Додаток А

Лістинг додатку «Ідентифікації простих чисел»

```
//---------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include "Unit1.h"
#include "Unit2.h"
#include "Unit3.h"
#include "Unit4.h"
#include "Unit6.h"
#include "Unit7.h"
#include "Unit8.h"
#include "Unit9.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
big_prime *prime;
TChrtBuilder *ChartBuilder1;
remaining *remainingtablebuilder1;
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{

}
//---------------------------------------------------------------------------



void __fastcall TForm1::Button2Click(TObject *Sender)
{
```

```
prime->Suspend();
Label2->Caption=IntToStr(prime->inumber);
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button3Click(TObject *Sender)
{

 prime= new big_prime(false);

}
//---------------------------------------------------------------------------

void __fastcall TForm1::Close2Click(TObject *Sender)
{
Close();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Close1Click(TObject *Sender)
{
 if (SaveDialog1->Execute()){TestingMemo->Lines->SaveToFile(SaveDialog1->FileName);}
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button4Click(TObject *Sender)
{
prime->Resume();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
Form1->TestingMemo->Clear();
 TDateTime DateTime = Time();  // store the current date and time
  AnsiString str = TimeToStr(DateTime); // convert the time to a string
  Form1->TestingMemo->Lines->Add(str);
```

```
//testing_number(StrToInt(Form1->Edit3->Text),prime_list,Form1->TestingMemo);
if (prime)prime->from_vectro_to_memo(Form1->TestingMemo,prime->prime_list);
DateTime = Time();
str = TimeToStr(DateTime); // convert the time to a string
Form1->TestingMemo->Lines->Add(str);
//if (prime)prime_list=prime->prime_list;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button5Click(TObject *Sender)
{
Memo1->Clear();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button6Click(TObject *Sender)
{
bool isprime=true;
bool limitout=true;
long i=1;
long num=StrToInt(Edit1->Text);
  //Memo1->Lines->Add(IntToStr(prime->prime_list.size()));
long temp;//= new long;
while (limitout) {
if (i<prime->prime_list.size()){
//Memo1->Lines->Add(IntToStr(prime->prime_list[i]));
        if (prime->prime_list[i]<num){
        temp=(num-prime->prime_list[i]);
        Memo1->Lines->Add(IntToStr(temp));
                // for (int j=0;j==0; *temp>>=1, j++){
                 if (temp&(1<<0)?1:0){
                 isprime=false;
                 Memo1->Lines->Add(IntToStr(0000000000000));
                 };
                // };
        i++;
```

```
      }else{limitout=false;};
}else{limitout=false;};
};
if (isprime){Label4->Caption="Число просте ...";}else{Label4->Caption=" ";};
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button8Click(TObject *Sender)
{
//thre = new remaining(false);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ScrollBar1Change(TObject *Sender)
{
Chart1->Chart3DPercent=ScrollBar1->Position;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ScrollBar2Change(TObject *Sender)
{
if (ScrollBar1->Enabled ){
Chart1->View3DOptions->Orthogonal=false;
Chart1->View3DOptions->Rotation=ScrollBar2->Position;
};
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ScrollBar3Change(TObject *Sender)
{
if (ScrollBar1->Enabled) {
   Chart1->View3DOptions->Orthogonal=false;
   Chart1->View3DOptions->Elevation=ScrollBar3->Position;
  }
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::CheckBox5Click(TObject *Sender)
{
Chart1->View3D=CheckBox5->Checked;
ScrollBar1->Enabled=Form1->Chart1->View3D;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button11Click(TObject *Sender)
{
ListBox1->Clear();
for (int i=0;i<Chart1->SeriesCount();i++){
ListBox1->Items->Add("series "+IntToStr(i));
};
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button10Click(TObject *Sender)
{
Chart1->RemoveSeries(Chart1->Series[ListBox1->ItemIndex]);
// Chart1->Series[ListBox1->ItemIndex]->DestroyComponents();
