**Приложение А**

**Текст программы**

Модуль UMain

unit Umain;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.Buttons, Vcl.Menus,

Vcl.ExtCtrls, Vcl.ComCtrls, ustr, Vcl.Grids, FireDAC.Stan.Intf,

FireDAC.Stan.Option, FireDAC.Stan.Error, FireDAC.UI.Intf, FireDAC.Phys.Intf,

FireDAC.Stan.Def, FireDAC.Stan.Pool, FireDAC.Stan.Async, FireDAC.Phys,

FireDAC.VCLUI.Wait, FireDAC.Stan.Param, FireDAC.DatS, FireDAC.DApt.Intf,

FireDAC.DApt, Data.DB, FireDAC.Comp.DataSet, FireDAC.Comp.Client, inifiles, ud,

FireDAC.Phys.Oracle, ComObj, FireDAC.Phys.OracleDef;

type

TForm1 = class(TForm)

StatusBar1: TStatusBar;

MainMenu1: TMainMenu;

Panel1: TPanel;

N1: TMenuItem;

BitBtn1: TBitBtn;

BitBtn2: TBitBtn;

BitBtn3: TBitBtn;

BitBtn4: TBitBtn;

BitBtn5: TBitBtn;

N2: TMenuItem;

N3: TMenuItem;

N4: TMenuItem;

N5: TMenuItem;

PageControl1: TPageControl;

Оплаты: TTabSheet;

TabSheet2: TTabSheet;

TabSheet3: TTabSheet;

Label1: TLabel;

Label2: TLabel;

Timer1: TTimer;

GroupBox1: TGroupBox;

GroupBox3: TGroupBox;

GroupBox4: TGroupBox;

Label3: TLabel;

Edit1: TEdit;

Button1: TButton;

OpenDialog1: TOpenDialog;

StringGrid1: TStringGrid;

BitBtn6: TBitBtn;

ProgressBar1: TProgressBar;

Label4: TLabel;

Edit2: TEdit;

CheckBox1: TCheckBox;

Label5: TLabel;

Edit3: TEdit;

Button2: TButton;

Label6: TLabel;

Edit4: TEdit;

Button3: TButton;

BitBtn7: TBitBtn;

GroupBox5: TGroupBox;

GroupBox6: TGroupBox;

GroupBox2: TGroupBox;

Memo1: TMemo;

Edit5: TEdit;

Button4: TButton;

Button7: TButton;

Button8: TButton;

Button9: TButton;

CheckBox2: TCheckBox;

TabSheet1: TTabSheet;

GroupBox7: TGroupBox;

Edit6: TEdit;

Label8: TLabel;

GroupBox8: TGroupBox;

Button11: TButton;

StringGrid3: TStringGrid;

Button12: TButton;

Splitter3: TSplitter;

BitBtn10: TBitBtn;

StringGrid2: TStringGrid;

BitBtn11: TBitBtn;

procedure Timer1Timer(Sender: TObject);

procedure FormActivate(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure BitBtn1Click(Sender: TObject);

procedure BitBtn6Click(Sender: TObject);

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

procedure Button3Click(Sender: TObject);

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

procedure TLog(txt:string; tip:integer);

procedure TLogFile(txt:string);

function LoadNBRBFile(Sf:string):integer;

function LoadString(s:string;sp:string):integer;

procedure Auto(stringgridn:tstringgrid);

function GetSQL(s:string):string;

function ExecSQL(s:string):string;

function fillstrgr(q:tfdquery; s:tstringgrid):integer;

procedure BitBtn5Click(Sender: TObject);

procedure BitBtn2Click(Sender: TObject);

procedure BitBtn4Click(Sender: TObject);

procedure BitBtn3Click(Sender: TObject);

procedure BitBtn10Click(Sender: TObject);

function MoveToArc(s:string; del:integer):integer;

procedure BitBtn11Click(Sender: TObject);

procedure Button4Click(Sender: TObject);

procedure StringGrid3DrawCell(Sender: TObject; ACol, ARow: Integer;

Rect: TRect; State: TGridDrawState);

procedure StringGrid3Click(Sender: TObject);

procedure Button7Click(Sender: TObject);

procedure Button8Click(Sender: TObject);

procedure Button9Click(Sender: TObject);

procedure FormPrintF(q:string;z:string);

procedure BitBtn7Click(Sender: TObject);

procedure N8Click(Sender: TObject);

procedure N9Click(Sender: TObject);

procedure Button11Click(Sender: TObject);

procedure Button12Click(Sender: TObject);

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

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

ferrname:string;

mode:integer;//0 - load 1 - refresh

arcpath:string;

usr:string;

implementation

{$R \*.dfm}

uses ULOgin, UFind, UUsers, UWorker, UPrint, UHelp, UPeriod;

function TForm1.GetSQL(s:string):string;

begin

with DataModule1 do

begin

qsql.Open('select \* from sql where name='+Quotedstr(s));

GetSQL:=qsql.FieldByName('SQL').Asstring;

end;

end;

function TForm1.ExecSQL(s:string):string;

begin

with DataModule1 do

begin

esql.SQL.Clear;

esql.sql.add(s);

esql.ExecSQL;

ExecSQL:='1';

end;

end;

procedure TForm1.TLogFile(txt:string);

var

s:string;

begin

s:=UpperCase(Getsql('INSFILES'));

s:=strreplace(s,':FN',Quotedstr(txt));

ExecSQL(s);

end;

procedure TForm1.TLog(txt:string; tip:integer);

var

s:string;

begin

s:=UpperCase(getSQL('INSLOG'));

s:=strreplace(s,':TEXT',Quotedstr(txt+' ('+USR+')'));

s:=strreplace(s,':TIP',inttostr(tip));

ExecSQL(s);

Memo1.Lines.Add(txt);

end;

function tform1.fillstrgr(q:tfdquery; s:tstringgrid):integer;

var

st:string;

i,j:integer;

begin

s.Color:=clRed;

s.Repaint;

s.ColCount:=q.FieldCount;

s.RowCount:=2;

i:=1;

for j:=0 to s.ColCount-1 do

s.Cells[j,1]:='';

q.First;

for j:=0 to q.FieldCount-1 do

s.Cells[j,0]:=q.Fields.Fields[j].FieldName;

while (not q.eof) do

begin

for j:=0 to q.FieldCount-1 do

s.Cells[j,i]:=q.Fields.Fields[j].AsString;

q.Next;

inc(i);

Application.ProcessMessages;

s.RowCount:=s.RowCount+1;

end;

if i>1 then

s.RowCount:=s.RowCount-1;

q.Close;

result:= i-1;

auto(s);

s.Repaint;

end;

function Tform1.MoveToArc(s:string; del:integer):integer;

begin // moving file into archive folder

MoveToArc:=Filemove(s,edit4.Text+extractfilename(s),del);// 1 mean move, 0 - copy

end;

procedure TForm1.N8Click(Sender: TObject);

begin

Form7.ShowModal;

end;

procedure TForm1.N9Click(Sender: TObject);

begin

//Button5.Click;

end;

procedure TForm1.StringGrid3Click(Sender: TObject);

begin

Form5.Label2.Caption:=Stringgrid3.Cells[0, Stringgrid3.Row];

Form5.Edit2.Text:=Stringgrid3.Cells[1, Stringgrid3.Row];

Form5.Edit1.Text:=Stringgrid3.Cells[3, Stringgrid3.Row];

Form5.Label6.Caption:=Stringgrid3.Cells[2, Stringgrid3.Row];

Form5.CheckBox1.Checked:=false;

end;

procedure TForm1.FormPrintF(q:string;z:string);

var

s:string;

i,row:integer;

begin

Form6.Caption:=('Печать отчета '+z);

s:=z;

delete(s,1,2);

