Приложение А
Листинг программы

```
unit Unit7;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Menus, Vcl.StdCtrls,
  Vcl.Imaging.jpeg, Vcl.ExtCtrls;

type
  TMenu2 = class(TForm)
    Image1: TImage;
    Label1: TLabel;
    Button1: TButton;
    Button3: TButton;
    MainMenu1: TMainMenu;
    N1: TMenuItem;
    N2: TMenuItem;
    N3: TMenuItem;
    N5: TMenuItem;
    N4: TMenuItem;
    N7: TMenuItem;
    Label2: TLabel;
    Label3: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure N2Click(Sender: TObject);
    procedure N3Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure N4Click(Sender: TObject);
    procedure N5Click(Sender: TObject);
    procedure N7Click(Sender: TObject);
    procedure FormKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
    procedure FormActivate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Menu2: TMenu2;

implementation

{$R *.dfm}

uses unit6, unit5, Unit2, Unit1;


procedure TMenu2.Button1Click(Sender: TObject);
begin
  Grafik.Show;
  Hide;
end;

procedure TMenu2.Button3Click(Sender: TObject);
begin
  Close;
end;

procedure TMenu2.FormActivate(Sender: TObject);
begin
Label2.Caption:='Квадратичная функция это - функция,' +#13#10+'  которую можно записать формулой' +#13#10+'  вида y = ax2 + bx + c,' +#13#10+ '  где x – независимая переменная,' +#13#10+'  a, b и c – некоторые числа, причем a≠0.';
Label3.Caption:='Исследование функции это - определении ' +#13#10+'  основных параметров заданной функции. ' +#13#10+'  Одной из целей исследования является ' +#13#10+'  построение графика функции.'
end;

procedure TMenu2.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if Key=VK_F1 then
    begin
      winExec('hh HelpDemo.chm', SW_RESTORE);
    end;
end;

procedure TMenu2.N2Click(Sender: TObject);
begin
  Developer.Show;
end;

procedure TMenu2.N3Click(Sender: TObject);
begin
  Programm.Show;
end;

procedure TMenu2.N4Click(Sender: TObject);
begin
Close;
end;

procedure TMenu2.N5Click(Sender: TObject);
begin
  Grafik.Show;
  Hide;
end;

procedure TMenu2.N7Click(Sender: TObject);
begin
winExec('hh HelpDemo.chm', SW_RESTORE);
end;

end.

unit Unit8;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, Vcl.StdCtrls,
  Vcl.Imaging.jpeg, Vcl.Imaging.pngimage;

type
  TError = class(TForm)
    Label1: TLabel;
    Button1: TButton;
    Image1: TImage;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Error: TError;

implementation

{$R *.dfm}

procedure TError.Button1Click(Sender: TObject);
begin
Close;
end;

end.

unit Unit2;
```

```
interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, VCLTee.TeeGDIPlus, VCLTee.TeEngine,
  VCLTee.Series, VCLTee.TeeProcs, VCLTee.Chart, Vcl.ExtCtrls, Vcl.Imaging.jpeg,
  Vcl.Imaging.pngimage, Vcl.Menus, Math;

type
  TGrafik = class(TForm)
    Button1: TButton;
    Edita: TEdit;
    Editb: TEdit;
    Editc: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Фон: TImage;
    Button3: TButton;
    Chart1: TChart;
    MainMenu1: TMainMenu;
    N1: TMenuItem;
    N2: TMenuItem;
    Edit1: TEdit;
    Edit2: TEdit;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Button5: TButton;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    Edit3: TEdit;
    Label14: TLabel;
    Label15: TLabel;
    Edit4: TEdit;
    Edit5: TEdit;
    Label16: TLabel;
    Label17: TLabel;
    Button4: TButton;
    Button2: TButton;
    Button6: TButton;