ListBox1->Clear();
for (int i=0;i<Chart1->SeriesCount();i++){
ListBox1->Items->Add("series "+IntToStr(i));
};
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button9Click(TObject *Sender)
{
  ChartBuilder1= new TChrtBuilder(false);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button12Click(TObject *Sender)
{
///Edit13->Text=IntToStr(GetChargePartNumber(StrToInt(Edit10->Text),StrToInt(Edit11->Text),StrToInt(Edit12->Text)));
```

```
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Chart1Zoom(TObject *Sender)
{
ScrollBar4->Position=Chart1->View3DOptions->Zoom;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ScrollBar4Change(TObject *Sender)
{
Chart1->View3DOptions->Zoom=ScrollBar4->Position;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button7Click(TObject *Sender)
{
//void from_vector_to_table_string(vector<long>& mas1,TStringGrid *StringGrid1){
remainingtablebuilder1=new remaining(false);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button13Click(TObject *Sender)
{
if( ColorDialog1->Execute() )
        Color1 = ColorDialog1->Color;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::N2Click(TObject *Sender)
{
Form6->Show();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::N3Click(TObject *Sender)
{
Form7->Show();
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::N4Click(TObject *Sender)
{
Form8->Show();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::N5Click(TObject *Sender)
{
Form9->Show();
}
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit2.h"
#include "Unit1.h"
#pragma package(smart_init)
//---------------------------------------------------------------------------

//   Important: Methods and properties of objects in VCL can only be
//   used in a method called using Synchronize, for example:
//
//      Synchronize(UpdateCaption);
//
//   where UpdateCaption could look like:
//
//      void __fastcall big_prime::UpdateCaption()
//      {
//        Form1->Caption = "Updated in a thread";
//      }
//---------------------------------------------------------------------------
//***************************************************************************
void big_prime::from_memo_to_vector(TMemo *temp1,vector<long>& temp2){
temp2.resize(temp1->Lines->Count);
```

```
for (int i=0;i<temp2.size();i++){
temp2[i]=StrToInt(temp1->Lines->Strings[i]);
};
};
//***************************************************************************
String big_prime::from_vector_to_string(vector<bool>& temp2){
String s="";
for (int i=0;i<temp2.size();i++){
if (temp2[i]==0) {s=s+'0';} else {s=s+'1';};
}; return s;};

//***************************************************************************
bool big_prime::from_long_to_bit_vector(long num,vector<bool>& temp1)
{
  int i;
  for (i=0; num; num>>=1, i++){
  temp1.insert(temp1.begin(),bool((num&1)?(1):(0)));
  };
};
//***************************************************************************
bool big_prime::from_long_to_bit_vector_reverse(long num,vector<bool>& temp1)
{
  int i;
  for (i=0; num; num>>=1, i++){
  temp1.push_back(bool((num&1)?(1):(0))) ;
  };
};
//***************************************************************************
long big_prime::mod(long a,long p){
//if (a<1000){return a%p;}else{
long sum=0;
for (int i=0; a; a>>=1, i++){
if (bool((a&1)?(1):(0))==1){
sum=sum+(long(pow(2,i)) % (p));
```

```
};
};
return sum%p;
};
//};
//*************************************************************************
void big_prime::from_vectro_to_memo(TMemo *temp1,vector<long>& temp2){
AnsiString s;
char string[25];
for (int i=0;i<temp2.size();i++){
//temp2[i]
itoa(temp2[i], string, 2);
s=string;
if (ismersen(temp2[i])) s=s+"  !!!Число Мерсена !!!";
if (isferma(temp2[i])) s=s+"  !!!Число Ферма !!!";
temp1->Lines->Add(IntToStr(temp2[i])+" "+s);
};
};
//*************************************************************************
bool big_prime::mod_test(vector<long>& mas1,vector<long>& mas2,vector<long>& mas3, long squarelimit){
