Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа № 7

По дисциплине «Основы алгоритмизации и программирования»

На тему «Полустатические структуры данных: стеки»

Выполнил:

Студент 1 курса 8 группы

Статько Герман Вячеславович

Преподаватель: асс. Андронова М.В.

Минск, 2024

**Создать проект, демонстрирующий работу со стеком, организованным на основе списка, в соответствии со своим вариантом из таблицы ниже. Все операции со стеком реализовать через функции. Дополнить проект функциями очистки стека clear(), сохранения в файл и считывания из файла. Проект должен содержать три части: главная функция, файл с функциями работы со стеком и заголовочный файл. Создать интерфейс в виде меню.**

**Вариант 12**

****

**Main.cpp**

#include <iostream>

#include "myFunc.h"

#include <fstream> // Для работы с файлами

using namespace std;

// Структура для представления элемента стека

int main()

{

setlocale(LC\_CTYPE, "rus");

int choice;

stack\* p = nullptr; // Инициализация указателя на стек

// Основной цикл программы

do

{

// Вывод меню на экран

cout << "1 - Добавление элемента в стек" << endl

<< "2 - Формирование двух новых стеков их четных и нечетных элементов" << endl

<< "3 - Очистить стек" << endl

<< "4 - Сохранить стек в файл" << endl

<< "5 - Считать стек из файла" << endl

<< "6 - Выход" << endl;

// Чтение выбора пользователя

cin >> choice;

// Обработка выбора пользователя

switch (choice)

{

case 1:

{

int number;

cout << "Введите число: ";

cin >> number;

push(p, number); // Добавление элемента в стек

break;

}

case 2:

{

stack\* evenStack = nullptr; // Инициализация стека четных элементов

stack\* oddStack = nullptr; // Инициализация стека нечетных элементов

splitStack(p, evenStack, oddStack); // Формирование двух новых стеков

break;

}

case 3:

clear(p); // Очистка стека

cout << "Стек очищен." << endl;

break;

case 4:

saveToFile(p); // Сохранение стека в файл

cout << "Стек сохранен в файл." << endl;

break;

case 5:

clear(p); // Очистка стека перед загрузкой из файла

readFromFile(p); // Считывание стека из файла

cout << "Стек загружен из файла." << endl;

break;

case 6:

cout << "Выход из программы." << endl;

break;

default:

cout << "Неверный выбор." << endl;

}

} while (choice != 6); // Повторение цикла до выбора пользователем выхода

}

**MyFunc.h**

#pragma once

struct stack

{

int data; // Значение элемента

stack\* next; // Указатель на следующий элемент стека

};

// Функция извлечения элемента из стека

int pop(stack\*& st);

// Функция добавления элемента в стек

void push(stack\*& st, int d);

// Функция разделения стека на два: четных и нечетных элементов

void splitStack(stack\* sourceStack, stack\*& evenStack, stack\*& oddStack);

// Функция очистки стека

void clear(stack\*& st);

// Функция сохранения содержимого стека в файл

void saveToFile(stack\* st);

// Функция считывания содержимого стека из файла

void readFromFile(stack\*& st);

**MyFunc.cpp**

#include "myFunc.h"

#include <iostream>

#include <fstream>

using namespace std;

// Функция извлечения элемента из стека

int pop(stack\*& st)

{

// Проверка наличия элементов в стеке

if (st == nullptr)

{

cout << "Стек пуст" << endl;

return -1;

}

int data = st->data; // Сохранение значения извлекаемого элемента

stack\* temp = st; // Сохранение адреса извлекаемого элемента

st = st->next; // Перемещение указателя на следующий элемент

delete temp; // Освобождение памяти, занимаемой извлеченным элементом

return data; // Возвращение значения извлеченного элемента

}

// Функция добавления элемента в стек

void push(stack\*& st, int d)

{

stack\* pv = new stack; // Создание нового элемента стека

pv->data = d; // Присваивание значению элемента

pv->next = st; // Присваивание указателю на следующий элемент ссылку на текущий элемент

st = pv; // Перемещение указателя на вершину стека

}

// Функция разделения стека на два: четных и нечетных элементов

void splitStack(stack\* sourceStack, stack\*& evenStack, stack\*& oddStack)

{

while (sourceStack != nullptr)

{

int element = pop(sourceStack); // Извлечение элемента из исходного стека

// Проверка четности извлеченного элемента и добавление его в соответствующий стек

if (element % 2 == 0)

