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 10 группы

Жамойдо Артём Игоревич

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 4**

****

**main.cpp**

#include <iostream>

#include "myStack.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 - Проверка наличия элемента равного следующему" << endl;

cout << "0 - Выход" << endl;

cout << "\nВаш выбор: ";

cin >> choice;

switch (choice)

{

case 1:

cout << "\nВведите элемент: ";

cin >> x;

cout << endl;

push(x, myStk);

break;

case 2:

cout << "\nИзвлеченный элемент: " << pop(myStk) << endl;

break;

case 3:

toFile(myStk);

break;

case 4:

fromFile(myStk);

break;

case 5:

show(myStk);

break;

case 6:

clear(myStk);

break;

case 7:

if (hasEqualNextElement(myStk)) // true

{

cout << "\nВ стеке есть элемент, равный следующему за ним.\n" << endl;

}

else if (!hasEqualNextElement(myStk)) // false

{

cout << "\nВ стеке нет элементов, равных следующему за ним.\n" << endl;

}

break;

case 0:

return 0;

}

}

return 0;

}

**myStack.cpp**

#include <iostream>

#include <fstream>

#include "myStack.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

myStk = e; // Сдвиг вершины на позицию вперед

}

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

{

if (myStk == NULL)

{

cout << "\nСтек пуст!\n\n" << 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;

}

}

// Функция записи в файл

void toFile(Stack\*& myStk)

{

Stack\* e = myStk;

Stack buf;

ofstream frm("text.txt");

if (frm.fail())

{

cout << "\nОшибка открытия файла\n";

exit(1);

}

while (e)

{

buf = \*e;

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

e = e->next;

}

frm.close();

cout << "\nСтек записан в файл text.txt\n\n";

}

// Функция считывания из файла

void fromFile(Stack\*& myStk)

{

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

ifstream frm("text.txt");

if (frm.fail())

{

cout << "\nОшибка открытия файла\n";

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 text.txt\n\n";

}

// Функция вывода

void show(Stack\*& myStk)

{

Stack\* e = myStk;

if (e == NULL)

{

cout << "\nСтек пуст!\n\n";

return;

}

cout << "\nСодержимое стека: ";

while (e != NULL)

{

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

e = e->next;

}

cout << endl << endl;

}

// Функция очистки

void clear(Stack\*& myStk)

{

while (myStk != NULL)

{

pop(myStk);

}

cout << "\nСтек очищен.\n" << endl;

}

// Функция, которая определяет, есть ли в стеке элемент, равный следующему за ним элементу.

bool hasEqualNextElement(Stack\*& myStk)

{

Stack\* current = myStk;

while (current != NULL && current->next != NULL)

{

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

{

return true;

}

current = current->next;

}

return false;

}

**myStack.h**

#pragma once

struct Stack

{

char data;

Stack\* next;

};

// Прототипы

void show(Stack\*& myStk);

char pop(Stack\*& myStk);

