-2025, MetaQuotes Ltd.

MonInc

指定された月数を加算します。

```
void  MonInc(
   int   delta=1        // 月数
  )
```

パラメータ

delta

[in] 加算する月数

戻り値

なし

© 2000-2025, MetaQuotes Ltd.

YearDec

指定された年数を減算します。

```
void  YearDec(
   int   delta=1        // 年数
  )
```

パラメータ

delta

[in] 減算する年数

戻り値

なし

© 2000-2025, MetaQuotes Ltd.

YearInc

指定された年数を加算します。

```
void  YearInc(
   int   delta=1        // 年数
  )
```

パラメータ

delta

[in] 加算する年数.

戻り値

なし

© 2000-2025, MetaQuotes Ltd.

CWnd

CWnd は MQL5 標準ライブラリに含まれている全てのコントロールの基本クラスです。

説明

CWnd クラスは基本コントロールクラスの実装です。

宣言

```
class CWnd : public CObject
```

タイトル

```
#include <Controls\Wnd.mqh>
```

継承階層

CObject

CWnd

直接子孫

CDragWnd, CWndContainer, CWndObj

クラスメソッド

作成と破壊	
Create	コントロールの作成
Destroy	コントロールの破壊
チャートイベントハンドラ	
OnEvent	全てのチャートイベントのハンドラ
OnMouseEvent	CHARTEVENT_MOUSE_MOVE イベントのハンドラ
名称	
Name	コントロールの名称を取得します。
コンテナへのアクセス	
ControlsTotal	コンテナのコントロール数を取得します。
Control	コントロールをインデックスで取得します。
ControlFind	コントロールを識別子で取得します。
ジオメトリ	
Rect	CRect クラスオブジェクトへのポインタを取得します。
Left	左上の隅の X 座標を取得/設定します。

© 2000-2025, MetaQuotes Ltd.

作成と破壊	
Top	左上の隅のY座標を取得/設定します。
Right	右下の隅のX座標を取得/設定します。
Bottom	右下の隅のY座標を取得/設定します。
Width	幅を取得/設定します。
Height	高さを取得/設定します。
Move	コントロールの新しい座標を設定します。
Shift	コントロール座標の相対的なシフトを行います。
Resize	コントロールの新しい幅/高さを設定します。
Contains	点/コントロールがコントロール領域の内側に位置するかどうかをチェックします。
Align	
Alignment	コントロールのアラインメントプロパティを設定します。
Align	コントロールのアラインメントを行います。
識別	
Id	コントロールの識別子を取得/設定します。
State	
IsEnabled	コントロールが有効かどうかを示す値を取得します。
Enable	コントロールが有効かどうかを示す値を設定します。
Disable	コントロールを無効にします。
IsVisible	可視性フラグをチェックします。
Visible	可視性フラグを設定します。
Show	コントロールを表示します。
Hide	コントロールを非表示にします。
IsActive	コントロールのアクティビティをチェックします。
Activate	コントロールをアクティブにします。
Deactivate	コントロールを無効にします。
状態フラグ	
StateFlags	コントロールの状態フラグを取得/設定します。
StateFlagsSet	コントロールの状態フラグを設定します。
StateFlagsReset	コントロールの状態フラグをリセットします。
Properties flags	

© 2000-2025, MetaQuotes Ltd.

作成と破壊	
PropFlags	コントロールのプロパティフラグを取得/設定します。
PropFlagsSet	コントロールのプロパティフラグを設定します。
PropFlagsReset	コントロールのプロパティフラグをリセットします。
マウス操作	
MouseX	マウスのX座標を取得/保存します。
MouseY	マウスのY座標を取得/保存します。
MouseFlags	マウスボタンの状態を取得/保存します。
MouseFocusKill	マウスのフォーカスをなくします。
内部イベントハンドラ	
OnCreate	「Create」イベントハンドラ
OnDestroy	「Destroy」イベントハンドラ
OnMove	「Move」イベントハンドラ
OnResize	「Resize」イベントハンドラ
OnEnable	「Enable」イベントハンドラ
OnDisable	「Disable」イベントハンドラ
OnShow	「Show」イベントハンドラ
OnHide	「Hide」イベントハンドラ
OnActivate	「Activate」イベントハンドラ
OnDeactivate	「Deactivate」イベントハンドラ
OnClick	「Click」イベントハンドラ
OnChange	「Change」イベントハンドラ
マウスイベントハンドラ	
OnMouseDown	「MouseDown」イベントハンドラ
OnMouseUp	「MouseUp」イベントハンドラ
ドラッグイベントハンドラ	
OnDragStart	「DragStart」イベントハンドラ
OnDragProcess	「DragProcess」イベントハンドラ
OnDragEnd	「DragEnd」イベントハンドラ
ドラッグオブジェクト	
DragObjectCreate	ドラッグオブジェクトを作成します。

© 2000–2025, MetaQuotes Ltd.

作成と破壊	
DragObjectDestroy	ドラッグオブジェクトを破壊します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

Create

コントロールを作成します。

```
virtual bool  Create(
   const long     chart,       // チャート識別子
   const string   name,        // 名称
   const int      subwin,      // チャートサブウィンドウ
   const int      x1,          // x1 座標
   const int      y1,          // y1 座標
   const int      x2,          // x2 座標
   const int      y2           // y2 座標
  )
```

パラメータ

chart

[in] チャート識別子

name

[in] コントロールの一意の名称

subwin

[in] チャートサブウィンドウ

x1

[in] 左上の隅のX座標

y1

[in] 左上の隅のY座標

x2

[in] 右下の隅のX座標

y2

[in] 右下の隅のY座標

戻り値

成功の場合はtrue、それ以外の場合はfalse

注意事項

基本クラスはパラメータを保存するのみで常にtrueを返します。

© 2000-2025, MetaQuotes Ltd.

Destroy

コントロールを破壊します。

```
virtual bool  Destroy()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnEvent

チャート イベント ハンドラ

```
virtual bool  OnEvent(
   const int      id,          // 識別子
   const long&    lparam,      // long 型のイベントパラメータ
   const double&  dparam,      // double 型のイベントパラメータ
   const string&  sparam       // string 型のイベントパラメータ
  )
```

パラメータ

id

[in] イベント識別子

lparam

[in] 参照で渡された long 型のイベント パラメータ

dparam

[in] 参照で渡された double 型のイベント パラメータ

sparam

[in] 参照で渡されたstring 型のイベント パラメータ

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnMouseEvent

マウスイベント ハンドラ（CHARTEVENT_MOUSE_MOVE チャート イベント）

```
virtual bool  OnMouseEvent(
   const int  x,          // x 座標
   const int  y,          // y 座標
   const int  flags       // フラグ
  )
```

パラメータ

x

[in] チャートの左上隅の位置に対するマウスカーソルの X 座標

y

[in] チャートの左上隅の位置に対するマウスカーソルの X 座標.

flags

[in] マウスボタン状態のフラグ

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Name

コントロールの名称を取得します。

```
string  Name()  const
```

戻り値

コントロールの名称

© 2000-2025, MetaQuotes Ltd.

ControlsTotal

コンテナのコントロール数を取得します。

```
int  ControlsTotal()  const
```

戻り値

コンテナのコントロールの数

注意事項

基本クラスのメソッドはコンテナを持たず派生クラスにコンテナへのアクセスを提供します。また、常に NULL を返します。

© 2000–2025, MetaQuotes Ltd.

Control

コントロールをインデックスで取得します。

```
CWnd*  Control(
   const int  ind      // インデックス
   )  const
```

パラメータ

ind

[in] コントロールインデックス

戻り値

コントロールのポインタ

注意事項

基本クラスのメソッドはコンテナを持たず派生クラスにコンテナへのアクセスを提供します。また、常に NULL を返します。

© 2000-2025, MetaQuotes Ltd.

ControlFind

コンテナから指定された識別子を持つコントロールを取得します。

```
virtual CWnd*  ControlFind(
   const long  id      // 識別子
  )
```

パラメータ

id

[in] 見つけるコントロールの識別子

戻り値

コントロールへのポインタ

注意事項

基本クラスのメソッドはコンテナを持たず派生クラスにコンテナへのアクセスを提供します。指定されたIDがコンテナのIDと一致すると、それ自体へのポインタが返されます。

© 2000–2025, MetaQuotes Ltd.

Rect

クラスオブジェクト へのポインタを取 得します。

```
const CRect*  Rect()  const
```

戻り値

クラスオブジェクト へのポインタ

© 2000–2025, MetaQuotes Ltd.

Left（Get メソッド）

コントロールの左上隅のX座標を取得します。

```
int  Left()
```

戻り値

コントロールの左上隅のX座標

Left（Set メソッド）

コントロールの左上隅のX座標を設定します。

```
void  Left(
   const int  x      // 新しい x 座標
  )
```

パラメータ

x

[in] 左上の隅の新しいX座標

戻り値

なし

© 2000-2025, MetaQuotes Ltd.

Top（Get メソッド）

コントロールの左上隅のＹ座標を取得します。

```
int  Top()
```

戻り値

コントロールの左上隅のＹ座標

Top（Set メソッド）

コントロールの左上隅のＹ座標を設定します。

```
void  Top(
   const int  y      // y 座標
  )
```

パラメータ

y

[in] 左上の隅の新しいＹ座標

戻り値

なし

© 2000-2025, MetaQuotes Ltd.

Right（Get メソッド）

コントロールの右下隅のY座標を取得します。

```
int  Right()
```

戻り値

右下の隅のX座標

Right（Set メソッド））

コントロールの右下隅のY座標を設定します。

```
void  Right(
   const int  x      // x 座標
  )
```

パラメータ

x

[in] 右下の隅の新しいX座標

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

Bottom（Get メソッド）

コントロールの右下隅のY座標を取得します。

```
int  Bottom()
```

戻り値

コントロールの右下隅のY座標

Bottom（Set メソッド）

コントロールの右下隅のY座標を設定します。

```
void  Bottom(
   const int  y      // y 座標
  )
```

パラメータ

y

[in] 右下の隅の新しいY座標

戻り値

なし

© 2000-2025, MetaQuotes Ltd.

Width（Get メソッド）

コントロールの幅を取得します。

```
int  Width()
```

戻り値

コントロールの幅

Width（Set メソッド）

コントロールの新しい幅を設定します。

```
virtual bool  Width(
   const int  w      // 幅
  )
```

パラメータ

w

[in] 新しい幅

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

Height（Get メソッド）

コントロールの高さを取得します。

```
int  Height()
```

戻り値

コントロールの高さ.

Height（Set メソッド）

コントロールの新しい高さを設定します。

```
virtual bool  Height(
   const int  h      // 高さ
  )
```

パラメータ

h

[in] 新しい高さ

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

Move

コントロールの新しい座標を設定します。

```
void  Move(
   const int  x,     // X 座標
   const int  y      // Y 座標
  )
```

パラメータ

x

[in] 新しい X 座標

y

[in] 新しい Y 座標

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

Shift

コントロール座標の相対的なシフトを行います。

```
void  Shift(
   const int  dx,      // X のデルタ(差)
   const int  dy       // Y のデルタ(差)
  )
```

パラメータ

dx

[in] X のデルタ(差)

dy

[in] Y のデルタ(差)

戻り値

なし

© 2000-2025, MetaQuotes Ltd.

Resize

コントロールの新しい幅/高さを設定します。

```
virtual bool  Resize(
   const int  w,      // 幅
   const int  h       // 高さ
  )
```

パラメータ

w

[in] 新しい幅

h

[in] 新しい高さ

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

Contains

点がチャートのコントロール領域の内側に位置するかどうかをチェックします。<57/66/62% >

```
bool  Contains(
   const int  x,      // X 座標
   const int  y       // Y 座標
  )
```

パラメータ

x

[in] X 座標

y

[in] Y 座標

戻り値

点が領域の内側(境界線を含む)に位置する場合は true、それ以外の場合は false

Contains

指定されたコントローがチャートのコントロール領域の内側に位置するかどうかをチェックします。

```
bool  Contains(
   const CWnd*  control       // ポインタ
   )  const
```

パラメータ

control

[in] オブジェクトポインタ

戻り値

指定されたコントロールが　領域の内側(境界線を含む)に位置する場合は　true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Alignment

コントロールのアラインメントパラメータを設定します。

```
void  Alignment(
   const int  flags,       // アラインメントフラグ
   const int  left,        // 左の境界からのオフセット
   const int  top,         // 上の境界からのオフセット
   const int  right,       // 右の境界からのオフセット
   const int  bottom       // 下の境界からのオフセット
  )
```

パラメータ

flags

[in] アラインメントフラグ

left

[in] 左の境界からの固定オフセット

top

[in] 上の境界からの固定オフセット

right

[in] 右の境界からの固定オフセット

bottom

[in] 下の境界からの固定オフセット

戻り値

なし

注意事項

アラインメントフラグ

```
enum WND_ALIGN_FLAGS
{
   WND_ALIGN_NONE=0,                                    // なし
   WND_ALIGN_LEFT=1,                                    // 左合わせ
   WND_ALIGN_TOP=2,                                     // 上合わせ
   WND_ALIGN_RIGHT=4,                                   // 右合わせ
   WND_ALIGN_BOTTOM=8,                                  // 下合わせ
   WND_ALIGN_WIDTH = WND_ALIGN_LEFT|WND_ALIGN_RIGHT,    // 幅を合わせる
   WND_ALIGN_HEIGHT=WND_ALIGN_TOP|WND_ALIGN_BOTTOM,     // 高さを合わせる
   WND_ALIGN_CLIENT=WND_ALIGN_WIDTH|WND_ALIGN_HEIGHT, // 高さと幅を合わせる
   }
```

© 2000-2025, MetaQuotes Ltd.

Align

指定されたチャート領域でコントロールのアラインメントを行います。

```
virtual bool  Align(
   const CRect*  rect      // ポインタ
  )
```

パラメータ

rect

[in] チャート領域の座標を持つオブジェクトへのポインタ

戻り値

成功の場合は true、それ以外の場合は false

注意事項

アライメントパラメータを指定する必要があります（デフォルトは、アラインメントなし）

© 2000–2025, MetaQuotes Ltd.

Id（Get メソッド）

コントロールの識別子を取得します。

```
long  Id()  const
```

戻り値

コントロールの識別子

Id（Set メソッド）

コントロールの識別子の新しい値を設定します。

```
virtual long  Id(
   const long  id      // 識別子
  )
```

パラメータ

id

[in] コントロール識別子の新しい値

戻り値

なし

© 2000-2025, MetaQuotes Ltd.

IsEnabled

コントロールが有効かどうかを示す値を取得します。

```
bool  IsEnabled()  const
```

戻り値

コントロールが有効な場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

Enable

コントロールを有効にします。

```
virtual bool  Enable()
```

戻り値

成功の場合は true、それ以外の場合は false

注意事項

有効化されたコントロールは外部イベントを処理することが出来ます。

© 2000–2025, MetaQuotes Ltd.

Disable

コントロールを無効にします。

```
virtual bool  Disable()
```

戻り値

成功の場合は true、それ以外の場合は false

注意事項

無効なコントロールは外部イベントを処理することが出来ません。

© 2000-2025, MetaQuotes Ltd.

IsVisible

コントロールが表示されているかどうかを示す値を取得します。

```
bool  IsVisible()  const
```

戻り値

コントロールがチャートに表示されている場合は true、それ以外の場合は fals.

© 2000-2025, MetaQuotes Ltd.

Visible

可視性フラグを設定します。

```
virtual bool  Visible(
   const bool  flag      // フラグ
  )
```

パラメータ

flag

[in] 新しいフラグ

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Show

コントロールを表示します。

```
virtual bool  Show()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Hide

コントロールを非表示にします。

```
virtual bool  Hide()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

IsActive

コントロールがアクティブかどうかを示す値を取得します。

```
bool  IsActive()  const
```

戻り値

コントロールがアクティブな場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

Activate

コントロールをアクティブにします。

```
virtual bool  Activate()
```

戻り値

成功の場合は true、それ以外の場合は false

注意事項

コントロールは、マウスカーソルがその上でホバリングされた時にアクティブになります。

© 2000-2025, MetaQuotes Ltd.

Deactivate

コントロールを無効にします。

```
virtual bool  Deactivate()
```

戻り値

成功の場合は true、それ以外の場合は false

注意事項

コントロールは、マウスカーソルがコントロールを離れる時に非アクティブになります。

© 2000-2025, MetaQuotes Ltd.

StateFlags（Get メソッド）

コントロールの状態フラグを取得します。

```
int  StateFlags()
```

戻り値

コントロールの状態フラグ

StateFlags（Set メソッド）

コントロールの状態フラグを設定します。

```
virtual void  StateFlags(
   const int  flags      // フラグ
  )
```

パラメータ

flags

[in] コントロールの状態フラグ

戻り値

なし

© 2000-2025, MetaQuotes Ltd.

StateFlagsSet

コントロールの状態フラグを設定します。

```
virtual void  StateFlagsSet(
   const int  flags      // フラグ
  )
```

パラメータ

flags

[in] 設定するフラグ（ビットマスク）

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

StateFlagsReset

コントロールの状態フラグをリセットします。

```
virtual void  StateFlagsReset(
   const int  flags      // フラグ
  )
```

パラメータ

flags

[in] リセットするフラグ（ビットマスク）

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

PropFlags（Get メソッド）

コントロールのプロパティフラグを取得します。

```
void  PropFlags(
   const int  flags      // フラグ
  )
```

戻り値

コントロールのプロパティフラグ

PropFlags（Set メソッド）

コントロールのプロパティフラグを設定します。

```
virtual void  PropFlags(
   const int  flags      // フラグ
  )
```

パラメータ

flags

[in] 新しいフラグ

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

PropFlagsSet

コントロールのプロパティフラグを設定します。

```
virtual void  PropFlagsSet(
   const int  flags      // フラグ
  )
```

パラメータ

flags

[in] 設定するフラグ（ビットマスク）

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

PropFlagsReset

コントロールのプロパティフラグをリセットします。

```
virtual void  PropFlagsReset(
   const int  flags      // フラグ
  )
```

パラメータ

flags

[in] リセットするフラグ（ビットマスク）

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

MouseX（Set メソッド）

マウスの X 座標を保存します。

```
void  MouseX(
   const int  value      // 座標
  )
```

パラメータ

value

[in] マウスの X 座標

戻り値

なし

MouseX（Get メソッド）

保存されたマウスの X 座標を取得します。

```
int  MouseX()
```

戻り値

保存されたマウスの X 座標

© 2000–2025, MetaQuotes Ltd.

MouseY (Set Method)

マウスのＹ座標を保存します。

```
void  MouseY(
   const int  value      // 座標
  )
```

パラメータ

value

[in] マウスのＹ座標

戻り値

なし

MouseY（Get メソッド）

保存されたマウスのＹ座標を取得します。

```
int  MouseY()
```

戻り値

保存されたマウスのＹ座標

© 2000–2025, MetaQuotes Ltd.

MouseFlags（Set メソッド）

マウスボタンの状 態を保 存します。

```
virtual void  MouseFlags(
   const int  value      // 状態
  )
```

パラメータ

value

[in] マウスボタンの状 態

戻り値

なし

MouseFlags（Get メソッド）

保 存されたマウスボタンの状 態を取 得します。

```
int  MouseFlags()
```

戻り値

マウスボタンの状 態

© 2000–2025, MetaQuotes Ltd.

MouseFocusKill

マウスボタンの保存された状態をクリアし、コントロールを無効にします。

```
bool  MouseFocusKill(
   const long  id=CONTROLS_INVALID_ID      // 識別子
  )
```

パラメータ

id=CONTROLS_INVALID_ID

[in] マウスがフォーカスされたコントロールの識別子

戻り値

コントロールの無効化の結果

© 2000-2025, MetaQuotes Ltd.

OnCreate

コントロールの「Create」イベントハンドラです。

```
virtual bool  OnCreate()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

基本クラスメソッドは何の操作も行わずに常に true を返します。

© 2000–2025, MetaQuotes Ltd.

OnDestroy

コントロールの「Destroy」イベントハンドラです。

```
virtual bool  OnDestroy()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

基本クラスメソッドは何の操作も行わずに常に true を返します。

© 2000–2025, MetaQuotes Ltd.

OnMove

コントロールの「Move」イベントハンドラです。

```
virtual bool  OnMove()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

基本クラスメソッドは何の操作も行わずに常に true を返します。

© 2000–2025, MetaQuotes Ltd.

OnResize

コントロールの「Resize」イベントハンドラです。

```
virtual bool  OnResize()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

基本クラスメソッドは何の操作も行わずに常に true を返します。

© 2000–2025, MetaQuotes Ltd.

OnEnable

コントロールの「Enable」（有効にされるとユーザインターアクションへの反応が可能になります）イベントハンドラです。

```
virtual bool  OnEnable()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

基本クラスメソッドは何の操作も行わずに常に true を返します。

© 2000-2025, MetaQuotes Ltd.

OnDisable

コントロールの「Disable」（無効にされるとユーザインターアクションへの反応が不可能になります）イベントハンドラです。

```
virtual bool  OnDisable()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

基本クラスメソッドは何の操作も行わずに常に true を返します。

© 2000-2025, MetaQuotes Ltd.

OnShow

コントロールの「Show」イベントハンドラです。

```
virtual bool  OnShow()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

基本クラスメソッドは何の操作も行わずに常に true を返します。

© 2000–2025, MetaQuotes Ltd.

OnHide

コントロールの「Hide」イベントハンドラです。

```
virtual bool  OnHide()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

基本クラスメソッドは何の操作も行わずに常に true を返します。

© 2000–2025, MetaQuotes Ltd.

OnActivate

コントロールの「Activate」イベントハンドラです。

```
virtual bool  OnActivate()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

基本クラスメソッドは何の操作も行わずに常に true を返します。

© 2000–2025, MetaQuotes Ltd.

OnDeactivate

コントロールの「Deactivate」イベントハンドラです。

```
virtual bool  OnDeactivate()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

基本クラスメソッドは何の操作も行わずに常に true を返します。

© 2000–2025, MetaQuotes Ltd.

OnClick

コントロールの「Click」(マウスの左ボタンのクリック) イベントハンドラです。

```
virtual bool  OnClick()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnChange

コントロールの「Change」イベントハンドラです。

```
virtual bool  OnChange()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

基本クラスメソッドは何の操作も行わずに常に true を返します。

© 2000–2025, MetaQuotes Ltd.

OnMouseDown

コントロールの「MouseDown」イベントハンドラです。

```
virtual bool  OnMouseDown()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

「MouseDown」イベントは、コントロールの上で左マウスボタンが押された時に発生します。

© 2000-2025, MetaQuotes Ltd.

OnMouseUp

コントロールの「MouseUp」(マウスの左ボタンの開放)イベントハンドラです。

```
virtual bool  OnMouseUp()
```

戻り値

イベントが処理された場合はtrue、それ以外の場合はfalse

注意事項

「MouseUp」イベントは、コントロールの上でマウスの左ボタンが解放された時に発生します。

© 2000-2025, MetaQuotes Ltd.

OnDragStart

コントロールの「DragStart」イベントハンドラです。

```
virtual bool  OnDragStart()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

「DragStart」イベントは、ドラッグの開始時に発生します。

© 2000–2025, MetaQuotes Ltd.

OnDragProcess

コントロールの「DragProcess」イベントハンドラです。

```
virtual bool  OnDragProcess(
   const int  x,      // x 座標
   const int  y       // y 座標
  )
```

パラメータ

x

[in] マウスカーソルの現在の X 座標

y

[in] マウスカーソルの現在の Y 座標

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

「DragProcess」イベントは、コントロールの移動時に発生します。

© 2000–2025, MetaQuotes Ltd.

OnDragEnd

コントロールの「DragEnd」イベントハンドラです。

```
virtual bool  OnDragEnd()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

「DragEnd」イベントは、コントロールのドラッグの終了時に発生します。

© 2000–2025, MetaQuotes Ltd.

DragObjectCreate

ドラッグオブジェクトを作成します。

```
virtual bool  DragObjectCreate()
```

戻り値

成功の場合は true、それ以外の場合は false

注意事項

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

DragObjectDestroy

ドラッグオブジェクトを破壊します。

```
virtual bool  DragObjectDestroy()
```

戻り値

成功の場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

CWndObj

CWndObj 標準ライブラリのシンプルコントロールの基本クラスです。

説明

CWndObj クラスはシンプルコントロールの基本メソッドを実装します。

宣言

```
class CWndObj : public CWnd
```

タイトル

```
#include <Controls\WndObj.mqh>
```

継承階層

CObject
  CWnd
    CWndObj

直接子孫

CBmpButton, CButton, CEdit, CLabel, CPanel, CPicture

クラスメソッド

チャートイベント処理	
OnEvent	全てのチャートイベントのハンドラ
プロパティ	
Text	チャートオブジェクトの OBJPROP_TEXT プロパティを取得/設定します。
Color	チャートオブジェクトの OBJPROP_COLOR プロパティを取得/設定します。
ColorBackground	チャートオブジェクトの OBJPROP_BGCOLOR プロパティを取得/設定します。
ColorBorder	チャートオブジェクトの OBJPROP_BORDER_COLORプロパティを取得/設定します。
Font	チャートオブジェクトの OBJPROP_FONT プロパティを取得/設定します。
FontSize	チャートオブジェクトの OBJPROP_FONTSIZE プロパティを取得/設定します。
ZOrder	チャートオブジェクトの OBJPROP_ZORDER プロパティを取得/設定します。

© 2000-2025, MetaQuotes Ltd.

チャート イベント 処 理	
チャート オブジェクト イベント ハンドラ	
OnObjectCreate	CHARTEVENT_OBJECT_CREATE イベント ハンドラ
OnObjectChange	CHARTEVENT_OBJECT_CHANGE イベント ハンドラ
OnObjectDelete	CHARTEVENT_OBJECT_DELETE イベント ハンドラ
OnObjectDrag	CHARTEVENT_OBJECT_DRAG イベント ハンドラ
プロパティ変 更 イベント ハンドラ	
OnSetText	「SetText」イベント ハンドラ
OnSetColor	「SetColor」イベント ハンドラ
OnSetColorBackground	「SetColorBackground」イベント ハンドラ
OnSetFont	「SetFont」イベント ハンドラ
OnSetFontSize	「SetFontSize」イベント ハンドラ
OnSetZOrder	「SetZOrder」イベント ハンドラ
内 部 イベント ハンドラ	
OnDestroy	「Destroy」イベント ハンドラ
OnChange	「Change」イベント ハンドラ

クラスから継 承 されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

クラスから継 承 されたメソッド CWnd

Create, Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

© 2000–2025, MetaQuotes Ltd.

OnEvent

チャート イベント ハンドラ

```
virtual bool  OnEvent(
   const int      id,          // 識別子
   const long&    lparam,      // long 型のイベントパラメータ
   const double&  dparam,      // double 型のイベントパラメータ
   const string&  sparam       // string 型のイベントパラメータ
  )
```

パラメータ

id

[in] イベント識別子

lparam

[in] 参照で渡された long 型のイベント パラメータ

dparam

[in] 参照で渡された double 型のイベント パラメータ

sparam

[in] 参照で渡された string 型のイベント パラメータ

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Text（Get メソッド）

チャート オブジェクト の OBJPROP_TEXT （テキスト ）プロパティを取 得します。

```
string  Text()
```

戻 り値

OBJPROP_TEXT プロパティの値 .

Text（Set メソッド）

チャート オブジェクト の OBJPROP_TEXT （テキスト ）プロパティを設 定します。

```
bool  Text(
   const string  value      // 新しい値
  )
```

パラメータ

value

[in] OBJPROP_TEXT プロパティの新 しい値

戻 り値

成 功 の場 合 は true、それ以 外 の場 合 は false

© 2000–2025, MetaQuotes Ltd.

Color（Get メソッド）

チャート オブジェクト の OBJPROP_COLOR （色 ）プロパティを取 得します。

```
color  Color()
```

戻り値

OBJPROP_COLOR プロパティの値

Color（Set メソッド）

チャート オブジェクト の OBJPROP_COLOR（色 ）プロパティを設 定します。

```
bool  Color(
   const color  value      // 値
  )
```

パラメータ

value

[in] OBJPROP_COLOR プロパティの新しい値

戻り値

成 功 の場 合 は true、それ以 外 の場 合 は false

© 2000–2025, MetaQuotes Ltd.

ColorBackground（Get メソッド）

チャート オブジェクト の OBJPROP_BGCOLOR（背 景 色）プロパティを取 得します。

```
color  ColorBackground()
```

戻り値

OBJPROP_BGCOLOR プロパティの値

ColorBackground（Set メソッド）

チャート オブジェクト の OBJPROP_BGCOLOR（背 景 色）プロパティを設 定します。

```
bool  ColorBackground(
   const color  value      // 値
  )
```

パラメータ

value

[in] OBJPROP_BGCOLOR プロパティの新しい値

戻り値

成 功 の場 合 は true、それ以 外 の場 合 は false

© 2000–2025, MetaQuotes Ltd.

ColorBorder（Get メソッド）

チャート オブジェクト の OBJPROP_BORDER_COLOR（境 界 色）プロパティを取 得します。

```
color  ColorBorder()
```

戻り値

OBJPROP_BORDER_COLOR プロパティの値

ColorBorder（Set メソッド）

チャート オブジェクト の OBJPROP_BORDER_COLOR（境 界 色）プロパティを設 定します。

```
bool  ColorBorder(
   const color  value      // 値
  )
```

パラメータ

value

[in] OBJPROP_BORDER_COLOR プロパティの新しい値

戻り値

成 功 の場 合 は true、それ以 外 の場 合 は false

© 2000–2025, MetaQuotes Ltd.

Font（Get メソッド）

チャート オブジェクト の OBJPROP_FONT （フォント ）プロパティを取 得します。

```
string  Font()
```

戻り値

OBJPROP_FONT プロパティの値

Font（Set メソッド）

チャート オブジェクト OBJPROP_FONT （フォント ）プロパティを取 得します。

```
bool  Font(
   const string  value      // 新しい値
  )
```

パラメータ

value

[in] OBJPROP_FONT プロパティの新しい値

戻り値

成 功 の場 合 は true、それ以 外 の場 合 は false

© 2000–2025, MetaQuotes Ltd.

FontSize（Get メソッド）

チャート オブジェクト の OBJPROP_FONTSIZE （フォント サイズ）プロパティを取 得します。

```
int  FontSize()
```

戻り値

OBJPROP_FONTSIZE プロパティの値

FontSize（Set メソッド）

チャート オブジェクト OBJPROP_FONTSIZE （フォント サイズ）プロパティを取 得します。

```
bool  FontSize(
   const int  value      // 新しい値
  )
```

パラメータ

value

[in] OBJPROP_FONTSIZE プロパティの新しい値

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

ZOrder（Get メソッド）

チャート オブジェクト の OBJPROP_ZORDER プロパティを取 得します。

```
long  ZOrder()
```

戻り値

OBJPROP_ZORDER プロパティの値

ZOrder（Set メソッド）

チャート オブジェクト の OBJPROP_ZORDER プロパティを設 定します。

```
bool  ZOrder(
   const long  value      // 新しい値
  )
```

パラメータ

value

[in] OBJPROP_ZORDER プロパティの新しい値

戻り値

成 功 の場 合 は true、それ以 外 の場 合 は false

© 2000–2025, MetaQuotes Ltd.

OnObjectCreate

チャートオブジェクト CHARTEVENT_OBJECT_CREATE イベントハンドラです。

```
virtual bool  OnObjectCreate()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

基本クラスメソッドは何の操作も行わずに常に true を返します。

© 2000–2025, MetaQuotes Ltd.

OnObjectChange

チャート オブジェクト CHARTEVENT_OBJECT_CHANGE イベント ハンドラです。

```
virtual bool  OnObjectChange()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnObjectDelete

チャート オブジェクト CHARTEVENT_OBJECT_DELETE イベント ハンドラです。

```
virtual bool  OnObjectDelete()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnObjectDrag

チャート オブジェクト CHARTEVENT_OBJECT_DRAG イベント ハンドラです。

```
virtual bool  OnObjectDrag()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnSetText

コントロールの「SetText(OBJPROP_TEXT プロパティの変更) イベントハンドラです。

```
virtual bool  OnSetText()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

基本クラスメソッドは何の操作も行わずに常に true を返します。

© 2000–2025, MetaQuotes Ltd.

OnSetColor

コントロールの「SetColor」(OBJPROP_COLOR プロパティの変更)イベントハンドラです。

```
virtual bool  OnSetColor()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

基本クラスメソッドは何の操作も行わずに常に true を返します。

© 2000-2025, MetaQuotes Ltd.

OnSetColorBackground

コントロールの「SetColorBackground（OBJPROP_BGCOLOR プロパティの変更）イベントハンドラです。

```
virtual bool  OnSetColorBackground()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

基本クラスメソッドは何の操作も行わずに常に true を返します。

© 2000–2025, MetaQuotes Ltd.

OnSetFont

コントロールの「SetFont(OBJPROP_FONTプロパティの変更)イベントハンドラです。

```
virtual bool  OnSetFont()
```

戻り値

イベントが処理された場合はtrue、それ以外の場合はfalse

注意事項

基本クラスメソッドは何の操作も行わずに常にtrueを返します。

© 2000-2025, MetaQuotes Ltd.

OnSetFontSize

コントロールの「SetFontSize（OBJPROP_FONTSIZE プロパティの変更）イベントハンドラです。

```
virtual bool  OnSetFontSize()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

基本クラスメソッドは何の操作も行わずに常に true を返します。

© 2000-2025, MetaQuotes Ltd.

OnSetZOrder

コントロールの「OBJPROP_ZORDER プロパティの変更」イベントハンドラです。

```
virtual bool  OnSetZOrder()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

基本クラスメソッドは何の操作も行わずに常に true を返します。

© 2000–2025, MetaQuotes Ltd.

OnDestroy

コントロールの「Destroy」イベントハンドラです。

```
virtual bool  OnDestroy()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnChange

コントロールの「Change」イベントハンドラです。

```
virtual bool  OnChange()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

CWndContainer

CWndContainer は標準ライブラリの（依存コントロールを含む）複雑なコントロールのクラスです。

説明

CWndContainer クラスは複雑なコントロールの基本メソッドを実装します。

宣言

```
class CWndContainer : public CWnd
```

タイトル

```
#include <Controls\WndContainer.mqh>
```

継承階層

CObject
  CWnd
    CWndContainer

直接子孫

CCheckBox, CComboBox, CDateDropList, CDatePicker, CDialog, CRadioButton, CScroll, CSpinEdit, CWndClient

クラスメソッド

Destroy	
Destroy	全てのコンテナコントロールを破壊します。
チャートイベントハンドラ	
OnEvent	全てのチャートイベントのハンドラ
OnMouseEvent	CHARTEVENT_MOUSE_MOVE イベントハンドラ
コンテナへのアクセス	
ControlsTotal	コンテナのコントロール数を取得します。
Control	コントロールをインデックスで取得します。
ControlFind	コントロールを識別子で取得します。
追加と削除	
Add	コンテナにコントロールを追加します。
Delete	コンテナからコントロールを削除します。
ジオメトリ	
Move	コンテナの全てのコントロールに新しい座標を設定します。

© 2000-2025, MetaQuotes Ltd.

Destroy	
Shift	コンテナの全てのコントロール座標の相対的なシフトを行います。
識別	
Id	コンテナの全てのコントロールに識別子を設定します。
State	
Enable	コンテナの全てのコントロールを有効にします。
Disable	コンテナの全てのコントロールを無効にします。
Show	コンテナの全てのコントロールを表示します。
Hide	コンテナの全てのコントロールを非表示にします。
マウス操作	
MouseFocusKill	マウスのフォーカスをなくします。
ファイル操作	
Save	コンテナ情報をファイルに保存します。
Load	コンテナ情報をファイルから読み込みます。
内部イベントハンドラ	
OnResize	「Resize」イベントハンドラ
OnActivate	「Activate」イベントハンドラ
OnDeactivate	「Deactivate」イベントハンドラ

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Type, Compare

クラスから継承されたメソッド CWnd

Create, Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

© 2000–2025, MetaQuotes Ltd.

Destroy

全てのコンテナコントロールを破壊します。

```
virtual bool  Destroy()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnEvent

チャート イベント ハンドラ

```
virtual bool  OnEvent(
   const int      id,          // 識別子
   const long&    lparam,      // long 型のイベントパラメータ
   const double&  dparam,      // double 型のイベントパラメータ
   const string&  sparam       // string 型のイベントパラメータ
  )
```

パラメータ

id

[in] イベント識別子

lparam

[in] 参照で渡された long 型のイベント パラメータ

dparam

[in] 参照で渡された double 型のイベント パラメータ

sparam

[in] 参照で渡された string 型のイベント パラメータ

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnMouseEvent

マウスイベントハンドラ

```
virtual bool  OnMouseEvent(
   const int  x,          // x 座標
   const int  y,          // y 座標
   const int  flags       // フラグ
  )
```

パラメータ

x

[in] チャートの左上隅の位置に対するマウスカーソルの X 座標

y

[in] チャートの左上隅の位置に対するマウスカーソルの X 座標.

flags

[in] マウスボタン状態のフラグ

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

ControlsTotal

コンテナのコントロール数を取得します。

```
int  ControlsTotal()  const
```

戻り値

コンテナのコントロールの数

© 2000–2025, MetaQuotes Ltd.

Control

コントロールをコンテナからインデックスで取得します。

```
CWnd*  Control(
   const int  ind      // インデックス
   )  const
```

パラメータ

ind

[in] 必要なコントロールのインデックス

戻り値

コントロールへのポインタ、コントロールが見つからなかった場合は NULL

© 2000-2025, MetaQuotes Ltd.

ControlFind

コントロールをコンテナから識別子で取得します。

```
virtual CWnd*  ControlFind(
   const long  id      // 識別子
  )
```

パラメータ

id

[in] コントロールの識別子

戻り値

コントロールへのポインタ、コントロールが見つからなかった場合は NULL

© 2000-2025, MetaQuotes Ltd.

Add

コンテナにコントロールを追加します。

```
bool  Add(
   CWnd&  control      // 参照
  )
```

パラメータ

control

[in] 参照で渡された追加するコントロール

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Delete

コンテナからコントロールを削除します。

```
bool  Delete(
   CWnd&  control      // 参照
  )
```

パラメータ

control

[in] 参照で渡された削除するコントロール

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Move

コンテナの全てのコントロールに新しい座標を設定します。

```
virtual bool  Move(
   const int  x,      // x 座標
   const int  y       // y 座標
  )
```

パラメータ

x

[in] 左上の隅の新しい X 座標

y

[in] 左上の隅の新しい Y 座標

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Shift

コンテナの全てのコントロール座標の相対的なシフトを行います。

```
virtual bool  Shift(
   const int  dx,      // X のデルタ(差)
   const int  dy       // Y のデルタ(差)
  )
```

パラメータ

dx

[in] X のデルタ(差)

dy

[in] Y のデルタ(差)

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

Id

コンテナの全てのコントロールに識別子を設定します。

```
virtual long  Id(
   const long  id      // 識別子
  )
```

パラメータ

id

[in] 基本グループの識別子.

戻り値

コンテナコントロールに使用される識別子の数.

© 2000–2025, MetaQuotes Ltd.

Enable

コンテナの全てのコントロールを有効にします。

```
virtual bool  Enable()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Disable

コンテナの全てのコントロールを無効にします。

```
virtual bool  Disable()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Show

コンテナの全てのコントロールを表示します。

```
virtual bool  Show()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Hide

コンテナの全てのコントロールを非表示にします。

```
virtual bool  Hide()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

MouseFocusKill

マウスボタンの保存された状態をクリアし、コントロールを無効にします。

```
bool  MouseFocusKill(
   const long  id=CONTROLS_INVALID_ID      // 識別子
  )
```

パラメータ

id=CONTROLS_INVALID_ID

[in] マウスがフォーカスされたコントロールの識別子

戻り値

コントロールの無効化の結果

© 2000-2025, MetaQuotes Ltd.

Save

コンテナ情報をファイルに保存します。

```
virtual bool  Save(
   const int  file_handle      // ハンドル
  )
```

パラメータ

file_handle

[in] バイナリファイルのハンドル（書き込みのために開かれた事が必要）

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Load

コンテナ情報をファイルから読み込みます。

```
virtual bool  Load(
   const int  file_handle      // ハンドル
  )
```

パラメータ

file_handle

[in] バイナリファイルのハンドル（読み込みのために開かれた事が必要）

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnResize

コントロールの「Resize」イベントハンドラです。

```
virtual bool  OnResize()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

基本クラスメソッドは何の操作も行わずに常に true を返します。

© 2000–2025, MetaQuotes Ltd.

OnActivate

コントロールの「Activate」イベントハンドラです。

```
virtual bool  OnActivate()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

基本クラスメソッドは何の操作も行わずに常に true を返します。

© 2000–2025, MetaQuotes Ltd.

OnDeactivate

コントロールの「Deactivate」イベントハンドラです。

```
virtual bool  OnDeactivate()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

基本クラスメソッドは何の操作も行わずに常に true を返します。

© 2000–2025, MetaQuotes Ltd.

CLabel

CLabel は「Text label」チャート オブジェクト に基 づいたシンプルコント ロールのクラスです。

説 明

CLabel はシンプルなラベルの作 成 に意 図 されています。

宣 言

```
class CLabel : public CWndObj
```

タイト ル

```
#include <Controls\Label.mqh>
```

継 承 階 層

CObject
CWnd
CWndObj
CLabel

下 記 のコード の結 果 :



クラスメソッド

Create	
Create	コント ロールの作 成

© 2000–2025, MetaQuotes Ltd.

Create	
プロパティ変更イベントハンドラ	
OnSetText	「SetText」イベントハンドラ
OnSetColor	「SetColor」イベントハンドラ
OnSetFont	「SetFont」イベントハンドラ
OnSetFontSize	「SetFontSize」イベントハンドラ
内部イベントハンドラ	
OnCreate	「Create」イベントハンドラ
OnShow	「Show」イベントハンドラ
OnHide	「Hide」イベントハンドラ
OnMove	「Move」イベントハンドラ

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

クラスから継承されたメソッド CWnd

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

クラスから継承されたメソッド CWndObj

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

テキストラベルを使ったパネルの作成例：

```
//+------------------------------------------------------------------+
//|                                                ControlsLabel.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CLabel"
#include <Controls\Dialog.mqh>
#include <Controls\Label.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
```

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with
allowance for border width)
#define INDENT_TOP                          (11)      // indent from top (with
allowance for border width)
#define INDENT_RIGHT                        (11)      // indent from right (with
allowance for border width)
#define INDENT_BOTTOM                       (11)      // indent from bottom (with
allowance for border width)
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CLabel            m_label;                       // CLabel object
public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2);
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double
&dparam,const string &sparam);
protected:
   //--- create dependent controls
   bool              CreateLabel(void);
   //--- handlers of the dependent controls events
   void              OnClickLabel(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
```

© 2000–2025, MetaQuotes Ltd.

```
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2)
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateLabel())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "CLabel"                                              |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateLabel(void)
  {
//--- coordinates
   int x1=INDENT_RIGHT;
   int y1=INDENT_TOP+CONTROLS_GAP_Y;
   int x2=x1+100;
   int y2=y1+20;
//--- create
   if(!m_label.Create(m_chart_id,m_name+"Label",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_label.Text("Label"))
      return(false);
   if(!Add(m_label))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
```

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000–2025, MetaQuotes Ltd.

Create

新しい CLabel コントロールを作成します。

```
virtual bool  Create(
   const long     chart,       // チャート識別子
   const string   name,        // 名称
   const int      subwin,      // チャートサブウィンドウ
   const int      x1,          // x1 座標
   const int      y1,          // y1 座標
   const int      x2,          // x2 座標
   const int      y2           // y2 座標
  )
```

パラメータ

chart

[in] チャート識別子

name

[in] コントロールの一意の名称

subwin

[in] チャートサブウィンドウ

x1

[in] 左上の隅の X 座標

y1

[in] 左上の隅の Y 座標

x2

[in] 右下の隅の X 座標

y2

[in] 右下の隅の Y 座標

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnSetText

コントロールの「SetText(OBJPROP_TEXT プロパティの変更) イベントハンドラです。

```
virtual bool  OnSetText()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnSetColor

コントロールの「SetColor（OBJPROP_COLOR プロパティの変更）イベントハンドラです。

```
virtual bool  OnSetColor()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnSetFont

コントロールの「SetFont（OBJPROP_FONTプロパティの変更）イベントハンドラです。

```
virtual bool  OnSetFont()
```

戻り値

イベントが処理された場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

OnSetFontSize

コントロールの「SetFontSize（OBJPROP_FONTSIZE プロパティの変更）イベントハンドラです。

```
virtual bool  OnSetFontSize()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnCreate

コントロールの「Create」イベントハンドラです。

```
virtual bool  OnCreate()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnShow

コントロールの「Show」イベントハンドラです。

```
virtual bool  OnShow()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnHide

コントロールの「Hide」イベントハンドラです。

```
virtual bool  OnHide()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnMove

コントロールの「Move」イベントハンドラです。

```
virtual bool  OnMove()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CBmpButton

CBmpButton は「Bitmap Label（ビットマップラベル）」チャートオブジェクトに基づいたシンプルコントロールのクラスです。

説明

CBmpButton は、グラフィックイメージのボタン作成に意図されています。

宣言

```
class CBmpButton : public CWndObj
```

タイトル

```
#include <Controls\BmpButton.mqh>
```

継承階層

CObject
CWnd
CWndObj
CBmpButton

下記のコードの結果：



クラスメソッド

Create	
Create	コントロールの作成

© 2000–2025, MetaQuotes Ltd.

Create	
プロパティ	
Border	コントロールの「Border」プロパティを取得/設定します。
BmpNames	コントロールの bmp ファイルの名称を設定します。
BmpOffName	OFF 状態での bmp ファイルの名称を取得/設定します。
BmpOnName	ON 状態での bmp ファイルの名称を取得/設定します。
BmpPassiveName	パッシブ状態での bmp ファイルの名称を取得/設定します。
BmpActiveName	アクティブ状態での bmp ファイルの名称を取得/設定します。
State	
Pressed	コントロールの状態を取得/設定します。
Locking	コントロールの「Locking」プロパティを取得/設定します。
内部イベントハンドラ	
OnSetZOrder	「SetZOrder」イベントハンドラ
OnCreate	「Create」イベントハンドラ
OnShow	「Show」イベントハンドラ
OnHide	「Hide」イベントハンドラ
OnMove	「Move」イベントハンドラ
OnChange	「Change」イベントハンドラ
OnActivate	「Activate」イベントハンドラ
OnDeactivate	「Deactivate」イベントハンドラ
OnMouseDown	「MouseDown」イベントハンドラ
OnMouseUp	「MouseUp」イベントハンドラ

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

クラスから継承されたメソッド CWnd

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

クラスから継承されたメソッド CWndObj

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

ビットマップラベルを使ったパネルの作成例：

```
//+------------------------------------------------------------------+
//|                                              ControlsBmpButton.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CBmpButton"
#include <Controls\Dialog.mqh>
#include <Controls\BmpButton.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with
allowance for border width)
#define INDENT_TOP                          (11)      // indent from top (with
allowance for border width)
#define INDENT_RIGHT                        (11)      // indent from right (with
allowance for border width)
#define INDENT_BOTTOM                       (11)      // indent from bottom (with
allowance for border width)
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
```

© 2000–2025, MetaQuotes Ltd.

