**СОДЕРЖАНИЕ**

[1. ОБЩИЙ ФАЙЛ ПРОЕКТА RD.DPR. 2](#_Toc216356293)

[2. МОДУЛЬ AboutUnit.pas 2](#_Toc216356294)

[3. МОДУЛЬ AdminUnit.pas 3](#_Toc216356295)

[4. МОДУЛЬ BeginUnit.pas 16](#_Toc216356296)

[5. МОДУЛЬ ConstUnit.pas 28](#_Toc216356297)

[6. МОДУЛЬ DataUnit.pas 37](#_Toc216356298)

[7. МОДУЛЬ DeviceThread.pas 38](#_Toc216356299)

[8. МОДУЛЬ DeviceRD.pas 48](#_Toc216356300)

[9. МОДУЛЬ IzmGraphUnit.pas 57](#_Toc216356301)

[10. МОДУЛЬ IzmUnit.pas 58](#_Toc216356302)

[11. МОДУЛЬ MainUnit.pas 65](#_Toc216356303)

[12. МОДУЛЬ MultiInstUnit.pas 108](#_Toc216356304)

# ОБЩИЙ ФАЙЛ ПРОЕКТА RD.DPR.

program rd;

uses

Forms,

MainUnit in 'MainUnit.pas' {MainForm},

DataUnit in 'DataUnit.pas' {SQL\_DM: TDataModule},

BeginUnit in 'BeginUnit.pas' {BeginForm},

ConstUnit in 'ConstUnit.pas',

D2XXUnit in 'D2XXUnit.pas',

DeviceRD in 'DeviceRD.pas',

device\_thread in 'device\_thread.pas',

IzmUnit in 'IzmUnit.pas' {IzmForm},

AdminUnit in 'AdminUnit.pas' {AdminForm},

AboutUnit in 'AboutUnit.pas' {AboutBox},

IzmGraphUnit in 'IzmGraphUnit.pas' {IzmGraphFrame: TFrame},

MultiInstUnit in 'MultiInstUnit.pas';

const

UniqueString = 'RDProject';

{Может быть любое слово. Желательно латинскими буквами.}

{$E exe}

{$R \*.res}

begin

if init\_mutex(UniqueString) then

begin

Application.Initialize;

//Application.MainFormOnTaskbar := True;

Application.Title := 'RD Project';

Application.CreateForm(TMainForm, MainForm);

Application.CreateForm(TSQL\_DM, SQL\_DM);

Application.CreateForm(TBeginForm, BeginForm);

Application.CreateForm(TIzmForm, IzmForm);

Application.CreateForm(TAdminForm, AdminForm);

Application.CreateForm(TAboutBox, AboutBox);

Application.Run;

//Application.BringToFront;

end

else begin

Show\_mes;

exit; {Выходим до инициализации, если мьютекс уже есть}

end;

end.

# МОДУЛЬ AboutUnit.pas

unit AboutUnit;

interface

uses Windows, SysUtils, Classes, Graphics, Forms, Controls, StdCtrls,

Buttons, ExtCtrls, AdvGlowButton, AdvPanel, jpeg;

type

TAboutBox = class(TForm)

OKButton: TAdvGlowButton;

AdvPanel1: TAdvPanel;

Comments: TLabel;

Copyright: TLabel;

ProductName: TLabel;

ProgramIcon: TImage;

Version: TLabel;

procedure OKButtonClick(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

AboutBox: TAboutBox;

implementation

{$R \*.dfm}

procedure TAboutBox.OKButtonClick(Sender: TObject);

begin

self.Close;

end;

end.

# МОДУЛЬ AdminUnit.pas

unit AdminUnit;

interface

uses

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

Dialogs, Grids, BaseGrid, DBAdvGrid, StdCtrls, AdvGlowButton,

AdvEdit, Spin, AdvPicture, ExtCtrls, AdvPageControl, ComCtrls, DB, StdActns,

ExtActns, ActnList, ExtDlgs, AdvPanel, AdvGrid, Jpeg;

type

TAdminForm = class(TForm)

Admin\_PC: TAdvPageControl;

PointsPanel: TAdvTabSheet;

Panel1: TPanel;

Point\_param\_label: TLabel;

point\_param\_name\_label: TLabel;

Point\_param\_opis\_label: TLabel;

pict: TAdvPicture;

point\_param\_name\_znach: TLabel;

point\_param\_num\_znach: TLabel;

point\_param\_opis\_znach: TLabel;

Panel2: TPanel;

point\_param\_\_ins\_num\_label: TLabel;

point\_image\_label: TLabel;

point\_image\_size\_label: TLabel;

point\_image\_size\_znach\_label: TLabel;

point\_image: TAdvPicture;

point\_image\_size\_znach: TLabel;

Point\_param\_SE: TSpinEdit;

point\_param\_name\_edit: TAdvEdit;

point\_param\_ops\_edit: TAdvEdit;

point\_image\_label\_load: TAdvGlowButton;

point\_param\_reset\_btn: TAdvGlowButton;

point\_param\_save\_btn: TAdvGlowButton;

AdvTabSheet8: TAdvTabSheet;

AdvTabSheet9: TAdvTabSheet;

AdvTabSheet10: TAdvTabSheet;

Button1: TButton;

Memo1: TMemo;

Button2: TButton;

AdvTabSheet11: TAdvTabSheet;

Admin\_pp\_DBGrid: TDBAdvGrid;

ActionList: TActionList;

Show\_points\_table: TAction;

Show\_params\_table: TAction;

Add\_point: TAction;

Add\_param: TAction;

Del\_point: TAction;

ImageFileOpen: TOpenPicture;

Del\_param: TAction;

Red\_point: TAction;

Red\_param: TAction;

Initial\_load: TAction;

Admin\_add\_point\_param: TAdvGlowButton;

Admin\_del\_point\_param: TAdvGlowButton;

Admin\_red\_point\_param: TAdvGlowButton;

Admin\_navigate\_panel: TAdvPanelGroup;

Panel3: TPanel;

Panel4: TPanel;

Admin\_show\_points\_panel: TAdvPanel;

Admin\_show\_params\_panel: TAdvPanel;

Admin\_show\_measure\_panel: TAdvPanel;

Admin\_show\_points\_label: TLabel;

Admin\_show\_params\_label: TLabel;

Admin\_show\_measure\_label: TLabel;

default\_points\_params: TAdvGlowButton;

Make\_default: TAction;

procedure Show\_points\_tableExecute(Sender: TObject);

procedure Add\_pointExecute(Sender: TObject);

procedure point\_image\_label\_loadClick(Sender: TObject);

procedure point\_param\_save\_btnClick(Sender: TObject);

procedure point\_param\_reset\_btnClick(Sender: TObject);

procedure Del\_pointExecute(Sender: TObject);

procedure Admin\_pp\_DBGridSelectCell(Sender: TObject; ACol, ARow: Integer;

var CanSelect: Boolean);

procedure Show\_params\_tableExecute(Sender: TObject);

procedure Add\_paramExecute(Sender: TObject);

procedure Del\_paramExecute(Sender: TObject);

procedure Red\_pointExecute(Sender: TObject);

procedure Red\_paramExecute(Sender: TObject);

procedure Initial\_loadExecute(Sender: TObject);

procedure FormShow(Sender: TObject);

procedure Admin\_pp\_DBGridDblClickCell(Sender: TObject; ARow, ACol: Integer);

procedure FormKeyPress(Sender: TObject; var Key: Char);

procedure Admin\_pp\_DBGridKeyPress(Sender: TObject; var Key: Char);

procedure Admin\_show\_points\_labelClick(Sender: TObject);

procedure Admin\_show\_params\_labelClick(Sender: TObject);

procedure ImageFileOpenAccept(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure GetResourceAsJpeg(const resname: string; AInstance: Cardinal; aImage: TAdvPicture);

procedure Make\_defaultExecute(Sender: TObject);

private

{ Private declarations }

public

admin\_db\_cur\_table:string; //точки или параметры{ Public declarations }

db\_grid\_cur\_row\_num:integer; //текущий ряд таблички

act:string; //действие для кнопки сохранить

//варианты - сохранение и обновление

backrow:integer; // номер ряда на который пойдет возвращение

end;

var

AdminForm: TAdminForm;

implementation

uses MainUnit, DataUnit, ConstUnit;

{$R \*.dfm}

////////////////////////////////

/// ОБЩИЕ ПРОЦЕДУРЫ ФОРМЫ ///

////////////////////////////////

procedure TAdminForm.FormClose(Sender: TObject; var Action: TCloseAction);

begin

MainForm.Show\_all\_pat.Execute;

end;

procedure TAdminForm.FormKeyPress(Sender: TObject; var Key: Char);

begin

// при нажатии Esc

if key=#27 then

begin

// в случае если не происходит добавления или редактирования точки/параметрыа

// закрываем форму

if act='' then

begin

AdminForm.CloseModal;

AdminForm.Close;

end

//если происходит то возращаемся к исходному состоянию

else if admin\_db\_cur\_table='points' then Show\_points\_table.Execute

else if admin\_db\_cur\_table='params' then Show\_params\_table.Execute;

//нажатие f5 на таблице - обновление

if (GetKeyState(vk\_f5) AND 128)=128 then

if admin\_db\_cur\_table='points' then Show\_points\_table.Execute

else if admin\_db\_cur\_table='params' then Show\_params\_table.Execute;

end;

end;

procedure TAdminForm.FormShow(Sender: TObject);

begin

//пока не сделано: инициализация переменной для возвращения на запись в БД

backrow:=1;

Initial\_load.Execute;

end;

procedure TAdminForm.ImageFileOpenAccept(Sender: TObject);

begin

if ImageFileOpen.Dialog.FileName<>'' then

try

point\_image.Picture.LoadFromFile(ImageFileOpen.Dialog.FileName);

finally

point\_image\_size\_znach\_label.Caption:=image\_size\_label+GetFileSize(ImageFileOpen.Dialog.FileName);

end

else

begin

//что будет если файл не выбран

end;

//ImageFileOpen.Dialog.FileName:='';

end;

procedure TAdminForm.point\_image\_label\_loadClick(Sender: TObject);

begin

//картинка по идее нахлодится в корневой директории приложения, поэтому

// устанавливаем начальный путь в директории с exe файлом

ImageFileOpen.Dialog.InitialDir:=ExtractFilePath(Application.ExeName);

ImageFileOpen.Execute;

end;

procedure TAdminForm.Initial\_loadExecute(Sender: TObject);

begin

if admin\_db\_cur\_table='points' then

begin

AdminForm.Show\_points\_table.Execute;

end

else if admin\_db\_cur\_table='params' then

begin

AdminForm.Show\_params\_table.Execute;

end;

Admin\_pp\_DBGrid.SetFocus;

act:='';

end;

procedure TAdminForm.Make\_defaultExecute(Sender: TObject);

var

h: THandle;

stream:TMemoryStream;

points\_Array: array[1..24,1..2] of string;

params\_Array: array[1..10,1..2] of string;

I: Integer;

begin

if MessageBox(0,

PChar(make\_defaut\_message),

PChar(mess\_caption),

MB\_ICONQUESTION or MB\_OKCANCEL

)=mrOK then

begin

if AdminForm.admin\_db\_cur\_table='points' then

begin

//Записываем всю информацию о точках

points\_Array[1,1]:='PL1\_r';points\_Array[13,1]:='RP1\_r';

points\_Array[2,1]:='GL1\_r';points\_Array[14,1]:='F1\_r';

points\_Array[3,1]:='MC9\_r';points\_Array[15,1]:='E45\_r';

points\_Array[4,1]:='TR1\_r';points\_Array[16,1]:='VB44\_r';

points\_Array[5,1]:='C9\_r';points\_Array[17,1]:='R0\_r';

points\_Array[6,1]:='IG1\_r';points\_Array[18,1]:='V67\_r';

points\_Array[7,1]:='PL1\_l';points\_Array[19,1]:='RP1\_l';

points\_Array[8,1]:='GL1\_l';points\_Array[20,1]:='F1\_l';

points\_Array[9,1]:='MC9\_l';points\_Array[21,1]:='E45\_l';

points\_Array[10,1]:='TR1\_l';points\_Array[22,1]:='VB44\_l';

points\_Array[11,1]:='C9\_l';points\_Array[23,1]:='R0\_l';

points\_Array[12,1]:='IG1\_l';points\_Array[24,1]:='V67\_l';

points\_Array[1,2]:='PL1\_r';points\_Array[13,2]:='RP1\_r';

points\_Array[2,2]:='GL1\_r';points\_Array[14,2]:='F1\_r';

points\_Array[3,2]:='MC9\_r';points\_Array[15,2]:='E45\_r';

points\_Array[4,2]:='TR1\_r';points\_Array[16,2]:='VB44\_r';

points\_Array[5,2]:='C9\_r';points\_Array[17,2]:='R0\_r';

points\_Array[6,2]:='IG1\_r';points\_Array[18,2]:='V67\_r';

points\_Array[7,2]:='PL1\_l';points\_Array[19,2]:='RP1\_l';

points\_Array[8,2]:='GL1\_l';points\_Array[20,2]:='F1\_l';

points\_Array[9,2]:='MC9\_l';points\_Array[21,2]:='E45\_l';

points\_Array[10,2]:='TR1\_l';points\_Array[22,2]:='VB44\_l';

points\_Array[11,2]:='C9\_l';points\_Array[23,2]:='R0\_l';

points\_Array[12,2]:='IG1\_l';points\_Array[24,2]:='V67\_l';

//загружаем библиотеку

h := LoadLibrary('pictures.dll');

//загружаем все в базу

if h>0 then

begin

//удаляем все точки:

mysql\_query(SQL\_DM.mysql\_query1,'DELETE FROM `'+table\_points\_name+';',2);

mysql\_query(SQL\_DM.mysql\_query1,'DELETE FROM `'+table\_izm\_name+';',2);

for I := 1 to 24 do

begin

//запихиваем инфу в датасет, и соответственно в базу

//запрос на выбор всего что есть

mysql\_query(SQL\_DM.mysql\_query2, 'SELECT num, name, opisanie, image From '+table\_points\_name+';', 1);

try

with SQL\_DM.mysql\_DSQ2.DataSet do

begin

Append;

stream := TMemoryStream.Create;

GetResourceAsJpeg(points\_Array[i,1],h,pict);

pict.picture.SaveToStream(stream);

(FieldByName('image') as TBlobField).LoadFromStream(Stream);

FieldByName('num').AsString:=inttostr(i);

FieldByName('opisanie').AsString:=points\_Array[i,2];

FieldByName('name').AsString:=points\_Array[i,1];

Post;

end;

stream.Free;

finally

end;

end;

end

else Showmessage(make\_default\_dll\_error);

FreeLibrary(h);

Show\_points\_table.Execute;

end;

if AdminForm.admin\_db\_cur\_table='params' then

begin

params\_Array[1,1]:=params\_default\_name1;

params\_Array[1,2]:=params\_default\_name\_opisnie\_1;

params\_Array[2,1]:=params\_default\_name2;

params\_Array[2,2]:=params\_default\_name\_opisnie\_2;

params\_Array[3,1]:=params\_default\_name3;

params\_Array[3,2]:=params\_default\_name\_opisnie\_3;

params\_Array[4,1]:=params\_default\_name4;

params\_Array[4,2]:=params\_default\_name\_opisnie\_4;

params\_Array[5,1]:=params\_default\_name5;

params\_Array[5,2]:=params\_default\_name\_opisnie\_5;

params\_Array[6,1]:=params\_default\_name6;

params\_Array[6,2]:=params\_default\_name\_opisnie\_6;

params\_Array[7,1]:=params\_default\_name7;

params\_Array[7,2]:=params\_default\_name\_opisnie\_7;

params\_Array[8,1]:=params\_default\_name8;

params\_Array[8,2]:=params\_default\_name\_opisnie\_8;

params\_Array[9,1]:=params\_default\_name9;

params\_Array[9,2]:=params\_default\_name\_opisnie\_9;

params\_Array[10,1]:=params\_default\_name10;

params\_Array[10,2]:=params\_default\_name\_opisnie\_10;

//удаляем все точки:

mysql\_query(SQL\_DM.mysql\_query1,'DELETE FROM `'+table\_param\_name+';',2);

mysql\_query(SQL\_DM.mysql\_query1,'DELETE FROM `'+table\_izm\_name+';',2);

for I := 1 to 10 do

begin

//запихиваем инфу в датасет, и соответственно в базу

//запрос на выбор всего что есть

mysql\_query(SQL\_DM.mysql\_query2, 'SELECT id, name, opisanie From '+table\_param\_name+';', 1);

try

with SQL\_DM.mysql\_DSQ2.DataSet do begin

Append;

FieldByName('id').AsString:=inttostr(i);

FieldByName('opisanie').AsString:=params\_Array[i,2];

FieldByName('name').AsString:=params\_Array[i,1];

Post;

end;

finally

end;

end;

Show\_params\_table.Execute;

end;

end;

end;

procedure TAdminForm.Admin\_show\_params\_labelClick(Sender: TObject);

begin

AdminForm.admin\_db\_cur\_table:='params';

Show\_params\_table.Execute;

end;

procedure TAdminForm.Admin\_show\_points\_labelClick(Sender: TObject);

begin

AdminForm.admin\_db\_cur\_table:='points';

Show\_points\_table.Execute;

end;

procedure TAdminForm.point\_param\_save\_btnClick(Sender: TObject);

var stream:TMemoryStream;

begin

if (admin\_db\_cur\_table='points') and (act='add') then

begin

//запихиваем инфу в датасет, и соответственно в базу

//запрос на выбор всего что есть

mysql\_query(SQL\_DM.mysql\_query2, 'SELECT num, name, opisanie, image From '+table\_points\_name+';', 1);

try

with SQL\_DM.mysql\_DSQ2.DataSet do begin

Append;

stream := TMemoryStream.Create;

point\_image.picture.SaveToStream(stream);

(FieldByName('image') as TBlobField).LoadFromStream(Stream);

FieldByName('num').AsString:=inttostr(Point\_param\_SE.Value);

FieldByName('opisanie').AsString:=point\_param\_ops\_edit.Text;

FieldByName('name').AsString:=point\_param\_name\_edit.Text;

Post;

end;

stream.Free;

finally

MessageBox(0,

PChar(add\_point\_to\_database\_mes),

PChar(mess\_caption),

MB\_ICONINFORMATION or MB\_OK

);

Show\_points\_table.Execute;

end;

end

else if (admin\_db\_cur\_table='params') and (act='add') then

begin

//запихиваем инфу в датасет, и соответственно в базу

//запрос на выбор всего что есть

mysql\_query(SQL\_DM.mysql\_query2, 'SELECT id, name, opisanie From '+table\_param\_name+';', 1);

try

with SQL\_DM.mysql\_DSQ2.DataSet do begin

Append;

FieldByName('id').AsString:=inttostr(Point\_param\_SE.Value);

FieldByName('opisanie').AsString:=point\_param\_ops\_edit.Text;

FieldByName('name').AsString:=point\_param\_name\_edit.Text;

Post;

end;

finally

MessageBox(0,

PChar(add\_param\_to\_database\_mes),

PChar(mess\_caption),

MB\_ICONINFORMATION or MB\_OK

);

Show\_params\_table.Execute;

end;

end

else if (admin\_db\_cur\_table='points') and (act='update') then

begin

backrow:=Point\_param\_SE.Value;

//запихиваем инфу в датасет, и соответственно в базу

//запрос на выбор всего что есть

mysql\_query(SQL\_DM.mysql\_query2, 'SELECT num, name, opisanie, image From '+table\_points\_name+

' WHERE `num`='+inttostr(Point\_param\_SE.Value)+';', 1);

try

with SQL\_DM.mysql\_DSQ2.DataSet do begin

Edit;

stream := TMemoryStream.Create;

point\_image.picture.SaveToStream(stream);

(FieldByName('image') as TBlobField).LoadFromStream(Stream);

FieldByName('num').AsString:=inttostr(Point\_param\_SE.Value);

FieldByName('opisanie').AsString:=point\_param\_ops\_edit.Text;

FieldByName('name').AsString:=point\_param\_name\_edit.Text;

UpdateRecord;

Post;

end;

stream.Free;

finally

MessageBox(0,

PChar(red\_point\_in\_database\_mes),

PChar(mess\_caption),

MB\_ICONINFORMATION or MB\_OK

);

Show\_points\_table.Execute;

end;

end

else if (admin\_db\_cur\_table='params') and (act='update') then

begin

//запихиваем инфу в датасет, и соответственно в базу

//запрос на выбор всего что есть

mysql\_query(SQL\_DM.mysql\_query2, 'SELECT id, name, opisanie From '+table\_param\_name+

' WHERE `id`='+inttostr(Point\_param\_SE.Value)+';', 1);

try

with SQL\_DM.mysql\_DSQ2.DataSet do begin

Edit;

FieldByName('id').AsString:=inttostr(Point\_param\_SE.Value);

FieldByName('opisanie').AsString:=point\_param\_ops\_edit.Text;

FieldByName('name').AsString:=point\_param\_name\_edit.Text;

UpdateRecord;

Post;

end;

finally

MessageBox(0,

PChar(red\_param\_in\_database\_mes),

PChar(mess\_caption),

MB\_ICONINFORMATION or MB\_OK

);

Show\_params\_table.Execute;

end;

end;

end;

procedure TAdminForm.point\_param\_reset\_btnClick(Sender: TObject);

begin

Initial\_load.Execute;

end;

////////////////////////////////

/// ПРОЦЕДУРЫ ТАБЛИЦЫ ///

////////////////////////////////

procedure TAdminForm.Admin\_pp\_DBGridDblClickCell(Sender: TObject; ARow,

ACol: Integer);

begin

//запоминаем текущую строку

db\_grid\_cur\_row\_num:=Arow;

//тоже самое, хз пока почему :)

backrow:=ARow;

//переходим в режим редактирования

if admin\_db\_cur\_table='points' then

Red\_point.Execute

else if admin\_db\_cur\_table='params' then

Red\_param.Execute;

end;

procedure TAdminForm.Admin\_pp\_DBGridSelectCell(Sender: TObject; ACol, ARow: Integer;

var CanSelect: Boolean);

var stream:TMemoryStream;

begin

//если в поле номера есть что-то

if Admin\_pp\_DBGrid.Cells[1,Arow]<>'' then

//в зависимости от того активна какая-либо из точек делаем доступными

//оперделенными действия

begin

//записываем текущее значение поля num, по пути проверяем что не

//нажат та же строка

if db\_grid\_cur\_row\_num<>strtoint(Admin\_pp\_DBGrid.Cells[1,Arow]) then

begin

db\_grid\_cur\_row\_num:=strtoint(Admin\_pp\_DBGrid.Cells[1,Arow]);

//если открыта таблица точек

if admin\_db\_cur\_table='points' then

begin

//показываем текущие данные о точке

point\_param\_num\_znach.Caption:=(Admin\_pp\_DBGrid.Cells[1,Arow]);

point\_param\_name\_znach.Caption:=Admin\_pp\_DBGrid.Cells[2,Arow];

point\_param\_opis\_znach.Caption:=Admin\_pp\_DBGrid.Cells[3,Arow];

// показываем изображение

try

//загружаем изображение в отдельный запрос

mysql\_query(SQL\_DM.mysql\_query1,'SELECT `image` FROM '+table\_points\_name+' WHERE `num`='+

Admin\_pp\_DBGrid.Cells[1,Arow]+';',1);

Pict.Picture:=nil;

//создаем поток в памяти

stream := TMemoryStream.Create;

//созхраняем туда картинку

(SQL\_DM.mysql\_DSQ1.DataSet.fieldbyname('image') as TBlobField).SaveToStream(stream);

//чтобы что-нибудь записать, необходимо установить позицию потока в ноль!

Stream.Position := 0;

//загружаем картинку

pict.Picture.LoadFromStream(stream);

//уничтожаем поток

stream.Free;

except

end;

//показываем картинку

if pict.Picture<>nil then pict.Show;

end;

//если открыта таблица параметров

if admin\_db\_cur\_table='params' then

begin

//показываем текущие данные о параметре

point\_param\_num\_znach.Caption:=(Admin\_pp\_DBGrid.Cells[1,Arow]);

point\_param\_name\_znach.Caption:=Admin\_pp\_DBGrid.Cells[2,Arow];

point\_param\_opis\_znach.Caption:=Admin\_pp\_DBGrid.Cells[3,Arow];

end;

end;

end;

end;

procedure TAdminForm.Admin\_pp\_DBGridKeyPress(Sender: TObject; var Key: Char);

begin

// см. Процедуру FormKeyPress

if key=#27 then

begin

if act='' then

begin

AdminForm.CloseModal;

AdminForm.Close;

end

else if admin\_db\_cur\_table='points' then Show\_points\_table.Execute

else if admin\_db\_cur\_table='params' then Show\_params\_table.Execute;

end;

//нажатие f5 на таблице - обновление

if (GetKeyState(vk\_f5) AND 128)=128 then

if admin\_db\_cur\_table='points' then Show\_points\_table.Execute

else if admin\_db\_cur\_table='params' then Show\_params\_table.Execute;

end;

////////////////////////////////

/// ПРОЦЕДУРЫ ДЛЯ ТОЧЕК ///

////////////////////////////////

procedure TAdminForm.Show\_points\_tableExecute(Sender: TObject);

var sel:boolean;

begin

//ЭЛЕМЕНТЫ ИНТЕРФЕЙСА

//Панели

Panel2.Hide;

Panel1.Show;

//Панель с общей информацией

point\_param\_num\_znach.Caption:='';

point\_param\_name\_znach.Caption:='';

point\_param\_opis\_znach.Caption:='';

//Панель ввода

Point\_param\_SE.Value:=0;

point\_param\_name\_edit.Clear;

point\_param\_ops\_edit.Clear;

//все что относитмя к картинке

pict .Show;

point\_image\_label\_load .Show;

point\_image\_label .Show;

point\_image\_size\_label .Show;

point\_image\_size\_znach\_label .Show;

//имена меток

point\_param\_\_ins\_num\_label.Caption:=point\_label\_name;

Point\_param\_label.Caption:=point\_label\_name;

//действия

Admin\_add\_point\_param.Action:=AdminForm.Add\_point;

Admin\_red\_point\_param.Action:=AdminForm.Red\_point;

Admin\_del\_point\_param.Action:=AdminForm.Del\_point;

db\_grid\_cur\_row\_num:=-1;

//загрузка таблицы

mysql\_query(SQL\_DM.mysql\_query3, 'SELECT num, name, opisanie FROM '+table\_points\_name+';', 1);

with Admin\_pp\_DBGrid do

begin

DataSource:=SQL\_DM.mysql\_DSQ3;

MainForm.Col\_resize(Admin\_pp\_DBGrid, 'num', point\_param\_num\_name, 45);

MainForm.Col\_resize(Admin\_pp\_DBGrid, 'name', point\_param\_name\_name, 100);

MainForm.Col\_resize(Admin\_pp\_DBGrid, 'opisanie', point\_param\_opis\_name\_point,

width-ColumnByFieldName['name'].Width-

ColumnByFieldName['num'].Width-50-Admin\_add\_point\_param.Width-5);

Enabled:=true;

end;

sel:=true;

Admin\_pp\_DBGridSelectCell(self,1,backrow, sel);

act:='';

end;

procedure TAdminForm. Add\_pointExecute(Sender: TObject);

begin

// приводим все в исходное состояние

Show\_points\_table.Execute;

// таблица недоступна

Admin\_pp\_DBGrid.Enabled:= false;

// режим добавления для кнопки сохранения

act:='add';

// показываем панель ввода

Panel1.Hide;

Panel2.Show;

point\_image\_size\_znach\_label.Caption:=image\_size\_label;

//устанавливаем минимальное значение номера точки

mysql\_query(SQL\_DM.mysql\_query1, 'SELECT max(num) From '+table\_points\_name+';', 1);

Point\_param\_SE.MinValue:=SQL\_DM.mysql\_DSQ1.DataSet.FieldByName('max(num)').AsInteger+1;

Point\_param\_SE.MaxValue:=99;

Point\_param\_SE.Value:=Point\_param\_SE.MinValue;

point\_image.Picture:=nil;

end;

procedure TAdminForm. Red\_pointExecute(Sender: TObject);

begin

Admin\_pp\_DBGrid.Enabled := false;

act:='update';

Panel1.Hide;

Panel2.Show;

//копируем текущие значения в поля

Point\_param\_SE.Value:= strtoint(point\_param\_num\_znach.Caption);

point\_param\_name\_edit.Text:= point\_param\_name\_znach.Caption;

point\_param\_ops\_edit.Text:= point\_param\_opis\_znach.Caption;

point\_image.Picture:= pict.Picture;

pict.Picture.SaveToFile('tmp');

point\_image\_size\_znach\_label.Caption:=image\_size\_label\_name+GetFileSize('tmp');

DeleteFile('tmp');

backrow:=strtoint(point\_param\_num\_znach.Caption);

end;

procedure TAdminForm. Del\_pointExecute(Sender: TObject);

begin

if MessageBox(0,

PChar(del\_point\_question\_mes),

PChar(mess\_caption),

MB\_ICONQUESTION or MB\_OKCANCEL

)=mrOK then

try

mysql\_query(SQL\_DM.mysql\_query1,'DELETE FROM `'+table\_points\_name+

'` WHERE `num`='+inttostr(db\_grid\_cur\_row\_num)+';',2);

finally

MessageBox(0,

PChar(del\_point\_from\_database\_mes),

PChar(mess\_caption),

MB\_ICONINFORMATION or MB\_OK

);

Show\_points\_table.Execute;

end;

end;

////////////////////////////////

/// ПРОЦЕДУРЫ ДЛЯ ПАРАМЕТРОВ ///

////////////////////////////////

procedure TAdminForm.Show\_params\_tableExecute(Sender: TObject);

var sel:boolean;

begin

//ЭЛЕМЕНТЫ ИНТЕРФЕЙСА

//Панели

Panel2.Hide;

Panel1.Show;

//Панель с общей информацией

point\_param\_num\_znach.Caption:='';

point\_param\_name\_znach.Caption:='';

point\_param\_opis\_znach.Caption:='';

//Панель ввода

Point\_param\_SE.Value:=0;

point\_param\_name\_edit.Clear;

point\_param\_ops\_edit.Clear;

//все что относитмя к картинке

pict .hide;

pict.Picture:=nil;

point\_image\_label\_load .hide;

point\_image\_label .hide;

point\_image\_size\_label .hide;

point\_image\_size\_znach\_label .Hide;

point\_param\_\_ins\_num\_label.Caption:=param\_label\_name;

Point\_param\_label.Caption:=param\_label\_name;

//действия

Admin\_add\_point\_param.Action:=AdminForm.Add\_param;

Admin\_red\_point\_param.Action:=AdminForm.Red\_param;

Admin\_del\_point\_param.Action:=AdminForm.Del\_param;

//загрузка таблицы

db\_grid\_cur\_row\_num:=-1;

mysql\_query(SQL\_DM.mysql\_query3, 'SELECT id, name, opisanie From '+table\_param\_name+';', 1);

with Admin\_pp\_DBGrid do

begin

DataSource:=SQL\_DM.mysql\_DSQ3;

MainForm.Col\_resize(Admin\_pp\_DBGrid, 'id', point\_param\_num\_name, 45);

MainForm.Col\_resize(Admin\_pp\_DBGrid, 'name', point\_param\_name\_name, 100);

MainForm.Col\_resize(Admin\_pp\_DBGrid, 'opisanie', point\_param\_opis\_name\_param, width-ColumnByFieldName['name'].Width-

ColumnByFieldName['id'].Width-

50-Admin\_add\_point\_param.Width-5);

Enabled:=true;

end;

sel:=true;

Admin\_pp\_DBGridSelectCell(self,1,backrow, sel);

act:='';

end;

procedure TAdminForm. Add\_paramExecute(Sender: TObject);

begin

// см. Процедуру для точек

Show\_params\_table.Execute;

Admin\_pp\_DBGrid.Enabled := false;

act:='add';

Panel1.Hide;

Panel2.Show;

//устанавливаем минимальное значение номера параметра

mysql\_query(SQL\_DM.mysql\_query1, 'SELECT max(id) From '+table\_param\_name+';', 1);

Point\_param\_SE.MinValue:=SQL\_DM.mysql\_DSQ1.DataSet.FieldByName('max(id)').AsInteger+1;

Point\_param\_SE.MaxValue:=99;

Point\_param\_SE.Value:=Point\_param\_SE.MinValue;

end;

procedure TAdminForm. Red\_paramExecute(Sender: TObject);

begin

Admin\_pp\_DBGrid.Enabled:= false;

act:='update';

Panel1.Hide;

Panel2.Show;

//копируем текущие значения в поля

Point\_param\_SE.Value:=strtoint(point\_param\_num\_znach.Caption);

point\_param\_name\_edit.Text:=point\_param\_name\_znach.Caption;

point\_param\_ops\_edit.Text:=point\_param\_opis\_znach.Caption;

end;

procedure TAdminForm. Del\_paramExecute(Sender: TObject);

begin

//удаление параметра

if MessageBox(0,

PChar(del\_param\_question\_mes),

PChar(mess\_caption),

MB\_ICONQUESTION or MB\_OKCANCEL

)=mrOK then

try

mysql\_query(SQL\_DM.mysql\_query1, 'DELETE FROM `'+table\_param\_name+

'` WHERE `id`='+inttostr(db\_grid\_cur\_row\_num)+';',2);

finally

MessageBox(0,

PChar(del\_param\_from\_database\_mes),

PChar(mess\_caption),

MB\_ICONINFORMATION or MB\_OK

);

Show\_params\_table.Execute;

end;

end;

//вытаскивание картинок из dll

procedure TAdminForm.GetResourceAsJpeg(const resname: string; AInstance: Cardinal; aImage: TAdvPicture);

var

Stream: TResourceStream;

begin

try

Stream := TResourceStream.Create(AInstance, ResName, 'JPEG');

try

aImage.Picture := nil;

aImage.Picture.LoadFromStream(Stream);

aImage.Repaint;

finally

Stream.Free;

end;

except

on ex: EResNotFound do

begin

//здесь обработка исключения если ресурс не найден

end;

end;

end;

end.