bool have_null;
bool limit_out=false;
long temp;
mas3.resize(mas1.size());
for (int i=0;(i<mas1.size())&&(!limit_out);i++){
limit_out=(mas2[i]>squarelimit+1);
temp=(mas1[i]) % (mas2[i]);
if (temp==0)have_null=true;
if (temp==0){
mas3[i]=mas1[i]+2;
}else {
mas3[i]=temp;
};
```

```
};
return have_null;
};
//*************************************************************************
bool big_prime::mod_test(vector<long>& mas1,vector<long>& mas2,vector<long>& mas3){
bool have_null;
long temp;
mas3.resize(mas1.size());
for (int i=0;(i<mas1.size());i++){
temp=(mas1[i]) % (mas2[i]);
if (temp==0)have_null=true;
if (temp==0){
mas3[i]=mas1[i]+2;
}else {
mas3[i]=temp;
};
};
return have_null;
};
//*************************************************************************
bool big_prime::mod_test(long num1,vector<long>& mas1,vector<long>& mas2){
long num2=sqrt(num1);
vector<long> mas3;
mas3.resize(mas1.size());
for(int i=0;i<mas1.size();i++){
mas3[i]=num1;
};
return mod_test(mas3,mas1,mas2,num2);
};
//*************************************************************************
long big_prime::modexp(long a,long b,long p){
long temp1,dob=mod(a,p);
for (int i=0; b; b>>=1, i++){
if (bool((b&1)?(1):(0))==1){
```

```
temp1=dob;
dob=modmul(a,temp1,p);
};
};
return dob%p;
};
//***************************************************************************
long nsd(long a,long b,vector<long>& mas1){
int i=0;
while ((mas1[i]<=a)&&(mas1[i]<=b)&&i<=mas1.size()){
i++;
};
};
//***************************************************************************
long big_prime::modmul(long a,long b,long p){
long sum=0;
long b1=b;
for (int i=0; a; a>>=1, i++){
 if (bool((a&1)?(1):(0))==1){
// Form1->TestingMemo->Lines->Add(IntToStr(i)+" біт 1 числа "+a);
   b=b1;
   for (int i1=0; b; b>>=1, i1++){
      if (bool((b&1)?(1):(0))==1){
 //      Form1->TestingMemo->Lines->Add(IntToStr(i1)+" біт 1 числа "+b);
      sum=sum+(mod(pow(2,i+i1),p));
   //    Form1->TestingMemo->Lines->Add(IntToStr(sum));
      };
   };
 };
};
return mod(sum,p);
};
void big_prime::fill_vector_from_interval(long num1,long num2,vector<long>& mas1){
for(long i=0;i<num2-num1;i++){
```

```
mas1[i]=num1+i;
};
};
bool big_prime::rulepermit(long inumber){
bool is=true;
for (int i;i<rules.size();i++){
if ((inumber-rules[i].number)==rules[i].modul){
rules[i].number=inumber;
is=false;
};
};
return is;
};
void big_prime::set_prime_numbers(vector<long>& prime_numbers){
prime_numbers.resize(3);
prime_numbers[0]=2;
prime_numbers[1]=3;
prime_numbers[2]=5;
//prime_numbers[3]=7;
};
void big_prime::fill_vector_by_number(long num,vector<long>& mas1,long size){
mas1.resize(size);
for(int i=0;i<mas1.size();i++){mas1[i]=num;};
};
bool big_prime::ismersen(long a){
bool is=true;
for (int i=0; a; a>>=1, i++){
if (bool((a&1)?(1):(0))!=1){
is=false;
};
};
return is;
};
bool big_prime::isferma(long a){
```

```
int countone=0;
bool is=false;
bool isbegin=false;
if ((bool((a&1)?(1):(0))==1))isbegin=true;
for (int i=0; a; a>>=1, i++){
if (bool((a&1)?(1):(0))==1){
countone++;
};
};
if (isbegin&&countone==2)is=true;
return is;
};
//---------------------------------------------------------------------------
void big_prime::testing_number(long number,vector<long>& prime_numbers,TMemo *temp1){
  set_prime_numbers(prime_numbers);
  //testing information++++++++++
 // temp1->Lines->Add("результати заповнення простими числами");
 // from_vectro_to_memo(temp1,prime_numbers);
  //testing information++++++++++
  vector<long> ostacha;
  vector<long> next_ostacha;
  fill_vector_by_number(7,ostacha,1);
  //testing information++++++++++
//  temp1->Lines->Add("результати заповнення вектора тестовим стартовим числом");
//  from_vectro_to_memo(temp1,ostacha);
  //testing information++++++++++
  bool have_null;
  bool limit_out=false;
  long temp,squarelimit;
  //testing information++++++++++
//  temp1->Lines->Add("число з якого починається пошук - inumber"+IntToStr(inumber) + " пошук буде відбуватися до - number "+ IntToStr(number));