{

push(evenStack, element); // Добавление в стек четных элементов

}

else

{

push(oddStack, element); // Добавление в стек нечетных элементов

}

}

cout << "Четные элементы стека:" << endl;

while (evenStack != nullptr)

{

cout << pop(evenStack) << endl; // Вывод четных элементов

}

cout << "Нечетные элементы стека:" << endl;

while (oddStack != nullptr)

{

cout << pop(oddStack) << endl; // Вывод нечетных элементов

}

}

// Функция очистки стека

void clear(stack\*& st)

{

while (st != nullptr)

{

pop(st); // Извлечение элементов из стека

}

}

// Функция сохранения содержимого стека в файл

void saveToFile(stack\* st)

{

ofstream file("stack.txt"); // Открытие файла для записи

while (st != nullptr)

{

file << pop(st) << endl; // Запись элементов стека в файл

}

file.close(); // Закрытие файла

}

// Функция считывания содержимого стека из файла

void readFromFile(stack\*& st)

{

ifstream file("stack.txt"); // Открытие файла для чтения

int number;

while (file >> number)

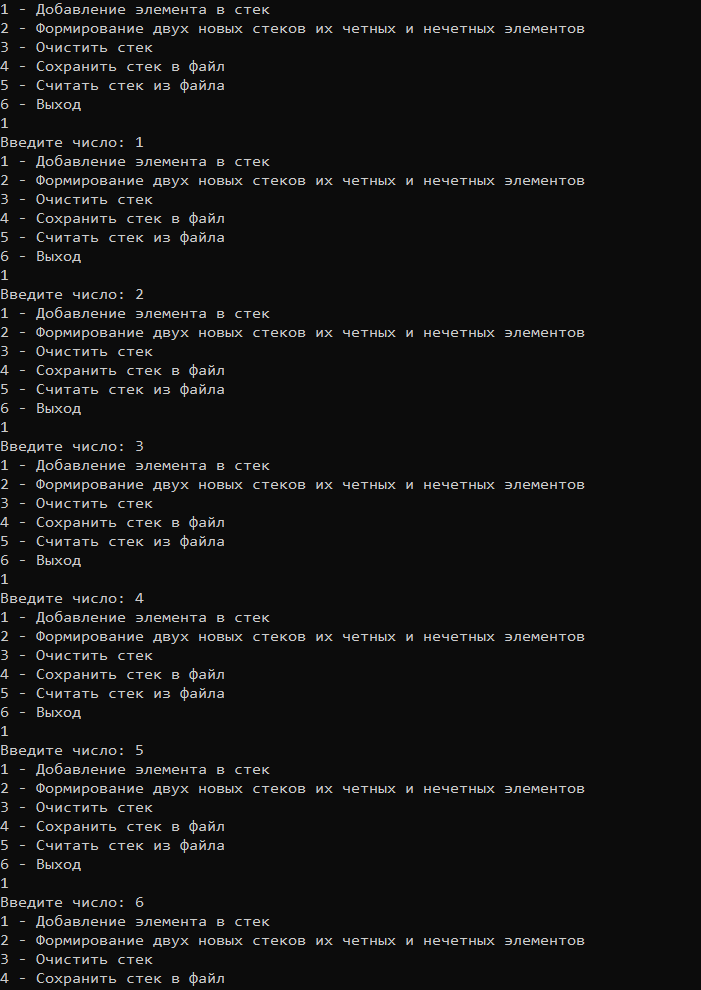
{

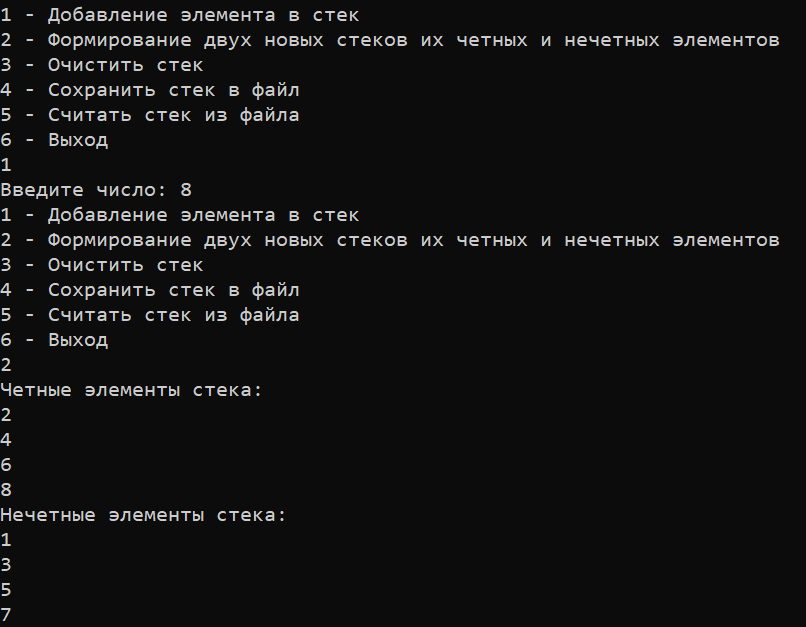
push(st, number); // Добавление элементов из файла в стек

}

file.close(); // Закрытие файла

}

****

****

**Доп. Задания**

**Вариант 1**

****

**Main.cpp**

#include <iostream>

#include <fstream> // Для работы с файлами

#include "myFunc.h"

using namespace std;

int main()

{

setlocale(LC\_CTYPE, "rus");

int choice;

stack\* p = nullptr; // Инициализация указателя на стек

// Основной цикл программы

do

{

// Вывод меню на экран

cout << "1 - Добавление элемента в стек" << endl

<< "2 - Разделение стека на два: одного из положительных и другого из отрицательных элементов" << endl

<< "3 - Очистить стек" << endl

<< "4 - Сохранить стек в файл" << endl

<< "5 - Считать стек из файла" << endl

<< "6 - Выход" << endl;

// Чтение выбора пользователя

cin >> choice;

// Обработка выбора пользователя

switch (choice)

{

case 1:

{

int number;

cout << "Введите число: ";

cin >> number;

push(p, number); // Добавление элемента в стек

break;

}

case 2:

{

stack\* positiveStack = nullptr; // Инициализация стека положительных элементов

stack\* negativeStack = nullptr; // Инициализация стека отрицательных элементов

splitStack(p, positiveStack, negativeStack); // Разделение стека на два новых

break;

}

case 3:

clear(p); // Очистка стека

cout << "Стек очищен." << endl;

break;

case 4:

saveToFile(p); // Сохранение стека в файл

cout << "Стек сохранен в файл." << endl;

break;

case 5:

clear(p); // Очистка стека перед загрузкой из файла

readFromFile(p); // Считывание стека из файла

cout << "Стек загружен из файла." << endl;

break;

case 6:

cout << "Выход из программы." << endl;

break;

default:

cout << "Неверный выбор." << endl;

}

} while (choice != 6); // Повторение цикла до выбора пользователем выхода

}

**myFunc.h**

#pragma once

// Структура для представления элемента стека

struct stack

{

int data; // Значение элемента

stack\* next; // Указатель на следующий элемент стека

};

// Функция извлечения элемента из стека

int pop(stack\*& st);

// Функция добавления элемента в стек

void push(stack\*& st, int d);

// Функция разделения стека на два: одного из положительных и другого из отрицательных элементов

void splitStack(stack\* sourceStack, stack\*& positiveStack, stack\*& negativeStack);

// Функция очистки стека

void clear(stack\*& st);

// Функция сохранения содержимого стека в файл

void saveToFile(stack\* st);

// Функция считывания содержимого стека из файла

void readFromFile(stack\*& st);

**myFunc.cpp**

#include "myFunc.h"

#include <iostream>

#include <fstream>

using namespace std;

// Функция извлечения элемента из стека

int pop(stack\*& st)

{

// Проверка наличия элементов в стеке

if (st == nullptr)

{

cout << "Стек пуст" << endl;

return -1;

}

int data = st->data; // Сохранение значения извлекаемого элемента

stack\* temp = st; // Сохранение адреса извлекаемого элемента

st = st->next; // Перемещение указателя на следующий элемент

delete temp; // Освобождение памяти, занимаемой извлеченным элементом

return data; // Возвращение значения извлеченного элемента

}

// Функция добавления элемента в стек

void push(stack\*& st, int d)

{

stack\* pv = new stack; // Создание нового элемента стека

pv->data = d; // Присваивание значению элемента

pv->next = st; // Присваивание указателю на следующий элемент ссылку на текущий элемент

st = pv; // Перемещение указателя на вершину стека

}

// Функция разделения стека на два: одного из положительных и другого из отрицательных элементов

void splitStack(stack\* sourceStack, stack\*& positiveStack, stack\*& negativeStack)

{

while (sourceStack != nullptr)

{

int element = pop(sourceStack); // Извлечение элемента из исходного стека

// Проверка знака извлеченного элемента и добавление его в соответствующий стек

if (element >= 0)

{

push(positiveStack, element); // Добавление в стек положительных элементов

}

else

{

push(negativeStack, element); // Добавление в стек отрицательных элементов

}

}

cout << "Стек положительных элементов:" << endl;

while (positiveStack != nullptr)

{

cout << pop(positiveStack) << endl; // Вывод положительных элементов

}

cout << "Стек отрицательных элементов:" << endl;

while (negativeStack != nullptr)

{

cout << pop(negativeStack) << endl; // Вывод отрицательных элементов

}

}

// Функция очистки стека

void clear(stack\*& st)

{

while (st != nullptr)

{

pop(st); // Извлечение элементов из стека

}

}

// Функция сохранения содержимого стека в файл

void saveToFile(stack\* st)

{

ofstream file("stack.txt"); // Открытие файла для записи

while (st != nullptr)

{

file << pop(st) << endl; // Запись элементов стека в файл

}

file.close(); // Закрытие файла

}

// Функция считывания содержимого стека из файла

void readFromFile(stack\*& st)

{

ifstream file("stack.txt"); // Открытие файла для чтения

int number;

while (file >> number)

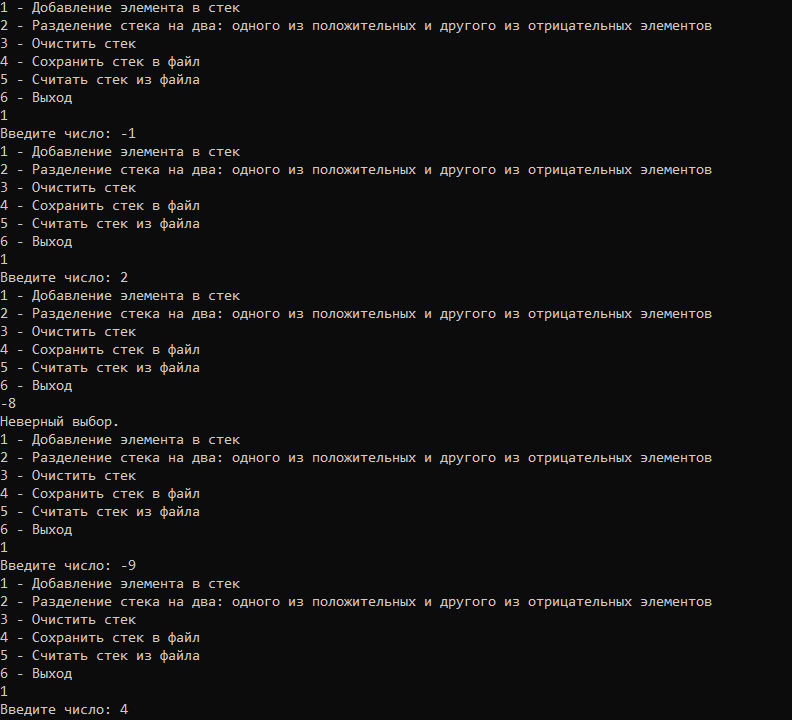
{

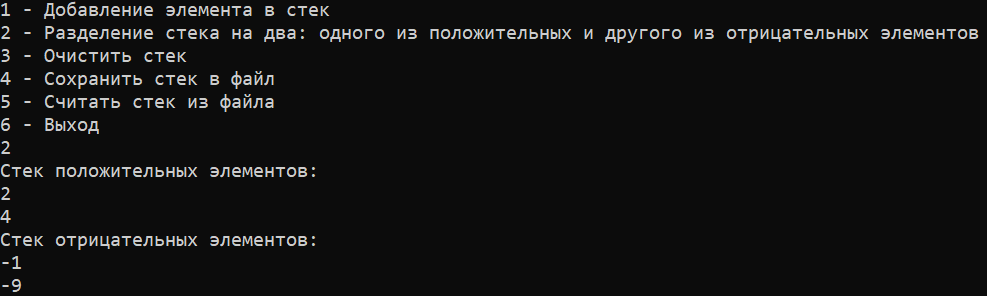
push(st, number); // Добавление элементов из файла в стек

}

file.close(); // Закрытие файла

}

****

****

**Вариант 8**

****

**Main.cpp**

#include <iostream>

#include <fstream> // Для работы с файлами

#include "myFunc.h"

using namespace std;

int main()

{

setlocale(LC\_CTYPE, "rus");

int choice;

stack\* p = nullptr; // Инициализация указателя на стек

// Основной цикл программы

do

{

// Вывод меню на экран

cout << "1 - Добавление элемента в стек" << endl

<< "2 - Разделение стека на два: одного из элементов, значение которых превышает 50, и другого - из остальных элементов" << endl

<< "3 - Очистить стек" << endl

<< "4 - Сохранить стек в файл" << endl

<< "5 - Считать стек из файла" << endl

<< "6 - Выход" << endl;

// Чтение выбора пользователя

cin >> choice;

// Обработка выбора пользователя

switch (choice)

{

case 1:

{

int number;

cout << "Введите число: ";

cin >> number;

push(p, number); // Добавление элемента в стек

break;

}

case 2:

{

stack\* stack1 = nullptr; // Инициализация стека элементов, значение которых превышает 50

stack\* stack2 = nullptr; // Инициализация стека остальных элементов

splitStack(p, stack1, stack2); // Разделение стека на два новых

break;

}

case 3:

clear(p); // Очистка стека

cout << "Стек очищен." << endl;

break;

case 4:

saveToFile(p); // Сохранение стека в файл

cout << "Стек сохранен в файл." << endl;

break;

case 5:

clear(p); // Очистка стека перед загрузкой из файла

readFromFile(p); // Считывание стека из файла

cout << "Стек загружен из файла." << endl;

break;

case 6:

cout << "Выход из программы." << endl;

break;

default:

cout << "Неверный выбор." << endl;

}

} while (choice != 6); // Повторение цикла до выбора пользователем выхода

}

**myFunc.h**

#pragma once

// Структура для представления элемента стека

struct stack

{

int data; // Значение элемента

stack\* next; // Указатель на следующий элемент стека

};

// Функция извлечения элемента из стека

int pop(stack\*& st);

// Функция добавления элемента в стек

void push(stack\*& st, int d);

// Функция разделения стека на два: одного из элементов, значение которых превышает 50, и другого - из остальных элементов

void splitStack(stack\* sourceStack, stack\*& stack1, stack\*& stack2);

// Функция очистки стека

void clear(stack\*& st);

// Функция сохранения содержимого стека в файл

void saveToFile(stack\* st);

// Функция считывания содержимого стека из файла

void readFromFile(stack\*& st);

**myFunc.cpp**

#include "myFunc.h"

#include <iostream>

#include <fstream>

using namespace std;

// Функция извлечения элемента из стека

int pop(stack\*& st)

{

// Проверка наличия элементов в стеке

if (st == nullptr)

{

cout << "Стек пуст" << endl;

return -1;

}

int data = st->data; // Сохранение значения извлекаемого элемента

stack\* temp = st; // Сохранение адреса извлекаемого элемента

st = st->next; // Перемещение указателя на следующий элемент

delete temp; // Освобождение памяти, занимаемой извлеченным элементом

return data; // Возвращение значения извлеченного элемента

}

// Функция добавления элемента в стек

void push(stack\*& st, int d)

{

stack\* pv = new stack; // Создание нового элемента стека

pv->data = d; // Присваивание значению элемента

pv->next = st; // Присваивание указателю на следующий элемент ссылку на текущий элемент

st = pv; // Перемещение указателя на вершину стека

}

// Функция разделения стека на два: одного из элементов, значение которых превышает 50, и другого - из остальных элементов

void splitStack(stack\* sourceStack, stack\*& stack1, stack\*& stack2)

{

while (sourceStack != nullptr)

{

int element = pop(sourceStack); // Извлечение элемента из исходного стека

// Проверка значения извлеченного элемента и добавление его в соответствующий стек

if (element > 50)

{

push(stack1, element); // Добавление в стек элементов, значение которых превышает 50

}

else

{

push(stack2, element); // Добавление в стек остальных элементов

}

}

cout << "Стек элементов, значение которых превышает 50:" << endl;

while (stack1 != nullptr)

{

cout << pop(stack1) << endl; // Вывод элементов, значение которых превышает 50

}

cout << "Стек остальных элементов:" << endl;

while (stack2 != nullptr)

{

cout << pop(stack2) << endl; // Вывод остальных элементов

}

}

// Функция очистки стека

void clear(stack\*& st)

{

while (st != nullptr)

{

pop(st); // Извлечение элементов из стека

}

}

// Функция сохранения содержимого стека в файл

void saveToFile(stack\* st)

{

ofstream file("stack.txt"); // Открытие файла для записи

while (st != nullptr)

{

file << pop(st) << endl; // Запись элементов стека в файл

}

file.close(); // Закрытие файла

}

// Функция считывания содержимого стека из файла

void readFromFile(stack\*& st)

{

ifstream file("stack.txt"); // Открытие файла для чтения

int number;

while (file >> number)

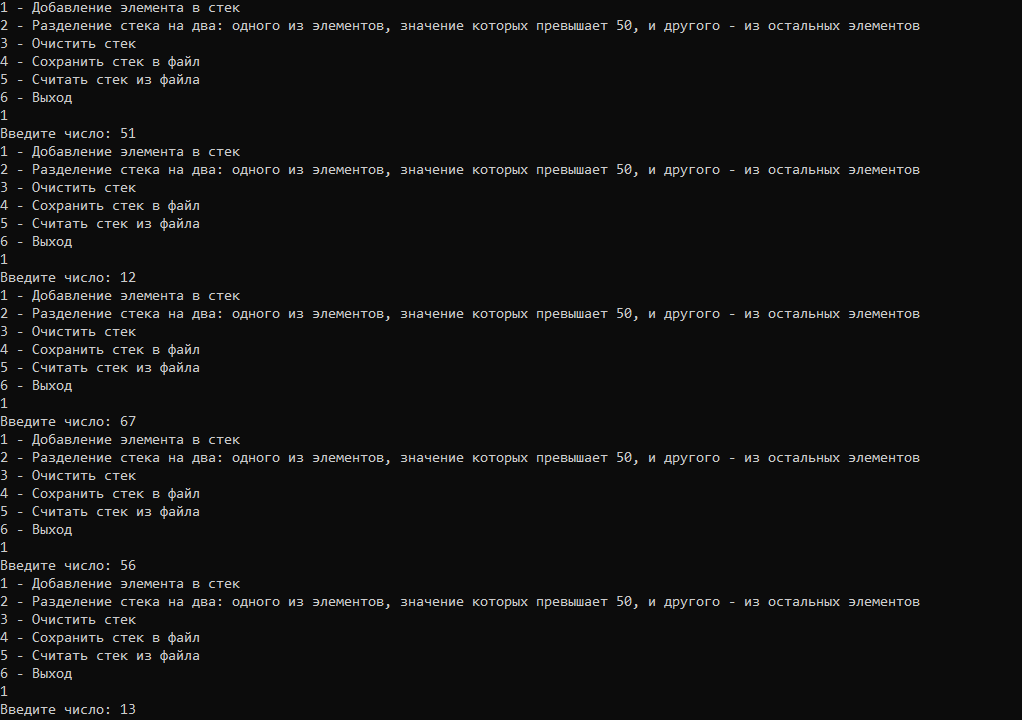
{

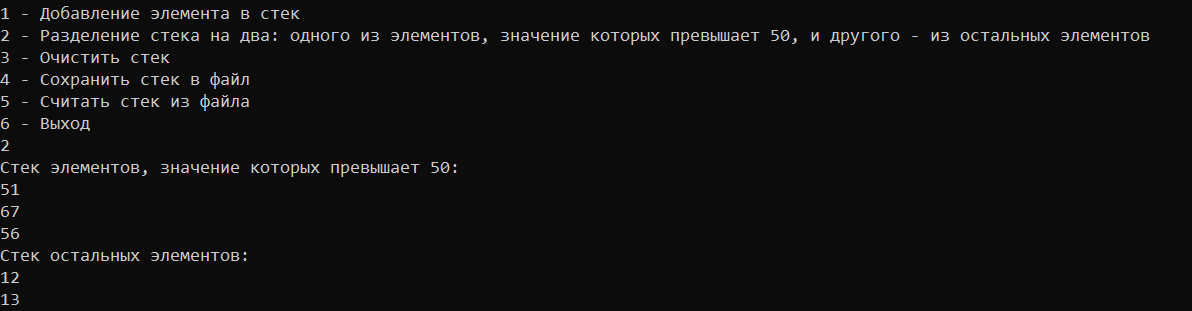
push(st, number); // Добавление элементов из файла в стек

}

file.close(); // Закрытие файла

}

****

****

**Вариант 6**

****

**Main.cpp**

#include <iostream>

#include <fstream>

#include "myFunc.h"

using namespace std;

void push(char x, Stack\*& myStk) //Добавление элемента х в стек

{

Stack\* e = new Stack; //выделение памяти для нового элемента

e->data = x; //запись элемента x в поле data

e->next = myStk; //перенос вершины на следующий элемент

e->before = nullptr; //перенос вершины на предыдущий элемент

myStk = e; //сдвиг вершины на позицию вперед

}

char pop(Stack\*& myStk) //Извлечение (удаление) элемента из стека

{

if (myStk == NULL)

{

cout << "Стек пуст!" << endl;

return -1; //если стек пуст - возврат (-1)

}

else

{

Stack\* e = myStk; //е-переменная для хранения адреса элемента

char x = myStk->data; //запись элемента из поля data в перем. x

if (myStk)

myStk = myStk->next; //перенос вершины

delete e;

return x;

}

}

void toFile(Stack\*& myStk) //Запись в файл

{

Stack\* e = myStk;

Stack buf;

ofstream frm("mStack.dat");

if (frm.fail())

{

cout << "\n Ошибка открытия файла";

exit(1);

}

while (e)

{

buf = \*e;

frm.write((char\*)&buf, sizeof(Stack));

e = e->next;

}

frm.close();

cout << "Стек записан в файл mStack.dat\n";

}

void fromFile(Stack\*& myStk) //Считывание из файла

{

Stack buf, \* p = nullptr, \* e = nullptr;

ifstream frm("mStack.dat");

if (frm.fail())

{

cout << "\n Ошибка открытия файла";

exit(1);

}

frm.seekg(0);

frm.read((char\*)&buf, sizeof(Stack));

while (!frm.eof())

{

push(buf.data, e);

frm.read((char\*)&buf, sizeof(Stack));

}

frm.close();

while (e != NULL)

{

buf.data = pop(e);

push(buf.data, p);

myStk = p;

}

cout << "\nСтек считан из файла mStack.dat\n\n";

}

void show(Stack\*& myStk) //Вывод стека с очисткой

{

Stack\* e = myStk;

if (e == NULL)

cout << "Стек пуст!" << endl;

while (e != NULL)

{

cout << e->data << " ";

e = e->next;

}

cout << endl;

}

void clear(Stack\*& myStk)

{

while (myStk != nullptr)

{

Stack\* temp = myStk;

myStk = myStk->next;

delete temp;

}

cout << "\nСтек пуст.\n";

}

void counterEqualLeftAndRight(Stack\*& myStk)

{

int counter = 0; // Счетчик для подсчета соответствующих пар

// Проверяем, что текущий элемент и его следующие два элемента не являются нулевыми

while (myStk != nullptr && myStk->next != nullptr && myStk->next->next != nullptr)

{

if (myStk->data == myStk->next->next->data)

{

counter++; // Увеличиваем счетчик, если значения текущего элемента и элемента через один равны

}

myStk = myStk->next; // Переходим к следующему элементу стека

}

cout << counter << endl; // Выводим значение счетчика

}

**myFunc.h**

#pragma once

struct Stack

{

char data; //информационный элемент

Stack\* next; //указатель на следующий элемент

Stack\* before; //указатель на предыдущий элемент

};

void show(Stack\*& myStk);

char pop(Stack\*& myStk);

void push(char x, Stack\*& myStk);

void toFile(Stack\*& myStk);

void fromFile(Stack\*& myStk);

void clear(Stack\*& myStk);

void counterEqualLeftAndRight(Stack\*& myStk);

**myFunc.cpp**

#include <iostream>

#include "myFunc.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "Rus");

int choice; char x;

Stack\* myStk = new Stack; //выделение памяти для стека

myStk = NULL; //инициализация первого элемента

for (;;)

{

cout << "Выберите команду:" << endl;

cout << "1 - Добавление элемента в стек" << endl;

cout << "2 - Извлечение элемента из стека" << endl;

cout << "3 - Запись в файл" << endl;

cout << "4 - Чтение из файла" << endl;

cout << "5 - Вывод стека" << endl;

cout << "6 - Очистка стека" << endl;

cout << "7 - Подсчёт количества элементов стека, у которых равные соседи" << endl;

cout << "0 - Выход" << endl;

cin >> choice;

switch (choice)

{

case 1:

cout << "Введите элемент: " << endl;

cin >> x;

push(x, myStk);

break;

case 2:

x = pop(myStk);

if (x != -1)

cout << "Извлеченный элемент: " << x << endl;

break;

case 3:

toFile(myStk);

break;

case 4:

fromFile(myStk);

break;

case 5:

cout << "Весь стек: " << endl;

show(myStk);

break;

case 6:

clear(myStk);

break;

case 7:

cout << "Количество элементов стека, у которых равные соседи: ";

counterEqualLeftAndRight(myStk);

break;

case 0:

return 0;

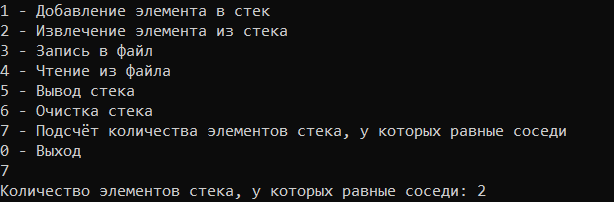
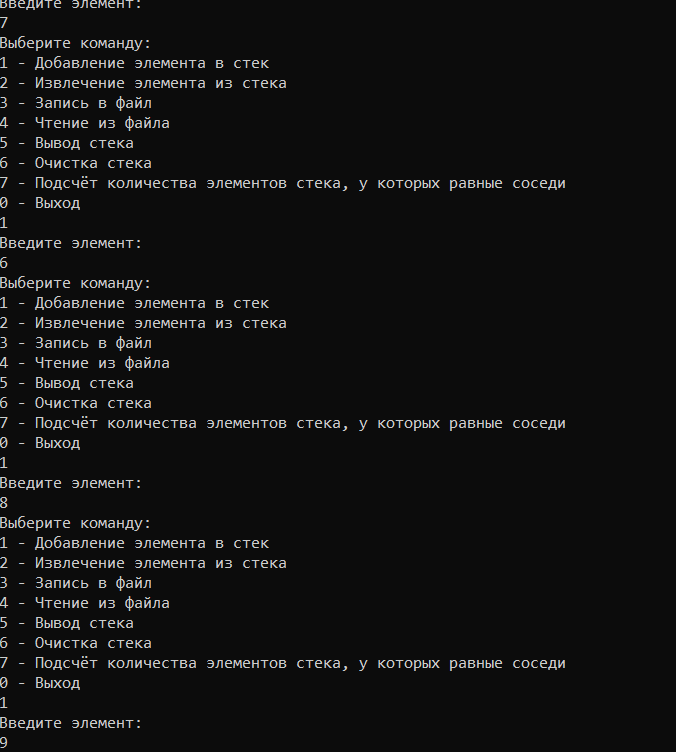
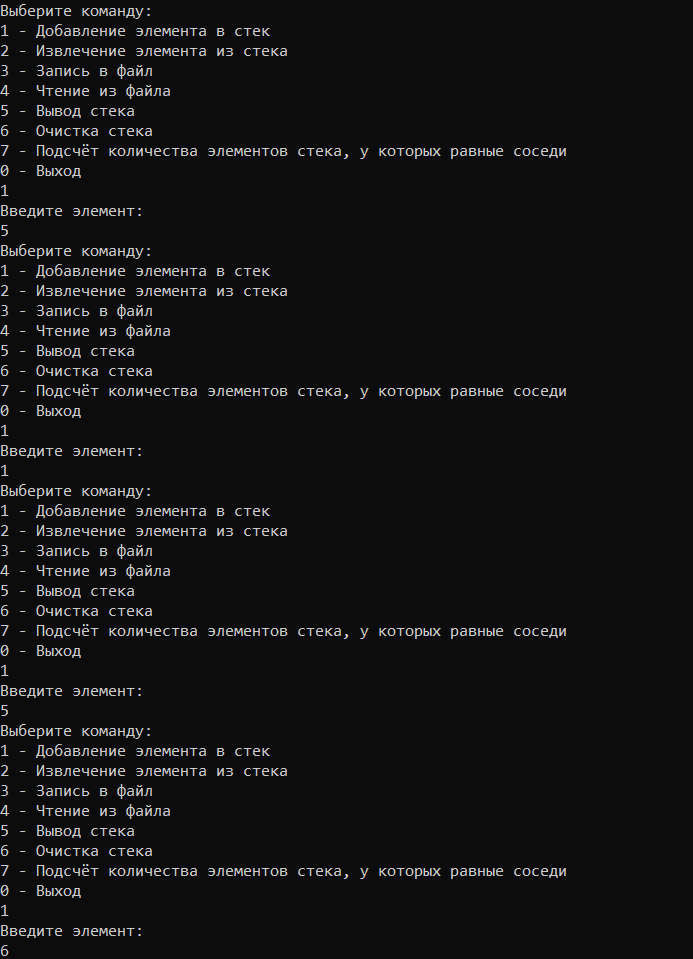
break;

}

}

return 0;

}

****