УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №4.1

по предмету

Основы алгоритмизации и программирования

Вариант 14

Выполнил:

Крутько А.А.

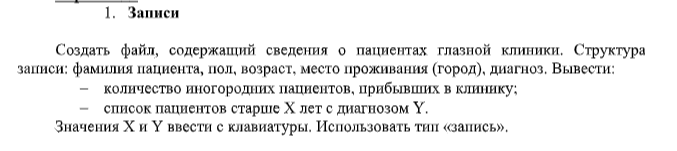
Проверила:

Данилова Г.В.

Группа 251004

Минск 2022

Задание:



Код программы на **Delphi**:

unit lab4\_1;

interface

uses

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

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.Grids, Vcl.Menus, Vcl.Imaging.GIFImg,

Vcl.ExtCtrls, Vcl.ComCtrls, System.ImageList, Vcl.ImgList;

type

TMainForm = class(TForm)

MainLabel: TLabel;

AddButton: TButton;

ChangeButton: TButton;

DeleteButton: TButton;

CountButton: TButton;

ListButton: TButton;

ListGrid: TStringGrid;

MainMenu1: TMainMenu;

N1: TMenuItem;

Creator: TMenuItem;

OpenFile: TMenuItem;

SaveFile: TMenuItem;

OpenDialog: TOpenDialog;

SaveDialog: TSaveDialog;

Image: TImage;

Timer1: TTimer;

TimeLabel: TLabel;

procedure FormCreate(Sender: TObject);

procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);

procedure AddButtonClick(Sender: TObject);

procedure ChangeButtonClick(Sender: TObject);

procedure DeleteButtonClick(Sender: TObject);

procedure ListGridDblClick(Sender: TObject);

procedure ListGridKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

procedure CountButtonClick(Sender: TObject);

procedure ListButtonClick(Sender: TObject);

procedure CreatorClick(Sender: TObject);

procedure OpenFileClick(Sender: TObject);

procedure SaveFileClick(Sender: TObject);

procedure Timer1Timer(Sender: TObject);

procedure ImageClick(Sender: TObject);

procedure AddButtonMouseEnter(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

Type

TInfoAboutPatients = Packed Record

SecondName: String[18];

Pol: String[7];

Age: Byte;

City: String[18];

Diagnos: String[18];

End;

Var

MainForm: TMainForm;

ArrOfPatients: Array of TInfoAboutPatients;

SearchListOfPatients: Array of TInfoAboutPatients;

IsEditing: Boolean;

Sign: Integer;

implementation

{$R \*.dfm}

uses AddUnit, ChangeUnit, ListUnit, FinalUnit;

procedure TMainForm.AddButtonClick(Sender: TObject);

begin

AddForm.PolComboBox.Text := '';

AddForm.AgeEdit.Text := '';

AddForm.SecondNameEdit.Text := '';

AddForm.CityEdit.Text := '';

AddForm.DiagnosEdit.Text := '';

AddForm.ShowModal;

end;

Procedure ChangeInformation;

Begin

IsEditing := True;

AddForm.PolComboBox.Text := ArrOfPatients[MainForm.ListGrid.Row - 1].Pol;

AddForm.AgeEdit.Text := IntToStr(ArrOfPatients[MainForm.ListGrid.Row - 1].Age);

AddForm.SecondNameEdit.Text := ArrOfPatients[MainForm.ListGrid.Row -   
 1].SecondName;

AddForm.CityEdit.Text := ArrOfPatients[MainForm.ListGrid.Row - 1].City;

AddForm.DiagnosEdit.Text := ArrOfPatients[MainForm.ListGrid.Row - 1].Diagnos;

AddForm.ShowModal;

End;

procedure TMainForm.AddButtonMouseEnter(Sender: TObject);

begin

If ListGrid.RowCount = 21 Then

Begin

AddButton.Left := Random(ClientWidth - 40) + 20;

AddButton.Top:= Random(ClientHeight - 100) + 50;

End;

end;

procedure TMainForm.ChangeButtonClick(Sender: TObject);

begin

ChangeInformation;

end;

Function CountFromOtherCities: String;

Var

I, Counter: Integer;

Begin

Counter := 0;

For I := 1 to MainForm.ListGrid.RowCount - 1 do

If MainForm.ListGrid.Cells[3, I] <> 'Минск' Then

Inc(Counter);

CountFromOtherCities := IntToStr(Counter);

End;

procedure TMainForm.CountButtonClick(Sender: TObject);

begin

MessageBox(MainForm.Handle, PChar('Количество ваших иногородних пациентов: ' + CountFromOtherCities), 'Внимание', MB\_ICONINFORMATION);

end;

procedure TMainForm.CreatorClick(Sender: TObject);

begin

ShowMessage('Студент группы 251004, Крутько Андрей Алексеевич');

end;

Procedure CheckButtons;

Begin

With MainForm do

If ListGrid.RowCount = 1 Then

Begin

ChangeButton.Enabled := False;

DeleteButton.Enabled := False;

CountButton.Enabled := False;

ListButton.Enabled := False;

End

Else

Begin

ChangeButton.Enabled := True;

DeleteButton.Enabled := True;

CountButton.Enabled := True;

ListButton.Enabled := True;

End;

End;

Procedure AdaptWindowAfterDelete;

Begin

If MainForm.ListGrid.RowCount > 5 Then

Begin

MainForm.ListGrid.Height := MainForm.ListGrid.Height - 26;

MainForm.Height := MainForm.Height - 24;

End;

If MainForm.ListGrid.RowCount = 1 Then

MainForm.SaveFile.Enabled := False;

End;

Procedure DeleteInformation;

Var

I: Integer;

Begin

Case MessageBox(MainForm.Handle, 'Вы действительно хотите удалить этого пациента?', 'Внимание', MB\_YESNO + MB\_ICONINFORMATION) Of

ID\_NO: ;

ID\_YES:

Begin

Finalize(ArrOfPatients[MainForm.ListGrid.Row - 1]);

For I := MainForm.ListGrid.Row - 1 to High(ArrOfPatients) - 1 do

ArrOfPatients[I] := ArrOfPatients[I + 1];

For I := MainForm.ListGrid.Row to MainForm.ListGrid.RowCount - 2 do

MainForm.ListGrid.Rows[I].Assign(MainForm.ListGrid.Rows[I + 1]);

MainForm.ListGrid.Rows[MainForm.ListGrid.RowCount - 1].Clear;

MainForm.ListGrid.RowCount := MainForm.ListGrid.RowCount - 1;

SetLength(ArrOfPatients, MainForm.ListGrid.RowCount);

CheckButtons;

AdaptWindowAfterDelete;

MainForm.AddButton.Top := 369;

MainForm.AddButton.Left := 16;

End;

End;

End;

procedure TMainForm.DeleteButtonClick(Sender: TObject);