  //testing information++++++++++
   Form1->ProgressBar1->Max=number;
```

```
for (inumber=7;inumber<=number;inumber=inumber+2){
   Form1->ProgressBar1->Position=inumber;
  //testing information++++++++++
// temp1->Lines->Add("");
//  temp1->Lines->Add("число яке перевіряється - inumber "+IntToStr(inumber));
  //testing information++++++++++
if (rulepermit(inumber)){
//temp1->Lines->Add();
//*****************************
      have_null=false;
      limit_out=false;
      squarelimit=int(sqrt(inumber))+1;
      int i=0;
   //testing information++++++++++
 // temp1->Lines->Add("корінь з числа +1 - squarelimit "+IntToStr(squarelimit));
  //testing information++++++++++
  //       next_ostacha.resize(ostacha.size());
    //testing information++++++++++
  //temp1->Lines->Add("перевірка на остачу нуль в циклі до ostacha.size() або ж до перевищення ліміту "+IntToStr(ostacha.size()));
  //testing information++++++++++
   //Form1->ProgressBar2->Max=squarelimit;
      while (!limit_out)
                                          // for (int i=0;(i<ostacha.size())or(!limit_out);i++)
      {
  //      Form1->ProgressBar2->Position=prime_numbers[i];
      limit_out=(prime_numbers[i+1]>squarelimit);
                                         //testing information++++++++++
                                         // temp1->Lines->Add("статус  limit_out - "+BoolToStr(limit_out));
                                        //testing information++++++++++
      if (ostacha.size()-1>=i){                                  //  if (ostacha[i]>=prime_numbers[i]){
      temp=mod((ostacha[i]) , (prime_numbers[i]));
      }else{
```

```
            ostacha.resize(ostacha.size()+1);
            ostacha[ostacha.size()-1]=inumber;
            temp=mod((ostacha[i]) , (prime_numbers[i]));
            };
                                                //;}else{temp=0;};
                                                //testing information++++++++++
          //  temp1->Lines->Add(IntToStr(ostacha[i])+" mod "+IntToStr(prime_numbers[i])+" =
"+IntToStr(temp));
                                               //testing information++++++++++
            if (temp==0){
            ostacha[i]=2;
            have_null=true;                                 //testing information++++++++++
         //   temp1->Lines->Add("temp==0 залишок для наступного числа "+IntToStr(ostacha[i]));
            }else {
            ostacha[i]=temp+2;
       //     temp1->Lines->Add("temp!=0 залишок для наступного числа "+IntToStr(ostacha[i]));
            };
            i++;
            };
           if (!have_null){
           prime_numbers.resize(prime_numbers.size()+1);
           prime_numbers[prime_numbers.size()-1]=inumber;
           have_null=false;
       //testing information++++++++++
       // temp1->Lines->Add("
       // жодна остача не дорівнює нулю нове просте число додане до списку - "+
       // temp1->Lines->Add(IntToStr(inumber));
        //+" ************************************************************");
        //testing information++++++++++
           };
      }else {for (int j=0;j<ostacha.size();j++)ostacha[j]=ostacha[j]+2;};
      };
      };
```

```
__fastcall big_prime::big_prime(bool CreateSuspended)
        : TThread(CreateSuspended)
{
}
//---------------------------------------------------------------------------
void __fastcall big_prime::Execute()
{
Form1->TestingMemo->Clear();
TDateTime DateTime = Time();  // store the current date and time
AnsiString str = TimeToStr(DateTime); // convert the time to a string
Form1->TestingMemo->Lines->Add(str);
testing_number(StrToInt(Form1->Edit3->Text),prime_list,Form1->TestingMemo);
from_vectro_to_memo(Form1->TestingMemo,prime_list);
DateTime = Time();
str = TimeToStr(DateTime); // convert the time to a string
Form1->TestingMemo->Lines->Add(str);
Form1->prime_list=prime_list;
}
//---------------------------------------------------------------------------
```