```
   CBmpButton        m_bmpbutton1;                    // CBmpButton object
   CBmpButton        m_bmpbutton2;                    // CBmpButton object

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2);
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double
&dparam,const string &sparam);

protected:
   //--- create dependent controls
   bool              CreateBmpButton1(void);
   bool              CreateBmpButton2(void);
   //--- handlers of the dependent controls events
   void              OnClickBmpButton1(void);
   void              OnClickBmpButton2(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CLICK,m_bmpbutton1,OnClickBmpButton1)
ON_EVENT(ON_CLICK,m_bmpbutton2,OnClickBmpButton2)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2)
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
```

© 2000–2025, MetaQuotes Ltd.

```
   if(!CreateBmpButton1())
      return(false);
   if(!CreateBmpButton2())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "BmpButton1" button                                   |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateBmpButton1(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- create
   if(!m_bmpbutton1.Create(m_chart_id,m_name+"BmpButton1",m_subwin,x1,y1,x2,y2))
      return(false);
//--- sets the name of bmp files of the control CBmpButton
   m_bmpbutton1.BmpNames("\\Images\\euro.bmp","\\Images\\dollar.bmp");
   if(!Add(m_bmpbutton1))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "BmpButton2" fixed button                             |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateBmpButton2(void)
  {
//--- coordinates
   int x1=INDENT_LEFT+2*(BUTTON_WIDTH+CONTROLS_GAP_X);
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- create
   if(!m_bmpbutton2.Create(m_chart_id,m_name+"BmpButton2",m_subwin,x1,y1,x2,y2))
      return(false);
//--- sets the name of bmp files of the control CBmpButton
   m_bmpbutton2.BmpNames("\\Images\\euro.bmp","\\Images\\dollar.bmp");
   if(!Add(m_bmpbutton2))
      return(false);
   m_bmpbutton2.Locking(true);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000–2025, MetaQuotes Ltd.

```
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickBmpButton1(void)
  {
   Comment(__FUNCTION__);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickBmpButton2(void)
  {
   if(m_bmpbutton2.Pressed())
      Comment(__FUNCTION__+" State of the control is: On");
   else
      Comment(__FUNCTION__+" State of the control is: Off");
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
```

© 2000–2025, MetaQuotes Ltd.

```
                  const string& sparam) // event parameter of the string type
{
 ExtDialog.ChartEvent(id,lparam,dparam,sparam);
}
```

© 2000–2025, MetaQuotes Ltd.

Create

新しい CBmpButton コントロールを作成します。

```
virtual bool  Create(
   const long     chart,       // チャート識別子
   const string   name,        // 名称
   const int      subwin,      // チャートサブウィンドウ
   const int      x1,          // x1 座標
   const int      y1,          // y1 座標
   const int      x2,          // x2 座標
   const int      y2           // y2 座標
  )
```

パラメータ

chart

[in] チャート識別子

name

[in] コントロールの一意の名称

subwin

[in] チャートサブウィンドウ

x1

[in] 左上の隅の X 座標

y1

[in] 左上の隅の Y 座標

x2

[in] 右下の隅の X 座標

y2

[in] 右下の隅の Y 座標

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

Border（Get メソッド）

コントロールの「Border（境界の幅）」プロパティを取得します。

```
int  Border()  const
```

戻り値

「Border」プロパティ

Border（Set メソッド）

コントロールの「Border（境界の幅）」プロパティを設定します。

```
bool  Border(
   const int  value      // 新しい値
  )
```

パラメータ

value

[in]「Border」プロパティの新しい値

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

BmpNames

コントロールの bmp ファイルの名称を設定します。

```
bool  BmpNames(
   const string  off="",      // ファイル名
   const string  on=""        // ファイル名
  )
```

パラメータ

off=""

[in] OFF 状態での bmp ファイルの名称

on=""

[in] ON 状態での bmp ファイルの名称

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

BmpOffName（Get メソッド）

OFF 状態での bmp ファイルの名称を取得します。

```
string  BmpOffName()  const
```

戻り値

OFF 状態での bmp ファイルの名称

BmpOffName（Set メソッド）

OFF 状態での bmp ファイルの名称を設定します。

```
bool  BmpOffName(
   const string  name      // ファイル名
  )
```

パラメータ

name

[in] OFF 状態での bmp ファイルの名称

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

BmpOnName（Get メソッド）

ON 状態での bmp ファイルの名称を取得します。

```
string  BmpOnName()  const
```

戻り値

ON 状態での bmp ファイルの名称

BmpOnName（Set メソッド）

ON 状態での bmp ファイルの名称を設定します。

```
bool  BmpOnName(
   const string  name      // ファイル名
  )
```

パラメータ

name

[in] ON 状態での bmp ファイルの名称

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

BmpPassiveName（Get メソッド）

パッシブ状態での bmp ファイルの名称を取得します。

```
string  BmpPassiveName()  const
```

戻り値

パッシブ状態での bmp ファイルの名称

BmpPassiveName（Set メソッド）

パッシブ状態での bmp ファイルの名称を設定します。

```
bool  BmpPassiveName(
   const string  name      // ファイル名
  )
```

パラメータ

name

[in] パッシブ状態での bmp ファイルの名称

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

BmpActiveName（Get メソッド）

アクティブ状態での bmp ファイルの名称を取得します。

```
string  BmpActiveName()  const
```

戻り値

アクティブ状態での bmp ファイルの名称

注意事項

コントロールは、マウスカーソルがその上でホバリングされた時にアクティブになります。

BmpActiveName（Set メソッド）

アクティブ状態での bmp ファイルの名称を設定します。

```
bool  BmpActiveName(
   const string  name       // ファイル名
  )
```

パラメータ

name

[in] アクティブ状態での bmp ファイルの名称

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Pressed（Get メソッド）

コントロールの状態（「Pressed」プロパティ）を取得します。

```
bool  Pressed()  const
```

戻り値

コントロールの状態

Pressed（Set メソッド）

コントロールの状態（「Pressed」プロパティ）を設定します。

```
bool  Pressed(
   const bool  pressed      // 新しい状態
  )
```

パラメータ

pressed

[in] コントロールの新し状態

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

Locking（Get メソッド）

コントロールの「Locking」プロパティを取得します。

```
bool  Locking()  const
```

戻り値

「Locking」プロパティの値

Locking（Set メソッド）

コントロールの「Locking」プロパティを設定します。

```
void  Locking(
   const bool  locking      // 新しい値
  )
```

パラメータ

locking

[in] 「Locking」プロパティの新しい値

戻り値

なし

© 2000-2025, MetaQuotes Ltd.

OnSetZOrder

コントロールの「SetZOrder（OBJPROP_ZORDER プロパティの変更）イベントハンドラです。

```
virtual bool  OnSetZOrder()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnCreate

コントロールの「Create」イベントハンドラです。

```
virtual bool  OnCreate()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnShow

コントロールの「Show」イベントハンドラです。

```
virtual bool  OnShow()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnHide

コントロールの「Hide」イベントハンドラです。

```
virtual bool  OnHide()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnMove

コントロールの「Move」イベントハンドラです。

```
virtual bool  OnMove()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnChange

コントロールの「Change」イベントハンドラです。

```
virtual bool  OnChange()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnActivate

コントロールの「Activate」イベントハンドラです。

```
virtual bool  OnActivate()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnDeactivate

コントロールの「Deactivate」イベントハンドラです。

```
virtual bool  OnDeactivate()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnMouseDown

コントロールの「MouseDown」イベントハンドラです。

```
virtual bool  OnMouseDown()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

「MouseDown」イベントは、コントロールの上で左マウスボタンが押された時に発生します。

© 2000-2025, MetaQuotes Ltd.

OnMouseUp

コントロールの「MouseUp」(マウスの左ボタンの開放)イベントハンドラです。

```
virtual bool  OnMouseUp()
```

戻り値

イベントが処理された場合はtrue、それ以外の場合はfalse

注意事項

「MouseUp」イベントは、コントロールの上でマウスの左ボタンが解放された時に発生します。

© 2000-2025, MetaQuotes Ltd.

CButton

CButton は「Button」チャート オブジェクト に基 づいたシンプルコント ロールのクラスです。

説 明

CButton はシンプルなボタンの作 成 に意 図 されています。

宣 言

```
class CButton : public CWndObj
```

タイト ル

```
#include <Controls\Button.mqh>
```

継 承 階 層

CObject
CWnd
CWndObj
CButton

下 記 のコード の結 果 :



クラスメソッド

Create	
Create	コント ロールの作 成

© 2000–2025, MetaQuotes Ltd.

Create	
State	
Pressed	「Pressed」プロパティを取得/設定します。
Locking	「Locking」プロパティを取得/設定します。
プロパティ変更イベントハンドラ	
OnSetText	「SetText」イベントハンドラ
OnSetColor	「SetColor」イベントハンドラ
OnSetColorBackground	「SetColorBackground」イベントハンドラ
OnSetColorBorder	「SetColorBorder」イベントハンドラ
OnSetFont	「SetFont」イベントハンドラ
OnSetFontSize	「SetFontSize」イベントハンドラ
内部イベントハンドラ	
OnCreate	「Create」イベントハンドラ
OnShow	「Show」イベントハンドラ
OnHide	「Hide」イベントハンドラ
OnMove	「Move」イベントハンドラ
OnResize	「Resize」イベントハンドラ
OnMouseDown	「MouseDown」イベントハンドラ
OnOnMouseUp	「MouseUp」イベントハンドラ

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

クラスから継承されたメソッド CWnd

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

クラスから継承されたメソッド CWndObj

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

ボタンを使ったパネルの作成例：

```
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|                                                ControlsButton.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CButton"
#include <Controls\Dialog.mqh>
#include <Controls\Button.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with
allowance for border width)
#define INDENT_TOP                          (11)      // indent from top (with
allowance for border width)
#define INDENT_RIGHT                        (11)      // indent from right (with
allowance for border width)
#define INDENT_BOTTOM                       (11)      // indent from bottom (with
allowance for border width)
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CButton           m_button1;                     // the button object
   CButton           m_button2;                     // the button object
   CButton           m_button3;                     // the fixed button object

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
```

© 2000–2025, MetaQuotes Ltd.

```
   virtual bool      Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2);
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double
&dparam,const string &sparam);

protected:
   //--- create dependent controls
   bool              CreateButton1(void);
   bool              CreateButton2(void);
   bool              CreateButton3(void);
   //--- handlers of the dependent controls events
   void              OnClickButton1(void);
   void              OnClickButton2(void);
   void              OnClickButton3(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CLICK,m_button1,OnClickButton1)
ON_EVENT(ON_CLICK,m_button2,OnClickButton2)
ON_EVENT(ON_CLICK,m_button3,OnClickButton3)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2)
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateButton1())
      return(false);
   if(!CreateButton2())
      return(false);
```

© 2000–2025, MetaQuotes Ltd.

```
   if(!CreateButton3())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "Button1" button                                      |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButton1(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- create
   if(!m_button1.Create(m_chart_id,m_name+"Button1",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_button1.Text("Button1"))
      return(false);
   if(!Add(m_button1))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "Button2" button                                      |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButton2(void)
  {
//--- coordinates
   int x1=INDENT_LEFT+(BUTTON_WIDTH+CONTROLS_GAP_X);
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- create
   if(!m_button2.Create(m_chart_id,m_name+"Button2",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_button2.Text("Button2"))
      return(false);
   if(!Add(m_button2))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "Button3" fixed button                                |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButton3(void)
```

© 2000–2025, MetaQuotes Ltd.

```
  {
//--- coordinates
   int x1=INDENT_LEFT+2*(BUTTON_WIDTH+CONTROLS_GAP_X);
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- create
   if(!m_button3.Create(m_chart_id,m_name+"Button3",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_button3.Text("Locked"))
      return(false);
   if(!Add(m_button3))
      return(false);
   m_button3.Locking(true);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickButton1(void)
  {
   Comment(__FUNCTION__);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickButton2(void)
  {
   Comment(__FUNCTION__);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickButton3(void)
  {
   if(m_button3.Pressed())
      Comment(__FUNCTION__+" State of the control: On");
   else
      Comment(__FUNCTION__+" State of the control: Off");
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
```

© 2000–2025, MetaQuotes Ltd.

```
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- clear comments
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000–2025, MetaQuotes Ltd.

Create

新しい CButton コントロールを作成します。

```
virtual bool  Create(
   const long     chart,       // チャート識別子
   const string   name,        // 名称
   const int      subwin,      // チャートサブウィンドウ
   const int      x1,          // x1 座標
   const int      y1,          // y1 座標
   const int      x2,          // x2 座標
   const int      y2           // y2 座標
  )
```

パラメータ

chart

[in] チャート識別子

name

[in] コントロールの一意の名称

subwin

[in] チャートサブウィンドウ

x1

[in] 左上の隅の X 座標

y1

[in] 左上の隅の Y 座標

x2

[in] 右下の隅の X 座標

y2

[in] 右下の隅の Y 座標

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

Pressed（Get メソッド）

コントロールの状態（「Pressed」プロパティ）を取得します。

```
bool  Pressed()  const
```

戻り値

コントロールの状態

Pressed（Set メソッド）

コントロールの状態（「Pressed」プロパティ）を設定します。

```
bool  Pressed(
   const bool  pressed      // 新しい状態
  )
```

パラメータ

pressed

[in] コントロールの新し状態

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

Locking（Get メソッド）

コントロールの「Locking」プロパティを取得します。

```
bool  Locking()  const
```

戻り値

「Locking」プロパティの値

Locking（Set メソッド）

コントロールの「Locking」プロパティの新しい値を設定します。

```
void  Locking(
   const bool  locking      // 新しい値
  )
```

パラメータ

locking

[in] 「Locking」プロパティの新しい値

戻り値

なし

© 2000-2025, MetaQuotes Ltd.

OnSetText

コントロールの「SetText(OBJPROP_TEXT プロパティの変更) イベントハンドラです。

```
virtual bool  OnSetText()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnSetColor

コントロールの「SetColor（OBJPROP_COLOR プロパティの変更）イベントハンドラです。

```
virtual bool  OnSetColor()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnSetColorBackground

コントロールの「SetColorBackground（OBJPROP_BGCOLOR プロパティの変更）イベントハンドラです。

```
virtual bool  OnSetColorBackground()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnSetColorBorder

コントロールの「SetColorBorder（OBJPROP_BORDER_COLORプロパティの変更）イベントハンドラです。

```
virtual bool  OnSetColorBackground()
```

戻り値

イベントが処理された場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

OnSetFont

コントロールの「SetFont（OBJPROP_FONTプロパティの変更）イベントハンドラです。

```
virtual bool  OnSetFont()
```

戻り値

イベントが処理された場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

OnSetFontSize

コントロールの「SetFontSize（OBJPROP_FONTSIZE プロパティの変更）イベントハンドラです。

```
virtual bool  OnSetFontSize()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnCreate

コントロールの「Create」イベントハンドラです。

```
virtual bool  OnCreate()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnShow

コントロールの「Show」イベントハンドラです。

```
virtual bool  OnShow()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnHide

コントロールの「Hide」イベントハンドラです。

```
virtual bool  OnHide()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnMove

コントロールの「Move」イベントハンドラです。

```
virtual bool  OnMove()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnResize

コントロールの「Resize」イベントハンドラです。

```
virtual bool  OnResize()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnMouseDown

コントロールの「MouseDown」イベントハンドラです。

```
virtual bool  OnMouseDown()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

「MouseDown」イベントは、コントロールの上で左マウスボタンが押された時に発生します。

© 2000-2025, MetaQuotes Ltd.

OnMouseUp

コントロールの「MouseUp」(マウスの左ボタンの開放)イベントハンドラです。

```
virtual bool  OnMouseUp()
```

戻り値

イベントが処理された場合はtrue、それ以外の場合はfalse

注意事項

「MouseUp」イベントは、コントロールの上でマウスの左ボタンが解放された時に発生します。

© 2000-2025, MetaQuotes Ltd.

CEdit

CEdit は「Edit」チャート オブジェクト に基 づいたシンプルコント ロールのクラスです。

説 明

CEdit クラスは、ユーザのテキスト 入 力 を可 能 にするコント ロールの作 成 に意 図 されています。

宣 言

```
class CEdit : public CWndObj
```

タイト ル

```
#include <Controls\Edit.mqh>
```

継 承 階 層

CObject
CWnd
CWndObj
CEdit

下 記 のコード の結 果 :



クラスメソッド

Create	
Create	コント ロールの作 成

© 2000–2025, MetaQuotes Ltd.

Create	
プロパティ	
ReadOnly	「ReadOnly」プロパティを取得/設定します。
TextAlign	「TextAlign」プロパティを取得/設定します。
チャートオブジェクトイベントハンドラ	
OnObjectEndEdit	CHARTEVENT_OBJECT_ENDEDIT イベントハンドラ(仮想)
プロパティ変更イベントハンドラ	
OnSetText	「SetText」イベントハンドラ
OnSetColor	「SetColor」イベントハンドラ
OnSetColorBackground	「SetColorBackground」イベントハンドラ
OnSetColorBorder	「SetColorBorder」イベントハンドラ
OnSetFont	「SetFont」イベントハンドラ
OnSetFontSize	「SetFontSize」イベントハンドラ
OnSetZOrder	「SetZOrder」イベントハンドラ
内部イベントハンドラ	
OnCreate	「Create」イベントハンドラ
OnShow	「Show」イベントハンドラ
OnHide	「Hide」イベントハンドラ
OnMove	「Move」イベントハンドラ
OnResize	「Resize」イベントハンドラ
OnChange	「Change」イベントハンドラ
OnClick	「Click」イベントハンドラ

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

クラスから継承されたメソッド CWnd

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

クラスから継承されたメソッド CWndObj

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

エディットを使ったパネルの作成例：

```
//+------------------------------------------------------------------+
//|                                                 ControlsEdit.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CEdit"
#include <Controls\Dialog.mqh>
#include <Controls\Edit.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with
allowance for border width)
#define INDENT_TOP                          (11)      // indent from top (with
allowance for border width)
#define INDENT_RIGHT                        (11)      // indent from right (with
allowance for border width)
#define INDENT_BOTTOM                       (11)      // indent from bottom (with
allowance for border width)
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
```

© 2000–2025, MetaQuotes Ltd.

```
   CEdit             m_edit;                          // CEdit object

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2);
   //--- chart event handler

protected:
   //--- create dependent controls
   bool              CreateEdit(void);
  };
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2)
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateEdit())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the display field                                         |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateEdit(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP;
   int x2=ClientAreaWidth()-INDENT_RIGHT;
```

© 2000–2025, MetaQuotes Ltd.

```
   int y2=y1+EDIT_HEIGHT;
//--- create
   if(!m_edit.Create(m_chart_id,m_name+"Edit",m_subwin,x1,y1,x2,y2))
      return(false);
//--- allow editing the content
   if(!m_edit.ReadOnly(false))
      return(false);
   if(!Add(m_edit))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- clear comments
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000–2025, MetaQuotes Ltd.

Create

新しい CEdit コントロールを作成します。

```
virtual bool  Create(
   const long     chart,       // チャート識別子
   const string   name,        // 名称
   const int      subwin,      // チャートサブウィンドウ
   const int      x1,          // x1 座標
   const int      y1,          // y1 座標
   const int      x2,          // x2 座標
   const int      y2           // y2 座標
  )
```

パラメータ

chart

[in] チャート識別子

name

[in] コントロールの一意の名称

subwin

[in] チャートサブウィンドウ

x1

[in] 左上の隅の X 座標

y1

[in] 左上の隅の Y 座標

x2

[in] 右下の隅の X 座標

y2

[in] 右下の隅の Y 座標

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

ReadOnly（Get メソッド）

コントロールの「ReadOnly」プロパティを取得します。

```
bool  ReadOnly()
```

戻り値

「ReadOnly」プロパティの値

ReadOnly（Set メソッド）

コントロールの「ReadOnly」プロパティを設定します。

```
bool  ReadOnly(
   const bool  flag      // 新しい値
  )
```

パラメータ

flag

[in] 「ReadOnly」プロパティの新しい値"

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

TextAlign（Get メソッド）

Gets the value of "TextAlign" property (text alignment mode) of the control.

```
ENUM_ALIGN_MODE  TextAlign() const
```

戻り値

コントロールの「TextAlign」プロパティの値

TextAlign（Set メソッド）

Sets new value of "TextAlign" property (text aligment mode) of the control.

```
bool  TextAlign(
   ENUM_ALIGN_MODE  align      // 新しい値
  )
```

パラメータ

align

[in]「TextAlign」プロパティの新しい値

戻り値

成功の場合は true、プロパティ値が変更できなかった場合は false

© 2000–2025, MetaQuotes Ltd.

OnObjectEndEdit

CHARTEVENT_OBJECT_ENDEDIT イベントハンドラです。

```
virtual bool  OnObjectEndEdit()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnSetText

コントロールの「SetText(OBJPROP_TEXT プロパティの変更) イベントハンドラです。

```
virtual bool  OnSetText()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnSetColor

コントロールの「SetColor（OBJPROP_COLOR プロパティの変更）イベントハンドラです。

```
virtual bool  OnSetColor()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnSetColorBackground

コントロールの「SetColorBackground（OBJPROP_BGCOLOR プロパティの変更）イベントハンドラです。

```
virtual bool  OnSetColorBackground()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnSetColorBorder

コントロールの「SetColorBorder」(OBJPROP_BORDER_COLOR プロパティの変更) イベントハンドラです。

```
virtual bool  OnSetColorBackground()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnSetFont

コントロールの「SetFont（OBJPROP_FONTプロパティの変更）イベントハンドラです。

```
virtual bool  OnSetFont()
```

戻り値

イベントが処理された場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

OnSetFontSize

コントロールの「SetFontSize（OBJPROP_FONTSIZE プロパティの変更）イベントハンドラです。

```
virtual bool  OnSetFontSize()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnSetZOrder

コントロールの「SetZOrder（OBJPROP_ZORDER プロパティの変更）イベントハンドラです。

```
virtual bool  OnSetZOrder()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnCreate

コントロールの「Create」イベントハンドラです。

```
virtual bool  OnCreate()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnShow

コントロールの「Show」イベントハンドラです。

```
virtual bool  OnShow()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnHide

コントロールの「Hide」イベントハンドラです。

```
virtual bool  OnHide()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnMove

コントロールの「Move」イベントハンドラです。

```
virtual bool  OnMove()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnResize

コントロールの「Resize」イベントハンドラです。

```
virtual bool  OnResize()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnChange

コントロールの「Change」イベントハンドラです。

```
virtual bool  OnChange()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnClick

コントロールの「Click」(マウスの左ボタンのクリック) イベントハンドラです。

```
virtual bool  OnClick()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

CPanel

CPanel は「Rectangle label」チャート オブジェクト に基 づいたシンプルコント ロールのクラスです。

説 明

CPanel クラスは、グループ内 のいろいろな の関 数 の制 御 を組 み合 わせることを意 図 しています。

宣 言

```
class CPanel : public CWndObj
```

タイト ル

```
#include <Controls\Panel.mqh>
```

継 承 階 層

CObject
CWnd
CWndObj
CPanel

下 記 のコード の結 果 :



クラスメソッド

Create	
Create	コント ロールの作 成

© 2000–2025, MetaQuotes Ltd.

Create	
チャート オブジェクト プロパティ	
BorderType	チャート オブジェクト の「BorderType」プロパティを取 得します。
チャート オブジェクト イベント ハンドラ	
OnSetText	「SetText」イベント ハンドラ
OnSetColorBackground	「SetColorBackground」イベント ハンドラ
OnSetColorBorder	「SetColorBorder」イベント ハンドラ
内 部 イベント ハンドラ	
OnCreate	「Create」イベント ハンドラ
OnShow	「Show」イベント ハンドラ
OnHide	「Hide」イベント ハンドラ
OnMove	「Move」イベント ハンドラ
OnResize	「Resize」イベント ハンドラ
OnChange	「Change」イベント ハンドラ

クラスから継 承 されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

クラスから継 承 されたメソッド CWnd

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

クラスから継 承 されたメソッド CWndObj

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

四 角 形 ラベルを使 ったパネルの作 成 例 :

```
//+------------------------------------------------------------------+
//|                                                ControlsPanel.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

© 2000–2025, MetaQuotes Ltd.

```
#property description "Control Panels and Dialogs. Demonstration class CPanel"
#include <Controls\Dialog.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with
allowance for border width)
#define INDENT_TOP                          (11)      // indent from top (with
allowance for border width)
#define INDENT_RIGHT                        (11)      // indent from right (with
allowance for border width)
#define INDENT_BOTTOM                       (11)      // indent from bottom (with
allowance for border width)
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2);

protected:
   //--- create dependent controls
   bool              CreatePanel(void);
  };
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
```

© 2000–2025, MetaQuotes Ltd.

```
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2)
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreatePanel())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "CPanel"                                              |
//+------------------------------------------------------------------+
bool CControlsDialog::CreatePanel(void)
  {
//--- coordinates
   int x1=20;
   int y1=20;
   int x2=ExtDialog.Width()/3;
   int y2=ExtDialog.Height()/3;
//--- create
   if(!my_white_border.Create(0,ExtDialog.Name()
+"MyWhiteBorder",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!my_white_border.ColorBackground(CONTROLS_DIALOG_COLOR_BG))
      return(false);
   if(!my_white_border.ColorBorder(CONTROLS_DIALOG_COLOR_BORDER_LIGHT))
      return(false);
   if(!ExtDialog.Add(my_white_border))
      return(false);
   my_white_border.Alignment(WND_ALIGN_CLIENT,0,0,0,0);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//---
CPanel   my_white_border;        // object CPanel
```

© 2000–2025, MetaQuotes Ltd.

```
bool     pause=true;             // true - pause
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   EventSetTimer(3);
   pause=true;
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- clear comments
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,          // event ID
                  const long& lparam,    // event parameter of the long type
                  const double& dparam,  // event parameter of the double type
                  const string& sparam)  // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
//+------------------------------------------------------------------+
//|  Timer                                                           |
//+------------------------------------------------------------------+
void OnTimer()
  {
   pause=!pause;
  }
```

© 2000–2025, MetaQuotes Ltd.

Create

新しい CPanel コントロールを作成します。

```
virtual bool  Create(
   const long     chart,       // チャート識別子
   const string   name,        // 名称
   const int      subwin,      // チャートサブウィンドウ
   const int      x1,          // x1 座標
   const int      y1,          // y1 座標
   const int      x2,          // x2 座標
   const int      y2           // y2 座標
  )
```

パラメータ

chart

[in] チャート識別子

name

[in] コントロールの一意の名称

subwin

[in] チャートサブウィンドウ

x1

[in] 左上の隅の X 座標

y1

[in] 左上の隅の Y 座標

x2

[in] 右下の隅の X 座標

y2

[in] 右下の隅の Y 座標

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

BorderType（Get メソッド）

チャート オブジェクト の「BorderType」プロパティを取 得します。

```
ENUM_BORDER_TYPE  BorderType()
```

戻り値

「BorderType」プロパティの値

BorderType（Set メソッド）

チャート オブジェクト の「BorderType」プロパティを設 定します。

```
bool  BorderType(
   const ENUM_BORDER_TYPE  type      // 値
  )
```

パラメータ

型

[in] 「BorderType」プロパティの新しい値

戻り値

成 功 の場 合 は true、それ以 外 の場 合 は false

© 2000–2025, MetaQuotes Ltd.

OnSetText

コントロールの「SetText(OBJPROP_TEXT プロパティの変更) イベントハンドラです。

```
virtual bool  OnSetText()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnSetColorBackground

コントロールの「SetColorBackground（OBJPROP_BGCOLOR プロパティの変更）イベントハンドラです。

```
virtual bool  OnSetColorBackground()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnSetColorBorder

コントロールの「SetColorBorder」(OBJPROP_BORDER_COLOR プロパティの変更)イベントハンドラです。

```
virtual bool  OnSetColorBackground()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnCreate

コントロールの「Create」イベントハンドラです。

```
virtual bool  OnCreate()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnShow

コントロールの「Show」イベントハンドラです。

```
virtual bool  OnShow()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnHide

コントロールの「Hide」イベントハンドラです。

```
virtual bool  OnHide()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnMove

コントロールの「Move」イベントハンドラです。

```
virtual bool  OnMove()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnResize

コントロールの「Resize」イベントハンドラです。

```
virtual bool  OnResize()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnChange

コントロールの「Change」イベントハンドラです。

```
virtual bool  OnChange()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CPicture

CPicture は「Bitmap Label（ビットマップラベル）」チャートオブジェクトに基づいたシンプルコントロールのクラスです。

説明

CPicture クラスは、シンプルなグラフィックイメージの作成に意図されています。

宣言

```
class CPicture : public CWndObj
```

タイトル

```
#include <Controls\Picture.mqh>
```

継承階層

CObject
CWnd
CWndObj
CPicture

下記のコードの結果：



クラスメソッド

Create	
Create	コントロールの作成

© 2000–2025, MetaQuotes Ltd.

Create	
チャート オブジェクト プロパティ	
Border	チャート オブジェクト の幅を取 得/設 定します。
BmpName	コント ロールの bmp ファイルの名 称を取 得/設 定します。
内 部 イベント	
OnCreate	「Create」イベント ハンドラ
OnShow	「Show」イベント ハンドラ
OnHide	「Hide」イベント ハンドラ
OnMove	「Move」イベント ハンドラ
OnChange	「Change」イベント ハンドラ

クラスから継 承 されたメソッド CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

クラスから継 承 されたメソッド CWnd

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

クラスから継 承 されたメソッド CWndObj

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

ビット マップラベルを使 ったパネルの作 成 例 :

```
//+------------------------------------------------------------------+
//|                                                ControlsPicture.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CPicture"
#include <Controls\Dialog.mqh>
#include <Controls\Picture.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
```

© 2000–2025, MetaQuotes Ltd.

```
#define INDENT_LEFT                         (11)      // indent from left (with
allowance for border width)
#define INDENT_TOP                          (11)      // indent from top (with
allowance for border width)
#define INDENT_RIGHT                        (11)      // indent from right (with
allowance for border width)
#define INDENT_BOTTOM                       (11)      // indent from bottom (with
allowance for border width)
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CPicture          m_picture;                      // CPicture object

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2);
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double
&dparam,const string &sparam);

protected:
   //--- create dependent controls
   bool              CreatePicture(void);
   //--- handlers of the dependent controls events
   void              OnClickPicture(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CLICK,m_picture,OnClickPicture)
```

© 2000–2025, MetaQuotes Ltd.

```
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2)
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreatePicture())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "Picture"                                             |
//+------------------------------------------------------------------+
bool CControlsDialog::CreatePicture(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+32;
   int y2=y1+32;
//--- create
   if(!m_picture.Create(m_chart_id,m_name+"Picture",m_subwin,x1,y1,x2,y2))
      return(false);
//--- set the name of bmp files to display the CPicture control
   m_picture.BmpName("\\Images\\euro.bmp");

   if(!Add(m_picture))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000–2025, MetaQuotes Ltd.

```
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickPicture(void)
  {
   Comment(__FUNCTION__);
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- clear comments
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000–2025, MetaQuotes Ltd.

Create

新しい CPicture コントロールを作成します。

```
virtual bool  Create(
   const long     chart,       // チャート識別子
   const string   name,        // 名称
   const int      subwin,      // チャートサブウィンドウ
   const int      x1,          // x1 座標
   const int      y1,          // y1 座標
   const int      x2,          // x2 座標
   const int      y2           // y2 座標
  )
```

パラメータ

chart

[in] チャート識別子

name

[in] コントロールの一意の名称

subwin

[in] チャートサブウィンドウ

x1

[in] 左上の隅の X 座標

y1

[in] 左上の隅の Y 座標

x2

[in] 右下の隅の X 座標

y2

[in] 右下の隅の Y 座標

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Border（Get メソッド）

コントロールの「Border（境界の幅）」プロパティを取得します。

```
int  Border()  const
```

戻り値

「Border」プロパティ

Border（Set メソッド）

コントロールの「Border（境界の幅）」プロパティを設定します。

```
bool  Border(
   const int  value      // 新しい値
  )
```

パラメータ

value

[in]「Border」プロパティの新しい値

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

BmpName（Get メソッド）

コントロールの bmp ファイルの名称を取得します。

```
string  BmpName()  const
```

戻り値

コントロールの bmp ファイルの名称

BmpName（Set メソッド）

コントロールの bmp ファイルの名称を設定します。

```
bool  BmpName(
   const string  name      // ファイル名
  )
```

パラメータ

name

[in] コントロールの bmp ファイルの名称

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnCreate

コントロールの「Create」イベントハンドラです。

```
virtual bool  OnCreate()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnShow

コントロールの「Show」イベントハンドラです。

```
virtual bool  OnShow()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnHide

コントロールの「Hide」イベントハンドラです。

```
virtual bool  OnHide()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnMove

コントロールの「Move」イベントハンドラです。

```
virtual bool  OnMove()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnChange

コントロールの「Change」イベントハンドラです。

```
virtual bool  OnChange()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CScroll

CScroll はスクロールバーの基本クラスです。

説明

CScroll は（依存コントロールを含む）複雑なコントロールで、スクロールバー作成の基本的な機能を含みます。基本クラスはそれ自体のコントロールとしては使用されず、2 つの派生クラス（CScrollV 及び CScrollH クラス）がコントロールとして使用されます。

宣言

```
class CScroll : public CWndContainer
```

タイトル

```
#include <Controls\Scrolls.mqh>
```

継承階層

CObject
  CWnd
    CWndContainer
      CScroll

直接子孫

CScrollH, CScrollV

クラスメソッド

Create	
Create	コントロールの作成
チャートオブジェクトイベントハンドラ	
OnEvent	全てのチャートイベントのハンドラ
プロパティ	
MinPos	最小位置を取得/設定します。
MaxPos	最大位置を取得/設定します。
CurrPos	現在位置を取得/設定します。
依存コントロールの作成	
CreateBack	背景ボタンを作成します。
CreateInc	スクロールバーのインクリメントボタンを作成します。
CreateDec	スクロールバーのデクリメントボタンを作成します。

© 2000–2025, MetaQuotes Ltd.

Create	
CreateThumb	スクロールバーの(ドラッグ可能な)サムボタンを作成します。
依存コントロールのイベントハンドラ	
OnClickInc	インクリメントボタンイベント処理に使用されるイベントハンドラ
OnClickDec	デクリメントボタンイベント処理に使用されるイベントハンドラ
内部イベントハンドラ	
OnShow	「Create」イベントハンドラ
OnHide	「Hide」イベントハンドラ
OnChangePos	「ChangePosition」イベントハンドラ
オブジェクトドラッグハンドラ	
OnThumbDragStart	「ThumbDragStart」イベントハンドラ
OnThumbDragProcess	「ThumbDragProcess」イベントハンドラ
OnThumbDragEnd	「ThumbDragEnd」イベントハンドラ
位置	
CalcPos	スクロールバーの位置を座標で取得します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Type, Compare

クラスから継承されたメソッド CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

クラスから継承されたメソッド CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

© 2000–2025, MetaQuotes Ltd.

Create

新しい CScroll コントロールを作成します。

```
virtual bool  Create(
   const long     chart,       // チャート識別子
   const string   name,        // 名称
   const int      subwin,      // チャートサブウィンドウ
   const int      x1,          // x1 座標
   const int      y1,          // y1 座標
   const int      x2,          // x2 座標
   const int      y2           // y2 座標
  )
```

パラメータ

chart

[in] チャート識別子

name

[in] コントロールの一意の名称

subwin

[in] チャートサブウィンドウ

x1

[in] 左上の隅の X 座標

y1

[in] 左上の隅の Y 座標

x2

[in] 右下の隅の X 座標

y2

[in] 右下の隅の Y 座標

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnEvent

チャート イベント ハンドラ

```
virtual bool  OnEvent(
   const int      id,          // 識別子
   const long&    lparam,      // long 型のイベントパラメータ
   const double&  dparam,      // double 型のイベントパラメータ
   const string&  sparam       // string 型のイベントパラメータ
  )
```

パラメータ

id

[in] イベント識別子

lparam

[in] 参照で渡された long 型のイベント パラメータ

dparam

[in] 参照で渡された double 型のイベント パラメータ

sparam

[in] 参照で渡された string 型のイベント パラメータ

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

MinPos（Get メソッド）

CScroll コントロールの「MinPos」（最小位置）の値を取得します。

```
int  MinPos()  const
```

戻り値

「MinPos」プロパティの新しい値

MinPos（Set メソッド）

CScroll コントロールの「MinPos」（最小位置）の値を設定します。

```
void  MinPos(
   const int  value      // 新しい値
  )
```

パラメータ

value

[in]「MinPos」プロパティの新しい値

戻り値

なし

© 2000-2025, MetaQuotes Ltd.

MaxPos（Get メソッド）

CScroll コントロールの「MaxPos」（最大位置）の値を取得します。

```
int  MaxPos()  const
```

戻り値

「MaxPos」の新しい値

MaxPos（Set メソッド）

CScroll コントロールの「MaxPos」（最大位置）の値を設定します。

```
void  MaxPos(
   const int  value      // 新しい値
  )
```

パラメータ

value

[in]「MaxPos」プロパティの新しい値

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

CurrPos（Get メソッド）

CScroll コントロールの「CurrPos」（現在位置）の値を取得します。

```
int  CurrPos()  const
```

戻り値

「CurrPos」プロパティの新しい値

CurrPos（Set メソッド）

CScroll コントロールの「CurrPos」（現在位置）の値を設定します。

```
void  CurrPos(
   const int  value      // 新しい値
  )
```

パラメータ

value

[in]「CurrPos」プロパティの新しい値

戻り値

なし

© 2000-2025, MetaQuotes Ltd.

CreateBack

CScroll コントロールの背景ボタンを作成します。

```
virtual bool  CreateBack()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CreateInc

CScroll コントロールのインクリメントボタンを作成します。

```
virtual bool  CreateInc()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

CreateDec

CScroll コントロールのデクリメントボタンを作成します。

```
virtual bool  CreateDec()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

CreateThumb

CScroll コントロールのサムボタンを作成します。

```
virtual bool  CreateThumb()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnClickInc

コントロールの「ClickInc（インクレメントボタンのマウス左クリック）」イベントハンドラです。

```
virtual bool  OnClickInc()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnClickDec

コントロールの「ClickDec（デクレメントボタンのマウス左クリック）」イベントハンドラです。

```
virtual bool  OnClickDec()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnShow

コントロールの「Show」イベントハンドラです。

```
virtual bool  OnShow()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnHide

コントロールの「Hide」イベントハンドラです。

```
virtual bool  OnHide()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnChangePos

コントロールの「ChangePos」(位置の変更)イベントハンドラです。

```
virtual bool  OnChangePos()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

基本クラスメソッドは何の操作も行わずに常に true を返します。

© 2000–2025, MetaQuotes Ltd.

OnThumbDragStart

コントロールの「ThumbDragStart（ドラッグの開始）」イベントハンドラです。

```
virtual bool  OnThumbDragStart()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

「ThumbDragStart」イベントは、ドラッグの開始時に発生します。

© 2000–2025, MetaQuotes Ltd.

OnThumbDragProcess

コントロールの「ThumbDragProcess」イベントハンドラです。

```
virtual bool  OnThumbDragProcess(
   const int  x,      // x 座標
   const int  y       // y 座標
  )
```

パラメータ

x

[in] マウスカーソルの現在の X 座標

y

[in] マウスカーソルの現在の Y 座標

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

「ThumbDragProcess」は、スクロールバーコントロール（サムボタン）の移動時に発生します。

© 2000–2025, MetaQuotes Ltd.

OnThumbDragEnd

コントロールの「ThumbDragEnd（ドラッグの終了）」イベントハンドラです。

```
virtual bool  OnThumbDragEnd()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

「ThumbDragEnd」イベントは、スクロールバーコントロールのドラッグの終了時に発生します。

© 2000–2025, MetaQuotes Ltd.

CalcPos

スクロールバーの位置を座標で取得します。

```
virtual int  CalcPos(
   const int       coord      // 座標
  )
```

パラメータ

coord

[in] スクロールバーの座標

戻り値

スクロールバーの位置

© 2000–2025, MetaQuotes Ltd.

CScrollV

CScrollV は「Vertical scroll bar」の複雑なコントロールのクラスです。

説明

CScrollV クラスは垂直方向のスクロールバーの作成に意図されています。

宣言

```
class CScrollV : public CScroll
```

タイトル

```
#include <Controls\Scrolls.mqh>
```

継承階層

CObject
CWnd
CWndContainer
CScroll
CScrollV

下記のコードの結果：



クラスメソッド

© 2000–2025, MetaQuotes Ltd.

依存コントロール	
CreateInc	スクロールバーのインクレメントボタンを作成します。
CreateDec	スクロールバーのデクレメントボタンを作成します。
CreateThumb	スクロールバーの(ドラッグ可能な)サムボタンを作成します。
内部イベントハンドラ	
OnResize	「Resize」イベントハンドラ
OnChangePos	「ChangePosition」イベントハンドラ
ドラッグイベントハンドラ	
OnThumbDragStart	「ThumbDragStart」イベントハンドラ
OnThumbDragProcess	「ThumbDragProcess」イベントハンドラ
OnThumbDragEnd	「ThumbDragEnd」イベントハンドラ
位置	
CalcPos	スクロールバーの位置を座標で取得します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Type, Compare

クラスから継承されたメソッド CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

クラスから継承されたメソッド CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

クラスから継承されたメソッド CScroll

Create, OnEvent, MinPos, MinPos, MaxPos, MaxPos, CurrPos, CurrPos

縦方向のスクロールバーを使ったパネルの作成例：

```
//+------------------------------------------------------------------+
//|                                                ControlsScrollV.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CScrollV"
```

© 2000-2025, MetaQuotes Ltd.

```
#include <Controls\Dialog.mqh>
#include <Controls\Scrolls.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with
allowance for border width)
#define INDENT_TOP                          (11)      // indent from top (with
allowance for border width)
#define INDENT_RIGHT                        (11)      // indent from right (with
allowance for border width)
#define INDENT_BOTTOM                       (11)      // indent from bottom (with
allowance for border width)
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CScrollV          m_scroll_v;                     // CScrollV object

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2);
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double
&dparam,const string &sparam);

protected:
   //--- create dependent controls
   bool              CreateScrollV(void);
   //--- handlers of the dependent controls events
   void              OnScrollInc(void);
```

© 2000–2025, MetaQuotes Ltd.

```
   void              OnScrollDec(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_SCROLL_INC,m_scroll_v,OnScrollInc)
ON_EVENT(ON_SCROLL_DEC,m_scroll_v,OnScrollDec)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2)
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateScrollV())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the CScrollsV object                                      |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateScrollV(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP;
   int x2=x1+18;
   int y2=y1+LIST_HEIGHT;
//--- create
   if(!m_scroll_v.Create(m_chart_id,m_name+"ScrollV",m_subwin,x1,y1,x2,y2))
      return(false);
//--- set up the scrollbar
```

© 2000–2025, MetaQuotes Ltd.

```
   m_scroll_v.MinPos(0);
//--- set up the scrollbar
   m_scroll_v.MaxPos(10);
   if(!Add(m_scroll_v))
      return(false);
   Comment("Position of the scrollbar ",m_scroll_v.CurrPos());
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollInc(void)
  {
   Comment("Position of the scrollbar ",m_scroll_v.CurrPos());
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollDec(void)
  {
   Comment("Position of the scrollbar ",m_scroll_v.CurrPos());
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- clear comments
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
```

© 2000–2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,          // event ID
                  const long& lparam,    // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000–2025, MetaQuotes Ltd.

CreateInc

コントロールのインクリメントボタンを作成します。

```
virtual bool  CreateInc()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CreateDec

コントロールのデクリメントボタンを作成します。

```
virtual bool  CreateDec()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CreateThumb

コントロールの(ドラッグ可能な)サムボタンを作成します。

```
virtual bool  CreateThumb()
```

戻り値

成功の場合はtrue、それ以外の場合はfalse

© 2000-2025, MetaQuotes Ltd.

OnResize

コントロールの「Resize」イベントハンドラです。

```
virtual bool  OnResize()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnChangePos

コントロールの「ChangePos」(位置の変更)イベントハンドラです。

```
virtual bool  OnChangePos()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnThumbDragStart

コントロールの「ThumbDragStart（ドラッグの開始）」イベントハンドラです。

```
virtual bool  OnThumbDragStart()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

「ThumbDragStart」イベントは、ドラッグの開始時に発生します。

© 2000-2025, MetaQuotes Ltd.