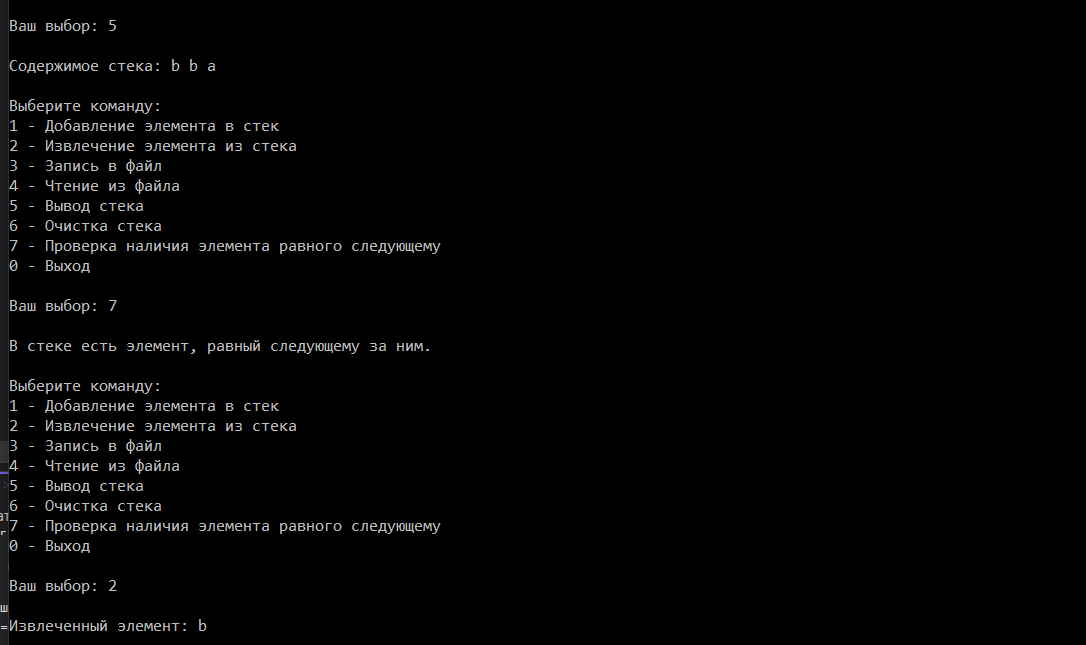
void push(char x, Stack\*& myStk);

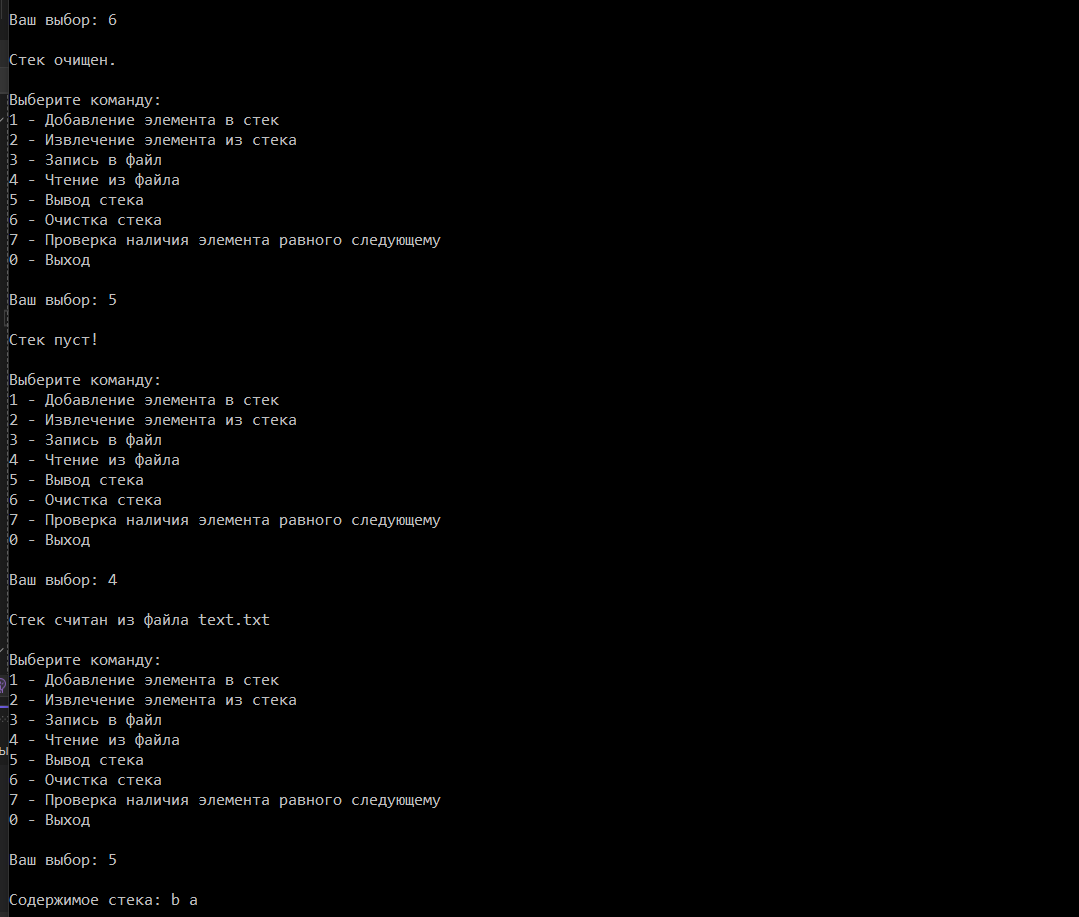
void toFile(Stack\*& myStk);