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit3.h"
#include "Unit1.h"
#pragma package(smart_init)
//---------------------------------------------------------------------------

//   Important: Methods and properties of objects in VCL can only be
//   used in a method called using Synchronize, for example:
//
//      Synchronize(UpdateCaption);
//
//   where UpdateCaption could look like:
//
//      void __fastcall remaining::UpdateCaption()
//      {
//        Form1->Caption = "Updated in a thread";
//      }
//---------------------------------------------------------------------------
long remaining::mod(long a,long b){
long temp=a%b;
return temp;
};
//---------------------------------------------------------------------------

void remaining::from_vector_to_table_string(vector<long>& mas1,TStringGrid *StringGrid1){
for(int i=0;i<mas1.size();i++){
if (i>=StringGrid1->ColCount)StringGrid1->ColCount++;
StringGrid1->Cells[i][StringGrid1->RowCount]=IntToStr(mas1[i]);};
StringGrid1->RowCount++;
};
```

```
void remaining::get_remainders(long n,long prime,vector<long>& remainders){
//remainders.resize(n+1);
bool haveretry=false;
int i=1;
remainders.resize(2);
remainders[0]=prime;
remainders[i]=mod(pow(2,i),prime);



while ((i<=n)&&(haveretry==false)){
i++;
remainders.resize(remainders.size()+1);
remainders[i]=mod(pow(2,i),prime);
haveretry=remainders[i]==remainders[1];


};
};

void remaining::build_table_remainder(long maxn,long maxprime,vector<long>& primes,TStringGrid
*StringGrid1){


StringGrid1->Cells[0][StringGrid1->RowCount]="Прості числа";
for(int i=0;i<=maxn;i++){
if (i>=StringGrid1->ColCount)StringGrid1->ColCount++;
if (i>0)StringGrid1->Cells[i][StringGrid1->RowCount]="2^"+IntToStr(i);
};
StringGrid1->RowCount++;




    int i=0;
    vector<long> remainders;
    while (i<primes.size()&&primes[i]<=maxprime){
```

```
        //  Form1->Memo3->Lines->Add("Get remainders for "+IntToStr(primes[i]));
          get_remainders(maxn,primes[i],remainders);
        //  Form1->Memo3->Lines->Add("Size of remainder "+IntToStr(remainders.size()));
          from_vector_to_table_string(remainders,StringGrid1);
          i++;
         // remainders.
          };
      };



      __fastcall remaining::remaining(bool CreateSuspended)
            : TThread(CreateSuspended)
      {
      }
      //---------------------------------------------------------------------------
      void __fastcall remaining::Execute()
      {
      Form1->StringGrid1->ColCount=1;
      Form1->StringGrid1->RowCount=1;
      //get_remainders(StrToInt(Form1->Edit4->Text),3,remainders_table[0].remainders);
       build_table_remainder(StrToInt(Form1->Edit2->Text),StrToInt(Form1->Edit4->Text),Form1->prime_list,Form1->StringGrid1);
      }
      //---------------------------------------------------------------------------
```

Додаток Б

Програмний код алгоритмів пошуку найбільшого спільного дільника

```
#include <vcl.h>
#pragma hdrstop

#include "Unit4.h"
#include "Unit1.h"
#include "Unit2.h"
#pragma package(smart_init)
//---------------------------------------------------------------------------