OnThumbDragProcess

コントロールの「ThumbDragProcess」イベントハンドラです。

```
virtual bool  OnThumbDragProcess(
   const int  x,      // x 座標
   const int  y       // y 座標
  )
```

パラメータ

x

[in] マウスカーソルの現在の X 座標

y

[in] マウスカーソルの現在の Y 座標

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

「ThumbDragProcess」は、スクロールバーコントロール（サムボタン）の移動時に発生します。

© 2000–2025, MetaQuotes Ltd.

OnThumbDragEnd

コントロールの「ThumbDragEnd（ドラッグの終了）」イベントハンドラです。

```
virtual bool  OnThumbDragEnd()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

「ThumbDragEnd」イベントは、スクロールバーコントロールのドラッグの終了時に発生します。

© 2000–2025, MetaQuotes Ltd.

CalcPos

スクロールバーの位置を座標で取得します。

```
virtual int  CalcPos(
   const int       coord      // 座標
  )
```

パラメータ

coord

[in] スクロールバーの座標

戻り値

スクロールバーの位置

© 2000–2025, MetaQuotes Ltd.

CScrollH

CScrollH は「Horizontal scroll bar」の複雑なコントロールのクラスです。

説明

CScrollH クラスは水平方向のスクロールバーの作成に意図されています。

宣言

```
class CScrollH : public CScroll
```

タイトル

```
#include <Controls\Scrolls.mqh>
```

継承階層

COBject
CWnd
CWndContainer
CScroll
CScrollH

下記のコードの結果：



クラスメソッド

© 2000–2025, MetaQuotes Ltd.

依存コントロール	
CreateInc	スクロールバーのインクレメントボタンを作成します。
CreateDec	スクロールバーのデクレメントボタンを作成します。
CreateThumb	スクロールバーの(ドラッグ可能な)サムボタンを作成します。
内部イベントハンドラ	
OnResize	「Resize」イベントハンドラ
OnChangePos	「ChangePosition」イベントハンドラ
ドラッグイベントハンドラ	
OnThumbDragStart	「ThumbDragStart」イベントハンドラ
OnThumbDragProcess	「ThumbDragProcess」イベントハンドラ
OnThumbDragEnd	「ThumbDragEnd」イベントハンドラ
位置	
CalcPos	スクロールバーの位置を座標で取得します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Type, Compare

クラスから継承されたメソッド CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

クラスから継承されたメソッド CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

クラスから継承されたメソッド CScroll

Create, OnEvent, MinPos, MinPos, MaxPos, MaxPos, CurrPos, CurrPos

鉾方向のスクロールバーを使ったパネルの作成例：

```
//+------------------------------------------------------------------+
//|                                                ControlsScrollH.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CScrollH"
```

© 2000-2025, MetaQuotes Ltd.

```
#include <Controls\Dialog.mqh>
#include <Controls\Scrolls.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with
allowance for border width)
#define INDENT_TOP                          (11)      // indent from top (with
allowance for border width)
#define INDENT_RIGHT                        (11)      // indent from right (with
allowance for border width)
#define INDENT_BOTTOM                       (11)      // indent from bottom (with
allowance for border width)
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CScrollH          m_scroll_v;                     // CScrollH object

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2);
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double
&dparam,const string &sparam);

protected:
   //--- create dependent controls
   bool              CreateScrollsH(void);
   //--- handlers of the dependent controls events
   void              OnScrollInc(void);
```

© 2000–2025, MetaQuotes Ltd.

```
   void              OnScrollDec(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_SCROLL_INC,m_scroll_v,OnScrollInc)
ON_EVENT(ON_SCROLL_DEC,m_scroll_v,OnScrollDec)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2)
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateScrollsH())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the CScrollsH object                                      |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateScrollsH(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP;
   int x2=x1+3*BUTTON_WIDTH;
   int y2=y1+18;
//--- create
   if(!m_scroll_v.Create(m_chart_id,m_name+"ScrollsH",m_subwin,x1,y1,x2,y2))
      return(false);
//--- set up the scrollbar
```

© 2000–2025, MetaQuotes Ltd.

```
   m_scroll_v.MinPos(0);
//--- set up the scrollbar
   m_scroll_v.MaxPos(10);
   if(!Add(m_scroll_v))
      return(false);
   Comment("Position of the scrollbar ",m_scroll_v.CurrPos());
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollInc(void)
  {
   Comment("Position of the scrollbar ",m_scroll_v.CurrPos());
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollDec(void)
  {
   Comment("Position of the scrollbar ",m_scroll_v.CurrPos());
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- clear comments
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
```

© 2000–2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,          // event ID
                  const long& lparam,    // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000–2025, MetaQuotes Ltd.

CreateInc

コントロールのインクリメントボタンを作成します。

```
virtual bool  CreateInc()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

CreateDec

コントロールのデクリメントボタンを作成します。

```
virtual bool  CreateDec()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CreateThumb

コントロールの(ドラッグ可能な)サムボタンを作成します。

```
virtual bool  CreateThumb()
```

戻り値

成功の場合はtrue、それ以外の場合はfalse

© 2000-2025, MetaQuotes Ltd.

OnResize

コントロールの「Resize」イベントハンドラです。

```
virtual bool  OnResize()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnChangePos

コントロールの「ChangePos」(位置の変更)イベントハンドラです。

```
virtual bool  OnChangePos()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnThumbDragStart

コントロールの「ThumbDragStart（ドラッグの開始）」イベントハンドラです。

```
virtual bool  OnThumbDragStart()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

「ThumbDragStart」イベントは、ドラッグの開始時に発生します。

© 2000–2025, MetaQuotes Ltd.

OnThumbDragProcess

コントロールの「ThumbDragProcess」イベントハンドラです。

```
virtual bool  OnThumbDragProcess(
   const int  x,      // x 座標
   const int  y       // y 座標
  )
```

パラメータ

x

[in] マウスカーソルの現在の X 座標

y

[in] マウスカーソルの現在の Y 座標

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

「ThumbDragProcess」は、スクロールバーコントロール（サムボタン）の移動時に発生します。

© 2000–2025, MetaQuotes Ltd.

OnThumbDragEnd

コントロールの「ThumbDragEnd（ドラッグの終了）」イベントハンドラです。

```
virtual bool  OnThumbDragEnd()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

「ThumbDragEnd」イベントは、スクロールバーコントロールのドラッグの終了時に発生します。

© 2000–2025, MetaQuotes Ltd.

CalcPos

スクロールバーの位置を座標で取得します。

```
virtual int  CalcPos(
   const int      coord      // 座標
  )
```

パラメータ

coord

[in] スクロールバーの座標

戻り値

スクロールバーの位置

© 2000–2025, MetaQuotes Ltd.

CWndClient

CWndClient は「Client area」の(依存コントロールを含む)複雑なコントロールのクラスです。これは、スクロールバー領域作成の基本クラスです。

説明

CWndClient はスクロールバー付きのクライアント領域の作成を実装します。

宣言

```
class CWndClient : public CWndContainer
```

タイトル

```
#include <Controls\WndClient.mqh>
```

継承階層

CObject
  CWnd
    CWndContainer
      CWndClient

直接子孫

CCheckGroup, CListView, CRadioGroup

クラスメソッド

Create	
Create	コントロールの作成
チャートイベントハンドラ	
OnEvent	全てのチャートイベントのハンドラ
プロパティ	
ColorBackground	背景色を設定します。
ColorBorder	境界色を設定します。
BorderType	境界線の種類を設定します。
設定	
VScrolled	垂直方向のスクロールバーの使用を示すフラグを取得/設定します。
HScrolled	水平方向のスクロールバーの使用を示すフラグを取得/設定します。
依存コントロール	

© 2000-2025, MetaQuotes Ltd.

Create	
CreateBack	スクロールバーの背景を作成します。
CreateScrollV	垂直方向のスクロールバーを作成します。
CreateScrollH	水平方向のスクロールバーを作成します。
内部イベントハンドラ	
OnResize	「Resize」イベントハンドラ
依存コントロールのイベントハンドラ	
OnVScrollShow	VScroll 依存コントロールの「Show」イベントハンドラ(仮想)
OnVScrollHide	VScroll 依存コントロールの「Hide」イベントハンドラ(仮想)
OnHScrollShow	HScroll 依存コントロールの「Show」イベントハンドラ(仮想)
OnHScrollHide	HScroll 依存コントロールの「Hide」イベントハンドラ(仮想)
OnScrollLineDown	VScroll 依存コントロールの「ScrollLineDown」イベントハンドラ(仮想)
OnScrollLineUp	VScroll 依存コントロールの「ScrollLineUp」イベントハンドラ(仮想)
OnScrollLineLeft	HScroll 依存コントロールの「ScrollLineLeft」イベントハンドラ(仮想)
OnScrollLineRight	HScroll 依存コントロールの「ScrollLineRight」イベントハンドラ(仮想)
Resize	
Rebound	CRect クラスの座標を使用してコントロールの新しい座標を設定します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Type, Compare

クラスから継承されたメソッド CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

クラスから継承されたメソッド CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide, Save, Load

© 2000–2025, MetaQuotes Ltd.

Create

新しい CWndClient コントロールを作成します。

```
virtual bool  Create(
   const long     chart,       // チャート識別子
   const string   name,        // 名称
   const int      subwin,      // チャートサブウィンドウ
   const int      x1,          // x1 座標
   const int      y1,          // y1 座標
   const int      x2,          // x2 座標
   const int      y2           // y2 座標
  )
```

パラメータ

chart

[in] チャート識別子

name

[in] コントロールの一意の名称

subwin

[in] チャートサブウィンドウ

x1

[in] 左上の隅の X 座標

y1

[in] 左上の隅の Y 座標

x2

[in] 右下の隅の X 座標

y2

[in] 右下の隅の Y 座標

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnEvent

チャート イベント ハンドラ

```
virtual bool  OnEvent(
   const int      id,          // 識別子
   const long&    lparam,      // long 型のイベントパラメータ
   const double&  dparam,      // double 型のイベントパラメータ
   const string&  sparam       // string 型のイベントパラメータ
  )
```

パラメータ

id

[in] イベント識別子

lparam

[in] 参照で渡された long 型のイベント パラメータ

dparam

[in] 参照で渡された double 型のイベント パラメータ

sparam

[in] 参照で渡された string 型のイベント パラメータ

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

ColorBackground

コントロールの背景色を設定します。

```
bool  ColorBackground(
   const color  value      // 新しい色
  )
```

パラメータ

value

[in] コントロールの新しい背景色

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

ColorBorder

コントロールの境界色を設定します。

```
bool  ColorBorder(
   const color  value      // 色
  )
```

パラメータ

value

[in] コントロールの新しい境界色

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

BorderType

コントロールの境界線の種類を設定します。

```
bool  BorderType(
   const ENUM_BORDER_TYPE  type      // 境界線の種類
  )
```

パラメータ

型

[in] コントロールの境界線の種類

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

VScrolled（Get メソッド）

垂直方向のスクロールバーの使用を示すフラグを取得します。

```
bool  VScrolled()
```

戻り値

垂直方向のスクロールバーが使用されている場合は true、それ以外の場合は false

VScrolled（Set メソッド）

垂直方向のスクロールバーの使用を示すフラグを設定します。

```
bool  VScrolled(
   const bool  flag      // フラグ
  )
```

パラメータ

flag

[in] フラグ

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

HScrolled（Get メソッド）

水平方向のスクロールバーの使用を示すフラグを取得します。

```
bool  HScrolled()
```

戻り値

水平方向のスクロールバーが使用されている場合は true、それ以外の場合は false

HScrolled（Set メソッド））

水平方向のスクロールバーの使用を示すフラグを設定します。

```
bool  HScrolled(
   const bool  flag      // フラグ
  )
```

パラメータ

flag

[in] フラグ

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CreateBack

コントロールの背景ボタンを作成します。

```
virtual bool  CreateBack()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CreateScrollV

垂直方向のスクロールバーを作成します。

```
virtual bool  CreateScrollV()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CreateScrollH

水平方向のスクロールバーを作成します。

```
virtual bool  CreateScrollH()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnResize

コントロールの「Resize」イベントハンドラです。

```
virtual bool  OnResize()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnVScrollShow

コントロールの「VScrollHide」(垂直スクロールバーの表示)イベントハンドラです。

```
virtual bool  OnVScrollShow()
```

戻り値

イベントが処理された場合はtrue、それ以外の場合はfalse

注意事項

基本クラスメソッドは何の操作も行わずに常にtrueを返します。

© 2000-2025, MetaQuotes Ltd.

OnVScrollHide

コントロールの「VScrollHide」(垂直スクロールバーの非表示)イベントハンドラです。

```
virtual bool  OnVScrollHide()
```

戻り値

イベントが処理された場合はtrue、それ以外の場合はfalse

注意事項

基本クラスメソッドは何の操作も行わずに常にtrueを返します。

© 2000-2025, MetaQuotes Ltd.

OnHScrollShow

コントロールの「HScrollShow」(水平スクロールバーの表示)イベントハンドラです。

```
virtual bool  OnHScrollShow()
```

戻り値

イベントが処理された場合はtrue、それ以外の場合はfalse

注意事項

基本クラスメソッドは何の操作も行わずに常にtrueを返します。

© 2000-2025, MetaQuotes Ltd.

OnHScrollHide

コントロールの「HScrollHide」（水平スクロールバーの非表示）イベントハンドラです。

```
virtual bool  OnHScrollHide()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

基本クラスメソッドは何の操作も行わずに常に true を返します。

© 2000-2025, MetaQuotes Ltd.

OnScrollLineDown

コントロールの「ScrollLineDown」(垂直スクロールの下向きスクロール) イベントハンドラです。

```
virtual bool  OnScrollLineDown()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

基本クラスメソッドは何の操作も行わずに常に true を返します。

© 2000-2025, MetaQuotes Ltd.

OnScrollLineUp

コントロールの「ScrollLineUp」(垂直スクロールの上向きスクロール) イベントハンドラです。

```
virtual bool  OnScrollLineUp()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

基本クラスメソッドは何の操作も行わずに常に true を返します。

© 2000-2025, MetaQuotes Ltd.

OnScrollLineLeft

コントロールの「ScrollLineLeft」（水平スクロールの左へのスクロール）イベントハンドラです。

```
virtual bool  OnScrollLineLeft()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

基本クラスメソッドは何の操作も行わずに常に true を返します。

© 2000-2025, MetaQuotes Ltd.

OnScrollLineRight

コントロールの「ScrollLineRight」(水平スクロールの右へのスクロール) イベントハンドラです。

```
virtual bool  OnScrollLineRight()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

基本クラスメソッドは何の操作も行わずに常に true を返します。

© 2000-2025, MetaQuotes Ltd.

ReBound

CRect クラスの座標を使用してコントロールの新しい座標を設定します。

```
void  ReBound(
   const & CRect  rect      // CRect クラス
  )
```

戻り値

なし

© 2000-2025, MetaQuotes Ltd.

CListView

CListView は ListView の（依存コントロールを含む）複雑なコントロールのクラスです。

説明

CListView クラスはリストコントロールの機能をカプセル化します。

宣言

```
class CListView : public CWndClient
```

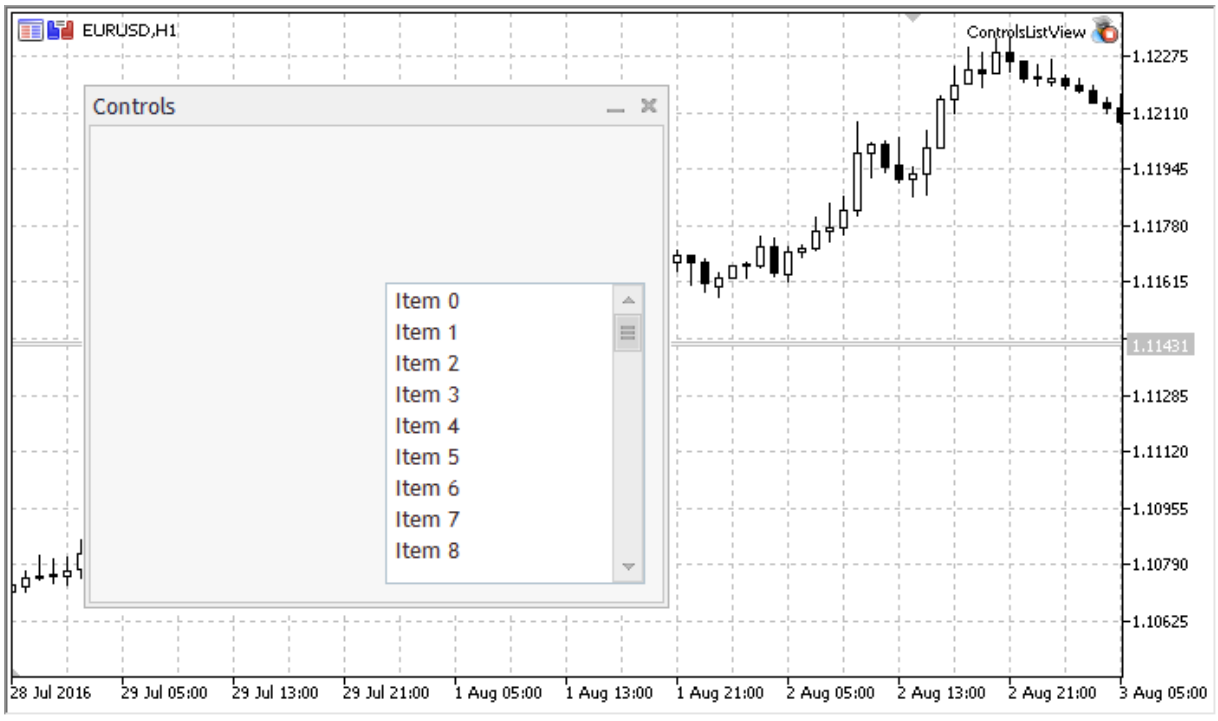
タイトル

```
#include <Controls\ListView.mqh>
```

継承階層

CObject
CWnd
CWndContainer
CWndClient
CListView

下記のコードの結果：



クラスメソッド

© 2000–2025, MetaQuotes Ltd.

Create	
Create	コントロールの作成
チャートイベントハンドラ	
OnEvent	全てのチャートイベントのハンドラ
設定	
TotalView	コントロールに表示される項目の数の設定
追加と削除	
AddItem	項目の追加
Data	
Select	現在のリスト要素をインデックスで選択します。
SelectByText	現在のリスト要素をインデックスで選択します。
SelectByValue	現在のリスト要素を値で選択します。
読み込み専用のデータ	
値	Gets the value of current list element
依存コントロール	
CreateRow	ListView の列を作成します。
内部イベントハンドラ	
OnResize	「Resize」イベントハンドラ(仮想)
依存コントロールのイベントハンドラ	
OnVScrollShow	VScroll 依存コントロールの「Show」イベントハンドラ(仮想)
OnVScrollHide	VScroll 依存コントロールの「Hide」イベントハンドラ(仮想)
OnScrollLineDown	VScroll 依存コントロールの「ScrollLineDown」イベントハンドラ(仮想)
OnScrollLineUp	VScroll 依存コントロールの「ScrollLineUp」イベントハンドラ(仮想)
OnItemClick	「ItemClick」イベントハンドラ(仮想)
Redraw	
Redraw	コントロールを再描画します。
RowState	指定された行の状態を設定します。
CheckView	指定された行の「visibility」をチェックします。

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Type, Compare

クラスから継承されたメソッド CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

クラスから継承されたメソッド CWndContainer

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide, Save, Load

クラスから継承されたメソッド CWndClient

ColorBackground, ColorBorder, BorderType, VScrolled, VScrolled, HScrolled, HScrolled, Id

リストビューを使ったパネルの作成例：

```
//+------------------------------------------------------------------+
//|                                                ControlsListView.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CListView"
#include <Controls\Dialog.mqh>
#include <Controls\ListView.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with
allowance for border width)
#define INDENT_TOP                          (11)      // indent from top (with
allowance for border width)
#define INDENT_RIGHT                        (11)      // indent from right (with
allowance for border width)
#define INDENT_BOTTOM                       (11)      // indent from bottom (with
allowance for border width)
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
```

© 2000–2025, MetaQuotes Ltd.

```
//--- for group controls
#define GROUP_WIDTH                             (150)      // size by X coordinate
#define LIST_HEIGHT                             (179)      // size by Y coordinate
#define RADIO_HEIGHT                            (56)       // size by Y coordinate
#define CHECK_HEIGHT                            (93)       // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CListView         m_list_view;                     // CListView object

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2);
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double
&dparam,const string &sparam);

protected:
   //--- create dependent controls
   bool              CreateListView(void);
   //--- handlers of the dependent controls events
   void              OnChangeListView(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_list_view,OnChangeListView)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
```

© 2000–2025, MetaQuotes Ltd.

```
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2)
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateListView())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "ListView" element                                    |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateListView(void)
  {
//--- coordinates
   int x1=INDENT_LEFT+GROUP_WIDTH+2*CONTROLS_GAP_X;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+2*CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+LIST_HEIGHT-CONTROLS_GAP_Y;
//--- create
   if(!m_list_view.Create(m_chart_id,m_name+"ListView",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!Add(m_list_view))
      return(false);
//--- fill out with strings
   for(int i=0;i<16;i++)
      if(!m_list_view.AddItem("Item "+IntegerToString(i)))
         return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeListView(void)
  {
   Comment(__FUNCTION__+" \""+m_list_view.Select()+"\"");
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
```

© 2000–2025, MetaQuotes Ltd.

```
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- clear comments
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000–2025, MetaQuotes Ltd.

Create

新しい CListView コントロールを作成します。

```
virtual bool  Create(
   const long     chart,       // チャート識別子
   const string   name,        // 名称
   const int      subwin,      // チャートサブウィンドウ
   const int      x1,          // x1 座標
   const int      y1,          // y1 座標
   const int      x2,          // x2 座標
   const int      y2           // y2 座標
  )
```

パラメータ

chart

[in] チャート識別子

name

[in] コントロールの一意の名称

subwin

[in] チャートサブウィンドウ

x1

[in] 左上の隅の X 座標

y1

[in] 左上の隅の Y 座標

x2

[in] 右下の隅の X 座標

y2

[in] 右下の隅の Y 座標

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnEvent

チャート イベント ハンドラ

```
virtual bool  OnEvent(
   const int      id,          // 識別子
   const long&    lparam,      // long 型のイベントパラメータ
   const double&  dparam,      // double 型のイベントパラメータ
   const string&  sparam       // string 型のイベントパラメータ
  )
```

パラメータ

id

[in] イベント識別子

lparam

[in] 参照で渡された long 型のイベント パラメータ

dparam

[in] 参照で渡された double 型のイベント パラメータ

sparam

[in] 参照で渡された string 型のイベント パラメータ

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

TotalView

コントロールに表示される項目の数を設定します。

```
bool  TotalView(
   const int  value      // 表示される項目
  )
```

パラメータ

value

[in] コントロールに表示される項目の数

戻り値

成功の場合は true、それ以外の場合は false

注意事項

表示される項目の数は、一度だけ指定することが出来ます。

© 2000–2025, MetaQuotes Ltd.

AddItem

コントロールに新しい項目を追加します。

```
bool  AddItem(
   const string  item,      // テキスト
   const long    value      // 値
  )
```

パラメータ

item

[in] テキスト

value

[in] long 型の値

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Select

現在のリスト要素をインデックスで選択します。

```
bool  Select(
   const int  index     // インデックス
  )
```

パラメータ

index

[in] 項目のインデックス

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

SelectByText

現在のリスト要素をインデックスで選択します。

```
bool  SelectByText(
   const string  text      // テキスト
  )
```

パラメータ

text

[in] テキスト

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

SelectByValue

現在のリスト要素を値で選択します。

```
bool  SelectByValue(
   const long  value     // 値
  )
```

パラメータ

value

[in] long 型の値

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Value

現在のリスト要素の値を取得します。

```
long  Value()
```

戻り値

現在のリスト要素の値

© 2000–2025, MetaQuotes Ltd.

CreateRow

CListView コントロールの列を作成します。

```
bool  CreateRow(
   const int  index     // インデックス
  )
```

パラメータ

index

[in] 項目のインデックス

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnResize

コントロールの「Resize」イベントハンドラです。

```
virtual bool  OnResize()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnVScrollShow

コントロールの「VScrollHide」(垂直スクロールバーの表示)イベントハンドラです。

```
virtual bool  OnVScrollShow()
```

戻り値

イベントが処理された場合はtrue、それ以外の場合はfalse

© 2000-2025, MetaQuotes Ltd.

OnVScrollHide

コントロールの「VScrollHide」(垂直スクロールバーの非表示)イベントハンドラです。

```
virtual bool  OnVScrollHide()
```

戻り値

イベントが処理された場合はtrue、それ以外の場合はfalse

© 2000-2025, MetaQuotes Ltd.

OnScrollLineDown

コントロールの「ScrollLineDown」(垂直スクロールの下向きスクロール) イベントハンドラです。

```
virtual bool  OnScrollLineDown()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnScrollLineUp

コントロールの「ScrollLineUp」(垂直スクロールの上向きスクロール) イベントハンドラです。

```
virtual bool  OnScrollLineUp()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnItemClick

コントロールの「ItemClick（列のマウスクリック）」イベントハンドラです。

```
virtual bool  OnItemClick()
   const int     index      // 列のインデックス
  )
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Redraw

コントロールを再描画します。

```
bool  Redraw()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

RowState

指定された行の状態を設定します。

```
bool  RowState(
   const int   index      // インデックス
   const bool  select     // 状態
  )
```

パラメータ

index

[in] 行のインデックス

select

[in] 行の状態

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CheckView

指定された行の「visibility」をチェックします。

```
bool  CheckView()
```

戻り値

指定された行が表示されている場合は true それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CComboBox

CComboBox は ComboBox の（依存コントロールを含む）複雑なコントロールのクラスです。

説明

ComboBox は、選択のために意図されたスタティックコントロールと組み合わせたリストボックスで構成されています。コントロールのリストボックス部分は、ユーザがコントロールの隣にあるドロップダウン矢印を選択した際にドロップダウンすることが出来ます。

宣言

```
class CComboBox : public CWndContainer
```

タイトル

```
#include <Controls\ComboBox.mqh>
```

継承階層

CObject
CWnd
CWndContainer
CComboBox

下記のコードの結果：



クラスメソッド

© 2000–2025, MetaQuotes Ltd.

Create	
Create	コントロールの作成
チャートイベントハンドラ	
OnEvent	全てのチャートイベントのハンドラ
Add	
AddItem	項目の追加
設定	
ListViewItems	コントロールに表示される項目の数の設定
Data	
Select	現在のリスト要素をインデックスで選択します。
SelectByText	現在のリスト要素をインデックスで選択します。
SelectByValue	現在のリスト要素を値で選択します。
読み込み専用のデータ	
値	現在のリスト要素の値を取得します。
依存コントロール	
CreateEdit	依存コントロール(編集)を作成します。
CreateButton	依存コントロール(ボタン)を作成します。
CreateList	依存コントロール(リストビュー)を作成します。
依存コントロールのイベントハンドラ	
OnClickEdit	「ClickEdit」イベントハンドラ(仮想)
OnClickButton	「ClickButton」イベントハンドラ(仮想)
OnChangeList	「ChangeList」イベントハンドラ(仮想)
Show/Hide	
ListShow	項目リストの表示
ListHide	項目リストの非表示

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Type, Compare

クラスから継承されたメソッド CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet,

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Type, Compare

StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

クラスから継承されたメソッド CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Hide

コンボボックスを使ったパネルの作成例：

```
//+------------------------------------------------------------------+
//|                                                ControlsComboBox.mq5 |
//|                        Copyright 2015, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2015, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CComboBox"
#include <Controls\Dialog.mqh>
#include <Controls\ComboBox.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with
allowance for border width)
#define INDENT_TOP                          (11)      // indent from top (with
allowance for border width)
#define INDENT_RIGHT                        (11)      // indent from right (with
allowance for border width)
#define INDENT_BOTTOM                       (11)      // indent from bottom (with
allowance for border width)
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
```

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CComboBox         m_combo_box;;                     // CComboBox object

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2);
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double
&dparam,const string &sparam);

protected:
   //--- create dependent controls
   bool              CreateComboBox(void);
   //--- handlers of the dependent controls events
   void              OnChangeComboBox(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_combo_box,OnChangeComboBox)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2)
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
```

© 2000–2025, MetaQuotes Ltd.

```
   if(!CreateComboBox())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "ComboBox" element                                    |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateComboBox(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+EDIT_HEIGHT;
//--- create
   if(!m_combo_box.Create(m_chart_id,m_name+"ComboBox",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!Add(m_combo_box))
      return(false);
//--- fill out with strings
   for(int i=0;i<16;i++)
      if(!m_combo_box.ItemAdd("Item "+IntegerToString(i)))
         return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeComboBox(void)
  {
   Comment(__FUNCTION__+" \""+m_combo_box.Select()+"\"");
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
```

© 2000–2025, MetaQuotes Ltd.

```
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000–2025, MetaQuotes Ltd.

Create

新しい CComboBox コントロールを作成します。

```
virtual bool  Create(
   const long     chart,       // チャート識別子
   const string   name,        // 名称
   const int      subwin,      // チャートサブウィンドウ
   const int      x1,          // x1 座標
   const int      y1,          // y1 座標
   const int      x2,          // x2 座標
   const int      y2           // y2 座標
  )
```

パラメータ

chart

[in] チャート識別子

name

[in] コントロールの一意の名称

subwin

[in] チャートサブウィンドウ

x1

[in] 左上の隅の X 座標

y1

[in] 左上の隅の Y 座標

x2

[in] 右下の隅の X 座標

y2

[in] 右下の隅の Y 座標

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnEvent

チャート イベント ハンドラ

```
virtual bool  OnEvent(
   const int      id,          // 識別子
   const long&    lparam,      // long 型のイベントパラメータ
   const double&  dparam,      // double 型のイベントパラメータ
   const string&  sparam       // string 型のイベントパラメータ
  )
```

パラメータ

id

[in] イベント識別子

lparam

[in] 参照で渡された long 型のイベント パラメータ

dparam

[in] 参照で渡された double 型のイベント パラメータ

sparam

[in] 参照で渡された string 型のイベント パラメータ

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

AddItem

コントロールに新しい項目を追加します。

```
bool  AddItem(
   const string  item,      // テキスト
   const long    value      // 値
  )
```

パラメータ

item

[in] テキスト

value=0

[in] long 型の値

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

ListViewItems

CComboBox コントロールのリスト要素の数を設定します。

```
void  ListViewItems(
   const int     value      // リスト要素の数
  )
```

パラメータ

value

[in] リスト要素の数

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Select

現在のリスト要素をインデックスで選択します。

```
bool  Select(
   const int  index     // インデックス
  )
```

パラメータ

index

[in] 項目のインデックス

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

SelectByText

現在のリスト要素をインデックスで選択します。

```
bool  SelectByText(
   const string  text      // テキスト
  )
```

パラメータ

text

[in] 項目のテキスト

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

SelectByValue

現在のリスト要素を値で選択します。

```
bool  SelectByValue(
   const long  value     // 値
  )
```

パラメータ

value

[in] long 型の値

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Value

現在のリスト要素の値を取得します。

```
long  Value()
```

戻り値

現在のリスト要素の値

© 2000–2025, MetaQuotes Ltd.

CreateEdit

コントロールに依存するコントロール(編集)を作成します。

```
virtual bool  CreateEdit()
```

戻り値

成功の場合はtrue、それ以外の場合はfalse

© 2000-2025, MetaQuotes Ltd.

CreateButton

依存コントロール(ボタン)を作成します。

```
virtual bool  CreateButton()
```

戻り値

成功の場合はtrue、それ以外の場合はfalse

© 2000-2025, MetaQuotes Ltd.

CreateList

依存コントロール(リストビュー)を作成します。

```
virtual bool  CreateList()
```

戻り値

成功の場合はtrue、それ以外の場合はfalse

© 2000-2025, MetaQuotes Ltd.

OnClickEdit

コントロールの「ClickEdit」(「編集」のマウスクリック) イベントハンドラです。

```
virtual bool  OnClickEdit()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnClickButton

コントロールの「ClickButton（ボタンのマウスクリック）」イベントハンドラです。

```
virtual bool  OnClickButton()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnChangeList

「ChangeList」(リストの変更)イベントハンドラです。

```
virtual bool  OnChangeList()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

ListShow

項目リストを表示します。

```
virtual bool  ListShow()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

ListHide

項目リストを非表示にします。

```
virtual bool  ListHide()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CCheckBox

CCheckBox は CheckBox の複雑なコントロールのクラスです。

説明

CCheckBox コントロールは、ユーザが true または false の条件を選択することを可能にするチェックボックスが表示します。

宣言

```
class CCheckBox : public CWndContainer
```

タイトル

```
#include <Controls\CheckBox.mqh>
```

継承階層

CObject
CWnd
CWndContainer
CCheckBox

下記のコードの結果：



クラスメソッド

Create	
Create	コントロールの作成

© 2000–2025, MetaQuotes Ltd.

Create	
チャート イベント ハンドラ	
OnEvent	全てのチャート イベントのハンドラ
プロパティ	
Text	コントロールに関連付けられたラベルを取得/設定します。
Color	コントロールに関連付けられたラベルの色を取得/設定します。
State	
Checked	コントロールがチェックされているかどうかを示す値を取得/設定します。
Data	
値	コントロールに関連付けられた値を取得/設定します。
依存コントロール	
CreateButton	依存コントロール(ボタン)を作成します。
CreateLabel	依存コントロール(ラベル)を作成します。
依存コントロールのイベント ハンドラ	
ClickButton	「ClickButton」イベント ハンドラ(仮想)
ClickLabel	「ClickLabe」イベント ハンドラ(仮想)

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Type, Compare

クラスから継承されたメソッド CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

クラスから継承されたメソッド CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

Example of creating a panel with Checkbox control:

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

© 2000–2025, MetaQuotes Ltd.

```
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CCheckBox"
#include <Controls\Dialog.mqh>
#include <Controls\CheckBox.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with
allowance for border width)
#define INDENT_TOP                          (11)      // indent from top (with
allowance for border width)
#define INDENT_RIGHT                        (11)      // indent from right (with
allowance for border width)
#define INDENT_BOTTOM                       (11)      // indent from bottom (with
allowance for border width)
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CCheckBox         m_check_box1;                    // CCheckBox object
   CCheckBox         m_check_box2;                    // CCheckBox object
public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2);
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double
&dparam,const string &sparam);
protected:
   //--- create dependent controls
```

© 2000–2025, MetaQuotes Ltd.

```
   bool              CreateCheckBox1(void);
   bool              CreateCheckBox2(void);
   //--- handlers of the dependent controls events
   void              OnChangeCheckBox1(void);
   void              OnChangeCheckBox2(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_check_box1,OnChangeCheckBox1)
ON_EVENT(ON_CHANGE,m_check_box2,OnChangeCheckBox2)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2)
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateCheckBox1())
      return(false);
   if(!CreateCheckBox2())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "CheckBox" element                                    |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateCheckBox1(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
```

© 2000–2025, MetaQuotes Ltd.

```
            (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
            (EDIT_HEIGHT+CONTROLS_GAP_Y)+
            (EDIT_HEIGHT+CONTROLS_GAP_Y)+
            (RADIO_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- create
   if(!m_check_box1.Create(m_chart_id,m_name+"CheckBox1",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_check_box1.Text("CheckBox1"))
      return(false);
   if(!m_check_box1.Color(clrBlue))
      return(false);
   if(!Add(m_check_box1))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "CheckBox" element                                    |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateCheckBox2(void)
  {
//--- coordinates
   int x1=INDENT_LEFT+GROUP_WIDTH+CONTROLS_GAP_X;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (RADIO_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- create
   if(!m_check_box2.Create(m_chart_id,m_name+"CheckBox2",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_check_box2.Text("CheckBox2"))
      return(false);
   if(!m_check_box2.Color(clrBlue))
      return(false);
   if(!Add(m_check_box2))
      return(false);
   m_check_box2.Checked(true);
   Comment(__FUNCTION__+" : Checked="+IntegerToString(m_check_box2.Checked()));
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
```

© 2000–2025, MetaQuotes Ltd.

```
void CControlsDialog::OnChangeCheckBox1(void)
  {
   Comment(__FUNCTION__+" : Checked="+IntegerToString(m_check_box1.Checked()));
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeCheckBox2(void)
  {
   Comment(__FUNCTION__+" : Checked="+IntegerToString(m_check_box2.Checked()));
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000–2025, MetaQuotes Ltd.

© 2000–2025, MetaQuotes Ltd.

Create

新しい CCheckBox コントロールを作成します。

```
virtual bool  Create(
   const long     chart,       // チャート識別子
   const string   name,        // 名称
   const int      subwin,      // チャートサブウィンドウ
   const int      x1,          // x1 座標
   const int      y1,          // y1 座標
   const int      x2,          // x2 座標
   const int      y2           // y2 座標
  )
```

パラメータ

chart

[in] チャート識別子

name

[in] コントロールの一意の名称

subwin

[in] チャートサブウィンドウ

x1

[in] 左上の隅の X 座標

y1

[in] 左上の隅の Y 座標

x2

[in] 右下の隅の X 座標

y2

[in] 右下の隅の Y 座標

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnEvent

チャート イベント ハンドラ

```
virtual bool  OnEvent(
   const int      id,          // 識別子
   const long&    lparam,      // long 型のイベントパラメータ
   const double&  dparam,      // double 型のイベントパラメータ
   const string&  sparam       // string 型のイベントパラメータ
  )
```

パラメータ

id

[in] イベント識別子

lparam

[in] 参照で渡された long 型のイベント パラメータ

dparam

[in] 参照で渡された double 型のイベント パラメータ

sparam

[in] 参照で渡された string 型のイベント パラメータ

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Text（Get メソッド）

コントロールに関連付けられたラベルのテキストを取得します。

```
string  Text()
```

戻り値

ラベルのテキスト

Text（Set メソッド）

コントロールに関連付けられたラベルのテキストを設定します。

```
bool  Text(
   const string  value      // テキスト
  )
```

パラメータ

value

[in] ラベルの新しいテキスト

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

Color（Get メソッド）

コントロールに関連付けられたラベルの色を取得します。

```
color  Color()  const
```

戻り値

ラベルの色

Color（Set メソッド）

コントロールに関連付けられたラベルの色を設定します。

```
bool  Color(
   const color  value      // 色
  )
```

パラメータ

value

[in] 新しいラベルの色

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Checked（Get メソッド）

コントロールの状態を取得します。

```
bool  Checked()  const
```

戻り値

コントロールの状態

Checked（Set メソッド）

コントロールの状態を設定します。

```
bool  Checked(
   const bool  flag      // 状態
  )
```

パラメータ

flag

[in] 新しい状態

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Value（Get メソッド）

コントロールに関連付けられた値を取得します。

```
int  Value()  const
```

戻り値

コントロールに関連付けられた値

Value（Set メソッド）

コントロールに関連付けられた値を設定します。

```
void  Value(
   const int  value      // 新しい値
  )
```

パラメータ

value

[in] 新しい値

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

CreateButton

依存コントロール(ボタン)を作成します。

```
virtual bool  CreateButton()
```

戻り値

成功の場合はtrue、それ以外の場合はfalse

© 2000-2025, MetaQuotes Ltd.

CreateLabel

依存コントロール(ラベル)を作成します。

```
virtual bool  CreateLabel()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnClickButton

コントロールの「ClickButton（ボタンのマウスクリック）」イベントハンドラです。

```
virtual bool  OnClickButton()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnClickLabel

コントロールの「ClickLabel（ラベルのマウスクリック）」イベントハンドラです。

```
virtual bool  OnClickLabel()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CCheckGroup

CCheckGroup は CheckGroup の（依存コントロールを含む）複雑なコントロールのクラスです。

説明

CCheckGroup はフラグの表示及び編集を可能にするコントロールを作成する可能性を提供します。

宣言

```
class CCheckGroup : public CWndClient
```

タイトル

```
#include <Controls\CheckGroup.mqh>
```

継承階層

CObject
  CWnd
    CWndContainer
      CWndClient
        CCheckGroup

下記のコードの結果：



クラスメソッド

© 2000–2025, MetaQuotes Ltd.

Create	
Create	コントロールの作成
チャートイベントハンドラ	
OnEvent	全てのチャートイベントのハンドラ
Add	
AddItem	新しい項目の追加
読み込み専用のデータ	
値	コントロールに関連付けられた値を取得します。
依存コントロール	
CreateButton	新しい CCheckBox 項目の作成
依存コントロールのイベントハンドラ	
OnVScrollShow	VScroll 依存コントロールの「Show」イベントハンドラ(仮想)
OnVScrollHide	VScroll 依存コントロールの「Hide」イベントハンドラ(仮想)
OnScrollLineDown	VScroll 依存コントロールの「ScrollLineUp」イベントハンドラ(仮想)
OnScrollLineUp	VScroll 依存コントロールの「ScrollLineDown」イベントハンドラ(仮想)
OnChangeItem	VScroll 依存コントロールの「ChangeItem」イベントハンドラ(仮想)
Redraw	
Redraw	グループの再描画
RowState	指定された項目の状態の設定

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Type, Compare

クラスから継承されたメソッド CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

クラスから継承されたメソッド CWndContainer

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide

クラスから継承されたメソッド CWndClient

ColorBackground, ColorBorder, BorderType, VScrolled, VScrolled, HScrolled, HScrolled, Id

© 2000–2025, MetaQuotes Ltd.

チェックボックスグループを使ったパネルの作成例：

```
//+------------------------------------------------------------------+
//|                                            ControlsCheckGroup.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CCheckGroup"
#include <Controls\Dialog.mqh>
#include <Controls\CheckGroup.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with
allowance for border width)
#define INDENT_TOP                          (11)      // indent from top (with
allowance for border width)
#define INDENT_RIGHT                        (11)      // indent from right (with
allowance for border width)
#define INDENT_BOTTOM                       (11)      // indent from bottom (with
allowance for border width)
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CCheckGroup       m_check_group;                  // CCheckGroup object

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
```

```
   virtual bool      Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2);
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double
&dparam,const string &sparam);

protected:
   //--- create dependent controls
   bool              CreateCheckGroup(void);
   //--- handlers of the dependent controls events
   void              OnChangeCheckGroup(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_check_group,OnChangeCheckGroup)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2)
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateCheckGroup())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "CheckGroup" element                                  |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateCheckGroup(void)
  {
```

© 2000–2025, MetaQuotes Ltd.

```
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (RADIO_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+CHECK_HEIGHT;
//--- create
   if(!m_check_group.Create(m_chart_id,m_name+"CheckGroup",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!Add(m_check_group))
      return(false);
//--- fill out with strings
   for(int i=0;i<5;i++)
      if(!m_check_group.AddItem("Item "+IntegerToString(i),1<<i))
         return(false);
   m_check_group.Check(0,1<<0);
   m_check_group.Check(2,1<<2);
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_check_group.Value()));
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeCheckGroup(void)
  {
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_check_group.Value()));
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(ChartID(),"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

© 2000–2025, MetaQuotes Ltd.

```
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000–2025, MetaQuotes Ltd.

Create

新しい CCheckGroup コントロールを作成します。

```
virtual bool  Create(
   const long     chart,       // チャート識別子
   const string   name,        // 名称
   const int      subwin,      // チャートサブウィンドウ
   const int      x1,          // x1 座標
   const int      y1,          // y1 座標
   const int      x2,          // x2 座標
   const int      y2           // y2 座標
  )
```

パラメータ

chart

[in] チャート識別子

name

[in] コントロールの一意の名称

subwin

[in] チャートサブウィンドウ

x1

[in] 左上の隅の X 座標

y1

[in] 左上の隅の Y 座標

x2

[in] 右下の隅の X 座標

y2

[in] 右下の隅の Y 座標

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnEvent

チャート イベント ハンドラ

```
virtual bool  OnEvent(
   const int      id,          // 識別子
   const long&    lparam,      // long 型のイベントパラメータ
   const double&  dparam,      // double 型のイベントパラメータ
   const string&  sparam       // string 型のイベントパラメータ
  )
```

パラメータ

id

[in] イベント識別子

lparam

[in] 参照で渡された long 型のイベント パラメータ

dparam

[in] 参照で渡された double 型のイベント パラメータ

sparam

[in] 参照で渡された string 型のイベント パラメータ

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

AddItem

コントロールに新しい項目を追加します。

```
bool  AddItem(
   const string  item,      // テキスト
   const long    value      // 値
  )
```

パラメータ

item

[in] テキスト

value=0

[in] long 型の値

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Value

コントロールに関連付けられた値を取得します。

```
long  Value()
```

戻り値

コントロールに関連付けられた値

注意事項

値は CCheckGroup の全ての項目の状態に依存します。

© 2000–2025, MetaQuotes Ltd.

CreateButton

新しい CCheckBox クラスインスタンスを指定されたインデックスで作成します。

```
bool  CreateButton(
   int  index      // インデックス
  )
```

パラメータ

index

[in] CCheckGroup の新しい項目のインデックス

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnVScrollShow

コントロールの「VScrollHide」(垂直スクロールバーの表示)イベントハンドラです。

```
virtual bool  OnVScrollShow()
```

戻り値

イベントが処理された場合はtrue、それ以外の場合はfalse

© 2000-2025, MetaQuotes Ltd.

OnVScrollHide

コントロールの「VScrollHide」(垂直スクロールバーの非表示)イベントハンドラです。

```
virtual bool  OnVScrollHide()
```

戻り値

イベントが処理された場合はtrue、それ以外の場合はfalse

© 2000-2025, MetaQuotes Ltd.

OnScrollLineDown

コントロールの「ScrollLineDown」(垂直スクロールの下向きスクロール) イベントハンドラです。

```
virtual bool  OnScrollLineDown()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnScrollLineUp

コントロールの「ScrollLineUp」(垂直スクロールの上向きスクロール) イベントハンドラです。

```
virtual bool  OnScrollLineUp()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnChangeItem

コントロールの「ChangeItem」(項目の変更)イベントハンドラです。

```
virtual bool  OnChangeItem(
   const int  index      // インデックス
  )
```

パラメータ

index

[in] 変更された項目のインデックス

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

Redraw

コントロールを再描画します。

```
bool  Redraw()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

RowState

指定された項目の状態を設定します。

```
bool  RowState(
   const int   index,      // 項目のインデックス
   const bool  select      // 状態
  )
```

パラメータ

index

[in] 変更する項目のインデックス

select

[in] 新しい状態

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CRadioButton

CRadioButton は RadioButton の複雑なコントロールのクラスです。

説明

CRadioButton 自体は使用されず CRadioGroup 項目の作成に使用されています。

宣言

```
class CRadioButton : public CWndContainer
```

タイトル

```
#include <Controls\RadioButton.mqh>
```

継承階層

CObject
CWnd
CWndContainer
CRadioButton

クラスメソッド

Create	
Create	コントロールの作成
チャートイベントハンドラ	
OnEvent	全てのチャートイベントのハンドラ
プロパティ	
Text	コントロールに関連付けられたラベルを取得/設定します。
Color	コントロールに関連付けられたラベルの色を取得/設定します。
State	
State	状態を取得/設定します。
依存コントロール	
CreateButton	ボタンを作成します。
CreateLabel	ラベルを作成します。
依存コントロールのイベントハンドラ	
OnClickButton	「ClickButton」イベントハンドラ(仮想)
OnClickLabel	「ClickLabe」イベントハンドラ(仮想)

© 2000-2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Type, Compare

クラスから継承されたメソッド CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

クラスから継承されたメソッド CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

© 2000–2025, MetaQuotes Ltd.

