```
# Hello World project
# python 3
# برنامه مک دی برای متاتریدر5و4
# برنامه ای مفید برای معامله دربورس جهانی
#  MACD program for Metatrader 5 and 4
# A useful program for the global stock market
# فایل فوق رودر ام کیو ال 5 کپی کنید


/*
* Place the SmoothAlgorithms.mqh file
* to the terminal_data_folder\MQL5\Include
*/
//+------------------------------------------
------------------+
//|
ColorMACD_Histogram.mq5 |
//|                                  Copyright © 2011,
Nikolay Kositsin |
//|                                 Khabarovsk,
farria@mail.redcom.ru |
//+------------------------------------------
------------------+
#property copyright "Copyright © 2011, Nikolay
Kositsin"
#property link "farria@mail.redcom.ru"
//---- indicator version
#property version   "1.00"
//---- drawing the indicator in a separate window
```

```
#property indicator_separate_window
//---- number of indicator buffers 6
#property indicator_buffers 6
//---- only five plots are used
#property indicator_plots   5

//+-----------------------------------------------+
//|  Bullish indicator drawing parameters         |
//+-----------------------------------------------+
//---- drawing the indicator 1 as a symbol
#property indicator_type1   DRAW_ARROW
//---- lime color is used for the indicator
#property indicator_color1  Lime
//---- thickness of the indicator 1 line is equal
to 1
#property indicator_width1  1
//---- displaying the indicator label
#property indicator_label1 "Buy"

//+-----------------------------------------------+
//|  Bearish indicator drawing parameters         |
//+-----------------------------------------------+
//---- drawing the indicator 2 as a symbol
#property indicator_type2   DRAW_ARROW
//---- magenta color is used for the indicator
#property indicator_color2  Magenta
//---- thickness of the indicator 2 line is equal
to 1
#property indicator_width2  1
//---- displaying the indicator label
```

```
#property indicator_label2  "Sell"

//+-----------------------------------+
//|  Indicator drawing parameters     |
//+-----------------------------------+
//---- drawing the indicator as a line
#property indicator_type3 DRAW_LINE
//---- use dodger blue color for the line
#property indicator_color3 DodgerBlue
//---- indicator line is a solid one
#property indicator_style3 STYLE_SOLID
//---- the width of the indicator line is 3
#property indicator_width3 2
//---- displaying the indicator line label
#property indicator_label3 "MACD"

//---- drawing the indicator as a line
#property indicator_type4 DRAW_LINE
//---- use red color for the line
#property indicator_color4 Red
//---- the indicator line is a dash-dotted curve
#property indicator_style4 STYLE_DASHDOTDOT
//---- indicator line width is equal to 1
#property indicator_width4 1
//---- displaying of the indicator signal line
label
#property indicator_label4  "Signal Line"

//---- drawing the indicator as a histogram
#property indicator_type5 DRAW_COLOR_HISTOGRAM
```

```
//---- the following colors are used in the colored
histogram
#property indicator_color5
Gray,Teal,DarkViolet,IndianRed,Magenta
//---- the indicator line is a dash-dotted curve
#property indicator_style5 STYLE_SOLID
//---- indicator line width is equal to 2
#property indicator_width5 2
//---- displaying the indicator line label
#property indicator_label5  "MACD Histogram"

//----
#define arrowsDisplacement 0.0001
//+-----------------------------------+
//|  Indicator input parameters       |
//+-----------------------------------+
enum Applied_price_ //Type of constant
  {
   PRICE_CLOSE_ = 1,     //Close
   PRICE_OPEN_,          //Open
   PRICE_HIGH_,          //High
   PRICE_LOW_,           //Low
   PRICE_MEDIAN_,        //Median Price (HL/2)
   PRICE_TYPICAL_,       //Typical Price (HLC/3)
   PRICE_WEIGHTED_,      //Weighted Close (HLCC/4)
   PRICE_SIMPLE,         //Simple Price (OC/2)
   PRICE_QUARTER_,       //Quarted Price (HLOC/4)
   PRICE_TRENDFOLLOW0_,  //TrendFollow_1 Price
   PRICE_TRENDFOLLOW1_   //TrendFollow_2 Price
  };
```

```
input int Fast_MA = 12;                                  //
Fast MA period
input int Slow_MA = 26;                                  //
SMMA smoothing depth
input ENUM_MA_METHOD MA_Method_=MODE_EMA;                //
Indicator smoothing method
input int Signal_SMA=9;                                  //
Signal line period
input Applied_price_ AppliedPrice=PRICE_CLOSE_; //
Price constant
/*used for calculation of the indicator ( 1-CLOSE,
2-OPEN, 3-HIGH, 4-LOW,
  5-MEDIAN, 6-TYPICAL, 7-WEIGHTED, 8-SIMPLE,
9-QUARTER, 10-TRENDFOLLOW, 11-0.5 * TRENDFOLLOW.)