    LabelCoordinates: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure N1Click(Sender: TObject);
    procedure FormKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
    procedure N2Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure Chart1MouseMove(Sender: TObject; Shift: TShiftState; X,
      Y: Integer);
    procedure FormActivate(Sender: TObject);
    procedure Chart1MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);

  private
  public
  end;

var
  Grafik: TGrafik;

implementation

{$R *.dfm}

uses unit7, Unit8;

procedure TGrafik.Button1Click(Sender: TObject);
var
  a, b, c, x1, x2,y1,y2, h, maxY, minY, margin: Double;
  x, y: Double;
  Series: TLineSeries;
  Discriminant: Double;
  VertexX, VertexY: Double;
  AxisOfSymmetry: Double;
  Root1, Root2: Double;
  CenterLineX, CenterLineY :TLineSeries;
  IntersectionPoints: TPointSeries;
  XIntersection, YIntersection: Double;
begin
  try
    a := StrToFloat(EditA.Text);
    b := StrToFloat(EditB.Text);
    c := StrToFloat(EditC.Text);
    x1 := StrToFloat(Edit1.Text);
    x2 := StrToFloat(Edit2.Text);
    y1 := StrToFloat(Edit4.Text);
    y2 := StrToFloat(Edit5.Text);
    h  := StrToFloat(Edit3.Text);

  // Обновляем график
  Chart1.Invalidate;
    if h<=0 then
  begin
     ShowMessage('Шаг должен быть положительным числом.');
      Exit;
    end;

  if x1 >= x2 then
   begin
    ShowMessage('Начальное значение X должно быть меньше конечного значения X.');
     Exit;
   end;
  except
    on E: EConvertError do
    begin
      Error.Show;
      Edita.Text:='';
      Editb.Text:='';
      Editc.Text:='';
      Edit1.Text := '';
      Edit2.Text := '';
      Edit3.Text := '';
      Edit4.Text := '';
      Edit5.Text := '';
      Exit;
    end;
  end;
  Series := TLineSeries.Create(Self);
  Chart1.AddSeries(Series);

  x := x1;
  while x <= x2 do
  begin
    y := a * x * x + b * x + c;
    Series.AddXY(x, y);
    x := x + h;
  end;

Chart1.BottomAxis.Automatic := False;
Chart1.LeftAxis.Automatic := False;
Chart1.BottomAxis.Minimum := x1; // Пользователь задает минимальное значение оси X
Chart1.BottomAxis.Maximum := x2; // Пользователь задает максимальное значение оси X
```

```
Chart1.LeftAxis.Minimum := y1;    // Пользователь задает минимальное значение оси Y
Chart1.LeftAxis.Maximum := y2;    // Пользователь задает максимальное значение оси Y
Chart1.BottomAxis.Increment := (Chart1.BottomAxis.Maximum - Chart1.BottomAxis.Minimum) / 10;
Chart1.LeftAxis.Increment := (Chart1.LeftAxis.Maximum - Chart1.LeftAxis.Minimum) / 10;
Chart1.MarginBottom := 0;
Chart1.MarginTop := 0;
Chart1.MarginLeft := 0;
Chart1.MarginRight := 0;
CenterLineX := TLineSeries.Create(Self);
CenterLineY := TLineSeries.Create(Self);
CenterLineX.AddXY(Chart1.BottomAxis.Minimum, 0);
CenterLineX.AddXY(Chart1.BottomAxis.Maximum, 0);
CenterLineY.AddXY(0, Chart1.LeftAxis.Minimum);
CenterLineY.AddXY(0, Chart1.LeftAxis.Maximum);
CenterLineX.SeriesColor := clRed; // Цвет оси X
CenterLineY.SeriesColor := clBlue; // Цвет оси Y
Chart1.AddSeries(CenterLineX);
Chart1.AddSeries(CenterLineY);
// Проверяем, пересекает ли график оси
IntersectionPoints := TPointSeries.Create(Chart1);
Chart1.AddSeries(IntersectionPoints);
XIntersection := -b / (2 * a);  // Это значение для вершины параболы
YIntersection := c;  // Это значение функции при X = 0
// Проверка пересечения с осью Y (X = 0)