Form6.Label1.Caption:=s;

With DataModule1 do

With Form6 do

begin

TSQL.Close;

TSQL.SQL.Clear;

TSQL.Open(q);

Fillstrgr(TSQL,Form6.StringgridP);

end;

if Form6.ShowModal=mrOk then

begin

end;

end;

procedure TForm1.StringGrid3DrawCell(Sender: TObject; ACol, ARow: Integer;

Rect: TRect; State: TGridDrawState);

begin

//Если ячейка получает фокус, то нам надо закрасить её другими цветами

if (gdFocused in State)or (gdSelected in State) then

begin

StringGrid3.Canvas.Brush.Color := clBlue;

StringGrid3.Canvas.Font.Color := clWhite;

end

else

begin

StringGrid3.canvas.brush.Color := clWhite;

try

if Incmonth(strtodate(StringGrid3.Cells[acol,arow]),11) <= date then

StringGrid3.Canvas.Brush.color := clYellow;// последний месяц

if Incmonth(strtodate(StringGrid3.Cells[acol,arow]),12) <= date then

StringGrid3.Canvas.Brush.color := clRed ;// больше года

except

on e:exception do e:=nil;

end;

end;

//Теперь закрасим ячейки, но только, если ячейка не Title- Row/Column

// это завит от того, есть у нас title-Row/Columns или нет.

if (ACol > 0) and (ARow > 0) then

begin

//Закрашиваем бэкграунд

StringGrid3.Canvas.FillRect(Rect);

//Закрашиваем текст (Text). Также здесь можно добавить выравнивание и т.д..

StringGrid3.Canvas.TextOut(Rect.Left, Rect.Top, StringGrid3.Cells[ACol, ARow]);

end;

end;

function TForm1.LoadString(s:string;sp:string):integer;

var

sq:string;

begin

// load string from file returning resultcode=1 if OK

// SELECT \* FROM PLAT where FIO=:fio and ADR=:adr

// and DP=:dp and SUMM=summ and nplat=:nplat

Datamodule1.TSQL.sql.Clear;

Datamodule1.TSQL.sql.Add(getsql('GETSTR'));

Datamodule1.TSQL.parambyname('nplat').AsString:=sp;

Datamodule1.TSQL.parambyname('fio').AsString:=mypos(s,'^',3);

Datamodule1.TSQL.parambyname('adr').AsString:=mypos(s,'^',4);

sq:=mypos(s,'^',10);

Datamodule1.TSQL.parambyname('dp').AsString:=copy(sq,7,2)+'.'+copy(sq,5,2)+'.'+copy(sq,1,4);

Datamodule1.TSQL.parambyname('summ').AsString:=strreplace(mypos(s,'^',7),'.',',');

Datamodule1.TSQL.Open();

if Datamodule1.TSQL.RecordCount=0 then

begin

{INSERT INTO GIGI.PLAT (FIO, ADR,DP, SUMM, SUMMP,NPLAT)

VALUES (:FIO,:ADR,:DP,:SUMM,:SUMMP,:NPLAT);}

Datamodule1.TSQL.sql.Clear;

Datamodule1.TSQL.sql.Add(getsql('INSSTR'));

Datamodule1.TSQL.parambyname('NPLAT').AsString:=sp;

Datamodule1.TSQL.parambyname('FIO').AsString:=mypos(s,'^',3);

Datamodule1.TSQL.parambyname('ADR').AsString:=mypos(s,'^',4);

Datamodule1.TSQL.parambyname('SUMM').AsString:=strreplace(mypos(s,'^',7),'.',',');

Datamodule1.TSQL.parambyname('SUMMP').AsString:=strreplace(mypos(s,'^',9),'.',',');

Datamodule1.TSQL.parambyname('dp').AsString:=copy(sq,7,2)+'.'+copy(sq,5,2)+'.'+copy(sq,1,4);

Datamodule1.TSQL.prepare;

Datamodule1.TSQL.ExecSQL;

LoadString := Datamodule1.TSQL.RowsAffected;

end

else

begin

LoadString := 1000;

end;

end;

function TForm1.LoadNBRBFile(Sf:string):integer;///////////////////////////////////////////////

var

errc,er:integer;

fi:textfile;

fo:textfile;

s,s1,snp:string;

begin // loading one file from NBRB

errc:=0;

ferrname:=sf+'.err';

assignfile(fi,sf);

reset(fi);// open for read

ReadLn(fi,s);

snp:=mypos(s,'^',3);

TLog('=Старт загрузки - '+sf,1);

if CheckBox1.Checked then // checking for first string

begin

if pos(edit2.Text,s)<>1 then

begin

errc:=errc+1;

TLog('-Ошибка первой строки!',1);

end;

end;

if errc=0 then

While not(eof(fi)) do // loading every string

begin

ReadLn(fi,s);

// parse and load this string

er:=LoadString(s,snp);

if er<>1 then

begin

if er=1000 then TLog('-Строка уже была загружена ранее',1)

else TLog('-Ошибка загрузки - '+inttostr(er),1);

errc:=errc+1;

end

else

begin

TLog('+Строка загружена!',1);

end;

end;

CloseFile(fi); // and closing file

LoadNBRBFile:=errc;// returning result

TLog('= Ошибок - '+inttostr(errc),1);

if errc=0 then

TLogFile(sf);

TLog('= Загрузка завершена',1);

ferrname:='';

end;

procedure TForm1.Auto(stringgridn:tstringgrid);

var

x, y, w: integer;

MaxWidth: integer;

begin

with StringGridn do

begin

for x := 0 to ColCount - 1 do

begin

MaxWidth := 0;

for y := 0 to RowCount - 1 do

begin

w := Canvas.TextWidth(Cells[x,y]);

if w > MaxWidth then

MaxWidth := w;

end;

ColWidths[x] := MaxWidth + 8;

end;

end;

end;

procedure TForm1.BitBtn10Click(Sender: TObject);

var

qs:string;

begin

with datamodule1 do

begin

TSQL.SQL.Clear;

tsql.SQL.Add(GetSQL('FINDPLAT'));

tsql.ParamByName('fstr').AsString:='%'+Edit6.text+'%';

tsql.Open();

fillstrgr(tsql,StringGrid2);

end;

end;

procedure TForm1.BitBtn11Click(Sender: TObject);

begin

With Datamodule1 do

begin

TSQL.Close;

TSQL.SQL.Clear;

TSQL.Open(Getsql('GETWORKER')+' order by dp');

Fillstrgr(TSQL,Stringgrid3);

end;

end;

procedure TForm1.BitBtn1Click(Sender: TObject);

var

i:integer;

begin

OpenDialog1.InitialDir:=Edit1.Text;

if OpenDialog1.Execute then

begin

for i:=0 to StringGrid1.RowCount-1 do Stringgrid1.Rows[i].Clear;

BitBtn6.Hint:='Загрузить выбранные файлы';

//Bitbtn6.Glyph:=Bitbtn8.Glyph;

mode:=0;

Stringgrid1.Rowcount:=OpenDialog1.Files.Count;

Stringgrid1.Cols[1]:=OpenDialog1.Files;

for i:=0 to Stringgrid1.Rowcount-1 do

begin

StringGrid1.Cells[0,i]:='0';

end;

Auto(Stringgrid1);

end;

PageControl1.ActivePageIndex:=0;

Statusbar1.Panels[0].Text:='Выбрано файлов: '+inttostr(OpenDialog1.Files.Count);

end;

procedure TForm1.BitBtn2Click(Sender: TObject);

begin

Form3.ShowModal;//Ufind

end;

procedure TForm1.BitBtn3Click(Sender: TObject);

var

i:integer;

begin

if Pagecontrol1.ActivePageIndex=3 then// вывод платежей

begin

Form6.StringGridP.RowCount:= Form1.StringGrid2.RowCount;

Form6.StringGridP.ColCount:= Form1.StringGrid2.ColCount;

for i:=0 to Form1.StringGrid2.RowCount-1 do

Form6.StringGridP.rows[i]:=Form1.StringGrid2.Rows[i];

Auto(Form6.StringGridP);

Form6.Label1.Caption:='Список платежей';

end;

Form6.ShowModal;//UPrint

end;

procedure TForm1.BitBtn4Click(Sender: TObject);

begin

Form5.ShowModal; //uWorker

end;

procedure TForm1.BitBtn5Click(Sender: TObject);

begin

Form4.ShowModal;// uUsers

end;

procedure TForm1.BitBtn6Click(Sender: TObject);

var

i,errc:integer;

sq:string;

begin

if mode=0 then

begin

Progressbar1.Max:=Stringgrid1.RowCount-1;

ProgressBar1.Min:=0;

ProgressBar1.Position:=0;

errc:=0;

for i:=0 to Stringgrid1.RowCount-1 do

begin

if LoadNBRBFile(Stringgrid1.Cells[1,i])=0 then

begin

Stringgrid1.Cells[0,i]:='Загружен';

TLog('= Файл перемещён в архив ('+Inttostr(MoveToArc(Stringgrid1.Cells[1,i],1))+')',1);

end

else

begin

Stringgrid1.Cells[0,i]:='Есть ошибки';

errc:=errc+1;

end;

ProgressBar1.Position:=i;

end;

If errc>0 then

begin

ShowMessage('Обнаружены ошибки при загрузке');

end

else

begin

ShowMessage('Все файлы загружены успешно');

end;

mode:=1;

Auto(Stringgrid1);

end;

if mode=1 then

begin

// Выводим список платежей

for i:=0 to StringGrid1.RowCount-1 do Stringgrid1.Rows[i].Clear;

BitBtn6.Hint:='Обновить данные таблицы';

//Bitbtn6.Glyph:=Bitbtn9.Glyph;

datamodule1.TSQL.Open(getsql('GETALLOPL'));

fillstrgr(datamodule1.TSQL,stringgrid1);

end;

end;

procedure TForm1.BitBtn7Click(Sender: TObject);

begin

Form7.ShowModal;//UHelp

end;

procedure TForm1.Button11Click(Sender: TObject);

begin

with datamodule1 do

begin

TSQL.SQL.Clear;

tsql.SQL.Add(GetSQL('FINDPLAT'));

tsql.ParamByName('fstr').AsString:='%';

tsql.Open();

fillstrgr(tsql,StringGrid2);

end;

end;

procedure TForm1.Button12Click(Sender: TObject);

begin

if Form8.Showmodal=mrOk then

begin

With DataModule1 do

begin

TSQL.Close;

TSQL.SQL.Clear;

TSQL.SQL.Add(GetSQL('GETOPLPER'));

TSQL.ParamByName('f').AsDate:=Form8.DateTimePicker1.Date;

TSQL.ParamByName('s').AsDate:=Form8.DateTimePicker2.Date;

TSQL.Open();

FillSTrGr(TSQL,StringGrid2);

end;

end;

end;

procedure TForm1.Button2Click(Sender: TObject);

begin

if OpenDialog1.Execute then

begin

Edit3.Text:=ExtractFilePath(OpenDialog1.FileName);

end;

end;

procedure TForm1.Button3Click(Sender: TObject);

begin

if OpenDialog1.Execute then

begin

Edit4.Text:=ExtractFilePath(OpenDialog1.FileName);

end;

end;

procedure TForm1.Button4Click(Sender: TObject);

begin

With Datamodule1 do

begin

TSQL.Close;

TSQL.SQL.Clear;

TSQL.SQL.Add(Getsql('GETWORKER'));

TSQL.SQL.Add(' where fio like '+Quotedstr('%'+Edit5.text+'%')+' order by dp');

TSQL.Open();

Fillstrgr(TSQL,Stringgrid3);

end;

end;

procedure TForm1.Button7Click(Sender: TObject);

begin

Form5.Label2.Caption:=Stringgrid3.Cells[0, Stringgrid3.Row];

Form5.Edit2.Text:=Stringgrid3.Cells[1, Stringgrid3.Row];

Form5.Edit1.Text:=Stringgrid3.Cells[3, Stringgrid3.Row];

Form5.Label6.Caption:=Stringgrid3.Cells[2, Stringgrid3.Row];

Form5.CheckBox1.Checked:=false;

if Form5.ShowModal=mrOk then

begin

//UPDATE WORKER SET FIO=:FIO, DOP=:DOP, DP=TO\_DATE(:DP,'DD.MM.YYYY') WHERE ID=:ID

With Datamodule1 do

begin

TSQL.SQL.Clear;

TSQL.SQL.Add(GEtSQL('UPDWORKER'));

TSQL.ParamByName('FIO').AsString:=Form5.Edit2.Text;

TSQL.ParamByName('DOP').AsString:=Form5.Edit1.Text;

if Form5.CheckBox1.Checked then

TSQL.ParamByName('DP').AsString:=DateToStr(Form5.MonthCalendar1.Date)

else TSQL.ParamByName('DP').AsString:=Form5.Label6.Caption;

TSQL.ParamByName('ID').AsString:=Form5.Label2.Caption;

TSQL.ExecSQL;

end;

end;

BitBtn11.Click;

end;

procedure TForm1.Button8Click(Sender: TObject);

begin

FormPrintF(GetSql('GETYEAR'),': Просроченные профосмотры');

end;

procedure TForm1.Button9Click(Sender: TObject);

begin

FormPrintF(GetSql('GETMONTH'),': Последний месяц профосмотра');

end;

procedure TForm1.FormActivate(Sender: TObject);

var

f:TIniFile;

s:string;

begin

Timer1Timer(Form1 as TObject);

f:=Tinifile.Create(extractfilepath(application.exename)+'/settings.ini');

s:=edit1.Text;

Edit1.Text:=f.ReadString('MAIN', 'INPUT FOLDER', s);

createdirex(edit1.Text);

s:=edit2.Text;

Edit2.Text:=f.ReadString('MAIN', 'FIRST PATTERN', s);

s:=edit3.Text;

Edit3.Text:=f.ReadString('MAIN', 'OUTPUT FOLDER', s);

createdirex(edit3.Text);

s:=edit4.Text;

Edit4.Text:=f.ReadString('MAIN', 'BACKUP FOLDER', s);

arcpath:=Edit4.Text+datetostr(date)+'\';

createdirex(arcpath);

f.Free;

PageControl1.ActivePageIndex:=0;

TLog('Запуск программы',0);

if Form2.showmodal=mrOk then

begin

TLog('Пользователь: '+Form2.Edit1.text,0);

with datamodule1 do

begin

tsql.sql.clear;

tsql.SQL.Add('select activ from usr where name='+quotedstr(Form2.Edit1.text)+

' and pass='+quotedstr(Form2.Edit2.text)+' and activ=1');

tsql.Open();

if tsql.FieldByName('activ').AsInteger<>1 then

begin

ShowMessage('В доступе отказано');

TLog('Отказ, пользователь: '+Form2.Edit1.text,0);

Application.Terminate;

end

else

begin

TLog('Вход, пользователь: '+Form2.Edit1.text,0);

Usr:= Form2.Edit1.text;

end;

end;

end

else Application.Terminate;

mode:=1;

BitBtn6.Hint:='Обновить данные таблицы';

//Bitbtn6.Glyph:=Bitbtn9.Glyph;

BitBtn6.Click;

BitBtn11.Click;

Form5.MonthCalendar1.Date:=date;

end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);

var

f:TIniFile;

s:string;

begin

f:=Tinifile.Create(extractfilepath(application.exename)+'/settings.ini');

f.WriteString('MAIN', 'INPUT FOLDER', Edit1.Text);

f.WriteString('MAIN', 'FIRST PATTERN', Edit2.Text);

f.WriteString('MAIN', 'OUTPUT FOLDER', Edit3.Text);

f.WriteString('MAIN', 'BACKUP FOLDER', Edit4.Text);

f.Free;

Tlog('Завершение работы',0);

end;

procedure TForm1.Timer1Timer(Sender: TObject);

begin

Label1.Caption:=datetostr(Date);

Label2.Caption:=TimeToStr(Time);

end;

end.

Модуль UHelp

unit UHelp;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.Buttons, Vcl.OleCtrls,

SHDocVw, Vcl.Grids;

type

TForm7 = class(TForm)

GroupBox1: TGroupBox;

GroupBox2: TGroupBox;

WebBrowser1: TWebBrowser;

BitBtn1: TBitBtn;

StringGrid1: TStringGrid;

procedure FormActivate(Sender: TObject);

Function GetHelp(s:string):string;

procedure StringGrid1Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form7: TForm7;

implementation

uses Umain,ud;

{$R \*.dfm}

Function TForm7.GetHelp(s:string):string;

var

qs:string;

begin

qs:=Form1.GetSQL('GETHELP');

With DataModule1 do

begin

TSQL.Close;

TSQL.SQL.Clear;

TSQL.SQL.add(qs);

TSQL.ParamByName('name').AsString:=s;

TSQL.Open();

Gethelp:=TSQL.fieldbyname('res').AsString;

end;

end;

procedure TForm7.FormActivate(Sender: TObject);

var

qs:string;

begin

qs:=Form1.GetSQL('GETALLHELP');

With DataModule1 do

begin

TSQL.Close;

TSQL.SQL.Clear;

TSQL.Open(qs);

Form1.fillstrgr(TSQL,Stringgrid1);

end;

WebBrowser1.Navigate(extractfilepath(application.exename)+'help\'+GetHelp('Содержание'));

end;

procedure TForm7.StringGrid1Click(Sender: TObject);

var

qs:string;

begin

//Загрузим нужный раздел в браузер из папки help

// из таблицы заполняем левый бокс

// по клику выбираем из таблицы имя файла

// и грузим его в браузер

//

WebBrowser1.Navigate(extractfilepath(application.exename)+'help\'+GetHelp(Stringgrid1.Cells[0,Stringgrid1.row]));

end;

end.

Модуль UPrint

unit UPrint;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.Buttons,ComObj,system.UITypes, Vcl.Grids;

type

TForm6 = class(TForm)

StringGridP: TStringGrid;

BitBtn1: TBitBtn;

BitBtn2: TBitBtn;

Label1: TLabel;

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

procedure BitBtn1Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form6: TForm6;

implementation

{$R \*.dfm}

procedure TForm6.BitBtn1Click(Sender: TObject);

var

Ap : Variant;

fn:string;

top,left,i,j:integer;

begin

top:=3;

left:=1;

try

Ap := GetActiveOleObject('Excel.Application');

except

on e:exception do

begin

Ap := CreateOleObject('Excel.Application');

e:=nil;

end;

end;

fn:=extractfilepath(paramstr(0))+'Template.xlsx';

Ap.Workbooks.Open(fn);

Ap.ActiveWorkbook.Sheets.Item[1].Range[Ap.Cells[top+1,left+1],Ap.Cells[top+StringGridP.RowCount,left+StringGridP.ColCount]].Borders.Weight := 2;

//Ap.ActiveWorkbook.Sheets.Item[1].Range[Ap.Cells[top+1+2,left+1+6],Ap.Cells[top+StringGrid7.RowCount,left+StringGrid7.ColCount]].Orientation:=90;

//Ap.ActiveWorkbook.Sheets.Item[1].Range[Ap.Cells[top+1+2,left+1+6],Ap.Cells[top+StringGrid7.RowCount,left+StringGrid7.ColCount]].RowHeight:=65;

//Ap.ActiveWorkbook.Sheets.Item[1].Range[Ap.Cells[top+1,left+1],Ap.Cells[top+StringGrid7.RowCount,left+StringGrid7.ColCount]].VerticalAlignment:=xlVAlignCenter;

//for j:=0 to StringGrid7.ColCount-1 do

// if j+1+left>6 then

// begin

// Ap.ActiveWorkbook.Sheets.Item[1].Columns[j+1+left].ColumnWidth:=2.2;//10;

// end;

//Ap.Cells[1,left+StringGrid7.ColCount-7] := 'Утверждаю';

//Ap.Cells[2,left+StringGrid7.ColCount-7] := 'Главный врач';

//Ap.Cells[3,left+StringGrid7.ColCount-7] := 'УЗ «Быховский райЦГЭ»';

Ap.Cells[2,2]:=Label1.Caption;

for i:=0 to Stringgridp.ColCount-1 do

begin

for j := 0 to Stringgridp.RowCount-1 do

begin

Ap.Cells[j+top+1,i+left+1]:=Stringgridp.cells[i,j];

end;

end;

Ap.Visible := True;

Ap.ActiveWorkBook.SaveAs(fn+'\_TMP.XLSX');

end;

procedure TForm6.FormClose(Sender: TObject; var Action: TCloseAction);

begin

Form6.Caption:=('Печать отчета');

end;

end.

Модуль Ustr

unit Ustr;

interface

uses

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

Dialogs, StdCtrls, // DBTables,

comctrls, ShellAPI, ShlObj, grids;

// , SQLiteTable3;

// ==============================================================================

const

cpWin = 01;

cpAlt = 02;

cpKoi = 03;

AltSet = ['А' .. 'Я', 'а' .. 'п', 'р' .. 'я'];

KoiSet = ['Б' .. 'Р', 'Т' .. 'С'];

WinSet = ['а' .. 'п', 'р' .. #255];

type

ArrOfStr = array of string;

MyRec = record

serkod: integer;

summa: real;

end;

A = array [byte] of MyRec;

var

MaskString: string;

dates: array [1 .. 12] of integer = (

31,

28,

31,

30,

31,

30,

31,

31,

30,

31,

30,

31

);

lat: array [1 .. 26] of char = (

'A',

'B',

'C',

'D',

'E',

'F',

'G',

'H',

'I',

'J',

'K',

'L',

'M',

'N',

'O',

'P',

'Q',

'R',

'S',

'T',

'U',

'V',

'W',

'X',

'Y',

'Z'

);

rus: array [1 .. 26] of char = (

'А',

'В',

'С',

'D',

'Е',

'F',

'G',

'Н',

'I',

'J',

'К',

'L',

'М',

'N',

'О',

'Р',

'Q',

'R',

'S',

'Т',

'U',

'V',

'W',

'Х',

'Y',

'Z'

);

russet: set of char = ['А', 'В', 'С', 'Е', 'Н', 'К', 'М', 'О', 'Р', 'Т', 'Х'];

Bufsize: longint;

tsql: string;

// ==============================================================================

procedure Progress2status(p: tprogressbar; s: tstatusbar);

// ======== SQLite3

// procedure fillstringgrid(sldb:TSQLiteDatabase; stringgrid1:tstringgrid; ssql:string; f:integer);

// procedure fillcombobox (sldb:TSQLiteDatabase; combobox1:tcombobox; ssql:string);

// function savestringgrid(sldb:TSQLiteDatabase; stringgrid : tstringgrid; tbname:string):string;

// function instringrec(sldb:TSQLiteDatabase; stringgrid : tstringgrid; tbname:string; n:longint):integer;

// function createstringgridtable(sldb:tsqlitedatabase; stringgrid : tstringgrid; tbname:string):string;

// ======== SQLite3

function parsehome(hin: string; var h\_num, h\_let, h\_drob,

h\_korp: string): boolean;

function parseflat(fin: string; var f\_num: string; var f\_let: string): boolean;

procedure address(s: string; var street: string; var home: string;

var flat: string);

function explode(sPart, sInput: string): ArrOfStr;

function implode(sPart: string; arrInp: ArrOfStr): string;

procedure sort(arrInp: ArrOfStr);

procedure rsort(arrInp: ArrOfStr);

// str - исходная строка

// str1 - подстрока, подлежащая замене

// str2 - заменяющая строка

function nvl(s: string): string;

function StrReplace(const Str, Str1, Str2: string): string;

function pnext(del: string; var s: string): string;

function pnexte(del: string; var s: string): string;

Function lTrim(var s: string): string;

Function Trims(s: string): string;

Function Count\_pos(s: string): integer;

function Mypos(s: string; d: string; p: integer): string;

function MyposReplace(sin: string; d: string; p: integer; sub: string): string;

function DetermineCodepage(const st: string): byte;

function DosToWin(st: string): string;

function WinToDos(st: string): string;

function CreateDirEx(Dir: string): boolean;

Procedure ListFileDir(Path: string; FileList: TStrings; mask: string);

function GetDate(f: char; dt: tdatetime): string;

Function parsedatetimeget(s: string; d: tdatetime): string;

Function parsedatetime(s: string): string;

Function ParseOracleDate(s: string): string;

Function ParseMySQLDate(s: string): string;

function ConvertBankDateToDate(s: string): string;

// function ConvertDateToBankDate(s:string):string;

function strmonth(d: tdatetime; caps: boolean): string;

function invalid(s: string): boolean;

function validatedate(s: string): boolean;

function testnum(s: string): boolean;

procedure nop;

procedure pause(t: int64);

function CopyFileMy(FromPath, ToPath: string): integer;

function CopyFileProgress(FromPath, ToPath: string; p: tprogressbar): integer;

Function FileMove(fs1, fs2: string; del: integer): integer;

procedure StartProg(handle: thandle; prog, param: string);

procedure StartCMD(handle: thandle; param: string);

function ExecAndWait(aCmd: string; WaitTimeOut: cardinal = INFINITE): cardinal;

procedure Copyfilewin(handle: thandle; from, tof: string);

function FExist(s: string): boolean;

Function OpenFolder(form1: tform): string;

function Getfiles(Memo1: TMemo; ext: string): boolean;

function procent(org: double; pr: double): double;

function percent(A, b: double): double;

// Function proportional(summa:string; summap:string; q1:tquery; q2:tquery):A;

// ==============================================================================

implementation

// uses Unit1;

function nvl(s: string): string;

begin

Result := s;

if s = '' then

Result := ' ';

end;

// ========================== DB SQLite 3 ========================

{ procedure fillstringgrid(sldb:TSQLiteDatabase; stringgrid1:tstringgrid; ssql:string;f:integer);

var

i,j:cardinal;

sltb:TSQLIteTable;

begin

sltb:=sldb.gettable(ssql);

// ============ do not likes empty table

if sltb.Count=0 then exit;

// ======================================

stringgrid1.ColCount:=sltb.ColCount;

stringgrid1.RowCount:=sltb.RowCount+1;

stringgrid1.repaint;

// fill headers

if f=1 then

for i:=0 to sltb.colcount-1 do

begin

stringgrid1.Cells[i,0]:='('+inttostr(i)+')-';

stringgrid1.Cells[i,0]:=stringgrid1.Cells[i,0] + utf8decode(sltb.columns[i]);

end;

// -- END -- fill headers

j:=0;

while not sltb.EOF do

begin

for i:=0 to sltb.colcount-1 do

begin

stringgrid1.Cells[i,j+1]:=utf8decode(sltb.FieldAsString(i));

end;

sltb.Next;

inc(j);

end;

stringgrid1.repaint;

sltb.Free;

end;

procedure fillcombobox(sldb:TSQLiteDatabase; combobox1:tcombobox; ssql:string);

var

sltb:tsqlitetable;

begin

sltb:=sldb.gettable(ssql);

while not sltb.EOF do

begin

combobox1.items.add(utf8decode(sltb.FieldAsString(0)));

sltb.Next;

end;

combobox1.text:='';

sltb.Free;

end;

function instringrec(sldb:TSQLiteDatabase; stringgrid : tstringgrid; tbname:string; n:longint):integer;

// insert record in table from position

var

i:longint;

s:string;

begin

tsql:='insert into '+tbname+' (';

for i:=0 to stringgrid.ColCount-1 do

begin

s:=stringgrid.Cells[i,0];

delete(s,1,pos('-',s));

s:=utf8encode(s);

if i= stringgrid.ColCount-1 then

tsql:=tsql+s+') values ("'

else tsql:=tsql+S+', ';

end;

for i:=0 to stringgrid.ColCount-1 do

begin

s:=stringgrid.Cells[i,n];

s:=utf8encode(s);

if i= stringgrid.ColCount-1 then

tsql:=tsql+s+'");'

else tsql:=tsql+s+'", "';

end;

sldb.ExecSQL(tsql);

instringrec:=0;

end;

function savestringgrid(sldb:TSQLiteDatabase; stringgrid : tstringgrid; tbname:string):string;

var

i:longint;

// s:string;

begin

sldb.BeginTransaction;

tsql:='drop table if exists '+tbname+'\_old;';

sldb.ExecSQL(tsql);

tsql:='alter table '+tbname+' rename to '+ tbname+'\_old;';

sldb.ExecSQL(tsql);

createstringgridtable(sldb,stringgrid,'worker');

//

for i:=1 to stringgrid.RowCount-1 do

instringrec(sldb,stringgrid,tbname,i);

//

// drop database

tsql:='drop table if exists '+ tbname+'\_old;';

sldb.ExecSQL(tsql);

sldb.Commit;

end;

function createstringgridtable(sldb:tsqlitedatabase; stringgrid : tstringgrid; tbname:string):string;

var

i:integer;

s:string;

//sldb: tsqlitedatabase;

begin

//sldb.BeginTransaction;

tsql:='create table '+tbname +'(id integer primary key ,';

for i:=1 to stringgrid.ColCount-1 do

begin

s:=stringgrid.Cells[i,0];

delete(s,1,pos('-',s));

if i= stringgrid.ColCount-1 then

tsql:=tsql+s+' varchar(255)'

else tsql:=tsql+s+' varchar(255), ';

end;

tsql:=tsql+');';

sldb.ExecSQL(tsql);

//sldb.Commit;

end;

}

// ========================

function GetDate(f: char; dt: tdatetime): string;

var

s: string;

begin

s := datetostr(dt);

case f of

'd', 'D':

begin

delete(s, 3, length(s));

end;

'm', 'M':

begin

delete(s, 1, 3);

delete(s, 3, length(s));

end;

'y', 'Y':

begin

delete(s, 1, 6);

end;

end; // case

GetDate := s;

end;

function Getfiles(Memo1: TMemo; ext: string): boolean;

var

tgd: TOpenDialog;

res: boolean;

begin

tgd := TOpenDialog.Create(Memo1);

if ext <> '' then

tgd.Filter := ext;

tgd.Options := [ofHideReadOnly, ofAllowMultiSelect, ofEnableSizing];

res := tgd.execute;

if res then

begin

Memo1.Lines := tgd.Files;

end;

tgd.Free;

Getfiles := res;

end;

function testnum(s: string): boolean;

var

r: double;

begin

Result := true;

try

r := 0 + strtofloat(s);

except

on EConvertError do

begin

Result := false;

exit;

end;

end;

r := r + 0;

end;

function validatedate(s: string): boolean;

var

day, month, year: integer;

begin

Result := true;

dates[2] := 28;

day := strtoint(pnext('.', s));

month := strtoint(pnext('.', s));

year := strtoint(s);

if (day < 1) or (month < 1) or (year < 1) then

begin

Result := false;

exit;

end;

if (year mod 4 = 0) then

dates[2] := 29;

if day > dates[month] then

Result := false;

end;

{ /\*Function proportional(summa:string; summap:string; q1:tquery; q2:tquery):A;

var

i:byte;

m:A;

s:real;

begin

//Q1.Open;

//Q2.Open;

Q2.First;

m[0].serkod:=0;

s:=0;

while not Q2.Eof do

begin

inc(m[0].serkod);

m[m[0].serkod].serkod:=Q2.FieldByName('serkod').AsInteger;

m[m[0].serkod].summa:=round(Q2.FieldByName('summa').AsFloat/Q1.fieldByName('summa').AsFloat\*strtofloat(summa));

m[m[0].serkod].summa:=m[m[0].serkod].summa\*(-1);

s:=s-m[m[0].serkod].summa;

Q2.Next;

end;

Q2.First;

if summap<>'0' then

m[1].summa:=m[1].summa-strtofloat(summap);

m[1].summa:=round(m[1].summa-strtofloat(summa)+s);

m[0].summa:=s;

proportional:=m;

//Q1.Close;

//Q2.Close;

end;

\*/ }

function parseflat(fin: string; var f\_num: string; var f\_let: string): boolean;

var

i: integer;

s: string;

res: boolean;

cod: integer;

c: char;

begin

parseflat := false;

//

// for i:=1 to length(fin) do

// if (fin[i] <'1') or (fin[i]>'0') then

// res:=true;

//

val(fin, i, cod);

if cod <> 0 then

begin

f\_num := copy(fin, 1, cod - 1);

f\_let := copy(fin, cod, 10);

c := f\_let[1];

f\_let := inttostr(ord(c) - ord('А') + 1);

res := true;

end

else

begin

f\_num := fin;

f\_let := '0';

res := false;

end;

parseflat := res;

end;

function parsehome(hin: string; var h\_num, h\_let, h\_drob,

h\_korp: string): boolean;

var

s, ss, sss, ssss: string;

cod, i: integer;

begin

val(hin, i, cod);

if cod = 0 then

begin

h\_num := hin;

h\_let := '0';

h\_korp := '0';

h\_drob := '0';

parsehome := false;

end

else

begin // тут мы если буква или дробь

if pos('/', hin) <> 0 then // есть дробь

begin

s := copy(hin, 1, pos('/', hin) - 1);

ss := copy(hin, pos('/', hin) + 1, 10);

parseflat(s, h\_num, h\_let);

h\_drob := ss;

h\_korp := '0';

parsehome := true;

end

else

begin // может быть буква

parseflat(hin, h\_num, h\_let);

h\_drob := '0';

h\_korp := '0';

parsehome := true;

end;

parsehome := true;

end;

end;

Function Count\_pos(s: string): integer;

var

i, j: integer;

begin

j := 0;

for i := 1 to length(s) do

if s[i] = '|' then

inc(j);

Count\_pos := j;

end;

Function OpenFolder(form1: tform): string;

var

TitleName: string;

lpItemID: PItemIDList;

BrowseInfo: TBrowseInfo;

DisplayName: array [0 .. MAX\_PATH] of char;

TempPath: array [0 .. MAX\_PATH] of char;

begin

FillChar(BrowseInfo, sizeof(TBrowseInfo), #0);

BrowseInfo.hwndOwner := form1.handle;

BrowseInfo.pszDisplayName := @DisplayName;

TitleName := 'Выберите папку резервной копии';

BrowseInfo.lpszTitle := PChar(TitleName);

BrowseInfo.ulFlags := BIF\_RETURNONLYFSDIRS;

lpItemID := SHBrowseForFolder(BrowseInfo);

if lpItemID <> nil then

begin

SHGetPathFromIDList(lpItemID, TempPath);

// ShowMessage(TempPath);

GlobalFreePtr(lpItemID);

end;

OpenFolder := TempPath;

end;

function percent(A, b: double): double;

begin

percent := A / b;

end;

function invalid(s: string): boolean;

var

ts, rs: string;

// y,m,d:word;

cod: integer;

p: word;

begin

invalid := false;

// try

ts := pnext('.', s);

val(ts, p, cod);

if cod <> 0 then

begin

invalid := true;

exit;

end;

if (strtoint(ts) < 1) or (strtoint(ts) > 31) then

invalid := true;

rs := pnext('.', s);

val(rs, p, cod);

if cod <> 0 then

begin

invalid := true;

exit;

end;

if (strtoint(rs) < 1) or (strtoint(rs) > 12) then

invalid := true;

val(s, p, cod);

if cod <> 0 then

begin

invalid := true;

exit;

end;

if (strtoint(s) < 1900) or (strtoint(s) > 2050) then

invalid := true;

end;

function procent(org: double; pr: double): double;

var

r: double;

begin

procent := org / 100 \* pr;

end;

function FExist(s: string): boolean;

var

f: file;

begin

{$I-}

Assignfile(f, s);

Reset(f);

FExist := IoResult = 0;

{$I+}

end;

Function lTrim(var s: string): string;

begin

if s[1] <> ' ' then

begin

end

else

while s[1] = ' ' do

delete(s, 1, 1);

lTrim := s;

end;

procedure Copyfilewin(handle: thandle; from, tof: string);

var

OpStruc: TSHFileOpStruct;

frombuf, tobuf: array [0 .. 1024] of char;

begin

FillChar(frombuf, sizeof(frombuf), 0);

FillChar(tobuf, sizeof(tobuf), 0);

StrPCopy(frombuf, from);

StrPCopy(tobuf, tof);

with OpStruc do

begin

Wnd := handle;

wFunc := FO\_COPY;

pFrom := @frombuf;

pTo := @tobuf;

fFlags := FOF\_NOCONFIRMATION or FOF\_RENAMEONCOLLISION;

fAnyOperationsAborted := false;

hNameMappings := nil;

lpszProgressTitle := nil;

end;

ShFileOperation(OpStruc);

end;

function ExecAndWait(aCmd: string; WaitTimeOut: cardinal = INFINITE): cardinal;

var

si: STARTUPINFO;

pi: PROCESS\_INFORMATION;

res: BOOL;

r: cardinal;

begin

with si do

begin

cb := sizeof(si);

lpReserved := nil;

lpDesktop := nil;

lpTitle := PChar('External program "' + aCmd + '"');

dwFlags := 0;

cbReserved2 := 0;

lpReserved2 := nil;

end;

res := CreateProcess(nil, PChar(aCmd), nil, nil, false, 0, nil, nil, si, pi);

if res then

WaitForSingleObject(pi.hProcess, WaitTimeOut);

GetExitCodeProcess(pi.hProcess, r);

Result := r;

end;

procedure StartProg(handle: thandle; prog, param: string);

begin

ShellExecute(handle, 'open', PChar(prog), PChar(param), nil, SW\_SHOWNORMAL);

end;

procedure StartCMD(handle: thandle; param: string);

begin

ShellExecute(handle, 'open', PChar('CMD.EXE /C'), PChar(param), nil,

SW\_SHOWNORMAL);

end;

function strmonth(d: tdatetime; caps: boolean): string;

var

dm, m, y: word;

s: string;

begin

Decodedate(d, y, m, dm);

if caps then

begin

case m of

1:

s := 'ЯНВАРЬ';

2:

s := 'ФЕВРАЛЬ';

3:

s := 'МАРТ';

4:

s := 'АПРЕЛЬ';

5:

s := 'МАЙ';

6:

s := 'ИЮНЬ';

7:

s := 'ИЮЛЬ';

8:

s := 'АВГУСТ';

9:

s := 'СЕНТЯБРЬ';

10:

s := 'ОКТЯБРЬ';

11:

s := 'НОЯБРЬ';

12:

s := 'ДЕКАБРЬ';

end;

end

else

begin

case m of

1:

s := 'январь';

2:

s := 'февраль';

3:

s := 'март';

4:

s := 'апрель';

5:

s := 'май';

6:

s := 'июнь';

7:

s := 'июль';

8:

s := 'август';

9:

s := 'сентябрь';

10:

s := 'октябрь';

11:

s := 'ноябрь';

12:

s := 'декабрь';

end;

end;

strmonth := s;

end;

Procedure pause(t: int64);

var

c: int64;

begin

c := GetTickCount;

repeat

Application.ProcessMessages

until GetTickCount - c >= t;

end;

// --------------------------------------------------------------------------------------------

function CopyFileMy(FromPath, ToPath: string): integer;

var

F1: file;

F2: file;

NumRead: integer;

NumWritten: integer;

Buf: pointer;

// BufSize: longint;

Totalbytes: longint;

TotalRead: longint;

begin

Result := 0;

Assignfile(F1, FromPath);

Assignfile(F2, ToPath);

Reset(F1, 1);

Totalbytes := Filesize(F1);

Rewrite(F2, 1);

// BufSize := 16384;

GetMem(Buf, Bufsize);

TotalRead := 0;

repeat

BlockRead(F1, Buf^, Bufsize, NumRead);

inc(TotalRead, NumRead);

BlockWrite(F2, Buf^, NumRead, NumWritten);

Application.ProcessMessages;

until (NumRead = 0) or (NumWritten <> NumRead);

if (NumWritten <> NumRead) then

begin

// ошибка

Result := -1;

end;

Closefile(F1);

Closefile(F2);

end;

function CopyFileProgress(FromPath, ToPath: string; p: tprogressbar): integer;

var

F1: file;

F2: file;

NumRead: integer;

NumWritten: integer;

Buf: pointer;

// BufSize: longint;

Totalbytes: longint;

TotalRead: longint;

begin

Result := 0;

Assignfile(F1, FromPath);

Assignfile(F2, ToPath);

Reset(F1, 1);

Totalbytes := Filesize(F1);

p.Min := 0;

p.Max := Filesize(F1);

p.Position := 0;

Rewrite(F2, 1);

// BufSize := 16384;

GetMem(Buf, Bufsize);

TotalRead := 0;

repeat

BlockRead(F1, Buf^, Bufsize, NumRead);

inc(TotalRead, NumRead);

BlockWrite(F2, Buf^, NumRead, NumWritten);

p.Position := p.Position + NumWritten;

p.Repaint;

Application.ProcessMessages;

until (NumRead = 0) or (NumWritten <> NumRead);

if (NumWritten <> NumRead) then

begin

// ошибка

Result := -1;

end;

Closefile(F1);

Closefile(F2);

end;

procedure Progress2status(p: tprogressbar; s: tstatusbar);

begin

with p do

begin

Position := 0;

Repaint;

Parent := s;

// Position := 100;

Top := 2;

Left := 0;

Height := s.Height - Top;

Width := s.Panels[0].Width - Left;

end;

end;

procedure nop;

var

i: integer;

begin

for i := 0 to 1 do

begin;

end;

end;

Function FileMove(fs1, fs2: string; del: integer): integer;

var

F1, F2: textfile;

s: string;

res: integer;

begin

res := -1;

Assignfile(F1, fs1);

{$I-}

Reset(F1);

If IoResult <> 0 then

begin

res := 1; // cannot open src

exit;

End

ELSE

begin

Assignfile(F2, fs2);

Reset(F2);

If IoResult = 0 then

begin

res := 2; // cannot create dst - exist

exit;

end

else

begin

Rewrite(F2);

If IoResult <> 0 then

begin

res := 3; // cannot create dst - cannot create new one

exit;

end

else

begin

// ReadLn(f1,s);

While not eof(F1) do

begin

ReadLn(F1, s);

WriteLn(F2, s);

res := 0; // excellent

end;

// WriteLn(f2,s);

end;

end;

End;

FileMove := res;

Closefile(F1);

Closefile(F2);

If del = 1 then

deleteFile(fs1);

end;

Function ParseOracleDate(s: string): string;

begin

s := StrReplace(s, '.01.', '.jan.');

s := StrReplace(s, '.02.', '.feb.');

s := StrReplace(s, '.03.', '.mar.');

s := StrReplace(s, '.04.', '.apr.');

s := StrReplace(s, '.05.', '.may.');

s := StrReplace(s, '.06.', '.jun.');

s := StrReplace(s, '.07.', '.jul.');

s := StrReplace(s, '.08.', '.aug.');

s := StrReplace(s, '.09.', '.sep.');

s := StrReplace(s, '.10.', '.oct.');

s := StrReplace(s, '.11.', '.nov.');

s := StrReplace(s, '.12.', '.dec.');

ParseOracleDate := s;

end;

Function Trims(s: string): string;

var

i: integer;

begin

i := 1;

While i < length(s) do

begin

While (s[i] = ' ') and (s[i + 1] = ' ') and (i < length(s)) do

begin

delete(s, i + 1, 1);

end;

inc(i);

If i >= length(s) then

break;

end;

Trims := s;

end;

Procedure ListFileDir(Path: string; FileList: TStrings; mask: string);

var

SR: TSearchRec;

begin

if FindFirst(Path + mask, faAnyFile, SR) = 0 then

begin

repeat

if (SR.Attr <> faDirectory) then

begin

FileList.Add(Path + SR.Name);

end;

until FindNext(SR) <> 0;

FindClose(SR);

end;

end;

function CreateDirEx(Dir: string): boolean;

var

i, L: integer;

CurDir: string;

begin

if ExcludeTrailingBackslash(Dir) = '' then

exit;

Dir := IncludeTrailingBackslash(Dir);

L := length(Dir);

for i := 1 to L do

begin

CurDir := CurDir + Dir[i];

if Dir[i] = '\' then

begin

if not DirectoryExists(CurDir) then

if not CreateDir(CurDir) then

exit;

end;

end;

Result := true;

end;

function DosToWin(st: string): string;

var

Ch: PAnsiChar;

begin

Ch := PAnsiChar(StrAlloc(length(st) + 1));

OemToAnsi(PAnsiChar(st), Ch);

Result := Ch;

StrDispose(Ch)

end;

function WinToDos(st: string): string;

var

Ch: PAnsiChar;

begin

Ch := PAnsiChar(StrAlloc(length(st) + 1));

AnsiToOem(PAnsiChar(st), Ch);

Result := Ch;

StrDispose(Ch)

end;