*/
input bool drawIndicatorTrendLines=true;
input bool drawPriceTrendLines=true;
input bool displayAlert=true;
input color BulliDiverColor=Lime;
input color BearDiverColor=Red;
//+------------------------------------+

datetime lastAlerttime;
string   indicatorName;
//---- Declaration of the integer variables for the
start of data calculation
int start,macd_start=0;
//---- declaration of dynamic arrays that
// will be used as indicator buffers
double MACDLineBuffer[];
```

```
double SignalLineBuffer[];
double HistogramBuffer[];
double bullishDivergence[];
double bearishDivergence[];
double ColorHistogramBuffer[];
//+------------------------------------------------------------------+
// iPriceSeries function description
                    |
// Moving_Average class description
                    |
//+------------------------------------------------------------------+
#include <SmoothAlgorithms.mqh>
//+------------------------------------------------------------------+
//| MACD indicator initialization function
                    |
//+------------------------------------------------------------------+
void OnInit()
  {
//---- Initialization of variables of the start of data calculation
   if(MA_Method_!=MODE_EMA)
macd_start=MathMax(Fast_MA,Slow_MA);
   start=macd_start+Signal_SMA+1;
//---- set dynamic array as an indicator buffer

SetIndexBuffer(0,bullishDivergence,INDICATOR_DATA);
```

```
//---- shifting the start of drawing of the
indicator 4
   PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,start);
//---- create a label to display in DataWindow
   PlotIndexSetString(0,PLOT_LABEL,"Buy");
//---- indicator symbol
   PlotIndexSetInteger(0,PLOT_ARROW,233);
//---- indexing elements in the buffer as
timeseries
   ArraySetAsSeries(bullishDivergence,true);
//---- restriction to draw empty values for the
indicator

PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,EMPTY_VALUE);

//---- set bearishDivergence[] dynamic array as an
indicator buffer

SetIndexBuffer(1,bearishDivergence,INDICATOR_DATA);
//---- shifting the start of drawing of the
indicator 5
   PlotIndexSetInteger(1,PLOT_DRAW_BEGIN,start);
//---- create a label to display in DataWindow
   PlotIndexSetString(1,PLOT_LABEL,"Sell");
//---- indicator symbol
   PlotIndexSetInteger(1,PLOT_ARROW,234);
//---- indexing elements in the buffer as
timeseries
   ArraySetAsSeries(bearishDivergence,true);
//---- restriction to draw empty values for the
```

```
indicator

PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,EMPTY_VALUE);

//---- set MACDLineBuffer[] dynamic array as an
indicator buffer
   SetIndexBuffer(2,MACDLineBuffer,INDICATOR_DATA);
//---- performing the shift of beginning of the
indicator drawing

PlotIndexSetInteger(2,PLOT_DRAW_BEGIN,macd_start);
//---- create a label to display in DataWindow
   PlotIndexSetString(2,PLOT_LABEL,"MACD");
//---- setting values of the indicator that won't
be visible on a chart

PlotIndexSetDouble(2,PLOT_EMPTY_VALUE,EMPTY_VALUE);
//---- indexing elements in the buffer as
timeseries
   ArraySetAsSeries(MACDLineBuffer,true);

//---- set SignalLineBuffer[] dynamic array as an
indicator buffer

SetIndexBuffer(3,SignalLineBuffer,INDICATOR_DATA);
//---- performing the shift of beginning of the
indicator drawing
   PlotIndexSetInteger(3,PLOT_DRAW_BEGIN,start);
//---- create a label to display in DataWindow
   PlotIndexSetString(3,PLOT_LABEL,"Signal MA");
```

```
//---- setting values of the indicator that won't
be visible on a chart

PlotIndexSetDouble(3,PLOT_EMPTY_VALUE,EMPTY_VALUE);
//---- indexing elements in the buffer as
timeseries
   ArraySetAsSeries(SignalLineBuffer,true);

//---- set HistogramBuffer[] dynamic array as an
indicator buffer

SetIndexBuffer(4,HistogramBuffer,INDICATOR_DATA);
//---- performing the shift of beginning of the
indicator drawing
   PlotIndexSetInteger(4,PLOT_DRAW_BEGIN,start);
//---- create a label to display in DataWindow
   PlotIndexSetString(4,PLOT_LABEL,"Histogram");
//---- setting values of the indicator that won't
be visible on a chart

PlotIndexSetDouble(4,PLOT_EMPTY_VALUE,EMPTY_VALUE);
//---- indexing elements in the buffer as
timeseries
   ArraySetAsSeries(HistogramBuffer,true);

//---- set ColorHistogramBuffer[] dynamic array as
a colored index buffer

SetIndexBuffer(5,ColorHistogramBuffer,INDICATOR_COL
OR_INDEX);
```

```
//---- performing the shift of beginning of the
indicator drawing
   PlotIndexSetInteger(5,PLOT_DRAW_BEGIN,start+1);
//---- indexing elements in the buffer as
timeseries
   ArraySetAsSeries(ColorHistogramBuffer,true);

//---- initializations of a variable for the
indicator short name
   StringConcatenate(indicatorName,"MACD(
",Fast_MA,", ",Slow_MA,", ",Signal_SMA," )");
//---- creation of the name to be displayed in a
separate sub-window and in a tooltip

IndicatorSetString(INDICATOR_SHORTNAME,indicatorNam
e);
//---- determination of accuracy of displaying of
the indicator values
   IndicatorSetInteger(INDICATOR_DIGITS,_Digits+1);
//---- initialization end
  }
//+------------------------------------------
------------------+
//| Custom indicator deinitialization function
                  |
//+------------------------------------------
------------------+
void OnDeinit(const int reason)
  {
//----
```

```
   string Name;
   for(int obj=ObjectsTotal(0,-1,-1)-1; obj>=0;
obj--)
     {
      Name=ObjectName(0,obj,-1,-1);

if(StringSubstr(Name,0,21)=="MACD_DivergenceLine.0"
) ObjectDelete(0,Name);
     }
//----
  }
//+------------------------------------------------
------------------+
//| MACD iteration function
                  |
//+------------------------------------------------
------------------+
int OnCalculate(const int rates_total,    // number
of bars in history at the current tick
                const int prev_calculated,// number
of bars calculated at previous call
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
```

```
//---- checking the number of bars to be enough for
the calculation
   if(rates_total<start) return(0);

//---- declaration of integer variables
   int
MaxBar1,MaxBar2,MaxBar3,limit1,limit2,limit3,bar;
//---- declaration of variables with a floating
point
   double price_,fast_ma,slow_ma;

//---- indexing elements in arrays as timeseries
   ArraySetAsSeries(time,true);
   ArraySetAsSeries(open,true);
   ArraySetAsSeries(high,true);
   ArraySetAsSeries(low,true);
   ArraySetAsSeries(close,true);

//---- initialization of the indicator in the
OnCalculate() block
   if(prev_calculated>rates_total ||
prev_calculated<=0)// checking for the first start
of the indicator calculation
     {
      limit1=rates_total-1;       // starting index
for calculation of all first loop bars
      limit2=limit1-macd_start-1; // starting index
for calculation of all second loop bars
      limit3=limit1-start-start;  // starting index
for calculation of all third loop bars
```

```
     }
   else // starting index for calculation of new
bars
     {
      limit1=rates_total-prev_calculated;
      limit2=limit1;
      limit3=limit1;
     }

   MaxBar1=rates_total-1;
   MaxBar2=MaxBar1-macd_start-1;
   MaxBar3=MaxBar1-start;

//---- declaration of the CMoving_Average class
variables from the SmoothAlgorithms.mqh file
   static CMoving_Average MA1,MA2,MA3;

//---- main indicator calculation loop
   for(bar=limit1; bar>=0; bar--)
     {

price_=PriceSeries(AppliedPrice,bar,open,low,high,c
lose);

      fast_ma = MA1.MASeries(MaxBar1,
prev_calculated, rates_total, Fast_MA, MA_Method_,
price_, bar, true);
      slow_ma = MA2.MASeries(MaxBar1,
prev_calculated, rates_total, Slow_MA, MA_Method_,
price_, bar, true);
```

```
      MACDLineBuffer[bar]=fast_ma-slow_ma;

SignalLineBuffer[bar]=MA3.MASeries(MaxBar2,prev_cal
culated,rates_total,Signal_SMA,MODE_SMA,MACDLineBuf
fer[bar],bar,true);

      if(bar>MaxBar3)
         HistogramBuffer[bar]=0.0;
      else
HistogramBuffer[bar]=MACDLineBuffer[bar]-SignalLine
Buffer[bar];
     }

//---- Main indicator calculation loop
   for(bar=limit3+2; bar>=0; bar--)
     {
      bullishDivergence[bar]=EMPTY_VALUE;
      bearishDivergence[bar]=EMPTY_VALUE;

      CatchBullishDivergence(low,time,MaxBar3,bar);

CatchBearishDivergence(high,time,MaxBar3,bar);
     }

//---- Main cycle of the histogram coloring
   for(bar=limit3; bar>=0; bar--)
     {
      ColorHistogramBuffer[bar]=0;
```

```
      if(HistogramBuffer[bar]>0)
        {

if(HistogramBuffer[bar]>HistogramBuffer[bar+1])
ColorHistogramBuffer[bar]=1;

if(HistogramBuffer[bar]<HistogramBuffer[bar+1])
ColorHistogramBuffer[bar]=2;
        }

      if(HistogramBuffer[bar]<0)
        {

if(HistogramBuffer[bar]<HistogramBuffer[bar+1])
ColorHistogramBuffer[bar]=3;

if(HistogramBuffer[bar]>HistogramBuffer[bar+1])
ColorHistogramBuffer[bar]=4;
        }
     }
//----
   return(rates_total);
  }
//+------------------------------------------------
------------------+
//| Bullish divergence searching function
                  |
//+------------------------------------------------
------------------+
void CatchBullishDivergence(const double
```

```
&low[],const datetime &time[],int rates_total_,int
shift)
  {
//----
   if(IsIndicatorTrough(shift)==false) return;
   int currentTrough=shift;
   int
lastTrough=GetIndicatorLastTrough(shift,rates_total
_);
//----
   if(lastTrough==-1)return;


if(MACDLineBuffer[currentTrough]>MACDLineBuffer[las
tTrough] &&
      low[currentTrough]<low[lastTrough])
     {

bullishDivergence[currentTrough]=MACDLineBuffer[cur
rentTrough]-

arrowsDisplacement;
      //----
      if(drawPriceTrendLines==true)

DrawPriceTrendLine(time[currentTrough],time[lastTro
ugh],

low[currentTrough],low[lastTrough],BulliDiverColor,
STYLE_SOLID);
```

```
      //----
      if(drawIndicatorTrendLines==true)

DrawIndicatorTrendLine(time[currentTrough],time[las
tTrough],

MACDLineBuffer[currentTrough],MACDLineBuffer[lastTr
ough],BulliDiverColor,STYLE_SOLID);
      //----
      if(displayAlert==true)
         DisplayAlert(time,"Classical bullish
divergence on: ",currentTrough);
     }
//----

if(MACDLineBuffer[currentTrough]<MACDLineBuffer[las
tTrough] &&
      low[currentTrough]>low[lastTrough])
     {

bullishDivergence[currentTrough]=MACDLineBuffer[cur
rentTrough]-arrowsDisplacement;
      //----
      if(drawPriceTrendLines==true)

DrawPriceTrendLine(time[currentTrough],time[lastTro
ugh],

low[currentTrough],low[lastTrough],BulliDiverColor,
STYLE_DOT);
```

```
      //----
      if(drawIndicatorTrendLines==true)

DrawIndicatorTrendLine(time[currentTrough],time[las
tTrough],

MACDLineBuffer[currentTrough],MACDLineBuffer[lastTr
ough],BulliDiverColor,STYLE_DOT);
      //----
      if(displayAlert==true)
         DisplayAlert(time,"Reverse bullish
divergence on: ",currentTrough);
     }
//----
  }
//+------------------------------------------------
------------------+
//| Bearish divergence searching function
                  |
//+------------------------------------------------
------------------+
void CatchBearishDivergence(const double
&high[],const datetime &time[],int rates_total_,int
shift)
  {
//----
   if(IsIndicatorPeak(shift)==false)
      return;
   int currentPeak=shift;
   int
```

```
lastPeak=GetIndicatorLastPeak(shift,rates_total_);
//----
   if(lastPeak==-1)return;


if(MACDLineBuffer[currentPeak]<MACDLineBuffer[lastP
eak] &&
      high[currentPeak]>high[lastPeak])
     {

bearishDivergence[currentPeak]=MACDLineBuffer[curre
ntPeak]+

arrowsDisplacement;

      if(drawPriceTrendLines==true)

DrawPriceTrendLine(time[currentPeak],time[lastPeak]
,
                              high[currentPeak],

high[lastPeak],BearDiverColor,STYLE_SOLID);

      if(drawIndicatorTrendLines==true)

DrawIndicatorTrendLine(time[currentPeak],time[lastP
eak],

MACDLineBuffer[currentPeak],
```

```
MACDLineBuffer[lastPeak],BearDiverColor,STYLE_SOLID
);

      if(displayAlert==true)
         DisplayAlert(time,"Classical bearish
divergence on: ",currentPeak);
     }

if(MACDLineBuffer[currentPeak]>MACDLineBuffer[lastP
eak] &&
      high[currentPeak]<high[lastPeak])
     {

bearishDivergence[currentPeak]=MACDLineBuffer[curre
ntPeak]+

arrowsDisplacement;
      //----
      if(drawPriceTrendLines==true)

DrawPriceTrendLine(time[currentPeak],time[lastPeak]
,
                              high[currentPeak],

high[lastPeak],BearDiverColor,STYLE_DOT);
      //----
      if(drawIndicatorTrendLines==true)

DrawIndicatorTrendLine(time[currentPeak],time[lastP
eak],
```

```
MACDLineBuffer[currentPeak],

MACDLineBuffer[lastPeak],BearDiverColor,STYLE_DOT);
      //----
      if(displayAlert==true)
         DisplayAlert(time,"Reverse bearish
divergence on: ",
                      currentPeak);
     }
//----
  }
//+------------------------------------------------
------------------+
//| The indicator maximum checking function
                  |
//+------------------------------------------------
------------------+
bool IsIndicatorPeak(int shift)
  {
//----
   if(shift==0) return(false);


if(MACDLineBuffer[shift]>=MACDLineBuffer[shift+1]
      &&
MACDLineBuffer[shift]>MACDLineBuffer[shift+2]
      &&
MACDLineBuffer[shift]>MACDLineBuffer[shift-1])
      return(true);
```

```
//----
   return(false);
  }
//+------------------------------------------------
------------------+
//| The indicator minimum checking function
                  |
//+------------------------------------------------
------------------+
bool IsIndicatorTrough(int shift)
  {
//----
   if(shift==0) return(false);


if(MACDLineBuffer[shift]<=MACDLineBuffer[shift+1]
      &&
MACDLineBuffer[shift]<MACDLineBuffer[shift+2]
      &&
MACDLineBuffer[shift]<MACDLineBuffer[shift-1])
      return(true);
//----
   return(false);
  }
//+------------------------------------------------
------------------+
//| The indicator last maximum searching function
                  |
//+------------------------------------------------
------------------+
```

```
int GetIndicatorLastPeak(int shift,int
rates_total_)
  {
//----
   for(int i=shift+5; i<rates_total_; i++)
     {
      if(SignalLineBuffer[i]>=SignalLineBuffer[i+1]
         &&
SignalLineBuffer[i]>=SignalLineBuffer[i+2]
         &&
SignalLineBuffer[i]>=SignalLineBuffer[i-1]
         &&
SignalLineBuffer[i]>=SignalLineBuffer[i-2])
        {
         for(int j=i; j<rates_total_; j++)
           {

if(MACDLineBuffer[j]>=MACDLineBuffer[j+1]
                &&
MACDLineBuffer[j]>MACDLineBuffer[j+2]
                &&
MACDLineBuffer[j]>=MACDLineBuffer[j-1]
                &&
MACDLineBuffer[j]>MACDLineBuffer[j-2])
                return(j);
           }
        }
     }
//----
   return(-1);
```

```
  }
//+------------------------------------------------------------------+
//| The indicator last minimum searching function                    |
//+------------------------------------------------------------------+
int GetIndicatorLastTrough(int shift,int rates_total_)
  {
//----
   for(int i=shift+5; i<rates_total_; i++)
     {
      if(SignalLineBuffer[i]<=SignalLineBuffer[i+1]
         &&
SignalLineBuffer[i]<=SignalLineBuffer[i+2]
         &&
SignalLineBuffer[i]<=SignalLineBuffer[i-1]
         &&
SignalLineBuffer[i]<=SignalLineBuffer[i-2])
        {
         for(int j=i; j<rates_total_; j++)
           {

if(MACDLineBuffer[j]<=MACDLineBuffer[j+1]
                &&
MACDLineBuffer[j]<MACDLineBuffer[j+2]
                &&
MACDLineBuffer[j]<=MACDLineBuffer[j-1]
                &&
```

```
MACDLineBuffer[j]<MACDLineBuffer[j-2])
                return(j);
            }
         }
      }
//----
   return(-1);
  }
//+------------------------------------------------
------------------+
//| Messages display function
                  |
//+------------------------------------------------
------------------+
void DisplayAlert(const datetime &time[],string
message,int shift)
  {
//----
   if(shift<=2 && time[shift]!=lastAlerttime)
     {
      lastAlerttime=time[shift];
      Alert(message,Symbol()," ,
",EnumToString(Period())," minutes chart");
     }
//----
  }
//+------------------------------------------------
------------------+
//| Function for drawing a trend line in a price
chart window        |
```

```
//+------------------------------------------------------------------+
void DrawPriceTrendLine(datetime x1,datetime x2,double y1,
                        double y2,color lineColor,int style)
  {
//----
   string label="MACD_DivergenceLine.0#"+DoubleToString(x1,0);

   if(ObjectFind(0,label)==-1)
     {

ObjectCreate(0,label,OBJ_TREND,0,x1,y1,x2,y2);

ObjectSetInteger(0,label,OBJPROP_COLOR,lineColor);

ObjectSetInteger(0,label,OBJPROP_STYLE,style);
      ObjectSetInteger(0,label,OBJPROP_WIDTH,1);
      ObjectSetInteger(0,label,OBJPROP_RAY,0);
      ObjectSetInteger(0,label,OBJPROP_BACK,true);
     }
   else
     {
      ObjectMove(0,label,0,x1,y1);
      ObjectMove(0,label,1,x2,y2);
     }
//----
  }
```

```
//+------------------------------------------------------------------+
//| Function for drawing a trend line in the indicator window        |
//+------------------------------------------------------------------+
void DrawIndicatorTrendLine(datetime x1,datetime x2,double y1,
                            double y2,color lineColor,int style)
  {
//----
   int indicatorWindow=ChartWindowFind(0,indicatorName);
   if(indicatorWindow<0) return;

   string label="MACD_DivergenceLine.0$#"+DoubleToString(x1,0);

   if(ObjectFind(0,label)==-1)
     {

ObjectCreate(0,label,OBJ_TREND,indicatorWindow,x1,y1,x2,y2);

ObjectSetInteger(0,label,OBJPROP_COLOR,lineColor);

ObjectSetInteger(0,label,OBJPROP_STYLE,style);
      ObjectSetInteger(0,label,OBJPROP_WIDTH,1);
      ObjectSetInteger(0,label,OBJPROP_RAY,0);
```

```
       ObjectSetInteger(0,label,OBJPROP_BACK,true);
      }
    else
      {
       ObjectMove(0,label,0,x1,y1);
       ObjectMove(0,label,1,x2,y2);
      }
//----
  }
//+------------------------------------------------------------------+
Footer
```