Create

新しい CRadioButton コントロールを作成します。

```
virtual bool  Create(
   const long     chart,       // チャート識別子
   const string   name,        // 名称
   const int      subwin,      // チャートサブウィンドウ
   const int      x1,          // x1 座標
   const int      y1,          // y1 座標
   const int      x2,          // x2 座標
   const int      y2           // y2 座標
  )
```

パラメータ

chart

[in] チャート識別子

name

[in] コントロールの一意の名称

subwin

[in] チャートサブウィンドウ

x1

[in] 左上の隅の X 座標

y1

[in] 左上の隅の Y 座標

x2

[in] 右下の隅の X 座標

y2

[in] 右下の隅の Y 座標

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnEvent

チャート イベント ハンドラ

```
virtual bool  OnEvent(
   const int      id,          // 識別子
   const long&    lparam,      // long 型のイベントパラメータ
   const double&  dparam,      // double 型のイベントパラメータ
   const string&  sparam       // string 型のイベントパラメータ
  )
```

パラメータ

id

[in] イベント識別子

lparam

[in] 参照で渡された long 型のイベント パラメータ

dparam

[in] 参照で渡された double 型のイベント パラメータ

sparam

[in] 参照で渡された string 型のイベント パラメータ

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Text（Get メソッド）

コントロールに関連付けられたラベルのテキストを取得します。

```
string  Text()
```

戻り値

ラベルのテキスト

Text（Set メソッド）

コントロールに関連付けられたラベルのテキストを設定します。

```
bool  Text(
   const string  value      // テキスト
  )
```

パラメータ

value

[in] ラベルの新しいテキスト

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

Color（Get メソッド）

コントロールに関連付けられたラベルの色を取得します。

```
color  Color()  const
```

戻り値

ラベルの色

Color（Set メソッド）

コントロールに関連付けられたラベルの色を設定します。

```
bool  Color(
   const color  value      // 色
  )
```

パラメータ

value

[in] 新しいラベルの色

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

State（Get メソッド）

ボタンの状態を取得します。

```
bool  State()  const
```

戻り値

ボタンの状態

State（Set メソッド）

ボタンの状態を設定します。

```
bool  State(
   const bool  flag      // フラグ
  )
```

パラメータ

flag

[in] 新しいボタンの状態

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CreateButton

ボタンを作成します。

```
virtual bool  CreateButton()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

CreateLabel

ラベルを作成します。

```
virtual bool  CreateLabel()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnClickButton

コントロールの「ClickButton（ボタンのマウスクリック）」イベントハンドラです。

```
virtual bool  OnClickButton()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnClickLabel

コントロールの「ClickLabel（ラベルのマウスクリック）」イベントハンドラです。

```
virtual bool  OnClickLabel()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

CRadioGroup

CRadioGroup は RadioGroup の（依存コントロールを含む）複雑なコントロールのクラスです。

説明

CRadioGroup は他の CRadioButton コントロールと組み合わされて、選択グループからの 1 つのオプションの選択を可能にします。

宣言

```
class CRadioGroup : public CWndClient
```

タイトル

```
#include <Controls\RadioGroup.mqh>
```

継承階層

CObject
  CWnd
    CWndContainer
      CWndClient
        CRadioGroup

下記のコードの結果：



クラスメソッド

© 2000–2025, MetaQuotes Ltd.

Create	
Create	コントロールの作成
チャートイベントハンドラ	
OnEvent	全てのチャートイベントのハンドラ
Add	
AddItem	新しい項目の追加
読み込み専用のデータ	
値	コントロールに関連付けられた値を取得します。
依存コントロール	
CreateButton	新しい CRadioButton 項目を作成します。
依存コントロールのイベントハンドラ	
OnVScrollShow	VScroll 依存コントロールの「Show」イベントハンドラ（仮想）
OnVScrollHide	VScroll 依存コントロールの「Hide」イベントハンドラ（仮想）
OnScrollLineDown	VScroll 依存コントロールの「ScrollLineDown」イベントハンドラ（仮想）
OnScrollLineUp	VScroll 依存コントロールの「ScrollLineUp」イベントハンドラ（仮想）
OnChangeItem	「ChangeItem」イベントハンドラ（仮想）
Redraw	
Redraw	グループ項目の再描画
RowState	指定された項目の状態の設定
Select	現在の項目を選択します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Type, Compare

クラスから継承されたメソッド CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

クラスから継承されたメソッド CWndContainer

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide

クラスから継承されたメソッド CWndClient

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Type, Compare

ColorBackground, ColorBorder, BorderType, VScrolled, VScrolled, HScrolled, HScrolled, Id

ラジオボタングループを使ったパネルの作成例：

```
//+------------------------------------------------------------------+
//|                                          ControlsRadioGroup.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CRadioGroup"
#include <Controls\Dialog.mqh>
#include <Controls\RadioGroup.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with
allowance for border width)
#define INDENT_TOP                          (11)      // indent from top (with
allowance for border width)
#define INDENT_RIGHT                        (11)      // indent from right (with
allowance for border width)
#define INDENT_BOTTOM                       (11)      // indent from bottom (with
allowance for border width)
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CRadioGroup       m_radio_group;                  // CRadioGroup object
```

© 2000–2025, MetaQuotes Ltd.

```
public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2);
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double
&dparam,const string &sparam);

protected:
   //--- create dependent controls
   bool              CreateRadioGroup(void);
   //--- handlers of the dependent controls events
   void              OnChangeRadioGroup(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_radio_group,OnChangeRadioGroup)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2)
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateRadioGroup())
      return(false);
//--- succeed
   return(true);
  }
```

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Create the "RadioGroup" element                                  |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateRadioGroup(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+RADIO_HEIGHT;
//--- create
   if(!m_radio_group.Create(m_chart_id,m_name+"RadioGroup",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!Add(m_radio_group))
      return(false);
//--- fill out with strings
   for(int i=0;i<3;i++)
      if(!m_radio_group.AddItem("Item "+IntegerToString(i),1<<i))
         return(false);
   m_radio_group.Value(1<<2);
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_radio_group.Value()));
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeRadioGroup(void)
  {
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_radio_group.Value()));
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
```

© 2000–2025, MetaQuotes Ltd.

```
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- clear comments
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000–2025, MetaQuotes Ltd.

Create

新しい CRadioGroup コントロールを作成します。

```
virtual bool  Create(
   const long     chart,       // チャート識別子
   const string   name,        // 名称
   const int      subwin,      // チャートサブウィンドウ
   const int      x1,          // x1 座標
   const int      y1,          // y1 座標
   const int      x2,          // x2 座標
   const int      y2           // y2 座標
  )
```

パラメータ

chart

[in] チャート識別子

name

[in] コントロールの一意の名称

subwin

[in] チャートサブウィンドウ

x1

[in] 左上の隅の X 座標

y1

[in] 左上の隅の Y 座標

x2

[in] 右下の隅の X 座標

y2

[in] 右下の隅の Y 座標

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnEvent

チャート イベント ハンドラ

```
virtual bool  OnEvent(
   const int      id,          // 識別子
   const long&    lparam,      // long 型のイベントパラメータ
   const double&  dparam,      // double 型のイベントパラメータ
   const string&  sparam       // string 型のイベントパラメータ
  )
```

パラメータ

id

[in] イベント識別子

lparam

[in] 参照で渡された long 型のイベント パラメータ

dparam

[in] 参照で渡された double 型のイベント パラメータ

sparam

[in] 参照で渡された string 型のイベント パラメータ

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

AddItem

コントロールに新しい項目を追加します。

```
bool  AddItem(
   const string  item,      // テキスト
   const long    value=0    // 値
  )
```

パラメータ

item

[in] テキスト

value=0

[in] long 型の値

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Value

コントロールに関連付けられた値を取得します。

```
long  Value()
```

戻り値

コントロールに関連付けられた値

注意事項

値は CRadioGroup の全ての CRadioButton 項目の状態に依存します。

© 2000-2025, MetaQuotes Ltd.

CreateButton

新しい CRadioButton クラスインスタンスを指定されたインデックスで作成します。

```
bool  CreateButton(
   const int  index      // インデックス
  )
```

パラメータ

index

[in] CRadioGroup の新しい項目のインデックス

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnVScrollShow

コントロールの「VScrollHide」(垂直スクロールバーの表示)イベントハンドラです。

```
virtual bool  OnVScrollShow()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnVScrollHide

コントロールの「VScrollHide」(垂直スクロールバーの非表示)イベントハンドラです。

```
virtual bool  OnVScrollHide()
```

戻り値

イベントが処理された場合はtrue、それ以外の場合はfalse

© 2000-2025, MetaQuotes Ltd.

OnScrollLineDown

コントロールの「ScrollLineDown」(垂直スクロールの下向きスクロール) イベントハンドラです。

```
virtual bool  OnScrollLineDown()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnScrollLineUp

コントロールの「ScrollLineUp」(垂直スクロールの上向きスクロール) イベントハンドラです。

```
virtual bool  OnScrollLineUp()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnChangeItem

コントロールの「ChangeItem」(項目の変更)イベントハンドラです。

```
virtual bool  OnChangeItem(
   const int  index      // インデックス
  )
```

パラメータ

index

[in] 変更された項目のインデックス

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

Redraw

コントロールを再描画します。

```
bool  Redraw()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

RowState

指定された項目の状態を設定します。

```
bool  RowState(
   const int   index,      // 項目のインデックス
   const bool  select      // 状態
  )
```

パラメータ

index

[in] 変更する項目のインデックス

select

[in] 新しい状態

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Select

現在の項目を選択します。

```
void  Select(
   const int  index      // インデックス
  )
```

パラメータ

index

[in] 選択される項目のインデックス

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

CSpinEdit

CSpinEdit は SpinEdit の（依存コントロールを含む）複雑なコントロールのクラスです。

説明

CSpinEdit クラスは、整数型の値の編集を可能にするコントロールの作成に意図されています。データは、上ボタンを押すと自動に増加、下ボタンを押すと自動的減少します。

宣言

```
class CSpinEdit : public CWndContainer
```

タイトル

```
#include <Controls\SpinEdit.mqh>
```

継承階層

CObject
  CWnd
    CWndContainer
      CSpinEdit

下記のコードの結果：



クラスメソッド

Create	
Create	コントロールの作成

© 2000–2025, MetaQuotes Ltd.

Create	
チャート イベント ハンドラ	
OnEvent	全てのチャート イベントのハンドラ
プロパティ	
MinValue	最小の許容値を取得/設定します。
MaxValue	最大の許容値を取得/設定します。
State	
値	現在の値を取得/設定します。
依存コントロール	
CreateEdit	依存コントロール(編集)を作成します。
CreateInc	依存コントロール(インクレメント ボタン)を作成します。
CreateDec	依存コントロール(デクレメント ボタン)を作成します。
依存コントロールのイベント ハンドラ	
OnClickInc	「ClickInc」イベント ハンドラ(仮想)
OnClickDec	「ClickDec」イベント ハンドラ(仮想)
内部イベント ハンドラ	
OnChangeValue	「ChangeValue」イベント ハンドラ(仮想)

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Type, Compare

クラスから継承されたメソッド CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

クラスから継承されたメソッド CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

SpinEditコントロールを使ったパネルの作成例:

```
//+------------------------------------------------------------------+
//|                                            ControlsSpinEdit.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CSpinEdit"
#include <Controls\Dialog.mqh>
#include <Controls\SpinEdit.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with
allowance for border width)
#define INDENT_TOP                          (11)      // indent from top (with
allowance for border width)
#define INDENT_RIGHT                        (11)      // indent from right (with
allowance for border width)
#define INDENT_BOTTOM                       (11)      // indent from bottom (with
allowance for border width)
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CSpinEdit         m_spin_edit;                    // CSpinEdit object

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2);
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double
&dparam,const string &sparam);

protected:
```

© 2000–2025, MetaQuotes Ltd.

```
   //--- create dependent controls
   bool              CreateSpinEdit(void);
   //--- handlers of the dependent controls events
   void              OnChangeSpinEdit(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
   ON_EVENT(ON_CHANGE,m_spin_edit,OnChangeSpinEdit)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int
subwin,const int x1,const int y1,const int x2,const int y2)
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateSpinEdit())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "SpinEdit" element                                    |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateSpinEdit(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+(BUTTON_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+EDIT_HEIGHT;
//--- create
   if(!m_spin_edit.Create(m_chart_id,m_name+"SpinEdit",m_subwin,x1,y1,x2,y2))
```

© 2000–2025, MetaQuotes Ltd.

```
      return(false);
   if(!Add(m_spin_edit))
      return(false);
   m_spin_edit.MinValue(10);
   m_spin_edit.MaxValue(100);
   m_spin_edit.Value(50);
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_spin_edit.Value()));
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeSpinEdit(void)
  {
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_spin_edit.Value()));
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- clear comments
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
```

© 2000–2025, MetaQuotes Ltd.

```
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
{
 ExtDialog.ChartEvent(id,lparam,dparam,sparam);
}
```

© 2000–2025, MetaQuotes Ltd.

Create

新しい CSpinEdit コントロールを作成します。

```
virtual bool  Create(
   const long     chart,       // チャート識別子
   const string   name,        // 名称
   const int      subwin,      // チャートサブウィンドウ
   const int      x1,          // x1 座標
   const int      y1,          // y1 座標
   const int      x2,          // x2 座標
   const int      y2           // y2 座標
  )
```

パラメータ

chart

[in] チャート識別子

name

[in] コントロールの一意の名称

subwin

[in] チャートサブウィンドウ

x1

[in] 左上の隅の X 座標

y1

[in] 左上の隅の Y 座標

x2

[in] 右下の隅の X 座標

y2

[in] 右下の隅の Y 座標

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

OnEvent

チャート イベント ハンドラ

```
virtual bool  OnEvent(
   const int      id,          // 識別子
   const long&    lparam,      // long 型のイベントパラメータ
   const double&  dparam,      // double 型のイベントパラメータ
   const string&  sparam       // string 型のイベントパラメータ
  )
```

パラメータ

id

[in] イベント識別子

lparam

[in] 参照で渡された long 型のイベント パラメータ

dparam

[in] 参照で渡された double 型のイベント パラメータ

sparam

[in] 参照で渡された string 型のイベント パラメータ

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

MinValue（Get メソッド）

コントロールの「MinValue」（最小許容値）プロパティの値を取得します。

```
int  MinValue()  const
```

戻り値

「MinValue」プロパティの値

MinValue（Set メソッド）

コントロールの「MinValue」（最小許容値）プロパティの値を設定します。

```
void  MinValue(
   const int  value      // 新しい値
  )
```

パラメータ

value

[in]「MinValue」プロパティの新しい値

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

MaxValue（Get メソッド）

コントロールの「MaxValue」（最大許容値）プロパティの値を取得します。

```
int  MaxValue()  const
```

戻り値

「MaxValue」プロパティの値

MaxValue（Set メソッド）

コントロールの「MaxValue」（最大許容値）プロパティの値を設定します。

```
void  MaxValue(
   const int  value      // 新しい値
  )
```

パラメータ

value

[in] コントロールの「MaxValue」プロパティの新しい値

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

Value（Get メソッド）

コントロールの「Value（値）」プロパティを取得します。

```
int  Value()  const
```

戻り値

「Value（値）」プロパティ

Value（Set メソッド）

コントロールの「Value（値）」プロパティを設定します。

```
void  Value(
   const int  value      // 値
  )
```

パラメータ

value

[in]「Value（値）」プロパティの新しい値

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

CreateEdit

依存コントロール(CEdit)を作成します。

```
virtual bool  CreateEdit()
```

戻り値

成功の場合はtrue、それ以外の場合はfalse

© 2000-2025, MetaQuotes Ltd.

CreateInc

依存コントロール(インクレメントボタン)を作成します。

```
virtual bool  CreateInc()
```

戻り値

成功の場合はtrue、それ以外の場合はfalse

© 2000-2025, MetaQuotes Ltd.

CreateDec

Creates dependent control (decrement button).

```
virtual bool  CreateDec()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnClickInc

コントロールの「ClickInc（インクレメントボタンのマウス左クリック）」イベントハンドラです。

```
virtual bool  OnClickInc()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnClickDec

コントロールの「ClickDec（デクレメントボタンのマウス左クリック）」イベントハンドラです。

```
virtual bool  OnClickDec()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnChangeValue

コントロールの「ChangeValue」イベントハンドラです。

```
virtual bool  OnChangeValue()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CDialog

CDialog は Dialog の複雑なコントロールのクラスです。

説明

CDialog クラスは、グループ内のいろいろなの関数の制御を組み合わせることを意図しています。

宣言

```
class CDialog : public CWndContainer
```

タイトル

```
#include <Controls\Dialog.mqh>
```

継承階層

CObject
CWnd
CWndContainer
CDialog

直接子孫

CAppDialog

クラスメソッド

Create	
Create	コントロールの作成
チャートイベントハンドラ	
OnEvent	全てのチャートイベントのハンドラ
プロパティ	
Caption	「Caption」プロパティの値を取得/設定します。
Add	
Add	クライアント領域にコントロールを追加します。
依存コントロール	
CreateWhiteBorder	依存コントロール(白い境界)を作成します。
CreateBackground	依存コントロール(背景)を作成します。
CreateCaption	依存コントロール(キャプション)を作成します。
CreateButtonClose	依存コントロール(「close」ボタン)を作成します。
CreateClientArea	依存コントロール(クライアント領域)を作成します。

© 2000-2025, MetaQuotes Ltd.

Create	
依存コントロールのイベントハンドラ	
OnClickCaption	「ClickCaption」イベントハンドラ
OnClickButtonClose	「ClickButtonClose」イベントハンドラ
クライアント領域へのアクセス	
ClientAreaVisible	クライアント領域が表示されているかを示す値を設定します。
ClientAreaLeft	コントロールクライアント領域の左上隅のX座標を取得します。
ClientAreaTop	コントロールクライアント領域の左上隅のY座標を取得します。
ClientAreaRight	コントロールクライアント領域の右下隅のX座標を取得します。
ClientAreaBottom	コントロールクライアント領域の右下隅のY座標を取得します。
ClientAreaWidth	クライアント領域の幅を取得します。
ClientAreaHeight	クライアント領域の高さを取得します。
ドラッグイベントハンドラ	
OnDialogDragStart	「DialogDragStart」イベントハンドラ(仮想)
OnDialogDragProcess	「DialogDragProcess」イベントハンドラ(仮想)
OnDialogDragEnd	「DialogDragEnd」イベントハンドラ(仮想)

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Type, Compare

クラスから継承されたメソッド CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

クラスから継承されたメソッド CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

© 2000–2025, MetaQuotes Ltd.

Create

新しい CDialog コントロールを作成します。

```
virtual bool  Create(
   const long     chart,       // チャート識別子
   const string   name,        // 名称
   const int      subwin,      // チャートサブウィンドウ
   const int      x1,          // x1 座標
   const int      y1,          // y1 座標
   const int      x2,          // x2 座標
   const int      y2           // y2 座標
  )
```

パラメータ

chart

[in] チャート識別子

name

[in] コントロールの一意の名称

subwin

[in] チャートサブウィンドウ

x1

[in] 左上の隅の X 座標

y1

[in] 左上の隅の Y 座標

x2

[in] 右下の隅の X 座標

y2

[in] 右下の隅の Y 座標

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnEvent

チャート イベント ハンドラ

```
virtual bool  OnEvent(
   const int      id,          // 識別子
   const long&    lparam,      // long 型のイベントパラメータ
   const double&  dparam,      // double 型のイベントパラメータ
   const string&  sparam       // string 型のイベントパラメータ
  )
```

パラメータ

id

[in] イベント識別子

lparam

[in] 参照で渡された long 型のイベント パラメータ

dparam

[in] 参照で渡された double 型のイベント パラメータ

sparam

[in] 参照で渡された string 型のイベント パラメータ

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Caption（Get メソッド）

CDialog コントロールの「Caption」プロパティを取得します。

```
string  MinValue()  const
```

戻り値

「Caption」プロパティ

Caption（Set メソッド）

CDialog コントロールの「Caption」プロパティを設定します。

```
bool  Caption(
   const string  text      // テキスト
  )
```

パラメータ

text

[in]「Caption」プロパティの新しい値

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Add

ポインタによってクライアント領域にコントロールを追加します。

```
bool  Add(
   CWnd   *control,         // ポインタ
  )
```

パラメータ

control

[in] コントロールのポインタ

戻り値

成功の場合は true、それ以外の場合は false

Add

クライアント領域にコントロールを参照によって追加します。

```
bool  Add(
   CWnd   &control,         // 参照
  )
```

パラメータ

control

[in] コントロールへの参照

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CreateWhiteBorder

依存コントロール(白い境界)を作成します。

```
virtual bool  CreateWhiteBorder()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

CreateBackground

依存コントロール(背景)を作成します。

```
virtual bool  CreateBackground()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

CreateCaption

依存コントロール(キャプション)を作成します。

```
virtual bool  CreateCaption()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CreateButtonClose

依存コントロール(「close」ボタン)を作成します。

```
virtual bool  CreateButtonClose()
```

戻り値

成功の場合はtrue、それ以外の場合はfalse

© 2000-2025, MetaQuotes Ltd.

CreateClientArea

依存コントロール（クライアント領域）を作成します。

```
virtual bool  CreateClientArea()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

OnClickCaption

コントロールの「ClickCaption」イベントハンドラです。

```
virtual bool  OnClickCaption()
```

戻り値

成功の場合はtrue、それ以外の場合はfalse

© 2000-2025, MetaQuotes Ltd.

OnClickButtonClose

コントロールの「ClickButtonClose」イベントハンドラです。

```
virtual bool  OnClickButtonClose()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

ClientAreaVisible

クライアント領域が表示されているかを示すフラグを設定します。

```
bool  ClientAreaVisible(
   const bool  visible      // 可視性フラグ
  )
```

パラメータ

visible

[in] 可視性フラグ

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

ClientAreaLeft

コントロールクライアント領域の左上隅のX座標を取得します。

```
int  ClientAreaLeft()
```

戻り値

コントロールクライアント領域の左上隅のX座標

© 2000-2025, MetaQuotes Ltd.

ClientAreaTop

コントロールクライアント領域の左上隅のＹ座標を取得します。

```
int  ClientAreaTop()
```

戻り値

コントロールクライアント領域の左上隅のＹ座標

© 2000–2025, MetaQuotes Ltd.

ClientAreaRight

コントロールクライアント領域の右下隅のX座標を取得します。

```
int  ClientAreaTop()
```

戻り値

コントロールクライアント領域の右下隅のX座標

© 2000-2025, MetaQuotes Ltd.

ClientAreaBottom

コントロールクライアント領域の右下隅のＹ座標を取得します。

```
int  ClientAreaBottom()
```

戻り値

コントロールクライアント領域の右下隅のＹ座標

© 2000–2025, MetaQuotes Ltd.

ClientAreaWidth

コントロールクライアント領域の幅を取得します。

```
int  ClientAreaWidth()
```

戻り値

コントロールクライアント領域の幅

© 2000–2025, MetaQuotes Ltd.

ClientAreaHeight

コントロールクライアント領域の高さを取得します。

```
int  ClientAreaHeight()
```

戻り値

コントロールクライアント領域の高さ

© 2000–2025, MetaQuotes Ltd.

OnDialogDragStart

コントロールの「DialogDragStart」イベントハンドラです。

```
virtual bool  OnDialogDragStart()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

「DialogDragStart」イベントは、コントロールのドラッグの開始時に発生します。

© 2000–2025, MetaQuotes Ltd.

OnDialogDragProcess

コントロールの「DialogDragProcess」イベントハンドラです。

```
virtual bool  OnDialogDragProcess()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

「DialogDragProcess」イベントは、コントロールのドラッグ時に発生します。

© 2000-2025, MetaQuotes Ltd.

OnDialogDragEnd

コントロールの「DialogDragEnd」イベントハンドラです。

```
virtual bool  OnDialogDragEnd()
```

戻り値

イベントが処理された場合は true、それ以外の場合は false

注意事項

「DialogDragEnd」イベントは、コントロールのドラッグの終了時に発生します。

© 2000–2025, MetaQuotes Ltd.

CAppDialog

CAppDialog は、アプリケーションダイアログの（依存コントロールを含む）複雑なコントロールのクラスです。

説明

CAppDialog クラスは、MQL5 プログラム内のグループ内のいろいろな関数の制御を組み合わせることを意図しています。

宣言

```
class CAppDialog : public CDialog
```

タイトル

```
#include <Controls\Dialog.mqh>
```

継承階層

CObject
CWnd
CWndContainer
CDialog
CAppDialog

クラスメソッド

作成と破壊	
Create	コントロールの作成
Destroy	コントロールの破壊
イベント処理	
OnEvent	全てのチャートイベントのハンドラ
Run	
Run	コントロールの実行
チャートイベント処理	
ChartEvent	全てのチャートイベントのハンドラ
設定	
Minimized	コントロールが最小化されているかを示す値を設定します。
Save/Load	
IniFileSave	コントロールの状態をファイルに保存します。
IniFileLoad	コントロールの状態をファイルから読み込みます。

© 2000–2025, MetaQuotes Ltd.

作成と破壊	
IniFileName	コントロールの状態を読み込み/保存するためのファイル名を設定します。
IniFileExt	コントロールの状態を読み込み/保存するためのファイル拡張子を/設定します。
初期化	
CreateCommon	共通の初期化メソッド
CreateExpert	エキスパートアドバイザーの初期化メソッド
CreateIndicator	指標の初期化メソッド
依存コントロール	
CreateButtonMinMax	依存コントロールを作成します。(ボタンを最小化/最大化)
依存コントロールのイベントハンドラ	
OnClickButtonClose	「ClickButtonClose」イベントハンドラ(仮想)
OnClickButtonMinMax	「ClickButtonMinMax」イベントハンドラ(仮想)
外部イベント	
OnAnotherApplicationClose	外部イベントハンドラ(仮想)
メソッド	
Rebound	CRect クラスの座標を使用してコントロールの新しい座標を設定します。
Minimize	最小化した状態でコントロールを表示します。
Maximize	最大化(元に返す)状態のコントロールを表示します。
CreateInstanceId	コントロールオブジェクト名の一意のIDを作成します。
ProgramName	MQL5 プログラムの名称を取得します。
SubwinOff	コントロールサブウィンドウの Y オフセットを取得します。

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Type, Compare

クラスから継承されたメソッド CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

クラスから継承されたメソッド CWndContainer

© 2000–2025, MetaQuotes Ltd.

クラスから継承されたメソッド CObject

Prev, Prev, Next, Next, Type, Compare

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

クラスから継承されたメソッド CDialog

Caption, Caption, Add, Add

© 2000–2025, MetaQuotes Ltd.

Create

新しい CAppDialog コントロールを作成します。

```
virtual bool  Create(
   const long     chart,       // チャート識別子
   const string   name,        // 名称
   const int      subwin,      // チャートサブウィンドウ
   const int      x1,          // x1 座標
   const int      y1,          // y1 座標
   const int      x2,          // x2 座標
   const int      y2           // y2 座標
  )
```

パラメータ

chart

[in] チャート識別子

name

[in] コントロールの一意の名称

subwin

[in] チャートサブウィンドウ

x1

[in] 左上の隅の X 座標

y1

[in] 左上の隅の Y 座標

x2

[in] 右下の隅の X 座標

y2

[in] 右下の隅の Y 座標

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Destroy

CAppDialogコントロールの初期化解除メソッドです。

```
virtual void  Destroy(
   const int  reason=REASON_PROGRAM    // 理由コード
)
```

パラメータ

reason

[in] 初期化解除の理由コード。REASON_PROGRAMがデフォルトで設定されています。

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

OnEvent

チャート イベント ハンドラ

```
virtual bool  OnEvent(
   const int      id,          // 識別子
   const long&    lparam,      // long 型のイベントパラメータ
   const double&  dparam,      // double 型のイベントパラメータ
   const string&  sparam       // string 型のイベントパラメータ
  )
```

パラメータ

id

[in] イベント識別子

lparam

[in] 参照で渡された long 型のイベント パラメータ

dparam

[in] 参照で渡された double 型のイベント パラメータ

sparam

[in] 参照で渡された string 型のイベント パラメータ

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Run

コントロールを実行します。

```
bool  Run()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

ChartEvent

チャート イベント ハンドラ

```
virtual bool  ChartEvent(
   const int      id,          // 識別子
   const long&    lparam,      // long 型のイベントパラメータ
   const double&  dparam,      // double 型のイベントパラメータ
   const string&  sparam       // string 型のイベントパラメータ
  )
```

パラメータ

id

[in] イベント識別子

lparam

[in] 参照で渡された long 型のイベント パラメータ

dparam

[in] 参照で渡された double 型のイベント パラメータ

sparam

[in] 参照で渡された string 型のイベント パラメータ

戻り値

イベントが処理された場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Minimized

コントロールの「Minimized」プロパティの値を設定します。

```
bool  Minimized(
   const bool  flag      // 状態
  )
```

パラメータ

flag

[in] 新しい状態

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

IniFileSave

コントロールの状態をファイルに保存します。

```
void  IniFileSave()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

IniFileLoad

コントロールの状態をファイルから読み込みます。

```
void  IniFileLoad()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

IniFileName

コントロールの状態を読み込み/保存するためのファイル名を設定します。

```
virtual string  IniFileName()  const
```

戻り値

コントロールの状態を読み込み/保存するファイルの名称.

注意事項

ファイル名は、MQL5 プログラムを起動しているエキスパートアドバイザー/指標及び作業シンボルの名称を含みます。

© 2000-2025, MetaQuotes Ltd.

IniFileExt

コントロールの状態を読み込み/保存するファイルの拡張子を/設定します。

```
virtual string  IniFileExt()  const
```

戻り値

コントロールの状態を読み込み/保存するファイルの拡張子

© 2000-2025, MetaQuotes Ltd.

CreateCommon

共通の初期化メソッドです。

```
bool  CreateCommon(
   const long     chart,       // チャート識別子
   const string   name,        // 名称
   const int      subwin,      // チャートサブウィンドウ
  )
```

パラメータ

chart

[in] チャート識別子

name

[in] コントロールの一意の名称

subwin

[in] チャートサブウィンドウ

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CreateExpert

エキスパートアドバイザーの初期化メソッドです。

```
bool  CreateExpert(
   const int      x1,          // x1 座標
   const int      y1,          // y1 座標
   const int      x2,          // x2 座標
   const int      y2           // y2 座標
  )
```

パラメータ

x1

[in] 左上の隅のX座標

y1

[in] 左上の隅のY座標

x2

[in] 右下の隅のX座標

y2

[in] 右下の隅のY座標

戻り値

成功の場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

CreateIndicator

指標の初期化メソッドです。

```
bool  CreateIndicator(
   const int      x1,          // x1 座標
   const int      y1,          // y1 座標
   const int      x2,          // x2 座標
   const int      y2           // y2 座標
  )
```

パラメータ

x1

[in] 左上の隅のX座標

y1

[in] 左上の隅のY座標

x2

[in] 右下の隅のX座標

y2

[in] 右下の隅のY座標

戻り値

成功の場合はtrue、それ以外の場合はfalse

© 2000–2025, MetaQuotes Ltd.

CreateButtonMinMax

依存コントロールを作成します。（ボタンを最小化/最大化）

```
virtual void  CreateButtonMinMax()
```

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

OnClickButtonClose

コントロールの「ClickButtonClose（「close」ボタンへのマウスクリック）」イベントハンドラです。

```
virtual void  OnClickButtonClose()
```

戻り値

なし

© 2000-2025, MetaQuotes Ltd.

OnClickButtonMinMax

コントロールの「ClickButtonMinMax（最大化/最小化ボタンのマウスクリック）」イベントハンドラです。

```
virtual void  OnClickButtonClose()
```

戻り値

なし

© 2000-2025, MetaQuotes Ltd.

OnAnotherApplicationClose

外部イベントのハンドラです。

```
virtual void  OnAnotherApplicationClose()
```

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

Rebound

CRect クラスの座標を使用してコントロールの新しい座標を設定します。

```
bool  Rebound(
   const & CRect  rect      // CRect クラス
  )
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000-2025, MetaQuotes Ltd.

Minimize

コントロールを最小化した状態で表示します。

```
virtual void  Minimize()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

Maximize

コントロールを最大化（元に返す）状態で表示します。

```
virtual void  Maximize()
```

戻り値

成功の場合は true、それ以外の場合は false

© 2000–2025, MetaQuotes Ltd.

CreateInstanceId

コントロールオブジェクト名の一意のIDを作成します。

```
string  CreateInstanceId()
```

戻り値

オブジェクト名のプレフィックス

© 2000–2025, MetaQuotes Ltd.

ProgramName

MQL5 プログラムの名 称を取 得します。

```
string  ProgramName()
```

戻り値

MQL5 プログラムの名 称

© 2000–2025, MetaQuotes Ltd.

SubwinOff

コントロールサブウィンドウの Y オフセットを取得します。

```
void  SubwinOff()
```

戻り値

なし

© 2000–2025, MetaQuotes Ltd.

MQL4 からMQL5 への移行

MQL5 は、その前身であり多数の指標、スクリプト、及びエキスパートアドバイザーを提供したMQL4 プログラミング言語が進化したものです 。この新しいプログラミング言語は、前世代の言語と最大限の互換性を保つものでありますが、これらの言語の間にはいくつかの違いが存在します。プログラムを移行する際には、これらの違いに注意する必要があります。

このセクションは、MQL4になじみのあるプログラマの MQL5 言語へのコード適応の簡易化を意図した情報が含まれています。

初めに留意点がいくつかあります。

- MQL5 言語は start()、init() 及び deinit() 関数を含みません。
- 指標バッファの数に制限はありません。
- DLLは、エキスパートアドバイザー(またはその他の MQL5 プログラム)をロードした直後にロードされます。
- 論理条件のチェックは短縮されます。
- 配列の制限が超えられた場合、実行が(重大エラーの発生とともに)終了します。
- 演算子の優先順位は C++ と同じです。
- MQL5 言語は(文字列から数値の場合にも)暗黙の型キャストを提供しています。
- 文字列を除いたローカル変数は自動的に初期化されません。
- 一般的なローカル配列は自動的に削除されます。

特別関数 init、start 及び deinit

MQL4 言語では、(*mqh インクルードファイルとライブラリファイルを除いて)指標、スクリプトまたはエキスパートアドバイザーで使用出来る 3 つの事前定義された関数が含まれていました。MQL5 では、これらの関数の代わりに類似した関数が存在します。下記の表は関数のおおよその対応を示しています。

MQL4	MQL5
init	OnInit
start	OnStart
deinit	OnDeinit

OnInit 及び OnDeinit 関数は MQL4 の init 及び deinit と同じ役割を果たします。これらは、MQL5 プログラムの初期化と初期化解除中に実行されるコードを見つけるために設計されています。MQL4 関数の名称を変更する、または、名称を変更せずに対応した場所でMQL5 関数の呼び出しをする選択肢があります。

例:

```
void OnInit()
  {
//--- 初期化の際に関数を呼び出す
   init();
  }
void OnDeinit(const int reason)
  {
//--- 初期化解除の際に関数を呼び出す
```

© 2000–2025, MetaQuotes Ltd.

```
   deinit();
//---
  }
```

start 関数はスクリプトでのみ OnStart で置き換えられます。エキスパートアドバイザーならびに指標ではそれぞれOnTick 及び OnCalculate が使用されるべきです。MQL5 プログラム動作中に実行されるコードはこれらの 3 つの関数内に置かれるべきです。

MQL5 プログラム	メインファンクション
script	OnStart
indicator	OnCalculate
エキスパートアドバイザー	OnTick

指標またはスクリプトコードにメインファンクションが含まれていない場合、もしくは関数名が必要なものと異なる場合、この関数の呼び出しは実行されません。つまり、スクリプトのソースコードが OnStart を含まない場合には、そのようなコードはエキスパートアドバイザーとしてコンパイルされます。

コードが OnCalculate 関数を含まない指標のコンパイルは不可能です。

定義済み変数

MQL5 には Ask、Bid、Bars といった事前定義された変数はありません。Point 及び Digits 変数はわずかに異なるスペルを持っています。

MQL4	MQL5
Digits	_Digits
Point	_Point
	_LastError
	_Period
	_Symbol
	_StopFlag
	_UninitReason

時系列へのアクセス

MQL5 では Open[]、High[]、Low[]、Close[]、Volume[] 及び Time[] の定義済み時系列は存在しません。時系列の必要な深さは、対応する時系列アクセス関数を使用して設定出来ます。

エキスパートアドバイザー

MQL4 と異なって、MQL5 の エキスパートアドバイザーは新しいティックの受信 イベント を処理する関数である OnTick を必要としません。（MQL4 iのstart 関数は新しいティックが受信された際に実行されます。）

© 2000–2025, MetaQuotes Ltd.

MQL5 では、エキスパートアドバイザーは複数の種類のイベントの事前定義されたハンドラ関数を含むことが出来ます。

- OnTick – 新しいティックの受信
- OnTimer – タイマーイベント
- OnTrade – 取引イベント
- OnChartEvent – キーボードやマウスからの入力のイベント、グラフィックオブジェクトの移動イベント、ラベル編集オブジェクトの入力フィールドでのテキスト編集完了のイベント
- OnBookEvent – 板情報の状態変化のイベント

カスタム指標

MQL4 では、指標バッファの数は制限されていて 8 を超えられません。MQL5 にはこのような制限はありませんが、それぞれの指標バッファが端末で存在するためにメモリの特定の部分の割り当てを必要とするので、この新しい権限は悪用されるべきではありません。

MQL4 で提供されたカスタム指標プロットは 6 種類のみでしたが、MQL5 では 18 の描画スタイルが提供されています。描画の種類の名称は変更されていませんが、指標のグラフィク表現のイデオロギーが大幅に変更されました。

指標バッファの索引付けの方向も異なります。デフォルトでは、MQL5 の全ての指標バッファが通常の配列のように振る舞います。すなわち、0 インデックスが付けられた要素は履歴中で最も古いものであり、インデックスの増加に伴い、最も古いデータから最新のデータへの移動がなされます。

MQL4 から保存された唯一のカスタム指標 関数は SetIndexBuffer です。しかしその呼び出しが変更されました。MQL5 では、指標バッファとリンクされた配列に格納されるデータ型を指定する必要があります。

カスタム指標のプロパティも変化し、拡大しています。時系列アクセスに使用される関数も加わったため、計算アルゴリズムを全体として考え直す必要があります。

グラフィックオブジェクト

MQL5 におけるグラフィックオブジェクトの数は大幅に増加しています。その上、グラフィックオブジェクトは、任意の時間軸のチャートで秒単位の精度をもって配置することが出来ます。オブジェクトアンカーポイントは現在の価格チャートにおいてバーの開始時刻に丸められていません。

矢印、テキスト及びラベルオブジェクトでは結合メソッドを、そして、ラベル、ボタン、チャート、ビットマップラベル及び編集オブジェクトではオブジェクトが接続されるチャートのコーナーを指定することが出来ます。

© 2000–2025, MetaQuotes Ltd.

MQL5関数の表

アルファベット順の MQL5 関数リスト

関数	アクション	セクション。
AccountInfoDouble	対応する口座のプロパティの double 型の値を返します。	口座情報
AccountInfoInteger	対応する口座のプロパティの integer 型(bool、int または long)の値を返します。	口座情報
AccountInfoString	対応する口座のプロパティの string 型の値を返します。	口座情報
acos	ラジアンで x のアークコサインを返します。	数学関数
Alert	別ウィンドウにメッセージを表示します。	共通関数
ArrayBsearch	昇順にソートされた多次元数値配列で指定された値を検索します。	配列関数
ArrayCompare	基本データ型または複合型オブジェクトを使用しないカスタム構造体から出来た 2 つの配列を比較した結果を返します。	配列関数
ArrayCopy	配列を別の配列に複製します。	配列関数
ArrayFill	配列を指定された値に設定します。	配列関数
ArrayFree	動的配列のバッファを解放し、ゼロ次元のサイズを 0 に設定します。	配列関数
ArrayGetAsSeries	配列の索引付けの方向をチェックします。	配列関数
ArrayInitialize	数値配列の全ての要素を 1 つの値に設定します。	配列関数
ArrayIsDynamic	配列が動的であるかどうかをチェックします。	配列関数
ArrayIsSeries	配列が時系列であるかどうかをチェックします。	配列関数
ArrayMaximum	多次元数値配列の最初の次元で最大の要素を検索します。	配列関数
ArrayMinimum	多次元数値配列の最初の次元で最小の要素を検索します。	配列関数

© 2000–2025, MetaQuotes Ltd.

関数	アクション	セクション。
ArrayRange	配列の指定された次元の要素数を返します。	配列関数
ArrayResize	配列の初次元のサイズを新しく設定します。	配列関数
ArraySetAsSeries	配列の索引付けの方向を設定します。	配列関数
ArraySize	配列の要素数を返します。	配列関数
ArraySort	数値配列を初次元によってソートします。	配列関数
ArrayPrint	単純型または単純な構造体の配列を操作ログに出力します	配列関数
ArrayInsert	指定されたインデックスから開始して、指定された数の要素をソース配列からターゲット配列に挿入します	配列関数
ArrayRemove	指定されたインデックスから開始して、配列内の指定された数の要素を削除します	配列関数
ArrayReverse	指定されたインデックスから開始して、配列内の指定された数の要素を反転します	配列関数
ArraySwap	同じ型の2つの動的配列の内容を交換します。	配列関数
asin	ラジアンで x のアークサインを返します。	数学関数
atan	ラジアンで x のアークタンジェントを返します。	数学関数
Bars	指定された銘柄と期間の履歴内のバーの数を返します。	時系列と指標へのアクセス
BarsCalculated	指標バッファで計算されたデータの数、またはエラー(データがまだ計算されていません。)の場合 -1 を返します。	時系列と指標へのアクセス
CalendarCountryById	IDによって国の説明を取得する	経済指標カレンダー
CalendarEventById	IDによってイベントの説明を取得する	経済指標カレンダー
CalendarValueById	IDによってイベントの値を取得する	経済指標カレンダー

© 2000–2025, MetaQuotes Ltd.

関数	アクション	セクション。
CalendarCountries	カレンダーで利用可能な国名の配列を取得する	経済指標カレンダー
CalendarEventByCountry	指定された国コードでカレンダーで利用可能なすべてのイベントの説明の配列を取得する	経済指標カレンダー
CalendarEventByCurrency	指定された通貨でカレンダーで利用可能なすべてのイベントの説明の配列を取得する	経済指標カレンダー
CalendarValueHistoryByEvent	イベントIDによって、指定された期間のすべてのイベントの値の配列を取得する	経済指標カレンダー
CalendarValueHistory	指定された期間のすべてのイベントの値の配列を取得する(国や通貨で並び替え可能)	経済指標カレンダー
CalendarValueLastByEvent	指定されたchange_idで、カレンダーデータベースのステータスからイベントIDの配列をIDで取得する	経済指標カレンダー
CalendarValueLast	指定されたchange_idで、カレンダデータベースのステータスからすべてのイベントの値の配列を取得する(国や通貨で並び替え可能)	経済指標カレンダー
ceil	切り上げされた整数値を返します。	数学関数
CharArrayToString	ANSI シンボルコードをーシンボルの文字列に変換します。	変換関数
ChartApplyTemplate	指定されたファイルの特定のテンプレートをチャートに適用します。	チャート操作
ChartClose	指定されたチャートを閉じます。	チャート操作
ChartFirst	クライアント端末の最初のチャートのIDを返します。	チャート操作
ChartGetDouble	指定されたチャートのプロパティのdouble値を返します。	チャート操作
ChartGetInteger	指定されたチャートのプロパティの整数値を返します。	チャート操作
ChartGetString	指定されたチャートのプロパティの文字列値を返します。	チャート操作
ChartID	現在のチャートのIDを返します。	チャート操作
ChartIndicatorAdd	指定されたハンドルと指標を指定されたチャートウィンドウに追加します。	チャート操作

© 2000–2025, MetaQuotes Ltd.