Var

I: Integer;

begin

DeleteInformation;

end;

procedure TMainForm.FormCloseQuery(Sender: TObject; var CanClose: Boolean);

begin

CanClose := MessageBox(MainForm.Handle, 'Вы действительно хотите выйти?',   
 'Выход', MB\_YESNO + MB\_ICONQUESTION) = ID\_YES;

end;

procedure TMainForm.FormCreate(Sender: TObject);

begin

BorderIcons := BorderIcons - [biMaximize] + [biHelp];

BorderStyle := BsSingle;

Position := PoDesktopCenter;

ListGrid.Cells[0,0] := 'Фамилия';

ListGrid.Cells[1,0] := 'Возраст';

ListGrid.Cells[2,0] := 'Пол';

ListGrid.Cells[3,0] := 'Город';

ListGrid.Cells[4,0] := 'Диагноз';

ChangeButton.Enabled := False;

DeleteButton.Enabled := False;

CountButton.Enabled := False;

ListButton.Enabled := False;

SaveFile.Enabled := False;

(Image.Picture.Graphic as TGIFImage).AnimateLoop := glDisabled;

(Image.Picture.Graphic as TGIFImage).Animate := True; // gets it goin'

//(Image.Picture.Graphic as TGIFImage ).AnimationSpeed:= 0;// adjust your speed

TimeLabel.Caption := TimeToStr(Time);

Sign := -1;

end;

procedure TMainForm.ImageClick(Sender: TObject);

begin

Sign := -Sign;

If Sign = 1 Then

(Image.Picture.Graphic as TGIFImage).Animate := False

Else

(Image.Picture.Graphic as TGIFImage).Animate := True;

end;

procedure TMainForm.ListButtonClick(Sender: TObject);

begin

ListForm.AgeEdit.Text := '';

ListForm.DiagnosEdit.Text := '';

ListForm.SearchButton.Enabled := False;

ListForm.ShowModal;

end;

procedure TMainForm.ListGridDblClick(Sender: TObject);

begin

If ListGrid.RowCount > 1 Then

ChangeInformation;

end;

procedure TMainForm.ListGridKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

Var

I: Integer;

begin

If ((Key = VK\_DELETE) or (Key = VK\_BACK)) and (ListGrid.RowCount > 1) Then

DeleteInformation;

If Key = VK\_ESCAPE Then

Close;

If Key = VK\_F1 Then

MessageBox(MainForm.Handle,

'Основное:'#13#10 +

'1. Добавлять записи пациентов можно через кнопку "добавить"'#13#10 +

'2. Изменять записи через кнопку "изменить"'#13#10 +

'3. Удалять записи через кнопку "удалить"'#13#10 +

'4. В списке пациентов можно составить список по возрасту и нужному диагнозу   
 пациентов'#13#10 +

#13#10 + 'Дополнительно:'#13#10 +

'1. Максимум 20 пациентов'#13#10 +

'2. Одинаковых пациентов не может быть'#13#10 +

'3. При пустом списке, доступна только кнопка "добавить"'#13#10 +

'4. Можно редактировать записи двойным кликом мыши (левой)'#13#10 +

'5. Можно удалять запись путем нажатия клавиши "BACKSPACE" или   
 "DELETE"'#13#10, 'Помощь', MB\_OK);

end;

procedure TMainForm.OpenFileClick(Sender: TObject);

Var

FileInput: File of TInfoAboutPatients;

Current: TInfoAboutPatients;

Count: Integer;

begin

If OpenDialog.Execute Then

Begin

Try

Try

Reset(FileInput, OpenDialog.FileName);

Count := 0;

ListGrid.RowCount := 1;

Finalize(ArrOfPatients);

While Not Eof(FileInput) Do

Begin

Read(FileInput, Current);

SetLength(ArrOfPatients, Count + 1);

ArrOfPatients[Count] := Current;

ListGrid.RowCount := ListGrid.RowCount + 1;

ListGrid.Cells[0, ListGrid.RowCount - 1] :=   
 ArrOfPatients[Count].SecondName;

ListGrid.Cells[1, ListGrid.RowCount - 1] :=   
 IntToStr(ArrOfPatients[Count].Age);

ListGrid.Cells[2, ListGrid.RowCount - 1] :=   
 ArrOfPatients[Count].Pol;

ListGrid.Cells[3, ListGrid.RowCount - 1] :=   
 ArrOfPatients[Count].City;

ListGrid.Cells[4, ListGrid.RowCount - 1] :=   
 ArrOfPatients[Count].Diagnos;

Inc(Count);

End;

CheckButtons;

SaveFile.Enabled := True;

MainForm.ListGrid.FixedRows := 1;

Except

MessageBox(MainForm.Handle, 'Файл закрыт', 'Ошибка', MB\_OK +   
 MB\_ICONERROR);

End;

Finally

CloseFile(FileInput);

End;

End;

end;

Function CheckForExtension(Str: String): String;

Var

Ext: String;

Begin

Ext := Str;

If Pos('.', Str) > 0 Then

Delete(Ext, Pos('.', Ext), Length(Ext) - Pos('.', Ext) + 1);

CheckForExtension := Ext;

End;

procedure TMainForm.SaveFileClick(Sender: TObject);

Var

FileOutput: File of TInfoAboutPatients;

I: Integer;

Ext: String;

begin

If SaveDialog.Execute Then

Begin

Try

Try

Ext := CheckForExtension(SaveDialog.FileName);

ReWrite(FileOutput, Ext + '.hospital');

For I := 0 to High(ArrOfPatients) do

Write(FileOutput, ArrOfPatients[I]);

Except

MessageBox(MainForm.Handle, 'Файл закрыт для вывода', 'Ошибка',   
 MB\_OK + MB\_ICONERROR);

End;

Finally

CloseFile(FileOutput);

End;

End;

end;

procedure TMainForm.Timer1Timer(Sender: TObject);

begin

TimeLabel.Caption := TimeToStr(Time);

end;

end.

unit AddUnit;

interface

uses

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

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.Menus, ClipBrd;

type

TAddForm = class(TForm)

MainLabel: TLabel;

SecondNameLabel: TLabel;

AgeLabel: TLabel;

PolLabel: TLabel;

CityLabel: TLabel;

DiagnosLabel: TLabel;

PolComboBox: TComboBox;

SecondNameEdit: TEdit;

AgeEdit: TEdit;

CityEdit: TEdit;

DiagnosEdit: TEdit;

AddButton: TButton;

PopupMenu: TPopupMenu;

Copy: TMenuItem;

Cut: TMenuItem;

Insert: TMenuItem;

Delete: TMenuItem;

PopupMenu1: TPopupMenu;

procedure FormCreate(Sender: TObject);

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

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

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

procedure EditChange(Sender: TObject);

procedure KeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

procedure CopyClick(Sender: TObject);

procedure CutClick(Sender: TObject);

procedure InsertClick(Sender: TObject);

procedure DeleteClick(Sender: TObject);

procedure AgeEditMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure AgeEditMouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure AddButtonClick(Sender: TObject);

