```
//+------------------------------------------------------------------+
//|                                                  esperimento.mq5 |
//|                                Copyright 2023, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2023, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------
#include <trade\trade.mqh>
CTrade trade;
ulong BilheteDeVenda = 0;  // Inicializado com 0 para indicar que não há ordem de venda aberta
ulong BilheteDeCompra = 0;// Inicializado com 0 para indicar que não há ordem de compra aberta
ulong ticketvenda =    0;   // Inicializado com 0 para indicar que não há ordem de venda aberta
ulong ticketcompra = 0;  // Inicializado com 0 para indicar que não há ordem de compra aberta
input datetime target_time = D'2023.12.05 20:10:00';
input int MagicNumber = 12345;                          // Unique identifier for the EA
input string password = "jossias";                      // passeword
input  ENUM_TIMEFRAMES PeriodoOperacional = PERIOD_M15 ; // Periodo Operacional
input bool ativar_ou_desativar_venda = true;            // ativar_ou_desativar_venda
input bool ativar_ou_desativar_compra  = true;          // ativar_ou_desativar_compra
input double compra ;                                   // preço de compra
input double venda ;                                    // preço de venda
input double lot = 0.01;                                // lot
input double santo;                                     // pontos para fora da caixa
input bool Modificar_Sl_Para_OxO = true;                // colocar a ordem no 0
input double dedo = 0;                                  // pontos para rompimento da vela
input bool condicao_De_rompimento_v = true;             // abrir a condição  para rompimento da vela na venda
input bool condicao_De_rompimento_c = true;             // abrir a condição  para rompimento da vela na compra
input bool Costura = false;
input int Nives = 1;
input string _____________novas_funcoes_____________;
input bool NOVAS_CONDICOES = false;                     // NOVAS_CONDICOES
input double  preco_para_compra = 0.0;                  // preco_para_compra em NOVAS_CONDICOES
input double preco_para_venda = 0.0;                    // preco_para_venda em NOVAS_CONDICOES
double  fora ;                // variavel para santo
double pc;                    // preço de compra pra linha
double pv;                    // preço de venda  pra linha
double PrecoDeCompra;         // preço de compra pra ordem
double PrecoDeVenda;          // preço de venda pra ordem
double var1;                  // possível   preço de compra
double var2;                  // possível   preço de venda
double Pontos;                // pontos para linhas
double divisao;               // divisão
double Buytake;               // take de compra
double Buystop ;              // estop de compra
double Selltake ;             // teke de venda
double Sellestop ;            //estop de venda
double SellModif ;            // modificar estop sell
double Buymodif;              // modificar estop buy
double Buysubsicul;           // subsicol de compra
double Sellsubsicul ;         // subsicol de  venda
int tpAtingido = 0;           // contegem de take proft
int SlAtingido = 0;           // contagem de stop loss
int slzero = 0 ;              //contegem de zero zero
double preco_de_abertura_de_venda ; // preço de abertura de veda
double preco_de_abertura_de_compra ; // preço de abertura de compra
double nivelzerobuy;               //preco sem perda para compra
double nivelzerosell ;             // preço sem perda para venda
double nivelstoplos_sell;          // preço de stop loss de venda
double nivelstoplos_buy;           // preco de stop loss de compra
double linhasell;             // criar linha de venda para linha
double linhabuy;              // criar linha de compra  para linha
double linhasubsiculv;        // criar linha  de subsicol de venda
double linhasubsiculc;        //criar linha de subsicol de compra
double pontosf;               // pontos  para ordem
