**Исходный код**

(файл Unit1.pas)

*unit Unit1;*

*// filepos(f1) -> i*

*// startup graph and out*

*{$mode objfpc}{$H+}*

*interface*

*uses*

*Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, ExtCtrls,*

*ExtDlgs, Menus, ActnList, Grids;*

*type*

*{ TForm1 }*

*TForm1 = class(TForm)*

*Button1: TButton;*

*Button2: TButton;*

*Button3: TButton;*

*Button4: TButton;*

*Button5: TButton;*

*Button6: TButton;*

*Button7: TButton;*

*Button8: TButton;*

*Button9: TButton;*

*ComboBox1: TComboBox;*

*Edit1: TEdit;*

*Edit2: TEdit;*

*Edit3: TEdit;*

*Label1: TLabel;*

*Label10: TLabel;*

*Label11: TLabel;*

*Label2: TLabel;*

*Label3: TLabel;*

*Label4: TLabel;*

*Label5: TLabel;*

*Label6: TLabel;*

*Label7: TLabel;*

*Label8: TLabel;*

*Label9: TLabel;*

*Memo1: TMemo;*

*PaintBox1: TPaintBox;*

*Panel1: TPanel;*

*Panel2: TPanel;*

*StringGrid1: TStringGrid;*

*procedure Button1Click(Sender: TObject);*

*procedure Button2Click(Sender: TObject);*

*procedure Button3Click(Sender: TObject);*

*procedure Button4Click(Sender: TObject);*

*procedure Button5Click(Sender: TObject);*

*procedure Button6Click(Sender: TObject);*

*procedure Button7Click(Sender: TObject);*

*procedure Button8Click(Sender: TObject);*

*procedure Button9Click(Sender: TObject);*

*procedure Edit2Change(Sender: TObject);*

*procedure Edit3Change(Sender: TObject);*

*procedure FormCreate(Sender: TObject);*

*private*

*public*

*end;*

*var*

*Form1: TForm1;*

*implementation*

*type*

*comp = record*

*cost, proc, ram, hdd: integer;*

*exist: boolean;*

*end;*

*f = file of comp;*

*var*

*f1: f;*

*buf: comp;*

*{$R \*.lfm}*

*{ TForm1 }*

*procedure TForm1.FormCreate(Sender: TObject);*

*begin*

*Form1.Memo1.Text := 'Waiting';*

*PaintBox1.Canvas.Brush.Color := clWhite;*

*PaintBox1.Canvas.Clear;*

*Form1.Button7Click(Form1);*

*//Form1.Button4Click(Button4);*

*end;*

*//maxCost*

*procedure TForm1.Button1Click(Sender: TObject);*

*var*

*s, s1, s2, s3, s4: string;*

*maxCost, code, cnt: integer;*

*begin*

*assignFile(f1, 'comps.dat');*

*reset(f1);*

*//Form1.Memo1.Text := 'cost | ram | disk | proc' + #13 + #10;*

*Form1.StringGrid1.Clean;*

*val(Form1.Edit1.Text, maxCost, code);*

*cnt := -1;*

*if (code = 0) then*

*begin*

*while not EOF(f1) do*

*begin*

*Read(f1, buf);*

*if (buf.exist) and (buf.cost < maxCost) then*

*begin*

*cnt := cnt + 1;*

*str(buf.cost, s1);*

*str(buf.ram, s2);*