void fromFile(Stack\*& myStk);

void clear(Stack\*& myStk);

bool hasEqualNextElement(Stack\*& myStk);

****

****

Доп. задачи

**Вариант 10**

Разработать функцию, которая определяет, есть ли в стеке хотя бы один элемент, лежащий в заданном диапазоне.

**Dop\_1.cpp**

#include <iostream>

#include <fstream>

#include <stack>

#include "head.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "Rus");

int choice;

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

myStk->head = 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 - Вывод стека" << endl;

cout << "4 - Ввод стека в файл" << endl;

cout << "5 - Вывод стека из файла" << endl;

cout << "6 - Очистка стека" << endl;

cout << "7 - Поиск элемента из заданого диапазона" << endl;

cout << "8 - Выход" << endl;

cin >> choice;

switch (choice)

{

case 1:

cout << "Введите элемент: " << endl;

cin >> choice;

push(choice, myStk);

break;

case 2:

choice = pop(myStk);

if (choice != -1)

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

break;

case 3:

cout << "Весь стек: " << endl;

show(myStk);

break;

case 5:

cout << "Весь стек выведен из файла" << endl;

OutFile(myStk);

break;

case 4:

cout << "Весь стек записан в файл" << endl;

InFile(myStk);

break;

case 6:

cout << "стек очищен " << endl;

clear(myStk);

break;

case 7:

int sz1, sz2;

cout << "Введите диапазон чисел от ";

cin >> sz1;

cout << "до ";

cin >> sz2;

cout << "Элемент из заданного диапазона: " << endl;

fun(sz1, sz2, myStk);

break;

case 8:

return 0;

break;

}

}

return 0;

}

**myStack.cpp**

#include <iostream>

#include <fstream>

#include <stack>

#include "head.h"

using namespace std;

void push(int 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 v

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

myStk->head = e; //сдвиг вершины на позицию вперед

}

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

{

if (myStk->head == NULL)

{

cout << "Стек пуст!" << endl;

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

}

else

{

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

int a = myStk->head->data; //запись числа из поля data в переменную a

myStk->head = myStk->head->next; //перенос вершины

delete e; //удаление временной переменной

return a; //возврат значения удаляемого элемента

}

}

void show(Stack\* myStk) //Вывод стека

{

Stack\* e = myStk->head; //объявляется указатель на вершину стека

int a;

if (e == NULL)

cout << "Стек пуст!" << endl;

while (e != NULL)

{

a = e->data; //запись значения в переменную a

cout << a << " ";

e = e->next;

}

cout << endl;

}

void InFile(Stack\* myStk)

{

ofstream B("File.txt", ios::app);

if (B.fail())

{

cout << "\n Ошибка открытия файла для записи";

exit(1);

}

Stack\* e = myStk->head; //объявляется указатель на вершину стека

int a;

if (e == NULL)

cout << "Стек пуст!" << endl;

while (e != NULL)

{

a = e->data; //запись значения в переменную a

B << a << " ";

e = e->next;

}

cout << endl;

