unit UnitUML;

interface

uses

Classes, SysUtils, FileUtil, TAGraph, TASeries, Forms, Controls, Graphics,

Dialogs, StdCtrls , TALegend;

type

TForm1 = class(TForm)

Button1:TButton;

Button2:TButton;

Button3: TButton;

Button4:TButton;

Chart1: TChart;

Chart1LineSeries1: TLineSeries;

ComboBox1: TComboBox;

Edit1: TEdit;

Edit2: TEdit;

Edit3: TEdit;

Edit4: TEdit;

Edit5: TEdit;

Edit6: TEdit;

Edit7: TEdit;

Edit8: TEdit;

Edit9: TEdit;

Label1: TLabel;

Label10: TLabel;

Label11: TLabel;

Label12: TLabel;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

Label5: TLabel;

Label6: TLabel;

Label7: TLabel;

Label8: TLabel;

Label9: TLabel;

SaveDialog1: TSaveDialog;

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure Button4Click(Sender: TObject);

procedure ComboBox1Change(Sender: TObject);

end;

implementation

{$R \*.lfm}

{ TForm1 }

procedure TForm1.ComboBox1Change(Sender: TObject);

begin

end;

procedure TForm1.Button1Click(Sender: TObject);

begin

end;

procedure TForm1.Button2Click(Sender: TObject);

begin

end;

procedure TForm1.Button4Click(Sender: TObject);

begin

end;

//Хранилище параметров движения

TParametresOfFly = packed record

a: array[1..5,1..5] of double;

c: array[1..14] of double;

e: array[1..3] of double;

w0: double;

alpha0: double;

V0: double;

H0: double;

teta0: double;

h: double;

tt: double;

end;

TFormBase = class;

TBorder = class;

TController = class;

TStorage = class;

//

//

//Абстракнтый класс фармы

TFormBase = class abstract (TForm)

//Набор значений Ci

StringGrid1: TStringGrid;

//Набор значнеий Ei

StringGrid2: TStringGrid;

Label1: TLabel;

//Время интегрировния

Edit1: TEdit;

//Сохранить и выйти

Button1: TButton;

//Решалка с-мы ду и построение графиков

Button2: TButton;

Chart1: TChart;

Series1: TLineSeries;

Chart2: TChart;

Chart3: TChart;

Chart4: TChart;

Chart5: TChart;

Series2: TLineSeries;

Series3: TLineSeries;

Series4: TLineSeries;

Series5: TLineSeries;

//Набор значений начальных условий

StringGrid3: TStringGrid;

Label2: TLabel;

//Шаг интегрирования

Edit2: TEdit;

//Загрузить параметры движения по умолчанию

Button3: TButton;

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure FormActivate(Sender: TObject);

procedure Button3Click(Sender: TObject);

end;

//

//

//Класс-граница

TBorder = class (TFormBase)

private

/// <directed>True</directed>

\_Controller: TController;

//Метод, обновляющий параметры движения на форме

procedure UpdateParam;

public

//Метод, закрывающий и сохраняющий параметры движения из формы

procedure SaveCloseButton;

//Метод, решающий уравнения и отрисовывающий графики

procedure SolveButton;

//Метод, связывающий границу с контроллером

procedure Connect(\_Controller\_: TController);

//Метод, отрисовывающий графику

procedure PlotGraph(f: array of double; series: TLineSeries);

//Метод, выравнивающий элементы на форме(задает начальные положения)

procedure FormActivate;

//Метод, загружающий деолтне параметры движения

procedure LoadDefaultParametres;

end;

//

//

//

TController=class

private

//указатель на класс-границу

/// <directed>True</directed>

\_Border: TBorder;

//указатель на класс-хранилище

/// <directed>True</directed>

\_Storage: TStorage;

//возвращает значение функции угловой скорости для данного момента времени

function w\_t(w,alpha,V,H,teta: double):double;

//возвращает значение функции угла атаки для данного момента времени

function alpha\_t(w,alpha,V,H,teta: double):double;

//возвращает значение функции скорости для данного момента времени

function V\_t(w,alpha,V,H,teta: double):double;

//возвращает значение функции высоты для данного момента времени

function H\_t(w,alpha,V,H,teta: double):double;

//возвращает значение функции угла тангажа для данного момента времени

function teta\_t(w,alpha,V,H,teta: double):double;

public

//Значения соответствующих величин от времени

w: array of double;

alpha: array of double;