if (0 >= Chart1.BottomAxis.Minimum) and (0 <= Chart1.BottomAxis.Maximum) and (YIntersection >= Chart1.LeftAxis.Minimum) and (YIntersection <= Chart1.LeftAxis.Maximum) then
begin
  IntersectionPoints.AddXY(0, YIntersection);
end;
// Проверка пересечения с осью X (Y = 0)
Discriminant := b * b - 4 * a * c;
if Discriminant >= 0 then
begin
  Root1 := (-b + Sqrt(Discriminant)) / (2 * a);
  Root2 := (-b - Sqrt(Discriminant)) / (2 * a);
  if (Root1 >= Chart1.BottomAxis.Minimum) and (Root1 <= Chart1.BottomAxis.Maximum) then
  begin
    IntersectionPoints.AddXY(Root1, 0);
  end;
  if (Root2 >= Chart1.BottomAxis.Minimum) and (Root2 <= Chart1.BottomAxis.Maximum) then
  begin
    IntersectionPoints.AddXY(Root2, 0);
  end;
end;
end;

procedure TGrafik.Button2Click(Sender: TObject);
var
SaveDialog1,SaveDialog2: TSaveDialog;
Results: TStringList;
ChartData: TStringList;
i: Integer;
begin //сохранение исследования
 SaveDialog1 := TSaveDialog.Create(Self);
  try
    SaveDialog1.Filter := 'Text Files|*.txt';
    if SaveDialog1.Execute then
    begin
      Results := TStringList.Create;
      try
        Results.Add('Исследование квадратичной функции:');
        Results.Add(Label7.Caption);
        Results.Add(Label8.Caption);
        Results.Add(Label9.Caption);
        Results.Add(Label10.Caption);
        Results.Add(Label11.Caption);
        Results.Add(Label12.Caption);
        Results.Add(Label13.Caption);
        Results.Add(Label15.Caption);
        Results.Add(Edita.Text);
        Results.Add(Editb.Text);
        Results.Add(Editc.Text);
        Results.Add(Edit1.Text);
        Results.Add(Edit2.Text);
        Results.Add(Edit3.Text);
        Results.Add(Edit4.Text);
        Results.Add(Edit5.Text);
        Results.SaveToFile(SaveDialog1.FileName);
      finally
        Results.Free;
      end;
    end;
  finally
    SaveDialog1.Free;
  end;
   SaveDialog2 := TSaveDialog.Create(Self);
  try
    SaveDialog2.Filter := 'Data Files|*.dat';
    if SaveDialog2.Execute then
    begin
      ChartData := TStringList.Create;
      try
        //данные графика
        if Chart1.SeriesCount > 0 then
        begin
          for i := 0 to Chart1.Series[0].Count - 1 do
          begin
            ChartData.Add(Format('%f;%f', [Chart1.Series[0].XValues[i], Chart1.Series[0].YValues[i]]));
          end;
        end;
        ChartData.SaveToFile(SaveDialog2.FileName);
      finally
        ChartData.Free;
      end;
    end;
  finally
    SaveDialog2.Free;
  end;

   if SaveDialog1.Execute then
  begin
    Chart1.SaveToBitmapFile(SaveDialog1.FileName);
  end;

end;

procedure TGrafik.Button3Click(Sender: TObject);
begin
  Menu2.Show;
  Close;
end;

procedure TGrafik.Button4Click(Sender: TObject);
begin
  Chart1.SeriesList.Clear;

  EditA.Text := '';
  EditB.Text := '';
  EditC.Text := '';
  Edit1.Text := '';
  Edit2.Text := '';
  Edit3.Text := '';
  Edit4.Text := '';
  Edit5.Text := '';

  Chart1.Invalidate;
  Chart1.Update;

  Label7.Caption:='';
  Label8.Caption:='';
  Label9.Caption:='';
```

```
  Label10.Caption:='';
  Label11.Caption:='';
  Label12.Caption:='';
  Label13.Caption:='';
  Label15.Caption:='';

end;

procedure TGrafik.Button5Click(Sender: TObject);
var
  i:integer;
  IntersectionPoints: TPointSeries;
  XIntersection, YIntersection: Double;
  a, b, c: Double;
  Discriminant: Double;
  VertexX, VertexY: Double;
  AxisOfSymmetry: Double;
  Root1, Root2: Double;
  PointsList: TStringList;