# МОДУЛЬ BeginUnit.pas

unit BeginUnit;

interface

uses

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

Dialogs, StdCtrls, DataUnit, ConstUnit, MainUnit;

type

TBeginForm = class(TForm)

begin\_memo: TMemo;

procedure FormCreate(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

BeginForm: TBeginForm;

implementation

{$R \*.dfm}

procedure TBeginForm.FormCreate(Sender: TObject);

var database\_flag,

user\_flag,

userpriv\_flag,

table\_patient\_flag,

table\_izmereniy\_flag,

table\_param\_flag,

table\_znacheniya\_flag,

table\_points\_flag

: boolean;

priv\_str: string;

i,count: Integer;

begin

//флаги правильности работы программы,

database\_flag := false; //флаг существования базы данных

user\_flag := false; //флаг существования пользователя

userpriv\_flag := false; //флаг правильности привелегий пользователя

table\_patient\_flag := false;

table\_izmereniy\_flag := false;

table\_param\_flag := false;

table\_znacheniya\_flag := false;

table\_points\_flag := false;

//очищаем мемо и показываем форму

Begin\_memo.Clear;

BeginForm.Show;

//выводим первую запись в поле

Begin\_memo.Lines.Append(Begin\_mes\_check\_first);

//соединяемся с базой в качестве админа

with SQL\_DM.mysql\_con do

begin

loginprompt := Begin\_conn\_login\_prompt;

username := Begin\_conn\_username\_root;

server := Begin\_conn\_server;

password := Begin\_conn\_pass\_root;

try

connected := true;

except

//выводим сообщение об ошибке и прекращаем работу программы

Begin\_memo.Lines.Append(Begin\_mes\_fail\_connect);

if MessageBox(0,

PChar(begin\_mes\_terminate+#13+begin\_mes\_fail\_connect),

Pchar(begin\_caption\_error),

MB\_ICONERROR or MB\_OK

)=id\_OK then;

begin

beginForm.Close;

MainForm.Close;

end;

end;

connectdialog := SQL\_DM.mysql\_conn\_dialog;

end;

if SQL\_DM.mysql\_con.Connected then begin

Begin\_memo.Lines.Append(Begin\_mes\_root\_connected);

Begin\_memo.Lines.Append('');

Begin\_memo.Lines.Append(Begin\_mes\_check\_database);

//смотрим доступные базы данных на сервере

if mysql\_query(SQL\_DM.mysql\_query1, 'Show Databases;',1) then

begin

SQL\_DM.mysql\_DSQ1.DataSet.First;

//проходим все, если не найдем то флаг остается в false, а если найдем

//требуемую базу даннных - то в true

while not SQL\_DM.mysql\_DSQ1.DataSet.Eof do begin

if SQL\_DM.mysql\_DSQ1.DataSet.FieldByName('Database').AsString=database\_name then

database\_flag:=true;

SQL\_DM.mysql\_DSQ1.DataSet.Next;

end;

end;

//если база найдена то выводим сообщение, если нет, то пытаеммся создать

if database\_flag then

Begin\_memo.Lines.Append(Begin\_mes\_exist\_database)

else begin

Begin\_memo.Lines.Append(Begin\_mes\_notexist\_database);

Begin\_memo.Lines.Append(Begin\_mes\_create\_database);

if mysql\_query(SQL\_DM.mysql\_query1, 'CREATE DATABASE `rd` DEFAULT CHARACTER SET utf8 COLLATE utf8\_general\_ci;',2)

then

begin

//проверка на корректное создание базы данных

SQL\_DM.mysql\_DSQ1.Enabled:=false;

if mysql\_query(SQL\_DM.mysql\_query1, 'Show Databases;',1) then

begin

SQL\_DM.mysql\_DSQ1.Enabled:=true;

SQL\_DM.mysql\_DSQ1.DataSet.First;

while not SQL\_DM.mysql\_DSQ1.DataSet.Eof do begin

if SQL\_DM.mysql\_DSQ1.DataSet.FieldByName('Database').AsString=database\_name then

database\_flag:=true;

SQL\_DM.mysql\_DSQ1.DataSet.Next;

end;

end;

end;

if database\_flag then Begin\_memo.Lines.Append(Begin\_mes\_create\_database\_sucsess);

end;

//начинаем проверку существования пользователя

Begin\_memo.Lines.Append(Begin\_mes\_check\_user);

//выбираем нужную базу данных

mysql\_query(SQL\_DM.mysql\_query1, 'USE mysql;',2);

//проверяем наличие пользователя, если да, то устанавливаем флаг существования

//полтзователя

SQL\_DM.mysql\_DSQ1.Enabled:=false;

if mysql\_query(SQL\_DM.mysql\_query1, 'SELECT user from User;',1) then

begin

SQL\_DM.mysql\_DSQ1.DataSet.First;

while not SQL\_DM.mysql\_DSQ1.DataSet.Eof do begin

if SQL\_DM.mysql\_DSQ1.DataSet.FieldByName('user').AsString=user\_name then

user\_flag:=true;

SQL\_DM.mysql\_DSQ1.DataSet.Next;

end;

end;

if user\_flag then

Begin\_memo.Lines.Append(Begin\_mes\_check\_user\_sucsess)

else begin

//если пользователь не найден, то создаем его.

Begin\_memo.Lines.Append(Begin\_mes\_check\_user\_notsucsess);

Begin\_memo.Lines.Append(Begin\_mes\_user\_create);

SQL\_DM.mysql\_query1.Active:=false;

SQL\_DM.mysql\_query1.SQL.Clear;

SQL\_DM.mysql\_query1.SQL.Append('CREATE USER '''+user\_name+'''@''localhost'' IDENTIFIED BY '''+user\_pass+''' ;');

SQL\_DM.mysql\_query1.SQL.Append('');

SQL\_DM.mysql\_query1.SQL.Append('GRANT SELECT ,');

SQL\_DM.mysql\_query1.SQL.Append('INSERT ,');

SQL\_DM.mysql\_query1.SQL.Append('UPDATE ,');

SQL\_DM.mysql\_query1.SQL.Append('DELETE ,');

SQL\_DM.mysql\_query1.SQL.Append('CREATE ,');

SQL\_DM.mysql\_query1.SQL.Append('DROP , ');

SQL\_DM.mysql\_query1.SQL.Append('INDEX , ');

SQL\_DM.mysql\_query1.SQL.Append('ALTER ON \* . \* TO '''+user\_name+'''@''localhost'' IDENTIFIED BY '''+user\_pass+''' WITH MAX\_QUERIES\_PER\_HOUR 0 MAX\_CONNECTIONS\_PER\_HOUR 0 MAX\_UPDATES\_PER\_HOUR 0 MAX\_USER\_CONNECTIONS 0 ;');

try

SQL\_DM.mysql\_query1.Execute;

except

//

end;

//проверка на корректное создание пользователя

SQL\_DM.mysql\_DSQ1.Enabled:=false;

if mysql\_query(SQL\_DM.mysql\_query1, 'SELECT user from User;',1) then

begin

SQL\_DM.mysql\_DSQ1.DataSet.First;

while not SQL\_DM.mysql\_DSQ1.DataSet.Eof do begin

if SQL\_DM.mysql\_DSQ1.DataSet.FieldByName('user').AsString=user\_name then

user\_flag:=true;

SQL\_DM.mysql\_DSQ1.DataSet.Next;

end;

if user\_flag then Begin\_memo.Lines.Append(Begin\_mes\_user\_create\_suc);

end;

end;

//проверка на достаточность прав, осуществляется только

//при наличии пользователя

if user\_flag then

begin

Begin\_memo.Lines.Append(Begin\_mes\_check\_user\_priv);

if mysql\_query(SQL\_DM.mysql\_query1, 'SELECT \* from User Where user='''+user\_name+''';',1) then

begin

SQL\_DM.mysql\_DSQ1.DataSet.First;

priv\_str:='';

//в одну строку записываем все существующие права пользователя в виде Y/N

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Select\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Insert\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Update\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Delete\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Create\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Drop\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Reload\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Shutdown\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Process\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['File\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Grant\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['References\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Index\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Alter\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Show\_db\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Super\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Create\_tmp\_table\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Lock\_tables\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Execute\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Repl\_slave\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Repl\_client\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Create\_view\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Show\_view\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Create\_routine\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Alter\_routine\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Create\_user\_priv'];

count:=0;

//showmessage(priv\_str);

// showmessage(inttostr(length(priv\_str)));

for i:= 1 to length(priv\_str) do

if priv\_str[i]='Y' then inc(count);

if count=priv\_y\_count then

begin

userpriv\_flag:=true;

Begin\_memo.Lines.Append(Begin\_mes\_check\_user\_priv\_suc);

end

else

begin

//если права не соответствуют требуемым, то создаем их

Begin\_memo.Lines.Append(Begin\_mes\_check\_user\_priv\_notsuc);

SQL\_DM.mysql\_query1.Active:=false;

SQL\_DM.mysql\_query1.SQL.Clear;

SQL\_DM.mysql\_query1.SQL.Append('GRANT SELECT ,');

SQL\_DM.mysql\_query1.SQL.Append('INSERT ,');

SQL\_DM.mysql\_query1.SQL.Append('UPDATE ,');

SQL\_DM.mysql\_query1.SQL.Append('DELETE ,');

SQL\_DM.mysql\_query1.SQL.Append('CREATE ,');

SQL\_DM.mysql\_query1.SQL.Append('DROP , ');

SQL\_DM.mysql\_query1.SQL.Append('INDEX , ');

SQL\_DM.mysql\_query1.SQL.Append('ALTER ON \* . \* TO '''+user\_name+'''@''localhost'' IDENTIFIED BY '''+user\_pass+''' WITH MAX\_QUERIES\_PER\_HOUR 0 MAX\_CONNECTIONS\_PER\_HOUR 0 MAX\_UPDATES\_PER\_HOUR 0 MAX\_USER\_CONNECTIONS 0 ;');

try

SQL\_DM.mysql\_query1.Execute;

except

//

end;

//проверяем сделанные изменения

if mysql\_query(SQL\_DM.mysql\_query1, 'SELECT \* from User Where user='''+user\_name+''';',1) then

begin

SQL\_DM.mysql\_DSQ1.DataSet.First;

priv\_str:='';

//в одну строку записываем все существующие права пользователя в виде Y/N

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Select\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Insert\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Update\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Delete\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Create\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Drop\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Reload\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Shutdown\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Process\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['File\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Grant\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['References\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Index\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Alter\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Show\_db\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Super\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Create\_tmp\_table\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Lock\_tables\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Execute\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Repl\_slave\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Repl\_client\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Create\_view\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Show\_view\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Create\_routine\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Alter\_routine\_priv'];

priv\_str:=priv\_str+SQL\_DM.mysql\_DSQ1.DataSet.FieldValues['Create\_user\_priv'];

count:=0;

for i:= 0 to length(priv\_str)- 1 do

if priv\_str[i]='Y' then inc(count);

if count=priv\_y\_count then

begin

userpriv\_flag:=true;

Begin\_memo.Lines.Append(Begin\_mes\_check\_user\_priv\_change);

end

end;

end;

end;

end;

if user\_flag and userpriv\_flag and database\_flag then begin

Begin\_memo.Lines.Append('');

Begin\_memo.Lines.Append(Begin\_mes\_first\_stage\_complete);

Begin\_memo.Lines.Append('');

Begin\_memo.Lines.Append(Begin\_mes\_reconnect);

with SQL\_DM.mysql\_con do

begin

connected := false;

//sleep(300);

loginprompt := Begin\_conn\_login\_prompt;

username := user\_name;

server := Begin\_conn\_server;

password := user\_pass;

try

connect;

except

end;

connectdialog := SQL\_DM.mysql\_conn\_dialog;

end;

if SQL\_DM.mysql\_con.Connected then begin

Begin\_memo.Lines.Append(Begin\_mes\_reconnect\_sucsess );

//проверяем наличие таблиц

mysql\_query(SQL\_DM.mysql\_query1, 'USE '+database\_name+';',2);

if mysql\_query(SQL\_DM.mysql\_query1, 'Show Tables;',1) then

begin

SQL\_DM.mysql\_DSQ1.Enabled:=true;

SQL\_DM.mysql\_DSQ1.DataSet.First;

while not SQL\_DM.mysql\_DSQ1.DataSet.Eof do begin

if SQL\_DM.mysql\_DSQ1.DataSet.FieldByName('Tables\_in\_'+database\_name).AsString=table\_patient\_name then

table\_patient\_flag:=true;

if SQL\_DM.mysql\_DSQ1.DataSet.FieldByName('Tables\_in\_'+database\_name).AsString=table\_izm\_name then

table\_izmereniy\_flag:=true;

if SQL\_DM.mysql\_DSQ1.DataSet.FieldByName('Tables\_in\_'+database\_name).AsString=table\_param\_name then

table\_param\_flag:=true;

if SQL\_DM.mysql\_DSQ1.DataSet.FieldByName('Tables\_in\_'+database\_name).AsString=table\_znacheniya\_name then

table\_znacheniya\_flag:=true;

if SQL\_DM.mysql\_DSQ1.DataSet.FieldByName('Tables\_in\_'+database\_name).AsString=table\_points\_name then

table\_points\_flag:=true;

SQL\_DM.mysql\_DSQ1.DataSet.Next;

end;

//если таблица существует, то выводим сообщение об этом, если нет

//то создаем их

if table\_patient\_flag then Begin\_memo.Lines.Append(Begin\_mes\_table\_pat\_exist)

else begin

Begin\_memo.Lines.Append(Begin\_mes\_table\_not\_pat\_exist);

Begin\_memo.Lines.Append(Begin\_mes\_table\_pat\_create);

SQL\_DM.mysql\_query1.Active:=false;

SQL\_DM.mysql\_query1.SQL.Clear;

SQL\_DM.mysql\_query1.SQL.Append('CREATE TABLE `'+table\_patient\_name+'` (');

SQL\_DM.mysql\_query1.SQL.Append('`id` int(5) NOT NULL auto\_increment,');

SQL\_DM.mysql\_query1.SQL.Append('`fam` varchar(60) NOT NULL,');

SQL\_DM.mysql\_query1.SQL.Append('`name` varchar(60) NOT NULL,');

SQL\_DM.mysql\_query1.SQL.Append('`otch` varchar(60) NOT NULL,');

SQL\_DM.mysql\_query1.SQL.Append('`day` tinyint(3) unsigned NOT NULL,');

SQL\_DM.mysql\_query1.SQL.Append('`mon` tinyint(3) unsigned NOT NULL,');

SQL\_DM.mysql\_query1.SQL.Append('`year` smallint(4) unsigned NOT NULL,');

SQL\_DM.mysql\_query1.SQL.Append('`opisanie` text NOT NULL,');

SQL\_DM.mysql\_query1.SQL.Append('PRIMARY KEY (`id`)');

SQL\_DM.mysql\_query1.SQL.Append(') ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO\_INCREMENT=1 ;');

try

SQL\_DM.mysql\_query1.Execute;

except

//

end;

end;

if table\_izmereniy\_flag then Begin\_memo.Lines.Append(Begin\_mes\_table\_izm\_exist)

else begin

Begin\_memo.Lines.Append(Begin\_mes\_table\_not\_izm\_exist);

Begin\_memo.Lines.Append(Begin\_mes\_table\_izm\_create);

SQL\_DM.mysql\_query1.Active:=false;

SQL\_DM.mysql\_query1.SQL.Clear;

SQL\_DM.mysql\_query1.SQL.Append('CREATE TABLE `'+table\_izm\_name+'` (');

SQL\_DM.mysql\_query1.SQL.Append('`id` int(5) NOT NULL auto\_increment,');

SQL\_DM.mysql\_query1.SQL.Append('`date` datetime NOT NULL,');

SQL\_DM.mysql\_query1.SQL.Append('`patient\_num` int(11) NOT NULL default ''0'',');

SQL\_DM.mysql\_query1.SQL.Append('`opisanie` text NOT NULL,');

SQL\_DM.mysql\_query1.SQL.Append('PRIMARY KEY (`id`),');

SQL\_DM.mysql\_query1.SQL.Append('UNIQUE KEY `date` (`date`),');

SQL\_DM.mysql\_query1.SQL.Append('KEY `patient\_num` (`patient\_num`)');

SQL\_DM.mysql\_query1.SQL.Append(') ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO\_INCREMENT=1 ;');

try

SQL\_DM.mysql\_query1.Execute;

except

//

end;

end;

if table\_param\_flag then Begin\_memo.Lines.Append(Begin\_mes\_table\_param\_exist)

else begin

Begin\_memo.Lines.Append(Begin\_mes\_table\_not\_param\_exist);

Begin\_memo.Lines.Append(Begin\_mes\_table\_param\_create);

SQL\_DM.mysql\_query1.Active:=false;

SQL\_DM.mysql\_query1.SQL.Clear;

SQL\_DM.mysql\_query1.SQL.Append('CREATE TABLE `'+table\_param\_name+'` (');

SQL\_DM.mysql\_query1.SQL.Append('`id` tinyint(4) NOT NULL auto\_increment,');

SQL\_DM.mysql\_query1.SQL.Append('`name` varchar(5) NOT NULL,');

SQL\_DM.mysql\_query1.SQL.Append('`opisanie` varchar(100) NOT NULL,');

SQL\_DM.mysql\_query1.SQL.Append('PRIMARY KEY (`id`)');

SQL\_DM.mysql\_query1.SQL.Append(') ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO\_INCREMENT=1 ;');

try

SQL\_DM.mysql\_query1.Execute;

except

//

end;

end;

if table\_znacheniya\_flag then Begin\_memo.Lines.Append(Begin\_mes\_table\_znach\_exist)

else begin

Begin\_memo.Lines.Append(Begin\_mes\_table\_not\_znach\_exist);

Begin\_memo.Lines.Append(Begin\_mes\_table\_znach\_create);

SQL\_DM.mysql\_query1.Active:=false;

SQL\_DM.mysql\_query1.SQL.Clear;

SQL\_DM.mysql\_query1.SQL.Append('CREATE TABLE `'+table\_znacheniya\_name+'` (');

SQL\_DM.mysql\_query1.SQL.Append('`id` bigint NOT NULL auto\_increment,');

SQL\_DM.mysql\_query1.SQL.Append('`izm\_num` int(11) NOT NULL,');

SQL\_DM.mysql\_query1.SQL.Append('`param\_num` tinyint(4) NOT NULL,');

SQL\_DM.mysql\_query1.SQL.Append('`point\_num` tinyint(3) NOT NULL,');

SQL\_DM.mysql\_query1.SQL.Append('`povtor\_num` enum(''1'',''2'',''3'',''4'',''5'',''6'') NOT NULL,');

SQL\_DM.mysql\_query1.SQL.Append('`znach` float NOT NULL,');

SQL\_DM.mysql\_query1.SQL.Append('`znach\_UI` mediumtext NOT NULL,');

SQL\_DM.mysql\_query1.SQL.Append('PRIMARY KEY (`id`),');

SQL\_DM.mysql\_query1.SQL.Append('KEY `izm\_num` (`izm\_num`),');

SQL\_DM.mysql\_query1.SQL.Append('KEY `param\_num` (`param\_num`),');

SQL\_DM.mysql\_query1.SQL.Append('KEY `point\_num` (`point\_num`)');

SQL\_DM.mysql\_query1.SQL.Append(') ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO\_INCREMENT=1 ;');

SQL\_DM.mysql\_query1.SQL.Append('ALTER TABLE `rd\_znacheniya` ADD `znach\_UI` VARCHAR( 24000 ) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL AFTER `znach`;');

try

SQL\_DM.mysql\_query1.Execute;

except

//

end;

end;

if table\_points\_flag then Begin\_memo.Lines.Append(Begin\_mes\_table\_points\_exist)

else begin

Begin\_memo.Lines.Append(Begin\_mes\_table\_not\_points\_exist);

Begin\_memo.Lines.Append(Begin\_mes\_table\_points\_create);

SQL\_DM.mysql\_query1.Active:=false;

SQL\_DM.mysql\_query1.SQL.Clear;

SQL\_DM.mysql\_query1.SQL.Append('CREATE TABLE `'+table\_points\_name+'` (');

SQL\_DM.mysql\_query1.SQL.Append('`num` TINYINT( 3 ) NOT NULL ,');

SQL\_DM.mysql\_query1.SQL.Append('`opisanie` VARCHAR( 500 ) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL ,');

SQL\_DM.mysql\_query1.SQL.Append('`name` VARCHAR( 20 ) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL DEFAULT ''point'',');

SQL\_DM.mysql\_query1.SQL.Append('`image` LONGBLOB ,');

SQL\_DM.mysql\_query1.SQL.Append('PRIMARY KEY ( `num` )');

SQL\_DM.mysql\_query1.SQL.Append(') ENGINE = innodb CHARACTER SET utf8 COLLATE utf8\_general\_ci;');

try

SQL\_DM.mysql\_query1.Execute;

except

//

end;

end;

end;

//проверяем корректность создания таблиц

if mysql\_query(SQL\_DM.mysql\_query1, 'Show Tables;',1) then

begin

SQL\_DM.mysql\_DSQ1.Enabled:=true;

SQL\_DM.mysql\_DSQ1.DataSet.First;

while not SQL\_DM.mysql\_DSQ1.DataSet.Eof do begin

if SQL\_DM.mysql\_DSQ1.DataSet.FieldByName('Tables\_in\_'+database\_name).AsString=table\_patient\_name then

table\_patient\_flag:=true;

if SQL\_DM.mysql\_DSQ1.DataSet.FieldByName('Tables\_in\_'+database\_name).AsString=table\_izm\_name then

table\_izmereniy\_flag:=true;

if SQL\_DM.mysql\_DSQ1.DataSet.FieldByName('Tables\_in\_'+database\_name).AsString=table\_param\_name then

table\_param\_flag:=true;

if SQL\_DM.mysql\_DSQ1.DataSet.FieldByName('Tables\_in\_'+database\_name).AsString=table\_znacheniya\_name then

table\_znacheniya\_flag:=true;

if SQL\_DM.mysql\_DSQ1.DataSet.FieldByName('Tables\_in\_'+database\_name).AsString=table\_points\_name then

table\_points\_flag:=true;

SQL\_DM.mysql\_DSQ1.DataSet.Next;

end;

if not table\_patient\_flag and

table\_izmereniy\_flag and

table\_param\_flag and

table\_znacheniya\_flag and

table\_points\_flag then begin//обрабатываем полученные результаты

if table\_patient\_flag then Begin\_memo.Lines.Append(Begin\_mes\_table\_pat\_create\_sucsess)

else begin

//обработка ошибки

end;

if table\_izmereniy\_flag then Begin\_memo.Lines.Append(Begin\_mes\_table\_izm\_create\_sucsess)

else begin

//обработка ошибки

end;

if table\_param\_flag then Begin\_memo.Lines.Append(Begin\_mes\_table\_param\_create\_sucsess)

else begin

//обработка ошибки

end;

if table\_znacheniya\_flag then Begin\_memo.Lines.Append(Begin\_mes\_table\_znach\_create\_sucsess)

else begin

//обработка ошибки

end;

if table\_points\_flag then Begin\_memo.Lines.Append(Begin\_mes\_table\_points\_create\_sucsess)

else begin

//обработка ошибки

end;

end;

//создание ключей. Происходит в любом случае

Begin\_memo.Lines.Append(Begin\_mes\_keys\_create);

SQL\_DM.mysql\_query1.Active:=false;

SQL\_DM.mysql\_query1.SQL.Clear;

SQL\_DM.mysql\_query1.SQL.Append('ALTER TABLE `rd\_izmerenia`');

SQL\_DM.mysql\_query1.SQL.Append('ADD CONSTRAINT `rd\_izmerenie\_ibfk\_1` FOREIGN KEY (`patient\_num`) REFERENCES `rd\_patient` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;');

SQL\_DM.mysql\_query1.SQL.Append('ALTER TABLE `rd\_znacheniya` ADD FOREIGN KEY ( `izm\_num` ) REFERENCES `rd`.`rd\_izmerenia` (');

SQL\_DM.mysql\_query1.SQL.Append('`id`');

SQL\_DM.mysql\_query1.SQL.Append(') ON DELETE CASCADE ON UPDATE CASCADE ;');

SQL\_DM.mysql\_query1.SQL.Append('ALTER TABLE `rd\_znacheniya` ADD FOREIGN KEY ( `param\_num` ) REFERENCES `rd`.`rd\_param` (');

SQL\_DM.mysql\_query1.SQL.Append('`id`');

SQL\_DM.mysql\_query1.SQL.Append(') ON DELETE CASCADE ON UPDATE CASCADE ;');

SQL\_DM.mysql\_query1.SQL.Append('ALTER TABLE `rd\_znacheniya` ADD FOREIGN KEY ( `point\_num` ) REFERENCES `rd`.`rd\_points` (');

SQL\_DM.mysql\_query1.SQL.Append('`num`');

SQL\_DM.mysql\_query1.SQL.Append(') ON DELETE CASCADE ON UPDATE CASCADE ;');

try

SQL\_DM.mysql\_query1.Execute;

except

//

end;

if table\_patient\_flag and

table\_izmereniy\_flag and

table\_param\_flag and

table\_znacheniya\_flag and

table\_points\_flag then

begin

Begin\_memo.Lines.Append('');

Begin\_memo.Lines.Append(Begin\_mes\_table\_sucsess);

Begin\_memo.Lines.Append('');

Begin\_memo.Lines.Append(Begin\_mes\_check\_passed);

Begin\_memo.Lines.Append('');

Begin\_memo.Lines.Append(Begin\_mes\_program\_start);

//sleep(1000);

self.Close;

end;

end;

end;

end

else begin

if MessageBox(0,

PChar(begin\_mes\_second\_connect\_false),

Pchar(begin\_caption\_error),

MB\_ICONINFORMATION or MB\_OK

)=id\_OK then;

begin

BeginForm.Close;

MainForm.Close;

end;//выход из программы

end;

end;

end;

end.