double takbuy ;               // nivel de take buy
double taksell;               // nivel de take sell


//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
double precosArray [];
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
//+------------------------------------------------------------------+
//|                            controles                             |
//+------------------------------------------------------------------+
bool controlbuy = false;          // control de ordem de compra
bool controlsell = false;         // control de ordem de venda
```

```
bool controlbuymdf = false;       // control de modificaçao de ordem de compra
bool controlsellmdf = false;      //  control de modificaçao de ordem de venda
bool linhasc = false; // control de linha de  compra
bool linhasv = false; // control de linha de  venda
bool segundo_control_de_taksell = false;
bool segundo_control_de_takbuy = false ;
bool cont = false ;
bool cont1 = false ;
bool cont2 = false ;
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
void criarlinhascompra(string precocompra, double valor)
  {
   ObjectCreate(0, precocompra, OBJ_HLINE, 0, 0, valor);
   ObjectSetInteger(0, precocompra, OBJPROP_STYLE, STYLE_SOLID);
   ObjectSetInteger(0, precocompra, OBJPROP_COLOR, clrRed);
  }
//+------------------------------------------------------------------+
void criarlinhavenda(string precovanda, double valor)
  {
   ObjectCreate(0, precovanda, OBJ_HLINE, 0, 0, valor);
   ObjectSetInteger(0, precovanda, OBJPROP_STYLE, STYLE_SOLID);
   ObjectSetInteger(0, precovanda, OBJPROP_COLOR, clrRed);
  }
//
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
int OnInit()
  {

   if(password == "jossias")
     {
     }
   else
     {
      ExpertRemove();
     }
   var1 = compra;
   var2 = venda;
   PrecoDeCompra = var1;
   PrecoDeVenda = var2;
   pontosf = var1 - var2;
   divisao = pontosf / 2;
   Buytake = PrecoDeCompra + (pontosf * Nives);
   Buystop = PrecoDeCompra - pontosf;
   Buysubsicul = PrecoDeCompra + divisao;
   Buymodif = PrecoDeCompra;
   Selltake = PrecoDeVenda - (pontosf * Nives);
   Sellestop =  PrecoDeVenda + pontosf;
   Sellsubsicul = PrecoDeVenda - divisao;
   SellModif = PrecoDeVenda;


   double santinho =  SymbolInfoDouble(_Symbol, SYMBOL_POINT);
   fora = santinho * santo ;

// Definir o símbolo e o período
   string symbol = Symbol();

// Obter a hora da vela atual
   datetime currentTime = iTime(symbol, PeriodoOperacional, 0);

//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
   var1 =  compra;
   var2 = venda;
   pc = var1 ;
   pv = var2 ;
   Pontos = pc - pv;
   divisao = Pontos / 2;
   linhabuy = pc + Pontos;
   linhasubsiculc = pc + divisao;
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
   pc = var1 ;
   pv = var2 ;
   Pontos = pc - pv;
   divisao = Pontos / 2;
   linhasell = pv - Pontos;
   linhasubsiculv = pv - divisao;
```

```
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
   criarlinhavenda("prevodevenda", pv);
   criarlinhascompra("precode compra ", pc);
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   ObjectsDeleteAll(0, "precocompra", 0, OBJ_HLINE);
   ObjectsDeleteAll(0, "precovanda", 0, OBJ_HLINE);
   ObjectsDeleteAll(0, "linha3_", 0, OBJ_HLINE);
   ObjectsDeleteAll(0, "linha2_", 0, OBJ_HLINE);
   ObjectsDeleteAll(0, "linha1_", 0, OBJ_HLINE);
   ObjectsDeleteAll(0, "linha_", 0, OBJ_HLINE);
//---
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
   double nivel = MathAbs(PrecoDeCompra - PrecoDeVenda) /  SymbolInfoDouble(_Symbol, SYMBOL_POINT);
   double precocurrentecB = SymbolInfoDouble(_Symbol, SYMBOL_BID);
   double precocurrentevA = SymbolInfoDouble(_Symbol, SYMBOL_ASK);
   double closeprice = iClose(Symbol(), PeriodoOperacional, 1);
   double precodiferentev = MathAbs(closeprice - PrecoDeVenda) / Point();
   double precodiferentec = MathAbs(PrecoDeCompra - closeprice) / Point();
   datetime current_time = TimeLocal();
   string symbol = Symbol();
   datetime currentTime = iTime(symbol, PeriodoOperacional, 0);