*str(buf.hdd, s3);*

*case buf.proc of*

*0: s4 := 'x32';*

*1: s4 := 'x64';*

*2: s4 := 'other'*

*else*

*s4 := 'unknown'*

*end;*

*// Form1.Memo1.Text := Form1.Memo1.Text + format('%6s |%6s |%6s |%9s',*

*// [s1, s2, s3, s4]) + #13 + #10;*

*if Form1.StringGrid1.RowCount <= cnt then*

*Form1.StringGrid1.InsertColRow(False, cnt);*

*Form1.StringGrid1.Cells[0, cnt] := s1;*

*Form1.StringGrid1.Cells[1, cnt] := s2;*

*Form1.StringGrid1.Cells[2, cnt] := s3;*

*Form1.StringGrid1.Cells[3, cnt] := s4;*

*end;*

*end;*

*end*

*else*

*begin*

*str(code, s1);*

*//Form1.StringGrid1.InsertColRow(False, 1);*

*Form1.StringGrid1.Cells[0, 1] := 'error' + s1;*

*//Form1.StringGrid1.Cells[1, 1] := s1;*

*end;*

*//Form1.Memo1.Text := '(incorrect parametres)';*

*CloseFile(f1);*

*end;*

*//minRam*

*procedure TForm1.Button2Click(Sender: TObject);*

*var*

*s, s1, s2, s3, s4: string;*

*minRam, code, cnt: integer;*

*begin*

*cnt := -1;*

*assignFile(f1, 'comps.dat');*

*reset(f1);*

*//Form1.Memo1.Text := 'cost | ram | disk | proc' + #13 + #10;*

*Form1.StringGrid1.Clean;*

*val(Form1.Edit2.Text, minRam, code);*

*if (code = 0) then*

*begin*

*while not EOF(f1) do*

*begin*

*Read(f1, buf);*

*if (buf.exist) and (buf.ram > minRam) then*

*begin*

*cnt := cnt + 1;*

*str(buf.cost, s1);*

*str(buf.ram, s2);*

*str(buf.hdd, s3);*

*case buf.proc of*

*0: s4 := 'x32';*

*1: s4 := 'x64';*

*2: s4 := 'other'*

*else*

*s4 := 'unknown'*

*end;*

*// Form1.Memo1.Text := Form1.Memo1.Text + format('%6s |%6s |%6s |%9s',*

*// [s1, s2, s3, s4]) + #13 + #10;*

*if Form1.StringGrid1.RowCount <= cnt then*

*Form1.StringGrid1.InsertColRow(False, cnt);*

*Form1.StringGrid1.Cells[0, cnt] := s1;*

*Form1.StringGrid1.Cells[1, cnt] := s2;*

*Form1.StringGrid1.Cells[2, cnt] := s3;*

*Form1.StringGrid1.Cells[3, cnt] := s4;*

*end;*

*end;*

*end*

*else*

*begin*

*str(code, s1);*

*Form1.StringGrid1.Cells[0, 1] := 'error' + s1;*

*//Form1.StringGrid1.Cells[1, 1] := s1;*

*end;*

*//Form1.Memo1.Text := '(incorrect parametres)';*

*CloseFile(f1);*

*end;*

*//allParams*

*procedure TForm1.Button3Click(Sender: TObject);*

*var*

*s, s1, s2, s3, s4: string;*

*minRam, minDisk, proctype, code, k, cnt: integer;*

*begin*

*cnt := -1;*

*assignFile(f1, 'comps.dat');*

*reset(f1);*

*//Form1.Memo1.Text := 'cost | ram | disk | proc' + #13 + #10;*

*Form1.StringGrid1.Clean;*

*k := 0;*

*val(Form1.Edit1.Text, minDisk, code);*

