unit MainUnit;

interface

uses

 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

 Dialogs, StdCtrls, ExtCtrls, TeeProcs, TeEngine, Chart, Series;

type

 TMainForm = class(TForm)

 edtX: TEdit;

 rgFunction: TRadioGroup;

 lblX: TLabel;

 lblY: TLabel;

 lblEps: TLabel;

 edtEps: TEdit;

 lblMaxIters: TLabel;

 edtMaxIters: TEdit;

 edtY: TEdit;

 btnEvaluate: TButton;

 chart: TChart;

 edtX0: TEdit;

 edtX1: TEdit;

 lblX0: TLabel;

 lblX1: TLabel;

 btnGraph: TButton;

 series: TLineSeries;

 edtIters: TEdit;

 Label1: TLabel;

 edtStep: TEdit;

 lblStep: TLabel;

 btnExit: TButton;

 procedure btnEvaluateClick(Sender: TObject);

 procedure btnGraphClick(Sender: TObject);

 procedure FormCreate(Sender: TObject);

 procedure btnExitClick(Sender: TObject);

 private

 eps: Double;

 iters: Integer;

 maxiters: Integer;

 function ml\_sin(x: double): double;

 function ml\_cos(x: double): double;

 public

 end;

var

 MainForm: TMainForm;

implementation

{$R \*.dfm}

function TMainForm.ml\_sin(x: double): double;

var

 numer, denom, sign, sum, old\_sum: double;

begin

 numer := x;

 denom := 1;

 sign := 1;

 sum := x;

 iters := 1;

 repeat

 old\_sum := sum;

 numer := numer \* x \* x;

 denom := denom \* (2\*iters) \* (2\*iters+1);

 sign := sign \* (-1);

 sum := sum + (sign \* numer/denom);

 inc(iters);

 until (abs(old\_sum - sum) < eps) or (iters > maxiters);

 result := sum;

end;

function TMainForm.ml\_cos(x: double): double;

var

 numer, denom, sign, sum, old\_sum: double;

begin

 numer := 1;

 denom := 1;

 sign := 1;

 sum := 1;

 iters := 1;

 repeat

 old\_sum := sum;

 numer := numer \* x \* x;

 denom := denom \* (2\*iters-1) \* (2\*iters);

 sign := sign \* (-1);

 sum := sum + (sign \* numer/denom);

 inc(iters);

 until (abs(old\_sum - sum) < eps) or (iters > maxiters);

 result := sum;

end;

procedure TMainForm.btnEvaluateClick(Sender: TObject);

var

 x, y: double;

begin

 try

 x := strToFloat(edtX.Text);

 eps := strToFloat(edtEps.Text);

 maxiters := strToInt(edtMaxIters.Text);

 case rgFunction.ItemIndex of

 0: y := ml\_sin(x);

 1: y := ml\_cos(x);

 else

 raise Exception.Create('Invariant failed: only sin and cos are allowed');

 end;

 edtY.Text := FloatToStrF(y, ffGeneral, 15, 15);

 edtIters.Text := IntToStr(iters);

 except

 on EConvertError do edtY.Text := 'Неверный формат ввода';

 end;

end;

procedure TMainForm.btnGraphClick(Sender: TObject);

var

 x0, x1, h, x, y: double;

begin

 try

 x0 := strToFloat(edtX0.Text);

 x1 := strToFloat(edtX1.Text);

 h := strToFloat(edtStep.Text);

 eps := strToFloat(edtEps.Text);

 maxiters := strToInt(edtMaxIters.Text);

 series.Clear;

 x := x0;

 while x <= x1 do begin

 case rgFunction.ItemIndex of

 0: y := ml\_sin(x);

 1: y := ml\_cos(x);

 else

 raise Exception.Create('Invariant failed: only sin and cos are allowed');

 end;

 series.AddXY(x, y);

 x := x + h;

 end;

 except

 on EConvertError do edtY.Text := 'Неверный формат ввода';

 end;

end;

procedure TMainForm.FormCreate(Sender: TObject);

begin

 rgFunction.ItemIndex := 0;

 edtEps.Text := FloatToStr(10e-12);

 edtMaxIters.Text := FloatToStr(100);

 edtX.Text := FloatToStr(pi/4);

 edtX0.Text := FloatToStr(0);

 edtX1.Text := FloatToStr(2\*pi);

 edtStep.Text := FloatToStr(pi/400);

end;

procedure TMainForm.btnExitClick(Sender: TObject);

begin

 Application.Terminate;

end;

end.