   if(current_time >= target_time)
     {
     }

//+------------------------------------------------------------------+
//|                        CRIAÇÃO DE LINHAS                         |
//+------------------------------------------------------------------+
   if(closeprice > pc)
     {
      criarlinhaBuy1(linhasubsiculc);
      criarlinhaBuy2(linhabuy) ;
     }
//+------------------------------------------------------------------+
//|                        CRIAÇÃO DE LINHAS                         |
//+------------------------------------------------------------------+


   if(closeprice > linhabuy)
     {
      pc = pc + Pontos ;
      pv =  pv + Pontos ;
      Pontos = pc - pv;
      divisao = Pontos / 2;
      linhabuy = pc + Pontos;
      linhasubsiculc = pc + divisao;
      linhasell = pv - Pontos;
      linhasubsiculv = pv - divisao;
     }
//+------------------------------------------------------------------+
//|                        CRIAÇÃO DE LINHAS                         |
//+------------------------------------------------------------------+

   if(closeprice < pv)
     {
      criarlinhassell1(linhasubsiculv);
      criarlinhassell2(linhasell);
     }
//+------------------------------------------------------------------+
//|                        CRIAÇÃO DE LINHAS                         |
//+------------------------------------------------------------------+


   if(closeprice < linhasubsiculv)
     {
      pc = pc - Pontos ;
      pv = pv - Pontos ;
      Pontos = pc - pv;
      divisao = Pontos / 2;
      linhasell = pv - Pontos;
```

```
      linhasubsiculv = pv - divisao;
      linhabuy = pc + Pontos;
      linhasubsiculc = pc + divisao;
     }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
   if(precocurrentecB >= takbuy)
     {
      if(!segundo_control_de_takbuy)
        {
         PrecoDeVenda = PrecoDeCompra - pontosf;
         Buytake = PrecoDeCompra + (pontosf * Nives);
         Buystop = PrecoDeCompra - pontosf;
         Buysubsicul = PrecoDeCompra + divisao;
         Buymodif = PrecoDeCompra;
         Selltake = PrecoDeVenda - (pontosf * Nives);
         Sellestop =  PrecoDeVenda + pontosf;
         Sellsubsicul = PrecoDeVenda - divisao;
         SellModif = PrecoDeVenda;

         /*  Print("novo preço de compra. :", PrecoDeCompra);
           Print(" novo preço de venda. :", PrecoDeVenda);
           Print(" novo Buysubsicul. :", Buysubsicul);
           Print("novo Buytake. :", Buytake);
           Print("novo Buystop. :", Buystop);
           Print(" novo Sellsubsicul. :", Sellsubsicul);
           Print(" novo selltake :  ", Selltake);
           Print("novo Sellestop. :", Sellestop);
           */segundo_control_de_takbuy = true;
        }
     }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
   if(precocurrentevA <= taksell)
     {
      if(!segundo_control_de_taksell)
        {
         PrecoDeCompra = PrecoDeVenda + pontosf;
         Buytake = PrecoDeCompra + (pontosf * Nives);
         Buystop = PrecoDeCompra - pontosf;
         Buysubsicul = PrecoDeCompra + divisao;
         Buymodif = PrecoDeCompra;
         Selltake = PrecoDeVenda - (pontosf * Nives);
         Sellestop =  PrecoDeVenda + pontosf;
         Sellsubsicul = PrecoDeVenda - divisao;
         SellModif = PrecoDeVenda;

         /*     Print("novo preço de compra. :", PrecoDeCompra);
              Print(" novo preço de venda. :", PrecoDeVenda);
              Print(" novo Buysubsicul. :", Buysubsicul);
              Print("novo Buytake. :", Buytake);
              Print("novo Buystop. :", Buystop);
              Print(" novo Sellsubsicul. :", Sellsubsicul);
              Print(" novo selltake :  ", Selltake);
              Print("novo Sellestop. :", Sellestop);
           */   segundo_control_de_taksell = true;

        }

     }
   /*int totalPositions = PositionsTotal();
   Print("Número total de posições abertas: ", totalPositions);

       if (totalPositions == 1 )
       {