*k := k + code;*

*val(Form1.Edit2.Text, minRam, code);*

*k := k + code;*

*proctype := form1.ComboBox1.ItemIndex;*

*if (k = 0) then*

*begin*

*while not EOF(f1) do*

*begin*

*Read(f1, buf);*

*if (buf.exist) and (buf.ram > minRam) and (buf.hdd > minDisk) and*

*(buf.proc = proctype) then*

*begin*

*cnt := cnt + 1;*

*str(buf.cost, s1);*

*str(buf.ram, s2);*

*str(buf.hdd, s3);*

*case buf.proc of*

*0: s4 := 'x32';*

*1: s4 := 'x64';*

*2: s4 := 'other'*

*else*

*s4 := 'unknown'*

*end;*

*// Form1.Memo1.Text := Form1.Memo1.Text + format('%6s |%6s |%6s |%9s',*

*// [s1, s2, s3, s4]) + #13 + #10;*

*if Form1.StringGrid1.RowCount <= cnt then*

*Form1.StringGrid1.InsertColRow(False, cnt);*

*Form1.StringGrid1.Cells[0, cnt] := s1;*

*Form1.StringGrid1.Cells[1, cnt] := s2;*

*Form1.StringGrid1.Cells[2, cnt] := s3;*

*Form1.StringGrid1.Cells[3, cnt] := s4;*

*end;*

*end;*

*end*

*else*

*begin*

*str(code, s1);*

*Form1.StringGrid1.Cells[0, 1] := 'error' + s1;*

*//Form1.StringGrid1.Cells[1, 1] := s1;*

*end;*

*//Form1.Memo1.Text := '(incorrect parametres)';*

*CloseFile(f1);*

*end;*

*//TROUBLE SHOOTING*

*//Mem Table refresh*

*procedure TForm1.Button4Click(Sender: TObject);*

*var*

*maxCost, maxRam, minRam, id1, id2: integer;*

*k1, k2: real;*

*bufOld: comp;*

*f2: f;*

*endSort: boolean;*

*s1, s2, sdebug: string;*

*begin*

*// +refresh*

*AssignFile(f2, 'buffer.dat');*

*Rewrite(f2);*

*AssignFile(f1, 'comps.dat');*

*reset(f1);*

*while not EOF(f1) do*

*begin*

*Read(f1, buf);*

*if buf.exist then*

*Write(f2, buf);*

*end;*

*CloseFile(f1);*

*CloseFile(f2);*

*rewrite(f1);*

*reset(f2);*

*while not EOF(f2) do*

*begin*

*Read(f2, buf);*

*Write(f1, buf);*

*end;*

*closeFile(f1);*

*closeFile(f2);*

*// -refresh*

*// + sort*

*reset(f1);*

*endSort := False;*

*while not endSort do*

*begin*

*endSort := True;*

*reset(f1);*

*Read(f1, bufOld);*

*while not EOF(f1) do*

*begin*

*Read(f1, buf);*

*if bufOld.ram > buf.ram then*

*begin*

*endSort := False;*

*seek(f1, filepos(f1) - 2);*

*Write(f1, buf);*

*Write(f1, bufold);*

*id1 := filepos(f1) - 1;*

*CloseFile(f1);*

*Reset(f1);*

*Seek(f1, id1);*

*end;*

*bufOld := buf;*

*end;*

*CloseFile(f1);*

*end;*

*// -sort*

*reset(f1);*

*maxCost := 0;*

*maxRam := 0;*

*PaintBox1.Canvas.Clear;*

*while not EOF(f1) do*

*begin*

*Read(f1, buf);*

*if buf.exist then*

*begin*

*if buf.cost > maxCost then*