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

private

{ Private declarations }

public

{ Public declarations }

end;

var

AddForm: TAddForm;

implementation

{$R \*.dfm}

uses lab4\_1, ChangeUnit;

procedure TAddForm.AgeEditMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

begin

If Button = mbRight Then

Insert.Enabled := False;

end;

procedure TAddForm.AgeEditMouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

begin

Insert.Enabled := True;

end;

procedure TAddForm.CopyClick(Sender: TObject);

begin

If PopupMenu.PopupComponent = CityEdit Then

CityEdit.CopyToClipboard;

If PopupMenu.PopupComponent = SecondNameEdit Then

SecondNameEdit.CopyToClipboard;

If PopupMenu.PopupComponent = DiagnosEdit Then

DiagnosEdit.CopyToClipboard;

If PopupMenu.PopupComponent = AgeEdit Then

AgeEdit.CopyToClipboard;

Insert.Enabled := True;

end;

procedure TAddForm.CutClick(Sender: TObject);

begin

If PopupMenu.PopupComponent = CityEdit Then

CityEdit.CutToClipboard;

If PopupMenu.PopupComponent = SecondNameEdit Then

SecondNameEdit.CutToClipboard;

If PopupMenu.PopupComponent = DiagnosEdit Then

DiagnosEdit.CutToClipboard;

If PopupMenu.PopupComponent = AgeEdit Then

AgeEdit.CutToClipboard;

Insert.Enabled := True;

end;

procedure TAddForm.DeleteClick(Sender: TObject);

begin

If PopupMenu.PopupComponent = CityEdit Then

CityEdit.Text := '';

If PopupMenu.PopupComponent = SecondNameEdit Then

SecondNameEdit.Text := '';

If PopupMenu.PopupComponent = DiagnosEdit Then

DiagnosEdit.Text := '';

If PopupMenu.PopupComponent = AgeEdit Then

AgeEdit.Text := '';

end;

procedure TAddForm.EditChange(Sender: TObject);

begin

AddButton.Enabled := Not(''.IsNullOrWhiteSpace(CityEdit.Text) or   
 ''.IsNullOrWhiteSpace(DiagnosEdit.Text) or ''.IsNullOrWhiteSpace(AgeEdit.Text) or   
 ''.IsNullOrWhiteSpace(SecondNameEdit.Text) or   
 ''.IsNullOrWhiteSpace(PolComboBox.Text));

end;

Procedure CheckFirstSymbols();

Var

Temp: String;

TempLow: String;

Begin

AddForm.PolComboBox.Text := Trim(AddForm.PolComboBox.Text);

AddForm.AgeEdit.Text := Trim(AddForm.AgeEdit.Text);

AddForm.SecondNameEdit.Text := Trim(AddForm.SecondNameEdit.Text);

AddForm.CityEdit.Text := Trim(AddForm.CityEdit.Text);

AddForm.DiagnosEdit.Text := Trim(AddForm.DiagnosEdit.Text);

Temp := AddForm.SecondNameEdit.Text;

Insert(AnsiUpperCase(Temp[1]), Temp, 1);

Delete(Temp, 2, 1);

TempLow := Copy(Temp, 2, Length(Temp) - 1);

TempLow := AnsiLowerCase(TempLow);

Delete(Temp, 2, Length(Temp));

Temp := Concat(Temp, TempLow);

AddForm.SecondNameEdit.Text := Temp;

Temp := AddForm.CityEdit.Text;

Insert(AnsiUpperCase(Temp[1]), Temp, 1);

Delete(Temp, 2, 1);

TempLow := Copy(Temp, 2, Length(Temp) - 1);

TempLow := AnsiLowerCase(TempLow);

Delete(Temp, 2, Length(Temp));

Temp := Concat(Temp, TempLow);

AddForm.CityEdit.Text := Temp;

Temp := AddForm.DiagnosEdit.Text;

Insert(AnsiUpperCase(Temp[1]), Temp, 1);

Delete(Temp, 2, 1);

TempLow := Copy(Temp, 2, Length(Temp) - 1);

TempLow := AnsiLowerCase(TempLow);

Delete(Temp, 2, Length(Temp));

Temp := Concat(Temp, TempLow);

AddForm.DiagnosEdit.Text := Temp;

End;

Function IsNotPatientsSimilar: Boolean;

Var

IsCorrect: Boolean;

I: Integer;

Begin

CheckFirstSymbols();

IsCorrect := True;

I := 1;

While IsCorrect and (I < MainForm.ListGrid.RowCount) do

Begin

IsCorrect := Not((AddForm.SecondNameEdit.Text =   
 MainForm.ListGrid.Cells[0, I]) and (AddForm.AgeEdit.Text =   
 MainForm.ListGrid.Cells[1, I]) and (AddForm.PolComboBox.Text =   
 MainForm.ListGrid.Cells[2, I]) and (AddForm.CityEdit.Text =   
 MainForm.ListGrid.Cells[3, I]) and (AddForm.DiagnosEdit.Text =   
 MainForm.ListGrid.Cells[4, I]));

Inc(I);

End;

IsNotPatientsSimilar := IsCorrect;

End;

Procedure TakeInformationIntoRecord(Count: Integer);

Begin

ArrOfPatients[Count].Pol := AddForm.PolComboBox.Text;

ArrOfPatients[Count].Age := StrToInt(AddForm.AgeEdit.Text);

ArrOfPatients[Count].SecondName := AddForm.SecondNameEdit.Text;

ArrOfPatients[Count].City := AddForm.CityEdit.Text;

ArrOfPatients[Count].Diagnos := AddForm.DiagnosEdit.Text;

End;

Procedure ShowRecordInGrid(Count: Integer);

Begin

MainForm.ListGrid.Cells[0, Count] := ArrOfPatients[Count - 1].SecondName;

MainForm.ListGrid.Cells[1, Count] := IntToStr(ArrOfPatients[Count - 1].Age);

MainForm.ListGrid.Cells[2, Count] := ArrOfPatients[Count - 1].Pol;

MainForm.ListGrid.Cells[3, Count] := ArrOfPatients[Count - 1].City;

MainForm.ListGrid.Cells[4, Count] := ArrOfPatients[Count - 1].Diagnos;

End;

Procedure AdaptTheWindow;

Begin

If MainForm.ListGrid.RowCount > 6 Then

Begin

MainForm.ListGrid.Height := MainForm.ListGrid.Height + 26;

MainForm.Height := MainForm.Height + 24;

End;

End;

Function WasNotChanged: Boolean;

Var

IsNotChanged: Boolean;