B.close();

}

void OutFile(Stack\* myStk)

{

ifstream A("File.txt", ios::in);

if (A.fail())

{

cout << "\n Ошибка открытия файла для чтения";

exit(1);

}

char str[40];

while (!A.eof())

{

A.getline(str, 40, '\n');

cout << str << " ";

}

A.close();

}

void clear(Stack\* myStk)

{

myStk->head = NULL;

}

void fun(int sz1, int sz2, Stack\* myStk)

{

Stack\* e = myStk->head; //объявляется указатель на вершину стека

if (e == NULL)

cout << "Стек пуст!" << endl;

int a;

while (e != NULL)

{

a = e->data; //запись значения в переменную a

for (int i = sz1; i <= sz2; i++)

{

if (i == a)

{

cout << i << " ";

}

}

e = e->next;

}

cout << endl;

}

**head.h**

#pragma once

struct Stack

{

int data; //информационный элемент

Stack\* head; //вершина стека

Stack\* next; //указатель на следующий элемент

};

void push(int x, Stack\* myStk);

int pop(Stack\* myStk);

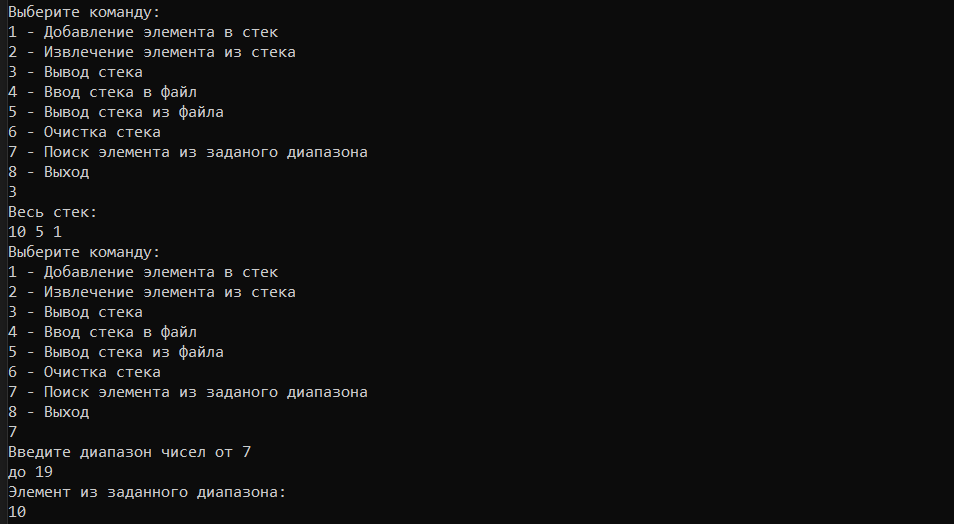
void show(Stack\* myStk);

void InFile(Stack\* myStk);

void OutFile(Stack\* myStk);

void clear(Stack\* myStk);

void fun(int sz1, int sz2, Stack\* myStk);

****

**Вариант 13**

Разработать функцию подсчета количества повторяющихся элементов стека.

**Dop\_2.cpp**

#include <iostream>

#include <fstream>

#include <stack>

#include "head.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "Rus");

int choice;

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

myStk->head = NULL; //инициализация первого элемента

for (;;)

{

cout << "Выберите команду:" << endl;

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

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

cout << "3 - Вывод стека" << endl;

cout << "4 - Ввод стека в файл" << endl;

cout << "5 - Вывод стека из файла" << endl;

cout << "6 - Очистка стека" << endl;

cout << "7 - Подсчет количества повторяющихся элементов стека" << endl;

cout << "8 - Выход" << endl << endl;

cout << "Ваш выбор: ";

cin >> choice;

switch (choice)

{

case 1:

cout << "Введите элемент: " << endl;

cin >> choice;

push(choice, myStk);

break;

case 2:

choice = pop(myStk);

if (choice != -1)