*maxCost := buf.cost;*

*if buf.Ram > maxRam then*

*maxRam := buf.ram;*

*end;*

*end;*

*k1 := PaintBox1.Width;*

*k2 := PaintBox1.Height;*

*reset(f1);*

*Read(f1, bufOld);*

*Read(f1, buf);*

*PaintBox1.Canvas.Pen.Width := 2;*

*while not EOF(f1) do*

*begin*

*//PaintBox1.Canvas.Brush.Color := clRed;*

*PaintBox1.Canvas.Pen.Color := clRed;*

*PaintBox1.Canvas.Line(*

*trunc(k1 \* (bufOld.ram / maxRam)),*

*PaintBox1.Height - trunc(k2 \* (bufOld.cost / maxCost)),*

*trunc(k1 \* (buf.ram / maxRam)),*

*PaintBox1.Height - trunc(k2 \* (buf.cost / maxCost)));*

*// +axisx*

*PaintBox1.Canvas.Pen.Color := clWhite;*

*str(bufOld.ram, s1);*

*PaintBox1.Canvas.TextOut(trunc(k1 \* (bufOld.ram / maxRam)), 330, s1);*

*PaintBox1.Canvas.Pen.Color := clBlack;*

*PaintBox1.Canvas.line(trunc(k1 \* (bufOld.ram / maxRam)), 0,*

*trunc(k1 \* (bufOld.ram / maxRam)), 327);*

*str(buf.ram, s2);*

*PaintBox1.Canvas.Pen.Color := clDefault;*

*if Buf.ram <> maxRam then*

*PaintBox1.Canvas.TextOut(trunc(k1 \* (buf.ram / maxRam)), 330, s2)*

*else*

*PaintBox1.Canvas.TextOut(trunc(k1 \* (buf.ram / maxRam)) - 10, 330, s2);*

*PaintBox1.Canvas.Pen.Color := clBlack;*

*if Buf.ram <> maxRam then*

*PaintBox1.Canvas.line(trunc(k1 \* (buf.ram / maxRam)),*

*0, trunc(k1 \* (buf.ram / maxRam)), 327)*

*else*

*PaintBox1.Canvas.line(trunc(k1 \* (buf.ram / maxRam)) - 10,*

*0, trunc(k1 \* (buf.ram / maxRam)) - 10, 327);*

*PaintBox1.Canvas.Pen.Color := clDefault;*

*// -axisx*

*// +axisy*

*str(bufOld.cost, s1);*

*PaintBox1.Canvas.TextOut(1, PaintBox1.Height -*

*trunc(k2 \* (bufOld.cost / maxCost)), s1);*

*str(buf.cost, s2);*

*PaintBox1.Canvas.Pen.Color := clBlack;*

*PaintBox1.Canvas.line(10, PaintBox1.Height - trunc(k2 \* (bufOld.cost / maxCost)),*

*350, PaintBox1.Height - trunc(k2 \* (bufOld.cost / maxCost)));*

*PaintBox1.Canvas.Pen.Color := clDefault;*

*if Buf.cost <> maxCost then*

*PaintBox1.Canvas.TextOut(1, PaintBox1.Height -*

*trunc(k2 \* (buf.cost / maxCost)), s2)*

*else*

*PaintBox1.Canvas.TextOut(1, PaintBox1.Height -*

*trunc(k2 \* (buf.cost / maxCost)) + 10, s2);*

*PaintBox1.Canvas.Pen.Color := clBlack;*

*if Buf.cost <> maxCost then*

*PaintBox1.Canvas.line(10, PaintBox1.Height - trunc(k2 \* (buf.cost / maxCost)),*

*350, PaintBox1.Height - trunc(k2 \* (buf.cost / maxCost)))*

*else*

*PaintBox1.Canvas.line(10, PaintBox1.Height - trunc(k2 \* (buf.cost / maxCost)) +*