begin
  try
    a := StrToFloat(EditA.Text);
    b := StrToFloat(EditB.Text);
    c := StrToFloat(EditC.Text);
  except
    on E: EConvertError do
    begin
      Error.Show;
      Edita.Text:='';
      Editb.Text:='';
      Editc.Text:='';
      Edit1.Text:='';
      Edit2.Text:='';
      Edit3.Text:='';
      Edit4.Text:='';
      Edit5.Text:='';
      Exit;
    end;
  end;
  Discriminant := b * b - 4 * a * c;

  VertexX := -b / (2 * a);
  VertexY := a * Sqr(VertexX) + b * VertexX + c;

  if a > 0 then
    Label7.Caption := 'Функция убывает при x < ' + Format-
FloatFloat('0.000', VertexX) + ' и возрастает при x > ' + Format-
FloatFloat('0.000', VertexX)
  else if a < 0 then
    Label7.Caption := 'Функция возрастает при x < ' +
FormatFloat('0.000', VertexX) + ' и убывает при x > ' +
FormatFloat('0.000', VertexX)
  else
    Label7.Caption := 'Функция имеет экстремум в точке x
= ' + FormatFloat('0.000', VertexX);

  if a <> 0 then
  begin
    Label8.Caption := 'Вершина параболы: (' + Format-
FloatFloat('0.000', VertexX) + ', ' + FormatFloat('0.000', VertexY)
+ ')';
    AxisOfSymmetry := -b / (2 * a);
    Label9.Caption := 'Ось симметрии: x = ' + Format-
FloatFloat('0.000', AxisOfSymmetry);
  end
  else
  begin
    Label8.Caption := 'a = 0, функция не является полной
квадратичной';
    Exit;
  end;

   if a > 0 then
    Label10.Caption := 'Ветви параболы направлены вверх.'
  else
    Label10.Caption := 'Ветви параболы направлены вниз.';

  if (a=0) or (b=0) or (c=0) then
    Label11.Caption := 'Это не полная квадратичная функ-
ция.'
  else if Discriminant > 0 then
    Label11.Caption := 'Уравнение имеет два вещественных корня.'
  else if Discriminant = 0 then
    Label11.Caption := 'Уравнение имеет один вещественный корень.'
  else
    Label11.Caption := 'Уравнение не имеет вещественных корней.';

  if Discriminant > 0 then
  begin
    Root1 :=(-b + Sqrt(Discriminant)) / (2 * a);
    Root2 := (-b - Sqrt(Discriminant)) / (2 * a);
    Label12.Caption := 'Корень 1: ' + FormatFloat('0.000'
,Root1) + ', Корень 2: ' + FormatFloat('0.000' ,Root2);
  end
  else if Discriminant = 0 then
  begin
    Root1 := -b / (2 * a);
    Label12.Caption := 'Единственный корень: ' + FloatTo-
Str(Root1);
  end
  else
    Label12.Caption := 'Нет вещественных корней';
    if a > 0 then
  begin
    if Discriminant > 0 then
      Label13.Caption := 'Функция положительна при x < ' +
FormatFloat('0.000' ,Root1) + ' или x > ' + Format-
Float('0.000' ,Root2) + #13#10+' и отрицательна при ' +
FormatFloat('0.000' ,Root1) + ' < x < ' + FormatFloat('0.000'
,Root2)
    else if Discriminant = 0 then
      Label13.Caption := 'Функция положительна при x ≠ ' +
FloatToStr(Root1) +#13#10+ ' и отрицательна в точке x = '
+ FloatToStr(Root1)
    else
      Label13.Caption := 'Функция положительна для всех
x';
  end
  else if a < 0 then
  begin
    if Discriminant > 0 then
      Label13.Caption := 'Функция отрицательна при x < ' +
FormatFloat('0.000' ,Root1) + ' или x > ' + Format-
Float('0.000' ,Root2) +#13#10+ ' и положительна при '
+FormatFloat('0.000' ,Root1) + ' < x < ' + FormatFloat('0.000'
,Root2)
    else if Discriminant = 0 then
      Label13.Caption := 'Функция отрицательна при x ≠ ' +
FormatFloat('0.000' ,Root1) +#13#10+ ' и положительна в
точке x = ' + FormatFloat('0.000' ,Root1)
    else
      Label13.Caption := 'Функция отрицательна для всех x';