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

break;

case 3:

cout << "Весь стек: " << endl;

show(myStk);

break;

case 4:

cout << "Весь стек записан в файл File.txt " << endl;

InFile(myStk);

break;

case 5:

cout << "Весь стек выведен из файла File.txt " << endl;

OutFile(myStk);

break;

case 6:

cout << "Стек очищен " << endl;

clear(myStk);

break;

case 7:

fun1(myStk);

break;

case 8:

return 0;

break;

}

}

}

**myStack.cpp**

#include <iostream>

#include <fstream>

#include <stack>

#include "head.h"

using namespace std;

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

{

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

e->data = x; //запись элемента x

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

myStk->head = e; //сдвиг вершины на позицию вперед

}

int pop(Stack\* myStk) //Удаление элемента из стека

{

if (myStk->head == NULL)

{

cout << "Стек пуст!" << endl;

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

}

else

{

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

int a = myStk->head->data; //запись числа из поля data в переменную a

myStk->head = myStk->head->next; //перенос вершины

delete e; //удаление временной переменной

return a; //возврат значения удаляемого элемента

}

}

void show(Stack\* myStk) //Вывод стека

{

Stack\* e = myStk->head; //объявляется указатель на вершину стека

int a;

if (e == NULL)

cout << "Стек пуст!" << endl;

while (e != NULL)

{

a = e->data; //запись значения в переменную a

cout << a << " ";

e = e->next;

}

cout << endl;

}

void InFile(Stack\* myStk)

{

ofstream B("File.txt", ios::app);

if (B.fail())

{

cout << "\n Ошибка открытия файла для записи";

exit(1);

}

Stack\* e = myStk->head; //объявляется указатель на вершину стека

int a;

if (e == NULL)

cout << "Стек пуст!" << endl;

while (e != NULL)

{

a = e->data; //запись значения в переменную a

B << a << " ";

e = e->next;

}

cout << endl;

B.close();

}

void OutFile(Stack\* myStk)

{

ifstream A("File.txt", ios::in);

char str[40];

while (!A.eof())

{

A.getline(str, 40, '\n');

cout << str << " ";

}

A.close();

}

void clear(Stack\* myStk)

{

myStk->head = NULL;

}

int fun1(Stack\* myStk) //Подсчет количества повторяющихся элементов стека

{

Stack\* e, \* k;

e = myStk->head;

int count = 0;

int a, b;

cout << "Повторяющиеся элементы: ";

while (e != NULL)

{

a = e->data;

k = e->next;

while (k != NULL)

{

b = k->data;

if (a == b)

{

cout << a << " ";

count++;

break;

}

k = k->next;

}

e = e->next;

}

cout << endl;

cout << "Число повторяющихся элементов: " << count << endl;

return 0;

}

**head.h**

#pragma once

struct Stack

{

int data; //информационный элемент

Stack\* head; //вершина стека

Stack\* next; //указатель на следующий элемент

};

void show(Stack\* myStk);

int pop(Stack\* myStk);

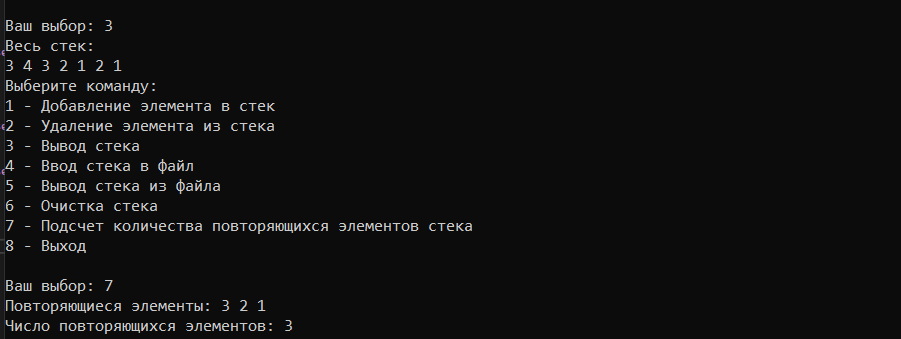
void push(int x, Stack\* myStk);

void InFile(Stack\* myStk);

void OutFile(Stack\* myStk);

void clear(Stack\* myStk);

int fun1(Stack\* myStk);



**Вариант 6**

Разработать функцию, которая удаляет из стека первый отрицательный элемент, если такой есть.