V: array of double;

H: array of double;

teta: array of double;

//Метод, соединяющий с границей и хранилищем

procedure Connect(\_Border\_: TBorder; \_Storage\_: TStorage);

//Метод, выполняющий операции при закрытии формы

procedure Exit;

//Метод, решающий с-му ду

procedure Solve;

//Метод, загружающий параметры движения из хранилища

function GetParametresOfFly: TParametresOfFly;

//Метод, устанавливающий параметры движения

procedure SetParametresOfFly(p: TParametresOfFly);

//Метод, загружающий параметры по умолчанию

procedure LoadDefaultParametres;

//Метод, инициализующий загрузку параметров из файла

procedure LoadParamFromFile;

end;

//

//

//Класс-хранилище

TStorage=class

private

//Указатель на класс-контроллер

/// <directed>True</directed>

\_Controller: TController;

//Метод, считающий коэффицинты движения линейной системы по параметрам режима полета

procedure UpdateA;

public

//Хранилище параметров полета

param: TParametresOfFly;

//Конструктор =))

constructor Create;

//Метод, соединяющий с контроллером

procedure Connect(\_Controller\_: TController);

//Метод, записывающий текущие параметра движения в файл

procedure SaveParamToFile;

//Метод, устанавливающий рабочие(текущие) параметры движения

procedure SetParametresOfFly(p: TParametresOfFly);

//Метод, получающий текущие параметры движения

function GetParametresOfFly: TParametresOfFly;

//Метод, устанавливающий параметры по умолчанию как текущие

procedure SetDefaultParametres;

//Метод, загружащий параметры из файла и устанавливающий их по умолчанию

procedure LoadParamFromFile;

end;

var

// FormBase: TFormBase;

//Экземпляр класса-границы

Border: TBorder;

//Экземпляр класса-контроллера

Controller: TController;

//Экземпляр класса-хранилища

Storage: TStorage;

implementation

{$R \*.dfm}

//

// Методы класса интерфейса TFormBase

//

procedure TFormBase.Button1Click(Sender: TObject);

begin

Controller.Exit;

end;

procedure TFormBase.Button2Click(Sender: TObject);

begin

Border.SolveButton;

end;

procedure TFormBase.Button3Click(Sender: TObject);

begin

Border.LoadDefaultParametres;

end;

procedure TFormBase.FormActivate(Sender: TObject);

begin

Border.FormActivate;

end;

//

// Методы класса интерфейса-границы TBorder

//

procedure TBorder.Connect(\_Controller\_: TController);

begin

\_Controller:=\_Controller\_;

end;

procedure TBorder.SaveCloseButton;

begin

\_Controller.Exit;

end;

procedure TBorder.SolveButton;

var

p: TParametresOfFly;

i: integer;

begin

for i := 1 to 14 do p.c[i]:=StrToFloat(Border.StringGrid1.Cells[i-1,1]);

for i := 1 to 3 do p.e[i]:=StrToFloat(Border.StringGrid2.Cells[i-1,1]);

p.tt:=StrToFloat(Edit1.Text);

p.h:=StrToFloat(Edit2.Text);

p.w0:=StrToFloat(Border.StringGrid3.Cells[0,1]);

p.alpha0:=StrToFloat(Border.StringGrid3.Cells[1,1]);

p.V0:=StrToFloat(Border.StringGrid3.Cells[2,1]);

p.H0:=StrToFloat(Border.StringGrid3.Cells[3,1]);

p.teta0:=StrToFloat(Border.StringGrid3.Cells[4,1]);

\_Controller.SetParametresOfFly(p);

\_Controller.Solve;

Series1.Clear;

Series2.Clear;

Series3.Clear;

Series4.Clear;

Series5.Clear;

PlotGraph(Controller.w, Series1);

PlotGraph(Controller.alpha, Series2);

PlotGraph(Controller.V, Series3);

PlotGraph(Controller.H, Series4);

PlotGraph(Controller.teta, Series5);

end;

procedure TBorder.PlotGraph(f: array of double; series: TLineSeries);

var

i,l: integer;

begin

l:=High(f);

for i := 0 to l do series.AddXY(Storage.param.h\*i,f[i],'',clRed);

end;

procedure TBorder.UpdateParam;

var

i:integer;

p: TParametresOfFly;

begin

p:=\_Controller.GetParametresOfFly;

for i := 1 to 14 do StringGrid1.Cells[i-1,1]:=FloatToStr(p.c[i]);

for i := 1 to 3 do StringGrid2.Cells[i-1,1]:=FloatToStr(p.e[i]);