*10, 350, PaintBox1.Height - trunc(k2 \* (buf.cost / maxCost)) + 10);*

*// -axisy*

*bufOld := buf;*

*Read(f1, buf);*

*end;*

*CloseFile(f1);*

*end;*

*procedure TForm1.Button5Click(Sender: TObject);*

*var*

*code: integer;*

*s: string;*

*begin*

*assignFile(f1, 'comps.dat');*

*reset(f1);*

*Val(Form1.Edit1.Text, buf.cost, code);*

*Val(Form1.Edit2.Text, buf.ram, code);*

*Val(Form1.Edit3.Text, buf.hdd, code);*

*// add protection by code*

*buf.exist := True;*

*buf.proc := Form1.ComboBox1.ItemIndex;*

*Seek(f1, FileSize(f1));*

*Write(f1, buf);*

*CloseFile(f1);*

*end;*

*procedure TForm1.Button6Click(Sender: TObject);*

*var*

*// rename vars*

*s: string;*

*cnt, minRam, minDisk, procType, code: integer;*

*begin*

*assignFile(f1, 'comps.dat');*

*reset(f1);*

*Form1.Memo1.Text := 'Deleted';*

*while not EOF(f1) do*

*begin*

*Read(f1, buf);*

*val(Form1.Edit3.Text, minDisk, code);*

*val(Form1.Edit2.Text, minRam, code);*

*// add protection*

*if (buf.exist) and (buf.ram = minRam) and (buf.hdd = minDisk) and*

*(buf.proc = Form1.ComboBox1.ItemIndex) then*

*begin*

*buf.cost := 0;*

*buf.exist := False;*

*buf.hdd := 0;*

*buf.proc := 0;*

*buf.ram := 0;*

*Seek(f1, filepos(f1) - 1);*

*Write(f1, buf);*