**Dop\_3.cpp**

#include <iostream>

#include "head.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "Rus");

int choice;

int x;

Stack\* myStk = nullptr;

// Главный цикл программы

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 - Удалить первый отрицательный элемент" << endl;

cout << "7 - Выход" << endl;

cin >> choice;

// Обработка выбора пользователя

switch (choice)

{

case 1:

cout << "Введите элемент: " << endl;

cin >> x;

push(x, myStk); // добавление элемента в стек

break;

case 2:

x = pop(myStk); // извлечение элемента из стека

if (x != -1)

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

break;

case 3:

toFile(myStk); // запись стека в файл

break;

case 4:

fromFile(myStk); // чтение стека из файла

break;

case 5:

cout << "Весь стек: " << endl;

show(myStk); // вывод стека

break;

case 6:

removeFirstNegative(myStk); // удалить первый отрицательный элемент

break;

case 7:

return 0;

default:

cout << "Неверная команда. Попробуйте снова\n";

}

}

return 0;

}

**myStack.cpp**

#include <iostream>

#include <fstream>

#include "head.h"

using namespace std;

void push(int x, Stack\*& myStk)

{

Stack\* e = new Stack;

e->data = x;

e->next = myStk;

e->prev = NULL;

// Обновление указателя на предыдущий элемент

if (myStk)

{

myStk->prev = e;

}

myStk = e;

}

int pop(Stack\*& myStk)

{

// Проверка на пустой стек

if (myStk == NULL)

{

cout << "Стек пуст!" << endl;

return -1;

}

else

{

// Извлечение верхнего элемента

Stack\* e = myStk;

int x = myStk->data;

myStk = myStk->next;

// Обновление указателя на предыдущий элемент

if (myStk)

{

myStk->prev = NULL;

}

delete e;

return x;

}

}

void toFile(Stack\*& myStk)

{

Stack\* e = myStk;

Stack buf;

// Открытие файла для записи

ofstream frm("File.txt");

if (frm.fail())

{

cout << "\nОшибка открытия файла\n";

exit(1);

}

// Запись стека в файл

while (e)

{

buf = \*e;

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

e = e->next;

}

// Закрытие файла

frm.close();

cout << "Стек записан в файл File.txt\n";

}

void fromFile(Stack\*& myStk)

{

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

// Открытие файла для чтения

ifstream frm("File.txt");

if (frm.fail())

{

cout << "\nОшибка открытия файла\n";

exit(1);

}

// Чтение стека из файла

frm.seekg(0);

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

while (!frm.eof())

{

push(buf.data, e);

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

}

frm.close();

// Перестройка стека

while (e != NULL)

{

buf.data = pop(e);

push(buf.data, p);

myStk = p;

}

// Вывод сообщения об успешном чтении

cout << "\nСтек считан из файла File.txt\n\n";

}

// Функция вывода

void show(Stack\*& myStk)

{

Stack\* e = myStk;

if (e == NULL)

{

cout << "\nСтек пуст!\n\n";

return;

}

cout << "\nСодержимое стека: ";

while (e != NULL)

{

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

e = e->next;

}

cout << endl << endl;

}

void removeFirstNegative(Stack\*& myStk)

{

Stack\* current = myStk;

// Поиск и удаление первого отрицательного элемента

while (current != NULL)