関数	アクション	セクション。
ChartIndicatorDelete	指定された名称の指標を指定されたチャートウィンドウから削除します。	チャート操作
ChartIndicatorGet	指定されたチャートウィンドウで短縮名によって指定された指標ハンドルを返します。	チャート操作
ChartIndicatorName	指定されたチャートの指標のリスト内の数によって指標の短縮名を返します。	チャート操作
ChartIndicatorsTotal	指定されたチャートウィンドウに適用された全ての指標の数を返します。	チャート操作
ChartNavigate	チャート内の指定された位置を基準に指定された足数によって指定されたチャートのシフトを実行します。	チャート操作
ChartNext	指定されたチャートの次のチャート識別子を返します。	チャート操作
ChartOpen	指定された銘柄と期間で新しいチャートを開きます。	チャート操作
CharToString	シンボルコードを 1 文字の文字列に変換します。	変換関数
ChartPeriod	指定されたチャートの期間値を返します。	チャート操作
ChartPriceOnDropped	エキスパートアドバイザーやスクリプトがドロップされたチャート点の価格座標を返します。	チャート操作
ChartRedraw	指定されたチャートの強制再描画を呼び出します。	チャート操作
ChartSaveTemplate	チャートの設定を指定された名称のテンプレートに保存します。	チャート操作
ChartScreenShot	指定された拡張子に応じてGIF、PNGまたはBMP形式で現在のチャートのスクリーンショットを提供します。	チャート操作
ChartSetDouble	指定されたチャートのプロパティにdouble値を設定します。	チャート操作
ChartSetInteger	指定されたチャートの対応するプロパティに整数値（datetime、int、color、bool または char）を設定します。	チャート操作

© 2000–2025, MetaQuotes Ltd.

関数	アクション	セクション。
ChartSetString	指定されたチャートのプロパティに文字列値を設定します。	チャート操作
ChartSetSymbolPeriod	指定されたチャートの銘柄値と期間を変更します。	チャート操作
ChartSymbol	指定されたチャートの銘柄名を返します。	チャート操作
ChartTimeOnDropped	エキスパートアドバイザーやスクリプトがドロップされたチャート点の時間座標を返します。	チャート操作
ChartTimePriceToXY	チャートの座標を時間/価格表現からXとYの座標に変換します。	チャート操作
ChartWindowFind	指標が描画されるサブウィンドウの数を返します。	チャート操作
ChartWindowOnDropped	エキスパートアドバイザーやスクリプトがドロップされたサブウィンドウチャートの番号（インデックス）を返します。	チャート操作
ChartXOnDropped	エキスパートアドバイザーやスクリプトがドロップされたチャート点のX座標を返します。	チャート操作
ChartXYToTimePrice	チャートの座標をX　とYの座標から時間/価格表現に変換します。	チャート操作
ChartYOnDropped	エキスパートアドバイザーやスクリプトがドロップされたチャート点のY座標を返します。	チャート操作
CheckPointer	オブジェクトポインタの型を返します。	共通関数
CLBufferCreate	OpenCL バッファを作成します。	OpenCL 操作
CLBufferFree	OpenCL バッファを削除します。	OpenCL 操作
CLBufferRead	OpenCL バッファを配列に読み込みます。	OpenCL 操作
CLBufferWrite	配列を OpenCL バッファに書き込みます。	OpenCL 操作
CLContextCreate	OpenCL コンテキストを作成します。	OpenCL 操作
CLContextFree	OpenCL コンテキストを削除します。	OpenCL 操作
CLExecute	OpenCL プログラムを実行します。	OpenCL 操作
CLGetDeviceInfo	OpenCL ドライバからデバイスプロパティを受け取ります。	OpenCL 操作

© 2000–2025, MetaQuotes Ltd.

関数	アクション	セクション。
CLGetInfoInteger	OpenCL オブジェクトまたはデバイスから整数プロパティ値を返します。	OpenCL 操作
CLHandleType	OpenCL ハンドルの型を ENUM_OPENCL_HANDLE_TYPE 列挙の値として返します。	OpenCL 操作
CLKernelCreate	OpenCL start 関数を作成します。	OpenCL 操作
CLKernelFree	OpenCL start 関数を削除します。	OpenCL 操作
CLProgramCreate	OpenCL プログラムをソースコードから作成します。	OpenCL 操作
。CLProgramFree	OpenCL プログラムを削除します。	OpenCL 操作
CLSetKernelArg	OpenCL 関数のパラメータを設定します。	OpenCL 操作
CLSetKernelArgMem	OpenCL バッファを OpenCL 関数のパラメータをして設定します。	OpenCL 操作
ColorToARGB	ARGB 表現を受信するための color 型を uint 型に変換します。	変換関数
ColorToString	color 値を「R, G, B」の文字列に変換します。	変換関数
Comment	チャートの左上隅にコメントを出力します。	共通関数
CopyBuffer	指定された指標の指定されたバッファデータを配列に取得します。	時系列と指標へのアクセス
CopyClose	配列に指定された銘柄と期間のバー終値の履歴データを取得します。	時系列と指標へのアクセス
CopyHigh	指定された銘柄と期間のバー高値の履歴データを配列に取得します。	時系列と指標へのアクセス
CopyLow	指定された銘柄と期間のバー安値の履歴データを配列に取得します。	時系列と指標へのアクセス
CopyOpen	指定された銘柄と期間のバー始値の履歴データを配列に取得します。	時系列と指標へのアクセス
CopyRates	指定された銘柄と期間の Rates 構造体の履歴データを配列に取得します。	時系列と指標へのアクセス

© 2000–2025, MetaQuotes Ltd.

関数	アクション	セクション。
CopyRealVolume	指定された銘柄と期間の取引高の履歴データを配列に取得します。	時系列と指標へのアクセス
CopySpread	指定された銘柄と期間のスプレッドの履歴データを配列に取得します。	時系列と指標へのアクセス
CopyTicks	現在の作業セッションのので端末に蓄積されたティックを配列に取得します。	時系列と指標へのアクセス
CopyTickVolume	指定された銘柄と期間のティックボリュームの履歴データを配列に取得します。	時系列と指標へのアクセス
CopyTime	指定された銘柄と期間のオープン時間の履歴データを配列に取得します。	時系列と指標へのアクセス
cos	数のコサインを返します。	数学関数
CryptDecode	配列からのデータの逆変換を行います。	共通関数
CryptEncode	指定されたメソッドを使用して配列からデータを変換します。	共通関数
CustomSymbolCreate	指定されたグループに指定された名前のカスタム銘柄を作成します。	カスタム銘柄
CustomSymbolDelete	指定された名前のカスタム銘柄を削除します。	カスタム銘柄
CustomSymbolSetInteger	カスタム銘柄の整数型プロパティ値を設定します。	カスタム銘柄
CustomSymbolSetDouble	カスタム銘柄の実数型プロパティ値を設定します。	カスタム銘柄
CustomSymbolSetString	カスタム銘柄の文字列型プロパティ値を設定します。	カスタム銘柄
CustomSymbolSetMarginRate	カスタム銘柄の注文の種類と方向に応じて証拠金率を設定します。	カスタム銘柄
CustomSymbolSetSessionQuote	指定された銘柄と曜日の指定された相場セッションの開始時刻と終了時刻を設定します。	カスタム銘柄
CustomSymbolSetSessionTrade	指定された銘柄と曜日の指定された取引セッションの開始時刻と終了時刻を設定します。	カスタム銘柄

© 2000–2025, MetaQuotes Ltd.

関数	アクション	セクション。
CustomRatesDelete	指定した時間間隔内のカスタム銘柄の価格履歴からすべてのバーを削除します。	カスタム銘柄
CustomRatesReplace	指定された時間間隔内のカスタム銘柄の価格履歴をMqlRates型の配列からのデータで完全に置き換えます。	カスタム銘柄
CustomRatesUpdate	カスタム銘柄履歴に欠けているバーを追加し、既存のデータをMqlRates型配列のものに置き換えます。	カスタム銘柄
CustomTicksAdd	カスタム銘柄の価格履歴にMqlTick型の配列からデータを追加します。カスタム銘柄は気配値ウィンドウで選択されなければなりません。	カスタム銘柄
CustomTicksDelete	指定した時間間隔内のカスタム銘柄の価格履歴からすべてのティックを削除します。	カスタム銘柄
CustomTicksReplace	指定された時間間隔内のカスタム銘柄の価格履歴をMqlTick型の配列からのデータで完全に置き換えます。	カスタム銘柄
CustomBookAdd	銘柄の板情報の構造体を渡します	カスタム銘柄
DatabaseOpen	指定したファイルでデータベースを開くか作成します	データベースの操作
DatabaseClose	データベースを閉じます	データベースの操作
DatabaseImport	ファイルからテーブルにデータをインポートします。	データベースの操作
DatabaseExport	テーブルまたはSQL要求の実行結果をCSVファイルにエクスポートします。ファイルはUTF-8エンコードで作成されます。	データベースの操作
DatabasePrint	テーブルまたはSQL要求の実行結果をエクスパート操作ログに出力します。	データベースの操作
DatabaseTableExists	データベース内の表の存在を確認します	データベースの操作
DatabaseExecute	指定されたデータベースへのリクエストを実行します	データベースの操作

© 2000–2025, MetaQuotes Ltd.

関数	アクション	セクション。
DatabasePrepare	DatabaseRead()を使用して実行できるリクエストのハンドルを作成します	データベースの操作
DatabaseReset	DatabasePrepare()の呼び出し後などに、リクエストをリセットします	データベースの操作
DatabaseBind	リクエストにパラメータ値を設定します	データベースの操作
DatabaseBindArray	パラメータ値として配列を設定します	データベースの操作
DatabaseRead	リクエストの結果として次のエントリに移動します	データベースの操作
DatabaseFinalize	DatabasePrepare()で作成されたリクエストを削除します	データベースの操作
DatabaseTransactionBegin	トランザクションの実行を開始します	データベースの操作
DatabaseTransactionCommit	トランザクションの実行を完了します	データベースの操作
DatabaseTransactionRollback	トランザクションをロールバックします	データベースの操作
DatabaseColumnsCount	リクエスト内のフィールドの数を取得します	データベースの操作
DatabaseColumnName	インデックスでフィールド名を取得します	データベースの操作
DatabaseColumnType	インデックスでフィールドタイプを取得します	データベースの操作
DatabaseColumnSize	バイト単位のフィールドサイズを取得します	データベースの操作
DatabaseColumnText	現在のレコードからフィールド値を文字列として取得します	データベースの操作
DatabaseColumnInteger	現在のレコードからint型の値を取得します	データベースの操作
DatabaseColumnLong	現在のレコードからlong型の値を取得します	データベースの操作
DatabaseColumnDouble	現在のレコードからdouble型の値を取得します	データベースの操作
DatabaseColumnBlob	現在のレコードからフィールド値を配列として取得します	データベースの操作
DebugBreak	デバグ中のプログラムのブレークポイント。	共通関数

© 2000–2025, MetaQuotes Ltd.

関数	アクション	セクション。
Digits	チャートシンボルの価格の値の精度を決定する小数点以下の桁数を返します。	チェックアップ
DoubleToString	指定された精度で数値をテキスト行に変換します。	変換関数
DXContextCreate	指定されたサイズのフレームをレンダリングするためのグラフィックコンテキストを作成します	DirectXの操作
DXContextSetSize	DXContextCreate()で作成されたグラフィックコンテキストのフレームサイズを変更します	DirectXの操作
DXContextSetSize	DXContextCreate()で作成されたグラフィックコンテキストのフレームサイズを取得します	DirectXの操作
DXContextClearColors	レンダリングバッファのすべてのピクセルに指定された色を設定します	DirectXの操作
DXContextClearDepth	深度バッファをクリアします	DirectXの操作
DXContextGetColors	グラフィックコンテキストから指定されたサイズとオフセットの画像を取得します	DirectXの操作
DXContextGetDepth	レンダリングされたフレームの深度バッファを取得します	DirectXの操作
DXBufferCreate	データ配列に基づいて、指定された型のバッファを作成します	DirectXの操作
DXTextureCreate	渡された画像から切り取られた指定サイズの長方形から2Dテクスチャを作成します	DirectXの操作
DXInputCreate	シェーダー入力を作成します	DirectXの操作
DXInputSet	シェーダー入力を設定します	DirectXの操作
DXShaderCreate	指定された種類のシェーダーを作成します	DirectXの操作
DXShaderSetLayout	頂点シェーダーの頂点レイアウトを設定します	DirectXの操作
DXShaderInputsSet	シェーダー入力を設定します	DirectXの操作
DXShaderTexturesSet	シェーダーテクスチャを設定します	DirectXの操作
DXDraw	DXBufferSet()で設定された頂点バッファの頂点をレンダリングします	DirectXの操作

© 2000–2025, MetaQuotes Ltd.

関数	アクション	セクション。
DXDrawIndexed	DXBufferSet()のインデックスバッファによって記述されたグラフィックプリミティブをレンダリングします	DirectXの操作
DXPrimiveTopologySet	DXDrawIndexed()を使用してレンダリングするプリミティブのタイプを設定します	DirectXの操作
DXBufferSet	現在のレンダリングにバッファを設定します	DirectXの操作
DXShaderSet	レンダリング用のシェーダーを設定します	DirectXの操作
DXHandleType	ハンドルの種類を返します	DirectXの操作
DXRelease	ハンドルを解放します	DirectXの操作
EnumToString	任意のタイプの列挙値を文字列に変換します。	変換関数
EventChartCustom	指定されたチャートのカスタムイベントを生成します。	イベント操作
EventKillTimer	現在のチャートのタイマーでイベントが発生するのを停止します。	イベント操作
EventSetMillisecondTimer	現在のチャートで使用される、期間が1秒未満の高分解能タイマーのイベントジェネレータを起動します。	イベント操作
EventSetTimer	現在のチャートで使用される、指定された時間軸のタイマーイベントジェネレータを開始します。	イベント操作
exp	数の指数を返します。	数学関数
ExpertRemove	エキスパートアドバイザーを停止しチャートからアンロードします。	共通関数
fabs	指定された数値の絶対値（係数）を返します。	数学関数
FileClose	以前に開いたファイルを閉じます。	ファイル関数
FileCopy	ファイルをローカルまたは共有フォルダから別のファイルに複製します。	ファイル関数
FileDelete	指定されたファイルを削除します。	ファイル関数
FileFindClose	検索ハンドルを閉じます。	ファイル関数
FileFindFirst	指定されたフィルタに基づいて、ディレクトリ内のファイルの検索を開始します。	ファイル関数

© 2000–2025, MetaQuotes Ltd.

関数	アクション	セクション。
FileFindNext	FileFindFirst() 関数で始めた検索を続けます。	ファイル関数
FileFlush	入力/出力ファイルのバッファに残っている全てのデータをディスクに書き込みます。	ファイル関数
FileGetInteger	ファイルの整数プロパティを取得します。	ファイル関数
FileIsEnding	読み出し処理において、ファイルの終わりを定義します。	ファイル関数
FileIsExist	ファイルの存在をチェックします。	ファイル関数
FileIsLineEnding	読み出し処理において、テキストファイル内の行の終わりを定義します。	ファイル関数
FileMove	ファイルを移動または名称を変更します。	ファイル関数
FileOpen	指定された名称とフラグでファイルを開きます。	ファイル関数
FileReadArray	BIN形式のファイルから文字列を除く全ての型の配列を読み込みます。	ファイル関数
FileReadBool	CSV ファイルから、現在の位置から区切り文字までの文字列（またはテキスト行の終わり）を読み込み、読んだ文字列を bool 型の値に変換します。	ファイル関数
FileReadDatetime	CSV ファイルから、「YYYY.MM.DD HH:MM:SS」、「YYYY.MM.DD」または「HH:MM:SS」フォーマットの文字列を読み込みdatetime 値に変換します。	ファイル関数
FileReadDouble	ファイルポインタの現在位置から double 値を読み込みます。	ファイル関数
FileReadFloat	ファイルポインタの現在位置から float 値を読み込みます。	ファイル関数
FileReadInteger	ファイルポインタの現在位置から int、short または char 値を読み込みます。	ファイル関数
FileReadLong	ファイルポインタの現在位置から long 値を読み込みます。	ファイル関数

© 2000–2025, MetaQuotes Ltd.

関数	アクション	セクション。
FileReadNumber	CSV ファイルから、現在の位置から区切り文字までの文字列（またはテキスト行の終わり）を読み込み、読んだ文字列を double 型の値に変換します。	ファイル関数
FileReadString	ファイルポインタの現在位置から string 値を読み込みます。	ファイル関数
FileReadStruct	バイナリファイルのファイルポインタの現在の位置から、パラメータとして渡される構造体にコンテンツを読み込みます。	ファイル関数
FileSeek	ファイルポインタの位置を指定された位置から指定されたバイト数で移動します。	ファイル関数
FileSize	対応する開かれているファイルのサイズを返します。	ファイル関数
FileTell	対応する開かれているファイルのファイルポインタの現在位置を返します。	ファイル関数
FileWrite	データを CSV または TXT 形式のファイルに書き込みます。	ファイル関数
FileWriteArray	BIN 形式のファイルに文字列を除いた任意の型の配列を書き込みます。	ファイル関数
FileWriteDouble	バイナリファイルのファイルポインタの現在位置にdouble 型の値を書き込みます。	ファイル関数
FileWriteFloat	バイナリファイルのファイルポインタの現在位置にfloat 型の値を書き込みます。	ファイル関数
FileWriteInteger	バイナリファイルのファイルポインタの現在位置に int 型の値を書き込みます。	ファイル関数
FileWriteLong	バイナリファイルのファイルポインタの現在位置に long 型の値を書き込みます。	ファイル関数
FileWriteString	BIN または TXT ファイルのファイルポインタの現在位置に string パラメータの値を書き込みます。	ファイル関数
FileWriteStruct	バイナリファイルのファイルのファイルポインタの現在位置にパラメータとし	ファイル関数

© 2000–2025, MetaQuotes Ltd.

関数	アクション	セクション。
	て割らされた構造体のコンテンツを書き込みます。	
floor	切り下げされた整数値を返します。	数学関数
fmax	2つの数値の最大値を返します。	数学関数
fmin	2つの数値の最小値を返します。	数学関数
fmod	2　つの数の分割後の実際の余りを返します。	数学関数
FolderClean	指定されたフォルダ内の全てのファイルを削除します。	ファイル関数
FolderCreate	Files　ディレクトリ内にフォルダを作成します。	ファイル関数
FolderDelete	選択されたディレクトリを削除します。フォルダが空でない場合は、削除出来ません。	ファイル関数
FrameAdd	データとフレームを追加します。	最適化の結果の操作
FrameFilter	フレーム読み込みフィルタを設定して、ポインタを先頭に移動します。	最適化の結果の操作
FrameFirst	フレーム読み込みのポインタを先頭に移動し、以前に設定したフィルタをリセットします。	最適化の結果の操作
FrameInputs	フレーム形成に使用される入力パラメータを受け取ります。	最適化の結果の操作
FrameNext	フレームを読んでポインタを次に移動します。	最適化の結果の操作
GetLastError	最後に発生したエラーを返します。	チェックアップ
GetPointer	オブジェクトポインタを返します。	共通関数
GetTickCount	システムが起動してからの経過時間をミリ秒数で返します。	共通関数
GlobalVariableCheck	指定された名称のグローバル変数の存在をチェックします。	端末のグローバル変数
GlobalVariableDel	グローバル変数を削除します。	端末のグローバル変数
GlobalVariableGet	グローバル変数の値を返します。	端末のグローバル変数
GlobalVariableName	グローバル変数の名称をグローバル変数のリスト内の序数で返します。	端末のグローバル変数

© 2000–2025, MetaQuotes Ltd.

関数	アクション	セクション。
GlobalVariablesDeleteAll	指定されたプレフィックスを持つグローバル変数を削除します。	端末のグローバル変数
GlobalVariableSet	グローバル変数に新しい値を設定します。	端末のグローバル変数
GlobalVariableSetOnCondition	条件付きで既存するグローバル変数に新しい値を設定します。	端末のグローバル変数
GlobalVariablesFlush	強制的にディスクに全てのグローバル変数の内容を保存します。	端末のグローバル変数
GlobalVariablesTotal	グローバル変数の合計数を返します。	端末のグローバル変数
GlobalVariableTemp	グローバル変数に端末の現在のセッションでのみ存在する新しい値を設定します。	端末のグローバル変数
GlobalVariableTime	グローバル変数の最終アクセス時刻を返します。	端末のグローバル変数
HistoryDealGetDouble	履歴内で指定された 約定の double 型プロパティを返します。	取引関数
HistoryDealGetInteger	履歴内で指定された 約定の datetime または int プロパティを返します。	取引関数
HistoryDealGetString	履歴内で指定された 約定の string 型プロパティを返します。	取引関数
HistoryDealGetTicket	履歴内の取引に対応するチケットを返します。	取引関数
HistoryDealSelect	更に適切な関数で呼び出すために履歴内の取引を選択します。	取引関数
HistoryDealsTotal	履歴内の取引数を返します。	取引関数
HistoryOrderGetDouble	履歴内で指定された double 注文プロパティを返します。	取引関数
HistoryOrderGetInteger	履歴内で指定された datetime または int 注文プロパティを返します。	取引関数
HistoryOrderGetString	履歴内で指定された string 注文プロパティを返します。	取引関数
HistoryOrderGetTicket	履歴内の注文に対応するチケットを返します。	取引関数
HistoryOrderSelect	操作を続けるために履歴内の注文を選択します。	取引関数

© 2000–2025, MetaQuotes Ltd.

関数	アクション	セクション。
HistoryOrdersTotal	履歴内の注文の数を返します。	取引関数
HistorySelect	指定されたサーバタイム期間のトランザクションと注文履歴を取得します。	取引関数
HistorySelectByPosition	指定されたポジション識別子を持った取引履歴をリクエストします。	取引関数
iBars	該当する銘柄と期間の履歴内のバーの数を返します。	時系列と指標へのアクセス
iBarShift	この関数は、指定された時刻に対応するバーのインデックスを返します。	時系列と指標へのアクセス
iClose	対応するチャートのバー（'shift'パラメータで示される）の終値を返します。	時系列と指標へのアクセス
iHigh	対応するチャートのバー（'shift'パラメータで示される）の高値を返します。	時系列と指標へのアクセス
iHighest	対応するチャート上で見つかった最大値のインデックスを返します（現在のバーとの相対的なシフト）。	時系列と指標へのアクセス
iLow	対応するチャートのバー（'shift'パラメータで示される）の安値を返します。	時系列と指標へのアクセス
iLowest	該当するチャート上で見つかった最小値のインデックスを返します（現在のバーとの相対的なシフト）。	時系列と指標へのアクセス
iOpen	対応するチャートのバー（'shift'パラメータで示される）の始値を返します。	時系列と指標へのアクセス
iTime	対応するチャートのバー（'shift'パラメータで示される）の時間を返します。	時系列と指標へのアクセス
iTickVolume	対応するチャートのバー（'shift'パラメータで示される）のティックボリュームを返します。	時系列と指標へのアクセス
iRealVolume	対応するチャートのバー（'shift'パラメータで示される）の実ボリュームを返します。	時系列と指標へのアクセス
iVolume	対応するチャートのバー（'shift'パラメータで示される）のティックボリュー	時系列と指標へのアクセス

© 2000–2025, MetaQuotes Ltd.

関数	アクション	セクション。
	ムを返します。	
iSpread	対応するチャートのバー('shift'パラメータで示される)のスプレッドを返します。	時系列と指標へのアクセス
iAD	蓄積/配信(Accumulation/Distribution)。	テクニカル指標
iADX	平均方向性指数(Average Directional Index)。	テクニカル指標
iADXWilder	ウェルズ·ワイルダーの平均方向性指数(Average Directional Index by Welles Wilder)	テクニカル指標
iAlligator	アリゲーター。	テクニカル指標
iAMA	適応型移動平均(Adaptive Moving Average)。	テクニカル指標
iAO	オーサムオシレーター。	テクニカル指標
iATR	ATR(Average True Range)。	テクニカル指標
iBands	ボリンジャーバンド®(Bollinger Bands®)。	テクニカル指標
iBearsPower	ベアパワー(Bears Power)。	テクニカル指標
iBullsPower	ブルパワー(Bulls Power)。	テクニカル指標
iBWMFI	ビル·ウィリアムズのマーケットファシリテーションインデックス	テクニカル指標
iCCI	コモディティチャンネルインデックス(Commodity Channel Index)。	テクニカル指標
iChaikin	チャイキンオシレーター(Chaikin Oscillator)。	テクニカル指標
iCustom	カスタム指標。	テクニカル指標
iDEMA	2 重指数移動平均(Double Exponential Moving Average)。	テクニカル指標
iDeMarker	デマーカー(DeMarker)。	テクニカル指標
iEnvelopes	エンベロープ(Envelopes)。	テクニカル指標
iForce	勢力指数(Force Index)。	テクニカル指標
iFractals	フラクタル。	テクニカル指標
iFrAMA	フラクタル適応型移動平均(Fractal Adaptive Moving Average)。	テクニカル指標

© 2000–2025, MetaQuotes Ltd.

関数	アクション	セクション。
iGator	ゲーターオシレーター。	テクニカル指標
iIchimoku	一目均衡表（Ichimoku Kinko Hyo）。	テクニカル指標
iMA	移動平均（Moving Average）。	テクニカル指標
iMACD	移動平均収束拡散法（Moving Averages Convergence-Divergence）	テクニカル指標
iMFI	マネーフローインデックス（Money Flow Index）。	テクニカル指標
iMomentum	モメンタム（Momentum）。	テクニカル指標
IndicatorCreate	MqlParam 型パラメータの配列によって作成された指定されたテクニカル指標へのハンドルを返します。	時系列と指標へのアクセス
IndicatorParameters	指定されたハンドルに基づいて、指標の入力パラメータの数、値と型を返します。	時系列と指標へのアクセス
IndicatorRelease	指標ハンドルを削除し、誰によっても使用されていない場合は指標の計算ブロックを解放します。	時系列と指標へのアクセス
IndicatorSetDouble	double 型の指標プロパティの値を設定します。	カスタム指標
IndicatorSetInteger	int 型の指標プロパティの値を設定します。	カスタム指標
IndicatorSetString	string 型の指標プロパティの値を設定します。	カスタム指標
IntegerToString	int 型を事前に決めた長さの文字列に変換します。	変換関数
iOBV	オンバランスボリューム（On Balance Volume）。	テクニカル指標
iOsMA	移動平均オシレーター（Moving Average of Oscillator）（MACD ヒストグラム）。	テクニカル指標
iRSI	相対力指数（Relative Strength Index）。	テクニカル指標
iRVI	相対活力指数（Relative Vigor Index）。	テクニカル指標
iSAR	パラボリック停止・リバースシステム（Parabolic Stop And Reverse	テクニカル指標

© 2000–2025, MetaQuotes Ltd.

関数	アクション	セクション。
	System)	
IsStopped	MQL5 プログラムがその処理を停止するコマンドを受けた場合は true を返します。	チェックアップ
iStdDev	標準偏差(Standard Deviation)。	テクニカル指標
iStochastic	ストキャスティックス(Stochastic Oscillator)。	テクニカル指標
iTEMA	3 重指数移動平均(Triple Exponential Moving Average)。	テクニカル指標
iTriX	3 重指数移動平均オシレーター(Triple Exponential Moving Averages Oscillator)。	テクニカル指標
iVIDyA	可変インデックス動的平均(Variable Index Dynamic Average)。	テクニカル指標
iVolumes	ボリューム。	テクニカル指標
iWPR	ウィリアムパーセントレンジ(Williams' Percent Range)。	テクニカル指標
log	自然対数を返します。	数学関数
log10	数の底が10の対数を返します。	数学関数
MarketBookAdd	選択したシンボルの板情報のオープンを提供し、DOMの変更の通知を受信にサブスクライブします。	市場情報
MarketBookGet	指定されたシンボルの板情報を含む MqlBookInfo 構造体の配列を返します。	市場情報
MarketBookRelease	指定されたシンボルの板情報の閉鎖を提供し、DOMの変更の通知受信のサブスクライブをキャンセルします。	市場情報
MathAbs	指定された数値の絶対値(係数)を返します。	数学関数
MathArccos	ラジアンで x のアークコサインを返します。	数学関数
MathArcsin	ラジアンで x のアークサインを返します。	数学関数
MathArctan	ラジアンで x のアークタンジェントを返します。	数学関数

© 2000–2025, MetaQuotes Ltd.

関数	アクション	セクション。
MathCeil	切り上げされた整数値を返します。	数学関数
MathCos	数のコサインを返します。	数学関数
MathExp	数の指数を返します。	数学関数
MathFloor	切り下げされた整数値を返します。	数学関数
MathIsValidNumber	実数の有効性をチェックします。	数学関数
MathLog	自然対数を返します。	数学関数
MathLog10	数の底が10の対数を返します。	数学関数
MathMax	2つの数値の最大値を返します。	数学関数
MathMin	2つの数値の最小値を返します。	数学関数
MathMod	2つの数の分割後の実際の余りを返します。	数学関数
MathPow	基数を指定された乗まで引き上げます。	数学関数
MathRand	0～32,767の範囲内で擬似乱数値を返します。	数学関数
MathRound	値を最も近い整数に丸めます。	数学関数
MathSin	数のサインを返します。	数学関数
MathSqrt	平方根を返します。	数学関数
MathSrand	擬似ランダム整数の系列を生成するための開始点を設定します。	数学関数
MathTan	数のタンジェントを返します。	数学関数
MessageBox	メッセージボックスを作成、表示及び管理します。	共通関数
MQLInfoInteger	実行中のMQL5プログラムの対応するプロパティの整数値を返します。	チェックアップ
MQLInfoString	実行中のMQL5プログラムの対応するプロパティの文字列値を返します。	チェックアップ
MT5Initialize	MetaTrader 5ターミナルとの接続を確立します	Python用MetaTrader
MT5Shutdown	MetaTrader 5ターミナルへの以前に確立された接続を閉じます	Python用MetaTrader
MT5TerminalInfo	接続されているMetaTrader 5ターミナルのステータスとパラメータを取得	Python用MetaTrader

© 2000–2025, MetaQuotes Ltd.

関数	アクション	セクション。
	します	
MT5Version	MetaTrader 5ターミナルバージョンを返します	Python用MetaTrader
MT5CopyRatesFrom	指定された日以降のバーをMetaTrader 5ターミナルから取得します	Python用MetaTrader
MT5CopyRatesFromPos	指定されたインデックス以降のバーをMetaTrader 5ターミナルから取得します	Python用MetaTrader
MT5CopyRatesRange	指定された期間のバーをMetaTrader 5ターミナルから取得します	Python用MetaTrader
MT5CopyTicksFrom	指定された日以降のティックをMetaTrader 5ターミナルから取得します	Python用MetaTrader
MT5CopyTicksRange	指定された期間のティックをMetaTrader 5ターミナルから取得します	Python用MetaTrader
NormalizeDouble	浮動小数点数を指定された精度に丸めます。	変換関数
ObjectCreate	指定されたチャートで指定された型のオブジェクトを作成します。	オブジェクト関数
ObjectDelete	指定された名称のオブジェクトを指定されたチャートまたはチャートサブウィンドウから削除します。	オブジェクト関数
ObjectFind	名称に指定された ID を持つオブジェクトを探します。	オブジェクト関数
ObjectGetDouble	オブジェクトプロパティの double 値を返します。	オブジェクト関数
ObjectGetInteger	オブジェクトプロパティの整数値を返します。	オブジェクト関数
ObjectGetString	オブジェクトプロパティの string 値を返します。	オブジェクト関数
ObjectGetTimeByValue	指定されたオブジェクトの価格値の時間値を返します。	オブジェクト関数
ObjectGetValueByTime	指定された時刻でのオブジェクトの価格値を返します。	オブジェクト関数
ObjectMove	指定されたオブジェクトのアンカーポイントの座標を変更します。	オブジェクト関数

© 2000–2025, MetaQuotes Ltd.

関数	アクション	セクション。
ObjectName	指定されたチャートまたはチャートサブウィンドウの指定された名称のオブジェクトを返します。	オブジェクト関数
ObjectsDeleteAll	指定された型のオブジェクトを全て指定されたチャートまたはチャートサブウィンドウから削除します。	オブジェクト関数
ObjectSetDouble	オブジェクトプロパティの値を設定します。	オブジェクト関数
ObjectSetInteger	オブジェクトプロパティの値を設定します。	オブジェクト関数
ObjectSetString	オブジェクトプロパティの値を設定します。	オブジェクト関数
ObjectsTotal	指定されたチャートまたはチャートサブウィンドウのオブジェクトの数を返します。	オブジェクト関数
OnStart	この関数は、Startイベントがスクリプト内で設定されたアクションを実行するために発生したときに呼び出されます	イベント処理
OnInit	この関数は、Initイベントが発生して起動したMQL5プログラムが初期化されるときに、指標やEAで呼び出されます。	イベント処理
OnDeinit	この関数は、Deinitイベントが発生して起動したMQL5プログラムが初期化解除されるときに、指標やEAで呼び出されます。	イベント処理
OnTick	この関数は、NewTickイベントが発生して新しい気配値が処理されるときに、EAで呼び出されます。	イベント処理
OnCalculate	この関数は、Calculateイベントが発生して価格の変化が処理されるときに、指標で呼び出されます。	イベント処理
OnTimer	この関数は、一定の時間間隔でターミナルによって生成されるTimerイベント中に、指標やEAで呼び出されます。	イベント処理
OnTrade	この関数は、取引サーバ上の取引操作の終了時に生成されるTradeイベント中に、EAで呼び出されます。	イベント処理

© 2000–2025, MetaQuotes Ltd.

関数	アクション	セクション。
OnTradeTransaction	この関数は、TradeTransactionイベントが発生して取引リクエスト実行結果が処理されるときに、EAで呼び出されます。	イベント処理
OnBookEvent	この関数は、BookEventイベントが発生して板情報の変化が処理されるときに、EAで呼び出されます。	イベント処理
OnChartEvent	この関数は、ChartEventイベントが発生してユーザまたはMQL5プログラムによるチャートの変化が処理されるときに、指標やEAで呼び出されます。	イベント処理
OnTester	この関数は、Testerイベントが発生してEAをテストした後に履歴データに対して必要なアクションが実行されるときに、EAで呼び出されます。	イベント処理
OnTesterInit	この関数は、 TesterInitイベントが発生してストラテジーテスターでの最適化に先立って必要なアクションが実行されるときに、EAで呼び出されます。	イベント処理
OnTesterDeinit	この関数は、ストラテジーテスターでの最適化の後にTesterDeinit イベントが発生したときに、EAで呼び出されます。	イベント処理
OnTesterPass	この関数は、ストラテジーテスターでのEAの最適化の際にTesterPassイベントが発生して新しい時間枠の到着を処理するときに、EAで呼び出されます。	イベント処理
OrderCalcMargin	指定された注文の種類に必要な証拠金を預金通貨で計算します。	取引関数
OrderCalcProfit	渡されたパラメータに基づいて、利益を預金通貨で計算します。	取引関数
OrderCheck	取引操作を実行するための充分な資金があるかをチェックします。	取引関数
OrderGetDouble	リクエストされた注文プロパティ（double）を返します。	取引関数
OrderGetInteger	リクエストされた注文プロパティ（datetime または int）を返します。	取引関数

© 2000–2025, MetaQuotes Ltd.

関数	アクション	セクション。
OrderGetString	リクエストされた注文プロパティ（string）を返します。	取引関数
OrderGetTicket	注文のチケットを返します。	取引関数
OrderSelect	操作を続けるために注文を選択します。	取引関数
OrderSend	サーバに取引リクエストを送ります。	取引関数
OrderSendAsync	取引サーバの返答を待たずに取引リクエストを非同期的に送ります。	取引関数
OrdersTotal	注文数を返します。	取引関数
ParameterGetRange	ストラテジーテスターでエキスパートアドバイザーを最適化する際に、値範囲のデータを受け取り入力変数のステップを変更します。	最適化の結果の操作
ParameterSetRange	ストラテジーテスターでエキスパートアドバイザーを最適化する際に、入力変数（値、チェンジステップ、最初と最終の値）の使い方を指定します。	最適化の結果の操作
Period	現在のチャートの時間軸を返します。	チェックアップ
PeriodSeconds	期間中の秒数を返します。	共通関数
PlaySound	サウンドファイルを再生します。	共通関数
PlotIndexGetInteger	integer 型の指標線プロパティの値を返します。	カスタム指標
PlotIndexSetDouble	double 型の指標線プロパティの値を設定します。	カスタム指標
PlotIndexSetInteger	int 型の指標線プロパティの値を設定します。	カスタム指標
PlotIndexSetString	string 型の指標線プロパティの値を設定します。	カスタム指標
Point	現在の通貨/シンボルのポイントサイズを返します。	チェックアップ
PositionGetDouble	リクエストされたポジションプロパティ（double）を返します。	取引関数
PositionGetInteger	リクエストされたポジションプロパティ（datetime または int）を返します。	取引関数
PositionGetString	リクエストされたポジションプロパティ（string）を返します。	取引関数

© 2000–2025, MetaQuotes Ltd.

関数	アクション	セクション。
PositionGetSymbol	ポジションに対応するシンボルを返します。	取引関数
PositionSelect	操作を続けるためにポジションを選択します。	取引関数
PositionsTotal	ポジションの数を返します。	取引関数
pow	基数を指定された乗まで引き上げます。	数学関数
Print	ログにメッセージを表示します。	共通関数
PrintFormat	シンボルと値のセットを事前定義された形式に従ってフォーマットし、ログファイルにプリントします。	共通関数
rand	0～32,767　の範囲内で擬似乱数値を返します。	数学関数
ResetLastError	所定の変数 _LastError の値をゼロに設定します。	共通関数
ResourceCreate	データセットに基づいて画像のリソースを作成します。	共通関数
ResourceFree	動的に作成されたリソースを削除します（割り当てられたメモリを解放します）。	共通関数
ResourceReadImage	ResourceCreate() 関数で作成されたまたはコンパイル中に EX5 ファイルに保存されたグラフィックリソースからデータを読みます。	共通関数
ResourceSave	リソースを指定されたファイルに保存します。	共通関数
round	値を最も近い整数に丸めます。	数学関数
SendFTP	「FTP」タブの設定ウィンドウで指定されたアドレスにファイルを送信します。	共通関数
SendMail	「メール」タブの設定ウィンドウで指定されたアドレスに電子メールを送信します。	共通関数
SendNotification	MetaQuotes ID が「通知」タブで指定されているモバイル端末にプッシュ通知を送信します。	共通関数
SeriesInfoInteger	履歴データの状態情報を返します。	時系列と指標へのアクセス

© 2000–2025, MetaQuotes Ltd.

関数	アクション	セクション。
SetIndexBuffer	指定された指標バッファをdouble型の1次元の動的配列と関係づけます。	カスタム指標
ShortArrayToString	配列の一部を文字列に複製します。	変換関数
ShortToString	Unicodeシンボルコードを一シンボルの文字列に変換します。	変換関数
SignalBaseGetDouble	選択されたシグナルのdouble型プロパティの値を返します。	売買シグナル
SignalBaseGetInteger	選択されたシグナルの整数型プロパティの値を返します。	売買シグナル
SignalBaseGetString	選択されたシグナルのstring型プロパティの値を返します。	売買シグナル
SignalBaseSelect	端末で利用可能なシグナルから操作を続けるためにシグナルを選択します。	売買シグナル
SignalBaseTotal	端末で使用可能なシグナルの合計量を返します。	売買シグナル
SignalInfoGetDouble	シグナルコピー設定のdouble型プロパティの値を返します。	売買シグナル
SignalInfoGetInteger	シグナルコピー設定の整数型プロパティの値を返します。	売買シグナル
SignalInfoGetString	シグナルコピー設定のstring型プロパティの値を返します。	売買シグナル
SignalInfoSetDouble	シグナルコピー設定のdouble型プロパティの値を返します。	売買シグナル
SignalInfoSetInteger	シグナルコピー設定の整数型プロパティの値を設定します。	売買シグナル
SignalSubscribe	売買シグナルにサブスクライブします。	売買シグナル
SignalUnsubscribe	サブスクライブをキャンセルします。	売買シグナル
sin	数のサインを返します。	数学関数
Sleep	指定された間隔内で現在のエキスパートアドバイザーまたはスクリプトの実行を一時停止します。	共通関数
SocketCreate	指定されたフラグを使ってソケットを作成し、そのハンドルを返す	ネットワーク関数
SocketClose	ソケットを閉じる	ネットワーク関数

© 2000-2025, MetaQuotes Ltd.

関数	アクション	セクション。
SocketConnect	タイムアウト管理付きでサーバに接続する	ネットワーク関数
SocketIsConnected	ソケットが現在接続されているかどうかを確認します	ネットワーク関数
SocketIsReadable	ソケットから読み込めるバイト数を取得する	ネットワーク関数
SocketIsWritable	現在の時点でデータをソケットに書き込めるかどうかを確認する	ネットワーク関数
SocketTimeouts	ソケットシステムオブジェクトのデータを送受信するためのタイムアウトを設定する	ネットワーク関数
SocketRead	ソケットからデータを読む	ネットワーク関数
SocketSend	ソケットにデータを書く	ネットワーク関数
SocketTlsHandshake	TLS Handshakeプロトコルを介して指定されたホストへの安全なTLS（SSL）接続を開始する	ネットワーク関数
SocketTlsCertificate	ネットワーク接続を保護するために使用される証明書に関するデータを取得する	ネットワーク関数
SocketTlsRead	安全なTLS接続からデータを読み込む	ネットワーク関数
SocketTlsReadAvailable	安全なTLS接続からすべてのデータを読み込む	ネットワーク関数
SocketTlsSend	安全なTLS接続を介してデータを送信する	ネットワーク関数
sqrt	平方根を返します。	数学関数
srand	擬似ランダム整数の系列を生成するための開始点を設定します。	数学関数
StringAdd	文字列の終わりに別の文字列を追加します。	文字列関数
StringBufferLen	文字列に割り当てられたバッファのサイズを返します。	文字列関数
StringCompare	2つの文字列を比較し、1番目の文字列が2番目より大きい場合は1、両方が等しい場合は0、1番目が2番目より小さい場合は-1を返します。	文字列関数

© 2000–2025, MetaQuotes Ltd.

関数	アクション	セクション。
StringConcatenate	渡されたパラメータの文字列を形成します。	文字列関数
StringFill	指定された文字列に選択されたシンボルを書き入れます。	文字列関数
StringFind	文字列内で部分文字列を検索します。	文字列関数
StringFormat	数を事前に決めたフォーマットの文字列に変換します。	変換関数
StringGetCharacter	指定された文字列の位置での数の値を返します。	文字列関数
StringInit	指定された記号で文字列を初期化し、指定された文字列の長さを提供します。	文字列関数
StringLen	文字列のシンボルの数を返します。	文字列関数
StringReplace	文字列で見つかった全ての部分文字列を指定されたシンボルの配列で置換します。	文字列関数
StringSetCharacter	指定された位置にシンボルの変更値を持つ文字列のコピーを返します。	文字列関数
StringSplit	指定された文字列から指定されたセパレータで部分文字列を取得し、取得された部分文字列の数を返します。	文字列関数
StringSubstr	指定された位置から開始してテキスト文字列から部分文字列を抽出します。	文字列関数
StringToCharArray	Unicode から ANSI, に変換された文字列をシンボルごとに uchar 型の配列の一部に複製します。	変換関数
StringToColor	「R, G, B」の文字列またはカラー名を含む文字列を color 型の値に変換します。	変換関数
StringToDouble	数のシンボル表現を含む文字列を double 型の数値に変換します。	変換関数
StringToInteger	数のシンボル表現を含む文字列を int 型の数値に変換します。	変換関数
StringToLower	選択された文字列の全てのシンボルを小文字に変換します。	文字列関数

© 2000–2025, MetaQuotes Ltd.

関数	アクション	セクション。
StringToShortArray	文字列をシンボルごとに ushort 型の配列の一部に複製します。	変換関数
StringToTime	時間や日付を「yyyy.mm.dd [hh:mi]」フォーマットで含む文字列を datetime 型に変換します。	変換関数
StringToUpper	選択された文字列の全てのシンボルを大文字に変換します。	文字列関数
StringTrimLeft	文字列の左部分の改行文字、スペースやタブを切り取ります。	文字列関数
StringTrimRight	文字列の右部分の改行文字、スペースやタブを切り取ります。	文字列関数
StructToTime	MqlDateTime 構造体型の変数を datetime 値に変換します。	日付と時刻
Symbol	現在のチャートの銘柄名を返します。	チェックアップ
SymbolInfoDouble	シンボルプロパティの double 値を返します。	市場情報
SymbolInfoInteger	指定されたシンボルの整数型（long、datetime、int または bool）プロパティの値を返します。	市場情報
SymbolInfoMarginRate	注文の種類と方向に応じて補償金率を返します。	市場情報
SymbolInfoSessionQuote	指定された銘柄と曜日での指定された相場セッションの開始と終了の時間を受け取ることができます。	市場情報
SymbolInfoSessionTrade	指定された銘柄と曜日での指定された相場セッションの開始と終了の時間を受け取ることができます。	市場情報
SymbolInfoString	対応するプロパティに指定されたシンボルの文字列型の値を返します。	市場情報
SymbolInfoTick	指定されたシンボルの現在の価格をMqlTick 型の変数として返します。	市場情報
SymbolIsSynchronized	端末で選択されたシンボルのデータがレードサーバ―上のデータと同期されているかどうかをチェックします。	市場情報
SymbolName	指定されたシンボルの名称を返します。	市場情報

© 2000–2025, MetaQuotes Ltd.

