УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №5.1

по предмету «Основы алгоритмизации и программирования»

Вариант 4

Выполнил:

Бражалович А. И.

Гр. 351004

Проверил:

Данилова Г. В.

Минск 2024

**Задание:**

Разработать программу решения задач с использованием процедур и функций.

1. Списки. Разработать программу работу с двусвязным списком. Программа должна содержать следующие процедуры, вызываемые из меню:

* построение пустого списка;
* добавление нового элемента;
* удаление указанного элемента;
* просмотр списка в прямом и обратном направлении.

**Код программы Delphi:**

Unit MainForm;

Interface

Uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Menus, Instruction, Developer,

Vcl.StdCtrls, Vcl.ExtDlgs, Vcl.Grids;

Type

TEStringGrid = Class(TStringGrid);

TArr = Array Of Integer;

ERRORS\_LIST = (CORRECT, NOT\_READABLE, NOT\_WRITEABLE, FILE\_EMPTY, LINE\_ERR,

NAME\_ERR, NUMBER\_ERR);

TMainTaskForm = Class(TForm)

MainFormMenu: TMainMenu;

FileMenu: TMenuItem;

InstructionMenu: TMenuItem;

DeveloperMenu: TMenuItem;

OpenMenu: TMenuItem;

SaveMenu: TMenuItem;

N1: TMenuItem;

QuitMenu: TMenuItem;

TaskLabel: TLabel;

OpenFile: TOpenDialog;

SaveTextFile: TSaveTextFileDialog;

StringGrid: TStringGrid;

AddButton: TButton;

DeleteButton: TButton;

ReverseButton: TButton;

StarightButton: TButton;

Procedure DeveloperMenuClick(Sender: TObject);

Procedure InstructionMenuClick(Sender: TObject);

Procedure GetDataFromFile(Var F: TextFile; Sender: TObject);

Function FileReading(Var F: TextFile): ERRORS\_LIST;

Procedure QuitMenuClick(Sender: TObject);

Procedure FormCloseQuery(Sender: TObject; Var CanClose: Boolean);

Procedure SaveMenuClick(Sender: TObject);

Procedure OpenMenuClick(Sender: TObject);

Function FormHelp(Command: Word; Data: NativeInt;

Var CallHelp: Boolean): Boolean;

Procedure AddButtonClick(Sender: TObject);

Procedure FormCreate(Sender: TObject);

Procedure DrawGrid(Var ListGrid: TStringGrid);

Procedure DeleteButtonClick(Sender: TObject);

Procedure ReverseButtonClick(Sender: TObject);

Procedure StarightButtonClick(Sender: TObject);

Private

{ Private declarations }

Public

{ Public declarations }

End;

Const

ERRORS: Array [ERRORS\_LIST] Of String = ('', 'Файл закрыт для чтения!',

'Файл закрыт для записи!', 'Файл пуст!', 'Неверное число строк в файле',

'Неверное имя пользователя!', 'Введён неверный номер!');

DIGITS = ['0' .. '9'];

NO\_ZERO\_DIGITS = ['1' .. '9'];

BACKSPACE = #8;

NONE = #0;

MIN\_N = 1;

MAX\_N = 100;

MIN\_X = -100;

MAX\_X = 100;

MAX\_SIGNS = 4;

LINES = 2;

ALPHABET = ['A' .. 'Z', 'a' .. 'z'];

Var

MainTaskForm: TMainTaskForm;

IsEdited: Boolean = False;

Saved: Boolean = True;

Implementation

Uses

AddContact;

// DoubleLinkedList;

Procedure AddNewContact(Name, Number: String; Var StringGrid: TStringGrid);

External 'DoubleLinkedList.dll';

Procedure DeleteContact(Place: Integer); External 'DoubleLinkedList.dll';

Procedure PrintUpDownList(Var ListGrid: TStringGrid);

External 'DoubleLinkedList.dll';

Procedure PrintDownUpList(Var ListGrid: TStringGrid);

External 'DoubleLinkedList.dll';

Procedure ClearList(); External 'DoubleLinkedList.dll';

{$R \*.dfm}

Var

PerformCloseQuery: Boolean = True;

CtrlPressed: Boolean = False;

Procedure TMainTaskForm.DeleteButtonClick(Sender: TObject);

Var

Confirmation: Integer;

Begin

If StringGrid.Row > 0 Then

Begin

Confirmation := Application.MessageBox

('Вы действительно хотите удалить телефон?', 'Удаление',

MB\_YESNO + MB\_ICONQUESTION + MB\_DEFBUTTON2);

If Confirmation = IDYES Then

Begin

DeleteContact(StringGrid.Row);

StringGrid.Cells[0, StringGrid.RowCount - 1] := '';

StringGrid.RowCount := StringGrid.RowCount - 1;

PrintUpDownList(StringGrid);

DrawGrid(StringGrid);

End;

End

Else

Application.MessageBox('Не выбрано редактируемое поле!', 'Ошибка',

MB\_OK + MB\_ICONERROR);

End;

Procedure TMainTaskForm.DeveloperMenuClick(Sender: TObject);

Var

DeveloperForm: TDeveloperForm;

Begin

DeveloperForm := TDeveloperForm.Create(Self);

DeveloperForm.ShowModal;

DeveloperForm.Free;

End;

Function TMainTaskForm.FormHelp(Command: Word; Data: NativeInt;

Var CallHelp: Boolean): Boolean;

Begin

CallHelp := False;

InstructionMenuClick(Self)

End;

Procedure TMainTaskForm.InstructionMenuClick(Sender: TObject);