StringGrid3.Cells[0,1]:=FloatToStr(p.w0);

StringGrid3.Cells[1,1]:=FloatToStr(p.alpha0);

StringGrid3.Cells[2,1]:=FloatToStr(p.V0);

StringGrid3.Cells[3,1]:=FloatToStr(p.H0);

StringGrid3.Cells[4,1]:=FloatToStr(p.teta0);

Edit2.Text:=FloatToStr(p.h);

Edit1.Text:=FloatToStr(p.tt);

end;

procedure TBorder.FormActivate;

var

i:integer;

w:integer;

begin

\_Controller.LoadParamFromFile;

UpdateParam;

StringGrid1.Width:=14\*StringGrid1.DefaultColWidth+7\*2;

StringGrid1.Height:=2\*StringGrid1.DefaultRowHeight;

StringGrid2.Left:=14\*StringGrid1.DefaultColWidth+170;

StringGrid2.Width:=3\*StringGrid2.DefaultColWidth+3\*2;

StringGrid2.Height:=2\*StringGrid2.DefaultRowHeight;

StringGrid3.Width:=5\*StringGrid3.DefaultColWidth+5\*2;

StringGrid3.Height:=2\*StringGrid3.DefaultRowHeight;

for i:=1 to 14 do StringGrid1.Cells[i-1,0]:='c'+IntToStr(i);

for i:=1 to 3 do StringGrid2.Cells[i-1,0]:='e'+IntToStr(i);

StringGrid3.Cells[0,0]:='Wz0';

StringGrid3.Cells[1,0]:='Alpha0';

StringGrid3.Cells[2,0]:='V0';

StringGrid3.Cells[3,0]:='H0';

StringGrid3.Cells[4,0]:='Teta0';

w:=round((ClientWidth-60)/5);

Border.Align:=alNone;

Border.Height:=170+w;

Chart1.Left:=10;

Chart1.Height:=w;

Chart1.Width:=w;

Chart2.Left:=2\*10+w;

Chart2.Height:=w;

Chart2.Width:=w;

Chart3.Left:=3\*10+2\*w;

Chart3.Height:=w;

Chart3.Width:=w;

Chart4.Left:=4\*10+3\*w;

Chart4.Height:=w;

Chart4.Width:=w;

Chart5.Left:=5\*10+4\*w;

Chart5.Height:=w;

Chart5.Width:=w;

end;

procedure TBorder.LoadDefaultParametres;

begin

\_Controller.LoadDefaultParametres;

end;

//

// Методы класса контроллера TController

//

procedure TController.Connect(\_Border\_: TBorder; \_Storage\_: TStorage);

begin

\_Border:=\_Border\_;

\_Storage:=\_Storage\_;

end;

//

procedure TController.Exit;

begin

\_Storage.SaveParamToFile;

\_Border.Close;

end;

function TController.w\_t(w,alpha,V,H,teta: double):double;

begin

with \_Storage.param do

Result:=a[1,1]\*w+a[1,2]\*alpha+a[1,3]\*V+a[1,4]\*H+a[1,5]\*teta;

end;

function TController.alpha\_t(w,alpha,V,H,teta: double):double;

begin

with \_Storage.param do

Result:=a[2,1]\*w+a[2,2]\*alpha+a[2,3]\*V+a[2,4]\*H+a[2,5]\*teta;

end;

function TController.V\_t(w,alpha,V,H,teta: double):double;

begin

with \_Storage.param do

Result:=a[3,1]\*w+a[3,2]\*alpha+a[3,3]\*V+a[4,4]\*H+a[5,5]\*teta;

end;

function TController.H\_t(w,alpha,V,H,teta: double):double;

begin

with \_Storage.param do

Result:=a[4,1]\*w+a[4,2]\*alpha+a[4,3]\*V+a[4,4]\*H+a[4,5]\*teta;

end;

function TController.teta\_t(w,alpha,V,H,teta: double):double;

begin

with \_Storage.param do

Result:=a[5,1]\*w+a[5,2]\*alpha+a[5,3]\*V+a[5,4]\*H+a[5,5]\*teta;

end;

procedure TController.Solve;

var

size,i: integer;

begin

size:=round(\_Storage.param.tt/\_Storage.param.h)+1;