       Print("uma posiçao aberta");
       }
       else{
       Print ("nenhuma posiçao aberta");
       }
     */


   /*


   // Obter a última e penúltima vela
   MqlRates rates[2];
   string symbol = Symbol();
   // Copiar os dados das últimas 2 velas para a matriz 'rates'
   int copied = CopyRates(symbol, PeriodoOperacional, 0, 2, rates);
```

```
if(copied > 0)
{
    // Última vela
    double lastOpen = rates[0].open;
    double lastHigh = rates[0].high;
    double lastLow = rates[0].low;
    double lastClose = rates[0].close;

    // Penúltima vela
    double prevOpen = rates[1].open;
    double prevHigh = rates[1].high;
    double prevLow = rates[1].low;
    double prevClose = rates[1].close;

    // Imprimir os dados
    Print("Última vela - Abertura: ", lastOpen, " Máximo: ", lastHigh, " Mínimo: ", lastLow, " Fechamento: ", lastClose);

    Print("Penúltima vela - Abertura: ", prevOpen, " Máximo: ", prevHigh, " Mínimo: ", prevLow, " Fechamento: ", prevClose);
}
else
{
    Print("Erro ao copiar os dados das velas: ", GetLastError());
}
 */

/*

// Definir o símbolo e o período
string symbol = Symbol();
ENUM_TIMEFRAMES period = PERIOD_M1; // Exemplo para gráfico de 1 minuto

// Obter a hora da vela atual
datetime currentTime = iTime(symbol, period, 0);

// Calcular a hora 45 minutos antes da vela atual
datetime time45MinutesBefore = currentTime - (45 * 60); // 45 minutos * 60 segundos

// Imprimir a hora 45 minutos antes da vela atual
Print("Hora 45 minutos antes da vela atual: ", TimeToStr(time45MinutesBefore, TIME_DATE|TIME_MINUTES));
*/

/*
// Definir o símbolo e o período
string symbol = Symbol();
double candleSize = 0.0 ;
// Obter a última vela
MqlRates rates[1];

// Copiar os dados da última vela para a matriz 'rates'
int copied = CopyRates(symbol, PeriodoOperacional, 0, 1, rates);

if(copied > 0)
{
   // Última vela
   double lastOpen = rates[0].open;
   double lastClose = rates[0].close;

   // Calcular o tamanho da vela
   candleSize = MathAbs(lastClose - lastOpen)/ _Point;

   // Imprimir o tamanho da vela
   Print("Tamanho da última vela: ", candleSize);
}
else
{
   Print("Erro ao copiar os dados da vela: ", GetLastError());
}
double tamanho = candleSize;

if (tamanho >= dedo)
{
// Definir o símbolo e o período
string symbol = Symbol();

// Obter as 20 últimas velas
MqlRates rates[45];
CopyRates(symbol, PeriodoOperacional, 0, 46, rates);


// Coordenadas para a linha de tendência
datetime time1 = rates[45].time;
double price1 = rates[0].close;
datetime time2 = rates[0].time;
double price2 = rates[0].close;
```

```
   // Criar a linha de tendência
   string trendlineName = "Trendline1";
   if(!ObjectCreate(0, trendlineName, OBJ_TREND, 0, time1,price1,time2,price2))
   {
      Print("Erro ao criar a linha de tendência: ", GetLastError());
   }
   else
   {
      Print("Linha de tendência criada com sucesso!");
   }

   Print("kkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk");
   }
   */

// Definir o símbolo e o período

   ENUM_TIMEFRAMES period = PERIOD_M1; // Exemplo para gráfico de 1 minuto

// Obter a última vela
   MqlRates rates[1];
   CopyRates(symbol, period, 0, 1, rates);

// Coordenadas para o texto
   datetime time = rates[0].time;
   double price = rates[0].close;

// Criar o objeto de texto
   string textName = "Text1";
   if(!ObjectCreate(0, textName, OBJ_TEXT, 0, time, price))
     {
      Print("Erro ao criar o objeto de texto: ", GetLastError());
     }
   else
     {
      // Definir o texto
      string text = "CANAL DE REFERENCIA";
      ObjectSetString(0, textName, OBJPROP_TEXT, text);

      // Definir a cor do texto
      ObjectSetInteger(0, textName, OBJPROP_COLOR, clrRed);

      Print("Texto criado com sucesso!");
     }

//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+

   int totalPositions = PositionsTotal();
   if(totalPositions == 1)
     {
      Print("uma posiçao aberta");
      Print("hora atual. ", currentTime);

     }
   else
     {



      bool condicao2 = (condicao_De_rompimento_c) ?