Var

InstructionForm: TInstructionForm;

Begin

InstructionForm := TInstructionForm.Create(Self);

InstructionForm.ShowModal;

InstructionForm.Free;

End;

Function IsReadable(Var F: TextFile): ERRORS\_LIST;

Var

ERRORS: ERRORS\_LIST;

Begin

ERRORS := CORRECT;

Try

Try

Reset(F);

Finally

CloseFile(F);

End;

Except

ERRORS := NOT\_READABLE;

End;

IsReadable := ERRORS;

End;

Function CheckNumOfLines(Var F: TextFile): ERRORS\_LIST;

Var

I: Integer;

Str: String;

Error: ERRORS\_LIST;

Begin

I := 0;

Str := '';

Error := CORRECT;

Reset(F);

While Not EOF(F) Do

Begin

Readln(F, Str);

Inc(I);

End;

CloseFile(F);

If (I Mod 2) <> 0 Then

Error := LINE\_ERR;

CheckNumOfLines := Error;

End;

Function CheckFileData(Var F: TextFile): ERRORS\_LIST;

Var

Error: ERRORS\_LIST;

Value: Integer;

Num, I: Integer;

Number, Name: String;

Begin

Error := CORRECT;

Reset(F);

While (Not EOF(F)) And (Error = CORRECT) Do

Begin

Readln(F, Name);

If (Length(Name) > 15) Or (Trim(Name) = '') Then

Error := NAME\_ERR;

Readln(F, Number);

If (Length(Number) <> 9) Or (Not(TryStrToInt(Number, Value))) Then

Error := NUMBER\_ERR;

End;

CloseFile(F);

CheckFileData := Error;

End;

Procedure ClearGrid(Grid: TStringGrid);

Var

I: Integer;

Begin

For I := 1 To Grid.ColCount Do

Begin

Grid.Cells[0, I] := '';

Grid.Cells[1, I] := '';

End;

Grid.RowCount := 1;

End;

Procedure TMainTaskForm.GetDataFromFile(Var F: TextFile; Sender: TObject);

Var

Number, Name: String;

I: Integer;

Begin

I := 1;

Reset(F);

ClearList();

ClearGrid(StringGrid);

While Not EOF(F) Do

Begin

Readln(F, Name);

Readln(F, Number);

AddNewContact(Name, Number, StringGrid);

StringGrid.RowCount := StringGrid.RowCount + 1;

End;

CloseFile(F);

PrintUpDownList(StringGrid);

DrawGrid(StringGrid);

End;

Function TMainTaskForm.FileReading(Var F: TextFile): ERRORS\_LIST;

Var

ERRORS: ERRORS\_LIST;

Begin

ERRORS := CORRECT;

Reset(F);

If EOF(F) Then

ERRORS := FILE\_EMPTY;

CloseFile(F);

If (ERRORS = CORRECT) Then

ERRORS := CheckNumOfLines(F);

If (ERRORS = CORRECT) Then

ERRORS := CheckFileData(F);

If (ERRORS = CORRECT) Then

Begin

GetDataFromFile(F, Self);

End;

FileReading := ERRORS;

End;

Procedure TMainTaskForm.OpenMenuClick(Sender: TObject);

Var

Error: ERRORS\_LIST;

F: TextFile;

Num, FileName: String;

Begin

If OpenFile.Execute Then

Begin

FileName := OpenFile.FileName;

AssignFile(F, FileName);

Error := IsReadable(F);

If Error = CORRECT Then

Begin

Error := FileReading(F);

End;

If Error <> CORRECT Then

Application.MessageBox(PWideChar(ERRORS[Error]), 'Ошибка',

MB\_OK Or MB\_ICONINFORMATION);

End;

End;

Procedure FillGrid(Grid: TStringGrid);

Var

I, J: Integer;

NumArr: TArr;

Begin

Grid.Cells[0, 0] := 'Имя';

Grid.Cells[1, 0] := 'Номер';

Grid.ColWidths[0] := Grid.DefaultColWidth \* 2;

Grid.ColWidths[1] := Grid.DefaultColWidth \* 2;

Grid.Width := Grid.DefaultColWidth \* 4;

Grid.Height := Grid.DefaultRowHeight + 4;

Grid.Enabled := True;

Grid.ColCount := 2;

Grid.RowCount := 1;

End;

Procedure TMainTaskForm.AddButtonClick(Sender: TObject);

Var

AddContactForm: TAddContactForm;

Begin

If StringGrid.RowCount <= MAX\_N Then

Begin

AddContactForm := TAddContactForm.Create(Self);

AddContactForm.ShowModal;

AddContactForm.Free;

If IsEdited Then

Begin

DrawGrid(StringGrid);

IsEdited := False;

End;

End

Else

Application.MessageBox('Слишком много номеров!', 'Ошибка',

MB\_OK + MB\_ICONERROR);

End;

Procedure TMainTaskForm.SaveMenuClick(Sender: TObject);

Var

Error: ERRORS\_LIST;

F: TextFile;

FileName: String;

I: Integer;

Begin

If SaveTextFile.Execute Then

Begin

FileName := SaveTextFile.FileName;

FileName := ChangeFileExt(FileName, '.txt');

AssignFile(F, FileName);

If FileExists(FileName) Then

Begin

Error := IsReadable(F);

If Error = CORRECT Then

Begin

Rewrite(F);

For I := 0 To StringGrid.RowCount - 1 Do

Begin

Write(F, StringGrid.Cells[0, I], ' ');

Writeln(F, Stringgrid.Cells[1, I]);

End;

CloseFile(F);