# МОДУЛЬ ConstUnit.pas

unit ConstUnit;

interface

uses SysUtils, MyAccess, Classes, Dialogs, MainUnit, Graphics, dateUtils;

const

report\_error = 1; //1 - ошибки sql не показываются

//2 - ошибки sql показываются

Begin\_conn\_login\_prompt = false;

Begin\_conn\_username\_root = 'root';

Begin\_conn\_server = 'localhost';

Begin\_conn\_pass\_root = '';

database\_name = 'rd';

user\_name = 'rd';

user\_pass = 'rd\_pass';

priv\_y\_count = 8; //количество привелегий у пользователя базы

table\_izm\_name = 'rd\_izmerenia';

table\_patient\_name = 'rd\_patient';

table\_param\_name = 'rd\_param';

table\_znacheniya\_name = 'rd\_znacheniya';

table\_points\_name = 'rd\_points';

izm\_form\_name = 'Проведение измрения';

izm\_device\_ready = 'Прибор готов к измерению';

izm\_get\_data = 'Получаем данные...';

izm\_mes\_decode = 'Декодирование посылки...';

begin\_caption\_error = 'Ошибка при инициализации программы';

Begin\_mes\_check\_first = 'Проверка на первый запуск программы....';

Begin\_mes\_root\_connected = 'Соединение с сервером баз данных установлено!';

begin\_mes\_fail\_connect = 'Невозможно найти сервер базы данных или подключиться к нему.'+#13+

'Проверьте параметры подключения и запущен ли сервер';

begin\_mes\_terminate = 'Программа прекращает свою работу...';

begin\_mes\_second\_connect\_false = 'Подключение не установлено. Программа завершает свою работу...';

Begin\_mes\_check\_database = '1. Проверяем наличие базы данных...';

Begin\_mes\_exist\_database = ' База данных "'+database\_name+'" существует';

Begin\_mes\_notexist\_database = ' База данных "'+database\_name+'" не существует!!!';

Begin\_mes\_create\_database = ' Попытка создать базу данных "'+database\_name+'"...';

Begin\_mes\_create\_database\_sucsess = ' База данных "'+database\_name+'" успешно создана!';

Begin\_mes\_check\_user = '2. Проверяем наличие пользователя...';

Begin\_mes\_check\_user\_sucsess = ' Пользователь "'+user\_name+'" обнаружен';

Begin\_mes\_check\_user\_notsucsess = ' Пользователь "'+user\_name+'"не обнаружен!!!';

Begin\_mes\_user\_create = ' Попытка создать пользователя "'+user\_name+'"...';

Begin\_mes\_user\_create\_suc = ' Пользователь "'+user\_name+'" успешно добавлен!';

Begin\_mes\_check\_user\_priv = '3. Проверяем пользователя "'+user\_name+'" на предмет достаточности прав для работы с программой';

Begin\_mes\_check\_user\_priv\_suc = ' Права пользователя подтверждены!';

Begin\_mes\_check\_user\_priv\_notsuc = ' Права пользователя не соответствуют заданным. Попытка изменить права...';

Begin\_mes\_check\_user\_priv\_change = ' Права пользователя успешно изменены!';

Begin\_mes\_first\_stage\_complete = 'Первый этап проверки успешно завершен!!!';

Begin\_mes\_reconnect = 'Устанавливаем повторное соединение с базой данных...';

Begin\_mes\_reconnect\_sucsess = 'Соединение успешно установлено';

Begin\_mes\_table\_pat\_exist = ' Таблица '''+table\_patient\_name+''' для пациентов существует';

Begin\_mes\_table\_izm\_exist = ' Таблица '''+table\_izm\_name+''' для измерений существует';

Begin\_mes\_table\_param\_exist = ' Таблица '''+table\_param\_name+''' для параметров существует';

Begin\_mes\_table\_znach\_exist = ' Таблица '''+table\_znacheniya\_name+''' для значений измерений существует';

Begin\_mes\_table\_points\_exist = ' Таблица '''+table\_points\_name+''' для описания точек существует';

Begin\_mes\_table\_not\_pat\_exist = ' Таблица '''+table\_patient\_name+''' для пациентов не существует';

Begin\_mes\_table\_not\_izm\_exist = ' Таблица '''+table\_izm\_name+''' для измерений не существует';

Begin\_mes\_table\_not\_param\_exist = ' Таблица '''+table\_param\_name+''' для параметров не существует';

Begin\_mes\_table\_not\_znach\_exist = ' Таблица '''+table\_znacheniya\_name+''' для значений измерений не существует';

Begin\_mes\_table\_not\_points\_exist = ' Таблица '''+table\_points\_name+''' для описания точек не существует';

Begin\_mes\_table\_pat\_create = ' Попытка создать таблицу '''+table\_patient\_name+''' для пациентов... ';

Begin\_mes\_table\_izm\_create = ' Попытка создать таблицу '''+table\_izm\_name+''' для измерений...';

Begin\_mes\_table\_param\_create = ' Попытка создать таблицу '''+table\_param\_name+''' для параметров...';

Begin\_mes\_table\_znach\_create = ' Попытка создать таблицу '''+table\_znacheniya\_name+''' для значений измерений...';

Begin\_mes\_table\_points\_create = ' Попытка создать таблицу '''+table\_points\_name+''' для описания точек...';

Begin\_mes\_table\_pat\_create\_sucsess = ' Таблица '''+table\_patient\_name+''' для пациентов успешно создана';

Begin\_mes\_table\_izm\_create\_sucsess = ' Таблица '''+table\_izm\_name+''' для измерений успешно создана';

Begin\_mes\_table\_param\_create\_sucsess= ' Таблица '''+table\_param\_name+''' для параметров успешно создана';

Begin\_mes\_table\_znach\_create\_sucsess= ' Таблица '''+table\_znacheniya\_name+''' для значений измерений успешно создана';

Begin\_mes\_table\_points\_create\_sucsess= ' Таблица '''+table\_points\_name+''' для описания точек успешно создана';

Begin\_mes\_keys\_create = ' Попытка создания внешних ключей...';

Begin\_mes\_table\_sucsess = 'Второй этап проверки пройден!';

Begin\_mes\_check\_passed = 'ПРОВЕРКА НА КОРРЕКТНОСТЬ ПАРМЕТРОВ СОЕДИНЕНИЯ С БАЗОЙ ДАННЫХ ПРОЙДЕНА!!!';

Begin\_mes\_program\_start = 'Запуск программы...';

/////////////////////////////////////////////////////////////////

/// Сообщения программы ///

/////////////////////////////////////////////////////////////////

mess\_caption = 'RD Project. БМТ1';

image\_size\_label = 'Текущий размер: ';

add\_point\_to\_database\_mes = 'Точка успешно записана в базу данных!';

add\_param\_to\_database\_mes = 'Параметр успешно записан в базу данных!';

red\_point\_in\_database\_mes = 'Точка успешно отредактирована!';

red\_param\_in\_database\_mes = 'Параметр успешно отредактирован!';

del\_param\_from\_database\_mes = 'Параметр успешно удален из базы данных.'+#13+

'Также удалены все измерения этого параметра';

del\_point\_from\_database\_mes = 'Точка успешно удалена из базы данных'+#13+

'Также удалены все измерения, связанные с этой точкой';

add\_pat\_to\_database\_mes = 'Информация о новом пациенте успешно записана!';

del\_pat\_from\_database\_mes = 'Информация о пациенте удалена из базы данных!'+#13+

'Также удалена информация о всех измерениях с участием этого пациента';

red\_pat\_in\_database\_mes = 'Информация о пациенте успешно отредактирована!';

izm\_data\_lost = 'Вы уверены, что хотите закрыть окно измерений?';

izm\_izm\_end = 'Измерение закончилось. Все точки замерены';

izm\_caption\_message = 'RD\_project. Измерение сопротивления БАТ';

del\_point\_question\_mes = 'Вы выбрали удаление точки. При будет удалена '+

'вся информация об измерениях и значениях параметров, '+

'связанных с этой точкой. Вы хотите продолжить?';

del\_param\_question\_mes = 'Вы выбрали удаление параметра. При будет удалена '+

'вся информация об измерениях и значениях параметров, '+

'связанных с этим параметром. Вы хотите продолжить?';

del\_measure\_question\_mes = 'Вы уверены что хотите удаить всю информацию об измерении?';

del\_patient\_question\_mes = 'Вы уверены что хотите удалить информацию о пациенте. '+

'Также будет удалена вся информация о его измерениях';

make\_defaut\_message = 'ВНИМАНИЕ! Все данные после выполнения этой операции будут Уничтожены! Вы уверены в продолжении?';

make\_default\_dll\_error = 'Не найдена библиотека pictures.dll!!!';

/////////////////////////////////////////////////////////////////

/// МЕТКИ и ТЕКСТ В ПРОГРАММЕ ///

////////////////////////////////////////////////////////////////

////////////////////////

/// MAIN UNIT //

////////////////////////

status\_bar\_database\_connection\_exisits = 'Подключение к базе данных: Есть';

status\_bar\_database\_connection\_not\_exisits = 'Подключение к базе данных: <b><FONT color="#FF0000"> НЕТ</font></b>';

status\_bar\_device\_connected = 'Состояние прибора: Подключен';

status\_bar\_device\_not\_connected = 'Состояние прибора: <b><FONT color="#FF0000"> НЕ ПОДКЛЮЧЕН</font></b>';

status\_bar\_number\_of\_measurments = 'Проведено исследований: ';

//Закладка выбора измерений

main\_unit\_measure\_choise\_not\_selected = 'Измерение не выбрано...';

//Закладка графики

main\_unit\_graphics\_compress = 'сжато в ';

main\_unit\_graphics\_compress\_unit = ' раз';

main\_unit\_graphics\_title\_R = 'Сопротивление БАТ, МОм';

main\_unit\_graphics\_title\_G0 = 'Проводиость БАТ, 1/Ом\*10^6';

main\_unit\_graphics\_title\_G1 = 'Проводиость БАТ, 1-ая гармоника, 1/Ом\*10^6';

main\_unit\_graphics\_title\_G2 = 'Проводиость БАТ, 2-ая гармоника, 1/Ом\*10^6';

main\_unit\_graphics\_title\_G3 = 'Проводиость БАТ, 3-ая гармоника, 1/Ом\*10^6';

main\_unit\_graphics\_title\_C = 'Емкость БАТ, нФ';

main\_unit\_graphics\_title\_Usm = '';

main\_unit\_graphics\_title\_Ism = '';

//// Закладка диагноза

summ\_left\_hands = 'Сумма левой=';

summ\_right\_hands = 'Сумма правой=';

summ\_hands = 'Сумма рук=';

summ\_left\_legs = 'Сумма левой=';

summ\_right\_legs = 'Сумма правой=';

summ\_legs = 'Сумма ног=';

summ\_all = 'Сумма =';

diagnoz\_table\_unit = 'МОм';

////////////////////////

/// ADMIN UNIT //

////////////////////////

params\_default\_name\_opisnie\_1 = 'Значения напряжения';

params\_default\_name\_opisnie\_2 = 'Значения тока';

params\_default\_name\_opisnie\_3 = 'Сопротивление';

params\_default\_name\_opisnie\_4 = 'Проводимость';

params\_default\_name\_opisnie\_5 = 'Первая гармоника проводимости';

params\_default\_name\_opisnie\_6 = 'Вторая гармоника проводимости';

params\_default\_name\_opisnie\_7 = 'Третья гармоника проводимости';

params\_default\_name\_opisnie\_8 = 'Емкость';

params\_default\_name\_opisnie\_9 = 'Напряжение смещения';

params\_default\_name\_opisnie\_10= 'Ток смещения';

params\_default\_name1 = 'U';

params\_default\_name2 = 'I';

params\_default\_name3 = 'R';

params\_default\_name4 = 'G0';

params\_default\_name5 = 'G1';

params\_default\_name6 = 'G2';

params\_default\_name7 = 'G3';

params\_default\_name8 = 'C';

params\_default\_name9 = 'Uсм';

params\_default\_name10= 'Iсм';

point\_label\_name = 'Точка №:';

param\_label\_name = 'Параметр №:';

image\_size\_label\_name = 'Текущий размер: ';

////////////////////////

/// IZM UNIT //

////////////////////////

izm\_stop\_measurement = 'Досрочно закончить измерение';

izm\_unit\_end\_measurment\_btn\_label = 'Закончить измерение';

izm\_unit\_measurment\_caption\_label = 'Измерение ';

////////////////////////

/// DeviceRD UNIT //

////////////////////////

devicerd\_device\_just\_connected = 'Прибор подключен. Перезапустите программу';

/////////////////////////////////////////////////////////////////

/// Названия колонок в таблицах //

/////////////////////////////////////////////////////////////////

/// Таблицы точек и параметров

point\_param\_num\_name = ' Номер';

point\_param\_name\_name = ' Название';

point\_param\_opis\_name\_param = ' Описание параметра';

point\_param\_opis\_name\_point = ' Описание точки';

/// Таблица пациентов

new\_izm\_num\_name = ' №';

new\_izm\_fam\_name = ' Фамилия';

new\_izm\_name\_name = ' Имя';

new\_izm\_otch\_name = ' Отчество';

new\_izm\_day\_name = ' Число';

new\_izm\_mon\_name = ' Месяц';

new\_izm\_year\_name = ' Год';

new\_izm\_opis\_name = ' Примечание';

function mysql\_query(used\_query:TMyQuery; zapros:String; mode:byte):boolean;

function BytesToStr(const i64Size: Int64): string;

function GetFileSize(namefile: string): string;

procedure load\_info\_to\_diagnoz\_tables(create\_files:boolean);

function Rounder(Value: extended; Decimals: Integer): extended;

implementation

uses DataUnit;

function Rounder(Value: extended; Decimals: Integer): extended;

var

j: Integer;

A: extended;

begin

A := 1;

case Decimals of

0: A := 1;

1: A := 10;

else

for j := 1 to Decimals do

A := A \* 10;

end;

Result := Int((Value \* A) + 0.5) / A;

end;

function mysql\_query(used\_query:TMyQuery; zapros:String; mode:byte):boolean;

begin

used\_query.Connection:=SQL\_DM.mysql\_con;

used\_query.Active:=false;

used\_query.SQL.Clear;

used\_query.SQL.Append(zapros);

try

if mode=1 then used\_query.Active:=true

else

if mode=2 then used\_query.Execute;

except

//

end;

if (used\_query.Active) or (not used\_query.Executing) then result:=true

else result:=false;

end;

function BytesToStr(const i64Size: Int64): string;

const

i64GB = 1024 \* 1024 \* 1024;

i64MB = 1024 \* 1024;

i64KB = 1024;

begin

if i64Size div i64GB > 0 then

Result := Format('%.2f GB', [i64Size / i64GB])

else if i64Size div i64MB > 0 then

Result := Format('%.2f MB', [i64Size / i64MB])

else if i64Size div i64KB > 0 then

Result := Format('%.2f KB', [i64Size / i64KB])

else

Result := IntToStr(i64Size) + ' Byte(s)';

end;

function GetFileSize(namefile: string): string;

var

InfoFile: TSearchRec;

AttrFile: Integer;

ErrorReturn: Integer;

begin

AttrFile := $0000003F; {Any file}

ErrorReturn := FindFirst(namefile, AttrFile, InfoFile);

if ErrorReturn <> 0 then

Result := '' {в случае, если файл не найден}

else

Result := BytesToStr(InfoFile.Size); {Размер файла в байтах}

FindClose(InfoFile);

end;

procedure load\_info\_to\_diagnoz\_tables(create\_files:boolean);

var stream:TFileStream;

begin

//Флаг, который устанавливает, будут ли заново создаваться файлы \*.dat

if create\_files then

begin

//загружаем начальную информацию и форматируем колонки в диагнозе

with MainForm.look\_res\_diagnoz\_data\_StringGrid do

begin

//Содержание

//Надписи

Cells[0,0]:='РУКИ';

Cells[6,0]:='НОГИ';

Cells[0,1]:='Левая рука';

Cells[3,1]:='Правая рука';

Cells[6,1]:='Левая нога';

Cells[9,1]:='Правая нога';

Cells[0,2]:='Инь';

Cells[0,7]:='Ян';

Cells[3,2]:='Инь';

Cells[3,7]:='Ян';

Cells[6,2]:='Инь';

Cells[6,7]:='Ян';

Cells[9,2]:='Инь';

Cells[9,7]:='Ян';

Cells[0,12]:='Сумма левой =';

Cells[3,12]:='Сумма правой =';

Cells[6,12]:='Сумма левой =';

Cells[9,12]:='Сумма правой =';

Cells[0,13]:='Сумма рук =';

Cells[6,13]:='Сумма ног =';

CellProperties[0,0].FontStyle:=[fsBold];

CellProperties[6,0].FontStyle:=[fsBold];

CellProperties[0,1].FontStyle:=[fsBold];

CellProperties[3,1].FontStyle:=[fsBold];

CellProperties[6,1].FontStyle:=[fsBold];

CellProperties[9,1].FontStyle:=[fsBold];

CellProperties[0,2].FontStyle:=[fsBold];

CellProperties[0,7].FontStyle:=[fsBold];

CellProperties[3,2].FontStyle:=[fsBold];

CellProperties[3,7].FontStyle:=[fsBold];

CellProperties[6,2].FontStyle:=[fsBold];

CellProperties[6,7].FontStyle:=[fsBold];

CellProperties[9,2].FontStyle:=[fsBold];

CellProperties[9,7].FontStyle:=[fsBold];

CellProperties[0,6].FontStyle:=[fsBold];

CellProperties[3,6].FontStyle:=[fsBold];

CellProperties[6,6].FontStyle:=[fsBold];

CellProperties[9,6].FontStyle:=[fsBold];

CellProperties[0,11].FontStyle:=[fsBold];

CellProperties[3,11].FontStyle:=[fsBold];

CellProperties[6,11].FontStyle:=[fsBold];

CellProperties[9,11].FontStyle:=[fsBold];

CellProperties[0,12].FontStyle:=[fsBold];

CellProperties[3,12].FontStyle:=[fsBold];

CellProperties[6,12].FontStyle:=[fsBold];

CellProperties[9,12].FontStyle:=[fsBold];

CellProperties[0,13].FontStyle:=[fsBold];

CellProperties[6,13].FontStyle:=[fsBold];

CellProperties[1,6].FontStyle:=[fsBold];

CellProperties[2,6].FontStyle:=[fsBold];

CellProperties[4,6].FontStyle:=[fsBold];

CellProperties[5,6].FontStyle:=[fsBold];

CellProperties[7,6].FontStyle:=[fsBold];

CellProperties[8,6].FontStyle:=[fsBold];

CellProperties[10,6].FontStyle:=[fsBold];

CellProperties[11,6].FontStyle:=[fsBold];

CellProperties[1,11].FontStyle:=[fsBold];

CellProperties[2,11].FontStyle:=[fsBold];

CellProperties[4,11].FontStyle:=[fsBold];

CellProperties[5,11].FontStyle:=[fsBold];

CellProperties[7,11].FontStyle:=[fsBold];

CellProperties[8,11].FontStyle:=[fsBold];

CellProperties[10,11].FontStyle:=[fsBold];

CellProperties[11,11].FontStyle:=[fsBold];

//Объединение

MergeCells(0,0,6,1); //руки

MergeCells(6,0,6,1); //ноги

MergeCells(0,1,3,1); //левая рука

MergeCells(3,1,3,1); //правая рука

MergeCells(6,1,3,1); //левая нога

MergeCells(9,1,3,1); //правая нога

MergeCells(0,2,3,1); //Инь и Ян

MergeCells(3,2,3,1); //Инь и Ян

MergeCells(6,2,3,1); //Инь и Ян

MergeCells(9,2,3,1); //Инь и Ян

MergeCells(0,7,3,1); //Инь и Ян

MergeCells(3,7,3,1); //Инь и Ян

MergeCells(6,7,3,1); //Инь и Ян

MergeCells(9,7,3,1); //Инь и Ян

MergeCells(0,13,6,1); //сумма рук

MergeCells(6,13,6,1); //сумма ног

MergeCells(0,12,3,1); //сумма руки

MergeCells(3,12,3,1); //сумма руки

MergeCells(6,12,3,1); //сумма ноги

MergeCells(9,12,3,1); //сумма ноги

// fmOpenRead = $0000;

// fmOpenWrite = $0001;

// fmOpenReadWrite = $0002;

stream:=TFileStream.Create('table1.dat',fmCreate);

stream.Position:=0;

//stream.Write(sizeof() 'Таблица коэффициентов',strlen('Таблица коэффициентов'))));

SaveToBinStream(stream);

stream.Destroy;

end;

with MainForm.look\_res\_diagnoz\_help\_StringGrid do

begin

Clear;

LoadFromCSV('Справочные материалы. Методика Риодораку. точки.csv');

InsertRows(0,1);

Cells[0,0]:=Cells[0,1];

MergeCells(0,0,1,3);

Cells[1,0]:=Cells[1,1];

MergeCells(1,0,1,3);

Cells[2,0]:=Cells[2,1];

MergeCells(2,0,6,1);

CellProperties[2,0].Alignment:=taCenter;

Cells[2,1]:=Cells[2,2];

Cells[3,1]:=Cells[3,2];

Cells[4,1]:=Cells[4,2];

Cells[5,1]:=Cells[5,2];

Cells[6,1]:=Cells[6,2];

Cells[7,1]:=Cells[7,2];

CellProperties[2,1].Alignment:=taCenter;

CellProperties[3,1].Alignment:=taCenter;

CellProperties[4,1].Alignment:=taCenter;

CellProperties[5,1].Alignment:=taCenter;

CellProperties[6,1].Alignment:=taCenter;

CellProperties[7,1].Alignment:=taCenter;

CellProperties[8,0].Alignment:=taCenter;

MergeCells(2,1,1,2);

MergeCells(3,1,1,2);

MergeCells(4,1,1,2);

MergeCells(5,1,1,2);

MergeCells(6,1,1,2);

MergeCells(7,1,1,2);

Cells[8,0]:=Cells[8,1];

MergeCells(8,0,1,3);

stream:=TFileStream.Create('table2.dat',fmCreate);

stream.Position:=0;

//stream.Write(sizeof() 'Таблица коэффициентов',strlen('Таблица коэффициентов'))));

SaveToBinStream(stream);

stream.Destroy;

end;

with MainForm.look\_res\_diagnoz\_help\_StringGrid do

begin

Clear;

ColCount:=4;

LoadFromCSV('Справочные материалы. Методика Риодораку. коэф.csv');

MergeCells(0,0,4,2);

CellProperties[0,0].FontStyle:=[fsBold];

CellProperties[0,0].Alignment:=taCenter;

stream:=TFileStream.Create('table3.dat',fmCreate);

stream.Position:=0;

//stream.Write(sizeof() 'Таблица коэффициентов',strlen('Таблица коэффициентов'))));

SaveToBinStream(stream);

stream.Destroy;

end;

end;

stream:=TFileStream.Create('table1.dat',fmOpenRead);

MainForm.look\_res\_diagnoz\_data\_StringGrid.LoadFromBinStream(stream);

stream.Destroy;

MainForm.look\_res\_diagnoz\_points\_table\_switchClick(MainForm);

end;

end.

# МОДУЛЬ DataUnit.pas

unit DataUnit;

interface

uses

SysUtils, Classes, dialogs, DBAccess, MyDacVcl, DB, MyAccess,

MyDacMonitor, MySQLMonitor, MemDS, DASQLMonitor, DADump, MyDump, MyBackup,

WideStrings, DBXMySql, SqlExpr;

type

TSQL\_DM = class(TDataModule)

mysql\_query1: TMyQuery;

mysql\_query2: TMyQuery;

MySQL\_mon: TMySQLMonitor;

mysql\_DSQ2: TMyDataSource;

mysql\_DSQ1: TMyDataSource;

mysql\_conn\_dialog: TMyConnectDialog;

mysql\_query3: TMyQuery;

mysql\_DSQ3: TMyDataSource;

mysql\_DSQ4: TMyDataSource;

mysql\_query4: TMyQuery;

mysql\_DSQ5: TMyDataSource;

mysql\_query5: TMyQuery;

MyBackup1: TMyBackup;

MyDump1: TMyDump;

mysql\_con: TMyConnection;

mysql\_DSQ6: TMyDataSource;

mysql\_query6: TMyQuery;

procedure mysql\_conError(Sender: TObject; E: EDAError; var Fail: Boolean);

procedure mysql\_query1PostError(DataSet: TDataSet; E: EDatabaseError;

var Action: TDataAction);

procedure mysql\_conBeforeConnect(Sender: TObject);

procedure MySQL\_monSQL(Sender: TObject; Text: string; Flag: TDATraceFlag);

procedure mysql\_query4PostError(DataSet: TDataSet; E: EDatabaseError;

var Action: TDataAction);

private

{ Private declarations }

public

{ Public declarations }

end;

var

SQL\_DM: TSQL\_DM;

implementation

uses ConstUnit;

{$R \*.dfm}

procedure TSQL\_DM.mysql\_conBeforeConnect(Sender: TObject);

begin

mysql\_con.Options.Charset:='cp1251';

end;

procedure TSQL\_DM.mysql\_conError(Sender: TObject; E: EDAError;

var Fail: Boolean);

begin

if (report\_error=2) then showmessage (inttostr(E.ErrorCode)+' '+E.Message);

end;

procedure TSQL\_DM.MySQL\_monSQL(Sender: TObject; Text: string;

Flag: TDATraceFlag);

begin

//

end;

procedure TSQL\_DM.mysql\_query1PostError(DataSet: TDataSet; E: EDatabaseError;

var Action: TDataAction);

begin

if (report\_error=2) or (report\_error=1) then showmessage(E.Message);

end;

procedure TSQL\_DM.mysql\_query4PostError(DataSet: TDataSet; E: EDatabaseError;

var Action: TDataAction);

begin

if (report\_error=2) or (report\_error=1) then showmessage(E.Message);

end;

end.

# МОДУЛЬ DeviceThread.pas

interface

uses

Classes, Windows, SysUtils, D2xxUnit, Dialogs, DeviceRD, IzmUnit, DateUtils;

type TIzmResult = IzmUnit.TIzmResult;

type

device\_thread\_th = class(TThread)

public

destructor Destroy; override;

var

points\_amount,

repeat\_amount,

params\_amount : integer;

private

decoded\_message: params\_record; { Private declarations }

protected

procedure Execute; override;

procedure Get\_message;

procedure Update;

procedure Update\_dev\_cond\_string\_getdata;

procedure Update\_dev\_cond\_string\_ready;

procedure Update\_dev\_cond\_string\_decode;

procedure Update\_message\_count;

procedure Check\_and\_load\_data;

procedure Check\_errors;

end;

Var

num: integer;

point,povtor: integer;

izm\_result: TIzmResult;

message\_: message\_Array;

execute\_t: boolean;

implementation

uses MainUnit, ConstUnit, DataUnit;

{ Important: Methods and properties of objects in visual components can only be

used in a method called using Synchronize, for example,

Synchronize(UpdateCaption);

and UpdateCaption could look like,

procedure device\_thread.UpdateCaption;

begin

Form1.Caption := 'Updated in a thread';

end; }

{ device\_thread }

destructor device\_thread\_th.Destroy;

begin

//Suspend;

//inherited;

//здесь выполняй все действия, что должны выполниться при разрушении потока,

// например пошли sendmessage "родительскому" окну

//что поток закончил свою работу (при этом Synchronize не нужна)

end;

procedure device\_thread\_th.Execute;

begin

self.OnTerminate := nil;

Get\_message;

end;

procedure device\_thread\_th.Get\_message;

var i,

j,

k,m: integer;

message\_begin: boolean;

begin

points\_amount:=24;

repeat\_amount:=5;

params\_amount:=7;

//устанавливаем длину массива результатов измерений

SetLength(izm\_result,points\_amount+1,repeat\_amount+1);

//инициализируем массив результатов измерений

for i:=1 to points\_amount do

for j:=1 to repeat\_amount do

begin

izm\_result[i,j].R:=0;

izm\_result[i,j].G0:=0;

izm\_result[i,j].G1:=0;

izm\_result[i,j].G2:=0;

izm\_result[i,j].G3:=0;

izm\_result[i,j].C:=0;

izm\_result[i,j].Usm:=0;

izm\_result[i,j].R\_e:=0;

izm\_result[i,j].G0\_e:=0;

izm\_result[i,j].G1\_e:=0;

izm\_result[i,j].G2\_e:=0;

izm\_result[i,j].G3\_e:=0;

izm\_result[i,j].C\_e:=0;

izm\_result[i,j].Usm\_e:=0;

for k:= 0 to 2400 do begin

izm\_result[i,j].U\_izm[k]:=0;

izm\_result[i,j].I\_izm[k]:=0;

end;

end;

j:=0;

num:=0;

povtor:=0;

point:=1;

message\_begin:=false;

for i:=0 to length(message\_)-1 do message\_[i]:=0;

execute\_t:=true;

//ждем сообщение

Purge\_USB\_Device\_In;

Purge\_USB\_Device\_Out;

Reset\_USB\_Device;

if IzmForm.Visible then

while execute\_t do

begin

repeat

Get\_USB\_Device\_Status; //проверяем буфер устройства

//в этом месте идет загрузка на 100%. Вопрос в том как справится!!!

until FT\_Q\_Bytes > 0;

Read\_USB\_Device\_Buffer(1); //читаем буфер устройства

i:=FT\_In\_Buffer[0]; //читаем первое число

if i=240 then

begin

for m:=0 to length(message\_)-1 do message\_[m]:=0;

message\_begin:=true;//посылка пошла

//Выводим сообщение о получении данных

//прогресс бар измерения в 0

Synchronize(Update\_dev\_cond\_string\_getdata);

end;

if message\_begin then

begin

//запоминаем число из посылки

message\_[j]:=i;

inc(j);

end;

if i=255 then begin //посылка закончилась

j:=0;

message\_begin:=false;

Purge\_USB\_Device\_In;

Purge\_USB\_Device\_Out;

Reset\_USB\_Device;

//Вывод сообщения о декодировании посылки

Synchronize(Update\_dev\_cond\_string\_decode);

/// Обрабатываем посылку

/// Выводим результаты в лист

/// Выводим сообщения об ошибках

Synchronize(Update);

//Вывод сообщения о готовности прибора к измерению

Synchronize(Update\_dev\_cond\_string\_ready);

// Уведичиваем порядковый номер замера

inc(num);

//Увеличиваем значение повтора

inc(povtor);

//Загружаем информацию в базу данных

Synchronize(Check\_and\_load\_data);

//проверяем условие перехода на новую точку

Synchronize(Update\_message\_count);

for i:=0 to length(message\_)-1 do message\_[i]:=0;

Purge\_USB\_Device\_Out;//очищаем буфер

end;

end;

//Suspend;

end;

procedure device\_thread\_th.Update;

var i:integer;

begin

/// Обрабатываем посылку

/// Выводим результаты в лист

/// Выводим сообщения об ошибках

decoded\_message:=MainForm.device.message\_decode(message\_);

IzmForm.incoming\_message(decoded\_message,povtor+1,point);

for i:=0 to length(message\_)-1 do message\_[i]:=0;

end;

procedure device\_thread\_th.Update\_dev\_cond\_string\_getdata;

begin

//Выводим сообщение о получении данных

IzmForm.Izm\_condition\_label.Caption:=ConstUnit.izm\_get\_data;

IZmForm.IzmPB.Position:=0;

IZmForm.IzmTimer.Enabled:=true;

end;

procedure device\_thread\_th.Update\_dev\_cond\_string\_ready;

begin

//Вывод сообщения о готовности прибора к измерению

IzmForm.Izm\_condition\_label.Caption:=ConstUnit.izm\_device\_ready;

IZmForm.IzmTimer.Enabled:=false;

end;

procedure device\_thread\_th.Update\_dev\_cond\_string\_decode;

begin

//Вывод сообщения о декодировании посылки

IzmForm.Izm\_condition\_label.Caption:=ConstUnit.izm\_mes\_decode;

end;

procedure device\_thread\_th.Update\_message\_count;

begin

IzmForm.izm\_mes\_count.Caption:=inttostr(num);

if ((num mod 5)=0) then

begin

povtor:=0;

inc(point);

IzmForm.izm\_next\_point.Execute;

end;

IzmForm.izm\_mes\_tek\_count.Caption:=inttostr(povtor);

end;

procedure device\_thread\_th.Check\_and\_load\_data;

var list:TStringList;

//i,j,

k: Integer;

str,

y1,

y2:string;

tmp:array of extended;

begin

setLength(tmp,params\_amount+1);

//Проверяем значения на ошибки

Check\_errors;

//в массив результатов записываем декодированное сообщение

izm\_result[point,povtor]:=decoded\_message;

//создаем запись об измерении

with SQL\_DM.mysql\_query4 do begin

Connection:=SQL\_DM.mysql\_con;

SQL.Clear;

//Создаем запись об измерении.

SQL.Add('SELECT \* FROM `rd`.`rd\_izmerenia` WHERE');

SQL.Add(' `id`='+IzmForm.Izm\_gen\_num.Caption+';');

try

Active:=true;

except

//

end;

if SQL\_DM.mysql\_DSQ4.DataSet.IsEmpty

then

begin

SQL.Clear;

//Создаем запись об измерении.