  end;

Discriminant := b * b - 4 * a * c;
  // Инициализация серии для точек пересечения
  IntersectionPoints := TPointSeries.Create(Self);
  Chart1.AddSeries(IntersectionPoints);
  // Точка пересечения с осью Y
  if (c >= Chart1.LeftAxis.Minimum) and (c <=
Chart1.LeftAxis.Maximum) then
  begin
    IntersectionPoints.AddXY(0, c);
  end;
  // Точки пересечения с осью X
  if Discriminant >= 0 then
  begin
    Root1 := (-b + Sqrt(Discriminant)) / (2 * a);
    Root2 := (-b - Sqrt(Discriminant)) / (2 * a);
```

```
    if (Root1 >= Chart1.BottomAxis.Minimum) and (Root1
<= Chart1.BottomAxis.Maximum) then
    begin
      IntersectionPoints.AddXY(Root1, 0);
    end;
    if (Root2 >= Chart1.BottomAxis.Minimum) and (Root2
<= Chart1.BottomAxis.Maximum) then
    begin
      IntersectionPoints.AddXY(Root2, 0);
    end;
  end;
  // Вывод точек пересечения
  PointsList := TStringList.Create;
  try
    if IntersectionPoints.Count > 0 then
    begin
      PointsList.Add('Точки пересечения с осями:');
      for i := 0 to IntersectionPoints.Count - 1 do
      begin
        PointsList.Add(Format('Точка %d: (%.2f, %.2f)', [i + 1,
IntersectionPoints.XValues[i], Intersection-
Points.YValues[i]]));
      end;
    end
    else
    begin
      PointsList.Add('На графике нет точек пересечения с
осями.');
    end;
    Label15.Caption := PointsList.Text;
  finally
    PointsList.Free;
  end;

  end;

procedure TGrafik.Button6Click(Sender: TObject);
var
OpenDialog1,OpenDialog2: TOpenDialog;
ChartData: TStringList;
i: Integer;
X, Y: Double;
LineSeries: TLineSeries;
Results: TStringList;
begin
 OpenDialog1 := TOpenDialog.Create(Self);//загрузка ис-
следования
  try
    OpenDialog1.Filter := 'Text Files|*.txt';
    if OpenDialog1.Execute then
    begin
      Results := TStringList.Create;
      try
        Results.LoadFromFile(OpenDialog1.FileName);
        if Results.Count >= 7 then
        begin
          Label7.Caption := Results[1];
          Label8.Caption := Results[2];
          Label9.Caption := Results[3];
          Label10.Caption := Results[4];
          Label11.Caption := Results[5];
          Label12.Caption := Results[6];
          Label13.Caption := Results[7];
          Label15.Caption := Results[8];
          Edita.Text := Results[11];
          Editb.Text := Results[12];
          Editc.Text := Results[13];
          Edit1.Text := Results[14];
          Edit2.Text := Results[15];
          Edit3.Text := Results[16];
          Edit4.Text := Results[17];
          Edit5.Text := Results[18];
        end;
      finally
        Results.Free;
      end;
    end;
  finally
    OpenDialog1.Free;
  end;
    OpenDialog2 := TOpenDialog.Create(Self);
  try
    OpenDialog2.Filter := 'Data Files|*.dat';
    if OpenDialog2.Execute then
    begin
      ChartData := TStringList.Create;
      try
        ChartData.LoadFromFile(OpenDialog2.FileName);
        if ChartData.Count >= 6 then
        begin
          //данные графика
          LineSeries := TLineSeries.Create(Self);
          Chart1.AddSeries(LineSeries);
          for i := 6 to ChartData.Count - 1 do
          begin
            X := StrToFloat(Copy(ChartData[i], 1, Pos(';',
ChartData[i]) - 1));
            Y := StrToFloat(Copy(ChartData[i], Pos(';',
ChartData[i]) + 1, Length(ChartData[i])));
            LineSeries.AddXY(X, Y);
          end;
        end
        else
        begin
