УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №5.1

по предмету «Основы алгоритмизации и программирования»

Вариант 15

Выполнила:

Городко К. Е.

Гр. 351005

Проверила:

Данилова Г. В.

Минск 2024

**ЗАДАНИЕ:**

Разработать программу работы со стеком. Программа должна содержать следующие процедуры, вызываемые из меню:

- построение пустого стека;

- добавление элемента;

- удаление элемента;

- уничтожение стека.

**КОД ПРОГРАММЫ DELPHI:**

**Library LIBRARY5\_1;**

Type

TStackPointer = ^TStack;

TStack = Record

Num: Integer;

Next: TStackPointer;

End;

Var

Head: TStackPointer;

Procedure CreateStack();

Begin

New(Head);

Head.Next := Nil;

End;

Procedure StackPush(Num: Integer);

Var

NewElement: TStackPointer;

Begin

New(NewElement);

NewElement.Num := Num;

NewElement.Next := Head;

Head := NewElement;

End;

Function StackPop(): Integer;

Var

DeletedElement: TStackPointer;

Begin

DeletedElement := Head;

Head := DeletedElement.Next;

StackPop := DeletedElement.Num;

Dispose(DeletedElement);

End;

Procedure DestroyStack();

Begin

While (Head <> Nil) Do

StackPop();

Dispose(Head);

End;

{$R \*.res}

Exports CreateStack, StackPush, StackPop, DestroyStack;

Begin

End.

**Unit MainUnit;**

Interface

Uses

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

System.Classes, Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Menus,

Vcl.ExtCtrls, Vcl.StdCtrls, Vcl.ComCtrls, StackElementUnit, InstructionUnit,

DevInfoUnit;

Const

MAXELEMENTS = 30;

MINNUM = -999999;

MAXNUM = 999999;

Type

TMainForm = Class(TForm)

ButtonCreate: TButton;

ButtonDestroy: TButton;

ButtonPop: TButton;

ButtonPush: TButton;

LabelProgramInfo: TLabel;

ListViewStack: TListView;

MainMenu: TMainMenu;

MMFile: TMenuItem;

MMOpenFile: TMenuItem;

MMSaveFile: TMenuItem;

MMSeparator: TMenuItem;

MMExit: TMenuItem;

MMInstruction: TMenuItem;

MMDevInfo: TMenuItem;

OpenDialog: TOpenDialog;

SaveDialog: TSaveDialog;

Procedure ButtonCreateClick(Sender: TObject);

Procedure ButtonDestroyClick(Sender: TObject);

Procedure ButtonPopClick(Sender: TObject);

Procedure ButtonPushClick(Sender: TObject);

Procedure AddItemToListView(ListView: TListView; Element: Integer);

Procedure ClearListView(ListView: TListView);

Procedure ListViewOnChange(Num: Integer);

Procedure ListViewStackChange(Sender: TObject; Item: TListItem; Change:

TItemChange);

Procedure ListViewStackDeletion(Sender: TObject; Item: TListItem);

Procedure EnableComponents(IsEnabled: Boolean);

Procedure FormClose(Sender: TObject; Var Action: TCloseAction);

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

Procedure FormCreate(Sender: TObject);

Procedure FormResize(Sender: TObject);

Function FormHelp(Command: Word; Data: NativeInt; Var CallHelp: Boolean):

Boolean;

Procedure MMDevInfoClick(Sender: TObject);

Procedure MMExitClick(Sender: TObject);

Procedure MMInstructionClick(Sender: TObject);

Procedure MMOpenFileClick(Sender: TObject);

Procedure MMSaveFileClick(Sender: TObject);

End;

Var

MainForm: TMainForm;

IsSaved, IsCreated: Boolean;

Implementation

{$R \*.dfm}

Procedure CreateStack(); External 'LIBRARY5\_1.dll';

Procedure StackPush(Num: Integer); External 'LIBRARY5\_1.dll';

Procedure StackPop(); External 'LIBRARY5\_1.dll';

Procedure DestroyStack(); External 'LIBRARY5\_1.dll';

Procedure TMainForm.ButtonCreateClick(Sender: TObject);

Begin

If IsCreated Then

Begin

DestroyStack();

ClearListView(ListViewStack);

End;

CreateStack();

IsSaved := True;

EnableComponents(True);

ButtonPush.SetFocus;

ClientHeight := 360;

End;

Procedure TMainForm.ButtonDestroyClick(Sender: TObject);

Var

ButtonSelected: Integer;

Begin

ButtonSelected := Application.MessageBox('Вы уверены, что хотите уничтожить

стек?', 'Уничтожение стека', MB\_YESNO +

MB\_ICONQUESTION);

If ButtonSelected = MrYes Then

Begin

DestroyStack();

ClearListView(ListViewStack);

EnableComponents(False);

ButtonCreate.SetFocus;

ClientHeight := 150;

End

End;

Procedure TMainForm.ButtonPopClick(Sender: TObject);

Var

ButtonSelected: Integer;

Begin

ButtonSelected := Application.MessageBox('Вы уверены, что хотите удалить

элемент?', 'Удаление элемента',

MB\_YESNO + MB\_ICONQUESTION);

If ButtonSelected = MrYes Then

Begin

StackPop();

ListViewStack.Items[0].Delete;

End

End;

Procedure TMainForm.ButtonPushClick(Sender: TObject);

Begin

StackElementForm.ShowModal;

End;

Procedure TMainForm.AddItemToListView(ListView: TListView; Element: Integer);

Var

NewItem: TListItem;

Begin

NewItem := ListView.Items.Insert(0);

NewItem.Caption := IntToStr(ListView.Items.Count);

NewItem.SubItems.Insert(0, IntToStr(Element));

End;

Procedure TMainForm.ClearListView(ListView: TListView);

Var

I: Integer;

Begin

For I := ListView.Items.Count - 1 DownTo 0 Do

ListView.Items[I].Delete;

End;

Procedure TMainForm.ListViewOnChange(Num: Integer);

Begin

ButtonPush.Enabled := ListViewStack.Items.Count < MAXELEMENTS + Num;

ButtonPop.Enabled := ListViewStack.Items.Count > Num;

MMSaveFile.Enabled := ButtonPop.Enabled;

MMOpenFile.Enabled := ButtonPush.Enabled;

IsSaved := False;

End;

Procedure TMainForm.ListViewStackChange(Sender: TObject; Item: TListItem; Change:

TItemChange);

Begin

ListViewOnChange(0);

End;

Procedure TMainForm.ListViewStackDeletion(Sender: TObject; Item: TListItem);

Begin

ListViewOnChange(1);

End;

Procedure TMainForm.EnableComponents(IsEnabled: Boolean);

Begin

ListViewStack.Visible := IsEnabled;

ListViewStack.Enabled := IsEnabled;

ButtonPush.Enabled := IsEnabled;

ButtonDestroy.Enabled := IsEnabled;

MMOpenFile.Enabled := IsEnabled;

End;

Procedure TMainForm.FormClose(Sender: TObject; Var Action: TCloseAction);

Begin

If IsCreated Then

DestroyStack();

End;

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

Var

ButtonSelected: Integer;

Begin

If MMSaveFile.Enabled And Not IsSaved Then

Begin

ButtonSelected := Application.MessageBox('Желаете сохранить стек в файл?',

'Выход', MB\_YESNOCANCEL +

MB\_ICONQUESTION);

If ButtonSelected = MrYes Then

Begin

MMSaveFile.Click;

If Not IsSaved Then

Close;

End

Else If ButtonSelected = MrNo Then

CanClose := True

Else

CanClose := False;

End

Else

Begin

ButtonSelected := Application.MessageBox('Вы уверены, что хотите выйти?',

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

If ButtonSelected = MrYes Then

CanClose := True

Else

CanClose := False;

End;

End;

Procedure TMainForm.FormCreate(Sender: TObject);

Begin

ClientHeight := 150;

IsCreated := False;

End;

Function TMainForm.FormHelp(Command: Word; Data: NativeInt; Var CallHelp: Boolean):

Boolean;

Begin

InstructionForm.ShowModal;

CallHelp := False;

End;

Procedure TMainForm.FormResize(Sender: TObject);

Begin

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

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

End;

Procedure TMainForm.MMDevInfoClick(Sender: TObject);

Begin

DevInfoForm.ShowModal;

End;

Procedure TMainForm.MMExitClick(Sender: TObject);

Begin

Close;

End;

Procedure TMainForm.MMInstructionClick(Sender: TObject);

Begin

InstructionForm.ShowModal;

End;

Function IsFilePathCorrect(Path: String): Boolean;

Var

IsCorrect: Boolean;

Begin

IsCorrect := True;

If ExtractFileExt(Path) <> '.txt' Then

Begin

Application.MessageBox('Файл должен иметь разрешение .txt!', 'Ошибка',

MB\_OK + MB\_ICONERROR);

IsCorrect := False;

End;

IsFilePathCorrect := IsCorrect;

End;

Procedure TMainForm.MMOpenFileClick(Sender: TObject);

Var

I, NewElement: Integer;

IsCorrect: Boolean;

FIn: TextFile;

Begin

IsCorrect := OpenDialog.Execute And IsFilePathCorrect(OpenDialog.FileName);

If IsCorrect Then

Try

Try

IsCorrect := True;

AssignFile(FIn, OpenDialog.FileName);

Reset(FIn);

Read(FIn, NewElement);

If (NewElement < MINNUM) Or (NewElement > MAXNUM) Then

Begin

IsCorrect := False;

Application.MessageBox('Число в файле не соответствует

диапазону!', 'Ошибка', MB\_ICONERROR);

End

Else If Not Eof(FIn) Then

Begin

IsCorrect := False;

Application.MessageBox('В файле некорректное количество строк!',

'Ошибка', MB\_ICONERROR);

End;

Finally

CloseFile(FIn);

End;

Except

IsCorrect := False;

Application.MessageBox('В файле некорректные данные!', 'Ошибка',

MB\_ICONERROR);

End;

If IsCorrect Then

Begin

StackPush(NewElement);

AddItemToListView(ListViewStack, NewElement);

IsSaved := False;

End;

End;

Procedure TMainForm.MMSaveFileClick(Sender: TObject);

Var

I: Integer;

IsCorrect: Boolean;

FOut: TextFile;

Begin

IsCorrect := SaveDialog.Execute And IsFilePathCorrect(SaveDialog.FileName);

If IsCorrect Then

Try

Try

AssignFile(FOut, SaveDialog.FileName);

Rewrite(FOut);

Writeln(FOut, 'Стек:');

With ListViewStack Do

For I := 0 To Items.Count - 1 Do

Writeln(FOut, Items[I].SubItems.Text, '--------');

Finally

CloseFile(FOut);

End;

Except

IsCorrect := False;

Application.MessageBox('Произошла ошибка при записи в файл!', 'Ошибка',

MB\_ICONERROR);

End;

IsSaved := IsCorrect;

End;

End.

**Unit StackElementUnit;**

Interface

Uses

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

System.Classes, Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Menus,

Vcl.StdCtrls, InstructionUnit, DevInfoUnit;

Type

TStackElementForm = Class(TForm)

MainMenu: TMainMenu;

MMInstruction: TMenuItem;

MMDevInfo: TMenuItem;

EditElement: TEdit;

LabelElement: TLabel;

ButtonPush: TButton;

ButtonCancel: TButton;

PopupMenu: TPopupMenu;

Procedure ButtonCancelClick(Sender: TObject);

Procedure ButtonPushClick(Sender: TObject);

Procedure MMInstructionClick(Sender: TObject);

Procedure MMDevInfoClick(Sender: TObject);

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

Procedure EditElementChange(Sender: TObject);

Procedure FormClose(Sender: TObject; Var Action: TCloseAction);

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

Procedure FormKeyDown(Sender: TObject; Var Key: Word; Shift: TShiftState);

Procedure FormShow(Sender: TObject);

End;

Var

StackElementForm: TStackElementForm;

Implementation

{$R \*.dfm}

Uses MainUnit;

Const

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

MINUS = '-';

NULL = #0;

BACKSPACE = #8;

Var

IsEditEmpty: Boolean;

Procedure StackPush(Num: Integer); External 'LIBRARY5\_1.dll';

Procedure TStackElementForm.ButtonCancelClick(Sender: TObject);

Begin

Close;

End;

Procedure TStackElementForm.ButtonPushClick(Sender: TObject);

Begin

StackPush(StrToInt(EditElement.Text));

MainForm.AddItemToListView(MainForm.ListViewStack, StrToInt(EditElement.Text));

IsEditEmpty := True;

Close;

End;

Procedure TStackElementForm.EditElementChange(Sender: TObject);

Begin

ButtonPush.Enabled := (EditElement.Text <> '') And (EditElement.Text <> MINUS);

IsEditEmpty := Not ButtonPush.Enabled;

End;

Procedure TStackElementForm.EditElementKeyPress(Sender: TObject; Var Key: Char);

Var

SelStart, Len: Integer;

EditText: String;

Begin

SelStart := EditElement.SelStart;

EditText := EditElement.Text;

Len := Length(EditText);

If Not CharInSet(Key, NUMBERS) And Not CharInSet(Key, [BACKSPACE, MINUS]) Then

Key := NULL

Else If (Selstart > 0) And (Key = MINUS) Then

Key := NULL

Else If (SelStart = 0) And (Len > 0) And (EditText[1] = MINUS) And

(Key = MINUS) Then

Key := NULL

Else If (Len > 5) And ((Abs(StrToInt(EditText)) > MAXNUM Div 10) Or

(Abs(StrToInt(EditText)) = MAXNUM)) And Not CharInSet(Key, [BACKSPACE,

MINUS]) Then

Key := NULL

Else If (Key = '0') And (Len > 0) And ((SelStart = 0) Or (SelStart = 1) And

(EditText[1] = MINUS)) Then

Key := NULL

Else If (EditText = '0') Then

If (SelStart = Len) And (Key <> BACKSPACE) Then

Key := NULL

Else If (SelStart = 0) And (Key = MINUS) Then

Key := NULL;

End;

Procedure TStackElementForm.FormClose(Sender: TObject; Var Action: TCloseAction);

Begin

EditElement.Text := '';

End;

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

Var

ButtonSelected: Integer;

Begin

If Not IsEditEmpty Then

Begin

ButtonSelected := Application.MessageBox('Вы уверены, что хотите отменить

ввод элемента?', 'Отмена',

MB\_YESNO + MB\_ICONQUESTION);

If ButtonSelected = MrYes Then

CanClose := True

Else

CanClose := False;

End

Else

CanClose := True;

End;

Procedure TStackElementForm.FormKeyDown(Sender: TObject; Var Key: Word; Shift:

TShiftState);

Begin

If Key = VK\_ESCAPE Then

Close

Else If (Key = 13) And ButtonPush.Enabled Then

ButtonPush.Click;

End;

Procedure TStackElementForm.FormShow(Sender: TObject);

Begin

IsEditEmpty := True;

End;

Procedure TStackElementForm.MMDevInfoClick(Sender: TObject);

Begin

DevInfoForm.ShowModal;

End;

Procedure TStackElementForm.MMInstructionClick(Sender: TObject);

Begin

InstructionForm.ShowModal;

End;

End.

**КОД ПРОГРАММЫ С++:**

#include <iostream>

#include <fstream>

#include <locale>

#include <Windows.h>

#include <string>

using namespace std;

const int MINNUM = -999999;

const int MAXNUM = 999999;

const int MAXELEMENTS = 30;

struct stack {

int num;

stack\* next;

};

stack\* createStack(bool& isCreated) {

stack\* head;

head = new stack;

head->next = nullptr;

cout << "Стек успешно создан!\n";

isCreated = true;

return head;

}

void stackPush(stack\*& head, int num) {

stack\* newElement;

newElement = new stack;

newElement->num = num;

newElement->next = head;

head = newElement;

cout << "Элемент успешно добавлен в стек!\n";

}

int stackPop(stack\*& head, bool isShowMessage, int& elementsAmount) {

stack\* deletedElement;

int num;

deletedElement = head;

head = deletedElement->next;

num = deletedElement->num;

delete deletedElement;

if (isShowMessage)

cout << "Элемент успешно удалён из стека!\n";

elementsAmount--;

return num;

}

void destroyStack(stack\*& head, bool& isCreated, int& elementsAmount, bool isShowMessage) {

while (head != nullptr)

stackPop(head, false, elementsAmount);

delete head;

isCreated = false;

elementsAmount = 0;

if (isShowMessage)

cout << "Стек успешно уничтожен!\n";

}

bool isStackEmpty(stack\* head) {

if (head->next == nullptr) {

cout << "На данный момент стек пуст.\n";

return true;

}

else

return false;

}

bool isStackFull(int elementsAmount) {

if (elementsAmount == MAXELEMENTS) {

cout << "Достигнуто максимальное количество элементов!\n";

return true;

}

else

return false;

}

int inputNum(string outputMessage, const int MIN, const int MAX) {

int num;

bool isIncorrect;

do {

isIncorrect = false;

cout << outputMessage;

cin >> num;

if (cin.get() != '\n') {

cin.clear();

while (cin.get() != '\n');

isIncorrect = true;

cout << "Введенное значение должно быть целым числом! Повторите

попытку.\n";

}

if (!isIncorrect && ((num < MIN) || (num > MAX))) {

cout << "Диапазон числа не соответствует условию! Повторите

попытку.\n";

isIncorrect = true;

}

} while (isIncorrect);

return num;

}

void addElementToStack(stack\*& head, int& elementsAmount) {

if (!isStackFull(elementsAmount)) {

int num;

num = inputNum("Введите значение для добавления в стек (-999999..

999999): ", MINNUM, MAXNUM);

stackPush(head, num);

elementsAmount++;

}

}

void outputStack(stack\* head, int elementsAmount) {

if (!isStackEmpty(head)) {

stack\* temp;

temp = head;

cout << "Элементы стека:\n";

for (int i = elementsAmount; i > 0; i--) {

cout << i << ". " << temp->num << "\n";

temp = temp->next;

}

}

}

bool isFileExtIncorrect(string path, bool isIncorrect) {

if (!isIncorrect && (path.substr(path.size() - 4) != ".txt")) {

cout << "Введенный Вами файл не в формате \*.txt! Повторите попытку.\n";

isIncorrect = true;

}

return isIncorrect;

}

string getFileInputPath() {

string path;

bool isIncorrect;

do {

isIncorrect = false;

cout << "Введите путь к файлу:\n";

getline(cin, path, '\n');

ifstream fin(path);

fin.open(path);

if (!fin.is\_open()) {

cout << "Файл недоступен для чтения! Повторите попытку.\n";

isIncorrect = true;

}

isIncorrect = isFileExtIncorrect(path, isIncorrect);

fin.close();

} while (isIncorrect);

return path;

}

void getNumFromFile(stack\*& head, int& elementsAmount) {

if (!isStackFull(elementsAmount)) {

int num;

bool isIncorrect;

string path;

path = getFileInputPath();

do {

isIncorrect = false;

ifstream fin(path);

try {

fin >> num;

isIncorrect = (num > MAXNUM) || (num < MINNUM) ||

!fin.eof();

}

catch (string errorMessage)

isIncorrect = true;

fin.close();

if (isIncorrect) {

cout << "Данные выбранного файла некорректны! Повторите

попытку.\n";

path = getFileInputPath();

}

} while (isIncorrect);

cout << "Данные из файла успешно считаны.\n";

stackPush(head, num);

elementsAmount++;

}

}

string getFileOutputPath() {

string path;

bool isIncorrect;

do {

isIncorrect = false;

cout << "Введите путь к файлу, в который нужно записать результат:\n";

getline(cin, path, '\n');

isIncorrect = isFileExtIncorrect(path, isIncorrect);

if (!isIncorrect) {

ofstream fout(path);

fout.open(path);

if (!fout.is\_open()) {

cout << "Файл недоступен для записи! Повторите

попытку.\n";

isIncorrect = true;

}

fout.close();

}

} while (isIncorrect);

return path;

}

void writeStackIntoFile(int elementsAmount, stack\* head) {

if (!isStackEmpty(head)) {

bool isIncorrect;

string path;

stack\* temp;

path = getFileOutputPath();

do {

isIncorrect = false;

ofstream fout(path);

try {

temp = head;

fout << "Стек:\n\n";

for (int i = elementsAmount; i > 0; i--) {

fout << temp->num << "\n--------\n";

temp = temp->next;

}

}

catch (string errorMessage) {

cout << "Произошла ошибка. Повторите попытку.\n";

isIncorrect = true;

}

fout.close();

if (isIncorrect)

path = getFileOutputPath();

} while (isIncorrect);

cout << "Результат записан.\n";

}

}

void outputInstruction() {

cout << "Инструкция\n\n" << "ИЗМЕНЕНИЕ СТЕКА:\n1. Можно удалить только

последний добавленный в стек элемент.\n2. Элементы стека - целые

числа в диапазоне -999999..999999.\n3. Максимальное количество

элементов в стеке - 30.\n\n" << "ФАЙЛЫ:\n1. Открываемый / сохраняемый

файл должен быть формата \* .txt.\n2. В открываемом файле должно быть

записано только 1 число, которое добавится в стек.\n3. При сохранении

данные указанного файла перезаписываются.\n\n" << "Разработчик:

Городко Ксения\nЛабораторная №5.1\n";

}

void outputProgramInfo() {

cout << "Добро пожаловать в программу для работы со стеком!\n";

}

void outputMenu() {

cout << "Меню:\n";

cout << "1) Создать пустой стек;\n" << "2) Добавить элемент;\n" << "3) Удалить

элемент;\n" << "4) Вывести элементы стека;\n" << "5) Уничтожить

стек;\n" << "6) Открыть файл;\n" << "7) Сохранить файл;\n" << "8)

Вывести инструкцию;\n" << "9) Выход из программы.\n";

cout << "Для выбора пункта меню введите его номер.\n";

}

void returnToMenu() {

cout << "\nНажмите Enter для возвращения в меню...\n";

getchar();

}

void performMenuOption() {

const int CREATE = 1;

const int PUSH = 2;

const int POP = 3;

const int SHOW = 4;

const int DESTROY = 5;

const int OPENFILE = 6;

const int SAVEFILE = 7;

const int SHOWINSTRUCTION = 8;

const int EXIT = 9;

int choice;

int elementsAmount;

bool isCreated;

stack\* head;

choice = 0;

elementsAmount = 0;

isCreated = false;

head = nullptr;

while (choice != EXIT) {

outputMenu();

choice = inputNum("", CREATE, EXIT);

cout << "\n";

if ((choice == CREATE) || (choice > SAVEFILE) || isCreated) {

switch (choice) {

case CREATE: {

if (isCreated)

destroyStack(head, isCreated, elementsAmount,

false);

head = createStack(isCreated);

break;

}

case PUSH: {

addElementToStack(head, elementsAmount);

break;

}

case POP: {

if (!isStackEmpty(head))

stackPop(head, true, elementsAmount);

break;

}

case SHOW: {

outputStack(head, elementsAmount);

break;

}

case DESTROY: {

destroyStack(head, isCreated, elementsAmount, true);

break;

}

case OPENFILE: {

getNumFromFile(head, elementsAmount);

break;

}

case SAVEFILE: {

writeStackIntoFile(elementsAmount, head);

break;

}

case SHOWINSTRUCTION: {

outputInstruction();

break;

}

}

}

else

cout << "На данный момент стека не существует.\n";

if (choice != EXIT)

returnToMenu();

}

if (isCreated)

destroyStack(head, isCreated, elementsAmount, false);

cout << "Ждем Вашего возвращения...";

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

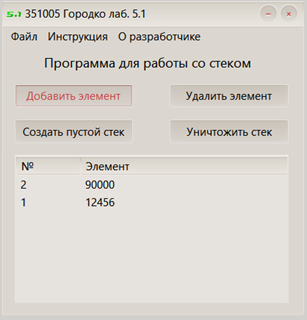
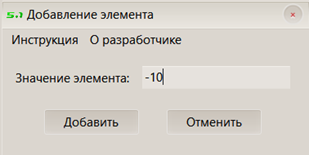
outputProgramInfo();

performMenuOption();

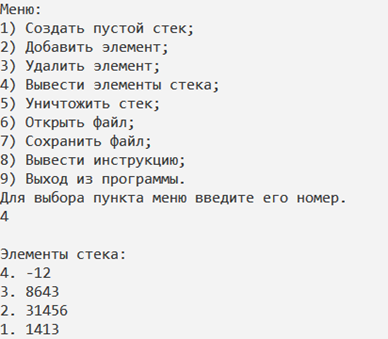
return 0;  
}

**СКРИНШОТЫ:**

**Delphi:**

**C++****:**

****

**БЛОК-СХЕМА:**