SQL.Add('INSERT INTO `rd`.`rd\_izmerenia` (');

SQL.Add('`id` ,');

SQL.Add('`date` ,');

SQL.Add('`patient\_num` ,');

SQL.Add('`opisanie`');

SQL.Add(')');

SQL.Add('VALUES (');

SQL.Add(' '''+IzmForm.Izm\_gen\_num.Caption+''' , NOW() , '''

+IzmForm.Izm\_pat\_id.Caption+''', ');

SQL.Add(' '''+IzmForm.izm\_opisanie\_edit.Text+''');');

try

Execute;

except

//

end;

end;

end;

//проверяем не было ли уже записано в базу параметры текущей точки:

with SQL\_DM.mysql\_query4 do begin

Connection:=SQL\_DM.mysql\_con;

SQL.Clear;

SQL.Add('SELECT \* FROM `rd`.`'+table\_znacheniya\_name+'` WHERE (');

SQL.Add(' (`izm\_num`='+IzmForm.Izm\_gen\_num.Caption+') AND ('+

' `param\_num`='+inttostr(3)+') AND ('+

' `povtor\_num`='+inttostr(povtor)+') AND ('+

' `point\_num`='+inttostr(point)+'));');

//showmessage(SQL.Text);

try

Active:=true;

except

//

end;

end;

//берем информацию с прибора

tmp[1]:=decoded\_message.R\_e;

tmp[2]:=decoded\_message.G0\_e;

tmp[3]:=decoded\_message.G1\_e;

tmp[4]:=decoded\_message.G2\_e;

tmp[5]:=decoded\_message.G3\_e;

tmp[6]:=decoded\_message.C\_e;

tmp[7]:=decoded\_message.Usm\_e;

y1:=IzmForm.get\_string(decoded\_message.U\_izm);

y2:=IzmForm.get\_string(decoded\_message.I\_izm);

//showmessage (inttostr(SQL\_DM.mysql\_DSQ4.DataSet.RecordCount));

if not (SQL\_DM.mysql\_DSQ4.DataSet.RecordCount=0)

then //в случае если апдейт

begin

//создаем стринглист для загрузки в него запроса к базе

list:=TStringList.Create;

list.Clear;

for k:=1 to params\_amount do

begin

list.Add('UPDATE `rd`.`'+table\_znacheniya\_name+'` SET `znach`=');

DecimalSeparator:='.';

str:=floattostrf(tmp[k],ffExponent, 6, 2);

list.Add(''''+str+''''+' WHERE ((`param\_num`='+inttostr(k+2)+

') AND (`izm\_num`='+IzmForm.Izm\_gen\_num.Caption+

') AND (`point\_num`='+inttostr(point)+

') AND (`povtor\_num`='+inttostr(povtor)+')); ');

//list.Add(str);

end;

//showmessage(list.Text);

list.SaveToFile('znachzapros.txt');

end

else //в случае если надо записать

/////////////////////////////////

///

/// Здесь остановились чтобы сделать Usm

///

///

/// //////////////////////

begin

//создаем стринглист для загрузки в него запроса к базе

list:=TStringList.Create;

with list do

begin

Clear;

Add('INSERT INTO `rd`.`'+table\_znacheniya\_name+'` (');

Add('`id` ,');

Add('`izm\_num` ,');

Add('`param\_num` ,');

Add('`point\_num` ,');

Add('`povtor\_num` ,');

Add('`znach` ,');

Add('`znach\_UI`');

Add(')');

Add('VALUES');

end;

for k:=1 to params\_amount do

begin

DecimalSeparator:='.';

str:='(NULL , '''+

IzmForm.Izm\_gen\_num.Caption+''', '''+

inttostr(k+2)+''', '''+

inttostr(point)+''', '''+

inttostr(povtor)+''', ''';

str:=str+floattostrf(tmp[k],ffExponent, 6, 2)+''',' +'NULL';

if (k=params\_amount) and not(MainForm.izm\_write\_ui.Checked) then

str:=str+')'

else str:=str+'),';

list.Add(str);

end;

//если необходимо также записывать значения тока и напряжения

if MainForm.izm\_write\_ui.Checked then

begin

str:='(NULL , '''+

IzmForm.Izm\_gen\_num.Caption+''', '''+

inttostr(1)+''', '''+

inttostr(point)+''', '''+

inttostr(povtor)+''', '''+floattostr(0)+''',' +y1+'),';

list.Add(str);

str:='(NULL , '''+

IzmForm.Izm\_gen\_num.Caption+''', '''+

inttostr(2)+''', '''+

inttostr(point)+''', '''+

inttostr(povtor)+''', '''+floattostr(0)+''',' +y2+')';

if not(k=params\_amount) then

str:=str+',';

list.Add(str);

end;

list.SaveToFile('znachzapros.txt');

end;

//непосредственная запись в базу

with SQL\_DM.mysql\_query4 do begin

Connection:=SQL\_DM.mysql\_con;

SQL.Clear;

SQL.AddStrings(list);

try

Execute;

except

//

end;

end;

list.Destroy;

{//запись текущего измерения в виде sql запросов в текстовый файл

AssignFile(file\_record, 'last\_measute.dat'); Assigns the Filename

if (point=1) and (povtor=1) then ReWrite(file\_record)

else Append(file\_record);

Writeln(file\_record, izm\_result[i,j].R+#9+

izm\_result[point,povtor].G0+#9+

izm\_result[point,povtor].G1+#9+

izm\_result[point,povtor].G2+#9+

izm\_result[point,povtor].G3+#9+

izm\_result[point,povtor].C;);

Closefile(file\_record); //Closes file F}

//если пришли к концу всего измерения: 24 точки по 5 повторов

if (point=points\_amount) and (povtor=repeat\_amount) then

//выводим сообщение о законченности измерения

if MessageBox(0,

PChar(izm\_izm\_end),

PChar(izm\_caption\_message),

MB\_ICONQUESTION or MB\_OKCANCEL

)=id\_OK then

//если пользователь выбрал загрузку

begin

IzmForm.Izm\_stop\_btn.Caption:=izm\_unit\_end\_measurment\_btn\_label;

Suspend;

IzmForm.Close;

MainForm.GoToLookResults.Execute;

{

//проходим по каждой точке, каждому повтору по точке, и каждому параметру

for i:=1 to points\_amount do

begin

//запись полученных значений

for j:=1 to repeat\_amount do

begin

// во временный файл заносим результаты для данной точки и повтора

tmp[1]:=izm\_result[i,j].R;

tmp[2]:=izm\_result[i,j].G0;

tmp[3]:=izm\_result[i,j].G1;

tmp[4]:=izm\_result[i,j].G2;

tmp[5]:=izm\_result[i,j].G3;

tmp[6]:=izm\_result[i,j].C;

y1:=IzmForm.get\_string(izm\_result[i,j].U\_izm);

y2:=IzmForm.get\_string(izm\_result[i,j].I\_izm);

//по параметрам, и сразу же формируем строку для записи

for k:=1 to params\_amount do

begin

DecimalSeparator:='.';

str:='(NULL , '''+

IzmForm.Izm\_gen\_num.Caption+''', '''+

inttostr(k+2)+''', '''+

inttostr(i)+''', '''+

inttostr(j)+''', ''';

str:=str+floattostrf(tmp[k],ffExponent, 6, 2)+''',' +'NULL';

if (i=points\_amount) and (j=repeat\_amount) and (k=params\_amount) and

not(IzmForm.izm\_write\_ui.Checked) then

str:=str+')'

else str:=str+'),';

list.Add(str);

end;

//если необходимо также записывать значения тока и напряжения

if IzmForm.izm\_write\_ui.Checked then begin

str:='(NULL , '''+

IzmForm.Izm\_gen\_num.Caption+''', '''+

inttostr(1)+''', '''+

inttostr(i)+''', '''+

inttostr(j)+''', '''+floattostr(0)+''',' +y1+'),';

list.Add(str);

str:='(NULL , '''+

IzmForm.Izm\_gen\_num.Caption+''', '''+

inttostr(2)+''', '''+

inttostr(i)+''', '''+

inttostr(j)+''', '''+floattostr(0)+''',' +y2+')';

if not((i=points\_amount) and (j=repeat\_amount) and (k=6)) then

str:=str+',';

list.Add(str);

end;

end;

end;

list.SaveToFile('znachzapros.txt');

//формируем время измерения

GetSystemTime(s);

date:=inttostr(s.wYear)+'-'+inttostr(s.wMonth)+'-'+inttostr(s.wDay)+' '+

inttostr(s.wHour+4)+':'+inttostr(s.wMinute)+':'+inttostr(s.wSecond);

//непосредственная запись в базу

with SQL\_DM.mysql\_query4 do begin

Connection:=SQL\_DM.mysql\_con;

SQL.Clear;

//Создаем запись об измерении. Почему только здесь???

SQL.Add('INSERT INTO `rd`.`rd\_izmerenia` (');

SQL.Add('`id` ,');

SQL.Add('`date` ,');

SQL.Add('`patient\_num` ,');

SQL.Add('`opisanie`');

SQL.Add(')');

SQL.Add('VALUES (');

SQL.Add(' '''+IzmForm.Izm\_gen\_num.Caption+''' , '''+date+

''', '''+IzmForm.Izm\_pat\_id.Caption+''', ');

SQL.Add(' '''+IzmForm.izm\_opisanie\_edit.Text+''');');

Execute;

Active:=false;

SQL.Clear;

SQL.AddStrings(list);

try

Execute;

except

//

end;

end;

list.Destroy;

//запись в файл токов и напряжений

{list:=Tstringlist.Create;

str:='';

for i:=1 to 24 do

for j:=1 to 5 do

begin

///формируем следующим образом:

/// создаем табулированный файл для последующего экспорта в эксель

/// в первой строчке номер точки, номер повтора

/// потом столбец значений

str:=str+'точка: '+inttostr(i)+'повтор: '+inttostr(j)+'U'+#9+

'точка: '+inttostr(i)+'повтор: '+inttostr(j)+'I'+#9;

end;

list.Add(str);

str:='';

for i:=1 to 24 do

for j:=1 to 5 do

for k:=0 to 2400 do

begin

begin

str:=str+inttostr(izm\_result[i,j].U\_izm[k])+#9+inttostr(izm\_result[i,j].I\_izm[k])+#9;

list.Add(str);

end;

list.SaveToFile(IzmForm.izm\_opisanie\_edit.Text+'номер пациента: '+IzmForm.Izm\_pat\_id.Caption+'точка №:'+inttostr(i));

end;

end;

showmessage('Токи и напряжения записаны');

list.Destroy;}

end;

// IzmForm.load\_data\_to\_database(izm\_result);

end;

procedure device\_thread\_th.Check\_errors;

begin

/////////////////////////////////

//Обработка ошибок в измерении //

/////////////////////////////////

if decoded\_message.U\_mis

then begin

//если была ошибка, то обнуляем результаты

if MainForm.Izm\_null\_result\_CB.Checked then

begin

decoded\_message.R:=0;

decoded\_message.C:=0;

decoded\_message.g0:=0;

decoded\_message.g1:=0;

decoded\_message.g2:=0;

decoded\_message.g3:=0;

decoded\_message.Usm:=0;

decoded\_message.R\_e:=0;

decoded\_message.C\_e:=0;

decoded\_message.g0\_e:=0;

decoded\_message.g1\_e:=0;

decoded\_message.g2\_e:=0;

decoded\_message.g3\_e:=0;

decoded\_message.Usm\_e:=0;

end;

end;

if decoded\_message.I\_mis then

begin

//если была ошибка, то обнуляем результаты

if MainForm.Izm\_null\_result\_CB.Checked then

begin

decoded\_message.R:=0;

decoded\_message.C:=0;

decoded\_message.g0:=0;

decoded\_message.g1:=0;

decoded\_message.g2:=0;

decoded\_message.g3:=0;

decoded\_message.Usm:=0;

decoded\_message.R\_e:=0;

decoded\_message.C\_e:=0;

decoded\_message.g0\_e:=0;

decoded\_message.g1\_e:=0;

decoded\_message.g2\_e:=0;

decoded\_message.g3\_e:=0;

decoded\_message.Usm\_e:=0;

end;

end;

end;

end.

# МОДУЛЬ DeviceRD.pas

unit DeviceRD;

interface

uses D2xxUnit, SysUtils, dialogs, Windows, math, ExtCtrls, Controls;

//пользовательские типы

type bit\_array\_8 = array[0..7] of byte; //массив из 8 чисел. Используется для байта

bit\_Array\_32 = array[0..31] of byte; //массив из 32 чисел. Используется для 32-битного числа

message\_Array = array[0..24002] of integer; //массив посылки

TMatrix = array [0..4,0..4] of extended; //тип массива

TIzm\_array = array [0..2400] of extended;

//для матрицы 5 на 5

TMes\_array = array [0..4] of message\_Array;

Tbool = array of boolean;

TCol = array [0..4] of extended;

params\_record = record //запись для результатов разбора

//значения программы

R:extended;

C:extended;

g0:extended;

g1:extended;

g2:extended;

g3:extended;

Usm:extended;

Ism:extended;

U\_izm: TIzm\_array;

I\_izm: TIzm\_array;

//значения прибора

R\_e:extended;

C\_e:extended;

g0\_e:extended;

g1\_e:extended;

g2\_e:extended;

g3\_e:extended;

Usm\_e:extended;

Ism\_e:extended;

U\_mis: boolean;

I\_mis: boolean;

end;

//непосредственно класс устройства

type

TDevice = class (TObject)

private

connected :boolean; //существует ли связь с прибором.

port\_opened :boolean; //открыт ли порт прибора

device\_name :string; //Название прибора

device\_serial\_no:string; //Серийный номер прибора

const version = '0.4.4';

function copy\_col(Al:TMatrix; Bl:TCol; coll:integer):TMatrix;

function det5(A:TMatrix):extended;

public

message\_ : message\_Array;

timer :TTimer; //таймер на подключение прибора

procedure TimerTimer(Sender: TObject);

constructor Create();

//destructor Destroy();

function IsConnected:boolean;

function IsPortOpened:boolean;

procedure Close\_thread;

function Show\_device\_name:string;

function Show\_device\_serial\_no:string;

function message\_decode(massiv:message\_Array):params\_record;

procedure Start\_waiting(mode:integer);

// procedure receive;

// procedure Check\_device;

procedure Open\_device;

procedure Close\_usb;

protected

{ protected declarations }

end;

implementation

uses MainUnit, device\_thread, ConstUnit;

var TTH: device\_thread\_th;

procedure TDevice.Close\_thread;

begin

TTH.Suspend;

//TTH.Terminate;

//FreeAndNil(TTH);

end;

procedure TDevice.Close\_usb;

begin

connected:=false;

Port\_opened:=false;

Close\_USB\_device;

end;

constructor TDevice.Create;

var device\_index,i:integer;

begin

GetFTDeviceCount;

FT\_Enable\_Error\_Report := true;

connected:=false;

port\_opened:=false;

Device\_Index := 0;

//предполагается что устройств всегда будет только одно

If FT\_Device\_Count > 0 then

For i:=1 to FT\_Device\_Count do

Begin

GetFTDeviceSerialNo(Device\_Index);

device\_serial\_no:=FT\_Device\_String;

GetFTDeviceDescription ( Device\_Index );

Device\_Index := Device\_Index + 1;

connected:=true;

End;

//создаем таймер для проверки подключения прибора

timer:=TTimer.Create(MainForm);

timer.Interval:=1000;

Timer.OnTimer:=TimerTimer;

end;

procedure TDevice.TimerTimer(Sender: TObject);

begin

//получаем количество подключенных устройств,

//считаем если их больше 0, то прибор подключен

GetFTDeviceCount;

if FT\_Device\_Count>0 then

begin

if not connected then

begin

MainForm.new\_izm\_warning\_label.Caption:=devicerd\_device\_just\_connected;

//sleep(6000);

//Open\_device;

//MainForm.enabled:=true;

end;

end

else

if connected then

begin

Close\_usb;

end;

end;

{destructor TDevice.Destroy;

begin

If port\_opened then begin

Set\_USB\_Device\_BitMode($ff,0);

Purge\_USB\_Device\_In;

Purge\_USB\_Device\_Out;

Close\_USB\_Device;

end

else //Close\_USB\_Device;

end;

}

{Procedure Tdevice.Check\_device;

Var

PortStatus : FT\_Result;

device\_index,i :integer;

begin

FT\_Enable\_Error\_Report := true;

If Not Connected then

Begin

Close\_USB\_Device; // In case device was already open

If Open\_USB\_Device = FT\_OK then // Device is Now Present !

Begin

Connected := True;

//Reset\_USB\_Device; // warning - this will destroy any pending data.

//получаем информацию о приборе

device\_index:=0;

if FT\_Device\_Count > 0 then

For i:= 1 to FT\_Device\_Count do

Begin

GetFTDeviceSerialNo(Device\_Index );

device\_serial\_no:=FT\_Device\_String;

GetFTDeviceDescription ( Device\_Index );

device\_desc:=FT\_Device\_String;

Device\_Index := Device\_Index + 1;

End;

End;

End

else

Begin

PortStatus := Get\_USB\_Device\_QueueStatus;

If PortStatus <> FT\_OK then

begin // Device has been Unplugged

Connected := False;

end

else

//

End;

end;

}

Procedure TDevice.Open\_device;

begin

{if port\_opened then

begin

//Reset\_USB\_Device;

//Close\_USB\_Device; // In case device was already open

end; }

GetFTDeviceCount;

If Open\_USB\_Device\_By\_Serial\_Number(device\_serial\_no) = FT\_OK then

begin

Set\_USB\_Parameters(16384,16384); //размеры буферов

FT\_Current\_Baud:=FT\_BAUD\_921600;

if (not Set\_USB\_Device\_BaudRate = FT\_OK) then

MessageBox(0,

PChar('Ошибка при установки скорости соединения'),

PChar('Ошибка устройства'),

MB\_ICONERROR or MB\_OK

);

FT\_Current\_DataBits:=FT\_DATA\_BITS\_8;

FT\_Current\_StopBits:=FT\_STOP\_BITS\_1;

FT\_Current\_Parity:=FT\_PARITY\_NONE;

if (not Set\_USB\_Device\_DataCharacteristics = FT\_OK) then

MessageBox(0,

PChar('Ошибка при установке параметров(Бит,СтопБит,Паритет)'),

PChar('Ошибка устройства'),

MB\_ICONERROR or MB\_OK

);

if not Set\_USB\_Device\_TimeOuts(5,5) = FT\_OK then

MessageBox(0,

PChar('Ошибка при установке таймаутов'),

PChar('Ошибка устройства'),

MB\_ICONERROR or MB\_OK

);

if not Set\_USB\_Device\_BitMode($ff,0) = FT\_OK then

MessageBox(0,

PChar('Ошибка при выключении Bit Mode'),

PChar('Ошибка устройства'),

MB\_ICONERROR or MB\_OK

);

if (not Purge\_USB\_Device\_In = FT\_OK) and (Purge\_USB\_Device\_Out = FT\_OK) then

MessageBox(0,

PChar('Ошибка при очистке буферов I/O'),

PChar('Ошибка устройства'),

MB\_ICONERROR or MB\_OK

);

port\_opened:=true;

Reset\_USB\_Device; // warning - this will destroy any pending data.

Set\_USB\_Device\_TimeOuts(500,500); // read and write timeouts = 500mS

end

else MessageBox(0,

PChar('Ошибка при открытии устройства. Проверьте подключение'),

PChar('Ошибка устройства'),

MB\_ICONERROR or MB\_OK

);

end;

procedure TDevice.Start\_waiting(mode:integer);

begin

if mode=1 then

begin

TTH:=device\_thread\_th.Create(true);

// sleep(500);

// TTH.Resume;

end;

if mode=2 then

TTH.Resume;

end;

//разбор посылки

function TDevice.message\_decode(massiv: message\_Array):params\_record;

const N=2400; //количество замеров тока и напряжения

T2=0.00083717036416911; //время в секундах между отсчетами

Ki=33554431999999.996; //коэф. пересчета из целочисленных в реальные тока

Ku=3355443.2; //коэф. пересчета из целочисленных в реальные напряжения

var k, //текущее значение строчки в массиве измерения

i, //счетчик в цикле

drei:integer;

a1,b1,c1,d1,a2,b2,c2,d2: //временные переменные для составных частей посылки

extended;

U0, //

I0, //

Ud0, //

Up0,

Det0 :Extended;

A: TMatrix;

B: TCol;

j: Integer;

begin

///инициализация переменных

//setlength(a,5,5);

U0:=0;Ud0:=0; I0:=0;

k:=-9;

for I:= 0 to 4 do

begin

B[i]:=0;

for j:= 0 to 4 do

A[i,j]:=0;

end;

a1:=0;

b1:=0;

c1:=0;

d1:=0;

a2:=0;

b2:=0;

c2:=0;

d2:=0;

for I := 0 to length(result.I\_izm) - 1 do

begin

result.I\_izm[i]:=0;

result.U\_izm[i]:=0;

end;

//ошибки

result.U\_mis:=false;

result.I\_mis:=false;;

for i:=0 to N do

begin

k:=k+10;

//разбор блока значений напряжения

//вся посылка 5 байт

a1:=(((((massiv[k] shl 4) and $80) or massiv[k+1]) shl 24) and $FF000000);

b1:=(((((massiv[k] shl 5) and $80) or massiv[k+2]) shl 16) and $FFFF0000);

c1:=(((((massiv[k] shl 6) and $80) or massiv[k+3]) shl 8) and $FFFFFF00);

d1:=(((((massiv[k] shl 7) and $80) or massiv[k+4]) shl 0) and $FFFFFFFF);

result.U\_izm[i]:=a1+b1+c1+d1;

a2:=(((((massiv[k+5] shl 4) and $80) or massiv[k+6]) shl 24) and $FF000000);

b2:=(((((massiv[k+5] shl 5) and $80) or massiv[k+7]) shl 16) and $FFFF0000);

c2:=(((((massiv[k+5] shl 6) and $80) or massiv[k+8]) shl 8) and $FFFFFF00);

d2:=(((((massiv[k+5] shl 7) and $80) or massiv[k+9]) shl 0) and $FFFFFFFF);

result.I\_izm[i]:=a2+b2+c2+d2;

//начало нарастания 977 отсчет

{ ///Подсчёт значений R, g0, g1, g2, g3, C силами программы

if (i=1000) then Ud0:=a1+b1+c1+d1;

if (i=1001) then

begin

U0:=a1+b1+c1+d1;

I0:=a2+b2+c2+d2;

end;

if ((i>=1002)and(i<=2201)) then

begin

Up0:=a1+b1+c1+d1;

A[0,0]:=A[0,0]+intpower(U0,2);

A[0,1]:=A[0,1]+intpower(U0,3);

A[0,2]:=A[0,2]+intpower(U0,4);

A[0,3]:=A[0,3]+intpower(U0,5);

A[1,3]:=A[1,3]+intpower(U0,6);

A[2,3]:=A[2,3]+intpower(U0,7);

A[3,3]:=A[3,3]+intpower(U0,8);

A[0,4]:=A[0,4]+U0\*(Up0-Ud0);

A[1,4]:=A[1,4]+intpower(U0,2)\*(Up0-Ud0);

A[2,4]:=A[2,4]+intpower(U0,3)\*(Up0-Ud0);

A[3,4]:=A[3,4]+intpower(U0,4)\*(Up0-Ud0);

A[4,4]:=A[4,4]+intpower((Up0-Ud0),2);

//приводим к диагональному виду матрицу А

A[1,0]:=A[0,1]; A[2,0]:=A[0,2]; A[3,0]:=A[0,3]; A[3,1]:=A[1,3];

A[4,0]:=A[0,4]; A[3,2]:=A[2,3]; A[4,1]:=A[1,4]; A[4,2]:=A[2,4];

A[4,3]:=A[3,4];

//НЕ ПОНЯТНО!!!!!!!!!!!!!!!

A[1,1]:=A[0,2]; A[1,2]:=A[0,3]; A[2,1]:=A[0,3]; A[2,2]:=A[1,3];

//формируем столбец В

B[0]:=B[0]+I0\*U0;

B[1]:=B[1]+I0\*intpower(U0,2);

B[2]:=B[2]+I0\*intpower(U0,3);

B[3]:=B[3]+I0\*intpower(U0,4);

B[4]:=B[4]+I0\*(Up0-Ud0);

I0:=a2+b2+c2+d2;

Ud0:=U0;

U0:=Up0;

end;

if i=2390 then

begin

Det0:=det5(A);

result.g0:=det5(copy\_col(A,B,0))/Det0\*Ku/Ki;

result.g1:=det5(copy\_col(A,B,1))/Det0\*Ku/Ki\*Ku;

result.g2:=det5(copy\_col(A,B,2))/Det0\*Ku/Ki\*Ku\*Ku;

result.g3:=det5(copy\_col(A,B,3))/Det0\*Ku/Ki\*Ku\*Ku\*Ku;

result.C:=det5(copy\_col(A,B,4))/Det0\*Ku/Ki\*T2;

result.R:= 1/result.g0;

//result.Usm:=det5(copy\_col(A,B,3))/Det0\*Ku/Ki\*Ku\*Ku\*Ku;

}

result.g0:=0;

result.g1:=0;

result.g2:=0;

result.g3:=0;

result.C:=0;

result.R:= 0;

result.Usm:=0;

//end;

//2392 - байт ошибки

//канал U-выход за диапазон измерения

//канал I - отпустили электрод

if i=2392 then

begin

if a1+b1+c1+d1<>0 then result.U\_mis:=true;

if a2+b2+c2+d2<>0 then result.I\_mis:=true;

end;

if i>=2393 then

begin

drei:=round(result.I\_izm[i]-round(result.I\_izm[i]/intpower(2,24))\*intpower(2,24)-1023);

if i=2393 then

begin

if round(result.I\_izm[i]/intpower(2,24)) mod 2=0 then

result.R\_e:=result.U\_izm[i]/intpower(2,30)\*intpower(2,drei)

else result.R\_e:=-1\*result.U\_izm[i]/intpower(2,30)\*intpower(2,drei)

end;

if i=2394 then

begin

if round(result.I\_izm[i]/intpower(2,24)) mod 2=0 then

result.g0\_e:=result.U\_izm[i]/intpower(2,30)\*intpower(2,drei)

else result.g0\_e:=-1\*result.U\_izm[i]/intpower(2,30)\*intpower(2,drei)

end;

if i=2395 then

begin

if round(result.I\_izm[i]/intpower(2,24)) mod 2=0 then

result.g1\_e:=result.U\_izm[i]/intpower(2,30)\*intpower(2,drei)

else result.g1\_e:=-1\*result.U\_izm[i]/intpower(2,30)\*intpower(2,drei)

end;

if i=2396 then

begin

if round(result.I\_izm[i]/intpower(2,24)) mod 2=0 then

result.g2\_e:=result.U\_izm[i]/intpower(2,30)\*intpower(2,drei)

else result.g2\_e:=-1\*result.U\_izm[i]/intpower(2,30)\*intpower(2,drei)

end;

if i=2397 then

begin

if round(result.I\_izm[i]/intpower(2,24)) mod 2=0 then

result.g3\_e:=result.U\_izm[i]/intpower(2,30)\*intpower(2,drei)

else result.g3\_e:=-1\*result.U\_izm[i]/intpower(2,30)\*intpower(2,drei)

end;

if i=2398 then

begin

if round(result.I\_izm[i]/intpower(2,24)) mod 2=0 then

result.C\_e:=result.U\_izm[i]/intpower(2,30)\*intpower(2,drei)

else result.C\_e:=-1\*result.U\_izm[i]/intpower(2,30)\*intpower(2,drei)

end;

if i=2399 then

begin

if round(result.I\_izm[i]/intpower(2,24)) mod 2=0 then

result.Usm\_e:=result.U\_izm[i]/Ku

else result.Usm\_e:=-1\*result.U\_izm[i]/Ku;

end;

end;

end;

end;

//функции вызова свойств класса

function TDevice.IsConnected;

begin

if connected then result:=true else result:=false;

end;

function TDevice.IsPortOpened;

begin

if port\_opened then result:=true else result:=false;

end;

function TDevice.Show\_device\_name;

begin result:=device\_name; end;

function TDevice.Show\_device\_serial\_no;

begin result:=device\_serial\_no; end;

//копирование столбца В в матрицу А в колонку col

function TDevice.copy\_col(Al:TMatrix; Bl:TCol; coll:integer):TMatrix;

var i:integer;

begin

for I := 0 to 4 do

Al[i,coll]:=Bl[i];

result:=Al;

end;

//определитель матрицы 5х5

function TDevice.det5(A:TMatrix):extended;

function det4(A11,A12,A13,A14,A21,A22,A23,A24,A31,A32,A33,A34,A41,A42,A43,A44:extended):extended;

function det3(A11,A12,A13,A21,A22,A23,A31,A32,A33:extended):extended;

function det2(A11,A12,A21,A22:extended):extended;

begin

result:=A11\*A22-A12\*A21;

end;

begin

result:=A11\*det2(A22,A23,A32,A33)-A12\*det2(A21,A23,A31,A33)+A13\*det2(A21,A22,A31,A32);

end;

begin

result:=A11\*det3(A22,A23,A24,A32,A33,A34,A42,A43,A44)-

A12\*det3(A21,A23,A24,A31,A33,A34,A41,A43,A44)+

A13\*det3(A21,A22,A24,A31,A32,A34,A41,A42,A44)-

A14\*det3(A21,A22,A23,A31,A32,A33,A41,A42,A43);

end;

begin

result:=A[0,0]\*det4(A[1,1],A[1,2],A[1,3],A[1,4],A[2,1],A[2,2],A[2,3],A[2,4],A[3,1],A[3,2],A[3,3],A[3,4],A[4,1],A[4,2],A[4,3],A[4,4])-

A[0,1]\*det4(A[1,0],A[1,2],A[1,3],A[1,4],A[2,0],A[2,2],A[2,3],A[2,4],A[3,0],A[3,2],A[3,3],A[3,4],A[4,0],A[4,2],A[4,3],A[4,4])+

A[0,2]\*det4(A[1,0],A[1,1],A[1,3],A[1,4],A[2,0],A[2,1],A[2,3],A[2,4],A[3,0],A[3,1],A[3,3],A[3,4],A[4,0],A[4,1],A[4,3],A[4,4])-

A[0,3]\*det4(A[1,0],A[1,1],A[1,2],A[1,4],A[2,0],A[2,1],A[2,2],A[2,4],A[3,0],A[3,1],A[3,2],A[3,4],A[4,0],A[4,1],A[4,2],A[4,4])+

A[0,4]\*det4(A[1,0],A[1,1],A[1,2],A[1,3],A[2,0],A[2,1],A[2,2],A[2,3],A[3,0],A[3,1],A[3,2],A[3,3],A[4,0],A[4,1],A[4,2],A[4,3]);

end;

end.

# МОДУЛЬ IzmGraphUnit.pas

unit IzmGraphUnit;

interface

uses

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

Dialogs, AdvToolBtn, ExtCtrls, StdCtrls;

type

TIzmGraphFrame = class(TFrame)

AdvToolButton1: TAdvToolButton;

Label14: TLabel;

Panel1: TPanel;

private

{ Private declarations }

public

{ Public declarations }

end;

implementation

{$R \*.dfm}

end.

# МОДУЛЬ IzmUnit.pas

unit IzmUnit;

interface

uses

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

Dialogs, StdCtrls, PictureList, ExtCtrls, AdvPanel, AdvPicture, AdvGlowButton,

AdvProgressBar, ActnList, DeviceRD, ComCtrls, DateUtils, DB, LayeredForm,

IzmGraphUnit;

type TIzmResult = array of array of DeviceRD.params\_record;

// type TIzmResult = array [1..24,1..5] of DeviceRD.params\_record;

type

TIzmForm = class(TForm)

Izm\_main\_panel: TAdvPanel;

Izm\_top\_panel: TPanel;

Izm\_middle\_panel: TPanel;

Izm\_prev\_point\_btn: TAdvGlowButton;

Izm\_next\_point\_btn: TAdvGlowButton;

Izm\_condition\_label: TLabel;

Izm\_stop\_btn: TAdvGlowButton;

Panel2: TPanel;

Panel3: TPanel;

Panel4: TPanel;

Izm\_info\_panel: TPanel;

Panel1: TPanel;

Izm\_progress\_bar: TAdvProgressBar;

Izm\_point\_picture: TAdvPicture;

ActionList: TActionList;

izm\_next\_point: TAction;

izm\_prev\_point: TAction;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

Label5: TLabel;

Label6: TLabel;

Izm\_patient\_name: TLabel;

Izm\_date: TLabel;

Izm\_pat\_num: TLabel;

Izm\_gen\_num: TLabel;

Izm\_pat\_id: TLabel;

Label7: TLabel;

izm\_mes\_tek\_count: TLabel;

Label9: TLabel;

izm\_mes\_count: TLabel;

izm\_device\_result\_LB: TListView;

izm\_device\_program\_LB: TListView;

Label8: TLabel;

Label10: TLabel;

izm\_U\_mis: TLabel;

izm\_I\_mis: TLabel;

Panel5: TPanel;

Izm\_DS: TDataSource;

Label12: TLabel;

izm\_point\_opis: TLabel;

izm\_point\_num: TLabel;

Label1: TLabel;

Label13: TLabel;

izm\_point\_name: TLabel;

IzmPB: TAdvProgressBar;

IzmTimer: TTimer;

Label11: TLabel;

izm\_opisanie\_edit: TEdit;

IzmGraphFrame1: TIzmGraphFrame;

procedure FormCreate(Sender: TObject);

procedure izm\_next\_pointExecute(Sender: TObject);

procedure izm\_prev\_pointExecute(Sender: TObject);

procedure FormShow(Sender: TObject);

procedure izm\_manual\_switchClick(Sender: TObject);

procedure Izm\_stop\_btnClick(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure FormKeyPress(Sender: TObject; var Key: Char);

procedure IzmTimerTimer(Sender: TObject);

private

{ Private declarations }

public

points\_name\_array: array [1..24] of string;

points\_array: array [1..24] of string;

procedure incoming\_message(message\_:params\_record; point,povtor:integer);

procedure load\_data\_to\_database(izm\_result:TIzmResult);

function get\_string(data:TIzm\_array):string;

procedure load\_picture(num:integer);

end;

var

IzmForm: TIzmForm;

pict\_num,

povrot\_num:integer;

implementation

uses ConstUnit, MainUnit, DataUnit, device\_thread, D2XXUnit;

{$R \*.dfm}

procedure TIzmForm.load\_picture(num: Integer);

var stream:TMemoryStream;

i:integer;

begin

SQL\_DM.mysql\_DSQ3.DataSet.Open;

SQL\_DM.mysql\_DSQ3.DataSet.First;

//Перемотка датасета

if num>1 then for i:=2 to num do SQL\_DM.mysql\_DSQ3.DataSet.Next;

try

Izm\_point\_picture.Picture:=nil;

stream := TMemoryStream.Create;

(SQL\_DM.mysql\_DSQ3.DataSet.fieldbyname('image') as TBlobField).SaveToStream(stream);

{чтобы что-нибудь записать, необходимо установить позицию потока в ноль!}

Stream.Position:= 0;

Izm\_point\_picture.Picture.LoadFromStream(stream);

stream.Free;

except

end;

if Izm\_point\_picture.Picture<>nil then Izm\_point\_picture.Show;

SQL\_DM.mysql\_DSQ3.DataSet.close;

end;

///для создания строки из значений тока и напряжения

///в настоящий момент не используется

function TIzmForm.get\_string(data: TIzm\_array):string;

var i:integer;

tmp:string;

begin

result:='';

for i:=0 to 2400 do

begin

tmp:=floattostr(data[i]);

if strlen(PChar(tmp))<>3 then

if strlen(PChar(tmp))=1 then tmp:='00'+tmp else

if strlen(PChar(tmp))=2 then tmp:='0'+tmp;

result:=result+tmp;

end;

end;

procedure TIzmForm.incoming\_message(message\_:params\_record; point,povtor:integer);

begin

// Показываем результаты расчета

//добавить результаты с программы

izm\_device\_result\_LB.Items[0].SubItems.Clear;

izm\_device\_result\_LB.Items[0].SubItems.Add(floattostrf(message\_.R, ffExponent, 4, 2));

izm\_device\_result\_LB.Items[1].SubItems.Clear;

izm\_device\_result\_LB.Items[1].SubItems.Add(floattostrf(message\_.g0, ffExponent, 4, 2));

izm\_device\_result\_LB.Items[2].SubItems.Clear;

izm\_device\_result\_LB.Items[2].SubItems.Add(floattostrf(message\_.g1, ffExponent, 4, 2));

izm\_device\_result\_LB.Items[3].SubItems.Clear;

izm\_device\_result\_LB.Items[3].SubItems.Add(floattostrf(message\_.g2, ffExponent, 4, 2));

izm\_device\_result\_LB.Items[4].SubItems.Clear;

izm\_device\_result\_LB.Items[4].SubItems.Add(floattostrf(message\_.g3, ffExponent, 4, 2));

izm\_device\_result\_LB.Items[5].SubItems.Clear;

izm\_device\_result\_LB.Items[5].SubItems.Add(floattostrf(message\_.C, ffExponent, 4, 2));

izm\_device\_result\_LB.Items[6].SubItems.Clear;

izm\_device\_result\_LB.Items[6].SubItems.Add(floattostrf(message\_.Usm, ffExponent, 4, 2));

//добавить результаты с прибора

izm\_device\_program\_LB.Items[0].SubItems.Clear;

izm\_device\_program\_LB.Items[0].SubItems.Add(floattostrf(message\_.R\_e, ffExponent, 4, 2));

izm\_device\_program\_LB.Items[1].SubItems.Clear;

izm\_device\_program\_LB.Items[1].SubItems.Add(floattostrf(message\_.g0\_e, ffExponent, 4, 2));

izm\_device\_program\_LB.Items[2].SubItems.Clear;

izm\_device\_program\_LB.Items[2].SubItems.Add(floattostrf(message\_.g1\_e, ffExponent, 4, 2));

izm\_device\_program\_LB.Items[3].SubItems.Clear;

izm\_device\_program\_LB.Items[3].SubItems.Add(floattostrf(message\_.g2\_e, ffExponent, 4, 2));

izm\_device\_program\_LB.Items[4].SubItems.Clear;

izm\_device\_program\_LB.Items[4].SubItems.Add(floattostrf(message\_.g3\_e, ffExponent, 4, 2));

izm\_device\_program\_LB.Items[5].SubItems.Clear;

izm\_device\_program\_LB.Items[5].SubItems.Add(floattostrf(message\_.C\_e, ffExponent, 4, 2));

izm\_device\_program\_LB.Items[6].SubItems.Clear;

izm\_device\_program\_LB.Items[6].SubItems.Add(floattostrf(message\_.Usm\_e, ffExponent, 4, 2));

//Показываем результаты ошибок

if message\_.U\_mis

then begin

izm\_U\_mis.Visible:=true;

//если была ошибка, то обнуляем результаты

if MainForm.Izm\_null\_result\_CB.Checked then

begin

message\_.R:=0;

message\_.C:=0;

message\_.g0:=0;

message\_.g1:=0;

message\_.g2:=0;

message\_.g3:=0;

message\_.R\_e:=0;

message\_.C\_e:=0;

message\_.g0\_e:=0;

message\_.g1\_e:=0;

message\_.g2\_e:=0;

message\_.g3\_e:=0;

message\_.Usm\_e:=0;

message\_.Ism\_e:=0;

message\_.Usm:=0;

message\_.Ism:=0;

end;

end

else izm\_U\_mis.Visible:=false;

if message\_.I\_mis then

begin

izm\_I\_mis.Visible:=true;

//если была ошибка, то обнуляем результаты

if MainForm.Izm\_null\_result\_CB.Checked then

begin

message\_.R:=0;

message\_.C:=0;

message\_.g0:=0;

message\_.g1:=0;

message\_.g2:=0;

message\_.g3:=0;

message\_.R\_e:=0;

message\_.C\_e:=0;

message\_.g0\_e:=0;

message\_.g1\_e:=0;

message\_.g2\_e:=0;

message\_.g3\_e:=0;

message\_.Usm\_e:=0;

message\_.Ism\_e:=0;

message\_.Usm:=0;

message\_.Ism:=0;

end;

end

else izm\_I\_mis.Visible:=false;

end;

procedure TIzmForm.IzmTimerTimer(Sender: TObject);

begin

IzmPB.Position:=IzmPB.Position+3;

end;

procedure TIzmForm.load\_data\_to\_database(izm\_result: TIzmResult);

begin

//

end;

procedure TIzmForm.FormClose(Sender: TObject; var Action: TCloseAction);

begin

//уничтожили поток

MainForm.device.Close\_thread;

//закрыли устройство

//MainForm.device.Close\_usb;

end;

procedure TIzmForm.izm\_manual\_switchClick(Sender: TObject);

begin

if MainForm.izm\_manual\_switch.Checked then begin

if pict\_num<>1 then

Izm\_prev\_point\_btn.Enabled:=true;

if pict\_num<>24 then

Izm\_next\_point\_btn.Enabled:=true;

end

else

begin

Izm\_prev\_point\_btn.Enabled:=false;

Izm\_next\_point\_btn.Enabled:=false;

end;

end;

procedure TIzmForm.FormCreate(Sender: TObject);

var i:integer;

begin

//загрузка точек...

mysql\_query(SQL\_DM.mysql\_query3, 'SELECT num, name, opisanie, image FROM '+table\_points\_name+';', 1);

Izm\_DS:=SQL\_DM.mysql\_DSQ3;

load\_picture(1);

Izm\_DS.DataSet.Open;

Izm\_DS.DataSet.First;

for I:=1 to 24 do

begin

points\_name\_array[i]:=Izm\_DS.DataSet.FieldByName('name').AsString;

points\_array[i]:=Izm\_DS.DataSet.FieldByName('opisanie').AsString;

Izm\_DS.DataSet.next;

end;

with izm\_device\_result\_LB do begin

Items.Add();

Items[0].Caption:=params\_default\_name3;

Items.Add();

Items[1].Caption:=params\_default\_name4;

Items.Add();

Items[2].Caption:=params\_default\_name5;

Items.Add();

Items[3].Caption:=params\_default\_name6;

Items.Add();

Items[4].Caption:=params\_default\_name7;

Items.Add();

Items[5].Caption:=params\_default\_name8;

Items.Add();

Items[6].Caption:=params\_default\_name9;

end;

with izm\_device\_program\_LB do begin

Items.Add();

Items[0].Caption:=params\_default\_name3;

Items.Add();

Items[1].Caption:=params\_default\_name4;

Items.Add();

Items[2].Caption:=params\_default\_name5;

Items.Add();

Items[3].Caption:=params\_default\_name6;

Items.Add();

Items[4].Caption:=params\_default\_name7;

Items.Add();

Items[5].Caption:=params\_default\_name8;

Items.Add();

Items[6].Caption:=params\_default\_name9;

end;

Izm\_DS.DataSet.Close;

end;

procedure TIzmForm.FormKeyPress(Sender: TObject; var Key: Char);

begin

if key=#13 then izmForm.izm\_next\_point.Execute;

end;

procedure TIzmForm.FormShow(Sender: TObject);

begin

//device\_thread.TTH.Resume;

//Панель информации

Izm\_patient\_name.Caption:=MainForm.new\_izm\_fam\_znach.Caption+' '+

MainForm.new\_izm\_name\_znach.Caption+' '+

MainForm.new\_izm\_otch\_znach.Caption;

Izm\_date.Caption:=datetostr(date());

Izm\_pat\_num.Caption:=inttostr(strtoint(MainForm.new\_izm\_kol\_znach.Caption)+1);

mysql\_query(SQL\_DM.mysql\_query1, 'SELECT MAX(id) FROM `'+table\_izm\_name+'`;',1);

Izm\_gen\_num.Caption:=inttostr(SQL\_DM.mysql\_DSQ1.DataSet.FieldByName('MAX(id)').AsInteger+1);

// обнуление параметров формы

Izm\_pat\_id.Caption:=MainForm.new\_izm\_pat\_id.Caption;

izm\_mes\_count.Caption:=inttostr(device\_thread.num);

izm\_mes\_tek\_count.Caption:='0';

izm\_mes\_count.Caption:='0';

device\_thread.povtor:=0;

device\_thread.point:=1;

device\_thread.num:=0;

pict\_num:=1;

izm\_point\_num.Caption:=inttostr(pict\_num);

load\_picture(1);

Izm\_progress\_bar.Position:=round(100/24\*(pict\_num-1));

Izm\_stop\_btn.Caption:=izm\_stop\_measurement;

izm\_opisanie\_edit.Text:=izm\_unit\_measurment\_caption\_label

+datetostr(today())+' '+timetostr(time());

Izm\_next\_point\_btn.Enabled:=true;

Izm\_prev\_point\_btn.Enabled:=false;

izm\_point\_opis.Caption:=points\_array[1];

izm\_point\_name.Caption:=points\_name\_array[1];

IzmPB.Position:=0;

IzmTimer.Enabled:=false;

//Состояние прибора

Purge\_USB\_Device\_In;

Purge\_USB\_Device\_Out;

Reset\_USB\_Device;

/// заносим нулевые значения в базу, потом только обновление

end;

procedure TIzmForm.izm\_next\_pointExecute(Sender: TObject);

begin

if pict\_num+1<25 then

begin

load\_picture(pict\_num+1);

Izm\_progress\_bar.Position:=round(100/24\*(pict\_num+1));

Izm\_prev\_point\_btn.Enabled:=true;

izm\_point\_num.Caption:=inttostr(pict\_num+1);

izm\_point\_opis.Caption:=points\_array[pict\_num+1];

izm\_point\_name.Caption:=points\_name\_array[pict\_num+1];

inc(pict\_num);

end;

if pict\_num+1=25 then begin

Izm\_progress\_bar.Position:=100;

Izm\_next\_point\_btn.Enabled:=false;

end;

izm\_mes\_tek\_count.Caption:='0';

device\_thread.povtor:=0;

device\_thread.point:=pict\_num;

end;

procedure TIzmForm.izm\_prev\_pointExecute(Sender: TObject);

begin

if pict\_num-1>0 then

begin

load\_picture(pict\_num-1);

Izm\_progress\_bar.Position:=round(100/24\*(pict\_num-1));

Izm\_next\_point\_btn.Enabled:=true;

izm\_point\_num.Caption:=inttostr(pict\_num-1);

izm\_point\_opis.Caption:=points\_array[pict\_num-1];

izm\_point\_name.Caption:=points\_name\_array[pict\_num-1];

dec(pict\_num);

end;

if pict\_num-1=0 then begin

Izm\_progress\_bar.Position:=round(100/24\*(pict\_num));

Izm\_prev\_point\_btn.Enabled:=false;

end;

izm\_mes\_tek\_count.Caption:='0';

device\_thread.povtor:=0;

device\_thread.point:=pict\_num;

end;

procedure TIzmForm.Izm\_stop\_btnClick(Sender: TObject);

begin

if MessageBox(0,

PChar(izm\_data\_lost),

PChar(izm\_caption\_message),

MB\_ICONQUESTION or MB\_OKCANCEL

)=id\_OK then

begin

self.Close;

end;

end;

end.

# МОДУЛЬ MainUnit.pas

unit MainUnit;

interface

uses

Messages, SysUtils, Variants, Classes, Graphics, Windows, dateUtils,

Controls, Forms, Dialogs, DeviceRd,

AdvOfficeStatusBar, Menus, AdvMenus, XPMan, ExtCtrls, AdvPanel,

ComCtrls, AdvPageControl, AdvGlowButton, AdvPicture, ActnList, StdCtrls,

DBAdvGrid, DBCtrls, AdvEdit, Spin, StdActns, DB, AdvMemo,

AdvDateTimePicker, DBAdvMemo, DBAdvLst, TeeProcs, TeEngine, Chart, DBChart,

AdvGroupBox, Grids, BaseGrid, AdvGrid, AdvListV, ExtActns, ToolPanels,

MyAccess, AdvOfficeButtons, Series, AdvEdBtn, Mask, AdvSpin,

AdvCombo, Lucombo, OleServer, ExcelXP, ComObj;

type

TMainForm = class(TForm)

Main\_Menu: TAdvMainMenu;

StatusBar: TAdvOfficeStatusBar;

N1: TMenuItem;

N2: TMenuItem;

N3: TMenuItem;

N4: TMenuItem;

N5: TMenuItem;

N6: TMenuItem;

N8: TMenuItem;

N9: TMenuItem;

N10: TMenuItem;

XPMan: TXPManifest;

N11: TMenuItem;

ActionList: TActionList;

GoToNewMeasure: TAction;

GoToLookResults: TAction;

ImageFileOpen: TOpenPicture;

Show\_all\_pat: TAction;

new\_patient: TAction;

del\_patient: TAction;

red\_patient: TAction;

look\_res\_load\_data: TAction;

look\_res\_show\_15: TAction;

look\_res\_show\_max: TAction;

look\_res\_show\_min: TAction;

look\_res\_show\_mean: TAction;

look\_res\_hide\_15: TAction;

look\_res\_hide\_max: TAction;

look\_res\_hide\_min: TAction;

look\_res\_hide\_mean: TAction;

Begin\_izm: TAction;

MainPanel: TPanel;

look\_res\_panel: TPanel;

look\_results\_DB\_Grid: TDBAdvGrid;

look\_results\_pc: TAdvPageControl;

look\_results\_choose\_info\_page: TAdvTabSheet;

look\_res\_enter\_panel: TPanel;

look\_res\_choise\_izm\_label: TLabel;

look\_res\_choise\_opis\_label: TLabel;

look\_res\_choise\_izm: TListView;

look\_res\_info\_panel: TPanel;

look\_res\_choise\_otch\_znach: TLabel;

look\_res\_choise\_info\_capt\_label: TLabel;

look\_res\_choise\_fam: TLabel;

look\_res\_choise\_name: TLabel;

look\_res\_choise\_otch: TLabel;

look\_res\_choise\_date: TLabel;

look\_res\_choise\_prim: TLabel;

look\_res\_choise\_fam\_znach: TLabel;

look\_res\_choise\_name\_znach: TLabel;

look\_res\_choise\_izm\_num: TLabel;

look\_res\_choise\_izm\_kol: TLabel;

look\_res\_choise\_date\_znach: TLabel;

look\_results\_number\_page: TAdvTabSheet;

look\_results\_graphics\_page: TAdvTabSheet;

look\_res\_graph\_params\_GB: TAdvGroupBox;

look\_res\_graph\_points\_GB: TAdvGroupBox;

look\_res\_graph\_points\_forw: TButton;

look\_res\_graph\_points\_back: TButton;

look\_results\_diagnoz\_page: TAdvTabSheet;

New\_izm\_Panel: TPanel;

New\_izm\_DB\_Grid: TDBAdvGrid;

New\_izm\_Add\_pat: TAdvGlowButton;

New\_izm\_red\_pat: TAdvGlowButton;

New\_izm\_del\_pat: TAdvGlowButton;

New\_izm\_pc: TAdvPageControl;

izm\_tab: TAdvTabSheet;

New\_izm\_info\_panel: TPanel;

new\_izm\_pat\_title\_label: TLabel;

new\_izm\_fam\_label: TLabel;

new\_izm\_name\_label: TLabel;

new\_izm\_otch\_label: TLabel;

new\_izm\_date\_label: TLabel;

new\_izm\_prim\_label: TLabel;

new\_izm\_fam\_znach: TLabel;

new\_izm\_name\_znach: TLabel;

new\_izm\_otch\_znach: TLabel;

new\_izm\_kol\_label: TLabel;

new\_izm\_kol\_znach: TLabel;

new\_izm\_date\_znach: TLabel;

new\_izm\_pat\_id: TLabel;

new\_izm\_warning\_label: TLabel;

new\_izm\_look\_red\_btn: TAdvGlowButton;

new\_izm\_DBMemo\_prim: TDBAdvMemo;

new\_izm\_begin\_izm\_btn: TAdvGlowButton;

New\_izm\_red\_panel: TPanel;

new\_izm\_prim\_edit\_label: TLabel;

new\_izm\_fam\_edit: TAdvEdit;

new\_izm\_name\_edit: TAdvEdit;

new\_izm\_otch\_edit: TAdvEdit;

new\_izm\_save\_btn: TAdvGlowButton;

new\_izm\_reset\_btn: TAdvGlowButton;

izm\_set\_tab: TAdvTabSheet;

top\_panel: TAdvPanel;

new\_measure\_btn: TAdvGlowButton;

look\_results\_btn: TAdvGlowButton;

new\_izm\_edit\_memo: TMemo;

Label1: TLabel;

look\_res\_izm\_description: TLabel;

look\_res\_point\_CB: TComboBox;

look\_res\_graph\_points\_label: TLabel;

look\_res\_point\_num\_label: TLabel;

N12: TMenuItem;

N13: TMenuItem;

N14: TMenuItem;

look\_res\_graph\_options\_GB: TAdvGroupBox;

look\_res\_graph\_options\_15\_CB: TCheckBox;

look\_res\_graph\_options\_15\_color: TShape;

look\_res\_graph\_options\_max\_color: TShape;

look\_res\_graph\_options\_min\_color: TShape;

look\_res\_graph\_options\_average\_color: TShape;

look\_res\_diagnoz\_info\_panel: TAdvPanel;

look\_res\_diagnoz\_PC: TAdvPageControl;

look\_res\_diagnoz\_data\_page: TAdvTabSheet;

look\_res\_diagnoz\_koeff\_page: TAdvTabSheet;

look\_res\_diagnoz\_help\_page: TAdvTabSheet;

look\_res\_fio\_capt\_label: TLabel;

look\_res\_iznmname\_capt\_label: TLabel;

look\_res\_izm\_capt\_label: TLabel;

look\_res\_fio\_label: TLabel;

look\_res\_izm\_name\_label: TLabel;

look\_res\_izm\_about\_label: TLabel;

look\_res\_diagnoz\_data\_top\_panel: TPanel;

look\_res\_diagnoz\_DBPoints\_panel: TPanel;

look\_res\_diagnoz\_data\_panel: TPanel;

look\_res\_diagnoz\_data\_bottom\_panel: TPanel;

look\_res\_diagnoz\_data\_name\_label: TLabel;

look\_res\_diagnoz\_data\_znach\_label: TLabel;

look\_res\_diagnoz\_data\_unit\_label: TLabel;

look\_res\_diagnoz\_help\_StringGrid: TAdvStringGrid;

ColorDialog: TColorDialog;

look\_res\_diagnoz\_koeff\_pat\_shape2: TShape;

look\_res\_diagnoz\_koeff\_neus\_shape1: TShape;

look\_res\_diagnoz\_koeff\_norma\_shape: TShape;

look\_res\_diagnoz\_koeff\_neus\_shape2: TShape;

look\_res\_diagnoz\_koeff\_pat\_shape1: TShape;

look\_res\_diagnoz\_koeff\_l1: TLabel;

look\_res\_diagnoz\_koeff\_l2: TLabel;

look\_res\_diagnoz\_koeff\_point1: TLabel;

look\_res\_diagnoz\_koeff\_point2: TLabel;

look\_res\_diagnoz\_koeff\_point3: TLabel;

look\_res\_diagnoz\_koeff\_point4: TLabel;

look\_res\_diagnoz\_koeff\_point5: TLabel;

look\_res\_diagnoz\_koeff\_point6: TLabel;

look\_res\_diagnoz\_koeff\_diagnoz\_shape: TShape;

look\_res\_diagnoz\_koeff\_l3: TLabel;

look\_res\_diagnoz\_koeff\_l4: TLabel;

look\_res\_diagnoz\_koeff\_l5: TLabel;

look\_res\_diagnoz\_koeff\_l6: TLabel;

look\_res\_diagnoz\_koeff\_l7: TLabel;

look\_res\_diagnoz\_koeff\_top\_panel: TPanel;

look\_res\_diagnoz\_koeff\_bottom\_panel: TPanel;

look\_res\_make\_report\_btn: TAdvGlowButton;

look\_res\_diagnoz\_data\_StringGrid: TAdvStringGrid;

look\_res\_diagnoz\_points\_table\_switch: TRadioButton;

look\_res\_diagnoz\_koeffs\_table\_switch: TRadioButton;

checkTimer: TTimer;

look\_res\_choise\_load\_data: TAdvGlowButton;

del\_izm: TAction;

N15: TMenuItem;

N16: TMenuItem;

N17: TMenuItem;

look\_res\_graph\_chart: TChart;

look\_res\_graph\_choose: TAdvOfficeRadioGroup;

look\_res\_graph\_izm\_id\_CB: TComboBox;

look\_res\_choise\_memo: TDBAdvMemo;

Izm\_to\_base\_cb: TCheckBox;

Izm\_null\_result\_CB: TCheckBox;

izm\_write\_ui: TCheckBox;

izm\_manual\_switch: TCheckBox;

new\_izm\_date\_picker: TAdvMaskEdit;

look\_res\_log\_scale\_CB: TCheckBox;

look\_res\_log\_scale: TAdvSpinEdit;

look\_res\_choise\_method\_CB: TLUCombo;

look\_res\_choise\_koeffs\_CB: TLUCombo;

look\_res\_graph\_options\_max\_CB: TCheckBox;

Label2: TLabel;

look\_res\_graph\_options\_min\_CB: TCheckBox;

look\_res\_graph\_options\_average\_CB: TCheckBox;

Label3: TLabel;

look\_res\_graph\_options\_15\_all\_CB: TCheckBox;

look\_res\_graph\_options\_15\_all\_color: TShape;

look\_res\_graph\_options\_max\_all\_CB: TCheckBox;

look\_res\_graph\_options\_max\_all\_color: TShape;

look\_res\_graph\_options\_min\_all\_CB: TCheckBox;

look\_res\_graph\_options\_min\_all\_color: TShape;

look\_res\_graph\_options\_average\_all\_CB: TCheckBox;

look\_res\_graph\_options\_average\_all\_color: TShape;

Panel1: TPanel;

Panel2: TPanel;

look\_res\_graph\_izm\_CB: TLUCombo;

look\_res\_graph\_izm\_desc: TLabel;

look\_res\_choise\_R\_CB: TLUCombo;

look\_res\_graph\_data\_DBGrid: TAdvStringGrid;

look\_res\_numbers\_filter\_GB: TAdvGroupBox;

look\_res\_number\_filter\_param\_CB: TLUCombo;

look\_res\_number\_filter\_point\_CB: TLUCombo;

look\_res\_number\_filter\_options\_RG: TAdvOfficeRadioGroup;

look\_res\_number\_filter\_genparam\_CB: TCheckBox;

look\_res\_number\_genparam\_panel: TAdvPanel;

look\_res\_number\_izm\_info\_GB: TAdvGroupBox;

Label4: TLabel;

Label5: TLabel;

Label6: TLabel;

Label7: TLabel;

Label8: TLabel;

look\_res\_diagnoz\_numbers\_grid: TAdvStringGrid;

Label9: TLabel;

Label10: TLabel;

Label11: TLabel;

Label12: TLabel;

look\_res\_number\_izm\_CB: TLUCombo;

look\_res\_number\_fio\_label: TLabel;

look\_res\_number\_izm\_label: TLabel;

look\_res\_number\_desc\_label: TLabel;

look\_res\_number\_izm\_id\_CB: TComboBox;

look\_res\_del\_izm: TAdvGlowButton;

CheckBox1: TCheckBox;

procedure FormKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);

procedure checkTimerTimer(Sender: TObject);

procedure FormShow(Sender: TObject);

procedure GoToNewMeasureExecute(Sender: TObject);

procedure GoToLookResultsExecute(Sender: TObject);

procedure New\_izm\_pcChanging(Sender: TObject; var AllowChange: Boolean);

procedure Show\_all\_patExecute(Sender: TObject);

Procedure Col\_resize(grid:TDBAdvGrid; name,header:string; width:integer);

procedure New\_izm\_DB\_GridSelectCell(Sender: TObject; ACol, ARow: Integer;

var CanSelect: Boolean);

procedure new\_izm\_DBMemo\_primSelectionChange(Sender: TObject);

procedure new\_patientExecute(Sender: TObject);

procedure enable\_disable\_new\_izm\_controls(dir:boolean);

procedure new\_izm\_reset\_btnClick(Sender: TObject);

procedure new\_izm\_save\_btnClick(Sender: TObject);

procedure del\_patientExecute(Sender: TObject);

procedure red\_patientExecute(Sender: TObject);

procedure look\_results\_DB\_GridSelectCell(Sender: TObject; ACol,

ARow: Integer; var CanSelect: Boolean);

procedure look\_res\_choise\_izmExit(Sender: TObject);

procedure look\_res\_load\_dataExecute(Sender: TObject);

procedure look\_res\_choise\_load\_dataClick(Sender: TObject);

procedure create\_series(name,num,izm\_num:string; x\_array:array of integer;

scale:extended);

procedure look\_res\_graph\_chooseClick(Sender: TObject);

procedure look\_res\_show\_15Execute(Sender: TObject);

procedure new\_izm\_look\_red\_btnClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure N10Click(Sender: TObject);

procedure Begin\_izmExecute(Sender: TObject);

procedure New\_izm\_DB\_GridDblClickCell(Sender: TObject; ARow, ACol: Integer);

procedure FormKeyPress(Sender: TObject; var Key: Char);

procedure New\_izm\_DB\_GridUpdateColumnSize(Sender: TObject; ACol: Integer;

var AWidth: Integer);

procedure look\_res\_choise\_izmDblClick(Sender: TObject);

procedure look\_res\_graph\_points\_forwClick(Sender: TObject);

procedure look\_res\_graph\_points\_backClick(Sender: TObject);

procedure make\_point\_active(num:integer);

procedure look\_res\_point\_CBChange(Sender: TObject);

procedure N12Click(Sender: TObject);

procedure N13Click(Sender: TObject);

procedure look\_res\_diagnoz\_data\_StringGridGetCellBorderProp(Sender: TObject;

ARow, ACol: Integer; LeftPen, TopPen, RightPen, BottomPen: TPen);

procedure look\_res\_diagnoz\_data\_StringGridGetAlignment(Sender: TObject;

ARow, ACol: Integer; var HAlign: TAlignment; var VAlign: TVAlignment);

procedure look\_res\_diagnoz\_data\_StringGridGetDisplText(Sender: TObject;

ACol, ARow: Integer; var Value: string);

procedure look\_res\_diagnoz\_points\_table\_switchClick(Sender: TObject);

procedure look\_res\_diagnoz\_koeffs\_table\_switchClick(Sender: TObject);

procedure del\_izmExecute(Sender: TObject);

function Get\_R (point:string; measure:string; type\_of:integer):string;

procedure look\_res\_diagnoz\_data\_StringGridGetFloatFormat(Sender: TObject;

ACol, ARow: Integer; var IsFloat: Boolean; var FloatFormat: string);

procedure N17Click(Sender: TObject);

procedure New\_izm\_DB\_GridGetCellBorderProp(Sender: TObject; ARow,

ACol: Integer; LeftPen, TopPen, RightPen, BottomPen: TPen);

procedure New\_izm\_DB\_GridGetCellBorder(Sender: TObject; ARow, ACol: Integer;

APen: TPen; var Borders: TCellBorders);

procedure look\_results\_DB\_GridGetCellBorder(Sender: TObject; ARow,

ACol: Integer; APen: TPen; var Borders: TCellBorders);

procedure look\_results\_DB\_GridGetCellBorderProp(Sender: TObject; ARow,

ACol: Integer; LeftPen, TopPen, RightPen, BottomPen: TPen);

procedure new\_izm\_fam\_editKeyPress(Sender: TObject; var Key: Char);

procedure new\_izm\_name\_editKeyPress(Sender: TObject; var Key: Char);

procedure new\_izm\_otch\_editKeyPress(Sender: TObject; var Key: Char);

procedure new\_izm\_date\_pickerKeyPress(Sender: TObject; var Key: Char);

procedure New\_izm\_DB\_GridKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

procedure look\_results\_DB\_GridKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

procedure look\_res\_graph\_chartClickSeries(Sender: TCustomChart;

Series: TChartSeries; ValueIndex: Integer; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure look\_res\_log\_scale\_CBClick(Sender: TObject);

function Sign(X:Extended):Integer;

function In\_power (x:extended; n:extended):extended;

procedure look\_res\_log\_scaleChange(Sender: TObject);

function get\_average\_point():extended;

function get\_average\_point\_all():extended;

procedure look\_res\_graph\_options\_average\_CBClick(Sender: TObject);

procedure look\_res\_graph\_options\_15\_all\_CBClick(Sender: TObject);

procedure look\_res\_graph\_options\_average\_all\_CBClick(Sender: TObject);

function get\_max\_min\_point(sw:integer):extended;

function get\_max\_min\_point\_all(sw:integer):extended;

procedure delete\_series(chart:TChart; name:string);

procedure look\_res\_graph\_options\_max\_CBClick(Sender: TObject);

procedure create\_series\_point(chart:TChart; name\_ser,type\_of:string);

procedure create\_series\_all(chart:TChart; name\_ser,type\_of:string);

procedure look\_res\_graph\_options\_min\_CBClick(Sender: TObject);

procedure look\_res\_graph\_options\_max\_all\_CBClick(Sender: TObject);

procedure look\_res\_graph\_options\_min\_all\_CBClick(Sender: TObject);

procedure look\_res\_graph\_izm\_CBChange(Sender: TObject);

procedure look\_res\_choise\_izmSelectItem(Sender: TObject; Item: TListItem;

Selected: Boolean);

procedure look\_res\_graph\_data\_DBGridGetAlignment(Sender: TObject; ARow,

ACol: Integer; var HAlign: TAlignment; var VAlign: TVAlignment);

procedure load\_number\_grid(izm\_num:string);

procedure look\_results\_pcChange(Sender: TObject);

procedure look\_res\_make\_report\_btnClick(Sender: TObject);

procedure look\_res\_number\_filter\_genparam\_CBClick(Sender: TObject);

procedure look\_res\_number\_izm\_CBChange(Sender: TObject);

procedure look\_res\_number\_fio\_labelMouseEnter(Sender: TObject);

procedure look\_res\_number\_fio\_labelMouseLeave(Sender: TObject);

procedure look\_res\_fio\_labelMouseEnter(Sender: TObject);

procedure look\_res\_fio\_labelMouseLeave(Sender: TObject);

procedure look\_res\_fio\_labelClick(Sender: TObject);

//procedure (izm\_num:string)

private

{ Private declarations }

public

var device: TDevice;{ Public declarations }

end;

Const number\_of\_params=7;

number\_of\_points=24;

number\_of\_repeat=5;

var

MainForm: TMainForm;

db\_grid\_cur\_row\_num:integer; //текцщий номер строки в гриде

lock\_pc:boolean; //переменная включающая режим блокирования изменения всех pageconrol'ov

act:string; //действие для кнопки сохранить

//варианты - сохранение и обновление

db\_grid\_wid: array[0..8] of integer;

gARow, gACol: integer; //текущие строка и столбец в таблице

koeff: array [0..number\_of\_params] of extended; //массив коэфициентов для сжатия графика

scales: array [0..number\_of\_params] of extended; //массив коэффициентов для уменьшения порядка

resized: array [0..number\_of\_params] of boolean; //массив фактов сжатия графиков

look\_res\_act\_page: integer; //активная вкладка просмотре результатов

implementation

uses DataUnit, ConstUnit, IzmUnit, AdminUnit, AboutUnit;

{$R \*.dfm}

//////////////////////////////////////////////////

/// ОБЩИЕ ПРОЦЕДУРЫ ФОРМЫ //

//////////////////////////////////////////////////

procedure TMainForm.checkTimerTimer(Sender: TObject);

begin

//Проверка на подключение к базе

if SQL\_DM.mysql\_con.Connected then

begin

MainForm.StatusBar.Panels[0].Text:=status\_bar\_database\_connection\_exisits;

MainForm.Enabled:=true;

end

else

begin

MainForm.StatusBar.Panels[0].Text:=status\_bar\_database\_connection\_not\_exisits;

MainForm.Enabled:=false;

end;

//проверка на подключение к прибору

if Device.IsConnected then

begin

MainForm.StatusBar.Panels[5].Text:=status\_bar\_device\_connected;

if New\_izm\_pc.ActivePageIndex=0 then

begin

new\_izm\_warning\_label.Hide;

new\_izm\_begin\_izm\_btn.Enabled:=true;

end;

end

else

begin

MainForm.StatusBar.Panels[5].Text:=status\_bar\_device\_not\_connected;

if New\_izm\_pc.ActivePageIndex=0 then

begin

new\_izm\_warning\_label.Show;

new\_izm\_begin\_izm\_btn.Enabled:=false;

end;

end;

end;

procedure TMainForm.FormShow(Sender: TObject);

begin

act:='';

//по умолчанию активен режим измерений

MainForm.GoToNewMeasure.Execute;

//проверяем устройство, в случае неподключения - показываем предупреждение

// и делаем недоступным начинание нового измерения

if not Device.IsConnected then

begin

new\_izm\_begin\_izm\_btn.Enabled:=false;

new\_izm\_warning\_label.Show;

end;

//проверяем количество пациентов, если равно 0 то запрещаем действия

// редактирования и удаления

if New\_izm\_DB\_Grid.RowCount=2 then

if New\_izm\_DB\_Grid.Cells[1,1]='' then

begin

New\_izm\_red\_pat.Enabled:=false;

New\_izm\_del\_pat.Enabled:=false;

new\_izm\_begin\_izm\_btn.Enabled:=false;

end;

//Получаем количество проведенных иссдедований

mysql\_query(SQL\_DM.mysql\_query1, 'SELECT Count(\*) FROM `'+table\_izm\_name+';',1);

MainForm.StatusBar.Panels[1].Text:=

status\_bar\_number\_of\_measurments+SQL\_DM.mysql\_DSQ1.DataSet.FieldByName('Count(\*)').AsString;

end;

procedure TMainForm.GoToLookResultsExecute(Sender: TObject);

begin

//доступность кнопок

MainForm.look\_results\_btn.Down:=true;

MainForm.new\_measure\_btn.Down:=false;

//видимость панелей и таблиц

look\_res\_panel.Show;

look\_results\_DB\_Grid.Show;

New\_izm\_DB\_Grid.SearchFooter.Visible:=false;

New\_izm\_Panel.Hide;

New\_izm\_DB\_Grid.Hide;

look\_results\_DB\_Grid.SearchFooter.Visible:=true;

//вкладки pagecontrol'a недоступны

look\_results\_pc.Pages[1].TabVisible:=false;

look\_results\_pc.Pages[2].TabVisible:=false;

look\_results\_pc.Pages[3].TabVisible:=false;

//Обновляем скроллы в мемо: все равно не отрисовывается(((

look\_res\_choise\_memo.RefreshMemo;

//показываем всех пациентов

MainForm.Show\_all\_pat.Execute;

end;

procedure TMainForm.GoToNewMeasureExecute(Sender: TObject);

begin

//доступность кнопок

MainForm.look\_results\_btn.Down:=false;

MainForm.new\_measure\_btn.Down:=true;

//видимость панелей и таблиц

look\_res\_panel.Hide;

look\_results\_DB\_Grid.Hide;

look\_results\_DB\_Grid.SearchFooter.Visible:=false;

New\_izm\_Panel.Show;

New\_izm\_DB\_Grid.Show;

New\_izm\_red\_panel.Hide;

//Обновляем скроллы в мемо: все равно не отрисовывается(((

new\_izm\_edit\_memo.Refresh;

new\_izm\_DBMemo\_prim.RefreshMemo;

//проверяем устройство

if not Device.IsConnected then

begin

new\_izm\_begin\_izm\_btn.Enabled:=false;

new\_izm\_warning\_label.Show;

end;

//показываем всех пациентов

MainForm.Show\_all\_pat.Execute;

New\_izm\_DB\_Grid.SearchFooter.Visible:=true;

end;

//переход в режим продвинутого пользователя

procedure TMainForm.FormKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

begin

If ((GetKeyState(VK\_CONTROL) AND 128)=128) and

((GetKeyState(VK\_menu) AND 128)=128) and

((GetKeyState(VK\_shift) AND 128)=128) and

((GetKeyState(vk\_f5) AND 128)=128)

then

ShowMessage('CTRL+Alt+Shift+F5 Pressed');

//при нажатии Crtl+Tab переходить между режимами просмотра и измерений

{If ((GetKeyState(VK\_CONTROL) AND 128)=128) and

((GetKeyState(VK\_TAB) AND 128)=128)

then

begin

if look\_results\_btn.Down then GoToNewMeasure.Execute

else if new\_measure\_btn.Down then GoToLookResults.Execute;

end;}

end;

procedure TMainForm.FormKeyPress(Sender: TObject; var Key: Char);

begin

//при нажатии Esc

if key=#27 then

//если находимся в редактировании пациента, то переходим обратно в просмотр

if (act<>'') and (new\_measure\_btn.Down) then

enable\_disable\_new\_izm\_controls(true);

end;

procedure TMainForm.FormCreate(Sender: TObject);

begin

//создание экземпляра класса устройства для дальнейшей с ним работы

device:=DeviceRD.TDevice.Create;

// проверяем подключено ли устройство и если да, то открываем его

if Device.IsConnected then

begin

Device.Open\_device;

Device.Start\_waiting(1);

end;

//задаем ширины колонок в таблицах информации о пользователях

db\_grid\_wid[1]:=20;

db\_grid\_wid[2]:=120;

db\_grid\_wid[3]:=120;

db\_grid\_wid[4]:=120;

db\_grid\_wid[5]:=40;

db\_grid\_wid[6]:=40;

db\_grid\_wid[7]:=40;

db\_grid\_wid[8]:=110;

// загружаем информацию в таблицы справочной информации по методике Риодораку

ConstUnit.load\_info\_to\_diagnoz\_tables(false);

//задаем коэффициенты сжатия для графика

koeff[0]:=5;

koeff[1]:=7;

koeff[2]:=9;

koeff[3]:=11;

koeff[4]:=9;

koeff[5]:=11;

koeff[6]:=5;

//koeff[7]:=;

scales[0]:=1E6;

scales[1]:=1E-6;

scales[2]:=1E-6;

scales[3]:=1E-6;

scales[4]:=1;

scales[5]:=1E-9;

scales[6]:=1;

//scales[7]:=;

resized[0]:=false;

resized[1]:=false;

resized[2]:=false;

resized[3]:=false;

resized[4]:=false;

resized[5]:=false;

resized[6]:=false;

//resized[7]:=false;

end;

//главное меню

procedure TMainForm.N10Click(Sender: TObject);

begin

MainForm.Close;

end;

procedure TMainForm.N12Click(Sender: TObject);

begin

AdminForm.admin\_db\_cur\_table:='points';

AdminForm.ShowModal;

end;

procedure TMainForm.N13Click(Sender: TObject);

begin

AdminForm.admin\_db\_cur\_table:='params';

AdminForm.ShowModal;

end;

procedure TMainForm.N17Click(Sender: TObject);

begin

AboutBox.ShowModal;

end;

//пользовательские процедуры

procedure TMainForm.Col\_resize(grid:TDBAdvGrid; name,header:string; width:integer);

begin

grid.ColumnByFieldName[name].Width:=width;

grid.ColumnByFieldName[name].Header:=header;

grid.ColumnByFieldName[name].HeaderAlignment:=taCenter;

grid.ColumnByFieldName[name].Alignment:=taCenter;

end;

////////////////////////////////////

/// Процедуры Нового измерения ///

////////////////////////////////////

procedure TMainForm.Begin\_izmExecute(Sender: TObject);

begin

//в случае если устройство подключено

if Device.IsConnected then

begin

//назваем форму

IzmForm.Caption:=izm\_form\_name;

IzmForm.Izm\_condition\_label.Caption:=izm\_device\_ready;

//запускаем поток

Device.Start\_waiting(2);

//показываем форму измерений

IzmForm.Update;

IzmForm.ShowModal;

end;

end;

procedure TMainForm.new\_izm\_look\_red\_btnClick(Sender: TObject);

var can:boolean;

begin

can:=true;

//если результат 1 то напрямую на просмотр

//если много, то на страниу выбора измерения

if strtoint(new\_izm\_kol\_znach.Caption)>1 then

begin

MainForm.GoToLookResults.Execute;

MainForm.Show\_all\_pat.Execute;

look\_results\_DB\_GridSelectCell(self, 0, strtoint(new\_izm\_pat\_id.Caption), can);

end

else

begin

mysql\_query(SQL\_DM.mysql\_query1, 'SELECT `id` FROM `'+table\_izm\_name+

'` WHERE `patient\_num`='+new\_izm\_pat\_id.Caption,1);

with look\_res\_choise\_izm do begin

Items.clear;

Items.Add;

Items[0].SubItems[0]:=(SQL\_DM.mysql\_DSQ1.DataSet.FieldByName('id').AsString);

SelectFirst;

end;

MainForm.look\_res\_load\_data.Execute;

end;

end;

//переход фокуса

procedure TMainForm.new\_izm\_fam\_editKeyPress(Sender: TObject; var Key: Char);

begin

if key=#13 then new\_izm\_name\_edit.SetFocus;

end;

procedure TMainForm.new\_izm\_name\_editKeyPress(Sender: TObject; var Key: Char);

begin

if key=#13 then new\_izm\_otch\_edit.SetFocus;

end;

procedure TMainForm.new\_izm\_otch\_editKeyPress(Sender: TObject; var Key: Char);

begin

if key=#13 then new\_izm\_date\_picker.SetFocus;

end;

procedure TMainForm.new\_izm\_date\_pickerKeyPress(Sender: TObject; var Key: Char);

begin

if key=#13 then new\_izm\_edit\_memo.SetFocus;

end;

//служебные запрещающие процедуры

procedure TMainForm.new\_izm\_DBMemo\_primSelectionChange(Sender: TObject);

begin

new\_izm\_DBMemo\_prim.Selection:='';

end;

procedure TMainForm.New\_izm\_pcChanging(Sender: TObject; var AllowChange: Boolean);

begin

if lock\_pc then AllowChange:=false;

end;

procedure TMainForm.Show\_all\_patExecute(Sender: TObject);

var sel:boolean;

begin

db\_grid\_cur\_row\_num:=-1;

mysql\_query(SQL\_DM.mysql\_query2, 'SELECT \* FROM '+table\_patient\_name+';', 1);

sel:=true;

//загрузка дааных в таблицу в зависимости от того, какая из них видна

if New\_izm\_DB\_Grid.Visible then

begin

New\_izm\_DB\_Grid.DataSource:=SQL\_DM.mysql\_DSQ2;

Col\_resize(New\_izm\_DB\_Grid, 'id', new\_izm\_num\_name, db\_grid\_wid[1]);

Col\_resize(New\_izm\_DB\_Grid, 'fam', new\_izm\_fam\_name, db\_grid\_wid[2]);

Col\_resize(New\_izm\_DB\_Grid, 'name', new\_izm\_name\_name, db\_grid\_wid[3]);

Col\_resize(New\_izm\_DB\_Grid, 'otch', new\_izm\_otch\_name, db\_grid\_wid[4]);

Col\_resize(New\_izm\_DB\_Grid, 'day', new\_izm\_day\_name, db\_grid\_wid[5]);

Col\_resize(New\_izm\_DB\_Grid, 'mon', new\_izm\_mon\_name, db\_grid\_wid[6]);

Col\_resize(New\_izm\_DB\_Grid, 'year', new\_izm\_year\_name, db\_grid\_wid[7]);

Col\_resize(New\_izm\_DB\_Grid, 'opisanie', new\_izm\_opis\_name, db\_grid\_wid[8]);

New\_izm\_DB\_Grid.Enabled:=true;

//выбираем первую запись

New\_izm\_DB\_GridSelectCell(self, 1,1,sel);

end

else if look\_results\_DB\_Grid.Visible then

begin

look\_results\_DB\_Grid.DataSource:=SQL\_DM.mysql\_DSQ2;

Col\_resize(look\_results\_DB\_Grid, 'id', new\_izm\_num\_name, db\_grid\_wid[1]);

Col\_resize(look\_results\_DB\_Grid, 'fam', new\_izm\_fam\_name, db\_grid\_wid[2]);

Col\_resize(look\_results\_DB\_Grid, 'name', new\_izm\_name\_name, db\_grid\_wid[3]);

Col\_resize(look\_results\_DB\_Grid, 'otch', new\_izm\_otch\_name, db\_grid\_wid[4]);

Col\_resize(look\_results\_DB\_Grid, 'day', new\_izm\_day\_name, db\_grid\_wid[5]);

Col\_resize(look\_results\_DB\_Grid, 'mon', new\_izm\_mon\_name, db\_grid\_wid[6]);

Col\_resize(look\_results\_DB\_Grid, 'year', new\_izm\_year\_name, db\_grid\_wid[7]);

Col\_resize(look\_results\_DB\_Grid, 'opisanie', new\_izm\_opis\_name, db\_grid\_wid[8]);

look\_results\_DB\_Grid.Enabled:=true;

//выбираем первую запись

look\_results\_DB\_GridSelectCell(self, 1,1,sel);

end;

end;

/// Служебные процедуры для таблицы

procedure TMainForm.New\_izm\_DB\_GridGetCellBorder(Sender: TObject; ARow,

ACol: Integer; APen: TPen; var Borders: TCellBorders);

begin

if act='update' then

if (ARow=gARow) then

begin

APen.Width := 2;

if ACol<>0 then Borders:=[cbTop, cbBottom];

if ACol=1 then Borders:=[cbLeft,cbTop, cbBottom]

else if ACol=8 then Borders:=[cbRight,cbTop, cbBottom];

end;

end;

procedure TMainForm.New\_izm\_DB\_GridGetCellBorderProp(Sender: TObject; ARow,

ACol: Integer; LeftPen, TopPen, RightPen, BottomPen: TPen);

begin

LeftPen.Color := clRed;

RightPen.Color := clRed;

BottomPen.Color := clRed;

TopPen.Color := clRed;

end;

procedure TMainForm.New\_izm\_DB\_GridUpdateColumnSize(Sender: TObject;

ACol: Integer; var AWidth: Integer);

var

i,wid: Integer;

begin

wid:=0;

for i:=0 to New\_izm\_DB\_Grid.ColCount - 1 do

wid:=wid+New\_izm\_DB\_Grid.ColWidths[i];

if wid>New\_izm\_DB\_Grid.Width-New\_izm\_Add\_pat.Width-25 then

AWidth:=db\_grid\_wid[ACol];

end;

procedure TMainForm.New\_izm\_DB\_GridDblClickCell(Sender: TObject; ARow,

ACol: Integer);

var sel:boolean;

begin

sel:=true;

MainForm.New\_izm\_DB\_GridSelectCell(self,Acol,ARow,sel);

MainForm.red\_patient.Execute;

end;

//обновление

procedure TMainForm.New\_izm\_DB\_GridKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

begin

if (GetKeyState(vk\_f5) AND 128)=128 then

begin

Show\_all\_pat.Execute;

end;

end;

//выбор пациента

procedure TMainForm.New\_izm\_DB\_GridSelectCell(Sender: TObject; ACol,

ARow: Integer; var CanSelect: Boolean);

begin

gARow:=ARow; gACol:=ACol;

if MainForm.New\_izm\_DB\_Grid.Cells[1,Arow]<>'' then

begin

//Доступны кнопки редактирования и удаления

New\_izm\_red\_pat.Enabled:=true;

New\_izm\_del\_pat.Enabled:=true;

//Запоминаем номер текущей строки

db\_grid\_cur\_row\_num:=Arow;

new\_izm\_pat\_id.Caption:=MainForm.New\_izm\_DB\_Grid.Cells[1,Arow];

//фамилия имя отчество

MainForm.new\_izm\_fam\_znach.Caption:=MainForm.New\_izm\_DB\_Grid.Cells[2,Arow];

MainForm.new\_izm\_name\_znach.Caption:=MainForm.New\_izm\_DB\_Grid.Cells[3,Arow];

MainForm.new\_izm\_otch\_znach.Caption:=MainForm.New\_izm\_DB\_Grid.Cells[4,Arow];

//дата рождения

MainForm.new\_izm\_date\_znach.Caption:=MainForm.New\_izm\_DB\_Grid.Cells[5,Arow]+

'.'+MainForm.New\_izm\_DB\_Grid.Cells[6,Arow]+'.'+

MainForm.New\_izm\_DB\_Grid.Cells[7,Arow]+'г.';

//примечания:

with new\_izm\_DBMemo\_prim do

begin

DataSource:=SQL\_DM.mysql\_DSQ2;

DataField:='opisanie';

end;

//количество измерений:

mysql\_query(SQL\_DM.mysql\_query1, 'SELECT Count(\*) FROM `'+table\_izm\_name+

'` WHERE `patient\_num`='+MainForm.New\_izm\_DB\_Grid.Cells[1,Arow],1);

MainForm.new\_izm\_kol\_znach.Caption:=SQL\_DM.mysql\_DSQ1.DataSet.

FieldByName('Count(\*)').AsString;

// в зависимости от количества приведенныз

if MainForm.new\_izm\_kol\_znach.Caption<>'0' then

begin

new\_izm\_look\_red\_btn.Enabled:=true;

end

else new\_izm\_look\_red\_btn.Enabled:=false;

// позволяеем запустить новое измерение

if Device.IsConnected then

new\_izm\_begin\_izm\_btn.Enabled:=true

else new\_izm\_begin\_izm\_btn.Enabled:=false;

end;

end;

//действия над пациентами

procedure TMainForm.new\_patientExecute(Sender: TObject);

begin

//открываем панель редактирования

enable\_disable\_new\_izm\_controls(false);

//устанавливаем действие в положение добавления информации

act:='add';

new\_izm\_fam\_edit.SetFocus;

end;

procedure TMainForm.red\_patientExecute(Sender: TObject);

var str: string;

begin

enable\_disable\_new\_izm\_controls(false); //доступная только панель ввода

//Действие кнопки сохранить - обновление

act:='update';

//Переносим значения полей

new\_izm\_fam\_edit.Text:=new\_izm\_fam\_znach.Caption;

new\_izm\_name\_edit.Text:=new\_izm\_name\_znach.Caption;

new\_izm\_otch\_edit.Text:=new\_izm\_otch\_znach.Caption;

//для даты: удаляем "г."

str:=new\_izm\_date\_znach.Caption;

delete(str, length(str)-1,2);

new\_izm\_date\_picker.text:=str;

//мемо - комментарии

new\_izm\_edit\_memo.Lines:=new\_izm\_DBMemo\_prim.Lines;

new\_izm\_fam\_edit.SetFocus;

end;

procedure TMainForm.del\_patientExecute(Sender: TObject);

begin

//удаление пациента

if MessageBox(0,

PChar(del\_patient\_question\_mes),

PChar(mess\_caption),

MB\_ICONQUESTION or MB\_OKCANCEL

)=mrOK then

begin

try

mysql\_query(SQL\_DM.mysql\_query1,'DELETE FROM `'+table\_patient\_name+

'` WHERE `id`='+MainForm.New\_izm\_DB\_Grid.cells[1,db\_grid\_cur\_row\_num]+';',2);

finally

MainForm.Show\_all\_pat.Execute;

MessageBox(0,

PChar(del\_pat\_from\_database\_mes),

PChar(mess\_caption),

MB\_ICONINFORMATION or MB\_OK

);

if New\_izm\_DB\_Grid.RowCount=2 then

if New\_izm\_DB\_Grid.Cells[1,1]='' then

begin

New\_izm\_red\_pat.Enabled:=false;

New\_izm\_del\_pat.Enabled:=false;

new\_izm\_begin\_izm\_btn.Enabled:=false;

end;

end;

end;

end;

procedure TMainForm.del\_izmExecute(Sender: TObject);

begin

if MessageBox(0,

PChar(del\_measure\_question\_mes),

PChar(mess\_caption),

MB\_ICONQUESTION or MB\_OKCANCEL

)=mrOK then

begin

mysql\_query(SQL\_DM.mysql\_query1,'DELETE FROM `'+table\_izm\_name+

'` WHERE `id`='+look\_res\_choise\_izm.Selected.SubItems.Strings[0]+';',2);

look\_res\_choise\_load\_data.Enabled:=false;

look\_res\_del\_izm.Enabled:=false;

look\_res\_izm\_description.Caption:='';

Show\_all\_pat.Execute;

end;

end;

//Кнопки сохранить и отменить

procedure TMainForm.new\_izm\_reset\_btnClick(Sender: TObject);

begin

enable\_disable\_new\_izm\_controls(true);

end;

procedure TMainForm.new\_izm\_save\_btnClick(Sender: TObject);

var day,mon,year:string;

stream:TMemoryStream;

begin

day:=(new\_izm\_date\_picker.text)[1]+(new\_izm\_date\_picker.text)[2];

mon:=(new\_izm\_date\_picker.text)[4]+(new\_izm\_date\_picker.text)[5];

year:=(new\_izm\_date\_picker.text)[7]+(new\_izm\_date\_picker.text)[8]+

(new\_izm\_date\_picker.text)[9]+(new\_izm\_date\_picker.text)[10];

if act='add' then

begin

//сохраняем информацию из полей в базу:

//запрос на выбор всего что есть

mysql\_query(SQL\_DM.mysql\_query2, 'SELECT \* FROM '+table\_patient\_name+';', 1);

try

SQL\_DM.mysql\_DSQ2.DataSet:=SQL\_DM.mysql\_query2;

with SQL\_DM.mysql\_DSQ2.DataSet do begin

Active:=true;

Append;

stream := TMemoryStream.Create;

new\_izm\_edit\_memo.Lines.SaveToStream(stream);

(FieldByName('opisanie') as TBlobField).LoadFromStream(Stream);

FieldByName('fam').AsString:=new\_izm\_fam\_edit.Text;

FieldByName('name').AsString:=new\_izm\_name\_edit.Text;

FieldByName('otch').AsString:=new\_izm\_otch\_edit.Text;

FieldByName('day').AsString:=day;

FieldByName('mon').AsString:=mon;

FieldByName('year').AsString:=year;

Post;

end;

stream.Free;

finally

MainForm.enable\_disable\_new\_izm\_controls(true);

MainForm.Show\_all\_pat.Execute;

MessageBox(0,

PChar(add\_pat\_to\_database\_mes),

PChar(mess\_caption),

MB\_ICONINFORMATION or MB\_OK

);

// делаем доступными действия редактирования и удаления

if New\_izm\_DB\_Grid.RowCount=2 then

if New\_izm\_DB\_Grid.Cells[1,1]='' then

begin

New\_izm\_red\_pat.Enabled:=true;

New\_izm\_del\_pat.Enabled:=true;

if not Device.IsConnected then

begin

new\_izm\_begin\_izm\_btn.Enabled:=true;

new\_izm\_warning\_label.hide;

end;

end;

end;

end

else if act='update' then begin

//запихиваем инфу в датасет, и соответственно в базу

//запрос на выбор всего что есть

mysql\_query(SQL\_DM.mysql\_query2, 'SELECT \* FROM '+table\_patient\_name+

' WHERE `id`='+MainForm.New\_izm\_DB\_Grid.cells[1,db\_grid\_cur\_row\_num]+';', 1);

try

with SQL\_DM.mysql\_DSQ2.DataSet do begin

Edit;

stream := TMemoryStream.Create;

new\_izm\_edit\_memo.Lines.SaveToStream(stream);

(FieldByName('opisanie') as TBlobField).LoadFromStream(Stream);

FieldByName('fam').AsString:=new\_izm\_fam\_edit.Text;

FieldByName('name').AsString:=new\_izm\_name\_edit.Text;

FieldByName('otch').AsString:=new\_izm\_otch\_edit.Text;

FieldByName('day').AsString:=day;

FieldByName('mon').AsString:=mon;

FieldByName('year').AsString:=year;

UpdateRecord;

Post;

end;

stream.Free;

finally

MainForm.enable\_disable\_new\_izm\_controls(true);

MainForm.Show\_all\_pat.Execute;

MessageBox(0,

PChar(red\_pat\_in\_database\_mes),

PChar(mess\_caption),

MB\_ICONINFORMATION or MB\_OK

);

end;

end;

end;

//Пользовательские процедуры

procedure TMainForm.enable\_disable\_new\_izm\_controls(dir:boolean);

begin

//скрываем панель информации

New\_izm\_info\_panel.Visible:=dir;

//показываем панель редактирования

New\_izm\_red\_panel.Visible:=not dir;

//инициализируем поля ввода

new\_izm\_fam\_edit.Clear;

new\_izm\_name\_edit.Clear;

new\_izm\_otch\_edit.Clear;

new\_izm\_date\_picker.text:='01.01.1940';

new\_izm\_edit\_memo.Clear;

//блокируем выход пользователя из вкладки

lock\_pc:=not dir;

// и с помощью кнопок

New\_izm\_DB\_Grid.Enabled:= dir;

New\_izm\_Add\_pat.Enabled:= dir;

New\_izm\_red\_pat.Enabled:= dir;

New\_izm\_del\_pat.Enabled:= dir;

//если выходим на первый контекст обнуляем действие

if dir then act:='';

look\_results\_btn.Enabled:= dir;

new\_measure\_btn.Enabled:= dir;

end;

////////////////////////////////////////////////////////////////////////////////

//процедуры панели ПРОСМОТР РЕЗУЛЬТАТОВ //

////////////////////////////////////////////////////////////////////////////////

procedure TMainForm.look\_results\_pcChange(Sender: TObject);

begin

if look\_results\_pc.ActivePageIndex=0 then

begin

//закладки делаем видимыми

look\_results\_pc.Pages[1].TabVisible:=false;

look\_results\_pc.Pages[2].TabVisible:=false;

look\_results\_pc.Pages[3].TabVisible:=false;

//снимаем выделение с измерений

look\_res\_choise\_izm.Selected:=nil;

//по умолчанию устанавливаем опции подсчета значений

look\_res\_choise\_method\_CB.ItemIndex:=0;

look\_res\_choise\_koeffs\_CB.ItemIndex:=1;

look\_res\_choise\_R\_CB.ItemIndex:=0;

end;

look\_res\_act\_page:=look\_results\_pc.ActivePageIndex;

end;

//загрузка данных

procedure TMainForm.look\_res\_choise\_load\_dataClick(Sender: TObject);

begin

MainForm.look\_res\_load\_data.Execute;

end;

procedure TMainForm.look\_res\_load\_dataExecute(Sender: TObject);

var

//znach\_kol, povtor, points\_kol,

i,k: Integer;

izm\_num, tmp1,tmp2:string;

x\_array: array [0..120] of integer;

ser, ser1:TPointSeries;

type\_of\_count: Integer;

begin

//закладки делаем видимыми

look\_results\_pc.Pages[1].TabVisible:=true;

look\_results\_pc.Pages[2].TabVisible:=true;

look\_results\_pc.Pages[3].TabVisible:=true;

//забираем номер измерения

izm\_num:=look\_res\_choise\_izm.Selected.SubItems[0];

//начинаем загрузку данных в график

//очищаем все что есть

look\_res\_graph\_chart.SeriesList.Clear;

look\_res\_graph\_choose.Items.Clear;

//определяем количество отсчетов по х, то есть сколько точек и сколько повторов

mysql\_query(SQL\_DM.mysql\_query1, 'SELECT Count(\*) FROM `'+table\_znacheniya\_name+

'` WHERE `izm\_num`='+izm\_num,1);

//znach\_kol:= SQL\_DM.mysql\_DSQ1.DataSet.FieldByName('Count(\*)').AsInteger;

mysql\_query(SQL\_DM.mysql\_query1, 'SELECT MAX(povtor\_num) FROM `'+table\_znacheniya\_name+

'` WHERE `izm\_num`='+izm\_num,1);

//povtor:=SQL\_DM.mysql\_DSQ1.DataSet.FieldByName('MAX(povtor\_num)').AsInteger;

//points\_kol:=round(znach\_kol/povtor/6);

//Численные значения

load\_number\_grid(izm\_num);

//формируем массив по х

for i:=0 to 120 do x\_array[i]:=i;

look\_res\_graph\_chart.SeriesList.Clear;

//определяем состав группы управления графиками, а также добавляем серии

create\_series('R','3',izm\_num,x\_array,1/scales[0]);

create\_series('G0','4',izm\_num,x\_array,1/scales[1]);

create\_series('G1','5',izm\_num,x\_array,1/scales[2]);

create\_series('G2','6',izm\_num,x\_array,1/scales[3]);

create\_series('G3','7',izm\_num,x\_array,1/scales[4]);

create\_series('C','8',izm\_num,x\_array,1/scales[5]);

create\_series('Uсм','9',izm\_num,x\_array,1/scales[6]);

//инициализация resized

resized[0]:=false;

resized[1]:=false;

resized[2]:=false;

resized[3]:=false;

resized[4]:=false;

resized[5]:=false;

resized[6]:=false;

//resized[7]:=false;

// видимо блок для custom axes... хз...

//создаем серию

ser:=TPointSeries.Create(self);

ser.AddXY(0,0,'',clRed);

//задаем необходимые свойства серии

ser.ParentChart:=look\_res\_graph\_chart;

ser.ColorEachPoint:=true;

ser.Pointer.Style:=psCircle;

ser.Pointer.VertSize:=1;

ser.Pointer.HorizSize:=1;

ser.Visible:=true;

//добавляем серию к графику

look\_res\_graph\_chart.AddSeries(ser);

//создаем серию

ser1:=TPointSeries.Create(self);

ser1.AddXY(0,0,'',clRed);

//задаем необходимые свойства серии

ser1.ParentChart:=look\_res\_graph\_chart;

ser1.ColorEachPoint:=true;

ser1.Pointer.Style:=psCircle;

ser1.Pointer.VertSize:=1;

ser1.Pointer.HorizSize:=1;

ser1.Visible:=true;

//добавляем серию к графику

look\_res\_graph\_chart.AddSeries(ser1);

look\_res\_graph\_choose.ItemIndex:=0;

look\_res\_graph\_chart.Axes.Left.Automatic:=true;

look\_res\_point\_num\_label.Caption:='1';

look\_res\_graph\_points\_back.Enabled:=false;

look\_res\_point\_CB.Items.Clear;

mysql\_query(SQL\_DM.mysql\_query1, 'SELECT `name` FROM `'+table\_points\_name+';',1);

SQL\_DM.mysql\_DSQ1.DataSet.First;

i:=0;k:=0;

//for i:=0 to 23 do

repeat

look\_res\_point\_CB.Items.Add(

inttostr(i+1)+'. '+SQL\_DM.mysql\_DSQ1.DataSet.FieldByName('name').AsString);

if i=11 then

begin

SQL\_DM.mysql\_DSQ1.DataSet.Next;

k:=0;

inc(i);

end

else

begin

if k=0 then

begin

inc(k);

SQL\_DM.mysql\_DSQ1.DataSet.Next;

SQL\_DM.mysql\_DSQ1.DataSet.Next;

SQL\_DM.mysql\_DSQ1.DataSet.Next;

SQL\_DM.mysql\_DSQ1.DataSet.Next;

SQL\_DM.mysql\_DSQ1.DataSet.Next;

SQL\_DM.mysql\_DSQ1.DataSet.Next;

end

else

if k=1 then

begin

dec(k);

SQL\_DM.mysql\_DSQ1.DataSet.Prior;

SQL\_DM.mysql\_DSQ1.DataSet.Prior;

SQL\_DM.mysql\_DSQ1.DataSet.Prior;

SQL\_DM.mysql\_DSQ1.DataSet.Prior;

SQL\_DM.mysql\_DSQ1.DataSet.Prior;

end;

inc(i);

end;

until i=24;

look\_res\_point\_CB.Text:=look\_res\_point\_CB.Items[0];

// степень сжатия для 0 серии:

look\_res\_log\_scale.Value:=round(koeff[0]);

look\_res\_log\_scale.Enabled:=true;

look\_res\_point\_CB.ItemIndex:=0;

//чек-боксы

look\_res\_graph\_options\_15\_CB.State:=cbUnchecked;

look\_res\_graph\_options\_max\_CB.State:=cbUnchecked;

look\_res\_graph\_options\_min\_CB.State:=cbUnchecked;

look\_res\_graph\_options\_average\_CB.State:=cbUnchecked;

look\_res\_graph\_options\_15\_all\_CB.State:=cbUnchecked;

look\_res\_graph\_options\_max\_all\_CB.State:=cbUnchecked;

look\_res\_graph\_options\_min\_all\_CB.State:=cbUnchecked;

look\_res\_graph\_options\_average\_all\_CB.State:=cbUnchecked;

look\_res\_graph\_chart.Axes.Bottom.Automatic:=false;

look\_res\_graph\_chart.Axes.Bottom.Maximum:=24;

look\_res\_graph\_chart.Axes.Bottom.Minimum:=0;

//комбо-бокс измерений на закладке графиков и чисел:

look\_res\_graph\_izm\_CB.Items.Clear;

look\_res\_number\_izm\_CB.Items.Clear;

look\_res\_graph\_izm\_id\_CB.Items.Clear;

look\_res\_number\_izm\_id\_CB.Items.Clear;

for i := 0 to look\_res\_choise\_izm.Items.Count - 1 do

begin

look\_res\_graph\_izm\_CB.Items.Add(look\_res\_choise\_izm.Items[i].Caption);

look\_res\_graph\_izm\_id\_CB.Items.Add(look\_res\_choise\_izm.Items[i].SubItems[0]);

look\_res\_number\_izm\_CB.Items.Add(look\_res\_choise\_izm.Items[i].Caption);

look\_res\_number\_izm\_id\_CB.Items.Add(look\_res\_choise\_izm.Items[i].SubItems[0]);

end;

look\_res\_graph\_izm\_CB.ItemIndex:=look\_res\_choise\_izm.Selected.Index;

look\_res\_number\_izm\_CB.ItemIndex:=look\_res\_choise\_izm.Selected.Index;

//Описание измерения сверху графика:

if strlen(PCHAR(look\_res\_izm\_description.Caption))<50 then

look\_res\_graph\_izm\_desc.Caption:=look\_res\_izm\_description.Caption

else

begin

look\_res\_graph\_izm\_desc.Caption:=copy(look\_res\_izm\_description.Caption,0,47)+'...';

end;

//Описание пациента и измерения на закладке диагноза:

tmp1:=look\_res\_choise\_name\_znach.Caption;

tmp2:=look\_res\_choise\_otch\_znach.Caption;

tmp2:=look\_res\_choise\_fam\_znach.Caption+' '+tmp1[1]+'.'+tmp2[1]+'.';

look\_res\_fio\_label.Caption:=tmp2;

look\_res\_izm\_name\_label.Caption:=look\_res\_choise\_izm.Selected.Caption;

look\_res\_izm\_about\_label.Caption:=look\_res\_izm\_description.Caption;

//Описание пациента и измерения на закладке чисел:

look\_res\_number\_fio\_label.Caption:=look\_res\_fio\_label.Caption;

look\_res\_number\_izm\_label.Caption:=look\_res\_izm\_name\_label.Caption;

look\_res\_number\_desc\_label.Caption:=look\_res\_izm\_about\_label.Caption;

//////////////////////////////////

//Закладка результаты и диагноз //

//////////////////////////////////

with look\_res\_diagnoz\_data\_StringGrid do

begin

DefaultRowHeight:=16;

//получаем способ расчета сопротивления

type\_of\_count:=look\_res\_choise\_R\_CB.ItemIndex;

Cells[0,3]:='PL1';

Cells[1,3]:=Get\_R('PL1\_l',izm\_num,type\_of\_count);

Cells[0,4]:='MC9';

Cells[1,4]:=Get\_R('MC9\_l',izm\_num,type\_of\_count);

Cells[0,5]:='C9';

Cells[1,5]:=Get\_R('C9\_l',izm\_num,type\_of\_count);

Cells[3,3]:='PL1';

Cells[4,3]:=Get\_R('PL1\_r',izm\_num,type\_of\_count);

Cells[3,4]:='MC9';

Cells[4,4]:=Get\_R('MC9\_r',izm\_num,type\_of\_count);

Cells[3,5]:='C9';

Cells[4,5]:=Get\_R('C9\_r',izm\_num,type\_of\_count);

//рука левая Инь и Ян

Cells[1,6]:=floattostr(strtofloat(Cells[1,3])+strtofloat(Cells[1,4])+strtofloat(Cells[1,5]));

Cells[4,6]:=floattostr(strtofloat(Cells[4,3])+strtofloat(Cells[4,4])+strtofloat(Cells[4,5]));

Cells[0,8]:='GL1';

Cells[1,8]:=Get\_R('GL1\_l',izm\_num,type\_of\_count);

Cells[0,9]:='TR1';

Cells[1,9]:=Get\_R('TR1\_l',izm\_num,type\_of\_count);

Cells[0,10]:='IG1';

Cells[1,10]:=Get\_R('IG1\_l',izm\_num,type\_of\_count);

Cells[3,8]:='GL1';

Cells[4,8]:=Get\_R('GL1\_r',izm\_num,type\_of\_count);

Cells[3,9]:='TR1';

Cells[4,9]:=Get\_R('TR1\_r',izm\_num,type\_of\_count);

Cells[3,10]:='IG1';

Cells[4,10]:=Get\_R('IG1\_r',izm\_num,type\_of\_count);

//рука правая Инь и Ян

Cells[1,11]:=floattostr(strtofloat(Cells[1,8])+strtofloat(Cells[1,9])+strtofloat(Cells[1,10]));

Cells[4,11]:=floattostr(strtofloat(Cells[4,8])+strtofloat(Cells[4,9])+strtofloat(Cells[4,10]));

Cells[1,12]:=summ\_left\_hands+floattostr(rounder(strtofloat(Cells[1,11])+strtofloat(Cells[1,6]),2))+'МОм';

Cells[4,12]:=summ\_right\_hands+floattostr(rounder(strtofloat(Cells[4,11])+strtofloat(Cells[4,6]),2))+'МОм';

Cells[4,13]:=summ\_hands+floattostr(

rounder(

strtofloat(Cells[4,11])+

strtofloat(Cells[4,6])+

strtofloat(Cells[1,11])+

strtofloat(Cells[1,6]),

2))+'МОм';

Cells[6,3]:='F1';

Cells[7,3]:=Get\_R('F1\_l',izm\_num,type\_of\_count);

Cells[6,4]:='RP1';

Cells[7,4]:=Get\_R('RP1\_l',izm\_num,type\_of\_count);

Cells[6,5]:='R0';

Cells[7,5]:=Get\_R('R0\_l',izm\_num,type\_of\_count);

Cells[9,3]:='F1';

Cells[10,3]:=Get\_R('F1\_r',izm\_num,type\_of\_count);

Cells[9,4]:='RP1';

Cells[10,4]:=Get\_R('RP1\_r',izm\_num,type\_of\_count);

Cells[9,5]:='R0';

Cells[10,5]:=Get\_R('R0\_r',izm\_num,type\_of\_count);

Cells[7,6]:=floattostr(strtofloat(Cells[7,3])+strtofloat(Cells[7,4])+strtofloat(Cells[7,5]));

Cells[10,6]:=floattostr(strtofloat(Cells[10,3])+strtofloat(Cells[10,4])+strtofloat(Cells[10,5]));

Cells[6,8]:='E45';

Cells[7,8]:=Get\_R('E45\_l',izm\_num,type\_of\_count);

Cells[6,9]:='VB44';

Cells[7,9]:=Get\_R('VB44\_l',izm\_num,type\_of\_count);

Cells[6,10]:='V67';

Cells[7,10]:=Get\_R('V67\_l',izm\_num,type\_of\_count);

Cells[9,8]:='E45';

Cells[10,8]:=Get\_R('E45\_r',izm\_num,type\_of\_count);

Cells[9,9]:='VB44';

Cells[10,9]:=Get\_R('VB44\_r',izm\_num,type\_of\_count);

Cells[9,10]:='V67';

Cells[10,10]:=Get\_R('V67\_r',izm\_num,type\_of\_count);

Cells[7,11]:=floattostr(strtofloat(Cells[7,8])+strtofloat(Cells[7,9])+strtofloat(Cells[7,10]));

Cells[10,11]:=floattostr(strtofloat(Cells[10,8])+strtofloat(Cells[10,9])+strtofloat(Cells[10,10]));

Cells[7,12]:=summ\_left\_legs+floattostr(rounder(strtofloat(Cells[7,11])+strtofloat(Cells[7,6]),2))+'МОм';

Cells[10,12]:=summ\_right\_legs+floattostr(rounder(strtofloat(Cells[10,11])+strtofloat(Cells[10,6]),2))+'МОм';

Cells[11,13]:=summ\_legs+floattostr(

rounder(

strtofloat(Cells[10,11])+

strtofloat(Cells[10,6])+

strtofloat(Cells[7,11])+

strtofloat(Cells[7,6]),

2))+'МОм';

case look\_res\_choise\_koeffs\_CB.ItemIndex of

0: begin //Руки + Ноги

look\_res\_diagnoz\_data\_znach\_label.Caption:=floattostr(rounder(

(strtofloat(Cells[10,11])+

strtofloat(Cells[10,6]) +

strtofloat(Cells[7,11]) +

strtofloat(Cells[7,6]) +

strtofloat(Cells[4,11]) + //рука правая Инь и Ян

strtofloat(Cells[4,6]) + //рука левая Инь и Ян

strtofloat(Cells[1,11]) + //рука правая Инь и Ян

strtofloat(Cells[1,6]) //рука левая Инь и Ян

)/24

,3));

end;

1: begin //Только руки

look\_res\_diagnoz\_data\_znach\_label.Caption:=floattostr(rounder(

(strtofloat(Cells[4,11]) + //рука правая Инь и Ян

strtofloat(Cells[4,6]) + //рука левая Инь и Ян

strtofloat(Cells[1,11]) + //рука правая Инь и Ян

strtofloat(Cells[1,6]) //рука левая Инь и Ян

)/12

,3));

end;

2: begin //Только ноги

look\_res\_diagnoz\_data\_znach\_label.Caption:=floattostr(rounder(

(strtofloat(Cells[10,11])+

strtofloat(Cells[10,6]) +

strtofloat(Cells[7,11]) +

strtofloat(Cells[7,6]))/12

,3));

end;

end;

end;

//индикация на графике сопротивлений

look\_res\_diagnoz\_koeff\_diagnoz\_shape.Left:=142+

round(strtofloat(look\_res\_diagnoz\_data\_znach\_label.Caption)\*100);

//активная закладка графиков

if look\_res\_act\_page<>0 then

look\_results\_pc.ActivePageIndex:=look\_res\_act\_page

else look\_results\_pc.ActivePageIndex:=2;

end;

//Создание серии с выборкой значений из базы

procedure TMainForm.create\_series(name,num,izm\_num:string;

x\_array:array of integer; scale:extended);

var i,j,k:integer;

y\_arrays: array [0..number\_of\_params-1,

1..number\_of\_points,

1..number\_of\_repeat] of double;

name\_array: array [0..number\_of\_points-1] of string;

ser:TPointSeries;

i1: Integer;

kk:boolean;

begin

//инициализация y\_arrays

for i:=0 to number\_of\_params-1 do

for j:=1 to number\_of\_points do

for k:=1 to number\_of\_repeat do

y\_arrays[i,j,k]:=0;

//добавляем параметр в окно выбора

look\_res\_graph\_choose.Items.Add(name);

//загружаем значения для параметра

//в формате PL1\_r PL1\_l и т.д.

i:=1;k:=0;i1:=1; kk:=false;

//for i:=1 to 24 do

repeat

//пока упрощенно на все точки

mysql\_query(SQL\_DM.mysql\_query1, 'SELECT `znach` FROM `'+table\_znacheniya\_name+

'` WHERE (`izm\_num`='+izm\_num+' AND `param\_num`='+num+' AND `point\_num`='+

inttostr(i)+') ORDER BY `rd\_znacheniya`.`povtor\_num` ASC',1);

SQL\_DM.mysql\_DSQ1.DataSet.First;

for j:=1 to number\_of\_repeat do

begin

if SQL\_DM.mysql\_DSQ1.DataSet.RecordCount>0 then

y\_arrays[strtoint(num)-3,i1,j]:=

SQL\_DM.mysql\_DSQ1.DataSet.FieldByName('znach').AsFloat\*scale

else y\_arrays[strtoint(num)-3,i1,j]:=0;

SQL\_DM.mysql\_DSQ1.DataSet.Next;

end;

if i=12 then begin kk:=true; k:=-1; end;

if k<=0 then

begin

if kk then begin inc(i); kk:=false; end else i:=i+6;

inc(k);

end

else if k=1 then

begin

i:=i-5;

dec(k);

end;

inc(i1);

until i1=25;

//выбираем имена точек

mysql\_query(SQL\_DM.mysql\_query1, 'SELECT `name` FROM `'+table\_points\_name+';',1);

SQL\_DM.mysql\_DSQ1.DataSet.First;

i:=0;k:=0;

//for i:=0 to 23 do

repeat

name\_array[i]:=SQL\_DM.mysql\_DSQ1.DataSet.FieldByName('name').AsString;

if i=11 then

begin

SQL\_DM.mysql\_DSQ1.DataSet.Next;

k:=0;

inc(i);

end

else

begin

if k=0 then

begin

inc(k);

SQL\_DM.mysql\_DSQ1.DataSet.Next;

SQL\_DM.mysql\_DSQ1.DataSet.Next;

SQL\_DM.mysql\_DSQ1.DataSet.Next;

SQL\_DM.mysql\_DSQ1.DataSet.Next;

SQL\_DM.mysql\_DSQ1.DataSet.Next;

SQL\_DM.mysql\_DSQ1.DataSet.Next;

end

else

if k=1 then

begin

dec(k);

SQL\_DM.mysql\_DSQ1.DataSet.Prior;

SQL\_DM.mysql\_DSQ1.DataSet.Prior;

SQL\_DM.mysql\_DSQ1.DataSet.Prior;

SQL\_DM.mysql\_DSQ1.DataSet.Prior;

SQL\_DM.mysql\_DSQ1.DataSet.Prior;

end;

inc(i);

end;

until i=24;

//создаем серию

ser:=TPointSeries.Create(self);

ser.Title:=name;

for i:=1 to number\_of\_points do

for j:=1 to number\_of\_repeat do

begin

//загружаем в нее значения из массива

ser.AddXY(j/5+i-1,y\_arrays[strtoint(num)-3,i,j],'',clRed);

end;

//загружаем названия точек

look\_res\_graph\_chart.Axes.Bottom.Items.Clear;

for i := 0 to ser.Count - 1 do

if ((i-2) mod 5=0) and (i>0) then

look\_res\_graph\_chart.Axes.Bottom.Items.Add(ser.XValue[i], name\_array[i div 5])

else look\_res\_graph\_chart.Axes.Bottom.Items.Add(ser.XValue[i], ' ');

//задаем необходимые свойства серии

ser.ParentChart:=look\_res\_graph\_chart;

ser.ColorEachPoint:=true;

ser.Pointer.Style:=psCircle;

ser.Pointer.VertSize:=3;

ser.Pointer.HorizSize:=3;

ser.Visible:=false;

//добавляем серию к графику

look\_res\_graph\_chart.AddSeries(ser);

end;

//Выбор измерений

procedure TMainForm.look\_res\_choise\_izmDblClick(Sender: TObject);

begin

if look\_res\_choise\_izm.Selected<>nil then

begin

look\_res\_load\_data.Execute;

end;

end;

procedure TMainForm.look\_res\_choise\_izmExit(Sender: TObject);

begin

look\_res\_izm\_description.Caption:='';

look\_res\_choise\_load\_data.Enabled:=false;

end;

procedure TMainForm.look\_res\_choise\_izmSelectItem(Sender: TObject;

Item: TListItem; Selected: Boolean);

begin

if Selected then begin

mysql\_query(SQL\_DM.mysql\_query4, 'SELECT `opisanie` FROM `'+table\_izm\_name+

'` WHERE `id`='+look\_res\_choise\_izm.Selected.SubItems[0],1);

SQL\_DM.mysql\_DSQ4.DataSet:=SQL\_DM.mysql\_query4;

look\_res\_izm\_description.Caption:=SQL\_DM.mysql\_DSQ4.DataSet.FieldByName('opisanie').AsString;

/// проблема с выходом из листа и входом в панель действий, в панели

/// действий пришлось убрать дествие "загрузить данные"

look\_res\_choise\_load\_data.Enabled:=true;

look\_res\_del\_izm.Enabled:=true;

end;

end;

//служебные процедуры таблицы пациентов

procedure TMainForm.look\_results\_DB\_GridGetCellBorder(Sender: TObject; ARow,

ACol: Integer; APen: TPen; var Borders: TCellBorders);

begin

APen.Width:=2;

if (ARow<>0) and (look\_results\_pc.Pages[1].TabVisible) and (db\_grid\_cur\_row\_num=ARow) then

begin

APen.Width := 2;

if ACol<>0 then Borders:=[cbTop, cbBottom];

if ACol=1 then Borders:=[cbLeft,cbTop, cbBottom]

else if ACol=8 then Borders:=[cbRight,cbTop, cbBottom];

end;

end;

procedure TMainForm.look\_results\_DB\_GridGetCellBorderProp(Sender: TObject; ARow,

ACol: Integer; LeftPen, TopPen, RightPen, BottomPen: TPen);

begin

LeftPen.Color := clRed;

RightPen.Color := clRed;

BottomPen.Color := clRed;

TopPen.Color := clRed;

end;

//Обновление

procedure TMainForm.look\_results\_DB\_GridKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

begin

if (GetKeyState(vk\_f5) AND 128)=128 then

begin

Show\_all\_pat.Execute;

end;

end;

//Выбор пациента

procedure TMainForm.look\_results\_DB\_GridSelectCell(Sender: TObject; ACol,

ARow: Integer; var CanSelect: Boolean);

var i:integer;

begin

//если в клетке что-то есть то

if MainForm.look\_results\_DB\_Grid.Cells[1,Arow]<>'' then

begin

//Запоминаем номер текущей строки

db\_grid\_cur\_row\_num:=Arow;

//Загружаем информацию

//фамилия имя отчество

look\_res\_choise\_fam\_znach.Caption:=look\_results\_DB\_Grid.Cells[2,Arow];

look\_res\_choise\_name\_znach.Caption:=look\_results\_DB\_Grid.Cells[3,Arow];

look\_res\_choise\_otch\_znach.Caption:=look\_results\_DB\_Grid.Cells[4,Arow];

//дата рождения

look\_res\_choise\_date\_znach.Caption:=look\_results\_DB\_Grid.Cells[5,Arow]+

'.'+look\_results\_DB\_Grid.Cells[6,Arow]+'.'+

look\_results\_DB\_Grid.Cells[7,Arow]+'г.';

//примечания:

with look\_res\_choise\_memo do

begin

DataSource:=SQL\_DM.mysql\_DSQ2;

DataField:='opisanie';

end;

//количество измерений:

SQL\_DM.mysql\_DSQ1.DataSet:=SQL\_DM.mysql\_query1;

mysql\_query(SQL\_DM.mysql\_query1, 'SELECT Count(\*) FROM `'+table\_izm\_name+

'` WHERE `patient\_num`='+MainForm.look\_results\_DB\_Grid.Cells[1,Arow],1);

look\_res\_choise\_izm\_kol.Caption:=SQL\_DM.mysql\_DSQ1.DataSet.FieldByName('Count(\*)').AsString;

//если клоичесвто измерений больше 0

if MainForm.look\_res\_choise\_izm\_kol.Caption<>'0' then

begin

new\_izm\_look\_red\_btn.Enabled:=true;

end;

//загрузка дат измерений даты измерений:

mysql\_query(SQL\_DM.mysql\_query5, 'SELECT `date`, `id` FROM `'+table\_izm\_name+

'` WHERE `patient\_num`='+MainForm.look\_results\_DB\_Grid.Cells[1,Arow],1);

SQL\_DM.mysql\_DSQ5.DataSet:=SQL\_DM.mysql\_query5;

SQL\_DM.mysql\_DSQ5.DataSet.First;

i:=0;

look\_res\_choise\_izm.Items.Clear;

while not SQL\_DM.mysql\_DSQ5.DataSet.Eof do

begin

look\_res\_choise\_izm.Items.Add;

look\_res\_choise\_izm.Items[i].Caption:=SQL\_DM.mysql\_DSQ5.DataSet.FieldByName('date').AsString;

look\_res\_choise\_izm.Items[i].SubItems.Add(SQL\_DM.mysql\_DSQ5.DataSet.FieldByName('id').AsString);

inc(i);

SQL\_DM.mysql\_DSQ5.DataSet.Next;

end;

//обнуляем описание измерений

look\_res\_izm\_description.Caption:=main\_unit\_measure\_choise\_not\_selected;

//активным в любом случае становится закладка выбора информации

look\_results\_pc.ActivePageIndex:=0;

//Остальные закладки делаем невидимыми

look\_results\_pc.Pages[1].TabVisible:=false;

look\_results\_pc.Pages[2].TabVisible:=false;

look\_results\_pc.Pages[3].TabVisible:=false;

//Опции расчета

look\_res\_choise\_method\_CB.ItemIndex:=0;

look\_res\_choise\_koeffs\_CB.ItemIndex:=1;

look\_res\_choise\_R\_CB.ItemIndex:=0;

end;

end;

//Клики на метке о пациенте, отображается информация

procedure TMainForm.look\_res\_fio\_labelClick(Sender: TObject);

begin

//show\_patient\_info\_from\_izm\_num(izm\_num);

end;

procedure TMainForm.look\_res\_fio\_labelMouseEnter(Sender: TObject);

begin

look\_res\_fio\_label.Font.Style:=[fsUnderline];

end;

procedure TMainForm.look\_res\_fio\_labelMouseLeave(Sender: TObject);

begin

look\_res\_fio\_label.Font.Style:=[];

end;

procedure TMainForm.look\_res\_number\_fio\_labelMouseEnter(Sender: TObject);

begin

look\_res\_number\_fio\_label.Font.Style:=[fsUnderline];

end;

procedure TMainForm.look\_res\_number\_fio\_labelMouseLeave(Sender: TObject);

begin

look\_res\_number\_fio\_label.Font.Style:=[];

end;

/////////////////////////////////

/// Закладка графиков //

////////////////////////////////

//Клики на чекбоксы линий

procedure TMainForm.look\_res\_graph\_options\_15\_all\_CBClick(Sender: TObject);

begin

look\_res\_log\_scale\_CB.Checked:=false;

look\_res\_log\_scale\_CBClick(self);

if look\_res\_graph\_options\_15\_all\_CB.Checked then

begin

create\_series\_all(look\_res\_graph\_chart, 'average\_up\_all', 'average\_15+');

create\_series\_all(look\_res\_graph\_chart, 'average\_down\_all', 'average\_15-');

end

else

begin

delete\_series(look\_res\_graph\_chart,'average\_up\_all');

delete\_series(look\_res\_graph\_chart,'average\_down\_all');

end;

end;

procedure TMainForm.look\_res\_show\_15Execute(Sender: TObject);

begin

look\_res\_log\_scale\_CB.Checked:=false;

look\_res\_log\_scale\_CBClick(self);

if look\_res\_graph\_options\_15\_CB.Checked then

begin

create\_series\_point(look\_res\_graph\_chart, 'average\_up', 'average\_15+');

create\_series\_point(look\_res\_graph\_chart, 'average\_down', 'average\_15-');

end

else

begin

delete\_series(look\_res\_graph\_chart,'average\_up');

delete\_series(look\_res\_graph\_chart,'average\_down');

end;

end;

procedure TMainForm.look\_res\_graph\_options\_average\_all\_CBClick(Sender: TObject);

begin

look\_res\_log\_scale\_CB.Checked:=false;

look\_res\_log\_scale\_CBClick(self);

if look\_res\_graph\_options\_average\_all\_CB.Checked then

begin

create\_series\_all(look\_res\_graph\_chart, 'average\_all', 'average');

end

else

begin

delete\_series(look\_res\_graph\_chart,'average\_all');

end;

end;

procedure TMainForm.look\_res\_graph\_options\_average\_CBClick(Sender: TObject);

begin

look\_res\_log\_scale\_CB.Checked:=false;

look\_res\_log\_scale\_CBClick(self);

if look\_res\_graph\_options\_average\_CB.Checked then

begin

create\_series\_point(look\_res\_graph\_chart, 'average', 'average');

end

else

begin

delete\_series(look\_res\_graph\_chart,'average');

end;

end;

procedure TMainForm.look\_res\_graph\_options\_max\_all\_CBClick(Sender: TObject);

begin

look\_res\_log\_scale\_CB.Checked:=false;

look\_res\_log\_scale\_CBClick(self);

if look\_res\_graph\_options\_max\_all\_CB.Checked then

begin

create\_series\_all(look\_res\_graph\_chart, 'max\_all', 'max');

end

else

begin

delete\_series(look\_res\_graph\_chart,'max\_all');

end;

end;

procedure TMainForm.look\_res\_graph\_options\_max\_CBClick(Sender: TObject);

begin

look\_res\_log\_scale\_CB.Checked:=false;

look\_res\_log\_scale\_CBClick(self);

if look\_res\_graph\_options\_max\_CB.Checked then

begin

create\_series\_point(look\_res\_graph\_chart, 'max', 'max');

end

else

begin

delete\_series(look\_res\_graph\_chart,'max');

end;

end;

procedure TMainForm.look\_res\_graph\_options\_min\_all\_CBClick(Sender: TObject);

begin

look\_res\_log\_scale\_CB.Checked:=false;

look\_res\_log\_scale\_CBClick(self);

if look\_res\_graph\_options\_min\_all\_CB.Checked then

begin

create\_series\_all(look\_res\_graph\_chart, 'min\_all', 'min');

end

else

begin

delete\_series(look\_res\_graph\_chart,'min\_all');

end;

end;

procedure TMainForm.look\_res\_graph\_options\_min\_CBClick(Sender: TObject);

begin

look\_res\_log\_scale\_CB.Checked:=false;

look\_res\_log\_scale\_CBClick(self);

if look\_res\_graph\_options\_min\_CB.Checked then

begin

create\_series\_point(look\_res\_graph\_chart, 'min', 'min');

end

else

begin

delete\_series(look\_res\_graph\_chart,'min');

end;

end;

//Создание и удаление серий с прямыми

procedure TMainForm.create\_series\_point(chart: TChart; name\_ser,type\_of: string);

var i,point\_num:integer;

value:extended;

shape:TShape;

ser:TLineSeries;

begin

value:=0;

shape:=nil;

if type\_of='average\_15+' then

begin

value:=get\_average\_point;

value:=value+0.15\*value;

shape:=look\_res\_graph\_options\_15\_color;

end

else if type\_of='average\_15-' then

begin

value:=get\_average\_point;

value:=value-0.15\*value;

shape:=look\_res\_graph\_options\_15\_color;

end

else if type\_of='average' then

begin

value:=get\_average\_point;

shape:=look\_res\_graph\_options\_average\_color;

end

else if type\_of='max' then

begin

value:=get\_max\_min\_point(1);

shape:=look\_res\_graph\_options\_max\_color;

end

else if type\_of='min' then

begin

value:=get\_max\_min\_point(0);

shape:=look\_res\_graph\_options\_min\_color;

end;

point\_num:=0;

for I := 0 to chart.Axes.Bottom.Items.Count - 1 do

if (chart.Axes.Bottom.Items[i].Font.Style=[fsBold]) and

(chart.Axes.Bottom.Items[i].Font.Height=-11) then

begin

point\_num:=(i div 5) +1;

end;

ser:=TLineSeries.Create(self);

with ser do

begin

Color:=shape.Brush.Color;

LinePen.Width:=2;

Title:=name\_ser;

Visible:=true;

ParentChart:=chart;

if not(point\_num in [1,24]) then

begin

AddXY(point\_num-1.15,value,floattostr(value));

AddXY(point\_num+0.3,value,'');

end

else

if point\_num=1 then

begin

AddXY(point\_num-1,value,floattostr(value));

AddXY(point\_num+0.3,value,'');

end

else

begin

AddXY(point\_num-1.1,value,floattostr(value));

AddXY(point\_num+0,value,'');

end;

end;

end;

procedure TMainForm.create\_series\_all(chart: TChart; name\_ser,type\_of: string);

var value:extended;

shape:TShape;

ser:TLineSeries;

begin

value:=0;

shape:=nil;

if type\_of='average\_15+' then

begin

value:=get\_average\_point\_all;

value:=value+0.15\*value;

shape:=look\_res\_graph\_options\_15\_all\_color;

end

else if type\_of='average\_15-' then

begin

value:=get\_average\_point\_all;

value:=value-0.15\*value;

shape:=look\_res\_graph\_options\_15\_all\_color;

end

else if type\_of='average' then

begin

value:=get\_average\_point\_all;

shape:=look\_res\_graph\_options\_average\_all\_color;

end

else if type\_of='max' then

begin

value:=get\_max\_min\_point\_all(1);

shape:=look\_res\_graph\_options\_max\_all\_color;

end

else if type\_of='min' then

begin

value:=get\_max\_min\_point\_all(0);

shape:=look\_res\_graph\_options\_min\_all\_color;

end;

ser:=TLineSeries.Create(self);

with ser do

begin

Color:=shape.Brush.Color;

LinePen.Width:=2;

Title:=name\_ser;

Visible:=true;

ParentChart:=chart;

AddXY(0,value,floattostr(value));

AddXY(24,value,'');

end;

end;

procedure TMainForm.delete\_series(chart: TChart; name: string);

var

I: Integer;

begin

for I := 0 to chart.SeriesList.Count - 1 do

if (chart.SeriesList.Items[i].Title=name) then

begin

chart.SeriesList.Items[i].Visible:=false;

chart.SeriesList.Items[i].Free;

end;

end;

//вытаскивание средних, +/-15%, макс. и мин. значений

function TMainForm.get\_average\_point():extended;

var i:integer;

point\_name, param\_name, izm\_num:string;

tmp:extended;

begin

//При расчетах нулевые-ошибочные значения выкидываются и не учитываются!!!