          ShowMessage('Файл данных поврежден или имеет
неправильный формат.');
        end;
      finally
        ChartData.Free;
      end;
    end;
  finally
    OpenDialog2.Free;
  end;
end;

procedure TGrafik.Chart1MouseDown(Sender: TObject;
Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
 if Button = mbRight then
  begin
    Abort;
  end;
end;

procedure TGrafik.Chart1MouseMove(Sender: TObject;
Shift: TShiftState; X, Y: Integer);
var
  XValue, YValue: Double;
begin
  // Преобразуем экранные координаты X и Y в значения
осей графика
  XValue := Chart1.BottomAxis.CalcPosPoint(X);
  YValue := Chart1.LeftAxis.CalcPosPoint(Y);

  // Отображаем координаты в LabelCoordinates
  LabelCoordinates.Caption := Format('X: %.2f, Y: %.2f',
[XValue, YValue]);
end;

procedure TGrafik.FormActivate(Sender: TObject);
begin
  Chart1.Zoom.Allow := False;
end;

procedure TGrafik.FormKeyDown(Sender: TObject; var
Key: Word;
  Shift: TShiftState);
begin
  if Key=VK_F1 then
    begin
```

```
    winExec('hh HelpDemo.chm', SW_RESTORE);
   end;
end;

procedure TGrafik.N1Click(Sender: TObject);
begin
Menu2.Close;
end;

procedure TGrafik.N2Click(Sender: TObject);
begin
winExec('hh HelpDemo.chm', SW_RESTORE);
end;

end.
unit Unit1;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Imaging.jpeg, Vcl.ExtCtrls,
  Vcl.Imaging.pngimage, Vcl.StdCtrls;

type
  TSpalshF = class(TForm)
    Image1: TImage;
    Timer1: TTimer;
    Label1: TLabel;
    Image2: TImage;
    Image3: TImage;
    Image4: TImage;
    procedure Timer1Timer(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  SpalshF: TSpalshF;

implementation

{$R *.dfm}

uses unit7;



procedure TSpalshF.Timer1Timer(Sender: TObject);
begin
Timer1.Enabled:=False;
end;

end.

unit Unit5;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, Vcl.Imaging.pngimage,
  Vcl.Imaging.jpeg, Vcl.StdCtrls, Vcl.OleCtrls, SHDocVw;

type
  TDeveloper = class(TForm)
    Image1: TImage;
    Image2: TImage;
    WebBrowser1: TWebBrowser;
    procedure FormCreate(Sender: TObject);
    procedure FormKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Developer: TDeveloper;

implementation

{$R *.dfm}


procedure TDeveloper.FormCreate(Sender: TObject);
var
  FilePath: string;
begin
FilePath := 'E:\Курсовые\Проект\Об авторе.htm';
FilePath := 'file:///' + StringReplace(FilePath, '\', '/', [rfReplaceAll]);
WebBrowser1.Navigate(FilePath);

end;

procedure TDeveloper.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if Key=VK_F1 then
    begin
      winExec('hh HelpDemo.chm', SW_RESTORE);
    end;
end;

end.

unit Unit6;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Imaging.jpeg, Vcl.ExtCtrls,
  Vcl.StdCtrls, Vcl.OleCtrls, SHDocVw;

type
  TProgramm = class(TForm)
    Image1: TImage;
    Label1: TLabel;
    WebBrowser1: TWebBrowser;
    procedure FormCreate(Sender: TObject);
    procedure FormKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Programm: TProgramm;

implementation

{$R *.dfm}

procedure TProgramm.FormCreate(Sender: TObject);
var
  FilePath: string;
begin
FilePath := 'E:\Курсовые\Проект\О программе.htm';
```

```
FilePath := 'file:///' + StringReplace(FilePath, '\', '/',
[rfReplaceAll]);
WebBrowser1.Navigate(FilePath);
end;

procedure TProgramm.FormKeyDown(Sender: TObject; var
Key: Word;
  Shift: TShiftState);
begin
  if Key=VK_F1 then
    begin
      winExec('hh HelpDemo.chm', SW_RESTORE);
    end;
end;

end.
```