Begin

IsNotChanged := (AddForm.SecondNameEdit.Text =   
 ArrOfPatients[MainForm.ListGrid.Row - 1].SecondName) and (AddForm.AgeEdit.Text =   
 IntToStr(ArrOfPatients[MainForm.ListGrid.Row - 1].Age)) and   
 (AddForm.PolComboBox.Text = ArrOfPatients[MainForm.ListGrid.Row - 1].Pol) and   
 (AddForm.CityEdit.Text = ArrOfPatients[MainForm.ListGrid.Row - 1].City) and   
 (AddForm.DiagnosEdit.Text = ArrOfPatients[MainForm.ListGrid.Row - 1].Diagnos);

WasNotChanged := IsNotChanged;

End;

procedure TAddForm.AddButtonClick(Sender: TObject);

Var

I: Integer;

IsCorrect: Boolean;

begin

If IsEditing Then

If IsNotPatientsSimilar xor WasNotChanged Then

Begin

TakeInformationIntoRecord(MainForm.ListGrid.Row - 1);

ShowRecordInGrid(MainForm.ListGrid.Row);

AddForm.Close;

End

Else

Application.MessageBox('Такой пациент уже есть в списке!', 'Ошибка', MB\_ICONERROR)

Else

If IsNotPatientsSimilar Then

Begin

SetLength(ArrOfPatients, MainForm.ListGrid.RowCount);

TakeInformationIntoRecord(High(ArrOfPatients));

MainForm.ListGrid.RowCount := MainForm.ListGrid.RowCount + 1;

ShowRecordInGrid(Length(ArrOfPatients));

MainForm.ChangeButton.Enabled := True;

MainForm.DeleteButton.Enabled := True;

MainForm.CountButton.Enabled := True;

MainForm.ListButton.Enabled := True;

MainForm.ListGrid.FixedRows := 1;

AdaptTheWindow;

MainForm.SaveFile.Enabled := True;

AddForm.Close;

End

Else

Application.MessageBox('Такой пациент уже есть в списке!', 'Ошибка', MB\_ICONERROR);

end;

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

begin

IsEditing := False;

end;

procedure TAddForm.FormCreate(Sender: TObject);

begin

BorderIcons := BorderIcons - [biMaximize] + [biHelp];

BorderStyle := BsSingle;

Position := PoDesktopCenter;

AddButton.Enabled := False;

end;

procedure TAddForm.InsertClick(Sender: TObject);

begin

If PopupMenu.PopupComponent = CityEdit Then

CityEdit.PasteFromClipboard;

If PopupMenu.PopupComponent = SecondNameEdit Then

SecondNameEdit.PasteFromClipboard;

If PopupMenu.PopupComponent = DiagnosEdit Then

DiagnosEdit.PasteFromClipboard;

end;

procedure TAddForm.KeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

begin

TEdit(Sender).ReadOnly := (((Shift=[ssShift]) and (Key = VK\_INSERT)) or (Shift=[ssCtrl]) or (Shift=[ssAlt]));

If Key = VK\_ESCAPE Then

AddForm.Close;

If Key = VK\_F1 Then

MessageBox(AddForm.Handle,

'Основное:'#13#10 +

'1. В поле "Имя" указывать имя до 18 символов'#13#10 +

'2. В поле "Пол" выбрать свой пол с помощью развернутого списка'#13#10 +

'3. В поле "Возраст" указывать возраст от 0 до 99 лет'#13#10 +

'4. В поле "Город" указывать город до 18 символов'#13#10 +

'5. В поле "Диагноз" указывать диагноз до 18 символов'#13#10 +

#13#10 + 'Дополнительно:'#13#10 +

'1. Первые буквы автоматически пишутся с большой'#13#10 +

'2. "Minsk" является городом поликлиники, все остальные указанные города   
 причисляют людей к иногородним'#13#10 +

'3. Кнопка "добавить" неактивна до тех пор, пока не будут заполнены все   
 ячейки'#13#10 +

'4. При нажатии ранее кнопки изменить и не изменив ничего среди полей,   
 пациент не будет перезаписан'#13#10 +

'5. Можно закрыть поле путем нажатии клавиши "ESC"'#13#10 +

'6. При нажатии ранее кнопки изменить и изменив поле пациента таким образом,   
 что уже есть такой же пациент, выскочит предупреждение'

, 'Помощь', MB\_OK);

end;

end.

unit FinalUnit;

interface

uses

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

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.Grids, Vcl.Menus;

type

TFinalForm = class(TForm)

MainLabel: TLabel;

ListGrid: TStringGrid;

MainMenu: TMainMenu;

SaveButton: TMenuItem;

SaveDialog: TSaveDialog;

procedure FormCreate(Sender: TObject);

procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);

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

procedure KeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);

procedure FormShow(Sender: TObject);

procedure SaveButtonClick(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

FinalForm: TFinalForm;

implementation

{$R \*.dfm}

Uses

lab4\_1, ListUnit;

Procedure TakeInformationIntoRecord(Count, I: Integer);

Begin

SearchListOfPatients[Count].Pol := ArrOfPatients[I - 1].Pol;

SearchListOfPatients[Count].Age := ArrOfPatients[I - 1].Age;

SearchListOfPatients[Count].City := ArrOfPatients[I - 1].City;

SearchListOfPatients[Count].Diagnos := ArrOfPatients[I - 1].Diagnos;

SearchListOfPatients[Count].SecondName := ArrOfPatients[I - 1].SecondName;

End;

Procedure ShowInformationIntoRecord(Count, I: Integer);

Begin

FinalForm.ListGrid.Cells[0, Count] := SearchListOfPatients[Count - 1].SecondName;

FinalForm.ListGrid.Cells[1, Count] := IntToStr(SearchListOfPatients[Count -   
 1].Age);

FinalForm.ListGrid.Cells[2, Count] := SearchListOfPatients[Count - 1].Pol;

FinalForm.ListGrid.Cells[3, Count] := SearchListOfPatients[Count - 1].City;

FinalForm.ListGrid.Cells[4, Count] := SearchListOfPatients[Count - 1].Diagnos;

End;

Procedure ClearMemory;

Var

I: Integer;

Begin

For I := 1 to FinalForm.ListGrid.RowCount - 1 do

FinalForm.ListGrid.Rows[I].Clear;

End;

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

begin

Finalize(SearchListOfPatients);

//SearchListOfPatients := nil;

ListGrid.Height := 161;

ListGrid.Width := 665;

ClearMemory;

ListGrid.FixedRows := 0;

ListGrid.RowCount := 1;

end;

procedure TFinalForm.FormCloseQuery(Sender: TObject; var CanClose: Boolean);

begin

CanClose := MessageBox(MainForm.Handle, 'Вы действительно хотите выйти?',   
 'Выход', MB\_YESNO + MB\_ICONQUESTION) = ID\_YES;

end;

Procedure AdaptTheWindow;

Begin

If FinalForm.ListGrid.RowCount > 6 Then

Begin

FinalForm.ListGrid.Height := FinalForm.ListGrid.Height + 26;

FinalForm.Height := FinalForm.Height + 24;