*end;*

*end;*

*CloseFile(f1);*

*end;*

*// main out*

*procedure TForm1.Button7Click(Sender: TObject);*

*var*

*s1, s2, s3, s4, ss: string;*

*i: byte;*

*cnt: integer;*

*begin*

*cnt := -1;*

*Form1.StringGrid1.Clean;*

*assignFile(f1, 'comps.dat');*

*reset(f1);*

*// Form1.Memo1.Text := format('%6s |%6s |%6s |%9s', ['Cost', 'RAM',*

*// 'Disk', 'Processor']) + #13 + #10;*

*while not EOF(f1) do*

*begin*

*Read(f1, buf);*

*if buf.exist then*

*begin*

*cnt := cnt + 1;*

*str(buf.cost, s1);*

*str(buf.ram, s2);*

*str(buf.hdd, s3);*

*case buf.proc of*

*0: s4 := 'x32';*

*1: s4 := 'x64';*

*2: s4 := 'other'*

*else*

*s4 := 'unknown'*

*end;*

*// Form1.Memo1.Text := Form1.Memo1.Text + format('%6s |%6s |%6s |%9s',*

*// [s1, s2, s3, s4]) + #13 + #10;*

*if Form1.StringGrid1.RowCount <= cnt then*

*Form1.StringGrid1.InsertColRow(False, cnt);*

*Form1.StringGrid1.Cells[0, cnt] := s1;*

*Form1.StringGrid1.Cells[1, cnt] := s2;*

*Form1.StringGrid1.Cells[2, cnt] := s3;*

*Form1.StringGrid1.Cells[3, cnt] := s4;*

*end;*

*end;*

*CloseFile(f1);*

*end;*

*procedure TForm1.Button8Click(Sender: TObject);*

*var*

*f2: f;*

*begin*

*Form1.Memo1.Text := 'Mem table refreshed';*

*AssignFile(f2, 'buffer.dat');*

*Rewrite(f2);*