SetLength(w,size);

SetLength(alpha,size);

SetLength(V,size);

SetLength(H,size);

SetLength(teta,size);

w[0]:=\_Storage.param.w0;

alpha[0]:=\_Storage.param.alpha0;

V[0]:=\_Storage.param.V0;

H[0]:=\_Storage.param.H0;

teta[0]:=\_Storage.param.teta0;

for i := 1 to size do begin

w[i]:=w[i-1]+\_Storage.param.h\*w\_t(alpha[i-1],w[i-1],V[i-1],H[i-1],teta[i-1]);

alpha[i]:=alpha[i-1]+\_Storage.param.h\*alpha\_t(alpha[i-1],w[i-1],V[i-1],H[i-1],teta[i-1]);

V[i]:=V[i-1]+\_Storage.param.h\*V\_t(alpha[i-1],w[i-1],V[i-1],H[i-1],teta[i-1]);

H[i]:=H[i-1]+\_Storage.param.h\*H\_t(alpha[i-1],w[i-1],V[i-1],H[i-1],teta[i-1]);

teta[i]:=teta[i-1]+\_Storage.param.h\*teta\_t(alpha[i-1],w[i-1],V[i-1],H[i-1],teta[i-1]);

end;

//end;

end;

//

procedure TController.SetParametresOfFly(p: TParametresOfFly);

begin

\_Storage.SetParametresOfFly(p);

end;

function TController.GetParametresOfFly: TParametresOfFly;

begin

Result:=\_Storage.GetParametresOfFly;

end;

procedure TController.LoadDefaultParametres;

begin

\_Storage.SetDefaultParametres;

\_Border.UpdateParam;

end;

procedure TController.LoadParamFromFile;

begin

\_Storage.LoadParamFromFile;

end;

//

// Методы класса хранилища TStorage

//

constructor TStorage.Create;

begin

SetDefaultParametres;

end;

procedure TStorage.Connect(\_Controller\_: TController);

begin

\_Controller:=\_Controller\_;

end;

procedure TStorage.SetDefaultParametres;

begin

with param do begin

c[1]:=0.5;

c[2]:=15;

c[3]:=2;

c[4]:=1.21;

c[5]:=0.204;

c[6]:=1.4;

c[7]:=0.17;

c[8]:=0.1;

c[9]:=0.15;

c[10]:=0;

c[11]:=0;

c[12]:=0.985;

c[13]:=0.4;

c[14]:=0.31;

e[1]:=0.023;

e[2]:=0.1;

e[3]:=0.05;

w0:=0.5;

alpha0:=0.2;

V0:=0.3;

H0:=0.4;

teta0:=0.11;

h:=0.05;

tt:=5;

end;

UpdateA;

end;

procedure TStorage.SetParametresOfFly(p: TParametresOfFly);

begin

param:=p;

UpdateA;

end;

function TStorage.GetParametresOfFly: TParametresOfFly;

begin

Result:=param;

end;

procedure TStorage.UpdateA;

begin

with param do begin

a[1,1]:=-(c[1]+c[5]);

a[1,2]:=-c[2]+c[4]\*c[5];

a[1,3]:=c[5]\*e[2]-e[3];

a[1,4]:=-c[3]+c[5]\*c[9];

a[1,5]:=c[10]\*c[5];

a[2,1]:=1;

a[2,2]:=-c[4];

a[2,3]:=-e[2];

a[2,4]:=-c[9];

a[2,5]:=-c[10];

a[3,1]:=0;

a[3,2]:=-c[8];

a[3,3]:=-e[1];

a[3,4]:=0;

a[3,5]:=-c[7];

a[4,1]:=0;

a[4,2]:=-c[6];

a[4,3]:=c[11];

a[4,4]:=0;

a[4,5]:=c[6];

a[5,1]:=1;

a[5,2]:=0;

a[5,3]:=0;

a[5,4]:=0;

a[5,5]:=0;

end;

end;

//

procedure TStorage.SaveParamToFile;

var

f: file of TParametresOfFly;

const

f\_name='config.txt';

begin

AssignFile(f,f\_name);

ReWrite(f);

Write(f,param);

CloseFile(f);

end;

//

procedure TStorage.LoadParamFromFile;

var

f: file of TParametresOfFly;

const

f\_name='config.txt';

begin

AssignFile(f,f\_name);

Reset(f);

Read(f,param);

CloseFile(f);

end;

end.