End;

End;

procedure TFinalForm.FormCreate(Sender: TObject);

begin

BorderIcons := BorderIcons - [biMaximize] + [biHelp];

BorderStyle := BsSingle;

Position := PoDesktopCenter;

ListGrid.Cells[0,0] := 'Фамилия';

ListGrid.Cells[1,0] := 'Возраст';

ListGrid.Cells[2,0] := 'Пол';

ListGrid.Cells[3,0] := 'Город';

ListGrid.Cells[4,0] := 'Диагноз';

end;

procedure TFinalForm.KeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

Var

CanClose: Boolean;

begin

If Key = VK\_ESCAPE Then

FinalForm.Close;

end;

procedure TFinalForm.SaveButtonClick(Sender: TObject);

Var

FileOutput: File of TInfoAboutPatients;

I: Integer;

begin

If SaveDialog.Execute Then

Begin

Try

Try

ReWrite(FileOutput, SaveDialog.FileName + '.hospital');

For I := 0 to High(SearchListOfPatients) do

Write(FileOutput, SearchListOfPatients[I]);

Except

MessageBox(MainForm.Handle, 'Файл закрыт для вывода', 'Ошибка',   
 MB\_OK + MB\_ICONERROR);

End;

Finally

CloseFile(FileOutput);

End;

End;

end;

procedure TFinalForm.FormShow(Sender: TObject);

Var

I: Integer;

begin

ListForm.Close;

ListForm.SetFocus;

For I := 1 to MainForm.ListGrid.RowCount - 1 do

Begin

If (MainForm.ListGrid.Cells[1, I] >= ListForm.AgeEdit.Text) and   
 (MainForm.ListGrid.Cells[4, I] = ListForm.DiagnosEdit.Text) Then

Begin

SetLength(SearchListOfPatients, FinalForm.ListGrid.RowCount);

TakeInformationIntoRecord(FinalForm.ListGrid.RowCount - 1, I);

FinalForm.ListGrid.RowCount := FinalForm.ListGrid.RowCount + 1;

ShowInformationIntoRecord(FinalForm.ListGrid.RowCount - 1, I);

AdaptTheWindow;

End;

End;

If FinalForm.ListGrid.RowCount > 1 Then

Begin

FinalForm.ListGrid.FixedRows := 1;

SaveButton.Enabled := True;

End

Else

Begin

MessageBox(ListForm.Handle, 'Не найдено ни одного совпадения!',   
 'Внимание', MB\_OK + MB\_ICONINFORMATION);

SaveButton.Enabled := False;

End;

end;

end.

unit ListUnit;

interface

uses

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

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.Menus, ClipBrd;

type

TListForm = class(TForm)

AgeLabel: TLabel;

MainLabel: TLabel;

AgeEdit: TEdit;

DiagnosLabel: TLabel;

DiagnosEdit: TEdit;

SearchButton: TButton;

PopupMenu: TPopupMenu;

Copy: TMenuItem;

Cut: TMenuItem;

Insert: TMenuItem;

Delete: TMenuItem;

procedure FormCreate(Sender: TObject);

procedure KeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);

procedure EditChange(Sender: TObject);

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

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

procedure CopyClick(Sender: TObject);

procedure CutClick(Sender: TObject);

procedure InsertClick(Sender: TObject);

procedure DeleteClick(Sender: TObject);

procedure AgeEditMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure AgeEditMouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure SearchButtonClick(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

ListForm: TListForm;

implementation

{$R \*.dfm}

uses lab4\_1, FinalUnit;

procedure TListForm.AgeEditKeyPress(Sender: TObject; var Key: Char);

begin

If Not (Key In ['0'..'9', #08]) Then

Key := #0;

If (Length(AgeEdit.Text) = 2) and (Key <> #08) Then

Key := #0;

end;

procedure TListForm.EditChange(Sender: TObject);

begin

SearchButton.Enabled := (AgeEdit.GetTextLen > 0) and (DiagnosEdit.GetTextLen > 0);

end;

procedure TListForm.FormCreate(Sender: TObject);

begin

BorderIcons := BorderIcons - [biMaximize] + [biHelp];

BorderStyle := BsSingle;

Position := PoDesktopCenter;

end;

procedure TListForm.KeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

begin

TEdit(Sender).ReadOnly := (((Shift=[ssShift]) and (Key = VK\_INSERT)) or   
 (Shift=[ssCtrl]) or (Shift=[ssAlt]));

Case Key Of

VK\_ESCAPE:

ListForm.Close;

VK\_F1:

MessageBox(ListForm.Handle,

'Основное:'#13#10 +

'1. В поле "Диагноз" указывать имя до 18 символов'#13#10 +

'2. В поле "Возраст" указывать возраст от 0 до 99 лет'#13#10 +

#13#10 + 'Дополнительно:'#13#10 +

'1. Первые буквы автоматически пишутся с большой'#13#10 +

'2. Кнопка "поиск" неактивна до тех пор, пока не будут заполнены все   
 ячейки'#13#10 +

'3. Можно закрыть поле путем нажатии клавиши "ESC"', 'Помощь', MB\_OK);

End;

end;

Procedure CheckFirstSymbols();

Var

Temp: String;

TempLow: String;

Begin

Temp := ListForm.DiagnosEdit.Text;

Insert(AnsiUpperCase(Temp[1]), Temp, 1);

Delete(Temp, 2, 1);

TempLow := Copy(Temp, 2, Length(Temp) - 1);

TempLow := AnsiLowerCase(TempLow);

Delete(Temp, 2, Length(Temp));

Temp := Concat(Temp, TempLow);

ListForm.DiagnosEdit.Text := Temp;

End;

procedure TListForm.SearchButtonClick(Sender: TObject);

begin

CheckFirstSymbols();

FinalForm.ShowModal;

end;

procedure TListForm.AgeEditMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

begin

If Button = mbRight Then

Insert.Enabled := False;

end;

procedure TListForm.AgeEditMouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

begin

If Button = mbRight Then

Insert.Enabled := True;

end;

procedure TListForm.CopyClick(Sender: TObject);

begin

If PopupMenu.PopupComponent = DiagnosEdit Then

DiagnosEdit.CopyToClipboard;

If PopupMenu.PopupComponent = AgeEdit Then

AgeEdit.CopyToClipboard;

end;

procedure TListForm.CutClick(Sender: TObject);

begin

If PopupMenu.PopupComponent = DiagnosEdit Then

DiagnosEdit.CutToClipboard;

If PopupMenu.PopupComponent = AgeEdit Then

AgeEdit.CutToClipboard;

end;

procedure TListForm.InsertClick(Sender: TObject);

begin

If PopupMenu.PopupComponent = DiagnosEdit Then

DiagnosEdit.PasteFromClipboard;

end;

procedure TListForm.DeleteClick(Sender: TObject);

begin

If PopupMenu.PopupComponent = AgeEdit Then

AgeEdit.Text := '';