{

if (current->data < 0)

{

if (current->prev)

{

current->prev->next = current->next;

}

if (current->next)

{

current->next->prev = current->prev;

}

if (current == myStk)

{

myStk = myStk->next;

}

delete current;

cout << "\nПервый отрицательный элемент удален\n";

return;

}

current = current->next;

}

cout << "\nОтрицательный элемент не найден\n";

}

**head.h**

#pragma once

struct Stack

{

int data; // Информационный элемент

Stack\* next; // Указатель на следующий элемент

Stack\* prev; // Указатель на предыдущий элемент

};

// Прототипы

void show(Stack\*& myStk);

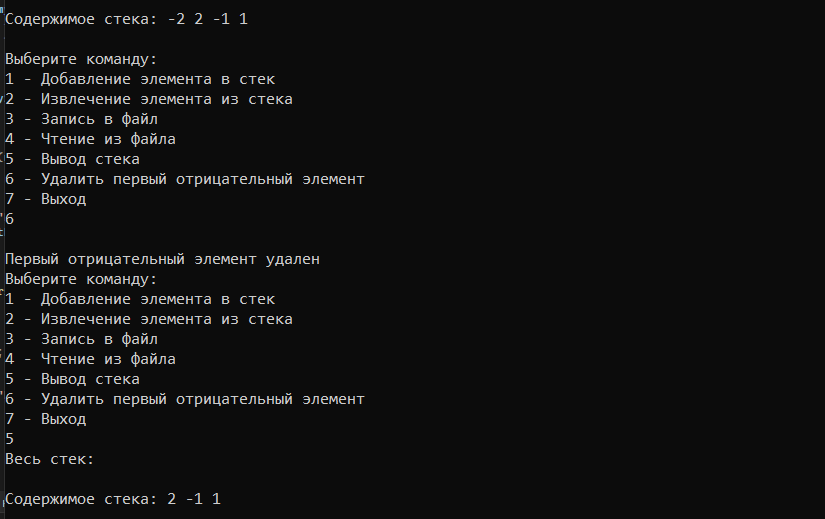
int pop(Stack\*& myStk);

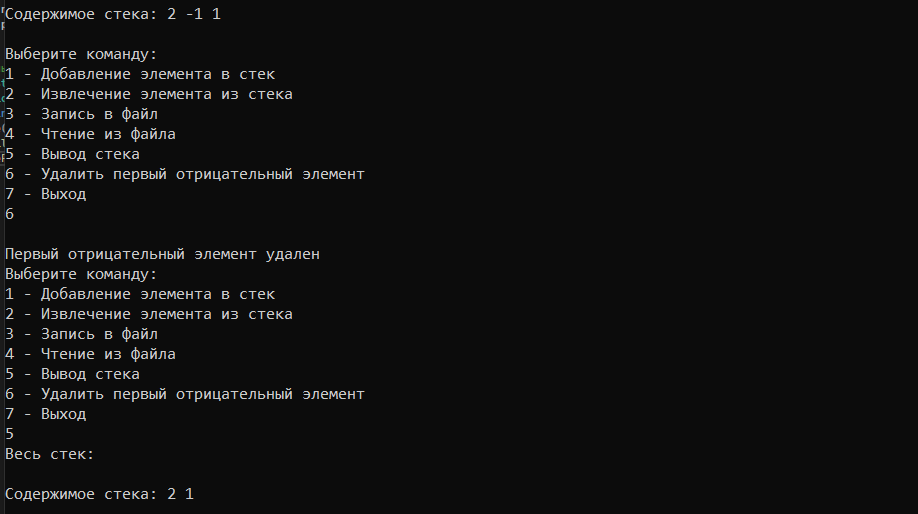
void push(int x, Stack\*& myStk);

void toFile(Stack\*& myStk);

void fromFile(Stack\*& myStk);

void removeFirstNegative(Stack\*& myStk);

****

****