Saved := True;

End;

If Error <> CORRECT Then

Begin

Application.MessageBox(PWideChar(ERRORS[Error]), 'Ошибка',

MB\_OK Or MB\_ICONINFORMATION);

Saved := False;

End;

End

Else

Begin

Rewrite(F);

For I := 0 To StringGrid.RowCount - 1 Do

Begin

Write(F, StringGrid.Cells[0, I], ' ');

Writeln(F, Stringgrid.Cells[1, I]);

End;

CloseFile(F);

Saved := True;

End;

End;

End;

Procedure TMainTaskForm.FormCloseQuery(Sender: TObject; Var CanClose: Boolean);

Var

Confirmation: Integer;

Begin

If PerformCloseQuery Then

Begin

If (Saved = False) Then

Begin

Confirmation := Application.MessageBox

('Вы не сохранили файл, хотите ли сохранить?', 'Выход',

MB\_YESNOCANCEl + MB\_ICONQUESTION + MB\_DEFBUTTON2);

Case Confirmation Of

MrYes:

Begin

SaveMenuClick(Sender);

If Saved = True Then

CanClose := True

Else

FormCloseQuery(Sender, CanClose);

End;

MrNo:

CanClose := True;

MrCancel:

CanClose := False;

End;

End

Else

Begin

Confirmation := Application.MessageBox

('Вы действительно хотите выйти?', 'Выход',

MB\_YESNO + MB\_ICONQUESTION + MB\_DEFBUTTON2);

CanClose := Confirmation = IDYES;

End;

End;

End;

Procedure TMainTaskForm.DrawGrid(Var ListGrid: TStringGrid);

Begin

ListGrid.ColCount := 2;

ListGrid.ColWidths[0] := ListGrid.DefaultColWidth \* 2;

ListGrid.ColWidths[1] := ListGrid.DefaultColWidth \* 2;

ListGrid.Cells[1, 0] := 'Номер';

ListGrid.Cells[0, 0] := 'Имя';

If ListGrid.RowCount > 10 Then

Begin

ListGrid.ScrollBars := TScrollStyle.SsVertical;

ListGrid.Height := 400

End

Else

Begin

ListGrid.ScrollBars := TScrollStyle.SsNone;

ListGrid.Height := (ListGrid.DefaultRowHeight + ListGrid.GridLineWidth)

\* ListGrid.RowCount + 5;

End;

SaveMenu.Enabled := StringGrid.RowCount > 1;

Saved := ListGrid.RowCount = 1;

ListGrid.Width := ListGrid.DefaultColWidth \* 4;

End;

Procedure TMainTaskForm.FormCreate(Sender: TObject);

Begin

FillGrid(StringGrid);

End;

Procedure TMainTaskForm.QuitMenuClick(Sender: TObject);

Var

Confirmation: Integer;

Begin

PerformCloseQuery := False;

If (Saved = False) Then

Begin

Confirmation := Application.MessageBox

('Вы не сохранили файл, хотите ли сохранить?', 'Выход',

MB\_YESNOCANCEl + MB\_ICONQUESTION + MB\_DEFBUTTON2);

Case Confirmation Of

MrYes:

Begin

SaveMenuClick(Sender);

If Saved = True Then

Close

Else

QuitMenuClick(Sender);

End;

MrNo:

Close;

End;

End

Else

Begin

Confirmation := Application.MessageBox('Вы действительно хотите выйти?',

'Выход', MB\_YESNO + MB\_ICONQUESTION + MB\_DEFBUTTON2);

If Confirmation = IDYES Then

Close;

End;

PerformCloseQuery := True;

End;

Procedure TMainTaskForm.ReverseButtonClick(Sender: TObject);

Begin

// ClearGrid(StringGrid);

PrintDownUpList(StringGrid);

DrawGrid(StringGrid);

End;

Procedure TMainTaskForm.StarightButtonClick(Sender: TObject);

Begin

PrintUpDownList(StringGrid);

DrawGrid(StringGrid);

End;

End.

Unit AddContact;

Interface

Uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics,

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

Type

TAddContactForm = Class(TForm)

StartLabel: TLabel;

NumberEdit: TEdit;

NameEdit: TEdit;

NameLabel: TLabel;

NemberLabel: TLabel;

AddButton: TButton;

CancelButton: TButton;

Procedure CancelButtonClick(Sender: TObject);

Procedure FormKeyDown(Sender: TObject; Var Key: Word;

Shift: TShiftState);

Procedure NumberEditChange(Sender: TObject);

Procedure NameEditChange(Sender: TObject);

Procedure FormCreate(Sender: TObject);

Procedure NumberEditKeyDown(Sender: TObject; Var Key: Word;

Shift: TShiftState);

Procedure NumberEditKeyPress(Sender: TObject; Var Key: Char);

Procedure NameEditKeyDown(Sender: TObject; Var Key: Word;

Shift: TShiftState);

Procedure NameEditKeyPress(Sender: TObject; Var Key: Char);

Procedure AddButtonClick(Sender: TObject);

Procedure NumberEditContextPopup(Sender: TObject; MousePos: TPoint;

Var Handled: Boolean);

Procedure NameEditContextPopup(Sender: TObject; MousePos: TPoint;

Var Handled: Boolean);

Function FormHelp(Command: Word; Data: NativeInt;

Var CallHelp: Boolean): Boolean;

Private

{ Private declarations }

Public

{ Public declarations }

End;

Var

AddContactForm: TAddContactForm;

Const

ALPHABET = ['А' .. 'Я', 'а' .. 'я'];

NUMBERS = ['0' .. '9'];

NUMBER\_LENGTH = 9;

Implementation

Uses

MainForm;

Procedure AddNewContact(Name, Number: String; Var StringGrid: TStringGrid);

External 'DoubleLinkedList.dll';

Procedure PrintUpDownList(Var ListGrid: TStringGrid);

External 'DoubleLinkedList.dll';

{$R \*.dfm}

Var

CanAdd: Boolean;

Procedure CenterFormOnScreen(AddContactForm: TAddContactForm);

Begin

AddContactForm.Left := (Screen.Width - AddContactForm.Width) Div 2;

AddContactForm.Top := (Screen.Height - AddContactForm.Height) Div 2;

End;

Procedure CheckComboButtons(Var Key: Char; Edit: TEdit;

Const Chariki: TSysCharSet);

Begin

If (Key = #22) Or ((Key = 'v') And (GetKeyState(VK\_CONTROL) < 0)) Then

Key := #0;

If Not CharInSet(Key, Chariki) And (Key <> #8) Then

Key := #0;

End;

Procedure CheckShftAndArrows(Var Key: Word; Shift: TShiftState);

Begin

If (Key = VK\_INSERT) And (Shift = [SsShift]) Then

Key := 0;

If (Key = VK\_LEFT) Or (Key = VK\_UP) Then

Key := 0

Else If (Key = VK\_RIGHT) Or (Key = VK\_DOWN) Then

Key := 0;

End;

Procedure CheckRange(Var Key: Char; Edit: TEdit; Const MAX: Integer);

Var

BuffString: String;

Begin

If (Length(Edit.Text) >= MAX) And (Key <> #8) Then

Key := #0;

End;

Procedure TAddContactForm.AddButtonClick(Sender: TObject);

Begin

AddNewContact(NameEdit.Text, NumberEdit.Text, MainTaskForm.StringGrid);

MainTaskForm.StringGrid.RowCount := MainTaskForm.StringGrid.RowCount + 1;

PrintUpDownList(MainTaskForm.StringGrid);

IsEdited := True;

Saved := False;

Close;

End;

Procedure TAddContactForm.CancelButtonClick(Sender: TObject);

Begin

NameEdit.Text := '';

NumberEdit.Text := '';

Close;

End;

Procedure TAddContactForm.FormCreate(Sender: TObject);

Begin

AddButton.Enabled := False;

End;

Function TAddContactForm.FormHelp(Command: Word; Data: NativeInt;

Var CallHelp: Boolean): Boolean;

Begin

CallHelp := False;

End;

Procedure TAddContactForm.FormKeyDown(Sender: TObject; Var Key: Word;

Shift: TShiftState);

Begin

If Key = VK\_ESCAPE Then

Close;

End;

Procedure TAddContactForm.NameEditChange(Sender: TObject);

Begin

If (NameEdit.Text <> '') And (NumberEdit.Text <> '') Then

AddButton.Enabled := True

Else

AddButton.Enabled := False;

End;

Procedure TAddContactForm.NameEditContextPopup(Sender: TObject;

MousePos: TPoint; Var Handled: Boolean);

Begin

Handled := True;

End;

Procedure TAddContactForm.NameEditKeyDown(Sender: TObject; Var Key: Word;

Shift: TShiftState);

Begin

CheckShftAndArrows(Key, Shift);

End;

Procedure TAddContactForm.NameEditKeyPress(Sender: TObject; Var Key: Char);

Begin

CheckRange(Key, NameEdit, 15);

End;

Procedure TAddContactForm.NumberEditChange(Sender: TObject);

Begin

If (NameEdit.Text <> '') And (Length(NumberEdit.Text) = 9) Then

AddButton.Enabled := True

Else

AddButton.Enabled := False;

End;

Procedure TAddContactForm.NumberEditContextPopup(Sender: TObject;

MousePos: TPoint; Var Handled: Boolean);

Begin

Handled := True;

End;

Procedure TAddContactForm.NumberEditKeyDown(Sender: TObject; Var Key: Word;

Shift: TShiftState);

Begin

CheckShftAndArrows(Key, Shift);

End;

Procedure TAddContactForm.NumberEditKeyPress(Sender: TObject; Var Key: Char);

Begin

CheckComboButtons(Key, NumberEdit, NUMBERS);

CheckRange(Key, NumberEdit, NUMBER\_LENGTH);

End;

End.

Unit Developer;

Interface

Uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics,

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

Type

TDeveloperForm = Class(TForm)

DeveloperLabel: TLabel;

Procedure Button1Click(Sender: TObject);

Procedure FormCreate(Sender: TObject);

Function FormHelp(Command: Word; Data: NativeInt;

Var CallHelp: Boolean): Boolean;

Private

{ Private declarations }

Public

{ Public declarations }

End;

Var

DeveloperForm: TDeveloperForm;

Implementation

{$R \*.dfm}

Procedure TDeveloperForm.Button1Click(Sender: TObject);

Begin

Close;

End;

Procedure CenterDeveloperFormOnScreen(DeveloperForm: TDeveloperForm);

Begin

DeveloperForm.Left := (Screen.Width - DeveloperForm.Width) Div 2;

DeveloperForm.Top := (Screen.Height - DeveloperForm.Height) Div 2;

End;

Procedure TDeveloperForm.FormCreate(Sender: TObject);

Begin

CenterDeveloperFormOnScreen(Self);

DeveloperLabel.Caption := 'Разработчик: Бражалович Александр Иванович' +

#13#10 + 'Группа: 351005' + #13#10 + 'Tg: @Sunn4es';

DeveloperLabel.Update;

End;

Function TDeveloperForm.FormHelp(Command: Word; Data: NativeInt;

Var CallHelp: Boolean): Boolean;

Begin

CallHelp := False;

End;

End.

Unit Instruction;

Interface

Uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics,

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

Type

TInstructionForm = Class(TForm)

InstructionLabel: TLabel;

Procedure CloseButtonClick(Sender: TObject);

Procedure FormCreate(Sender: TObject);

Function FormHelp(Command: Word; Data: NativeInt;

Var CallHelp: Boolean): Boolean;

Private

{ Private declarations }

Public

{ Public declarations }

End;

Var

InstructionForm: TInstructionForm;

Implementation

{$R \*.dfm}

Procedure CenterFormOnScreen(InstructionForm: TInstructionForm);

Begin

InstructionForm.Left := (Screen.Width - InstructionForm.Width) Div 2;

InstructionForm.Top := (Screen.Height - InstructionForm.Height) Div 2;

End;

Procedure TInstructionForm.CloseButtonClick(Sender: TObject);

Begin

Close;

End;

Procedure TInstructionForm.FormCreate(Sender: TObject);

Begin

CenterFormOnScreen(Self);

InstructionLabel.Caption :=

'1. Чтобы добавить новый контакт нужно нажать на кнопку "Добавить контакт".'

+ #13#10 +

'2. Для удаления нужно нажать на нужный контакт в списке и нажать кнопку "Удалить"'

+ #13#10 +

'3. Для просмотра списка в обратном порядке нужно нажать на кнопку "Обратный порядок"'

+ #13#10 + '' + #13#10 + 'Для загрузки контактов из текстового файла:' +

#13#10 + '1. На первой строке имя контакта.' + #13#10 +

'2. На второй строке номер телефона (без +375).';

End;

Function TInstructionForm.FormHelp(Command: Word; Data: NativeInt;

Var CallHelp: Boolean): Boolean;

Begin

CallHelp := False;

End;

End.

library DoubleLinkedList;

uses

Vcl.Grids;

Type

TNumString = String[10];

TNameString = String[16];

TDoubleLinkedList = ^TNode;

TNode = Record

Name: TNameString;

Number: TNumString;

Next: TDoubleLinkedList;

Prev: TDoubleLinkedList;

End;

Var

Head: TDoubleLinkedList = nil;

Tail: TDoubleLinkedList = nil;

Procedure AddNewContact(Name, Number: String; Var StringGrid: TStringGrid);

Var

NewItem: TDoubleLinkedList;

Begin

New(NewItem);

NewItem.Name := Name;

NewItem.Number := Number;

NewItem.Prev := Tail;

NewItem.Next := nil;

If Head = nil then

Head := NewItem

Else

Tail^.Next := NewItem;

Tail := NewItem;

End;

Procedure DeleteContact(Place: Integer);

Var

CurrCont, PrevCont, NextCont: TDoubleLinkedList;

Counter: Integer;

Begin

CurrCont := Head;

Counter := 1;

while (Counter < Place) do

Begin

CurrCont := CurrCont^.Next;

Inc(Counter);

End;

PrevCont := CurrCont^.Prev;

NextCont := CurrCont^.Next;

if PrevCont <> nil then

PrevCont^.Next := NextCont

Else

Head := NextCont;

if NextCont <> nil then

NextCont^.Prev := PrevCont

Else

Tail := CurrCont^.Prev;

Dispose(CurrCont);

End;

Procedure PrintUpDownList (Var ListGrid: TStringGrid);

Var

CurrCont: TDoubleLinkedList;

Count: Integer;

Begin

CurrCont := Head;

Count := 1;

While (CurrCont <> nil) Do

Begin

ListGrid.Cells[0, Count] := CurrCont.Name;

ListGrid.Cells[1, Count] := Concat('+375', CurrCont.Number);

CurrCont := CurrCont^.Next;

Inc(Count);

End;

End;

Procedure PrintDownUpList(Var ListGrid: TStringGrid);

Var

CurrCont: TDoubleLinkedList;

Count: Integer;

Begin

CurrCont := Tail;

Count := 1;

while CurrCont <> nil do

Begin

ListGrid.Cells[0, Count] := CurrCont.Name;

ListGrid.Cells[1, Count] := Concat('+375', CurrCont.Number);

CurrCont := CurrCont^.Prev;

Inc(Count);

End;

End;

Procedure ClearList();

Var

CurrCont, NextCont: TDoubleLinkedList;

Begin

CurrCont := Head;

while CurrCont <> nil do

Begin

NextCont := CurrCont.Next;

Dispose(CurrCont);

CurrCont := NextCont;

End;

Head := nil;

Tail := nil;

End;

Exports AddNewContact, PrintUpDownList, PrintDownUpList, DeleteContact, ClearList;

Begin

End.

**Код программы Java:**

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.Scanner;

public class Main {

enum ErrCode {

CORRECT,

FILE\_NOT\_EXIST,

NOT\_TXT,

EMPTY\_LIST,

INPUT\_ERR,

NOT\_READABLE,

FILE\_EMPTY,

NOT\_WRITEABLE,

RANGE\_ERR

}

static final String[] ERRORS = {"Удача",

"Такого файла не существует!",

"Файл не .txt!",

"Список контактов пуст!",

"Проверьте корректность ввода данных!",

"Файл закрыт для чтения!",

"Файл пуст!",

"Файл закрыт для записи!",

"Данные не входят в диапазон"

};

static final String INSTRUCTION = "\n1. Номера телефонов должны начинаться с кода.\n" +

"2. Чтобы добавлять контанты из файла нужно записывать имя на первой строке и номер с кода на второй.\n";

enum ChooseAction {

addToList("Добавить контакт"),

addToListFromFile("Добавить контакт(ы) из файла"),

deleteContact("Удалить контакт"),

printStraight("Вывести список контактов в прямом порядке"),

printReverse("Вывести список контактов в обратном порядке"),

saveList("Сохранить контакты в файл"),

exitProg("Заверишть программу");

private final String info;

ChooseAction (String inf) {

this.info = inf;

}

private String getInf(){

return this.ordinal() + ") " + this.info;

}

}

public static boolean doMenu (Scanner input) {

boolean close = false;

ErrCode error;

System.out.print("Введите действие: ");

ChooseAction option = getChoice(input);

System.out.println();

switch (option) {

case addToList -> {

createContact(input);

}

case addToListFromFile -> {

addContFromFile(input);

}

case deleteContact -> {

deleteContact(input);

}

case printStraight -> {

if (countOfContacts != 0) {

printUpDownListConsole();

} else {

printError(ErrCode.EMPTY\_LIST);

}

System.out.println();

}

case printReverse -> {

if (countOfContacts != 0) {

printDownUpListConsole();

} else {

printError(ErrCode.EMPTY\_LIST);

}

System.out.println();

}

case saveList -> {

saveUpDownList(input);

}

case exitProg -> {

close = true;

}

}

return close;

}

static void printMenu() {

ChooseAction[] choices = ChooseAction.values();

for (ChooseAction choice : choices) {

System.out.println(choice.getInf());

}

}

public static void addContFromFile (Scanner input) {

File file;

ErrCode error = ErrCode.CORRECT;

int number = 0;

String numberStr = "";

String name = "";

String nameBuf;

file = fileReading(input);

try (Scanner scanFile = new Scanner(file)) {

while ((error == ErrCode.CORRECT)) {

nameBuf = scanFile.nextLine();

if ((nameBuf.length() > 15) || (nameBuf.isEmpty())){

error = ErrCode.INPUT\_ERR;

} else {

name = nameBuf;

}

if (error == ErrCode.CORRECT) {

numberStr = scanFile.nextLine();

if ((numberStr.length() != 9) || (numberStr.isEmpty())) {

error = ErrCode.INPUT\_ERR;

} else {

try {

number = Integer.parseInt(numberStr);

} catch (NumberFormatException e) {

error = ErrCode.INPUT\_ERR;

}

}

}

if (error == ErrCode.CORRECT) {

addNewContact(name, numberStr);

System.out.println("Контакт добавлен успешно.");

}

}

} catch (Exception e) {

printError(ErrCode.NOT\_READABLE);

}

}

public static void deleteContact (Scanner input) {

int place = 0;

if (countOfContacts == 0) {

printError(ErrCode.EMPTY\_LIST);

} else {

System.out.print("Введите порядковый номер контакта, который хотите удалить (" + countOfContacts + "): ");

place = getNumConsole(input, 1, countOfContacts);

deleteContactFromList(place - 1);

System.out.println("Контакт успешно удалён.\n");

}

}

public static void printError (ErrCode error) {

System.out.println("\n" + ERRORS[error.ordinal()] + "\nПовторите попытку\n");

}

public static void createContact(Scanner input) {

String number;

String name;

name = getNameConsole(input, 15);

number = getNumberConsole(input);

addNewContact(name, number);

System.out.println("Контакт добавлен.\n");

}

public static String getNumberConsole (Scanner input) {

int number = 0;

String numberStr;

ErrCode error;

do {

error = ErrCode.CORRECT;

System.out.print("Введите номер телефона начная с кода: ");

numberStr = (input.nextLine());

if (numberStr.length() == 9) {

try {

number = Integer.parseInt(numberStr);

} catch (NumberFormatException e){

error = ErrCode.INPUT\_ERR;

}

} else {

error = ErrCode.INPUT\_ERR;

}

if (error != ErrCode.CORRECT) {

printError(error);

}

} while (error != ErrCode.CORRECT);

return numberStr;

}

public static ErrCode readOneNum(Scanner inputScanner, int[] numberArr, final int MIN, final int MAX) {

int number = 0;

ErrCode error;

error = ErrCode.CORRECT;

try {

number = Integer.parseInt(inputScanner.nextLine());

} catch (NumberFormatException e) {

error = ErrCode.INPUT\_ERR;

}

if (error == ErrCode.CORRECT && ((number < MIN) || (number > MAX)))

error = ErrCode.RANGE\_ERR;

numberArr[0] = error == ErrCode.CORRECT ? number : 0;

return error;

}

static int getNumConsole(Scanner input, final int MIN, final int MAX) {

ErrCode err;

int[] numberArr = {0};

do {

err = readOneNum(input, numberArr, MIN, MAX);

if (err != ErrCode.CORRECT) {

System.err.printf(ERRORS[err.ordinal()], MIN, MAX);

System.out.println("\nВведите снова");

}

} while (err != ErrCode.CORRECT);

return numberArr[0];

}

static String getNameConsole(Scanner input, final int MAX) {

ErrCode error;

String name;

do {

error = ErrCode.CORRECT;

System.out.print("Введите имя от 1 до " + MAX + " символов: ");

name = input.nextLine();

if (name.length() > MAX) {

error = ErrCode.INPUT\_ERR;

printError(error);

}

} while (error != ErrCode.CORRECT);

return name;