If PopupMenu.PopupComponent = DiagnosEdit Then

DiagnosEdit.Text := '';

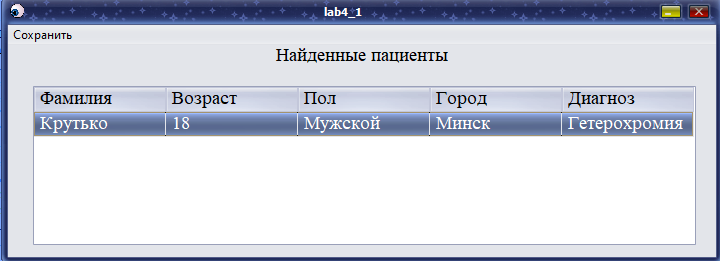
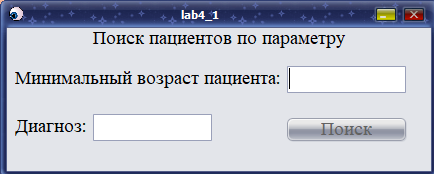
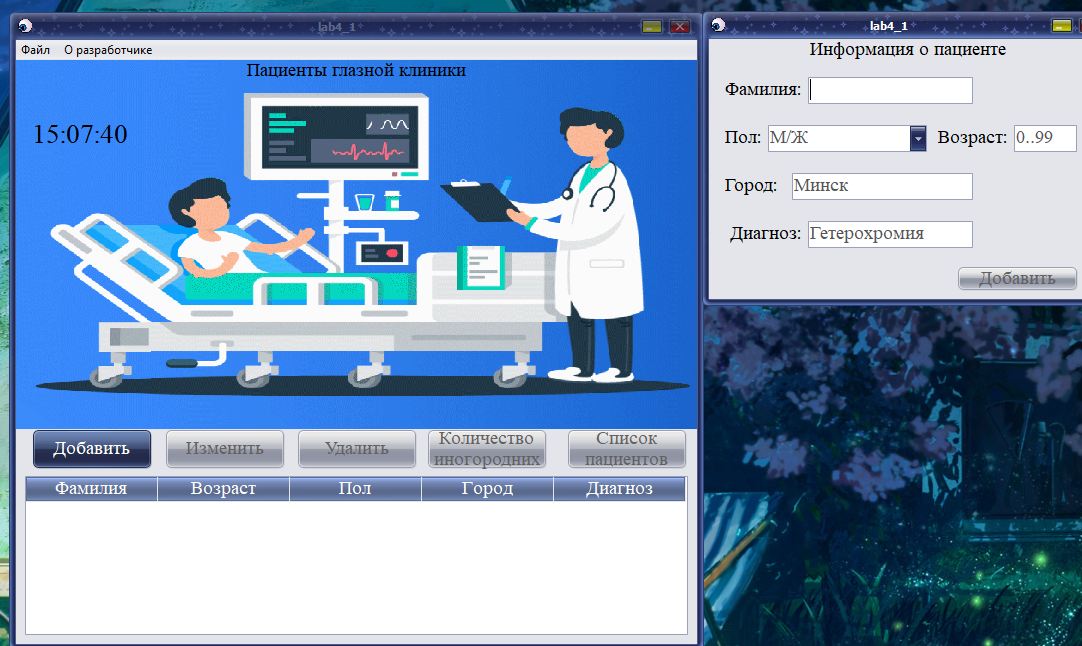
end;

end.

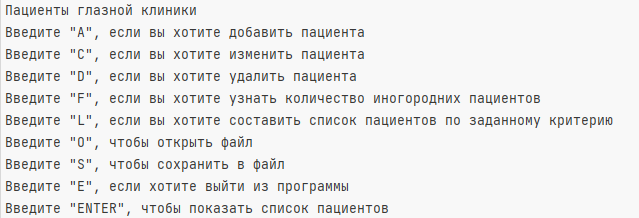
Код программы на **C#**:

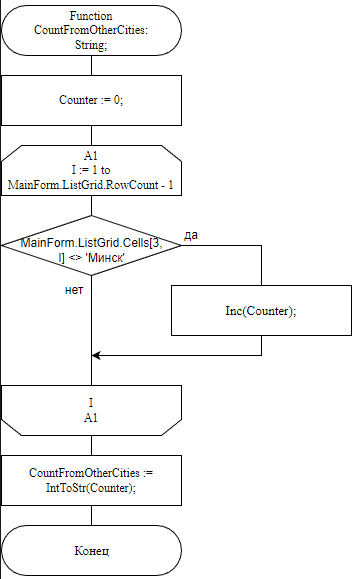
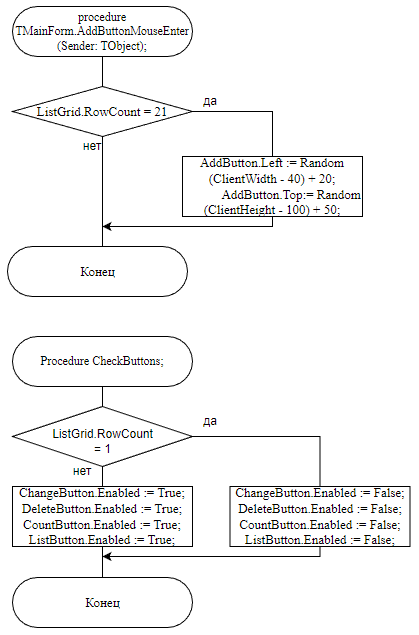
using lab4\_1;  
  
void InfoInMainMenu()  
{   
 Console.WriteLine("Введите \"A\", если вы хотите добавить   
 пациента\nВведите \"C\", если вы хотите изменить пациента\nВведите \"D\",   
 если вы хотите удалить пациента\nВведите \"F\", если вы хотите узнать   
 количество иногородних пациентов\nВведите \"L\", если вы хотите составить   
 список пациентов по заданному критерию\nВведите \"O\", чтобы открыть   
 файл\nВведите \"S\", чтобы сохранить в файл\nВведите \"E\", если хотите   
 выйти из программы\nВведите \"ENTER\", чтобы показать список пациентов");  
}  
  
void InfoForHelp()  
{  
 Console.WriteLine("\nВведите \"H\", если хотите вывести помощь по меню");  
}  
  
bool CheckForDigits(string str)  
{  
 bool isIncorrect = true;  
 for (int i = 0; i < str.Length && isIncorrect; i++)   
 isIncorrect = !Char.IsDigit(str[i]);  
 return isIncorrect;  
}  
  
string AddNameOfPatient()  
{  
 string? name;  
 do  
 {  
 Console.Write("Введите имя пациента: ");  
 name = Console.ReadLine();  
 } while (string.IsNullOrWhiteSpace(name) || !CheckForDigits(name));  
 return name.Substring(0, 1).ToUpper() + name.Substring(1).ToLower();  
}  
  