*AssignFile(f1, 'comps.dat');*

*reset(f1);*

*while not EOF(f1) do*

*begin*

*Read(f1, buf);*

*if buf.exist then*

*Write(f2, buf);*

*end;*

*CloseFile(f1);*

*CloseFile(f2);*

*rewrite(f1);*

*reset(f2);*

*while not EOF(f2) do*

*begin*

*Read(f2, buf);*

*Write(f1, buf);*

*end;*

*closeFile(f1);*

*closeFile(f2);*

*end;*

*procedure TForm1.Button9Click(Sender: TObject);*

*begin*

*AssignFile(f1, 'comps.dat');*

*rewrite(f1);*

*closeFile(f1);*

*end;*

*procedure TForm1.Edit2Change(Sender: TObject);*

*begin*

*// for id*

*end;*

*procedure TForm1.Edit3Change(Sender: TObject);*

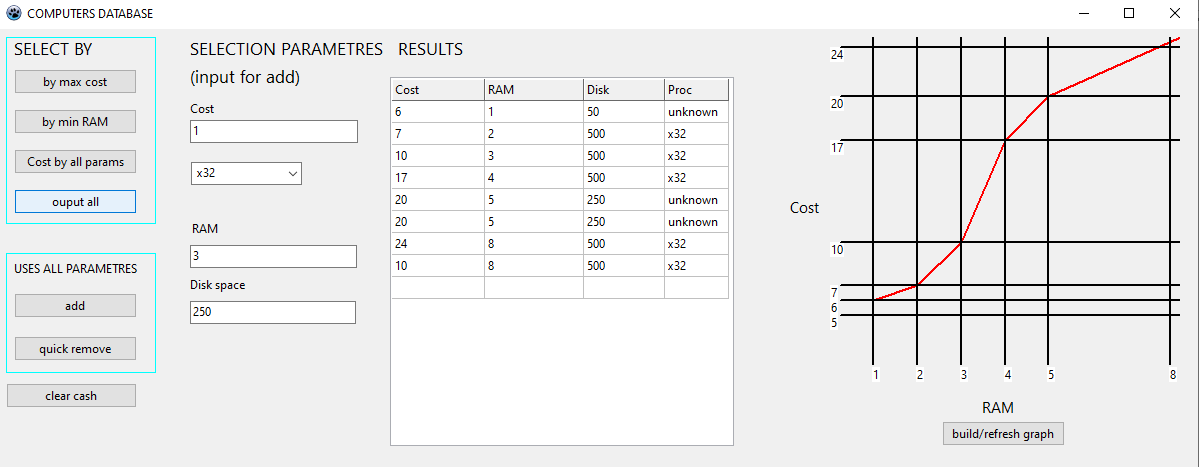
*begin*

*// for id*

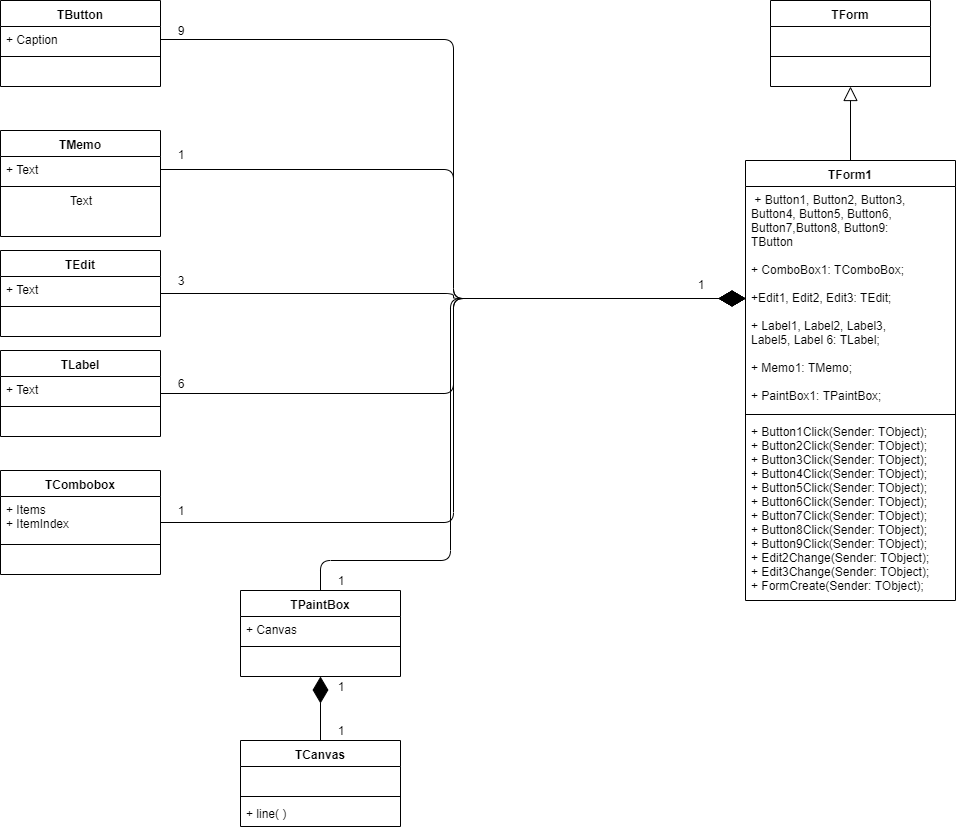
*end;*

*end.*

**Скриншоты**

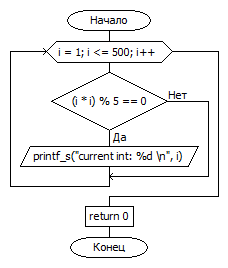
****

**Диаграмма классов**

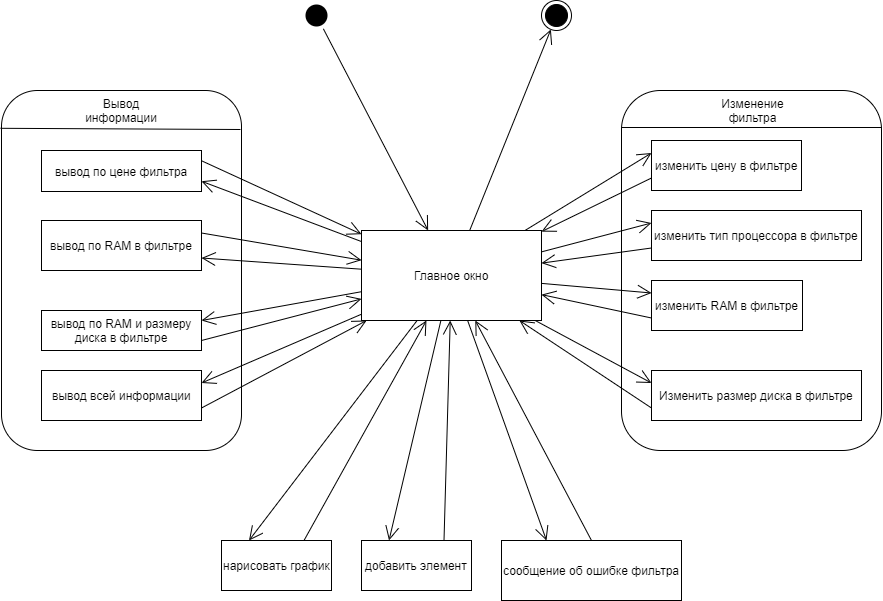
****

**Схема алгоритма**

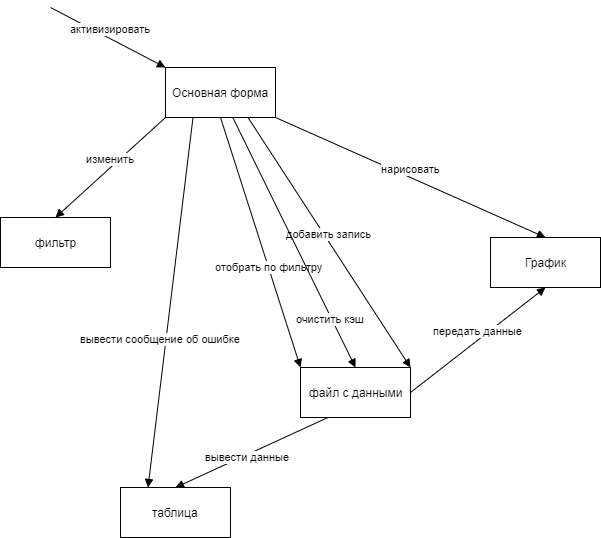
(Source.cpp)



**Диаграмма состояний интерфейса**

****

**Объектная декомпозиция**

****

**Вывод**

* Delphi предоставляет широкий набор средств для событийного программирования и создания графических интерфейсов
* Delphi предоставляет широкий набор средств для вывода информации в различных формах, таких как таблицы и графики