関数	アクション	セクション。
SymbolSelect	「気配値表示」ウィンドウでシンボルを選択したり、ウィンドウからシンボルを削除したりします。	市場情報
SymbolsTotal	(「気配値表示」で選択されたまたは全ての)利用可能なシンボルの数を返します。	市場情報
tan	数のタンジェントを返します。	数学関数
TerminalClose	端末に操作完了を命令します。	共通関数
TerminalInfoDouble	MQL5 プログラム環境の対応するプロパティの double 値を返します。	チェックアップ
TerminalInfoInteger	MQL5 プログラム環境の対応するプロパティの整数値を返します。	チェックアップ
TerminalInfoString	MQL5 プログラム環境の対応するプロパティの文字列値を返します。	チェックアップ
TesterStatistics	テストの結果に基づいて算出された指定の統計の値を返します。	共通関数
TextGetSize	現在のフォント設定での文字列の幅と高さを返します。	オブジェクト関数
TextOut	テキストをグラフィックリソース作成に意図されたカスタム配列(バッファ)に転送する。	オブジェクト関数
TextSetFont	描画メソッドを使用してテキストを表示するためのフォントを設定します。デフォルトとしてArial20 が使用されます。	オブジェクト関数
TimeCurrent	最後の既知のサーバ時間(相場受信の最終時刻)を datetime フォーマットで返します。	日付と時刻
TimeDaylightSavings	夏時間のスイッチの符号を返します。	日付と時刻
TimeGMT	クライアント端末が実行しているコンピュータのローカル時間をGMTを夏時間を考慮した datetime フォーマットで返します。	日付と時刻
TimeGMTOffset	DST スイッチ を考慮して、GMT時間とローカルコンピュータの現在時刻との差を秒で返します。	日付と時刻
TimeLocal	ローカルコンピューター時刻を datetime フォーマットで返します。	日付と時刻

© 2000–2025, MetaQuotes Ltd.

関数	アクション	セクション。
TimeToString	1970年1月1日からの経過秒を含む値を「yyyy.mm.dd hh:mi 」形式の文字列に変換します。	変換関数
TimeToStruct	datetime 値を MqlDateTime 構造体型の変数に変換します。	日付と時刻
TimeTradeServer	取引サーバの現在の計算時間を返します。	日付と時刻
UninitializeReason	初期化解除理由のコードを返します。	チェックアップ
WebRequest	指定されたサーバにHTTP要求を送信します。	共通関数
ZeroMemory	参照によって渡された変数をリセットします。変数は、クラスとコンストラクタを含む構造体を除いては任意の型のものとすることが出来ます。	共通関数

© 2000-2025, MetaQuotes Ltd.

MQL5定数の表

アルファベット順の MQL5 関数リスト

定数	説明	使用例
__DATE__	時刻を含まないファイルコンパイルの日付（時間、分、秒は0）。	Print
__DATETIME__	ファイルコンパイルの日付と時刻。	Print
__FILE__	現在コンパイルされているファイルの名称。	Print
__FUNCSIG__	本体にマクロが配置されている関数のシグネチャ。関数の完全な記述のロギ	Print

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	ングはオーバーロード関数の識別に役立つことがあります。	
__FUNCTION__	本体にマクロが配置されている関数の名称。	Print
__LINE__	マクロが配置されているソースコードの行番号。	Print
__MQLBUILD__, __MQL5BUILD__	コンパイラのビルド番号。	Print
__PATH__	現在コンパイルされているファイルへの絶対パス。	Print
ACCOUNT_ASSETS	口座の流	AccountInfoDouble

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	動資産。	
ACCOUNT_BALANCE	預金通貨での勘定残高。	AccountInfoDouble
ACCOUNT_COMMISSION_BLOCKED	口座の現在のブロックされた手数料の額。	AccountInfoDouble
ACCOUNT_COMPANY	口座を提供する会社の名称。	AccountInfoString
ACCOUNT_CREDIT	預金通貨での口座のクレジット。	AccountInfoDouble
ACCOUNT_CURRENCY	口座通貨。	AccountInfoString
ACCOUNT_EQUITY	預金通貨の口座株式。	AccountInfoDouble
ACCOUNT_LEVERAGE	口座レバレッジ。	AccountInfoInteger
ACCOUNT_LIABILITIES	口座の流	AccountInfoDouble

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	動負債。	
ACCOUNT_LIMIT_ORDERS	アクティブな未決注文の最大許容数。	AccountInfoInteger
ACCOUNT_LOGIN	口座番号。	AccountInfoInteger
ACCOUNT_MARGIN	預金通貨での使用された口座の証拠金。	AccountInfoDouble
ACCOUNT_MARGIN_FREE	預金通貨での口座の余剰証拠金。	AccountInfoDouble
ACCOUNT_MARGIN_INITIAL	当初証拠金。全ての未決注文の証拠金をカバーするために口座内でリザーブされた額。	AccountInfoDouble

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
ACCOUNT_MARGIN_LEVEL	百分率で表した口座証拠金レベル。	AccountInfoDouble
ACCOUNT_MARGIN_MAINTENANCE	維持証拠金。全ての未決済ポジションの最小額をカバーするために口座内でリザーブされた最低資本金。	AccountInfoDouble
ACCOUNT_MARGIN_SO_CALL	証拠金コールレベル。ACCOUNT_MARGIN_SO_MODE の設定によってパーセントまたは預金通貨で表されます。	AccountInfoDouble

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
ACCOUNT_MARGIN_SO_MODE	許容された最小証拠金を設定するモード。	AccountInfoInteger
ACCOUNT_MARGIN_SO_SO	証拠金ストップアウトレベル。ACCOUNT_MARGIN_SO_MODE の設定によってパーセントまたは預金通貨で表されます。	AccountInfoDouble
ACCOUNT_NAME	クライアント名。	AccountInfoString
ACCOUNT_PROFIT	預金通貨での現在の利益。	AccountInfoDouble
ACCOUNT_SERVER	取引サーバ名。	AccountInfoString
ACCOUNT_STOPOUT_MODE_MONEY	口座ストップアウトの	AccountInfoInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	通貨モード。	
ACCOUNT_STOPOUT_MODE_PERCENT	口座ストップアウトの百分率モード。	AccountInfoInteger
ACCOUNT_TRADE_ALLOWED	現在の口座で許可された取引。	AccountInfoInteger
ACCOUNT_TRADE_EXPERT	エキスパートアドバイザーで許可された取引。	AccountInfoInteger
ACCOUNT_TRADE_MODE	口座取引モード。	AccountInfoInteger
ACCOUNT_TRADE_MODE_CONTEST	コンテスト口座。	AccountInfoInteger
ACCOUNT_TRADE_MODE_DEMO	デモ口座。	AccountInfoInteger
ACCOUNT_TRADE_MODE_REAL	リアル口座。	AccountInfoInteger
ALIGN_CENTER	中央揃え（編集オブジェク	ObjectSetInteger, ObjectGetInteger, ChartScreenShot

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	トのみ）。	
ALIGN_LEFT	左揃え。	ObjectSetInteger, ObjectGetInteger, ChartScreenShot
ALIGN_RIGHT	右揃え。	ObjectSetInteger, ObjectGetInteger, ChartScreenShot
ANCHOR_CENTER	厳密にオブジェクトの中心にあるアンカーポイント。	ObjectSetInteger、ObjectGetInteger
ANCHOR_LEFT	中心の左側のアンカーポイント。	ObjectSetInteger、ObjectGetInteger
ANCHOR_LEFT_LOWER	左下の隅のアンカーポイント。	ObjectSetInteger、ObjectGetInteger
ANCHOR_LEFT_UPPER	左上の隅のアンカーポイント。	ObjectSetInteger、ObjectGetInteger
ANCHOR_LOWER	中心の下部のアンカーポイント。	ObjectSetInteger、ObjectGetInteger
ANCHOR_RIGHT	中心の右側のアンカ	ObjectSetInteger、ObjectGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	ーポイント。	
ANCHOR_RIGHT_LOWER	右下の隅のアンカーポイント。	ObjectSetInteger、ObjectGetInteger
ANCHOR_RIGHT_UPPER	右上の隅のアンカーポイント。	ObjectSetInteger、ObjectGetInteger
ANCHOR_UPPER	中心の上のアンカーポイント。	ObjectSetInteger、ObjectGetInteger
BASE_LINE	正中線。	指標線
BOOK_TYPE_BUY	買い注文（ビッド）。	MqlBookInfo
BOOK_TYPE_BUY_MARKET	買いの成行注文。	MqlBookInfo
BOOK_TYPE_SELL	売り注文（オファー）。	MqlBookInfo
BOOK_TYPE_SELL_MARKET	売りの成行注文。	MqlBookInfo
BORDER_FLAT	フラットフォーム。	ObjectSetInteger、ObjectGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
BORDER_RAISED	プロミネントフォーム。	ObjectSetInteger、ObjectGetInteger
BORDER_SUNKEN	凹フォーム。	ObjectSetInteger、ObjectGetInteger
CHAR_MAX	char型で表すことが出来る最大値。。	数値定数
CHAR_MIN	char型で表すことが出来る最小値。。	数値定数
CHART_AUTOSCROLL	チャートの右側の境界線への自動移動のモード。	ChartSetInteger, ChartGetInteger
CHART_BARS	バーの列として表示。	ChartSetInteger
CHART_BEGIN	チャート先頭（最古の価格）。	ChartNavigate

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
CHART_BRING_TO_TOP	他のチャートの上に重ねてチャートを表示します。	ChartSetInteger, ChartGetInteger
CHART_CANDLES	日本製ローソク足として表示。	ChartSetInteger
CHART_COLOR_ASK	買値レベルの色。	ChartSetInteger, ChartGetInteger
CHART_COLOR_BACKGROUND	チャートの背景色。	ChartSetInteger, ChartGetInteger
CHART_COLOR_BID	売値レベルの色。	ChartSetInteger, ChartGetInteger
CHART_COLOR_CANDLE_BEAR	弱気ローソク足の実体の色。	ChartSetInteger, ChartGetInteger
CHART_COLOR_CANDLE_BULL	強気ローソク足の実体の色。	ChartSetInteger, ChartGetInteger
CHART_COLOR_CHART_DOWN	弱気ローソク足のダウンバー、影とボーダーの色。	ChartSetInteger, ChartGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
CHART_COLOR_CHART_LINE	折れ線グラフの色と「同時線」日本製ローソク足の色。	ChartSetInteger, ChartGetInteger
CHART_COLOR_CHART_UP	強気ローソク足のアップバー、影とボーダーの色。	ChartSetInteger, ChartGetInteger
CHART_COLOR_FOREGROUND	軸、スケール及びOHLCラインの色。	ChartSetInteger, ChartGetInteger
CHART_COLOR_GRID	グリッドの色。	ChartSetInteger, ChartGetInteger
CHART_COLOR_LAST	直近の約定価格（Last）の線の色。	ChartSetInteger, ChartGetInteger
CHART_COLOR_STOP_LEVEL	逆指注文値（決済逆指値及び決済指値）レ	ChartSetInteger, ChartGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	ベルの色。	
CHART_COLOR_VOLUME	ボリュームやポジションオープンレベルの色。	ChartSetInteger, ChartGetInteger
CHART_COMMENT	チャート内のコメントテキスト。	ChartSetString、ChartGetString
CHART_CURRENT_POS	現在の位置。	ChartNavigate
CHART_DRAG_TRADE_LEVELS	チャートで取引レベルをマウスでドラッグする許可。ドラッグモードの初期値は有効（true）です。	ChartSetInteger, ChartGetInteger
CHART_EVENT_MOUSE_MOVE	チャート上の全てのMQL5プログラムに、マ	ChartSetInteger, ChartGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	ウス移動イベントとマウスクリックイベントの通知を送信します（CHARTEVENT_MOUSE_MOVE）。	
CHART_EVENT_OBJECT_CREATE	チャート上の全てのMQL5プログラムに、新規オブジェクト作成イベントの通知を送信します（CHARTEVENT_OBJECT_CREATE）。	ChartSetInteger, ChartGetInteger
CHART_EVENT_OBJECT_DELETE	チャート上の全てのMQL5	ChartSetInteger, ChartGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	プログラムに、削除イベントの通知を送信します（CHARTEVENT_OBJECT_DELETE）。	
CHART_FIRST_VISIBLE_BAR	チャート上で最初に見えるバーの番号。バーのインデックスは時系列式に付けけられます。	ChartSetInteger, ChartGetInteger
CHART_FIXED_MAX	固定されたチャートの最大値。	ChartSetDouble, ChartGetDouble
CHART_FIXED_MIN	固定されたチャートの最小値。	ChartSetDouble, ChartGetDouble

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
CHART_FIXED_POSITION	パーセント値で表された、左の境界からのチャートの固定位置。チャートの固定位置は、水平時間軸上の小さな灰色の三角形でマークされています。これはティック着信時のチャート右への自動スクロールが無効になっている場合にのみ表示されます（CHA	ChartSetDouble, ChartGetDouble

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	RT_AUTOSCROLL プロパティを参照）。固定位置にあるバーはズームインやズームアウトする時にも移動しません。	
CHART_FOREGROUND	前景の価格チャート。	ChartSetInteger, ChartGetInteger
CHART_HEIGHT_IN_PIXELS	チャートの高さ（ピクセル単位）。	ChartSetInteger, ChartGetInteger
CHART_IS_OBJECT	チャートオブジェクト（OBJ_CHART）を識別し、グラフィックオブジェクトにはtrueを返し	ChartSetInteger, ChartGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	ます。チャートの場合falseを返します。	
CHART_LINE	終値に引かれた線として表示。	ChartSetInteger
CHART_MODE	チャートの種類（ローソク足、バー、またはライン）。	ChartSetInteger, ChartGetInteger
CHART_MOUSE_SCROLL	マウスの左ボタンを使用してチャートを水平方向にスクロールします。CHART_SCALEFIX、CHART_SCALEFIX_11またはCHART_SC	ChartSetInteger, ChartGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	ALE_PT_PER_BAR プロパティーの値が true の場合には、垂直方向のスクロールも可能です。	
CHART_POINTS_PER_BAR	スケール（バーあたりのポイント単位）。	ChartSetDouble, ChartGetDouble
CHART_PRICE_MAX	チャートの最大値。	ChartSetDouble, ChartGetDouble
CHART_PRICE_MIN	チャートの最小値。	ChartSetDouble, ChartGetDouble
CHART_SCALE	Scale	ChartSetInteger, ChartGetInteger
CHART_SCALE_PT_PER_BAR	指定されたスケール（バーあたりのポイント単位）。	ChartSetInteger, ChartGetInteger
CHART_SCALEFIX	固定スケー	ChartSetInteger, ChartGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	ルモード。	
CHART_SCALEFIX_11	1:1 スケールモード。	ChartSetInteger, ChartGetInteger
CHART_SHIFT	右側の境界線からの価格チャートのインデントのモード。	ChartSetInteger, ChartGetInteger
CHART_SHIFT_SIZE	百分率で表された右境界線からのゼロバーインデントのサイズ。	ChartSetDouble, ChartGetDouble
CHART_SHOW_ASK_LINE	売気配値をチャートに水平線として表示。	ChartSetInteger, ChartGetInteger
CHART_SHOW_BID_LINE	買気配値をチャートに水平線として表示。	ChartSetInteger, ChartGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
CHART_SHOW_DATE_SCALE	チャートでタイムスケールを表示。	ChartSetInteger, ChartGetInteger
CHART_SHOW_GRID	チャートにグリッドを表示。	ChartSetInteger, ChartGetInteger
CHART_SHOW_LAST_LINE	直近価格をチャートに水平線として表示。	ChartSetInteger, ChartGetInteger
CHART_SHOW_OBJECT_DESCR	グラフィカルオブジェクトのポップアップ説明。	ChartSetInteger, ChartGetInteger
CHART_SHOW_OHLC	左上隅にOHLC値を表示。	ChartSetInteger, ChartGetInteger
CHART_SHOW_ONE_CLICK	チャートに「ワンクリック取引」パネルを表示。	ChartSetInteger, ChartGetInteger
CHART_SHOW_PERIOD_SEP	隣接する	ChartSetInteger, ChartGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	期間の間に垂直方向のセパレータを表示。	
CHART_SHOW_PRICE_SCALE	チャートに価格スケールを表示。	ChartSetInteger, ChartGetInteger
CHART_SHOW_TRADE_LEVELS	チャートでの取引レベル（ポジション、決済逆指値、決済指値及び未決注文のレベル）を表示。	ChartSetInteger, ChartGetInteger
CHART_SHOW_VOLUMES	チャートにボリュームを表示。	ChartSetInteger, ChartGetInteger
CHART_VISIBLE_BARS	チャートに表示出来	ChartSetInteger, ChartGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	るバーの数。	
CHART_VOLUME_HIDE	ボリュームの非表示。	ChartSetInteger
CHART_VOLUME_REAL	取引高。	ChartSetInteger
CHART_VOLUME_TICK	ティックボリューム。	ChartSetInteger
CHART_WIDTH_IN_BARS	チャートの幅（バー単位）。	ChartSetInteger, ChartGetInteger
CHART_WIDTH_IN_PIXELS	チャートの幅（ピクセル単位）。	ChartSetInteger, ChartGetInteger
CHART_WINDOW_HANDLE	チャートウィンドウハンドル（HWND）。	ChartSetInteger, ChartGetInteger
CHART_WINDOW_IS_VISIBLE	サブウィンドウの可視性。	ChartSetInteger, ChartGetInteger
CHART_WINDOW_YDISTANCE	指標サブウィンドウの上部フレームとメイン	ChartSetInteger, ChartGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	チャートウィンドウの上部フレームとの縦Y軸に沿った距離（ピクセル単位）。指標サブウィンドウでのグラフィカルオブジェクトの座標がサブウィンドウの左上隅を基準に設定されているのに比べ、マウスイベントでのカーソル座標は、メインチャートウィンドウの座標で渡さ	

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	れます。この値は、図形オブジェクトを正しく取り扱うために、メインチャートの絶対座標をサブウィンドウフレームの左上隅を基準に設定されているサブウィンドウのローカル座標に変換するために必要とされます。	
CHART_WINDOWS_TOTAL	指標サブウィンドウを含んだチャートウィン	ChartSetInteger, ChartGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	ドウの総数。	
CHARTEVENT_CHART_CHANGE	プロパティダイアログでのチャートサイズやチャートプロパティの変更。	OnChartEvent
CHARTEVENT_CLICK	チャートのクリック。	OnChartEvent
CHARTEVENT_CUSTOM	カスタムイベントの範囲からのイベントの初期番号。	OnChartEvent
CHARTEVENT_CUSTOM_LAST	カスタムイベントの範囲からのイベントの最終番号。	OnChartEvent
CHARTEVENT_KEYDOWN	キーストローク。	OnChartEvent
CHARTEVENT_MOUSE_MOVE	マウスの移動、マ	OnChartEvent

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	ウスクリック（チャートでCHART_EVENT_MOUSE_MOVE=true の場合）。	
CHARTEVENT_OBJECT_CHANGE	プロパティダイアログで変更されたグラフィックオブジェクトのプロパティ。	OnChartEvent
CHARTEVENT_OBJECT_CLICK	グラフィックオブジェクトのクリック。	OnChartEvent
CHARTEVENT_OBJECT_CREATE	作成されたグラフィックオブジェクト（チャートでCHART_EVENT_OBJECT_CRE	OnChartEvent

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	ATE=true の場合）。	
CHARTEVENT_OBJECT_DELETE	削除されたグラフィックオブジェクト（チャートでCHART_EVENT_OBJECT_DELETE=true の場合）。	OnChartEvent
CHARTEVENT_OBJECT_DRAG	グラフィックオブジェクトのドラッグアンドドロップ。	OnChartEvent
CHARTEVENT_OBJECT_ENDEDIT	Edit グラフィックオブジェクトのテキスト編集の終了。	OnChartEvent
CHARTS_MAX	端末内に同時に開ける	その他の定数

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	チャートの最大数。	
CHIKOUSPAN_LINE	遅行スパンライン。	指標線
clrAliceBlue	アリスブルー	ウェブカラー
clrAntiqueWhite	アンティークホワイト	ウェブカラー
clrAqua	アクア	ウェブカラー
clrAquamarine	アクアマリン	ウェブカラー
clrBeige	ベージュ	ウェブカラー
clrBisque	ビスク	ウェブカラー
clrBlack	黒	ウェブカラー
clrBlanchedAlmond	湯通しアーモンド	ウェブカラー
clrBlue	青	ウェブカラー
clrBlueViolet	青紫	ウェブカラー
clrBrown	茶	ウェブカラー
clrBurlyWood	バーリウッド	ウェブカラー
clrCadetBlue	カデットブルー	ウェブカラー
clrChartreuse	シャルトリューズ	ウェブカラー
clrChocolate	チョコレート	ウェブカラー

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
clrCoral	コーラル	ウェブカラー
clrCornflowerBlue	コーンフラワーブルー	ウェブカラー
clrCornsilk	コーンシルク	ウェブカラー
clrCrimson	真紅	ウェブカラー
clrDarkBlue	ダークブルー	ウェブカラー
clrDarkGoldenrod	ダークゴールデンロッド	ウェブカラー
clrDarkGray	ダークグレー	ウェブカラー
clrDarkGreen	深緑	ウェブカラー
clrDarkKhaki	ダークカーキ	ウェブカラー
clrDarkOliveGreen	ダークオリーブグリーン	ウェブカラー
clrDarkOrange	ダークオレンジ	ウェブカラー
clrDarkOrchid	ダークオーキッド	ウェブカラー
clrDarkSalmon	ダークサーモン	ウェブカラー
clrDarkSeaGreen	ダークシーグリーン	ウェブカラー
clrDarkSlateBlue	ダークスレートブルー	ウェブカラー

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
clrDarkSlateGray	ダークスレートグレー	ウェブカラー
clrDarkTurquoise	ダークターコイズ	ウェブカラー
clrDarkViolet	ダークバイオレット	ウェブカラー
clrDeepPink	ディープピンク	ウェブカラー
clrDeepSkyBlue	ディープスカイブルー	ウェブカラー
clrDimGray	薄暗いグレー	ウェブカラー
clrDodgerBlue	ドジャーブルー	ウェブカラー
clrFireBrick	ファイアブリック	ウェブカラー
clrForestGreen	フォレストグリーン	ウェブカラー
clrGainsboro	ゲインスボロ	ウェブカラー
clrGold	金	ウェブカラー
clrGoldenrod	ゴールデンロッド	ウェブカラー
clrGray	灰	ウェブカラー
clrGreen	緑	ウェブカラー
clrGreenYellow	グリーンイエロー	ウェブカラー

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
clrHoneydew	ハニーデュー	ウェブカラー
clrHotPink	ホットピンク	ウェブカラー
clrIndianRed	インディアンレッド	ウェブカラー
clrIndigo	インディゴ	ウェブカラー
clrIvory	アイボリー	ウェブカラー
clrKhaki	カーキ	ウェブカラー
clrLavender	ラベンダー	ウェブカラー
clrLavenderBlush	ラベンダーブラッシュ	ウェブカラー
clrLawnGreen	ローングリーン	ウェブカラー
clrLemonChiffon	レモンシフォン	ウェブカラー
clrLightBlue	ライトブルー	ウェブカラー
clrLightCoral	ライトコーラル	ウェブカラー
clrLightCyan	ライトシアン	ウェブカラー
clrLightGoldenrod	ライトゴールデンロッド	ウェブカラー
clrLightGray	ライトグレイ	ウェブカラー

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
clrLightGreen	ライトグリーン	ウェブカラー
clrLightPink	ライトピンク	ウェブカラー
clrLightSalmon	ライトサーモン	ウェブカラー
clrLightSeaGreen	ライトシーグリーン	ウェブカラー
clrLightSkyBlue	ライトスカイブルー	ウェブカラー
clrLightSlateGray	ライトスレートグレー	ウェブカラー
clrLightSteelBlue	ライトスチールブルー	ウェブカラー
clrLightYellow	ライトイエロー	ウェブカラー
clrLime	ライム	ウェブカラー
clrLimeGreen	ライムグリーン	ウェブカラー
clrLinen	リネン	ウェブカラー
clrMagenta	マゼンタ	ウェブカラー
clrMaroon	マルーン	ウェブカラー
clrMediumAquamarine	ミディアムアクアマリン	ウェブカラー
clrMediumBlue	ミディアムブ	ウェブカラー

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	ルー	
clrMediumOrchid	ミディアムオーキッド	ウェブカラー
clrMediumPurple	ミディアムパープル	ウェブカラー
clrMediumSeaGreen	ミディアムシーグリーン	ウェブカラー
clrMediumSlateBlue	ミディアムスレートブルー	ウェブカラー
clrMediumSpringGreen	ミディアムスプリンググリーン	ウェブカラー
clrMediumTurquoise	ミディアムターコイズ	ウェブカラー
clrMediumVioletRed	ミディアムバイオレットレッド	ウェブカラー
clrMidnightBlue	ミッドナイトブルー	ウェブカラー
clrMintCream	ミントクリーム	ウェブカラー
clrMistyRose	ミスティローズ	ウェブカラー
clrMoccasin	モカシン	ウェブカラー

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
clrNavajoWhite	ナバホホワイト	ウェブカラー
clrNavy	ネービー	ウェブカラー
clrNONE	無色。	その他の定数
clrOldLace	オールドレース	ウェブカラー
clrOlive	オリーブ	ウェブカラー
clrOliveDrab	オリーブドラブ	ウェブカラー
clrOrange	橙	ウェブカラー
clrOrangeRed	オレンジレッド	ウェブカラー
clrOrchid	オーキッド	ウェブカラー
clrPaleGoldenrod	ペールゴールデンロッド	ウェブカラー
clrPaleGreen	ペールグリーン	ウェブカラー
clrPaleTurquoise	ペールターコイズ	ウェブカラー
clrPaleVioletRed	ペールバイオレットレッド	ウェブカラー
clrPapayaWhip	パパイアホイップ	ウェブカラー
clrPeachPuff	ピーチパフ	ウェブカラー

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
clrPeru	ペルー	ウェブカラー
clrPink	ピンク	ウェブカラー
clrPlum	プラム	ウェブカラー
clrPowderBlue	パウダーブルー	ウェブカラー
clrPurple	紫	ウェブカラー
clrRed	赤	ウェブカラー
clrRosyBrown	ロージーブラウン	ウェブカラー
clrRoyalBlue	ロイヤルブルー	ウェブカラー
clrSaddleBrown	サドルブラウン	ウェブカラー
clrSalmon	サーモン	ウェブカラー
clrSandyBrown	サンディブラウン	ウェブカラー
clrSeaGreen	シーグリーン	ウェブカラー
clrSeashell	シーシェル	ウェブカラー
clrSienna	シエナ	ウェブカラー
clrSilver	銀	ウェブカラー
clrSkyBlue	スカイブルー	ウェブカラー
clrSlateBlue	スレートブルー	ウェブカラー
clrSlateGray	スレートグレー	ウェブカラー
clrSnow	スノー	ウェブカラー

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
clrSpringGreen	スプリンググリーン	ウェブカラー
clrSteelBlue	スチールブルー	ウェブカラー
clrTan	タン	ウェブカラー
clrTeal	ティール	ウェブカラー
clrThistle	アザミ	ウェブカラー
clrTomato	トマト	ウェブカラー
clrTurquoise	ターコイズ	ウェブカラー
clrViolet	バイオレット	ウェブカラー
clrWheat	ウィート	ウェブカラー
clrWhite	白	ウェブカラー
clrWhiteSmoke	ホワイトスモーク	ウェブカラー
clrYellow	黄	ウェブカラー
clrYellowGreen	イエローグリーン	ウェブカラー
CORNER_LEFT_LOWER	座標の中心がチャートの左下隅。	ObjectSetInteger、ObjectGetInteger
CORNER_LEFT_UPPER	座標の中心がチャートの右上隅。	ObjectSetInteger、ObjectGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
CORNER_RIGHT_LOWER	座標の中心がチャートの右下隅。	ObjectSetInteger、ObjectGetInteger
CORNER_RIGHT_UPPER	座標の中心がチャートの右上隅。	ObjectSetInteger、ObjectGetInteger
CP_ACP	現在のWindows ANSIコードページ。	CharArrayToString、StringToCharArray、FileOpen
CP_MACCP	現在のシステムのMacintoshコードページ。注：現代のMacintoshコンピュータはエンコーディングにUnicodeを使用しているため、この値	CharArrayToString、StringToCharArray、FileOpen

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	は、主に以前に作成したプログラムコードで使用され、今は使用されていません。	
CP_OEMCP	現在のシステムのOEMコードページ。	CharArrayToString、StringToCharArray、FileOpen
CP_SYMBOL	シンボルコードページ。	CharArrayToString、StringToCharArray、FileOpen
CP_THREAD_ACP	現在のスレッドのWindows ANSIコードページ。	CharArrayToString、StringToCharArray、FileOpen
CP_UTF7	UTF-7コードページ。	CharArrayToString、StringToCharArray、FileOpen
CP_UTF8	UTF-8コードページ。	CharArrayToString、StringToCharArray、FileOpen

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
CRYPT_AES128	128ビット（16バイト）キーでのAES暗号化。	CryptEncode、CryptDecode
CRYPT_AES256	256ビット（32バイト）キーでのAES暗号化。	CryptEncode、CryptDecode
CRYPT_ARCH_ZIP	ZIP アーカイブ。	CryptEncode、CryptDecode
CRYPT_BASE64	BASE64	CryptEncode、CryptDecode
CRYPT_DES	56ビット（7バイト）キーでのDES暗号化。	CryptEncode、CryptDecode
CRYPT_HASH_MD5	MD5 HASHの計算。	CryptEncode、CryptDecode
CRYPT_HASH_SHA1	SHA1 HASHの計算。	CryptEncode、CryptDecode
CRYPT_HASH_SHA256	SHA256 HASHの計算。	CryptEncode、CryptDecode

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
DBL_DIG	double 型の小数点以下の桁数。	数値定数
DBL_EPSILON	下記の条件を満たす最小値。1.0+DBL_EPSILON != 1.0 (double 型)	数値定数
DBL_MANT_DIG	double 型の仮数部のビット数。	数値定数
DBL_MAX	double 型で表すことが出来る最大値。。	数値定数
DBL_MAX_10_EXP	double 型の指数度の 10 進数での最大値。	数値定数
DBL_MAX_EXP	double 型の指	数値定数

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	数度の2進数での最大値。	
DBL_MIN	double型で表すことが出来る最小の正の値。	数値定数
DBL_MIN_10_EXP	double型の指数度の10進数での最小値。	数値定数
DBL_MIN_EXP	double型の指数度の2進数での最小値。	数値定数
DEAL_COMMENT	約定コメント。	HistoryDealGetString
DEAL_COMMISSION	約定手数料。	HistoryDealGetDouble
DEAL_ENTRY	約定エントリー（エントリーイン、	HistoryDealGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	エントリーアウト、リバース)。	
DEAL_ENTRY_IN	エントリーイン。	HistoryDealGetInteger
DEAL_ENTRY_INOUT	リバース。	HistoryDealGetInteger
DEAL_ENTRY_OUT	エントリーアウト。	HistoryDealGetInteger
DEAL_MAGIC	約定マジックナンバー(ORDER_MAGICを参照)。	HistoryDealGetInteger
DEAL_ORDER	約定の注文番号。	HistoryDealGetInteger
DEAL_POSITION_ID	約定の開始、変更や変化が起きたポジションのポジション識別子。ポジションは、ライフタイムを通して、シ	HistoryDealGetInteger

© 2000-2025, MetaQuotes Ltd.

定数	説明	使用例
	ンボルに実行された全ての約定に割り当てられている固有の識別子を持ちます。	
DEAL_PRICE	約定値。	HistoryDealGetDouble
DEAL_PROFIT	約定の利益。	HistoryDealGetDouble
DEAL_SWAP	終了時の累積スワップ。	HistoryDealGetDouble
DEAL_SYMBOL	約定シンボル。	HistoryDealGetString
DEAL_TIME	約定時刻。	HistoryDealGetInteger
DEAL_TIME_MSC	01.01.1970から経過したミリ秒数で表された約定実行時刻。	HistoryDealGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
DEAL_TYPE	約定の種類。	HistoryDealGetInteger
DEAL_TYPE_BALANCE	残高。	HistoryDealGetInteger
DEAL_TYPE_BONUS	ボーナス。	HistoryDealGetInteger
DEAL_TYPE_BUY	買い。	HistoryDealGetInteger
DEAL_TYPE_BUY_CANCELED	取り消しされた買い約定。以前に実行された買い約定が取り消しされる場合があります。この場合には、以前に実行された約定（DEAL_TYPE_BUY）の種類はDEAL_TYPE_BUY_CANCELEDに変更さ	HistoryDealGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	れ、その利益/損失はゼロ化されます。先に取得された利益/損失は、別の残高操作で実行されます。	
DEAL_TYPE_CHARGE	追加料金。	HistoryDealGetInteger
DEAL_TYPE_COMMISSION	追加手数料。	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_AGENT_DAILY	日ごとの仲介手数料。	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_AGENT_MONTHLY	月ごとの仲介手数料。	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_DAILY	日ごとの手数料。	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_MONTHLY	月ごとの手数料。	HistoryDealGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
DEAL_TYPE_CORRECTION	修正。	HistoryDealGetInteger
DEAL_TYPE_CREDIT	クレジット。	HistoryDealGetInteger
DEAL_TYPE_INTEREST	金利。	HistoryDealGetInteger
DEAL_TYPE_SELL	売り。	HistoryDealGetInteger
DEAL_TYPE_SELL_CANCELED	取り消しされた売り約定。以前に実行された売り約定が取り消しされる場合があります。この場合には、以前に実行された約定（DEAL_TYPE_SELL）の種類はDEAL_TYPE_SELL_CANCELEDに変	HistoryDealGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	更され、その利益/損失はゼロ化されます。先に取得された利益/損失は、別の残高操作で実行されます。	
DEAL_VOLUME	約定量。	HistoryDealGetDouble
DRAW_ARROW	矢印の描画。	描画スタイル
DRAW_BARS	バーの列として表示。	描画スタイル
DRAW_CANDLES	ローソク足の列として表示。	描画スタイル
DRAW_COLOR_ARROW	多色の矢印を描画。	描画スタイル
DRAW_COLOR_BARS	多色バー。	描画スタイル

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
DRAW_COLOR_CANDLES	多色ローソク足。	描画スタイル
DRAW_COLOR_HISTOGRAM	ゼロラインからの多色ヒストグラム。	描画スタイル
DRAW_COLOR_HISTOGRAM2	2つの指標バッファの多色ヒストグラム。	描画スタイル
DRAW_COLOR_LINE	多色のライン。	描画スタイル
DRAW_COLOR_SECTION	多色のセクション。	描画スタイル
DRAW_COLOR_ZIGZAG	多色のジグザグ。	描画スタイル
DRAW_FILLING	2レベル間の色の塗りつぶし。	描画スタイル
DRAW_HISTOGRAM	ゼロラインからのヒストグラム。	描画スタイル
DRAW_HISTOGRAM2	2つの指標バッファのヒスト	描画スタイル

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	グラム。	
DRAW_LINE	線。	描画スタイル
DRAW_NONE	描かれません。	描画スタイル
DRAW_SECTION	セクション。	描画スタイル
DRAW_ZIGZAG	ジグザグスタイルは、バーの垂直断面を可能にします。	描画スタイル
ELLIOTT_CYCLE	サイクル。	ObjectSetInteger、ObjectGetInteger
ELLIOTT_GRAND_SUPERCYCLE	グランドスーパーサイクル。	ObjectSetInteger、ObjectGetInteger
ELLIOTT_INTERMEDIATE	インターメディエイト。	ObjectSetInteger、ObjectGetInteger
ELLIOTT_MINOR	マイナー。	ObjectSetInteger、ObjectGetInteger
ELLIOTT_MINUETTE	ミヌエット。	ObjectSetInteger、ObjectGetInteger
ELLIOTT_MINUTE	ミニュート。	ObjectSetInteger、ObjectGetInteger
ELLIOTT_PRIMARY	プライマリ。	ObjectSetInteger、ObjectGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
ELLIOTT_SUBMINUETTE	サブミヌエット。	ObjectSetInteger、ObjectGetInteger
ELLIOTT_SUPERCYCLE	スーパーサイクル。	ObjectSetInteger、ObjectGetInteger
EMPTY_VALUE	指標バファーの空値。	その他の定数
ERR_ACCOUNT_WRONG_PROPERTY	不正な口座プロパティ識別子。	GetLastError
ERR_ARRAY_BAD_SIZE	リクエストされた配列のサイズが2GBを超えます。	GetLastError
ERR_ARRAY_RESIZE_ERROR	配列の移転に不充分なメモリ、または静的配列のサイズを変更する試み。	GetLastError
ERR_BOOKS_CANNOT_ADD	板情報の追加	GetLastError

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	が不可能。	
ERR_BOOKS_CANNOT_DELETE	板情報の削除が不可能。	GetLastError
ERR_BOOKS_CANNOT_GET	板情報データ取得が不可能。	GetLastError
ERR_BOOKS_CANNOT_SUBSCRIBE	板情報から新規データを取得するのにサブスクライブ中にエラーが発生。	GetLastError
ERR_BUFFERS_NO_MEMORY	指標バッファ配布に不充分なメモリ。	GetLastError
ERR_BUFFERS_WRONG_INDEX	不正な指標バッファインデックス。	GetLastError
ERR_CANNOT_CLEAN_DIRECTORY	ディレクトリ	GetLastError

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	のクリアに失敗（おそらく1つ以上のファイルがブロックされ、除去操作が失敗）。	
ERR_CANNOT_DELETE_DIRECTORY	ディレクトリを削除に失敗。	GetLastError
ERR_CANNOT_DELETE_FILE	ファイル削除エラー。	GetLastError
ERR_CANNOT_OPEN_FILE	ファイルオープンエラー。	GetLastError
ERR_CHAR_ARRAY_ONLY	char型の配列のみ可。	GetLastError
ERR_CHART_CANNOT_CHANGE	銘柄と期間の変更に失敗。	GetLastError
ERR_CHART_CANNOT_CREATE_TIMER	タイマー作成に失敗。	GetLastError

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
ERR_CHART_CANNOT_OPEN	チャートオープンエラー。	GetLastError
ERR_CHART_INDICATOR_CANNOT_ADD	指標をチャートに追加するのに失敗。	GetLastError
ERR_CHART_INDICATOR_CANNOT_DEL	指標をチャートから削除するのに失敗。	GetLastError
ERR_CHART_INDICATOR_NOT_FOUND	指標が指定されたチャートに不在。	GetLastError
ERR_CHART_NAVIGATE_FAILED	チャートナビゲートエラー。	GetLastError
ERR_CHART_NO_EXPERT	イベントを処理出来るエキスパートアドバイザーがチャートに不在。	GetLastError

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
ERR_CHART_NO_REPLY	チャートが応答しません。	GetLastError
ERR_CHART_NOT_FOUND	チャートが見つかりません。	GetLastError
ERR_CHART_SCREENSHOT_FAILED	スクリーンショット作成エラー。	GetLastError
ERR_CHART_TEMPLATE_FAILED	テンプレート適用エラー。	GetLastError
ERR_CHART_WINDOW_NOT_FOUND	指標を含むサブウィンドウが不在。	GetLastError
ERR_CHART_WRONG_ID	不正なチャート識別子。	GetLastError
ERR_CHART_WRONG_PARAMETER	チャート操作の関数のパラメータ値のエラー。	GetLastError
ERR_CHART_WRONG_PROPERTY	不正なチャ	GetLastError

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	ートプロパティ識別子。	
ERR_CUSTOM_WRONG_PROPERTY	不正なカスタム指標プロパティ識別子。	GetLastError
ERR_DIRECTORY_NOT_EXIST	ディレクトリ不在。	GetLastError
ERR_DOUBLE_ARRAY_ONLY	double 型の配列のみ可。	GetLastError
ERR_FILE_BINSTRINGSIZE	ファイルがバイナリとしてオープンされたため、文字列のサイズ指定が必要。	GetLastError
ERR_FILE_CACHEBUFFER_ERROR	読み込みのためにキャッシュに出来るメモリが不足。	GetLastError

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
ERR_FILE_CANNOT_REWRITE	ファイルの書き換えが不可。	GetLastError
ERR_FILE_IS_DIRECTORY	これはファイルではなくディレクトリです。	GetLastError
ERR_FILE_ISNOT_DIRECTORY	これはディレクトリではなくファイルです。	GetLastError
ERR_FILE_NOT_EXIST	ファイルが不在。	GetLastError
ERR_FILE_NOTBIN	ファイルはバイナリとして開かれる必要があります。	GetLastError
ERR_FILE_NOTCSV	ファイルはCSVとして開かれる必要があります。	GetLastError
ERR_FILE_NOTTOREAD	ファイルは	GetLastError

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	読み込むために開かれる必要があります。	
ERR_FILE_NOTTOWRITE	ファイルは書き込むために開かれる必要があります。	GetLastError
ERR_FILE_NOTTXT	ファイルはテキストとして開かれる必要があります。	GetLastError
ERR_FILE_NOTTXTORCSV	ファイルはテキストまたはCSVとして開かれる必要があります。	GetLastError
ERR_FILE_READERROR	ファイル読み込みエラー。	GetLastError
ERR_FILE_WRITEERROR	ファイルへのリソー	GetLastError

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	スの書き込みに失敗。	
ERR_FLOAT_ARRAY_ONLY	float型の配列のみ可。	GetLastError
ERR_FTP_SEND_FAILED	FTPでのファイル送信に失敗。	GetLastError
ERR_FUNCTION_NOT_ALLOWED	関数呼び出しの許可がありません。	GetLastError
ERR_GLOBALVARIABLE_EXISTS	同名のクライアント端末のグローバル変数が既存。	GetLastError
ERR_GLOBALVARIABLE_NOT_FOUND	クライアント端末のグローバル変数が見つかりません。	GetLastError

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
ERR_HISTORY_NOT_FOUND	リクエストされた履歴が見つかりません。	GetLastError
ERR_HISTORY_WRONG_PROPERTY	不正な履歴プロパティ識別子。	GetLastError
ERR_INCOMPATIBLE_ARRAYS	互換性のない配列の複製。文字列配列は文字列配列、数値配列は数値配列のみに複製することが出来ます。	GetLastError
ERR_INCOMPATIBLE_FILE	文字列配列はテキストファイル、他の配列はバイナリ	GetLastError

© 2000-2025, MetaQuotes Ltd.

定数	説明	使用例
	ファイルでなければいけません。	
ERR_INDICATOR_CANNOT_ADD	指標をチャートに適用するのに失敗。	GetLastError
ERR_INDICATOR_CANNOT_APPLY	指標の他の指標への適用は不可。	GetLastError
ERR_INDICATOR_CANNOT_CREATE	指標作成が不可。	GetLastError
ERR_INDICATOR_CUSTOM_NAME	配列の最初のパラメータは、カスタム指標の名称でなければなりません。	GetLastError
ERR_INDICATOR_DATA_NOT_FOUND	リクエストされたデータが見つかりません。	GetLastError

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
ERR_INDICATOR_NO_MEMORY	指標追加に不充分なメモリ。	GetLastError
ERR_INDICATOR_PARAMETER_TYPE	指標作成時の配列内の無効なパラメータ型。	GetLastError
ERR_INDICATOR_PARAMETERS_MISSING	指標作成時にパラメータ型が不在。	GetLastError
ERR_INDICATOR_UNKNOWN_SYMBOL	未知のシンボル。	GetLastError
ERR_INDICATOR_WRONG_HANDLE	不正な指標ハンドル。	GetLastError
ERR_INDICATOR_WRONG_INDEX	リクエストされた指標バッファの不正なインデックス。	GetLastError
ERR_INDICATOR_WRONG_PARAMETERS	指標作成時の不正なパラ	GetLastError

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	メータ数。	
ERR_INT_ARRAY_ONLY	int 型の配列のみ可。	GetLastError
ERR_INTERNAL_ERROR	予期しない内部エラー。	GetLastError
ERR_INVALID_ARRAY	型やサイズが不正な配列、または動的配列内に損傷を受けたオブジェクト。	GetLastError
ERR_INVALID_DATETIME	無効な日付/時刻。	GetLastError
ERR_INVALID_FILEHANDLE	このハンドルを使用したファイルは閉じられた、または、初めから開けられません	GetLastError

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	でした。	
ERR_INVALID_PARAMETER	システム関数呼び出し時の不正なパラメータ。	GetLastError
ERR_INVALID_POINTER	不正なポインタ。	GetLastError
ERR_INVALID_POINTER_TYPE	不正なポインタ型。	GetLastError
ERR_LONG_ARRAY_ONLY	long型の配列のみ可。	GetLastError
ERR_MAIL_SEND_FAILED	メール送信に失敗。	GetLastError
ERR_MARKET_LASTTIME_UNKNOWN	最後のティックの時間が未知（ティック不在）。	GetLastError
ERR_MARKET_NOT_SELECTED	「気配値表示」でシンボルが未選択。	GetLastError

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
ERR_MARKET_SELECT_ERROR	「気配値表示」のシンボル追加・削除に失敗。	GetLastError
ERR_MARKET_UNKNOWN_SYMBOL	未知のシンボル。	GetLastError
ERR_MARKET_WRONG_PROPERTY	シンボルプロパティの不正な識別子。	GetLastError
ERR_MQL5_WRONG_PROPERTY	プログラムプロパティの不正な識別子。	GetLastError
ERR_NO_STRING_DATE	文字列内に日付なし。	GetLastError
ERR_NOT_ENOUGH_MEMORY	システム関数の実行に不充分なメモリ。	GetLastError
ERR_NOTIFICATION_SEND_FAILED	通知送信に失敗。	GetLastError

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
ERR_NOTIFICATION_TOO_FREQUENT	頻繁過ぎる通知の送信。	GetLastError
ERR_NOTIFICATION_WRONG_PARAMETER	通知送信に無効なパラメータ（空の文字列かNULL）がSendNotification()関数に渡されました。	GetLastError
ERR_NOTIFICATION_WRONG_SETTINGS	端末内の不正な通知の設定（IDの未指定か、許可の無設定）。	GetLastError
ERR_NOTINITIALIZED_STRING	初期化されていない文字列。	GetLastError
ERR_NUMBER_ARRAYS_ONLY	数値配列	GetLastError

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	のみ可。	
ERR_OBJECT_ERROR	グラフィックオブジェクト操作エラー。	GetLastError
ERR_OBJECT_GETDATE_FAILED	値に対応する日付の取得が不可能。	GetLastError
ERR_OBJECT_GETVALUE_FAILED	日付に対応する値の取得が不可能。	GetLastError
ERR_OBJECT_NOT_FOUND	グラフィックオブジェクトが見つかりません。	GetLastError
ERR_OBJECT_WRONG_PROPERTY	グラフィックオブジェクトプロパティの不正なID。	GetLastError
ERR_ONEDIM_ARRAYS_ONLY	1 次元配列の	GetLastError

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	みが可。	
ERR_OPENCL_BUFFER_CREATE	OpenCLバッファの作成に失敗。	GetLastError
ERR_OPENCL_CONTEXT_CREATE	OpenCL コンテキスト作成に失敗。	GetLastError
ERR_OPENCL_EXECUTE	OpenCL プログラムランタイムエラー。	GetLastError
ERR_OPENCL_INTERNAL	OpenCL実行中中に内部エラー発生。	GetLastError
ERR_OPENCL_INVALID_HANDLE	無効なOpenCL ハンドル。	GetLastError
ERR_OPENCL_KERNEL_CREATE	OpenCL カーネル作成エラー。	GetLastError

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
ERR_OPENCL_NOT_SUPPORTED	OpenCL 関数がこのコンピューターでサポートされていません。	GetLastError
ERR_OPENCL_PROGRAM_CREATE	OpenCL プログラムのコンパイル中にエラー発生。	GetLastError
ERR_OPENCL_QUEUE_CREATE	OpenCL の実行キューの作成に失敗。	GetLastError
ERR_OPENCL_SET_KERNEL_PARAMETER	OpenCL カーネルパラメータ設定中にエラー発生。	GetLastError
ERR_OPENCL_TOO_LONG_KERNEL_NAME	カーネル名が長すぎます (OpenCL カーネル)。	GetLastError

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
ERR_OPENCL_WRONG_BUFFER_OFFSET	OpenCLバッファ内の無効なオフセット。	GetLastError
ERR_OPENCL_WRONG_BUFFER_SIZE	OpenCLバッファの無効なサイズ。	GetLastError
ERR_PLAY_SOUND_FAILED	音の再生に失敗。	GetLastError
ERR_RESOURCE_NAME_DUPLICATED	同一の動的及び静的リソース名。	GetLastError
ERR_RESOURCE_NAME_IS_TOO_LONG	リソース名が 63 字を超えます。	GetLastError
ERR_RESOURCE_NOT_FOUND	この名称のリソースがEX5で見つかりません。	GetLastError
ERR_RESOURCE_UNSUPPORTED_TYPE	リソースタイプがサポート	GetLastError

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	されていないかサイズが16 MBを超えます。	
ERR_SERIES_ARRAY	時系列は使用不可。	GetLastError
ERR_SHORT_ARRAY_ONLY	short型の配列のみ可。	GetLastError
ERR_SMALL_ARRAY	配列が小さすぎ、開始位置が配列の外側にあります。	GetLastError
ERR_SMALL_ASSERIES_ARRAY	受け取り側の配列はAS_SERIESとして宣言されていて、サイズが不充分です。	GetLastError
ERR_STRING_OUT_OF_MEMORY	文字列に不充	GetLastError

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	分なメモリ。	
ERR_STRING_RESIZE_ERROR	文字列の再配置に不充分なメモリ。	GetLastError
ERR_STRING_SMALL_LEN	予想より短い文字列。	GetLastError
ERR_STRING_TIME_ERROR	文字列から日付への変換エラー。	GetLastError
ERR_STRING_TOO_BIGNUMBER	大きすぎてULONG_MAXを超える数。	GetLastError
ERR_STRING_UNKNOWNTYPE	文字列への変換時の未知なデータ型。	GetLastError
ERR_STRING_ZEROADDED	文字列の末尾に 0 が追加されまし	GetLastError

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	た（無効な操作）。	
ERR_STRINGPOS_OUTOFRANGE	位置が文字列の範囲外。	GetLastError
ERR_STRUCT_WITHOBJECTS_ORCLASS	構造体に、文字列、及び/または動的な配列及び/またはそのようなオブジェクトの構造体及び/またはクラスが含まれます。	GetLastError
ERR_SUCCESS	操作が正常に完了しました。	GetLastError
ERR_TERMINAL_WRONG_PROPERTY	端末プロパティの不正な識別子。	GetLastError

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
ERR_TOO_LONG_FILENAME	長すぎるファイル名。	GetLastError
ERR_TOO_MANY_FILES	64 を超えるファイルを同時に開く事は不可能。	GetLastError
ERR_TOO_MANY_FORMATTERS	フォーマット指定子数がパラメータ数を超過。	GetLastError
ERR_TOO_MANY_PARAMETERS	パラメータ数がフォーマット指定子数を超過。	GetLastError
ERR_TRADE_DEAL_NOT_FOUND	約定が見つかりません。	GetLastError
ERR_TRADE_DISABLED	エキスパートアドバイザーでの取引が許可されてい	GetLastError

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	ません。	
ERR_TRADE_ORDER_NOT_FOUND	注文が見つかりません。	GetLastError
ERR_TRADE_POSITION_NOT_FOUND	ポジションが見つかりません。	GetLastError
ERR_TRADE_SEND_FAILED	取引リクエスト送信に失敗。	GetLastError
ERR_TRADE_WRONG_PROPERTY	不正な取引プロパティ識別子。	GetLastError
ERR_USER_ERROR_FIRST	ユーザ定義エラーはこのコードで始まります。	GetLastError
ERR_WEBREQUEST_CONNECT_FAILED	指定されたURLへの接続に失敗。	GetLastError
ERR_WEBREQUEST_INVALID_ADDRESS	無効なURL。	GetLastError

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
ERR_WEBREQUEST_REQUEST_FAILED	HTTPリクエストに失敗。	GetLastError
ERR_WEBREQUEST_TIMEOUT	タイムアウトを超過。	GetLastError
ERR_WRONG_DIRECTORYNAME	不正なディレクトリ名。	GetLastError
ERR_WRONG_FILEHANDLE	不正なファイルハンドル。	GetLastError
ERR_WRONG_FILENAME	無効なファイル名。	GetLastError
ERR_WRONG_FORMATSTRING	無効なフォーマット ストリング。	GetLastError
ERR_WRONG_INTERNAL_PARAMETER	クライアント端末関数の内部呼び出しでの不正なパラメータ。	GetLastError
ERR_WRONG_STRING_DATE	文字列で不正	GetLastError

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	な日付。	
ERR_WRONG_STRING_OBJECT	破損した文字列オブジェクト。	GetLastError
ERR_WRONG_STRING_PARAMETER	損傷した文字列型のパラメータ。	GetLastError
ERR_WRONG_STRING_TIME	文字列内の不正な時刻。	GetLastError
ERR_ZEROSIZE_ARRAY	長さゼロの配列。	GetLastError
FILE_ACCESS_DATE	最終アクセス日。	FileGetInteger
FILE_ANSI	ANSIタイプ（1バイトのシンボル）の文字列。フラグはFileOpen()で使用されます。	FileOpen

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
FILE_BIN	（文字列への変換文字列を含まない）バイナリ読み込み/書き込みモードです。フラグはFileOpen()で使用されます。	FileOpen
FILE_COMMON	全てのクライアント端末の共有フォルダ内のファイルパス\Terminal\Common\Filesフラグは FileOpen()、FileCopy()、FileMove()とFileIs	FileOpen、FileCopy、FileMove、FileIsExist

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	Exist() 関数で利用されます。	
FILE_CREATE_DATE	作成日。	FileGetInteger
FILE_CSV	CSVファイル(全ての要素がUnicodeまたはANSIの適切な型の文字列に変換されセパレータで区切られています)。フラグはFileOpen()で使用されます。	FileOpen
FILE_END	ファイル終了の記号を取得。	FileGetInteger

© 2000-2025, MetaQuotes Ltd.

定数	説明	使用例
FILE_EXISTS	ファイルの存在をチェック。	FileGetInteger
FILE_IS_ANSI	ファイルをANSI（FILE_ANSIを参照）として開きます。	FileGetInteger
FILE_IS_BINARY	ファイルをバイナリ（FILE_BINを参照）として開きます。	FileGetInteger
FILE_IS_COMMON	ファイルを全ての端末の共有フォルダ（FILE_COMMONを参照）で開きます。	FileGetInteger
FILE_IS_CSV	ファイルをCSV（FILE_CSV	FileGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	を参照）として開きます。	
FILE_IS_READABLE	開かれたファイルが読み込み可能（FILE_READ を参照）。	FileGetInteger
FILE_IS_TEXT	ファイルをテキスト（FILE_TXT を参照）として開きます。	FileGetInteger
FILE_IS_WRITABLE	開かれたファイルが書き込み可能（FILE_WRITE を参照）。	FileGetInteger
FILE_LINE_END	行終了の記号を取得。	FileGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
FILE_MODIFY_DATE	最終更新日。	FileGetInteger
FILE_POSITION	ファイル内のポインタ位置。	FileGetInteger
FILE_READ	ファイルは読み込むために開かれます。フラグはFileOpen()で使用されます。ファイルを開く際にはFILE_WRITE 及び/またはFILE_READ仕様が必要です。	FileOpen
FILE_REWRITE	FileCopy()及びFileMove()関数でのファイ	FileCopy, FileMove

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	ル上書きを可能にします。ファイルが存在し、書き込み用に開かれる必要があります。さもなければ、ファイルは開かれません。	
FILE_SHARE_READ	複数のプログラムからの読み込みのための共通のアクセス。フラグはFileOpen()で使用されますがファイルを開く時のFILE_WRITE 及	FileOpen

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	び/またはFILE_READフラグは必要です。	
FILE_SHARE_WRITE	複数のプログラムからの書き込みのための共通のアクセス。フラグはFileOpen()で使用されますがファイルを開く時のFILE_WRITE 及び/またはFILE_READフラグは必要です。	FileOpen
FILE_SIZE	ファイルのバイトサイズ。	FileGetInteger
FILE_TXT	単純なテキスト	FileOpen

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	ファイル（CSVファイルと同じですが、セパレータを考慮しません）。フラグはFileOpen()で使用されます。	
FILE_UNICODE	UNICODEタイプ（2バイトのシンボル）の文字列。フラグはFileOpen()で使用されます。	FileOpen
FILE_WRITE	ファイルは書き込むために開かれます。フラグはFileOpen()	FileOpen

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	で使用されます。ファイルを開く際にはFILE_WRITE 及び/またはFILE_READ仕様が必要です。	
FLT_DIG	float型の小数点以下の桁数。	数値定数
FLT_EPSILON	下記の条件を満たす最小値。1.0+DBL_EPSILON != 1.0 (float型)	数値定数
FLT_MANT_DIG	float型の仮数部のビット数。	数値定数

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
FLT_MAX	float型で表すことが出来る最大値。。	数値定数
FLT_MAX_10_EXP	float型の指数度の10進数での最大値。	数値定数
FLT_MAX_EXP	float型の指数度の2進数での最大値。	数値定数
FLT_MIN	float型で表すことが出来る最小の正の値。	数値定数
FLT_MIN_10_EXP	float型の指数度の10進数での最小値。	数値定数

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
FLT_MIN_EXP	float型の指数度の2進数での最小値。	数値定数
FRIDAY	金曜。	SymbolInfoInteger、SymbolInfoSessionQuote、SymbolInfoSessionTrade
GANN_DOWN_TREND	下落トレンドに対応する線。	ObjectSetInteger、ObjectGetInteger
GANN_UP_TREND	上昇トレンドに対応する線。	ObjectSetInteger、ObjectGetInteger
GATORJAW_LINE	顎線。	指標線
GATORLIPS_LINE	口線。	指標線
GATORTEETH_LINE	歯線。	指標線
IDABORT	「中止(A)」ボタンが押されました。	メッセージボックス
IDCANCEL	「キャンセル」ボタンが押されました。	メッセージボックス

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
IDCONTINUE	「続行(C)」ボタンが押されました。	メッセージボックス
IDIGNORE	「無視(I)」ボタンが押されました。	メッセージボックス
IDNO	「いいえ」ボタンが押されました。	メッセージボックス
IDOK	「OK」ボタンが押されました。	メッセージボックス
IDRETRY	「再試行(R)」ボタンが押されました。	メッセージボックス
IDTRYAGAIN	「再試行する」ボタンが押されました。	メッセージボックス
IDYES	「はい」ボタンが押されました。	メッセージボックス

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
IND_AC	ACオシレーター。	IndicatorCreate、IndicatorParameters
IND_AD	蓄積/配信（Accumulation/Distribution）。	IndicatorCreate、IndicatorParameters
IND_ADX	平均方向性指数（Average Directional Index）。	IndicatorCreate、IndicatorParameters
IND_ADXW	ウェルズ·ワイルダーのADX。	IndicatorCreate、IndicatorParameters
IND_ALLIGATOR	アリゲーター。	IndicatorCreate、IndicatorParameters
IND_AMA	適応型移動平均（Adaptive Moving Average）。	IndicatorCreate、IndicatorParameters
IND_AO	オーサムオシレーター。	IndicatorCreate、IndicatorParameters

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
IND_ATR	ATR（Average True Range）。	IndicatorCreate、IndicatorParameters
IND_BANDS	ボリンジャーバンド®（Bollinger Bands®）。	IndicatorCreate、IndicatorParameters
IND_BEARS	ベアパワー（Bears Power）。	IndicatorCreate、IndicatorParameters
IND_BULLS	ブルパワー（Bulls Power）。	IndicatorCreate、IndicatorParameters
IND_BWMFI	マーケットファシリテーションインデックス。	IndicatorCreate、IndicatorParameters
IND_CCI	コモディティチャンネルインデックス（Commodity Channel	IndicatorCreate、IndicatorParameters

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	Index）。	
IND_CHAIKIN	チャイキンオシレーター（Chaikin Oscillator）。	IndicatorCreate、IndicatorParameters
IND_CUSTOM	カスタム指標。	IndicatorCreate、IndicatorParameters
IND_DEMA	2重指数移動平均（Double Exponential Moving Average）。	IndicatorCreate、IndicatorParameters
IND_DEMARKER	デマーカー（DeMarker）。	IndicatorCreate、IndicatorParameters
IND_ENVELOPES	エンベロープ（Envelopes）。	IndicatorCreate、IndicatorParameters
IND_FORCE	勢力指数（Force Index）。	IndicatorCreate、IndicatorParameters
IND_FRACTALS	フラクタル。	IndicatorCreate、IndicatorParameters

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
IND_FRAMA	フラクタル適応型移動平均（Fractal Adaptive Moving Average）。	IndicatorCreate、IndicatorParameters
IND_GATOR	ゲーターオシレーター。	IndicatorCreate、IndicatorParameters
IND_ICHIMOKU	一目均衡表（Ichimoku Kinko Hyo）。	IndicatorCreate、IndicatorParameters
IND_MA	移動平均（Moving Average）。	IndicatorCreate、IndicatorParameters
IND_MACD	MACD	IndicatorCreate、IndicatorParameters
IND_MFI	マネーフローインデックス（Money Flow Index）。	IndicatorCreate、IndicatorParameters
IND_MOMENTUM	モメンタム（Mom	IndicatorCreate、IndicatorParameters

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	entum）。	
IND_OBV	オンバランスボリューム（On Balance Volume）。	IndicatorCreate、IndicatorParameters
IND_OSMA	OsMA	IndicatorCreate、IndicatorParameters
IND_RSI	相対力指数（Relative Strength Index）。	IndicatorCreate、IndicatorParameters
IND_RVI	相対活力指数（Relative Vigor Index）。	IndicatorCreate、IndicatorParameters
IND_SAR	パラボリックSAR。	IndicatorCreate、IndicatorParameters
IND_STDDEV	標準偏差（Standard Deviation）。	IndicatorCreate、IndicatorParameters
IND_STOCHASTIC	ストキャスティックス（Stoc	IndicatorCreate、IndicatorParameters

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	hastic Oscillator）。	
IND_TEMA	3 重指数移動平均（Triple Exponential Moving Average）。	IndicatorCreate、IndicatorParameters
IND_TRIX	3 重指数移動平均オシレーター（Triple Exponential Moving Averages Oscillator）。	IndicatorCreate、IndicatorParameters
IND_VIDYA	可変インデックス動的平均（Variable Index Dynamic Average）。	IndicatorCreate、IndicatorParameters

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
IND_VOLUMES	ボリューム。	IndicatorCreate、IndicatorParameters
IND_WPR	ウィリアムパーセントレンジ（Williams' Percent Range）。	IndicatorCreate、IndicatorParameters
INDICATOR_CALCULATIONS	中間計算のための補助バッファ	SetIndexBuffer
INDICATOR_COLOR_INDEX	色。	SetIndexBuffer
INDICATOR_DATA	描画するデータ。	SetIndexBuffer
INDICATOR_DIGITS	指標値描画の精度。	IndicatorSetInteger
INDICATOR_HEIGHT	指標ウィンドウの固定された高さ（プリプロセッサコマンド #property indicator_height）。	IndicatorSetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
INDICATOR_LEVELCOLOR	レベルの線の色。	IndicatorSetInteger
INDICATOR_LEVELS	指標ウィンドウのレベル数。	IndicatorSetInteger
INDICATOR_LEVELSTYLE	レベルの線のスタイル。	IndicatorSetInteger
INDICATOR_LEVELTEXT	レベルの説明。	IndicatorSetString
INDICATOR_LEVELVALUE	レベル値。	IndicatorSetDouble
INDICATOR_LEVELWIDTH	レベルの線の太さ。	IndicatorSetInteger
INDICATOR_MAXIMUM	指標ウィンドウの最大サイズ。	IndicatorSetDouble
INDICATOR_MINIMUM	指標ウィンドウの最小サイズ。	IndicatorSetDouble
INDICATOR_SHORTNAME	指標短縮名。	IndicatorSetString
INT_MAX	int 型で表すことが出来る最大値。。	数値定数

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
INT_MIN	int 型で表すことが出来る最小値。。	数値定数
INVALID_HANDLE	不正なハンドル。	その他の定数
IS_DEBUG_MODE	MQ5プログラムをデバグモードで動作するフラグ。	その他の定数
IS_PROFILE_MODE	MQ5プログラムをプロファイルモードで動作するフラグ。	その他の定数
KIJUNSEN_LINE	基準線。	指標線
LICENSE_DEMO	マーケットから購入した製品の試用版。ストラテジーテスター内のみで使用	MQLInfoInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	出来ます。	
LICENSE_FREE	無制限の無料版。	MQLInfoInteger
LICENSE_FULL	購入したライセンス版では、少なくとも 5 アクティベーションが可能です。アクティベーションの数は、売主によって指定されます。売主は、アクティベーションの許容数を増加させることが出来ます。	MQLInfoInteger
LICENSE_TIME	期間限定ライセ	MQLInfoInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	ンス版。	
LONG_MAX	long型で表すことが出来る最大値。。	数値定数
LONG_MIN	long型で表すことが出来る最小値。。	数値定数
LOWER_BAND	下限。	指標線
LOWER_HISTOGRAM	下部ヒストグラム。	指標線
LOWER_LINE	下部線。	指標線
M_1_PI	1/pi	数学定数
M_2_PI	2/pi	数学定数
M_2_SQRTPI	2/sqrt(pi)	数学定数
M_E	e	数学定数
M_LN10	ln(10)	数学定数
M_LN2	ln(2)	数学定数
M_LOG10E	log10(e)	数学定数
M_LOG2E	log2(e)	数学定数
M_PI	pi	数学定数
M_PI_2	pi/2	数学定数

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
M_PI_4	pi/4	数学定数
M_SQRT1_2	1/sqrt(2)	数学定数
M_SQRT2	sqrt(2)	数学定数
MAIN_LINE	正中線。	指標線
MB_ABORTRETRYIGNORE	メッセージウィンドウには、「中止(A)」「再試行(R)」「無視(I)」の3つのボタンがあります。	メッセージボックス
MB_CANCELTRYCONTINUE	メッセージウィンドウには、「キャンセル」「再試行(R)」「続行(C)」の3つのボタンがあります。	メッセージボックス

© 2000-2025, MetaQuotes Ltd.

定数	説明	使用例
MB_DEFBUTTON1	MB_DEFBUTTON2、MB_DEFBUTTON3及びMB_DEFBUTTON4が指定されてない場合、最初のボタンであるMB_DEFBUTTON1がデフォルトです。	メッセージボックス
MB_DEFBUTTON2	2番目のボタンがデフォルト。	メッセージボックス
MB_DEFBUTTON3	3番目のボタンがデフォルト。	メッセージボックス
MB_DEFBUTTON4	4番目のボタンがデフォルト。	メッセージボックス

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
MB_ICONEXCLAMATION, MB_ICONWARNING	感嘆/警告記号アイコン。	メッセージボックス
MB_ICONINFORMATION, MB_ICONASTERISK	囲まれたiサイン。	メッセージボックス
MB_ICONQUESTION	はてなマークのアイコン。	メッセージボックス
MB_ICONSTOP, MB_ICONERROR, MB_ICONHAND	STOPアイコン。	メッセージボックス
MB_OK	メッセージウィンドウには、「OK」ボタンが1つあります。（デフォルト）。	メッセージボックス
MB_OKCANCEL	メッセージウィンドウには、「OK」と「キャンセル」の2つのボタンがあります。	メッセージボックス

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
MB_RETRYCANCEL	メッセージウィンドウには、「再試行(R)」と「キャンセル」の2つのボタンがあります。	メッセージボックス
MB_YESNO	メッセージウィンドウには、「はい」と「いいえ」の2つのボタンがあります。	メッセージボックス
MB_YESNOCANCEL	メッセージウィンドウには、「はい」、「いいえ」と「キャンセル」の3つのボタンがあります。	メッセージボックス

© 2000-2025, MetaQuotes Ltd.

定数	説明	使用例
MINUSDI_LINE	線 – DI。	指標線
MODE_EMA	指数移動平均。	平滑化の方法
MODE_LWMA	線形加重移動平均。	平滑化の方法
MODE_SMA	単純平均。	平滑化の方法
MODE_SMMA	平滑平均。	平滑化の方法
MONDAY	月曜。	SymbolInfoInteger、SymbolInfoSessionQuote、SymbolInfoSessionTrade
MQL_DEBUG	デバグモードを示すフラグ。	MQLInfoInteger
MQL_DLLS_ALLOWED	実行プログラムのDLLを使用する許可。	MQLInfoInteger
MQL_FRAME_MODE	エキスパートアドバイザーが最適化結果のフレーム収集モード	MQLInfoInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	で動作されるというフラグ。	
MQL_LICENSE_TYPE	EX5 モジュールのライセンスの種類。ライセンスは、MQLInfoInteger(MQL_LICENSE_TYPE) を使用してリクエストされたEX5モジュールを指します。	MQLInfoInteger
MQL_MEMORY_LIMIT	MQL5 プログラムの最大動的メモリ量（MB 単位）。	MQLInfoInteger
MQL_MEMORY_USED	MQL5 プログラムのメモリ使用量（MB	MQLInfoInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	単位）。	
MQL_OPTIMIZATION	最適化プロセスを示すフラグ。	MQLInfoInteger
MQL_PROFILER	プログラムがコードプロファイリングモードで動作することを示すフラグ。	MQLInfoInteger
MQL_PROGRAM_NAME	実行されるMQL5プログラムの名称。	MQLInfoString
MQL_PROGRAM_PATH	指定された実行プログラムのパス。	MQLInfoString
MQL_PROGRAM_TYPE	MQL5プログラムの種類。	MQLInfoInteger
MQL_SIGNALS_ALLOWED	指定された実行プログラムの信号	MQLInfoInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	を変更する権限。	
MQL_TESTER	テスタプロセスを示すフラグ。	MQLInfoInteger
MQL_TRADE_ALLOWED	指定された実行中プログラムの取引許可。	MQLInfoInteger
MQL_VISUAL_MODE	ビジュアルテスタプロセスを示すフラグ。	MQLInfoInteger
NULL	すべての型でゼロ。	その他の定数
OBJ_ALL_PERIODS	オブジェクトは全ての時間軸に描画されています。	ObjectSetInteger、ObjectGetInteger
OBJ_ARROW	矢印。	オブジェクト型
OBJ_ARROW_BUY	買いサイン。	オブジェクト型
OBJ_ARROW_CHECK	チェックサイ	オブジェクト型

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	ン。	
OBJ_ARROW_DOWN	下矢印。	オブジェクト型
OBJ_ARROW_LEFT_PRICE	左プライスラベル。	オブジェクト型
OBJ_ARROW_RIGHT_PRICE	右プライスラベル。	オブジェクト型
OBJ_ARROW_SELL	売りサイン。	オブジェクト型
OBJ_ARROW_STOP	ストッププサイン。	オブジェクト型
OBJ_ARROW_THUMB_DOWN	サムズダウン。	オブジェクト型
OBJ_ARROW_THUMB_UP	サムズアップ。	オブジェクト型
OBJ_ARROW_UP	上矢印。	オブジェクト型
OBJ_ARROWED_LINE	矢印。	オブジェクト型
OBJ_BITMAP	ビットマップ。	オブジェクト型
OBJ_BITMAP_LABEL	ビットマップラベル。	オブジェクト型
OBJ_BUTTON	ボタン。	オブジェクト型
OBJ_CHANNEL	等距離チャンネル。	オブジェクト型
OBJ_CHART	チャート。	オブジェクト型

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
OBJ_CYCLES	サイクルライン。	オブジェクト型
OBJ_EDIT	編集。	オブジェクト型
OBJ_ELLIOTWAVE3	エリオット修正波。	オブジェクト型
OBJ_ELLIOTWAVE5	エリオット動機波。	オブジェクト型
OBJ_ELLIPSE	楕円形。	オブジェクト型
OBJ_EVENT	経済カレンダーのイベントに対応する「イベント」オブジェクト。	オブジェクト型
OBJ_EXPANSION	フィボナッチ拡張。	オブジェクト型
OBJ_FIBO	フィボナッチリトレースメント。	オブジェクト型
OBJ_FIBOARC	フィボナッチアーク（円弧）。	オブジェクト型
OBJ_FIBOCHANNEL	フィボナッチ	オブジェクト型

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	チャンネル。	
OBJ_FIBOFAN	フィボナッチファン（扇）。	オブジェクト型
OBJ_FIBOTIMES	フィボナッチタイムゾーン。	オブジェクト型
OBJ_GANNFAN	ギャンファン（扇）。	オブジェクト型
OBJ_GANNGRID	ギャングリッド。	オブジェクト型
OBJ_GANNLINE	ギャンライン。	オブジェクト型
OBJ_HLINE	水平線。	オブジェクト型
OBJ_LABEL	ラベル。	オブジェクト型
OBJ_NO_PERIODS	オブジェクトは全ての時間軸には描画されていません。	ObjectSetInteger、ObjectGetInteger
OBJ_PERIOD_D1	オブジェクトは1日チャートに描画	ObjectSetInteger、ObjectGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	されています。	
OBJ_PERIOD_H1	オブジェクトは 1 時間チャートに描画されています。	ObjectSetInteger、ObjectGetInteger
OBJ_PERIOD_H12	オブジェクトは 12 時間チャートに描画されています。	ObjectSetInteger、ObjectGetInteger
OBJ_PERIOD_H2	オブジェクトは 2 時間チャートに描画されています。	ObjectSetInteger、ObjectGetInteger
OBJ_PERIOD_H3	オブジェクトは 3 時間チャートに描画されています。	ObjectSetInteger、ObjectGetInteger
OBJ_PERIOD_H4	オブジェクトは 4	ObjectSetInteger、ObjectGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	時間チャートに描画されています。	
OBJ_PERIOD_H6	オブジェクトは 6 時間チャートに描画されています。	ObjectSetInteger、ObjectGetInteger
OBJ_PERIOD_H8	オブジェクトは 8 時間チャートに描画されています。	ObjectSetInteger、ObjectGetInteger
OBJ_PERIOD_M1	オブジェクトは 1 分チャートに描画されています。	ObjectSetInteger、ObjectGetInteger
OBJ_PERIOD_M10	オブジェクトは 10 分チャートに描画されています。	ObjectSetInteger、ObjectGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
OBJ_PERIOD_M12	オブジェクトは12分チャートに描画されています。	ObjectSetInteger、ObjectGetInteger
OBJ_PERIOD_M15	オブジェクトは15分チャートに描画されています。	ObjectSetInteger、ObjectGetInteger
OBJ_PERIOD_M2	オブジェクトは2分チャートに描画されています。	ObjectSetInteger、ObjectGetInteger
OBJ_PERIOD_M20	オブジェクトは20分チャートに描画されています。	ObjectSetInteger、ObjectGetInteger
OBJ_PERIOD_M3	オブジェクトは3分チャートに描画されて	ObjectSetInteger、ObjectGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	います。	
OBJ_PERIOD_M30	オブジェクトは30分チャートに描画されています。	ObjectSetInteger、ObjectGetInteger
OBJ_PERIOD_M4	オブジェクトは4分チャートに描画されています。	ObjectSetInteger、ObjectGetInteger
OBJ_PERIOD_M5	オブジェクトは5分チャートに描画されています。	ObjectSetInteger、ObjectGetInteger
OBJ_PERIOD_M6	オブジェクトは6分チャートに描画されています。	ObjectSetInteger、ObjectGetInteger
OBJ_PERIOD_MN1	オブジェクトは1ヶ月チャートに描画	ObjectSetInteger、ObjectGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	されています。	
OBJ_PERIOD_W1	オブジェクトは 1 週間チャートに描画されています。	ObjectSetInteger、ObjectGetInteger
OBJ_PITCHFORK	アンドリューズピッチフォーク。	オブジェクト型
OBJ_RECTANGLE	長方形。	オブジェクト型
OBJ_RECTANGLE_LABEL	カスタムグラフィカルインタフェースを作成及び設計するための「四角形ラベル」オブジェクト。	オブジェクト型
OBJ_REGRESSION	線形回帰チャンネル。	オブジェクト型
OBJ_STDDEVCHANNEL	標準偏差チャンネル。	オブジェクト型

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
OBJ_TEXT	テキスト。	オブジェクト型
OBJ_TREND	傾向線。	オブジェクト型
OBJ_TRENDBYANGLE	角度による傾向線。	オブジェクト型
OBJ_TRIANGLE	三角形。	オブジェクト型
OBJ_VLINE	垂直線。	オブジェクト型
OBJPROP_ALIGN	編集オブジェクト（OBJ_EDIT）の水平テキストの配置。	ObjectSetInteger、ObjectGetInteger
OBJPROP_ANCHOR	グラフィックオブジェクトのアンカーポイントの位置。	ObjectSetInteger、ObjectGetInteger
OBJPROP_ANGLE	角度。角度を指定されずにプログラムから作成されたオ	ObjectSetDouble, ObjectGetDouble

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	ブジェクトの場合、値はEMPTY_VALUE。	
OBJPROP_ARROWCODE	矢印オブジェクトの矢印コード。	ObjectSetInteger、ObjectGetInteger
OBJPROP_BACK	背景のオブジェクト。	ObjectSetInteger、ObjectGetInteger
OBJPROP_BGCOLOR	OBJ_EDIT、OBJ_BUTTON、OBJ_RECTANGLE_LABEL の背景色。	ObjectSetInteger、ObjectGetInteger
OBJPROP_BMPFILE	ビットマップラベルのBMPファイルの名称。リソースも参照。	ObjectSetString, ObjectGetString
OBJPROP_BORDER_COLOR	OBJ_EDIT 及び	ObjectSetInteger、ObjectGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	OBJ_BUTTON オブジェクトの境界線の色。	
OBJPROP_BORDER_TYPE	「四角形ラベル」オブジェクトの境界線の種類。	ObjectSetInteger、ObjectGetInteger
OBJPROP_CHART_ID	チャートオブジェクト（OBJ_CHART）の識別子。チャート操作で説明される関数を使用すると、通常のチャートと同様にこのオブジェクトのプロパティの	ObjectSetInteger、ObjectGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	操作が可能になりますが例外もあります。	
OBJPROP_CHART_SCALE	チャートオブジェクトのスケール。	ObjectSetInteger、ObjectGetInteger
OBJPROP_COLOR	色。	ObjectSetInteger、ObjectGetInteger
OBJPROP_CORNER	グラフィカルオブジェクトをリンクするチャートのコーナー。	ObjectSetInteger、ObjectGetInteger
OBJPROP_CREATETIME	オブジェクト作成時刻。	ObjectSetInteger、ObjectGetInteger
OBJPROP_DATE_SCALE	グラフオブジェクトの時間スケールの表示。	ObjectSetInteger、ObjectGetInteger
OBJPROP_DEGREE	エリオット波動マーキングの	ObjectSetInteger、ObjectGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	レベル。	
OBJPROP_DEVIATION	標準偏差チャンネルの偏差。	ObjectSetDouble, ObjectGetDouble
OBJPROP_DIRECTION	ギャンオブジェクトのトレンド。	ObjectSetInteger、ObjectGetInteger
OBJPROP_DRAWLINES	エリオット波動をマークするための線の表示。	ObjectSetInteger、ObjectGetInteger
OBJPROP_ELLIPSE	フィボナッチアークオブジェクトの完全な楕円形の表示(OBJ_FIBOARC)。	ObjectSetInteger、ObjectGetInteger
OBJPROP_FILL	色でオブジェクトを塗りつぶします（OBJ_RECTANGL	ObjectSetInteger、ObjectGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	E、OBJ_TRIANGLE、OBJ_ELLIPSE、OBJ_CHANNEL、OBJ_STDDEVCHANNEL、OBJ_REGRESSION）。	
OBJPROP_FONT	フォント。	ObjectSetString, ObjectGetString
OBJPROP_FONTSIZE	フォントサイズ。	ObjectSetInteger、ObjectGetInteger
OBJPROP_HIDDEN	端末の「チャート設定」->「オブジェクト」->「オブジェクト表示リスト」メニューで、オブジェクトリスト中のグラフィカルオブジェク	ObjectSetInteger、ObjectGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	トの名称を非表示にします。trueは、リストからのオブジェクトを非表示にします。MQL5プログラムで作成されたカレンダーイベントや取引履歴を表示するオブジェクトではデフォルトはtrueです。そのようなグラフィックオブジェクトを表示してプロパティにアクセスするには、	

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	「オブジェクト表示リスト」ウィンドウの「すべて」ボタンをクリックします。	
OBJPROP_LEVELCOLOR	ラインレベルの色。	ObjectSetInteger、ObjectGetInteger
OBJPROP_LEVELS	レベルの数。	ObjectSetInteger、ObjectGetInteger
OBJPROP_LEVELSTYLE	ラインレベルのスタイル。	ObjectSetInteger、ObjectGetInteger
OBJPROP_LEVELTEXT	レベルの説明。	ObjectSetString, ObjectGetString
OBJPROP_LEVELVALUE	レベル値。	ObjectSetDouble, ObjectGetDouble
OBJPROP_LEVELWIDTH	ラインレベルの太さ。	ObjectSetInteger、ObjectGetInteger
OBJPROP_NAME	オブジェクト名。	ObjectSetString, ObjectGetString
OBJPROP_PERIOD	チャートオブジェクトの時間軸。	ObjectSetInteger、ObjectGetInteger
OBJPROP_PRICE	価格座標。	ObjectSetDouble, ObjectGetDouble

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
OBJPROP_PRICE_SCALE	チャートオブジェクトの価格スケールの表示。	ObjectSetInteger、ObjectGetInteger
OBJPROP_RAY	縦線はチャートの全てのウィンドウを通過します。	ObjectSetInteger、ObjectGetInteger
OBJPROP_RAY_LEFT	レイは左に行きます。	ObjectSetInteger、ObjectGetInteger
OBJPROP_RAY_RIGHT	レイは右に行きます。	ObjectSetInteger、ObjectGetInteger
OBJPROP_READONLY	編集オブジェクトでテキストを編集する能力。	ObjectSetInteger、ObjectGetInteger
OBJPROP_SCALE	スケール（ギャンオブジェクトとフィボナッチアークのプロ	ObjectSetDouble, ObjectGetDouble

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	パティ)。	
OBJPROP_SELECTABLE	オブジェクトの可用性。	ObjectSetInteger、ObjectGetInteger
OBJPROP_SELECTED	オブジェクトが選択されています。	ObjectSetInteger、ObjectGetInteger
OBJPROP_STATE	ボタンの状態(押された/押されていない)。	ObjectSetInteger、ObjectGetInteger
OBJPROP_STYLE	スタイル。	ObjectSetInteger、ObjectGetInteger
OBJPROP_SYMBOL	チャートオブジェクトの記号。	ObjectSetString, ObjectGetString
OBJPROP_TEXT	オブジェクトの説明(オブジェクトに含まれるテキスト)。	ObjectSetString, ObjectGetString
OBJPROP_TIME	時間座	ObjectSetInteger、ObjectGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	標。	
OBJPROP_TIMEFRAMES	時間軸でのオブジェクトの可視性。	ObjectSetInteger、ObjectGetInteger
OBJPROP_TOOLTIP	ツールヒントのテキスト。プロパティが設定されていない場合は、端末によって自動的に生成されたツールヒントが表示されます。ツールヒントは「¥ n」（改行）の値を設定することによって無効に出来ます。	ObjectSetString, ObjectGetString
OBJPROP_TYPE	オブジェク	ObjectSetInteger、ObjectGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	トの型。	
OBJPROP_WIDTH	線の太さ。	ObjectSetInteger、ObjectGetInteger
OBJPROP_XDISTANCE	結合角からX 軸に沿ったピクセル単位での距離（注意事項を参照）。	ObjectSetInteger、ObjectGetInteger
OBJPROP_XOFFSET	ビットマップラベル（ OBJ_BITMAP_LABEL ）及びビットマップ（ OBJ_BITMAP ）グラフィックオブジェクトの四角形の表示領域の左上隅のX 座標。値は、	ObjectSetInteger、ObjectGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	元の画像の左上隅を基準にピクセル単位で設定されています。	
OBJPROP_XSIZE	X 軸に沿ったピクセル単位でのオブジェクトの幅。OBJ_LABEL（読み込みのみ）、OBJ_BUTTON, OBJ_CHART、OBJ_BITMAP、OBJ_BITMAP_LABEL、OBJ_EDIT、OBJ_RECTANGLE_LABEL オブ	ObjectSetInteger、ObjectGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	ジェクトに指定されます。	
OBJPROP_YDISTANCE	結合角からY軸に沿ったピクセル単位での距離（注意事項を参照）。	ObjectSetInteger、ObjectGetInteger
OBJPROP_YOFFSET	ビットマップラベル（OBJ_BITMAP_LABEL）及びビットマップ（OBJ_BITMAP）グラフィックオブジェクトの四角形の表示領域の左上隅のY座標。値	ObjectSetInteger、ObjectGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	は、元の画像の左上隅を基準にピクセル単位で設定されています。	
OBJPROP_YSIZE	Y 軸に沿ったオブジェクトの高さ（ピクセル単位）。OBJ_LABEL（読み込みのみ）、OBJ_BUTTON, OBJ_CHART、OBJ_BITMAP、OBJ_BITMAP_LABEL、OBJ_EDIT、OBJ_RECTANGLE	ObjectSetInteger、ObjectGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	_LABEL オブジェクトに指定されます。	
OBJPROP_ZORDER	チャート上のクリックイベント（CHARTEVENT_CLICK）を受け取るためのグラフィカルオブジェクトの優先順位。優先順位はオブジェクトの作成時にデフォルトのゼロに設定され、必要に応じて高く出来ます。オブジェク	ObjectSetInteger、ObjectGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	トを互いに適用した場合、最も優先順位の高いオブジェクトのみがCHARTEVENT_CLICKイベントを受け取ります。	
ORDER_COMMENT	注文コメント。	OrderGetString、HistoryOrderGetString
ORDER_FILLING_FOK	この充填ポリシーは、注文が指定された量でのみ満たすことが出来ることを意味します。金融商品の必要量は、	OrderGetInteger、HistoryOrderGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	市場で現在利用出来ない場合、注文は実行されません。必要なボリュームは、市場で現在利用可能ないくつかのオファーを使用して満たすことが出来ます。	
ORDER_FILLING_IOC	このモードは、トレーダーが注文の選択肢のうちで、市場で最大限に利用可能なボリュー	OrderGetInteger、HistoryOrderGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	ムの約定を実行することに同意することを意味します。注文の全てを満たすことが出来ない場合、利用可能なボリュームのみが満たされ、残りのボリュームはキャンセルされます。	
ORDER_FILLING_RETURN	このポリシーは、「マーケット」または「エクスチェンジ」の記号を持った	OrderGetInteger、HistoryOrderGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	成行注文（ORDER_TYPE_BUY 及び ORDER_TYPE_SELL）、指値注文及びストッププリミット注文（ORDER_TYPE_BUY_LIMIT、ORDER_TYPE_SELL_LIMIT、ORDER_TYPE_BUY_STOP_LIMIT 及び ORDER_TYPE_SELL_STOP_LIMIT）の 実行のみに使用されます。成行注文	

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	及び指値注文を部分的に満たす場合は、残りのボリュームはキャンセルされずに更に手続きされます。ORDER_TYPE_BUY_STOP_LIMITとORDER_TYPE_SELL_STOP_LIMIT注文の活性化のためには、ORDER_FILLING_RETURN実行型と対応する指値注文ORDE	

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	R_TYPE_BUY_LIMIT/ ORDER_TYPE_SELL_LIMIT が作成されます。	
ORDER_MAGIC	注文を出したエキスパートアドバイザーのID（各エキスパートアドバイザーは、独自のユニークな番号を作成するように設計されています）。	OrderGetInteger、HistoryOrderGetInteger
ORDER_POSITION_ID	実行後すぐに注文に設定されるポジション識別。	OrderGetInteger、HistoryOrderGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	注文の実行は注文を出すか、既存のポジション変更する約定となります。このポジションの識別子がこの時点で実行される注文のために設定されています。	
ORDER_PRICE_CURRENT	注文シンボルの現在の価格。	OrderGetDouble、HistoryOrderGetDouble
ORDER_PRICE_OPEN	注文で指定された価格。	OrderGetDouble、HistoryOrderGetDouble
ORDER_PRICE_STOPLIMIT	ストップリミット注文の指値注	OrderGetDouble、HistoryOrderGetDouble

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	文価格。	
ORDER_SL	決済逆指値。	OrderGetDouble、HistoryOrderGetDouble
ORDER_STATE	注文状態。	OrderGetInteger、HistoryOrderGetInteger
ORDER_STATE_CANCELED	クライアントによってキャンセルされた注文。	OrderGetInteger、HistoryOrderGetInteger
ORDER_STATE_EXPIRED	期限切れの注文。	OrderGetInteger、HistoryOrderGetInteger
ORDER_STATE_FILLED	実行済みの注文。	OrderGetInteger、HistoryOrderGetInteger
ORDER_STATE_PARTIAL	実行中の注文。	OrderGetInteger、HistoryOrderGetInteger
ORDER_STATE_PLACED	受付済みの注文。	OrderGetInteger、HistoryOrderGetInteger
ORDER_STATE_REJECTED	拒絶された注文。	OrderGetInteger、HistoryOrderGetInteger
ORDER_STATE_REQUEST_ADD	注文が登録中（取引システム	OrderGetInteger、HistoryOrderGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	に配置）。	
ORDER_STATE_REQUEST_CANCEL	注文が削除中（取引システムからの削除）。	OrderGetInteger、HistoryOrderGetInteger
ORDER_STATE_REQUEST_MODIFY	注文が更新中（パラメータの変更）。	OrderGetInteger、HistoryOrderGetInteger
ORDER_STATE_STARTED	注文がチェック済み、ブローカによっての受け入れがまだ。	OrderGetInteger、HistoryOrderGetInteger
ORDER_SYMBOL	注文シンボル。	OrderGetString、HistoryOrderGetString
ORDER_TIME_DAY	現在の取引日のみ有効である注文。	OrderGetInteger、HistoryOrderGetInteger
ORDER_TIME_DONE	注文の実行及びキャ	OrderGetInteger、HistoryOrderGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	ンセル時刻。	
ORDER_TIME_DONE_MSC	01.01.1970から経ったミリ秒で表された注文の実行/キャンセル時刻。	OrderGetInteger、HistoryOrderGetInteger
ORDER_TIME_EXPIRATION	注文の期限。	OrderGetInteger、HistoryOrderGetInteger
ORDER_TIME_GTC	GTC注文（キャンセルするまで有効）。	OrderGetInteger、HistoryOrderGetInteger
ORDER_TIME_SETUP	注文設定時刻。	OrderGetInteger、HistoryOrderGetInteger
ORDER_TIME_SETUP_MSC	01.01.1970から経過したミリ秒数で表された注文の実行が出された時刻。	OrderGetInteger、HistoryOrderGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
ORDER_TIME_SPECIFIED	有効期限まで有効な注文。	OrderGetInteger、HistoryOrderGetInteger
ORDER_TIME_SPECIFIED_DAY	注文が指定された日の23:59:59まで有効となります。この時刻が取引セッション外である場合は、注文は最も近い取引時間中に満了します。	OrderGetInteger、HistoryOrderGetInteger
ORDER_TP	決済指値。	OrderGetDouble、HistoryOrderGetDouble
ORDER_TYPE	注文の種類。	OrderGetInteger、HistoryOrderGetInteger
ORDER_TYPE_BUY	成行買い注文。	OrderGetInteger、HistoryOrderGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
ORDER_TYPE_BUY_LIMIT	買い指値注文。	OrderGetInteger、HistoryOrderGetInteger
ORDER_TYPE_BUY_STOP	買い逆指値注文。	OrderGetInteger、HistoryOrderGetInteger
ORDER_TYPE_BUY_STOP_LIMIT	注文価格に達すると、未決の買い指値注文はストップリミット価格で出されます。	OrderGetInteger、HistoryOrderGetInteger
ORDER_TYPE_FILLING	注文充填タイプ。	OrderGetInteger、HistoryOrderGetInteger
ORDER_TYPE_SELL	成行売り注文。	OrderGetInteger、HistoryOrderGetInteger
ORDER_TYPE_SELL_LIMIT	売り指値注文。	OrderGetInteger、HistoryOrderGetInteger
ORDER_TYPE_SELL_STOP	売り逆指値注文。	OrderGetInteger、HistoryOrderGetInteger
ORDER_TYPE_SELL_STOP_LIMIT	注文価格に達	OrderGetInteger、HistoryOrderGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	すると、未決の売り指値注文はストップリミット価格で出されます。	
ORDER_TYPE_TIME	注文ライフタイム。	OrderGetInteger、HistoryOrderGetInteger
ORDER_VOLUME_CURRENT	注文の現在ボリューム。	OrderGetDouble、HistoryOrderGetDouble
ORDER_VOLUME_INITIAL	注文の初期ボリューム。	OrderGetDouble、HistoryOrderGetDouble
PERIOD_CURRENT	現在の時間軸	チャート時間軸
PERIOD_D1	1日	チャート時間軸
PERIOD_H1	1時間	チャート時間軸
PERIOD_H12	12時間	チャート時間軸
PERIOD_H2	2時間	チャート時間軸
PERIOD_H3	3時間	チャート時間軸
PERIOD_H4	4時間	チャート時間軸
PERIOD_H6	6時間	チャート時間軸
PERIOD_H8	8時間	チャート時間軸
PERIOD_M1	1分	チャート時間軸