                       (precodiferentec < dedo) : true ;



      if(closeprice > PrecoDeCompra && condicao2)
        {
         if(! controlbuy)
           {
            if(ativar_ou_desativar_compra)
              {
               BuyMarkt();
              }
            /*   Print("novo preço de compra. :", PrecoDeCompra);
               Print(" novo preço de venda. :", PrecoDeVenda);
               Print(" novo Buysubsicul. :", Buysubsicul);
               Print("novo Buytake. :", Buytake);
               Print("novo Buystop. :", Buystop);
               Print(" novo Sellsubsicul. :", Sellsubsicul);
               Print(" novo selltake :  ", Selltake);
               Print("novo Sellestop. :", Sellestop);
               Print("nivel de stop loss buy :", nivelstoplos_buy);
               */
```

```
            PrecoDeVenda = PrecoDeCompra - pontosf ;
            Buytake = PrecoDeCompra + (pontosf * Nives);
            Buystop = PrecoDeCompra - pontosf;
            Buysubsicul = PrecoDeCompra + divisao;
            Buymodif = PrecoDeCompra;
            Selltake = PrecoDeVenda - (pontosf * Nives);
            Sellestop =  PrecoDeVenda + pontosf;
            Sellsubsicul = PrecoDeVenda - divisao;
            SellModif = PrecoDeVenda;

            if(Costura)
              {
               controlsell = false ;
              }
            controlbuymdf = false;
            segundo_control_de_takbuy = false;
            segundo_control_de_taksell = false;
            cont = false;
            controlbuy = true;
           }
        }
      Print("nenhuma posiçao aberta");

     }

//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+

   if(closeprice > Buysubsicul)
     {
      if(!controlbuymdf)
        {
         ModifyBuyOrder();
         PrecoDeVenda = PrecoDeCompra;
         PrecoDeCompra = Buytake;
         Buytake = PrecoDeCompra + (pontosf * Nives);
         Buystop = PrecoDeCompra - pontosf;
         Buysubsicul = PrecoDeCompra + divisao;
         Buymodif = PrecoDeCompra;
         Selltake = PrecoDeVenda - (pontosf * Nives);
         Sellestop =  PrecoDeVenda + pontosf;
         Sellsubsicul = PrecoDeVenda - divisao;
         SellModif = PrecoDeVenda;
         /*   Print("novo preço de compra. :", PrecoDeCompra);
            Print(" novo preço de venda. :", PrecoDeVenda);
            Print(" novo Buysubsicul. :", Buysubsicul);
            Print("novo Buytake. :", Buytake);
            Print("novo Buystop. :", Buystop);
            Print(" novo Sellsubsicul. :", Sellsubsicul);
            Print(" novo selltake :  ", Selltake);
            Print("novo Sellestop. :", Sellestop);
            Print("nivel de stop loss buy :", nivelstoplos_buy);
           */ if(Costura)
           {
            controlbuy = false;
            controlsell = false;
           }

         controlbuymdf = true;
        }
     }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+

   bool condicao = (condicao_De_rompimento_v) ?
                   (precodiferentev < dedo) : true ;


   if(totalPositions == 1)
     {
      Print("uma posiçao aberta");

     }
   else
     {
      if(closeprice < PrecoDeVenda && condicao)
        {


         if(!controlsell)
           {
            if(ativar_ou_desativar_venda)
```

```
            {
             SellMarkt();
            }
          /*
                   Print("novo preço de compra. :", PrecoDeCompra);
                   Print(" novo preço de venda. :", PrecoDeVenda);
                   Print(" novo Buysubsicul. :", Buysubsicul);
                   Print("novo Buytake. :", Buytake);
                   Print("novo Buystop. :", Buystop);
                   Print(" novo Sellsubsicul. :", Sellsubsicul);
                   Print(" novo selltake :  ", Selltake);
                   Print("novo Sellestop. :", Sellestop);
                   Print("nivel de stop loss sell :", nivelstoplos_sell);
            */
          PrecoDeCompra = PrecoDeVenda + pontosf;
          Buytake = PrecoDeCompra + (pontosf * Nives);
          Buystop = PrecoDeCompra - pontosf;
          Buysubsicul = PrecoDeCompra + divisao;
          Buymodif = PrecoDeCompra;
          Selltake = PrecoDeVenda - (pontosf * Nives);
          Sellestop =  PrecoDeVenda + pontosf;
          Sellsubsicul = PrecoDeVenda - divisao;
          SellModif = PrecoDeVenda;

          if(Costura)
            {
             controlbuy = false;
            }

          controlsellmdf = false;
          segundo_control_de_takbuy = false;
          segundo_control_de_taksell =  false;