//   Important: Methods and properties of objects in VCL can only be
//   used in a method called using Synchronize, for example:
//
//      Synchronize(UpdateCaption);
//
//   where UpdateCaption could look like:
//
//      void __fastcall TChrtBuilder::UpdateCaption()
//      {
//        Form1->Caption = "Updated in a thread";
//      }


long TChrtBuilder::GetChargePartNumber(long number,int charge_from,int chrge_to){
long num=0;
if (charge_from<chrge_to){

char string1[(sizeof(long)*8)+2];
AnsiString s;

itoa(number,string1, 2);
```

```
s=string1;
if (chrge_to>s.Length())chrge_to=s.Length()-1;
if (charge_from<0)charge_from=0;
if (charge_from>s.Length())charge_from=s.Length();
//if (charge_from=chrge_toLength())charge_from=s.Length();
//delete string1;
s=s.SubString(s.Length()-chrge_to,chrge_to-charge_from+1);

//Form1->Memo2->Lines->Add(string1);
//Form1->Memo2->Lines->Add(s1);
//Form1->Memo2->Lines->Add("length = "+IntToStr(s1.Length()));
for (int i =1;i<=s.Length();i++){
if (s.SubString(i,1)=='1'){
     num=num+pow(2,s.Length()-i);
    // Form1->Memo2->Lines->Add("s1[i] = "+s1.SubString(i,1));
    // Form1->Memo2->Lines->Add("power = "+IntToStr(s1.Length()-i));
};
};};
return num;



};

//---------------------------------------------------------------------------
void TChrtBuilder::addseries(TChart *Chart1,long from,long to,bool prime_only,vector<long>&
primes,bool discharge_limit,int charge_from,int chrge_to,bool isfermanumber,bool ismersennumber,TColor
Color1){
TLineSeries *series1;
big_prime *prime1;
int point1=0;
series1 = new TLineSeries(Chart1);
//series1->Name=name;
int i=0;
if (prime_only){
```

```
        while (i<=primes.size()&&primes[i]<=to){
            if (primes[i]>from&&primes[i]<to){
                if (discharge_limit==false){
                if (!isfermanumber&&!ismersennumber)series1->Add(primes[i],primes[i],Color1);
                if (isfermanumber)  if (prime1->isferma(primes[i])) series1->Add(primes[i],primes[i],Color1);
                if (ismersennumber) if (prime1->ismersen(primes[i])) series1->Add(primes[i],primes[i],Color1);
                }else{
                if (!isfermanumber&&!ismersennumber)series1->Add(primes[i],GetChargePartNumber(primes[i],charge_from,chrge_to),Color1);
                if (isfermanumber)  if (prime1->isferma(primes[i]))series1->Add(primes[i],GetChargePartNumber(primes[i],charge_from,chrge_to),Color1);
                if (ismersennumber) if (prime1->ismersen(primes[i]))series1->Add(primes[i],GetChargePartNumber(primes[i],charge_from,chrge_to),Color1);

                };
            };
        i++;
        };

    }else{
        for(i=from;i<=to;i++){
        if (isfermanumber)  if (prime1->isferma(i))series1->Add(i,i,Color1);

        if (ismersennumber) if (prime1->ismersen(i))series1->Add(i,i,Color1);

        if (!isfermanumber&&!ismersennumber)  series1->Add(i,i,Color1);

        };

    };
```

```
//series1->Marks->Style;
series1->Marks->Visible=true;
Chart1->AddSeries(series1);




};



__fastcall TChrtBuilder::TChrtBuilder(bool CreateSuspended)
        : TThread(CreateSuspended)
{
}
//---------------------------------------------------------------------------
void __fastcall TChrtBuilder::Execute()
{
addseries(Form1->Chart1,StrToInt(Form1->Edit5->Text),StrToInt(Form1->Edit6->Text),Form1->CheckBox3->Checked,Form1->prime_list,Form1->CheckBox4->Checked,StrToInt(Form1->Edit7->Text),StrToInt(Form1->Edit8->Text),Form1->CheckBox1->Checked,Form1->CheckBox2->Checked,Form1->Color1);
Form1->ListBox1->Clear();
for (int i=0;i<Form1->Chart1->SeriesCount();i++){
Form1->ListBox1->Items->Add("series "+IntToStr(i));
};
}
//---------------------------------------------------------------------------
```