}

static ChooseAction getChoice(Scanner input) {

int choice;

int maxChoice = ChooseAction.values().length - 1;

choice = getNumConsole(input, 0, maxChoice);

return ChooseAction.values()[choice];

}

public static String readPath (Scanner inputScanner) {

String pathTofile = "";

ErrCode error;

do {

System.out.print("Введите путь к txt файлу: ");

pathTofile = inputScanner.nextLine();

if (pathTofile.isEmpty()) {

pathTofile = inputScanner.nextLine();

}

if (!pathTofile.endsWith(".txt")) {

error = ErrCode.NOT\_TXT;

} else {

error = ErrCode.CORRECT;

}

if (error != ErrCode.CORRECT)

printError(error);

} while (error != ErrCode.CORRECT);

return pathTofile;

}

public static File fileReading (Scanner inputScanner) {

ErrCode error;

String pathToFile = "";

File file;

do {

error = ErrCode.CORRECT;

pathToFile = readPath(inputScanner);

file = new File(pathToFile);

if (!file.exists())

error = ErrCode.FILE\_NOT\_EXIST;

if ((error == ErrCode.CORRECT) && (!file.canRead()))

error = ErrCode.NOT\_READABLE;

if ((error == ErrCode.CORRECT) && (file.length() == 0))

error = ErrCode.FILE\_EMPTY;

if (error != ErrCode.CORRECT)

printError(error);

} while (error != ErrCode.CORRECT);

return file;

}

public static File fileWriting(Scanner inputScanner) {

ErrCode error;

File file;

String pathToFile;

do {

pathToFile = readPath(inputScanner);

file = new File(pathToFile);

error = ErrCode.CORRECT;

if (!file.exists())

error = ErrCode.FILE\_NOT\_EXIST;

if ((error == ErrCode.CORRECT) && !file.canWrite())

error = ErrCode.NOT\_WRITEABLE;

if (error != ErrCode.CORRECT)

printError(error);

} while (error != ErrCode.CORRECT);

return file;

}

static class DoubleLinkedList {

String name;

String number;

DoubleLinkedList next;

DoubleLinkedList prev;

}

static DoubleLinkedList head = null;

static DoubleLinkedList tail = null;

static int countOfContacts = 0;

static void addNewContact (String name, String number) {

DoubleLinkedList newCont = new DoubleLinkedList();

newCont.name = name;

newCont.number = number;

newCont.next = null;

newCont.prev = tail;

if (head == null) {

head = newCont;

} else {

tail.next = newCont;

}

tail = newCont;

countOfContacts++;

}

static void deleteContactFromList(int place) {

DoubleLinkedList currCont = head;

DoubleLinkedList nextCont = new DoubleLinkedList();

DoubleLinkedList prevCont = new DoubleLinkedList();

int counter = 1;

while (counter < place) {

currCont = currCont.next;

counter++;

}

prevCont = currCont.prev;

nextCont = currCont.next;

if (prevCont != null) {

prevCont.next = nextCont;

} else {

head = nextCont;

}

if (nextCont != null) {

nextCont.prev = prevCont;

} else {

tail = currCont.prev;

}

currCont = null;

countOfContacts--;

}

static void printUpDownListConsole () {

DoubleLinkedList currCont = head;

int i = 1;

while (currCont != null) {

System.out.print(i + ". " + currCont.name + " ");

System.out.println("+375" + currCont.number);

currCont = currCont.next;

i++;

}

}

static void printDownUpListConsole () {

DoubleLinkedList currCont = tail;

int i = countOfContacts;

while (currCont != null) {

System.out.print(i + ". " + currCont.name + " ");

System.out.println("+375" + currCont.number);

currCont = currCont.prev;

i--;

}

}

static void saveUpDownList (Scanner inputScanner) {

File file;

DoubleLinkedList currCont = head;

int i = 1;

if (countOfContacts == 0) {

printError(ErrCode.EMPTY\_LIST);

} else {

file = fileWriting(inputScanner);

try (FileWriter writer = new FileWriter(file, true)) {

while (currCont != null) {

writer.write(i + ". " + currCont.name + " ");

writer.write("+375" + currCont.number + "\n");

currCont = currCont.next;

i++;

}

System.out.println("Контакты сохранены успешно.\n");

} catch (Exception e) {

printError(ErrCode.NOT\_WRITEABLE);

}

}

}

public static void main(String[] args) {

boolean isExit;

Scanner input = new Scanner(System.in);

System.out.println(INSTRUCTION);

do {

printMenu();

System.out.println();

isExit = doMenu(input);

} while (!isExit);

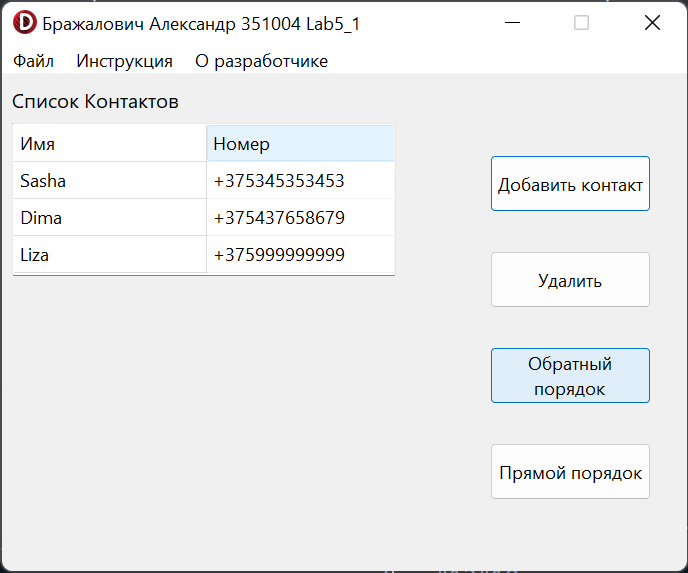
input.close();

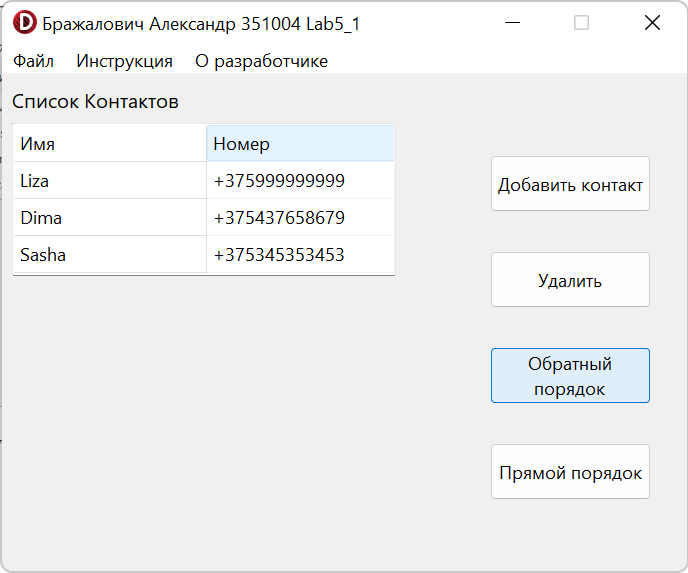
}

}

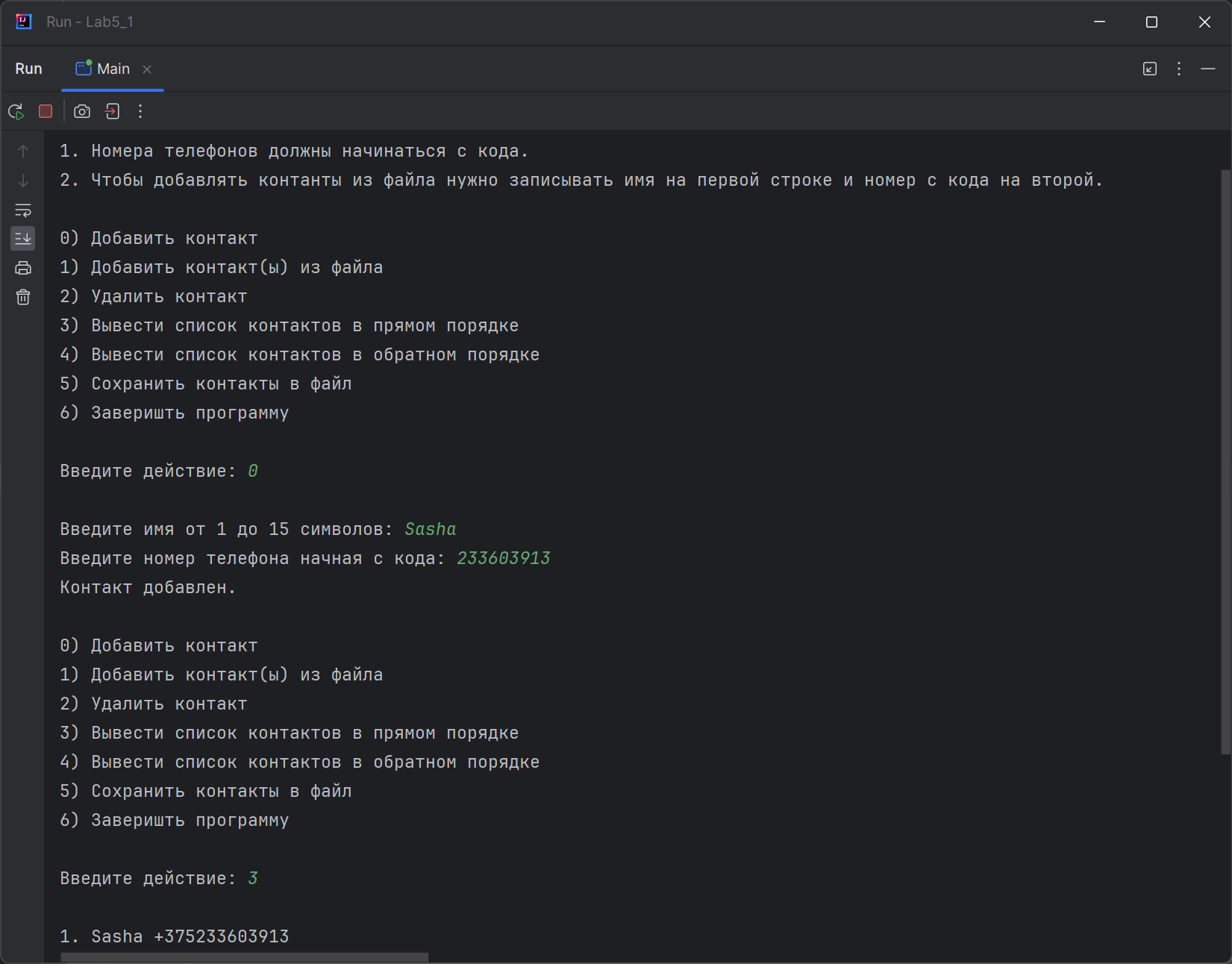
**Скриншоты:**

**Delphi:**

****

****

**Java:**



**Блок-схема:**