          cont1 = false ;
          controlsell = true;
         }
       }
     Print("nenhuma posiçao aberta");
    }


//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
 /*int totalPositions = PositionsTotal();
   Print("Número total de posições abertas: ", totalPositions);

       if (totalPositions == 1 )
       {
       Print("uma posiçao aberta");
       }
       else{
       Print ("nenhuma posiçao aberta");
       }
    */


   /*


   // Obter a última e penúltima vela
   MqlRates rates[2];
   string symbol = Symbol();
   // Copiar os dados das últimas 2 velas para a matriz 'rates'
   int copied = CopyRates(symbol, PeriodoOperacional, 0, 2, rates);

   if(copied > 0)
   {
       // Última vela
       double lastOpen = rates[0].open;
       double lastHigh = rates[0].high;
       double lastLow = rates[0].low;
       double lastClose = rates[0].close;

       // Penúltima vela
       double prevOpen = rates[1].open;
       double prevHigh = rates[1].high;
       double prevLow = rates[1].low;
       double prevClose = rates[1].close;

       // Imprimir os dados
       Print("Última vela - Abertura: ", lastOpen, " Máximo: ", lastHigh, " Mínimo: ", lastLow, " Fechamento: ", lastClose);

       Print("Penúltima vela - Abertura: ", prevOpen, " Máximo: ", prevHigh, " Mínimo: ", prevLow, " Fechamento: ", prevClose);
   }
```

```
   else
   {
       Print("Erro ao copiar os dados das velas: ", GetLastError());
   }
    */

   /*

   // Definir o símbolo e o período
   string symbol = Symbol();
   ENUM_TIMEFRAMES period = PERIOD_M1; // Exemplo para gráfico de 1 minuto

   // Obter a hora da vela atual
   datetime currentTime = iTime(symbol, period, 0);

   // Calcular a hora 45 minutos antes da vela atual
   datetime time45MinutesBefore = currentTime - (45 * 60); // 45 minutos * 60 segundos

   // Imprimir a hora 45 minutos antes da vela atual
   Print("Hora 45 minutos antes da vela atual: ", TimeToStr(time45MinutesBefore, TIME_DATE|TIME_MINUTES));
   */

   /*
   // Definir o símbolo e o período
   string symbol = Symbol();
   double candleSize = 0.0 ;
   // Obter a última vela
   MqlRates rates[1];

   // Copiar os dados da última vela para a matriz 'rates'
   int copied = CopyRates(symbol, PeriodoOperacional, 0, 1, rates);

   if(copied > 0)
   {
      // Última vela
      double lastOpen = rates[0].open;
      double lastClose = rates[0].close;

      // Calcular o tamanho da vela
      candleSize = MathAbs(lastClose - lastOpen)/ _Point;

      // Imprimir o tamanho da vela
      Print("Tamanho da última vela: ", candleSize);
   }
   else
   {
      Print("Erro ao copiar os dados da vela: ", GetLastError());
   }
   double tamanho = candleSize;

   if (tamanho >= dedo)
   {
   // Definir o símbolo e o período
   string symbol = Symbol();

   // Obter as 20 últimas velas
   MqlRates rates[45];
   CopyRates(symbol, PeriodoOperacional, 0, 46, rates);


   // Coordenadas para a linha de tendência
   datetime time1 = rates[45].time;
   double price1 = rates[0].close;
   datetime time2 = rates[0].time;
   double price2 = rates[0].close;

   // Criar a linha de tendência
   string trendlineName = "Trendline1";
   if(!ObjectCreate(0, trendlineName, OBJ_TREND, 0, time1,price1,time2,price2))
   {
      Print("Erro ao criar a linha de tendência: ", GetLastError());
   }
   else
   {
      Print("Linha de tendência criada com sucesso!");
   }

   Print("kkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk");
   }
   */

// Definir o símbolo e o período

   /*   ENUM_TIMEFRAMES period = PERIOD_M1; // Exemplo para gráfico de 1 minuto
```

```
   // Obter a última vela
      MqlRates rates[1];
      CopyRates(symbol, period, 0, 1, rates);

   // Coordenadas para o texto
      datetime time = rates[0].time;
      double price = rates[0].close;

   // Criar o objeto de texto
      string textName = "Text1";
      if(!ObjectCreate(0, textName, OBJ_TEXT, 0, time, price))
        {
         Print("Erro ao criar o objeto de texto: ", GetLastError());
        }
      else
        {
         // Definir o texto
         string text = "CANAL DE REFERENCIA";
         ObjectSetString(0, textName, OBJPROP_TEXT, text);

         // Definir a cor do texto
         ObjectSetInteger(0, textName, OBJPROP_COLOR, clrRed);

         //  Print("Texto criado com sucesso!");
        }
   */
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+

//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
   if(! controlsellmdf)
     {
      if(closeprice < Sellsubsicul)
        {
         ModifySellOrder();
         PrecoDeCompra = PrecoDeVenda;
         PrecoDeVenda = Selltake;
         Buytake = PrecoDeCompra + (pontosf * Nives);
         Buystop = PrecoDeCompra - pontosf;
         Buysubsicul = PrecoDeCompra + divisao;
         Buymodif = PrecoDeCompra;
         Selltake = PrecoDeVenda - (pontosf * Nives);
         Sellestop =  PrecoDeVenda + pontosf;
         Sellsubsicul = PrecoDeVenda - divisao;
         SellModif = PrecoDeVenda;

         if(Nives == 2)  // || precoatual > sellstop
           {
            Print("nivel é igual ha 2 ou 3");

            Print("canal rompido sem pegar o tak");


           }

         /*   Print("novo preço de compra. :", PrecoDeCompra);
            Print(" novo preço de venda. :", PrecoDeVenda);
            Print(" novo Buysubsicul. :", Buysubsicul);
            Print("novo Buytake. :", Buytake);
            Print("novo Buystop. :", Buystop);
            Print(" novo Sellsubsicul. :", Sellsubsicul);
            Print(" novo selltake :  ", Selltake);
            Print("novo Sellestop. :", Sellestop);
            Print("nivel de stop loss sell :", nivelstoplos_sell);
           */ if(Costura)
           {
            controlbuy = false;
            controlsell = false;
           }

         controlsellmdf = true;
        }

     }

  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
//+------------------------------------------------------------------+
//|                                                                  |
```

```
//+------------------------------------------------------------------+
bool BuyMarkt()
  {
   MqlTradeRequest request = { };
   MqlTradeResult result = { };
   request.action = TRADE_ACTION_DEAL;
   request.volume = lot;
   request.symbol = Symbol();
   request.type   = ORDER_TYPE_BUY;
   request.sl = Buystop - fora ;
   request.tp = Buytake - fora   ;
   takbuy = request.tp;
   nivelstoplos_buy = request.sl;
   if(OrderSend(request, result))
     {
      BilheteDeCompra = result.order;
      preco_de_abertura_de_compra = result.price;
      Print("ordem de compra emviada -Bilhete : ", BilheteDeCompra);
      return true;
     }
   else
     {
      Print("Erro ao enviar a ordem de compra -erro :", GetLastError());
      return false;
     }
  }
//+------------------------------------------------------------------+
bool SellMarkt()
  {
   MqlTradeRequest request = { };
   MqlTradeResult result = { };
   request.action = TRADE_ACTION_DEAL;
   request.volume = lot;
   request.symbol = Symbol();
   request.type   = ORDER_TYPE_SELL;
   request.sl = Sellestop + fora  ;
   request.tp = Selltake + fora ;
   taksell = request.tp;
   nivelstoplos_sell = request.sl;
   if(OrderSend(request, result))
     {
      BilheteDeVenda = result.order;
      preco_de_abertura_de_venda = result.price;
      Print("ordem de venda emviada -Bilhete : ", BilheteDeVenda);
      return true;
     }
   else
     {
      Print("Erro ao enviar a ordem de venda -erro :", GetLastError());
      return false;
     }
   return true;
  }
//+------------------------------------------------------------------+
bool ModifySellOrder()
  {
   if(PositionSelectByTicket(BilheteDeVenda))
     {
        {
         MqlTradeRequest request = { };
         MqlTradeResult result = { };
         request.action = TRADE_ACTION_SLTP;
         request.symbol = Symbol();
         request.tp = Selltake + fora;
         request.position = BilheteDeVenda;
         if(Modificar_Sl_Para_OxO)
           {
            request.sl = preco_de_abertura_de_venda  ;
           }
         else
           {
            request.sl = SellModif;
           }
         nivelzerosell = request.sl;
         taksell = request.tp;
         if(OrderSend(request, result))
           {
            Print("Ordem de venda modificada - Bilhete: ", BilheteDeVenda, " Novo SL: ", SellModif);
            return true;
           }
         else
           {
            Print("Erro ao modificar a ordem de venda - Bilhete: ", BilheteDeVenda);
            return false;
```

```
         }
       }
     }
   return true;
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
bool ModifyBuyOrder()
  {
   if(PositionSelectByTicket(BilheteDeCompra))
     {
        {
         MqlTradeRequest request = { };
         MqlTradeResult result = { };
         request.action = TRADE_ACTION_SLTP;
         request.symbol = Symbol();
         request.tp = Buytake - fora;
         request.position = BilheteDeCompra;

         if(Modificar_Sl_Para_OxO)
           {
            request.sl = preco_de_abertura_de_compra;
           }
         else
           {
            request.sl = Buymodif;
           }
         nivelzerobuy = request.sl;
         takbuy = request.tp;
         if(OrderSend(request, result))
           {
            Print("Ordem de compra modificada - Bilhete: ", BilheteDeCompra, " Novo SL: ", Buymodif);
            return true;
           }
         else
           {
            Print("Erro ao modificar a ordem de compra - Bilhete: ", BilheteDeCompra);
           }
         return false;
        }
     }
   return true;
  }
//+------------------------------------------------------------------+
void criarlinhassell1(double valor)
  {
   string subsicul = "linha3_" + DoubleToString(valor, 4);
   bool unica3 = true;
   for(int i = 0; i < ArraySize(precosArray); i++)
     {
      if(precosArray[i] == valor)
        {
         unica3 = false ;
         break;
        }
     }
   if(unica3)
     {
      int currentSize = ArraySize(precosArray);
      ArrayResize(precosArray, currentSize + 1);
      precosArray[currentSize] = valor;
      string subsicul = "linha3_" + DoubleToString(valor, 4);
      ObjectCreate(0, subsicul, OBJ_HLINE, 0, 0, valor);
      ObjectSetInteger(0, subsicul, OBJPROP_STYLE, STYLE_SOLID);
      ObjectSetInteger(0, subsicul, OBJPROP_COLOR, clrBlue);
     }
  }
//+------------------------------------------------------------------+
void criarlinhassell2(double valor)
  {
   string precosell = "linha2_" + DoubleToString(valor, 4);
   bool unica2 = true;
   for(int i = 0; i < ArraySize(precosArray); i++)
     {
      if(precosArray[i] == valor)
        {
         unica2 = false ;
         break;
        }
     }
   if(unica2)
     {
      int currentSize = ArraySize(precosArray);
```

```
      ArrayResize(precosArray, currentSize + 1);
      precosArray[currentSize] = valor;
      string precosell = "linha2_" + DoubleToString(valor, 4);
      ObjectCreate(0, precosell, OBJ_HLINE, 0, 0, valor);
      ObjectSetInteger(0, precosell, OBJPROP_STYLE, STYLE_SOLID);
      ObjectSetInteger(0, precosell, OBJPROP_COLOR, clrRed);
     }
  }
//+------------------------------------------------------------------+
//|                 // criar linha de compra  vermelho                                                  |
//+------------------------------------------------------------------+
void criarlinhaBuy1(double valor)
  {
   string subsiculbuy = "linha1_" + DoubleToString(valor, 4);
   bool unica1 = true;
   for(int i = 0; i < ArraySize(precosArray); i++)
     {
      if(precosArray[i] == valor)
        {
         unica1 = false ;
         break;
        }
     }
   if(unica1)
     {
      int currentSize = ArraySize(precosArray);
      ArrayResize(precosArray, currentSize + 1);
      precosArray[currentSize] = valor;
      string subsiculbuy = "linha1_" + DoubleToString(valor, 4);
      ObjectCreate(0, subsiculbuy, OBJ_HLINE, 0, 0, valor);
      ObjectSetInteger(0, subsiculbuy, OBJPROP_STYLE, STYLE_SOLID);
      ObjectSetInteger(0, subsiculbuy, OBJPROP_COLOR, clrBlue);
     }
  }
//+------------------------------------------------------------------+
void criarlinhaBuy2(double valor)
  {
   string precobuy = "linha_" + DoubleToString(valor, 4);
   bool unica = true;
   for(int i = 0; i < ArraySize(precosArray); i++)
     {
      if(precosArray[i] == valor)
        {
         unica = false ;
         break;
        }
     }
   if(unica)
     {
      int currentSize = ArraySize(precosArray);
      ArrayResize(precosArray, currentSize + 1);
      precosArray[currentSize] = valor;
      string precobuy = "linha_" + DoubleToString(valor, 4);
      ObjectCreate(0, precobuy, OBJ_HLINE, 0, 0, valor);
      ObjectSetInteger(0, precobuy, OBJPROP_STYLE, STYLE_SOLID);
      ObjectSetInteger(0, precobuy, OBJPROP_COLOR, clrRed);
     }
  }



//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+

// +------------------------------------------------------------------ +

//+------------------------------------------------------------------+

//+------------------------------------------------------------------+

//+------------------------------------------------------------------+

//+------------------------------------------------------------------+

//+------------------------------------------------------------------+

//+------------------------------------------------------------------+

//+------------------------------------------------------------------+

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

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