//выясняем какая точка активная настоящий момент:

for I := 0 to look\_res\_graph\_chart.Axes.Bottom.Items.Count - 1 do

if (look\_res\_graph\_chart.Axes.Bottom.Items[i].Font.Style=[fsBold]) and

(look\_res\_graph\_chart.Axes.Bottom.Items[i].Font.Height=-11) then

begin

point\_name:=look\_res\_graph\_chart.Axes.Bottom.Items[i].Text;

end;

//параметр

param\_name:=look\_res\_graph\_choose.Items[look\_res\_graph\_choose.ItemIndex];

//Номер измерения

izm\_num:=look\_res\_choise\_izm.Selected.SubItems[0];

mysql\_query(SQL\_DM.mysql\_query4,

'SELECT `znach` FROM `'+table\_znacheniya\_name+'` WHERE '+

'(`izm\_num`='+izm\_num+') AND'+

'(`param\_num`=(SELECT `id` FROM '+table\_param\_name+' WHERE `name`='''+param\_name+''')) AND'+

'(`point\_num`=(SELECT `num` FROM '+table\_points\_name+' WHERE `name`='''+point\_name+'''))',2);

try

SQL\_DM.mysql\_DSQ4.DataSet.First;

tmp:=0;i:=0;

while not SQL\_DM.mysql\_DSQ4.DataSet.Eof do

begin

if SQL\_DM.mysql\_DSQ4.DataSet.FieldByName('znach').AsFloat<>0 then

begin

inc(i);

tmp:=tmp+SQL\_DM.mysql\_DSQ4.DataSet.FieldByName('znach').AsFloat;

end;

SQL\_DM.mysql\_DSQ4.DataSet.Next;

end;

if i<>0 then

result:=tmp/i/scales[look\_res\_graph\_choose.ItemIndex]

else result:=0;

except

result:=0;

end;

end;

function TMainForm.get\_average\_point\_all():extended;

var i:integer;

param\_name, izm\_num:string;

tmp:extended;

begin

//При расчетах нулевые-ошибочные значения выкидываются и не учитываются!!!

//параметр

param\_name:=look\_res\_graph\_choose.Items[look\_res\_graph\_choose.ItemIndex];

//Номер измерения

izm\_num:=look\_res\_choise\_izm.Selected.SubItems[0];

mysql\_query(SQL\_DM.mysql\_query4,

'SELECT `znach` FROM `'+table\_znacheniya\_name+'` WHERE '+

'(`izm\_num`='+izm\_num+') AND'+

'(`param\_num`=(SELECT `id` FROM '+table\_param\_name+' WHERE `name`='''+param\_name+'''))',2);

try

SQL\_DM.mysql\_DSQ4.DataSet.First;

tmp:=0;i:=0;

while not SQL\_DM.mysql\_DSQ4.DataSet.Eof do

begin

if SQL\_DM.mysql\_DSQ4.DataSet.FieldByName('znach').AsFloat<>0 then

begin

inc(i);

tmp:=tmp+SQL\_DM.mysql\_DSQ4.DataSet.FieldByName('znach').AsFloat;

end;

SQL\_DM.mysql\_DSQ4.DataSet.Next;

end;

if i<>0 then

result:=tmp/i/scales[look\_res\_graph\_choose.ItemIndex]

else result:=0;

except

result:=0;

end;

end;

function TMainForm.get\_max\_min\_point(sw:integer):extended;

var i:integer;

point\_name, param\_name, izm\_num:string;

begin

//При расчетах нулевые-ошибочные значения выкидываются и не учитываются!!!

//выясняем какая точка активная настоящий момент:

for I := 0 to look\_res\_graph\_chart.Axes.Bottom.Items.Count - 1 do

if (look\_res\_graph\_chart.Axes.Bottom.Items[i].Font.Style=[fsBold]) and

(look\_res\_graph\_chart.Axes.Bottom.Items[i].Font.Height=-11) then

begin

point\_name:=look\_res\_graph\_chart.Axes.Bottom.Items[i].Text;

end;

//параметр

param\_name:=look\_res\_graph\_choose.Items[look\_res\_graph\_choose.ItemIndex];

//Номер измерения

izm\_num:=look\_res\_choise\_izm.Selected.SubItems[0];

case sw of

0:

mysql\_query(SQL\_DM.mysql\_query4,

'SELECT MIN(`znach`) FROM `'+table\_znacheniya\_name+'` WHERE '+

' (`izm\_num`='+izm\_num+') AND'+

' (`znach` != 0) AND'+

' (`param\_num`=(SELECT `id` FROM '+table\_param\_name+' WHERE `name`='''+param\_name+''')) AND'+

' (`point\_num`=(SELECT `num` FROM '+table\_points\_name+' WHERE `name`='''+point\_name+'''))',2);

1:

mysql\_query(SQL\_DM.mysql\_query4,

'SELECT MAX(`znach`) FROM `'+table\_znacheniya\_name+'` WHERE '+

' (`izm\_num`='+izm\_num+') AND'+

' (`znach` != 0) AND'+

' (`param\_num`=(SELECT `id` FROM '+table\_param\_name+' WHERE `name`='''+param\_name+''')) AND'+

' (`point\_num`=(SELECT `num` FROM '+table\_points\_name+' WHERE `name`='''+point\_name+'''))',2);

end;

try

case sw of

1:

result:=SQL\_DM.mysql\_DSQ4.DataSet.FieldByName('MAX(`znach`)').AsFloat/scales[look\_res\_graph\_choose.ItemIndex];

0:

result:=SQL\_DM.mysql\_DSQ4.DataSet.FieldByName('MIN(`znach`)').AsFloat/scales[look\_res\_graph\_choose.ItemIndex];

else

result:=0;

end;

except

result:=0;

end;

end;

function TMainForm.get\_max\_min\_point\_all(sw:integer):extended;

var param\_name, izm\_num:string;

begin

//При расчетах нулевые-ошибочные значения выкидываются и не учитываются!!!

//параметр

param\_name:=look\_res\_graph\_choose.Items[look\_res\_graph\_choose.ItemIndex];

//Номер измерения

izm\_num:=look\_res\_choise\_izm.Selected.SubItems[0];

case sw of

0:

mysql\_query(SQL\_DM.mysql\_query4,

' SELECT MIN(`znach`) FROM `'+table\_znacheniya\_name+'` WHERE '+

' (`izm\_num`='+izm\_num+') AND '+

' (`znach` != 0) AND '+

' (`param\_num`=(SELECT `id` FROM '+table\_param\_name+' WHERE `name`='''+param\_name+'''))',2);

1:

mysql\_query(SQL\_DM.mysql\_query4,

' SELECT MAX(`znach`) FROM `'+table\_znacheniya\_name+'` WHERE '+

' (`izm\_num`='+izm\_num+') AND'+

' (`znach` != 0) AND'+

' (`param\_num`=(SELECT `id` FROM '+table\_param\_name+' WHERE `name`='''+param\_name+'''))',2);

end;

try

case sw of

1:

result:=SQL\_DM.mysql\_DSQ4.DataSet.FieldByName('MAX(`znach`)').AsFloat/scales[look\_res\_graph\_choose.ItemIndex];

0:

result:=SQL\_DM.mysql\_DSQ4.DataSet.FieldByName('MIN(`znach`)').AsFloat/scales[look\_res\_graph\_choose.ItemIndex];

else

result:=0;

end;

except

result:=0;

end;

end;

//Навигация по точкам

procedure TMainForm.look\_res\_graph\_points\_backClick(Sender: TObject);

var

num: Integer;

begin

look\_res\_point\_num\_label.Caption:=inttostr(strtoint(look\_res\_point\_num\_label.Caption)-1);

num:=strtoint(look\_res\_point\_num\_label.Caption);

if num-1=0 then look\_res\_graph\_points\_back.Enabled:=false

else look\_res\_graph\_points\_back.Enabled:=true;

look\_res\_graph\_points\_forw.Enabled:=true;

make\_point\_active(num);

look\_res\_point\_CB.ItemIndex:=look\_res\_point\_CB.ItemIndex-1;

//сбрасываем чек-боксы для точек:

if look\_res\_graph\_options\_15\_CB.Checked then look\_res\_graph\_options\_15\_CB.Checked:=false;

if look\_res\_graph\_options\_max\_CB.Checked then look\_res\_graph\_options\_max\_CB.Checked:=false;

if look\_res\_graph\_options\_min\_CB.Checked then look\_res\_graph\_options\_min\_CB.Checked:=false;

if look\_res\_graph\_options\_average\_CB.Checked then look\_res\_graph\_options\_average\_CB.Checked:=false;

look\_res\_graph\_chart.Repaint;

end;

procedure TMainForm.look\_res\_graph\_points\_forwClick(Sender: TObject);

var

num:integer;

begin

look\_res\_point\_num\_label.Caption:=inttostr(strtoint(look\_res\_point\_num\_label.Caption)+1);

num:=strtoint(look\_res\_point\_num\_label.Caption);

if num+1>24 then look\_res\_graph\_points\_forw.Enabled:=false

else look\_res\_graph\_points\_forw.Enabled:=true;

look\_res\_graph\_points\_back.Enabled:=true;

make\_point\_active(num);

look\_res\_point\_CB.ItemIndex:=look\_res\_point\_CB.ItemIndex+1;

//сбрасываем чек-боксы для точек:

if look\_res\_graph\_options\_15\_CB.Checked then look\_res\_graph\_options\_15\_CB.Checked:=false;

if look\_res\_graph\_options\_max\_CB.Checked then look\_res\_graph\_options\_max\_CB.Checked:=false;

if look\_res\_graph\_options\_min\_CB.Checked then look\_res\_graph\_options\_min\_CB.Checked:=false;

if look\_res\_graph\_options\_average\_CB.Checked then look\_res\_graph\_options\_average\_CB.Checked:=false;

//look\_res\_graph\_chart.Repaint;

end;

procedure TMainForm.look\_res\_point\_CBChange(Sender: TObject);

begin

make\_point\_active(look\_res\_point\_CB.ItemIndex+1);

end;

//Выбор активной точки

procedure TMainForm.make\_point\_active(num: Integer);

var

i: integer;

begin

look\_res\_point\_num\_label.Caption:=inttostr(num);

with look\_res\_graph\_chart do begin

for i := 0 to Series[0].Count - 1 do

if ((i-2) mod 5=0) and (i>0) then

Axes.Bottom.Items.Item[i].Font.Height:=-9;

with Axes.Bottom.Items.Item[(num-1)\*5+2].Font do

begin

Height:=-11;

style:=[fsBold];

end;

for i:= 0 to Series[0].Count - 1 do

Series[look\_res\_graph\_choose.ItemIndex].ValueColor[i]:=clRed;

for i:= (num-1)\*5 to (num-1)\*5+4 do

Series[look\_res\_graph\_choose.ItemIndex].ValueColor[i]:=clYellow;

//Series[SeriesList.Count-1].CustomVertAxis:=CustomAxes[1];

//Series[SeriesList.Count-1].Visible:=true;

//Series[SeriesList.Count-2].CustomVertAxis:=CustomAxes[0];

//Series[SeriesList.Count-2].Visible:=true;

CustomAxes[1].Visible:=true;

//CustomAxes[1].

CustomAxes[1].PositionPercent:=100/24\*num;

CustomAxes[0].Visible:=true;

CustomAxes[0].PositionPercent:=100/24\*(num-1)+100/24/5;

end;

end;

//сжатие графиков

procedure TMainForm.look\_res\_log\_scaleChange(Sender: TObject);

var

act\_ser: Integer;

begin

act\_ser:=0;

if look\_res\_graph\_chart.Series[0].Visible then act\_ser:=0

else if look\_res\_graph\_chart.Series[1].Visible then act\_ser:=1

else if look\_res\_graph\_chart.Series[2].Visible then act\_ser:=2

else if look\_res\_graph\_chart.Series[3].Visible then act\_ser:=3

else if look\_res\_graph\_chart.Series[4].Visible then act\_ser:=4

else if look\_res\_graph\_chart.Series[5].Visible then act\_ser:=5;

koeff[act\_ser]:=look\_res\_log\_scale.Value;

end;

procedure TMainForm.look\_res\_log\_scale\_CBClick(Sender: TObject);

var i, act\_ser,sign2:integer;

tmp:extended;

begin

with look\_res\_graph\_chart do

begin

//определяем активную серию

act\_ser:=0;

if Series[0].Visible then act\_ser:=0

else if Series[1].Visible then act\_ser:=1

else if Series[2].Visible then act\_ser:=2

else if Series[3].Visible then act\_ser:=3

else if Series[4].Visible then act\_ser:=4

else if Series[5].Visible then act\_ser:=5;

//else if Series[6].Visible then act\_ser:=6

//else if Series[7].Visible then act\_ser:=7;

//для случая сжатия:

if look\_res\_log\_scale\_CB.Checked then

begin

look\_res\_graph\_options\_15\_CB.Enabled:=false;

look\_res\_graph\_options\_15\_all\_CB.Enabled:=false;

look\_res\_graph\_options\_max\_CB.Enabled:=false;

look\_res\_graph\_options\_max\_all\_CB.Enabled:=false;

look\_res\_graph\_options\_min\_CB.Enabled:=false;

look\_res\_graph\_options\_min\_all\_CB.Enabled:=false;

look\_res\_graph\_options\_average\_CB.Enabled:=false;

look\_res\_graph\_options\_average\_all\_CB.Enabled:=false;

//изменяем массив сжатий

resized[act\_ser]:=true;

//проходим все значения серии

for i:=0 to Series[act\_ser].Count-1 do

begin

//если 0, то так и оставляем, если нет

if Series[act\_ser].YValue[i]<>0 then

begin

//получаем знак

sign2:=sign(Series[act\_ser].YValue[i]);

//избавляемся от знака

tmp:=sign(Series[act\_ser].YValue[i])\*Series[act\_ser].YValue[i];

//сжимаем

tmp:=in\_power(tmp\*scales[act\_ser],1/koeff[act\_ser]);

//in\_power(scales[act\_ser],koeff[act\_ser]);

//возвращаем знак

Series[act\_ser].YValue[i]:=sign2\*tmp;

end;

end;

Title.Text.add(main\_unit\_graphics\_compress+

floattostr(koeff[act\_ser])+

main\_unit\_graphics\_compress\_unit);

end

else

begin

look\_res\_graph\_options\_15\_CB.Enabled:=true;

look\_res\_graph\_options\_15\_all\_CB.Enabled:=true;

look\_res\_graph\_options\_max\_CB.Enabled:=true;

look\_res\_graph\_options\_max\_all\_CB.Enabled:=true;

look\_res\_graph\_options\_min\_CB.Enabled:=true;

look\_res\_graph\_options\_min\_all\_CB.Enabled:=true;

look\_res\_graph\_options\_average\_CB.Enabled:=true;

look\_res\_graph\_options\_average\_all\_CB.Enabled:=true;

if resized[act\_ser] then begin

resized[act\_ser]:=false;

for i:=0 to Series[act\_ser].Count-1 do

begin

if Series[act\_ser].YValue[i]<>0 then

begin

sign2:=sign(Series[act\_ser].YValue[i]);

tmp:=sign(Series[act\_ser].YValue[i])\*Series[act\_ser].YValue[i];

tmp:=in\_power(tmp,koeff[act\_ser])/scales[act\_ser];

Series[act\_ser].YValue[i]:=sign2\*tmp;

end;

end;

look\_res\_log\_scale.Enabled:=true;

Title.Text.Delete(1);

end;

end;

end;

end;

//выбор параметра для отображения

procedure TMainForm.look\_res\_graph\_chooseClick(Sender: TObject);

var i:integer;

begin

with look\_res\_graph\_chart do

begin

//Axes.Left.AdjustMaxMin;

Title.Visible:=true;

Title.Text.Clear;

for i:=0 to number\_of\_params-1 do

if i=look\_res\_graph\_choose.ItemIndex then

SeriesList.Items[i].Visible:=true

else SeriesList.Items[i].Visible:=false;

case look\_res\_graph\_choose.ItemIndex of

0:begin

Title.Text.Append(main\_unit\_graphics\_title\_R);

end;

1:begin

Title.Text.Append(main\_unit\_graphics\_title\_G0);

end;

2:begin

Title.Text.Append(main\_unit\_graphics\_title\_G1);

end;

3:begin

Title.Text.Append(main\_unit\_graphics\_title\_G2);

end;

4:begin

Title.Text.Append(main\_unit\_graphics\_title\_G3);

end;

5:begin

Title.Text.Append(main\_unit\_graphics\_title\_C);

end;

6:begin

Title.Text.Append(main\_unit\_graphics\_title\_Usm);

end;

end;

//установка коэффициента сжатия

look\_res\_log\_scale.Value:=round(koeff[look\_res\_graph\_choose.ItemIndex]);

//сжатия

if resized[look\_res\_graph\_choose.ItemIndex] then

begin

look\_res\_log\_scale.Enabled:=false;

look\_res\_log\_scale\_CB.Checked:=true;

end

else

begin

look\_res\_log\_scale.Enabled:=true;

look\_res\_log\_scale\_CB.Checked:=false;

end;

//Axes.Left.AdjustMaxMin;

//цвета и шрифты значений

for i:=0 to Series[0].Count - 1 do

begin

if ((i-2) mod 5=0) and (i>0) then

Axes.Bottom.Items.Item[i].Font.Height:=-9;

Series[look\_res\_graph\_choose.ItemIndex].ValueColor[i]:=clRed;

end;

with Axes.Bottom.Items.Item[2].Font do

begin

Height:=-11;

style:=[fsBold];

end;

for i:= 0 to 4 do

Series[look\_res\_graph\_choose.ItemIndex].ValueColor[i]:=clYellow;

Series[SeriesList.Count-1].CustomVertAxis:=CustomAxes[1];

Series[SeriesList.Count-1].Visible:=true;

Series[SeriesList.Count-2].CustomVertAxis:=CustomAxes[0];

Series[SeriesList.Count-2].Visible:=true;

CustomAxes[1].Visible:=true;

CustomAxes[1].PositionPercent:=100/24;

CustomAxes[0].Visible:=true;

CustomAxes[0].PositionPercent:=100/24/5;

look\_res\_point\_CB.Text:=look\_res\_point\_CB.Items[0];

look\_res\_point\_num\_label.Caption:='1';

look\_res\_graph\_points\_back.Enabled:=false;

look\_res\_graph\_points\_forw.Enabled:=true;

//Axes.Left.AdjustMaxMin;

look\_res\_point\_CB.ItemIndex:=0;

//look\_res\_log\_scale\_CB.Checked:=false;

//look\_res\_log\_scale\_CBClick(self);

end;

end;

//Выбор измерения

procedure TMainForm.look\_res\_graph\_izm\_CBChange(Sender: TObject);

begin

look\_res\_choise\_izm.

Items[strtoint(look\_res\_graph\_izm\_id\_CB.Items[look\_res\_graph\_izm\_CB.ItemIndex])

-1].Selected:=true;

look\_res\_load\_data.Execute;

end;

//клик на графике

procedure TMainForm.look\_res\_graph\_chartClickSeries(Sender: TCustomChart;

Series: TChartSeries; ValueIndex: Integer; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

begin

//showmessage('value: '+inttostr(ValueIndex));

//showmessage('x: '+inttostr(x));

//showmessage('y: '+inttostr(y));

end;

/////////////////////////////////

/// Закладка численные значения//

////////////////////////////////

//загрузка закладки численные значения

procedure TMainForm.load\_number\_grid(izm\_num:string);

var

i: integer;

tmp1: string;

tmp2: string;

i1: integer;

begin

mysql\_query(SQL\_DM.mysql\_query6, 'SELECT '+

'(SELECT `name` FROM '+table\_points\_name+' WHERE `num`=`point\_num`) as point\_name, '+

'(SELECT `name` FROM '+table\_param\_name+' WHERE `id`=`param\_num`) as param\_name, '+

'(SELECT `id` FROM '+table\_izm\_name+' WHERE `id`=`izm\_num`) as izm\_name, '+

'znach FROM `'+table\_znacheniya\_name+

'` WHERE `izm\_num`='+izm\_num +

' ORDER BY `rd\_znacheniya`.`param\_num`, `point\_num` ASC;',1);

with look\_res\_graph\_data\_DBGrid do

begin

ColWidths[0]:=102;

ColWidths[1]:=65;

ColWidths[2]:=110;

ColWidths[3]:=80;

i:=0;i1:=0;

RowCount:=2;

SQL\_DM.mysql\_DSQ6.DataSet.First;

while not SQL\_DM.mysql\_DSQ6.DataSet.Eof do

begin

inc(i);

AddRow;

Cells[0,i+i1]:=SQL\_DM.mysql\_DSQ6.DataSet.FieldByName('point\_name').AsString;

Cells[1,i+i1]:=SQL\_DM.mysql\_DSQ6.DataSet.FieldByName('param\_name').AsString;

Cells[2,i+i1]:=SQL\_DM.mysql\_DSQ6.DataSet.FieldByName('izm\_name').AsString;

if Cells[1,i]='R' then

Cells[3,i+i1]:=floattostrf(

SQL\_DM.mysql\_DSQ6.DataSet.FieldByName('znach').AsFloat/scales[0], ffNumber,5,3)+' МОм'

else if Cells[1,i]='G0' then

Cells[3,i+i1]:=floattostrf(SQL\_DM.mysql\_DSQ6.DataSet.FieldByName('znach').AsFloat/scales[1], ffNumber,5,3)+' мкСм'

else if Cells[1,i]='G1' then

Cells[3,i+i1]:=floattostrf(SQL\_DM.mysql\_DSQ6.DataSet.FieldByName('znach').AsFloat/scales[2], ffNumber,5,3)+' мкСм'

else if Cells[1,i]='G2' then

Cells[3,i+i1]:=floattostrf(SQL\_DM.mysql\_DSQ6.DataSet.FieldByName('znach').AsFloat/scales[3], ffNumber,5,3)+' мкСм'

else if Cells[1,i]='G3' then

Cells[3,i+i1]:=floattostrf(SQL\_DM.mysql\_DSQ6.DataSet.FieldByName('znach').AsFloat/scales[4], ffNumber,4,3)+' мкСм'

else if Cells[1,i]='C' then

Cells[3,i+i1]:=floattostrf(SQL\_DM.mysql\_DSQ6.DataSet.FieldByName('znach').AsFloat/scales[5], ffNumber,4,3)+' нФ';

tmp1:=Cells[0,i+i1];

tmp2:='';

if i>1 then tmp2:=Cells[0,i-1+i1];

if (tmp1=tmp2) then

begin

MergeCells(0,i-1,1,i1+2);

MergeCells(1,i-1,1,i1+2);

MergeCells(2,i-1,1,i1+2);

dec(i);

inc(i1);

end

else

begin

i:=i+i1;

i1:=0;

end;

SQL\_DM.mysql\_DSQ6.DataSet.Next;

end;

end;

end;

procedure TMainForm.look\_res\_number\_filter\_genparam\_CBClick(Sender: TObject);

begin

if look\_res\_number\_filter\_genparam\_CB.Checked then

look\_res\_number\_genparam\_panel.Show

else look\_res\_number\_genparam\_panel.Hide;

end;

//выбор измерения

procedure TMainForm.look\_res\_number\_izm\_CBChange(Sender: TObject);

begin

look\_res\_choise\_izm.Items[

strtoint(look\_res\_number\_izm\_id\_CB.Items[look\_res\_number\_izm\_CB.ItemIndex])

-1].Selected:=true;

look\_res\_load\_data.Execute;

end;

//Служебные процедуры, таблица численных значений

procedure TMainForm.look\_res\_graph\_data\_DBGridGetAlignment(Sender: TObject;

ARow, ACol: Integer; var HAlign: TAlignment; var VAlign: TVAlignment);

begin

HAlign:=taCenter;

VAlign:=vtaBottom;

end;

/////////////////////////////////

/// Закладка диагноза //

////////////////////////////////

//Создание отчета

procedure TMainForm.look\_res\_make\_report\_btnClick(Sender: TObject);

var

XL: TExcelApplication;

WB: \_WorkBook;

//oRng: ExcelRange;

ASheet: \_WorkSheet;

const lcid=LOCALE\_USER\_DEFAULT;

begin

// запускаем новый экземпляр Excel'я

XL := TExcelApplication.Create(nil);

try

XL.ConnectKind := ckNewInstance;

XL.Connect; // подключение

XL.AutoQuit := False; // по умолчанию это свойство True только в unit ExcelXP

XL.Visible[lcid] := True;

// здесь работаем с Excel'ем

XL.DisplayAlerts [lcid]:= false;

// новый документ

WB:=XL.Workbooks.Add(EmptyParam, lcid);

ASheet := (XL.ActiveWorkbook.ActiveSheet as \_Worksheet);

ASheet.Range['A1', EmptyParam].Value2 := 'Любой текст';

finally

// отсоединяемся

XL.UserControl := True; // отдадим управление пользователю

//XL.Quit; // закрыть Excel

XL.Disconnect;

FreeAndNil(XL);

end;

end;

//Служебные процедуры таблицы результатов,

procedure TMainForm.look\_res\_diagnoz\_data\_StringGridGetAlignment(

Sender: TObject; ARow, ACol: Integer; var HAlign: TAlignment;

var VAlign: TVAlignment);

begin

if ACol in [0,3,6,9] then

if ARow in [0,1,2,6,7,11,13] then HAlign:=taCenter;

VAlign:=vtatop;

end;

procedure TMainForm.look\_res\_diagnoz\_data\_StringGridGetCellBorderProp(

Sender: TObject; ARow, ACol: Integer; LeftPen, TopPen, RightPen,

BottomPen: TPen);

begin

if (ACol=0) then LeftPen.Color:=clSilver;

if ACol in [1,4,7,10] then

begin

if ARow in [5,6,7,10,11,12] then

begin

if ARow in [6,7,11,12] then TopPen.Style:=psClear;

if ARow in [5,6,10,11] then BottomPen.Style:=psClear;

end;

end;

end;

procedure TMainForm.look\_res\_diagnoz\_data\_StringGridGetDisplText(

Sender: TObject; ACol, ARow: Integer; var Value: string);

begin

if ACol in [2,5,8,11] then

if ARow in [3,4,5,6,8,9,10,11] then Value:=diagnoz\_table\_unit;

if ACol in [0,3,6,9] then

if ARow in [6,11] then Value:=summ\_all;

end;

procedure TMainForm.look\_res\_diagnoz\_data\_StringGridGetFloatFormat(

Sender: TObject; ACol, ARow: Integer; var IsFloat: Boolean;

var FloatFormat: string);

begin

FLoatFormat:= '%.3f';

end;

procedure TMainForm.look\_res\_diagnoz\_points\_table\_switchClick(Sender: TObject);

var stream:TFileStream;

begin

look\_res\_diagnoz\_help\_StringGrid.ColCount:=9;

look\_res\_diagnoz\_help\_StringGrid.RowCount:=15;

stream:=TFileStream.Create('table2.dat', fmOpenRead);

look\_res\_diagnoz\_help\_StringGrid.LoadFromBinStream(stream);

stream.Destroy;

look\_res\_diagnoz\_help\_StringGrid.ColWidths[0]:=21;

look\_res\_diagnoz\_help\_StringGrid.ColWidths[1]:=58;

look\_res\_diagnoz\_help\_StringGrid.ColWidths[2]:=83;

look\_res\_diagnoz\_help\_StringGrid.ColWidths[3]:=73;

look\_res\_diagnoz\_help\_StringGrid.ColWidths[4]:=60;

look\_res\_diagnoz\_help\_StringGrid.ColWidths[5]:=87;

look\_res\_diagnoz\_help\_StringGrid.ColWidths[6]:=70;

look\_res\_diagnoz\_help\_StringGrid.ColWidths[7]:=105;

look\_res\_diagnoz\_help\_StringGrid.ColWidths[8]:=104;

end;

procedure TMainForm.look\_res\_diagnoz\_koeffs\_table\_switchClick(Sender: TObject);

var stream:TFileStream;

begin

look\_res\_diagnoz\_help\_StringGrid.ColCount:=4;

look\_res\_diagnoz\_help\_StringGrid.RowCount:=9;

stream:=TFileStream.Create('table3.dat', fmOpenRead);

look\_res\_diagnoz\_help\_StringGrid.LoadFromBinStream(stream);

stream.Destroy;

look\_res\_diagnoz\_help\_StringGrid.ColWidths[0]:=32;

look\_res\_diagnoz\_help\_StringGrid.ColWidths[1]:=32;

look\_res\_diagnoz\_help\_StringGrid.ColWidths[2]:=32;

look\_res\_diagnoz\_help\_StringGrid.ColWidths[3]:=422;

end;

//среднее значение сопротивления точки в измерении

function TMainForm.Get\_R(point:string; measure:string; type\_of:integer):string;

begin

//При расчетах нулевые-отрицательные-ошибочные значения выкидываются и не учитываются!!!

case type\_of of

0: mysql\_query(SQL\_DM.mysql\_query4,

'SELECT AVG(`znach`) as value FROM `'+table\_znacheniya\_name+'` WHERE '+

'(`izm\_num`='+measure+') AND '+

'(`param\_num`=(SELECT `id` FROM '+table\_param\_name+' WHERE `name`=''R'')) AND '+

'(`znach`>=0) AND '+

'(`point\_num`=(SELECT `num` FROM '+table\_points\_name+' WHERE `name`='''+point+'''))',2);

1: mysql\_query(SQL\_DM.mysql\_query4,

'SELECT `znach` as value FROM `'+table\_znacheniya\_name+'` WHERE '+

'(`izm\_num`='+measure+') AND'+

'(`param\_num`=(SELECT `id` FROM '+table\_param\_name+' WHERE `name`=''R'')) AND'+

'(`znach`>=0) AND '+

'(`povtor\_num`=1) AND '+

'(`point\_num`=(SELECT `num` FROM '+table\_points\_name+' WHERE `name`='''+point+'''))',2);

2: mysql\_query(SQL\_DM.mysql\_query4,

'SELECT MAX(`znach`) as value FROM `'+table\_znacheniya\_name+'` WHERE '+

'(`izm\_num`='+measure+') AND'+

'(`param\_num`=(SELECT `id` FROM '+table\_param\_name+' WHERE `name`=''R'')) AND'+

'(`znach`>=0) AND '+

'(`point\_num`=(SELECT `num` FROM '+table\_points\_name+' WHERE `name`='''+point+'''))',2);

3: mysql\_query(SQL\_DM.mysql\_query4,

'SELECT MIN(`znach`) as value FROM `'+table\_znacheniya\_name+'` WHERE '+

'(`izm\_num`='+measure+') AND'+

'(`param\_num`=(SELECT `id` FROM '+table\_param\_name+' WHERE `name`=''R'')) AND'+

'(`znach`>=0) AND '+

'(`point\_num`=(SELECT `num` FROM '+table\_points\_name+' WHERE `name`='''+point+'''))',2);

end;

try

result:=floattostr(SQL\_DM.mysql\_DSQ4.DataSet.FieldByName('value').AsFloat/scales[0])

except

result:='0';

end;

end;

/////////////////////////////////

/// Пользовательские процедуры //

////////////////////////////////

function TMainForm.Sign(X:Extended):Integer;

begin

if X>0 then

Result := 1

else if X<0 then

Result := -1

else

Result := 0;

end;

function TMainForm.In\_power (x:extended; n:extended):extended;

begin

result:=exp(n\*ln(x));

end;

end.

# МОДУЛЬ MultiInstUnit.pas

{ \*\*\*\* UBPFD \*\*\*\*\*\*\*\*\*\*\* by delphibase.endimus.com \*\*\*\*

>> Запуск одной копии приложения (Mutex-базированный)

Единственная функция Init\_Mutex этого модуля (only\_one.pas) создает мьютекс

с именем, переданным в параметре mid.

Возврат: true, если мьютекс создан (запущен первый экземпляр приложения)

или false, если уже имеется мьютекс с подобным именем (mid).

Особенности:

1. даже при "гибели" приложения все, относящиеся к нему мьютексы удаляются

с большой степенью вероятности.

2. Желательно "отметить" приложение в системе так, как указано в примере.

При таком подходе Ваше приложение почти со стапроцентной вероятностью

не будет запущено два раза.

Зависимости: Windows

Автор: Роман Василенко, romix@nm.ru, Пятигорск

Copyright: Роман Василенко

Дата: 14 июня 2002 г.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* }

unit MultiInstUnit;

interface

function Init\_Mutex(mid: string): boolean;

Procedure Show\_mes;

implementation

uses Windows, Dialogs;

var

mut: thandle;

function mut\_id(s: string): string;

var

f: integer;

begin

result := s;

for f := 1 to length(s) do

if result[f] = '\' then

result[f] := '\_';

end;

function Init\_Mutex(mid: string): boolean;

begin

Mut := CreateMutex(nil, false, pchar(mut\_id(mid)));

Result := not ((Mut = 0) or (GetLastError = ERROR\_ALREADY\_EXISTS));

end;

Procedure show\_mes;

begin

showmessage('Приложение уже запущено!')

end;

initialization

mut := 0;

finalization

if mut <> 0 then

CloseHandle(mut);

end.