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
PERIOD_M10	10分	チャート時間軸
PERIOD_M12	12分	チャート時間軸
PERIOD_M15	15分	チャート時間軸
PERIOD_M2	2分	チャート時間軸
PERIOD_M20	20分	チャート時間軸
PERIOD_M3	3分	チャート時間軸
PERIOD_M30	30分	チャート時間軸
PERIOD_M4	4分	チャート時間軸
PERIOD_M5	5分	チャート時間軸
PERIOD_M6	6分	チャート時間軸
PERIOD_MN1	1ヶ月	チャート時間軸
PERIOD_W1	1週間	チャート時間軸
PLOT_ARROW	DRAW_ARROW スタイルの矢印コード。	PlotIndexSetInteger、PlotIndexGetInteger
PLOT_ARROW_SHIFT	DRAW_ARROW スタイルの矢印の垂直シフト。	PlotIndexSetInteger、PlotIndexGetInteger
PLOT_COLOR_INDEXES	色の数。	PlotIndexSetInteger、PlotIndexGetInteger
PLOT_DRAW_BEGIN	データウィンドウでの描画前の初期のバーの	PlotIndexSetInteger、PlotIndexGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	数や値。	
PLOT_DRAW_TYPE	グラフィック構成のタイプ。	PlotIndexSetInteger、PlotIndexGetInteger
PLOT_EMPTY_VALUE	プロットに使用される空の値で、何も描画しません。	PlotIndexSetDouble
PLOT_LABEL	データウィンドウに表示する指標グラフィカルシリーズの名称。表示のためにいくつかの指標バッファを必要とする複雑なグラフィカルなスタイルで作業する場合、	PlotIndexSetString

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	各バッファの名称は「;」セパレータを使用して指定することが出来ます。サンプルコードはDRAW_CANDLESにあります。	
PLOT_LINE_COLOR	描画の色を含むバッファのインデックス。	PlotIndexSetInteger、PlotIndexGetInteger
PLOT_LINE_STYLE	線の描画スタイル。	PlotIndexSetInteger、PlotIndexGetInteger
PLOT_LINE_WIDTH	描画線の太さ。	PlotIndexSetInteger、PlotIndexGetInteger
PLOT_SHIFT	バーでの時間軸に沿っての指標のプロット	PlotIndexSetInteger、PlotIndexGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	シフト。	
PLOT_SHOW_DATA	データウィンドウでの建設値の表示のサイン。	PlotIndexSetInteger、PlotIndexGetInteger
PLUSDI_LINE	線 + DI。	指標線
POINTER_AUTOMATIC	自動的に（new()を使用せずに）作成されたオブジェクトのポインタ。	CheckPointer
POINTER_DYNAMIC	new()演算子で作成されたオブジェクトのポインタ。	CheckPointer
POINTER_INVALID	不正ポインタ。	CheckPointer
POSITION_COMMENT	ポジションコメント。	PositionGetString

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
POSITION_COMMISSION	手数料。	PositionGetDouble
POSITION_IDENTIFIER	ポジション識別子は、全ての新しくオープンしたポジションに割り当てられ、ポジションのライフタイムに一貫する固有の番号です。識別子はポジションのターンオーバによって変更されません。	PositionGetInteger
POSITION_MAGIC	ポジションマジックナンバー（ORDER_MAGIC	PositionGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	を参照）。	
POSITION_PRICE_CURRENT	ポジションシンボルの現在価格。	PositionGetDouble
POSITION_PRICE_OPEN	ポジションの始値。	PositionGetDouble
POSITION_PROFIT	現在の利益。	PositionGetDouble
POSITION_SL	未決済ポジションの決済逆指レベル。	PositionGetDouble
POSITION_SWAP	累積スワップ。	PositionGetDouble
POSITION_SYMBOL	ポジションのシンボル。	PositionGetString
POSITION_TIME	注文が出された時刻。	PositionGetInteger
POSITION_TIME_MSC	01.01.1970から経ったミリ秒で表された注文	PositionGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	が出された時刻。	
POSITION_TIME_UPDATE	01.01.1970から経った秒数で表されたポジション変更時刻。	PositionGetInteger
POSITION_TIME_UPDATE_MSC	01.01.1970から経ったミリ秒で表されたポジション変更時刻。	PositionGetInteger
POSITION_TP	未決済ポジションの決済指値。	PositionGetDouble
POSITION_TYPE	ポジションの種類。	PositionGetInteger
POSITION_TYPE_BUY	買い。	PositionGetInteger
POSITION_TYPE_SELL	売り。	PositionGetInteger
POSITION_VOLUME	ポジションボリューム。	PositionGetDouble

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
PRICE_CLOSE	終値。	価格定数
PRICE_HIGH	期間の高値。	価格定数
PRICE_LOW	期間の安値。	価格定数
PRICE_MEDIAN	中間価格（高＋低）÷2。	価格定数
PRICE_OPEN	始値。	価格定数
PRICE_TYPICAL	典型的価格（高＋低＋終）÷3。	価格定数
PRICE_WEIGHTED	平均価格（高＋低＋終＋終）÷4。	価格定数
PROGRAM_EXPERT	エキスパート。	MQLInfoInteger
PROGRAM_INDICATOR	指標。	MQLInfoInteger
PROGRAM_SCRIPT	スクリプト。	MQLInfoInteger
REASON_ACCOUNT	別のアカウントが有効化される	UninitializeReason、OnDeinit

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	か、アカウントの設定変更によって取引サーバへの再接続が発生しました。	
REASON_CHARTCHANGE	シンボルかチャート期間が変更されました。	UninitializeReason、OnDeinit
REASON_CHARTCLOSE	チャートが閉じられました。	UninitializeReason、OnDeinit
REASON_CLOSE	端末が閉じられました。	UninitializeReason、OnDeinit
REASON_INITFAILED	この値は OnInit() ハンドラがゼロ以外の値を返したことを示します。	UninitializeReason、OnDeinit

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
REASON_PARAMETERS	入力パラメータがユーザーによって変更されました。	UninitializeReason、OnDeinit
REASON_PROGRAM	エキスパートアドバイザーがExpertRemove()関数を呼び出して操作を終了しました。	UninitializeReason、OnDeinit
REASON_RECOMPILE	プログラムが再コンパイルされました。	UninitializeReason、OnDeinit
REASON_REMOVE	プログラムがチャートから削除されました。	UninitializeReason、OnDeinit
REASON_TEMPLATE	新しいテンプレートが適用されました。	UninitializeReason、OnDeinit

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
SATURDAY	土曜。	SymbolInfoInteger、SymbolInfoSessionQuote、SymbolInfoSessionTrade
SEEK_CUR	ファイルポインタの現在位置。	FileSeek
SEEK_END	ファイルの終わり。	FileSeek
SEEK_SET	ファイルの始め。	FileSeek
SENKOUSPANA_LINE	先行スパンA線。	指標線
SENKOUSPANB_LINE	先行スパンB線。	指標線
SERIES_BARS_COUNT	現在のシンボル期間のバーの数。	SeriesInfoInteger
SERIES_FIRSTDATE	現在のシンボル期間の初日。	SeriesInfoInteger
SERIES_LASTBAR_DATE	現在のシンボル期間の最終バーの開始時刻。	SeriesInfoInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
SERIES_SERVER_FIRSTDATE	時間軸に関係ない、サーバ上でのシンボル履歴の最初の日付。	SeriesInfoInteger
SERIES_SYNCHRONIZED	現時点の銘柄/期間のデータ同期フラグ	SeriesInfoInteger
SERIES_TERMINAL_FIRSTDATE	時間軸に関係ない、クライアント端末でのシンボル履歴の最初の日付。	SeriesInfoInteger
SHORT_MAX	short型で表すことが出来る最大値。。	数値定数
SHORT_MIN	short型で表す	数値定数

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	ことが出来る最小値。。	
SIGNAL_BASE_AUTHOR_LOGIN	著者のログイン。	SignalBaseGetString
SIGNAL_BASE_BALANCE	口座残高。	SignalBaseGetDouble
SIGNAL_BASE_BROKER	ブローカー名（会社）。	SignalBaseGetString
SIGNAL_BASE_BROKER_SERVER	ブローカーサーバ。	SignalBaseGetString
SIGNAL_BASE_CURRENCY	シグナル基準通貨。	SignalBaseGetString
SIGNAL_BASE_DATE_PUBLISHED	公開日（サブスクリプションが可能になった日付）。	SignalBaseGetInteger
SIGNAL_BASE_DATE_STARTED	モニターの開始日。	SignalBaseGetInteger
SIGNAL_BASE_EQUITY	口座株式。	SignalBaseGetDouble
SIGNAL_BASE_GAIN	口座ゲイン。	SignalBaseGetDouble

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
SIGNAL_BASE_ID	シグナルID。	SignalBaseGetInteger
SIGNAL_BASE_LEVERAGE	口座レバレッジ。	SignalBaseGetInteger
SIGNAL_BASE_MAX_DRAWDOWN	口座最大ドローダウン。	SignalBaseGetDouble
SIGNAL_BASE_NAME	シグナル名。	SignalBaseGetString
SIGNAL_BASE_PIPS	利益（ピップス単位）。	SignalBaseGetInteger
SIGNAL_BASE_PRICE	シグナルサブスクリプション価格。	SignalBaseGetDouble
SIGNAL_BASE_RATING	建て格付けでのポジション。	SignalBaseGetInteger
SIGNAL_BASE_ROI	投資収益（%）。	SignalBaseGetDouble
SIGNAL_BASE_SUBSCRIBERS	加入者数。	SignalBaseGetInteger
SIGNAL_BASE_TRADE_MODE	口座の種類（0 – 実際、1 – デ	SignalBaseGetInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	モ、2 – コンテスト）。	
SIGNAL_BASE_TRADES	取引数。	SignalBaseGetInteger
SIGNAL_INFO_CONFIRMATIONS_DISABLED	フラグは、確認ダイアログなしでの同期を可能にします。	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_COPY_SLTP	決済逆指値及び決済指値フラグを複製。	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_DEPOSIT_PERCENT	預金比率（%）。	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_EQUITY_LIMIT	株式の制限。	SignalInfoGetDouble, SignalInfoSetDouble
SIGNAL_INFO_ID	シグナル ID（読み込みのみ）。	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_NAME	シグナル名（読み込み	SignalInfoGetString

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	のみ）。	
SIGNAL_INFO_SLIPPAGE	スリッページ（ポジションと同期して成行注文をする時と取引の取引の複製時に使用）。	SignalInfoGetDouble, SignalInfoSetDouble
SIGNAL_INFO_SUBSCRIPTION_ENABLED	「サブスクリプションによっての取引コピー」許可フラグ。	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_TERMS_AGREE	「シグナルサービスの利用規約に同意」フラグ（読み込みのみ）。	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_VOLUME_PERCENT	預金使用の最	SignalInfoGetDouble, SignalInfoSetDouble

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	大率（%）（読み込みのみ）。	
SIGNAL_LINE	シグナル線。	指標線
STAT_BALANCE_DD	金額での最大残高ドローダウン。取引過程では多数の残高ドローダウンがあります。ここで最大値が取得されます。	TesterStatistics
STAT_BALANCE_DD_RELATIVE	最高残高ドローダウンの際に記録された、金額の百分率で表された残高ドローダウン(STAT	TesterStatistics

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	_BALANCE_DDREL_PERCENT)	
STAT_BALANCE_DDREL_PERCENT	最大残高ドローダウン（%）。取引過程では多数のドローダウンがあり、各ドローダウンで百分率での相対ドローダウン値が算出されます。最高値が返されます。	TesterStatistics
STAT_BALANCEDD_PERCENT	最高残高ドローダウンの際に記録された、金額の百分率	TesterStatistics

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	で表された残高ドローダウン(STAT_BALANCE_DD)	
STAT_BALANCEMIN	最低残高値。	TesterStatistics
STAT_CONLOSSMAX	一連の負け取引での最大損失。値はゼロ以下です。	TesterStatistics
STAT_CONLOSSMAX_TRADES	STAT_CONLOSSMAX(一連の負け取引での最大損失)を形成する取引の数	TesterStatistics
STAT_CONPROFITMAX	一連の勝ち取引での最大利益。	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

定数	説明	使用例
	値はゼロ以上です。	
STAT_CONPROFITMAX_TRADES	STAT_CONPROFITMAX（一連の勝ち取引での最大利益）を形成する取引の数	TesterStatistics
STAT_CUSTOM_ONTESTER	OnTester() 関数が返したカスタムの最適化基準の計算値。	TesterStatistics
STAT_DEALS	約定数。	TesterStatistics
STAT_EQUITY_DD	金額での最大株式ドローダウン。取引過程では多数のドロー	TesterStatistics

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	ダウンがあります。ここで最大値が取得されます。	
STAT_EQUITY_DD_RELATIVE	最高株式ドローダウンの際に記録された、金額の百分率で表された株式ドローダウン（STAT_EQUITY_DDREL_PERCENT）	TesterStatistics
STAT_EQUITY_DDREL_PERCENT	最大株式ドローダウン（%）。取引過程では多数の株式ドローダウンがあり、各ドロー	TesterStatistics

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	ダウンで百分率での相対ドローダウン値が算出されます。最高値が返されます。	
STAT_EQUITYDD_PERCENT	最高残高ドローダウンの際に記録された、金額の百分率で表された株式ドローダウン(STAT_EQUITY_DD).	TesterStatistics
STAT_EQUITYMIN	最低株式値。	TesterStatistics
STAT_EXPECTED_PAYOFF	期待利得。	TesterStatistics
STAT_GROSS_LOSS	総損失（全ての負	TesterStatistics

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	の取引の合計）。値はゼロ以下です。	
STAT_GROSS_PROFIT	総利益（全ての収益性（正）の取引の合計）。値はゼロ以上です。	TesterStatistics
STAT_INITIAL_DEPOSIT	頭金の値。	TesterStatistics
STAT_LONG_TRADES	長期取引。	TesterStatistics
STAT_LOSS_TRADES	負け取引。	TesterStatistics
STAT_LOSSTRADES_AVGCON	一連の負け取引の平均の長さ。	TesterStatistics
STAT_MAX_CONLOSS_TRADES	負け取引の最長シリーズの取引回数STAT	TesterStatistics

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	_MAX_CONLOSSES	
STAT_MAX_CONLOSSES	負け取引の最長シリーズでの損失総額。	TesterStatistics
STAT_MAX_CONPROFIT_TRADES	勝ち取引の最長シリーズの取引回数STAT_MAX_CONWINS	TesterStatistics
STAT_MAX_CONWINS	勝ち取引の最も長い一連の利益総額。	TesterStatistics
STAT_MAX_LOSSTRADE	最大損失 - 全ての負け取引の最安値。値はゼロ以下です。	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

定数	説明	使用例
STAT_MAX_PROFITTRADE	最大の利益 – 全ての勝ち取引の最大値。値はゼロ以上です。	TesterStatistics
STAT_MIN_MARGINLEVEL	証拠金レベルの最小値。	TesterStatistics
STAT_PROFIT	テスト後の純利益、すなわち STAT_GROSS_PROFIT と STAT_GROSS_LOSS の合計。(STAT_GROSS_LOSS は常にゼロ以下です)	TesterStatistics
STAT_PROFIT_FACTOR	STAT_GROSS_PROFIT/	TesterStatistics

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	STAT_GROSS_LOSS の比に等しい利益率 STAT_GROSS_LOSS=0 の場合、利益率は DBL_MAX	
STAT_PROFIT_LONGTRADES	収益性の高い長期取引。	TesterStatistics
STAT_PROFIT_SHORTTRADES	収益性の高い短期取引。	TesterStatistics
STAT_PROFIT_TRADES	勝ち取引。	TesterStatistics
STAT_PROFITTRADES_AVGCON	一連の収益性の高い取引の平均の長さ。	TesterStatistics
STAT_RECOVERY_FACTOR	STAT_PROFIT/STAT_	TesterStatistics

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	BALANCE_DD の比に等しい回復率	
STAT_SHARPE_RATIO	シャープレシオ。	TesterStatistics
STAT_SHORT_TRADES	短期取引。	TesterStatistics
STAT_TRADES	取引数。	TesterStatistics
STAT_WITHDRAWAL	口座から引き落とされたお金。	TesterStatistics
STO_CLOSECLOSE	計算は、クローズ（ローソク足の形成が終わった時点での価格）/終値に基づいています。	価格定数
STO_LOWHIGH	計算は、低/高価格に基づいています。	価格定数

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
STYLE_DASH	途切れ線。	描画スタイル
STYLE_DASHDOT	一点鎖線。	描画スタイル
STYLE_DASHDOTDOT	二点鎖線。	描画スタイル
STYLE_DOT	点線。	描画スタイル
STYLE_SOLID	実線。	描画スタイル
SUNDAY	日曜。	SymbolInfoInteger、SymbolInfoSessionQuote、SymbolInfoSessionTrade
SYMBOL_ASK	買値 – ベストバイオファー。	SymbolInfoDouble
SYMBOL_ASKHIGH	一日の最高買値。	SymbolInfoDouble
SYMBOL_ASKLOW	一日の最低買値。	SymbolInfoDouble
SYMBOL_BANK	現在の相場のフィーダー。	SymbolInfoString
SYMBOL_BASIS	デリバティブの原資産。	SymbolInfoString
SYMBOL_BID	売値 – ベストセル	SymbolInfoDouble

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	オファー。	
SYMBOL_BIDHIGH	一日の最高売値。	SymbolInfoDouble
SYMBOL_BIDLOW	一日の最低売値。	SymbolInfoDouble
SYMBOL_CALC_MODE_CFD	CFDモード - CFDの証拠金と利益計算。	SymbolInfoInteger
SYMBOL_CALC_MODE_CFDINDEX	CFDインデックスモード – インデックスを使用したCFDの証拠金と利益計算	SymbolInfoInteger
SYMBOL_CALC_MODE_CFDLEVERAGE	CFDレバレッジモード – レバレッジ取引でのCFDの証拠金と利	SymbolInfoInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	益計算	
SYMBOL_CALC_MODE_EXCH_FUTURES	先物モード - 証券取引所での取引先物契約の証拠金と利益計算。	SymbolInfoInteger
SYMBOL_CALC_MODE_EXCH_FUTURES_FORTS	FORTS 先物モード - FORTS での取引先物契約の証拠金と利益計算。	SymbolInfoInteger
SYMBOL_CALC_MODE_EXCH_STOCKS	交換モード - 証券取引所で有価証券を取引するための証拠金と利益計算。	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

定数	説明	使用例
SYMBOL_CALC_MODE_FOREX	外為モード - 外国為替の利益と証拠金の計算。	SymbolInfoInteger
SYMBOL_CALC_MODE_FUTURES	先物モード - 先物の証拠金と利益の計算。	SymbolInfoInteger
SYMBOL_CALC_MODE_SERV_COLLATERAL	担保モード - シンボルは、取引口座の取引不可能の資産として使用されています。	SymbolInfoInteger
SYMBOL_CURRENCY_BASE	シンボルの基本通貨。	SymbolInfoString
SYMBOL_CURRENCY_MARGIN	証拠金通貨。	SymbolInfoString
SYMBOL_CURRENCY_PROFIT	利益通貨。	SymbolInfoString

© 2000-2025, MetaQuotes Ltd.

定数	説明	使用例
SYMBOL_DESCRIPTION	シンボル説明。	SymbolInfoString
SYMBOL_DIGITS	小数点以下の桁数。	SymbolInfoInteger
SYMBOL_EXPIRATION_DAY	注文は日の終わりまで有効です。	SymbolInfoInteger
SYMBOL_EXPIRATION_GTC	注文は明示的にキャンセルされるまで、無制限の期間中有効です。	SymbolInfoInteger
SYMBOL_EXPIRATION_MODE	注文期限切れモードの可能なフラグ。	SymbolInfoInteger
SYMBOL_EXPIRATION_SPECIFIED	有効期限の時刻が注文に指定されています。	SymbolInfoInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
SYMBOL_EXPIRATION_SPECIFIED_DAY	有効期限の日付が注文に指定されています。	SymbolInfoInteger
SYMBOL_EXPIRATION_TIME	シンボル取引終了の日（通常は先物取引に使用）。	SymbolInfoInteger
SYMBOL_FILLING_FOK	このポリシーは、約定は、必ず指定のボリュームで実行することが出来ることを意味します。金融商品の必要量は、市場で現在利用出来な	SymbolInfoInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	い場合、注文は実行されません。必要なボリュームは、市場で現在利用可能ないくつかのオファーを使用して満たすことが出来ます。	
SYMBOL_FILLING_IOC	この場合、トレーダーが注文の選択肢のうちで、市場で最大限に利用可能なボリュームの約定実行に同意します。	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

定数	説明	使用例
	注文の全てを満たすことが出来ない場合、利用可能なボリュームのみが満たされ、残りのボリュームはキャンセルされます。IOC注文を使用する可能性は取引サーバで決定されます。	
SYMBOL_FILLING_MODE	注文充填モードの可能なフラグ。	SymbolInfoInteger
SYMBOL_ISIN	ISINシステム（Inte	SymbolInfoString

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	rnational Securities Identification Number ）での銘柄名。International Securities Identification Number はセキュリティを識別する一意の12桁の英数字コードです。このシンボルプロパティの存在は取引サーバ側で決定されます。	
SYMBOL_LAST	直近約定値。	SymbolInfoDouble

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
SYMBOL_LASTHIGH	一日の最高直近価格。	SymbolInfoDouble
SYMBOL_LASTLOW	一日の最低直近価格。	SymbolInfoDouble
SYMBOL_MARGIN_INITIAL	当初証拠金とは、1ロットのボリュームで注文を出すのに必要な証拠金の証拠金通貨での量です。これは、クライアントが市場に入った時に、資産をチェックするために使用されます。	SymbolInfoDouble
SYMBOL_MARGIN_MAINTENANCE	維持証拠	SymbolInfoDouble

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	金。設定されている場合、1ロットから課金される、シンボルの証拠金通貨での証拠金量を設定します。これは、クライアントの口座状態が変わった時に、資産をチェックするために使用されます。維持証拠金が0に等しい場合、当初証拠金が使用	

© 2000-2025, MetaQuotes Ltd.

定数	説明	使用例
	されます。	
SYMBOL_OPTION_MODE	オプションの種類。	SymbolInfoInteger
SYMBOL_OPTION_MODE_EUROPEAN	ヨーロッパのオプションは、指定された日付（有効期限、実行日、配達日）にのみ行使出来ます。	SymbolInfoInteger
SYMBOL_OPTION_MODE_AMERICAN	アメリカのオプションは満了前の任意の取引日に行使することが出来ます。買い手がオプションを行使すること	SymbolInfoInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	が出来る期間は指定されています。	
SYMBOL_OPTION_RIGHT	オプション特権（買い/売り）。	SymbolInfoInteger
SYMBOL_OPTION_RIGHT_CALL	コール・オプションを使用すると、指定された価格で資産を買うことが出来ます。	SymbolInfoInteger
SYMBOL_OPTION_RIGHT_PUT	プット・オプションを使用すると、指定された価格で資産を売ることが出来ます。	SymbolInfoInteger
SYMBOL_OPTION_STRIKE	オプションの行	SymbolInfoDouble

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	使価格。オプションの買い手が原資産を（コールオプションで）購入したり、（プットオプションで）販売することが出来、オプションの売り手が原資産の適切な量を売買する義務がある価格。	
SYMBOL_ORDER_LIMIT	指値注文（買い指値と売り指値）は使用可能です。	SymbolInfoInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
SYMBOL_ORDER_MARKET	成行注文は売買とも使用可能です。	SymbolInfoInteger
SYMBOL_ORDER_MODE	注文の種類に使用可能なフラグ。	SymbolInfoInteger
SYMBOL_ORDER_SL	逆指注文は可能です。	SymbolInfoInteger
SYMBOL_ORDER_STOP	逆指注文（売買）は使用可能です。	SymbolInfoInteger
SYMBOL_ORDER_STOP_LIMIT	指値つきストップ注文は売買とも使用可能です。	SymbolInfoInteger
SYMBOL_ORDER_TP	決済指は使用可能です。	SymbolInfoInteger
SYMBOL_PATH	シンボルツリーパス。	SymbolInfoString

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
SYMBOL_POINT	シンボルのポイント値。	SymbolInfoDouble
SYMBOL_SELECT	「気配値表示」でシンボルが選択されています。	SymbolInfoInteger
SYMBOL_SESSION_AW	現在のセッションの加重平均価格。	SymbolInfoDouble
SYMBOL_SESSION_BUY_ORDERS	現時点での買い注文の数。	SymbolInfoInteger
SYMBOL_SESSION_BUY_ORDERS_VOLUME	現在の買い注文のボリューム。	SymbolInfoDouble
SYMBOL_SESSION_CLOSE	現在のセッションの終値。	SymbolInfoDouble
SYMBOL_SESSION_DEALS	現在のセッションの約定数。	SymbolInfoInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
SYMBOL_SESSION_INTEREST	建玉概要。	SymbolInfoDouble
SYMBOL_SESSION_OPEN	現在のセッションの始値。	SymbolInfoDouble
SYMBOL_SESSION_PRICE_LIMIT_MAX	現在のセッションの最高値。	SymbolInfoDouble
SYMBOL_SESSION_PRICE_LIMIT_MIN	現在のセッションの最安値。	SymbolInfoDouble
SYMBOL_SESSION_PRICE_SETTLEMENT	現在のセッションの決済価格。	SymbolInfoDouble
SYMBOL_SESSION_SELL_ORDERS	現時点での売り注文の数。	SymbolInfoInteger
SYMBOL_SESSION_SELL_ORDERS_VOLUME	現在の売り注文のボリューム。	SymbolInfoDouble
SYMBOL_SESSION_TURNOVER	現在のセッションの概要売上高。	SymbolInfoDouble

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
SYMBOL_SESSION_VOLUME	現在のセッション約定ボリュームの概要。	SymbolInfoDouble
SYMBOL_SPREAD	ポイント単位でのスプレッド値。	SymbolInfoInteger
SYMBOL_SPREAD_FLOAT	変動スプレッドの表示。	SymbolInfoInteger
SYMBOL_START_TIME	シンボル取引開始の日（通常は先物取引に使用）	SymbolInfoInteger
SYMBOL_SWAP_LONG	長期スワップ値。	SymbolInfoDouble
SYMBOL_SWAP_MODE	スワップ計算モデル。	SymbolInfoInteger
SYMBOL_SWAP_MODE_CURRENCY_DEPOSIT	スワップは、クライアントの預金通貨で請求	SymbolInfoInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	されます。	
SYMBOL_SWAP_MODE_CURRENCY_MARGIN	スワップはシンボルの証拠金通貨で払われます。	SymbolInfoInteger
SYMBOL_SWAP_MODE_CURRENCY_SYMBOL	スワップはシンボルの基本通貨で払われます。	SymbolInfoInteger
SYMBOL_SWAP_MODE_DISABLED	スワップ無効。	SymbolInfoInteger
SYMBOL_SWAP_MODE_INTEREST_CURRENT	スワップはスワップの計算の商品価格の指定年利として請求されます（標準銀行年は360日です）。	SymbolInfoInteger
SYMBOL_SWAP_MODE_INTEREST_OPEN	スワップはポジションの	SymbolInfoInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	始値から指定年利として請求されます（標準銀行年は360日です）。	
SYMBOL_SWAP_MODE_POINTS	スワップはポイントとして請求されます。	SymbolInfoInteger
SYMBOL_SWAP_MODE_REOPEN_BID	スワップはポジションを再オープンすることによって請求されます。ポジションは取引日の終わりに閉じます。次の日に、現在の売値+/-指定されたポイン	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

定数	説明	使用例
	トの数（SYMBOL_SWAP_LONG及びSYMBOL_SWAP_SHORT パラメータ）で開けられます。	
SYMBOL_SWAP_MODE_REOPEN_CURRENT	スワップはポジションを再オープンすることによって請求されます。ポジションは取引日の終わりに閉じます。次の日に、終値 +/- 指定されたポイントの数（パラメータ SYMB	SymbolInfoInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	OL_SWAP_LONG及びSYMBOL_SWAP_SHORT)で開けられます。	
SYMBOL_SWAP_ROLLOVER3DAYS	Day of week to charge 3 days swap rollover	SymbolInfoInteger
SYMBOL_SWAP_SHORT	短期スワップ値。	SymbolInfoDouble
SYMBOL_TICKS_BOOKDEPTH	板情報で表示されるリクエストの最高数。要求のないキューを持っていないシンボルの場合、値はゼロに等しいです。	SymbolInfoInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
SYMBOL_TIME	最終の相場の時刻。	SymbolInfoInteger
SYMBOL_TRADE_CALC_MODE	契約価格計算モード。	SymbolInfoInteger
SYMBOL_TRADE_CONTRACT_SIZE	取引契約サイズ。	SymbolInfoDouble
SYMBOL_TRADE_EXECUTION_EXCHANGE	エクスチェンジ実行。	SymbolInfoInteger
SYMBOL_TRADE_EXECUTION_INSTANT	即時実行。	SymbolInfoInteger
SYMBOL_TRADE_EXECUTION_MARKET	マーケット実行。	SymbolInfoInteger
SYMBOL_TRADE_EXECUTION_REQUEST	リクエストによる実行。	SymbolInfoInteger
SYMBOL_TRADE_EXEMODE	約定実行モード。	SymbolInfoInteger
SYMBOL_TRADE_FREEZE_LEVEL	取引業務を凍結までの距離（ポイント単位）。	SymbolInfoInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
SYMBOL_TRADE_MODE	注文実行の種類。	SymbolInfoInteger
SYMBOL_TRADE_MODE_CLOSEONLY	ポジション決済操作のみを許可。	SymbolInfoInteger
SYMBOL_TRADE_MODE_DISABLED	シンボルの取引が無効。	SymbolInfoInteger
SYMBOL_TRADE_MODE_FULL	取引制限なし。	SymbolInfoInteger
SYMBOL_TRADE_MODE_LONGONLY	ロング（買い）ポジションのみ許可。	SymbolInfoInteger
SYMBOL_TRADE_MODE_SHORTONLY	ショート（売り）ポジションのみ許可。	SymbolInfoInteger
SYMBOL_TRADE_STOPS_LEVEL	現在の終値から逆指値注文を配置する場合の最少のインデ	SymbolInfoInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	ント（ポイント単位）。	
SYMBOL_TRADE_TICK_SIZE	最少の価格変化。	SymbolInfoDouble
SYMBOL_TRADE_TICK_VALUE	SYMBOL_TRADE_TICK_VALUE_PROFITの値。	SymbolInfoDouble
SYMBOL_TRADE_TICK_VALUE_LOSS	負けポジションの計算されたティック価格。	SymbolInfoDouble
SYMBOL_TRADE_TICK_VALUE_PROFIT	収益性の高いポジションの計算されたティック価格。	SymbolInfoDouble
SYMBOL_VOLUME	最終約定ボリューム。	SymbolInfoInteger
SYMBOL_VOLUME_LIMIT	シンボルの片方向（売ま	SymbolInfoDouble

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	たは買）のオープンポジションと未決注文の最大許容総額。例えば、制限が 5 ロットの場合、5 ロットのボリュームでオープン買いポジションを持つことができ、5 ロットのボリュームで、売り指値注文をすることが出来ます。しかし、この場合には、1 方向の総	

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	ボリュームが制限を超えてしまうため、買い指値注文をすることや 5 ロット以上の売り指値注文をすることは出来ません。	
SYMBOL_VOLUME_MAX	約定の最大ボリューム。	SymbolInfoDouble
SYMBOL_VOLUME_MIN	約定の最小ボリューム。	SymbolInfoDouble
SYMBOL_VOLUME_STEP	約定実行のための最小限のボリューム変化のステップ。	SymbolInfoDouble
SYMBOL_VOLUMEHIGH	一日の最大ボ	SymbolInfoInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	リューム。	
SYMBOL_VOLUMELOW	一日の最小ボリューム。	SymbolInfoInteger
TENKANSEN_LINE	転換線。	指標線
TERMINAL_BUILD	クライアント端末ビルド番号。	TerminalInfoInteger
TERMINAL_CODEPAGE	クライアント端末にインストールされた言語のコードページ。	TerminalInfoInteger
TERMINAL_COMMONDATA_PATH	コンピュータにインストールされている全ての端末の共通パス。	TerminalInfoString
TERMINAL_COMMUNITY_ACCOUNT	フラグは、端末内のMQL5.community 認	TerminalInfoInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	証データの存在を示します。	
TERMINAL_COMMUNITY_BALANCE	MQL5.community の残高。	TerminalInfoDouble
TERMINAL_COMMUNITY_CONNECTION	MQL5.community への接続。	TerminalInfoInteger
TERMINAL_COMPANY	会社名。	TerminalInfoString
TERMINAL_CONNECTED	取引サーバへの接続。	TerminalInfoInteger
TERMINAL_CPU_CORES	システム CPU コアの数。	TerminalInfoInteger
TERMINAL_DATA_PATH	端末データが格納されるフォルダ。	TerminalInfoString
TERMINAL_DISK_SPACE	端末（エージェント）の MQL5\Files フォルダの空き	TerminalInfoInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	ディスク容量（MB単位）。	
TERMINAL_DLLS_ALLOWED	DLLを使用する許可。	TerminalInfoInteger
TERMINAL_EMAIL_ENABLED	端末の設定で指定されたSMTPサーバへの電子メールの送信及びログインの許可。	TerminalInfoInteger
TERMINAL_FTP_ENABLED	端末の設定で指定されたFTPサーバへのレポート送信及びログインの許可。	TerminalInfoInteger
TERMINAL_LANGUAGE	端末の言語。	TerminalInfoString

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
TERMINAL_MAXBARS	チャートのバーの最大数。	TerminalInfoInteger
TERMINAL_MEMORY_AVAILABLE	端末（エージェント）プロセスの空きメモリ（MB単位）。	TerminalInfoInteger
TERMINAL_MEMORY_PHYSICAL	システムの物理メモリ（MB単位）。	TerminalInfoInteger
TERMINAL_MEMORY_TOTAL	端末のプロセス（エージェント）に使用可能なメモリ（MB単位）。	TerminalInfoInteger
TERMINAL_MEMORY_USED	端末（エージェント）で使用されるメモリ（MB単位）。	TerminalInfoInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
TERMINAL_MQID	このフラグはプッシュ通知のMetaQuotes ID データが存在することを示します。	TerminalInfoInteger
TERMINAL_NAME	端末名。	TerminalInfoString
TERMINAL_NOTIFICATIONS_ENABLED	スマートフォンに通知を送信するためのアクセス許可。	TerminalInfoInteger
TERMINAL_OPENCL_SUPPORT	0x00010002 = 1.2形式でのサポートされているOpenCL のバージョン。「0」はOpenCL をサポートがないこと	TerminalInfoInteger

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	を意味します。	
TERMINAL_PATH	端末が起動されるフォルダ。	TerminalInfoString
TERMINAL_PING_LAST	取引サーバまでの既知のping値は、マイクロ秒単位です。1秒は100万マイクロ秒です。	TerminalInfoInteger
TERMINAL_SCREEN_DPI	画像の情報出力解像度は、DPIで測定されます。	TerminalInfoInteger
TERMINAL_TRADE_ALLOWED	約定許可。	TerminalInfoInteger
TERMINAL_X64	「64ビット端末」の表示。	TerminalInfoInteger
THURSDAY	木曜。	SymbolInfoInteger、SymbolInfoSessionQuote、

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
		SymbolInfoSessionTrade
TRADE_ACTION_DEAL	指定されたパラメータ（成行注文）の即時実行のために約定注文を出します。	MqlTradeRequest
TRADE_ACTION_MODIFY	以前の注文のパラメータを変更します。	MqlTradeRequest
TRADE_ACTION_PENDING	指定された条件（未決注文）で実行するために取引注文を出します。	MqlTradeRequest
TRADE_ACTION_REMOVE	以前の未決注文を削除します。	MqlTradeRequest

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
TRADE_ACTION_SLTP	保有中ポジションの決済逆指値及び決済指値を変更します。	MqlTradeRequest
TRADE_RETCODE_CANCEL	トレーダーによるリクエストのキャンセル。	MqlTradeResult
TRADE_RETCODE_CLIENT_DISABLES_AT	クライアント端末が自動取引を無効化。	MqlTradeResult
TRADE_RETCODE_CONNECTION	取引サーバに未接続。	MqlTradeResult
TRADE_RETCODE_DONE	リクエスト完了。	MqlTradeResult
TRADE_RETCODE_DONE_PARTIAL	リクエストが一部のみ完了。	MqlTradeResult
TRADE_RETCODE_ERROR	リクエスト処	MqlTradeResult

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	理エラー。	
TRADE_RETCODE_FROZEN	注文やポジションが凍結。	MqlTradeResult
TRADE_RETCODE_INVALID	無効なリクエスト。	MqlTradeResult
TRADE_RETCODE_INVALID_EXPIRATION	リクエスト内の無効な注文有効期限。	MqlTradeResult
TRADE_RETCODE_INVALID_FILL	無効な注文充填タイプ。	MqlTradeResult
TRADE_RETCODE_INVALID_ORDER	不正または禁止された注文の種類。	MqlTradeResult
TRADE_RETCODE_INVALID_PRICE	リクエスト内の無効な価格。	MqlTradeResult
TRADE_RETCODE_INVALID_STOPS	リクエスト内の無効なストップ。	MqlTradeResult

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
TRADE_RETCODE_INVALID_VOLUME	リクエスト内の無効なボリューム。	MqlTradeResult
TRADE_RETCODE_LIMIT_ORDERS	未決注文の数が上限に達しました。	MqlTradeResult
TRADE_RETCODE_LIMIT_VOLUME	シンボルの注文やポジションのボリュームが限界に達しました。	MqlTradeResult
TRADE_RETCODE_LOCKED	リクエストが処理のためにロック中。	MqlTradeResult
TRADE_RETCODE_MARKET_CLOSED	市場が閉鎖中。	MqlTradeResult
TRADE_RETCODE_NO_CHANGES	リクエストに変更なし。	MqlTradeResult
TRADE_RETCODE_NO_MONEY	リクエストを完了するのに資金が	MqlTradeResult

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	不充分。	
TRADE_RETCODE_ONLY_REAL	操作は、ライブ口座のみで許可。	MqlTradeResult
TRADE_RETCODE_ORDER_CHANGED	注文状態の変化。	MqlTradeResult
TRADE_RETCODE_PLACED	注文が出されました。	MqlTradeResult
TRADE_RETCODE_POSITION_CLOSED	指定された POSITION_IDENTIFIER を持つポジションがすでに閉鎖。	MqlTradeResult
TRADE_RETCODE_PRICE_CHANGED	価格変更。	MqlTradeResult
TRADE_RETCODE_PRICE_OFF	リクエスト処理に必要な相場が不在。	MqlTradeResult
TRADE_RETCODE_REJECT	リクエストの拒否。	MqlTradeResult

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
TRADE_RETCODE_REQUOTE	リクオート。	MqlTradeResult
TRADE_RETCODE_SERVER_DISABLES_AT	サーバが自動取引を無効化。	MqlTradeResult
TRADE_RETCODE_TIMEOUT	リクエストが時間切れでキャンセル。	MqlTradeResult
TRADE_RETCODE_TOO_MANY_REQUESTS	頻繁過ぎるリクエスト。	MqlTradeResult
TRADE_RETCODE_TRADE_DISABLED	取引が無効化されています。	MqlTradeResult
TRADE_TRANSACTION_DEAL_ADD	約定を履歴に追加。このアクションは注文実行や口座残高の操作の結果として実行されます。	MqlTradeTransaction

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
TRADE_TRANSACTION_DEAL_DELETE	約定履歴の削除。以前に実行された約定がサーバから削除される場合があります。例えば、約定が以前にブローカによって転送された外部の取引システム（exchange）で削除されている場合です。	MqlTradeTransaction
TRADE_TRANSACTION_DEAL_UPDATE	約定を履歴での更新。以前に実行された約定	MqlTradeTransaction

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	がサーバで更新される場合があります。例えば、約定が以前にブローカによって転送された外部の取引システム（exchange）で更新される場合です。	
TRADE_TRANSACTION_HISTORY_ADD	実行またはキャンセルの結果として履歴に注文を追加します。	MqlTradeTransaction
TRADE_TRANSACTION_HISTORY_DELETE	注文を履歴から削除します。この型	MqlTradeTransaction

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	は、取引サーバ側の機能性を高めるために設けられています。	
TRADE_TRANSACTION_HISTORY_UPDATE	注文履歴内の注文を変更します。この型は、取引サーバ側の機能性を高めるために設けられています。	MqlTradeTransaction
TRADE_TRANSACTION_ORDER_ADD	新規の未執行注文の追加。	MqlTradeTransaction
TRADE_TRANSACTION_ORDER_DELETE	注文の未処理注文リストからの削除。注文	MqlTradeTransaction

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	は、適切なリクエストの設定または実行（充填）後、履歴に移動された結果として、未処理の注文のリストから削除することが出来ます。	
TRADE_TRANSACTION_ORDER_UPDATE	未執行注文の更新。アップデートは、クライアント端末や取引サーバ側からの明らかな変更だけでなく、設定	MqlTradeTransaction

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	時の注文状態の変化を含みます（例えばORDER_STATE_STARTEDからORDER_STATE_PLACEDへやORDER_STATE_PLACEDからORDER_STATE_PARTIALへの変化など）。	
TRADE_TRANSACTION_POSITION	取引の実行に関連していないポジションを変更します。このトランザクション型はポジ	MqlTradeTransaction

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	ションが取引サーバ側で変更されたことを示しています。ポジションボリューム、始値、決済逆指及び決済指レベルの変更が可能です。返られたデータはOnTradeTransaction ハンドラで MqlTradeTransaction 構造体に提示されます。約定実行によるポジション	

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	の変化（追加、変更、閉鎖）はTRADE_TRANSACTION_POSITIONトランザクションの発生にはつながりません。	
TRADE_TRANSACTION_REQUEST	取引リクエストがサーバと処理され結果が受け取られたことの通知。そのような取引ではMqlTradeTransaction構造体のtypeフィー	MqlTradeTransaction

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	ルド（取引タイプ）の分析のみが必要です。OnTradeTransaction 関数の2、3番目のパラメータ（request と result）は追加のデータを得るために分析されなければいけません。	
TUESDAY	火曜。	SymbolInfoInteger、SymbolInfoSessionQuote、SymbolInfoSessionTrade
TYPE_BOOL	bool	MqlParam
TYPE_CHAR	char	MqlParam
TYPE_COLOR	color	MqlParam
TYPE_DATETIME	datetime	MqlParam
TYPE_DOUBLE	double	MqlParam
TYPE_FLOAT	float	MqlParam
TYPE_INT	int	MqlParam

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
TYPE_LONG	long	MqlParam
TYPE_SHORT	short	MqlParam
TYPE_STRING	string	MqlParam
TYPE_UCHAR	uchar	MqlParam
TYPE_UINT	uint	MqlParam
TYPE_ULONG	ulong	MqlParam
TYPE_USHORT	ushort	MqlParam
UCHAR_MAX	uchar型で表すことが出来る最大値。。	数値定数
UINT_MAX	uint型で表すことが出来る最大値。。	数値定数
ULONG_MAX	ulong型で表すことが出来る最大値。。	数値定数
UPPER_BAND	上限。	指標線
UPPER_HISTOGRAM	上部ヒストグラム。	指標線
UPPER_LINE	上部線。	指標線
USHORT_MAX	ushort型で	数値定数

© 2000–2025, MetaQuotes Ltd.

定数	説明	使用例
	表すことが出来る最大値。。	
VOLUME_REAL	取引高。	価格定数
VOLUME_TICK	ティックボリューム。	価格定数
WEDNESDAY	水曜。	SymbolInfoInteger、SymbolInfoSessionQuote、SymbolInfoSessionTrade
WHOLE_ARRAY	配列の最後までに残っている要素の数を表します。配列全体が処理されます。	その他の定数
WRONG_VALUE	定数は任意の列挙型に暗黙的にキャストされることが出来ます。	その他の定数

© 2000–2025, MetaQuotes Ltd.