sbyte AddAgeOfPatient()  
{  
 sbyte age = 0;  
 bool isInCorrect;  
 do  
 {  
 Console.Write("Введите возраст: ");  
 isInCorrect = false;  
 try  
 {  
 age = Convert.ToSByte(Console.ReadLine()); //обернуть в checked  
 }  
 catch (Exception)  
 {  
 Console.Error.WriteLine("Некорректный ввод");  
 isInCorrect = true;  
 }  
 if (!isInCorrect && (age < 0 || age > 99))  
 {  
 isInCorrect = true;  
 Console.WriteLine("Неправильный диапазон данных");  
 }   
 } while (isInCorrect);  
 return age;  
}  
  
string AddPolOfPatient()  
{  
 string? pol;  
 do  
 {  
 Console.Write("Введите пол пациента: ");  
 pol = Console.ReadLine();  
 pol = pol?.Substring(0, 1).ToUpper() + pol?.Substring(1).ToLower();  
 } while (!(pol.Equals("Man") || pol.Equals("Woman")));  
 return pol;  
}  
  
string AddCityOfPatient()  
{  
 string? city;  
 do  
 {  
 Console.Write("Введите город пациента: ");  
 city = Console.ReadLine();  
 } while (string.IsNullOrWhiteSpace(city) || !CheckForDigits(city));  
 return city.Substring(0, 1).ToUpper() + city.Substring(1).ToLower();  
}  
  
string AddDiagnosOfPatient()  
{  
 string? diagnos;  
 do  
 {  
 Console.Write("Введите диагноз пациента: ");  
 diagnos = Console.ReadLine();  
 } while (string.IsNullOrWhiteSpace(diagnos) || !CheckForDigits(diagnos));  
 return diagnos.Substring(0, 1).ToUpper() + diagnos.Substring(1).ToLower();  
}  
  
void Resize(ref Person[] patient, int newSize)  
{  
 var newArray = new Person[newSize];  
 for (int i = 0; i < patient.Length; i++)  
 {  
 newArray[i] = patient[i];  
 }  
 patient = newArray;  
}  
  
void ResizeAfterDelete(ref Person[] patient, int inputIndex)   
{  
 var newArray = new Person[patient.Length - 1];  
 for (int i = inputIndex; i < patient.GetUpperBound(0); i++)  
 patient[i] = patient[i + 1];  
 for (int i = 0; i < newArray.Length; i++)  
 newArray[i] = patient[i];  
 patient = newArray;  
}  
  
void ShowList(Person[] patient)  
{  
 for (int i = 0; i < patient.Length; i++)  
 Console.WriteLine($"Пациент {i + 1}: " + $"Имя —   
 {patient[i].LastName} ; " + $"Возраст — {patient[i].Age} ; " + $"Пол   
 — {patient[i].Pol} ; " + $"Город — {patient[i].City} ; " + $"Диагноз   
 — {patient[i].Diagnos} ; ");  
}  
  
bool IsNotSimilarPatients(Person[] patient, string addName, sbyte addAge, string addPol, string addCity, string addDiagnos)  
{  
 bool isIncorrect = true;  
 for (int i = 0; i < patient.Length && isIncorrect; i++)  
 isIncorrect = !(patient[i].LastName == addName & patient[i].Age ==   
 addAge & patient[i].City == addCity & patient[i].Pol == addPol &   
 patient[i].Diagnos == addDiagnos);  
 return isIncorrect;  
}  
  
int InputIndexForChange(Person[] patient)  
{  
 int inputNumber = 0;  
 do  
 {  
 try  
 {  
 inputNumber = Convert.ToInt32(Console.ReadLine());  
 }  
 catch (Exception)  
 {  
 Console.Error.WriteLine("Некорректный ввод");  
 }  
 } while (!(inputNumber > 0 & inputNumber < patient.Length + 1));  
 return inputNumber;  
}  
  
void MessageAfterAction()  
{  
 Console.WriteLine("Вы вернулись в главное меню. Ожидается ваше следующее   
 действие");  
}  
  
void AddInfo(ref string name, ref sbyte age, ref string pol, ref string city, ref string diagnos)  
{  
 name = AddNameOfPatient();  
 age = AddAgeOfPatient();  
 pol = AddPolOfPatient();  
 city = AddCityOfPatient();  
 diagnos = AddDiagnosOfPatient();   
}  
  
bool IsNotChanged(int inputIndex, Person[] patient, string addName, sbyte addAge, string addPol, string addCity, string addDiagnos)  
{  
 return patient[inputIndex].LastName == addName & patient[inputIndex].Age   
 == addAge &  
 patient[inputIndex].Pol == addPol & patient[inputIndex].City ==   
 addCity &  
 patient[inputIndex].Diagnos == addDiagnos;  
}  
  
int NumberOfPatientsFromOtherCities(Person[] patient)  
{  
 int counter = 0;  
 for (int i = 0; i < patient.Length; i++)  
 if (patient[i].City.Equals("Minsk"))  
 counter++;  
 return counter;  
}  
  
string? TakeFilePath()  
{  
 string? path;  
 bool isIncorrect;  
 do {  
 isIncorrect = false;  
 Console.WriteLine("\nВведите путь к файлу");  
 path = Console.ReadLine();  
 if (!File.Exists(path) || !path.EndsWith(".txt"))  
 {  
 isIncorrect = true;  
 Console.WriteLine("Проверьте параметры файла");  
 }  
 } while (isIncorrect);  
 return path;  
}  
  
bool CheckInfoFromFile(string[] finalInfo)  
{  
 for (int i = 0; i < finalInfo.Length; i++)  
 finalInfo[i] = finalInfo[i].Substring(0, 1).ToUpper() +   
 finalInfo[i].Substring(1).ToLower();  
 return sbyte.TryParse(finalInfo[1], out \_) && (finalInfo[2].Equals("Man")   
 || finalInfo[2].Equals("Woman")) &&  
 CheckForDigits(finalInfo[0]) && CheckForDigits(finalInfo[3]) &&   
 CheckForDigits(finalInfo[4]);  
}  
  
void OpenFile(ref Person[] patient)  
{  
 patient = new Person[0];  
 string? path = TakeFilePath(), str = "";  
 try  
 {  
 using (StreamReader st = new StreamReader(path))  
 {  
 while ((str = st.ReadLine()) != null)  
 {  
 string[] arrStrings = str.Split(" ; ");  
 string[] finalInfo = new string[arrStrings.GetUpperBound(0)];  
 for (int i = 0; i < arrStrings.GetUpperBound(0); i++)  
 finalInfo[i] =   
 arrStrings[i].Substring(arrStrings[i].LastIndexOf('—') + 2);  
 if (finalInfo.Length == 5 && CheckInfoFromFile(finalInfo) &&   
 IsNotSimilarPatients(patient, finalInfo[0],   
 Convert.ToSByte(finalInfo[1]), finalInfo[2], finalInfo[3],   
 finalInfo[4]))  
 {  
 Resize(ref patient, patient.Length + 1);  
 patient[patient.GetUpperBound(0)] = new   
 Person(finalInfo[0], Convert.ToSByte(finalInfo[1]),   
 finalInfo[2], finalInfo[3], finalInfo[4]);   
 }  
 }  
 }  
 }  
 catch (Exception e)  
 {  
 Console.WriteLine(e.Message);  
 }  
}  
  