Function parsedatetime(s: string): string;

var

ts: string;

begin

ts := datetostr(Date);

delete(ts, pos('.', ts), length(ts));

s := StrReplace(s, '%D', ts);

s := StrReplace(s, '%d', inttostr(strtoint(ts)));

ts := datetostr(Date);

delete(ts, 1, pos('.', ts));

delete(ts, pos('.', ts), length(ts));

s := StrReplace(s, '%M', ts);

s := StrReplace(s, '%m', inttostr(strtoint(ts)));

ts := datetostr(Date);

delete(ts, 1, pos('.', ts));

delete(ts, 1, pos('.', ts));

s := StrReplace(s, '%Y', ts);

delete(ts, 1, 2);

s := StrReplace(s, '%y', ts);

ts := TimeToStr(Time);

delete(ts, pos(':', ts), length(ts));

s := StrReplace(s, '%h', ts);

if strtoint(ts) < 10 then

s := StrReplace(s, '%H', '0' + inttostr(strtoint(ts)))

else

s := StrReplace(s, '%H', inttostr(strtoint(ts)));

ts := TimeToStr(Time);

delete(ts, 1, pos(':', ts));

delete(ts, pos(':', ts), length(ts));

s := StrReplace(s, '%T', ts);

s := StrReplace(s, '%t', inttostr(strtoint(ts)));

ts := TimeToStr(Time);

delete(ts, 1, pos(':', ts));

delete(ts, 1, pos(':', ts));

s := StrReplace(s, '%S', ts);

s := StrReplace(s, '%s', inttostr(strtoint(ts)));

parsedatetime := s;

end;

Function parsedatetimeget(s: string; d: tdatetime): string;

var

ts: string;

begin

ts := datetostr(d);

delete(ts, pos('.', ts), length(ts));

s := StrReplace(s, '%D', ts);

s := StrReplace(s, '%d', inttostr(strtoint(ts)));

ts := datetostr(d);

delete(ts, 1, pos('.', ts));

delete(ts, pos('.', ts), length(ts));

s := StrReplace(s, '%M', ts);

s := StrReplace(s, '%m', inttostr(strtoint(ts)));

ts := datetostr(d);

delete(ts, 1, pos('.', ts));

delete(ts, 1, pos('.', ts));

s := StrReplace(s, '%Y', ts);

delete(ts, 1, 2);

s := StrReplace(s, '%y', ts);

ts := TimeToStr(d);

delete(ts, pos(':', ts), length(ts));

s := StrReplace(s, '%h', ts);

if strtoint(ts) < 10 then

s := StrReplace(s, '%H', '0' + inttostr(strtoint(ts)))

else

s := StrReplace(s, '%H', inttostr(strtoint(ts)));

ts := TimeToStr(d);

delete(ts, 1, pos(':', ts));

delete(ts, pos(':', ts), length(ts));

s := StrReplace(s, '%T', ts);

s := StrReplace(s, '%t', inttostr(strtoint(ts)));

ts := TimeToStr(d);

delete(ts, 1, pos(':', ts));

delete(ts, 1, pos(':', ts));

s := StrReplace(s, '%S', ts);

s := StrReplace(s, '%s', inttostr(strtoint(ts)));

parsedatetimeget := s;

end;

function StrReplace(const Str, Str1, Str2: string): string;

// str - исходная строка

// str1 - подстрока, подлежащая замене

// str2 - заменяющая строка

var

p, L: integer;

begin

Result := Str;

if length(Str) < length(Str1) then

exit;

if pos(Str1, Str) = 0 then

exit;

L := length(Str1);

repeat

p := pos(Str1, Result); // ищем подстроку

if p > 0 then

begin

delete(Result, p, L); // удаляем ее

Insert(Str2, Result, p); // вставляем новую

break;

end;

until p = 0;

end;

function pnexte(del: string; var s: string): string;

var

ts: string;

i: integer;

// st:string;

begin

for i := length(s) downto 1 do

begin

If s[i] = del then

begin

ts := copy(s, i + 1, length(s));

delete(s, i, length(s));

break;

end;

end;

// st:=ts;

pnexte := ts;

end;

function pnext(del: string; var s: string): string;

var

ts: string;

begin

ts := '';

If pos(del, s) <> 0 then

begin

ts := s;

delete(s, 1, pos(del, s));

delete(ts, pos(del, ts), length(ts));

end;

pnext := ts;

end;

function explode(sPart, sInput: string): ArrOfStr;

begin

setlength(Result, 0);

while pos(sPart, sInput) <> 0 do

begin

setlength(Result, length(Result) + 1);

Result[length(Result) - 1] := copy(sInput, 0, pos(sPart, sInput) - 1);

delete(sInput, 1, pos(sPart, sInput));

end;

setlength(Result, length(Result) + 1);

Result[length(Result) - 1] := sInput;

end;

function implode(sPart: string; arrInp: ArrOfStr): string;

var

i: integer;

begin

if length(arrInp) <= 1 then

Result := arrInp[0]

else

begin

for i := 0 to length(arrInp) - 2 do

Result := Result + arrInp[i] + sPart;

Result := Result + arrInp[length(arrInp) - 1];

end;

end;

procedure sort(arrInp: ArrOfStr);

var

slTmp: TStringList;

i: integer;

begin

slTmp := TStringList.Create;

for i := 0 to length(arrInp) - 1 do

slTmp.Add(arrInp[i]);

slTmp.sort;

for i := 0 to slTmp.Count - 1 do

arrInp[i] := slTmp[i];

slTmp.Free;

end;

procedure rsort(arrInp: ArrOfStr);

var

slTmp: TStringList;

i: integer;

begin

slTmp := TStringList.Create;

for i := 0 to length(arrInp) - 1 do

slTmp.Add(arrInp[i]);

slTmp.sort;

for i := 0 to slTmp.Count - 1 do

arrInp[slTmp.Count - 1 - i] := slTmp[i];

slTmp.Free;

end;

Function ParseMySQLDate(s: string): string;

var

ts1, ts2, ts3: string;

begin

ts1 := pnext('.', s);

ts2 := pnext('.', s);

ts3 := s;

ParseMySQLDate := ts3 + '-' + ts2 + '-' + ts1;

end;

procedure address(s: string; var street: string; var home: string;

var flat: string);

var

s1, s2, s3: string;

begin

flat := pnexte(' ', s);

home := pnexte(' ', s);

street := s;

end;

function Mypos(s: string; d: string; p: integer): string;

var

i: integer;

rs: string;

begin

for i := 1 to p do

begin

rs := pnext(d, s);

end;

Mypos := rs;

end;

function DetermineCodepage(const st: string): byte;

var

WinCount, AltCount, KoiCount, i, rslt: integer;

begin

DetermineCodepage := cpAlt;

WinCount := 0;

AltCount := 0;

KoiCount := 0;

for i := 1 to length(st) do

begin

if st[i] in AltSet then

inc(AltCount);

if st[i] in WinSet then

inc(WinCount);

if st[i] in KoiSet then

inc(KoiCount);

end;

DetermineCodepage := cpAlt;

if KoiCount > AltCount then

begin

DetermineCodepage := cpKoi;

if WinCount > KoiCount then

DetermineCodepage := cpWin;

end

else

begin

if WinCount > AltCount then

DetermineCodepage := cpWin;

end;

end;

function ConvertBankDateToDate(s: string): string;

var

rs, ts: string;

begin

rs := copy(s, 7, 2) + '.';

rs := rs + copy(s, 5, 2) + '.';

rs := rs + copy(s, 1, 4);

ConvertBankDateToDate := rs;

end;

function MyposReplace(sin: string; d: string; p: integer; sub: string): string;

var

s, s1, s2: string;

i: integer;

begin

for i := 1 to p - 1 do

begin

s := s + pnext(d, sin) + d;

end;

pnext(d, sin);

s := s + sub + d + sin;

MyposReplace := s;

end;

begin

MaskString := '\*.\*';

Bufsize := 32768;

end.