void SaveFile(Person[] patient)  
{  
 if (patient.Length > 0)  
 {  
 string? path = TakeFilePath();  
 try  
 {  
 using (StreamWriter sw = new StreamWriter(path, false))  
 {  
 for (int i = 0; i < patient.Length; i++)  
 sw.WriteLine($"Пациент {i + 1}: " + $"Имя —   
 {patient[i].LastName} ; " +  
 $"Возраст — {patient[i].Age} ; " + $"Пол —   
 {patient[i].Pol} ; " +  
 $"Город — {patient[i].City} ; " + $"Диагноз   
 — {patient[i].Diagnos} ; ");  
 }  
 }  
 catch (Exception e)  
 {  
 Console.WriteLine(e.Message);  
 }  
 }  
 else  
 Console.WriteLine("\nСначала добавьте минимум одного пациента\n");  
}  
  
void GetToFileThisList(Person[] patient)  
{  
 ConsoleKeyInfo key;  
 Console.Write("Введите \"Y\", чтобы сохранить в файл\n\"N\", чтобы не   
 сохранять");  
 do  
 {  
 key = Console.ReadKey();  
 } while(key.Key != ConsoleKey.Y || key.Key != ConsoleKey.N);  
 if (key.Key == ConsoleKey.Y)  
 SaveFile(patient);  
}  
  
void WorkWithPatients(ref Person[] patient, ref ConsoleKeyInfo key)  
{  
 var searchPatients = new Person[0];  
 string name = "", pol = "", city = "", diagnos = "";  
 sbyte age = 0;  
 switch (key.Key)  
 {  
 case ConsoleKey.A: //Add  
 Console.Clear();  
 AddInfo(ref name, ref age, ref pol, ref city, ref diagnos);  
 if (IsNotSimilarPatients(patient, name, age, pol, city,   
 diagnos))  
 {  
 Resize(ref patient, patient.Length + 1);  
 patient[patient.GetUpperBound(0)] = new Person(name, age,   
 pol, city, diagnos);   
 }  
 else  
 Console.WriteLine("Пациент с такими данными уже   
 существует");  
 MessageAfterAction();  
 InfoForHelp();  
 break;  
 case ConsoleKey.C: //Change  
 Console.Clear();  
 if (patient.Length > 0)  
 {  
 Console.Write("\nВведите номер пациента, которого нужно   
 изменить: " + $"в диапазоне от {patient.GetLowerBound(0) + 1}   
 до {patient.Length} включительно");  
 int inputIndex = InputIndexForChange(patient);  
 AddInfo(ref name, ref age, ref pol, ref city, ref diagnos);  
 if (IsNotSimilarPatients(patient, name, age, pol, city,   
 diagnos) ^ IsNotChanged(inputIndex - 1, patient, name, age,   
 pol, city, diagnos))  
 patient[inputIndex - 1] = new Person(name, age, pol,   
 city, diagnos);  
 else  
 Console.WriteLine("Пациент с такими данными уже   
 существует");  
 }  
 else  
 Console.WriteLine("\nСначала добавьте минимум одного   
 пациента\n");  
 MessageAfterAction();  
 InfoForHelp();  
 break;  
 case ConsoleKey.D: //Delete  
 Console.Clear();  
 if (patient.Length > 0)  
 {  
 Console.Write("\nВведите номер пациента, которого нужно   
 удалить: ");  
 int inputIndex = InputIndexForChange(patient);  
 ResizeAfterDelete(ref patient, inputIndex - 1);  
 }  
 else  
 Console.WriteLine("\nСначала добавьте минимум одного   
 пациента\n");  
 MessageAfterAction();  
 break;  
 case ConsoleKey.F: //FindCount  
 Console.Clear();  
 Console.WriteLine($"\nКоличество людей из других городов равно:   
 {NumberOfPatientsFromOtherCities(patient)}");  
 MessageAfterAction();  
 InfoForHelp();  
 break;  
 case ConsoleKey.L: //List  
 Console.Clear();  
 if (patient.Length > 0)  
 {  
 Console.WriteLine("\nБудет произведен поиск по критериям   
 минимального возраста и введенного диагноза");  
 age = AddAgeOfPatient();  
 diagnos = AddDiagnosOfPatient();  
 for (int i = 0; i < patient.Length; i++)  
 if (patient[i].Age > age - 1 &   
 patient[i].Diagnos.Equals(diagnos))  
 {  
 Resize(ref searchPatients, searchPatients.Length +   
 1);  
 searchPatients[searchPatients.GetUpperBound(0)] = new   
 Person(name, age, pol, city, diagnos);  
 }  
 if (searchPatients.Length > 0)  
 {  
 Console.WriteLine("Ваш найденный список пациентов:");  
 ShowList(searchPatients);  
 GetToFileThisList(searchPatients);  
 }  
 else  
 Console.WriteLine("Нужных пациентов не найдено");  
 }  
 else  
 Console.WriteLine("\nСначала добавьте минимум одного   
 пациента\n");  
 searchPatients = new Person[0];  
 MessageAfterAction();  
 InfoForHelp();  
 break;  
 case ConsoleKey.O: //OpenFile  
 Console.Clear();  
 OpenFile(ref patient);  
 MessageAfterAction();  
 InfoForHelp();  
 break;  
 case ConsoleKey.S: //SaveFile  
 Console.Clear();  
 SaveFile(patient);  
 MessageAfterAction();  
 InfoForHelp();  
 break;  
 case ConsoleKey.Enter: //ShowList  
 Console.Clear();  
 if (patient.Length > 0)  
 {  
 Console.WriteLine("Ваш список пациентов на данный момент:");  
 ShowList(patient);  
 }  
 else  
 Console.WriteLine("У вас нет пациентов на данный момент.   
 Добавьте их в главном меню.");  
 MessageAfterAction();  
 InfoForHelp();  
 break;  
 case ConsoleKey.H: //Help  
 Console.Clear();  
 InfoInMainMenu();  
 break;  
 }   
}  
  
Console.WriteLine("Пациенты глазной клиники");  
ConsoleKeyInfo key;  
InfoInMainMenu();  
var patient = new Person[0];  
do  
{  
 key = Console.ReadKey();  
 WorkWithPatients(ref patient, ref key);  
} while (key.Key != ConsoleKey.E); //Exit

Результат на **Delphi**:



Результат на **C#**:



**Блок-cхема**:

