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(а):

Студент(ка) 1 курса 8 группы

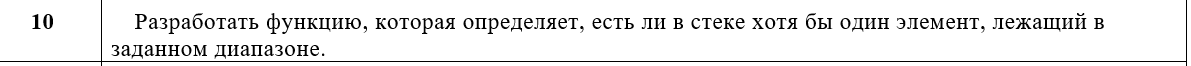
Семёнов Даниил Вячеславович

Преподаватель: асс. Андронова М.В.

Минск, 2024

4. 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 10



Main.cpp

#include <iostream>

#include <Windows.h>

#include "myStack10.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "Rus");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int choice; int x;

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

myStk = NULL; //инициализация первого элемента

do

{

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 - Поиск элемента из диапазона" << endl;

cout << "8 - 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: find(myStk); break;

}

} while (choice != 8);

return 0;

}

myStack10.cpp

#include <iostream>

#include <fstream>

#include "myStack10.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 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 << "Стек пуст!" << endl;

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

}

else

{

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

int 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 в стек

{

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) // функция очистки

{

Stack\* e = myStk;

if (e == NULL)

{

cout << "Стек пуст!" << endl;

return;

}

else

{

while (e != NULL)

{

Stack\* temp = e; // сохраняем текущий элемент

e = e->next; // переходим к следующему элементу

delete temp; // освобождаем память текущего элемента

}

myStk = NULL;

cout << "Стек очищен";

}

cout << endl;

}

void find(Stack\*& myStk) // функция поиска в стеке из диапазона

{

int min, max;

bool f = true;

cout << "Введите начало диапазона: "; cin >> min;

cout << "Введите конец диапазона: "; cin >> max;

Stack\* e = myStk;

while (e != NULL)

{

if (e->data >= min && e->data <= max && f)

{

cout << endl<<"Элемент из диапазона найден"<<endl;

f = false;

}

e = e->next;

}

if (f)

{

cout <<endl<<"Элементов не найдено"<<endl; // Если ни один элемент не нашелся в заданном диапазоне

}

}

myStack10.h

struct Stack

{

int data; //информационный элемент

Stack\* next; //указатель на следующий элемент

};

void show(Stack\*& myStk); // функция вывода

char pop(Stack\*& myStk); // функция извлечения элемента

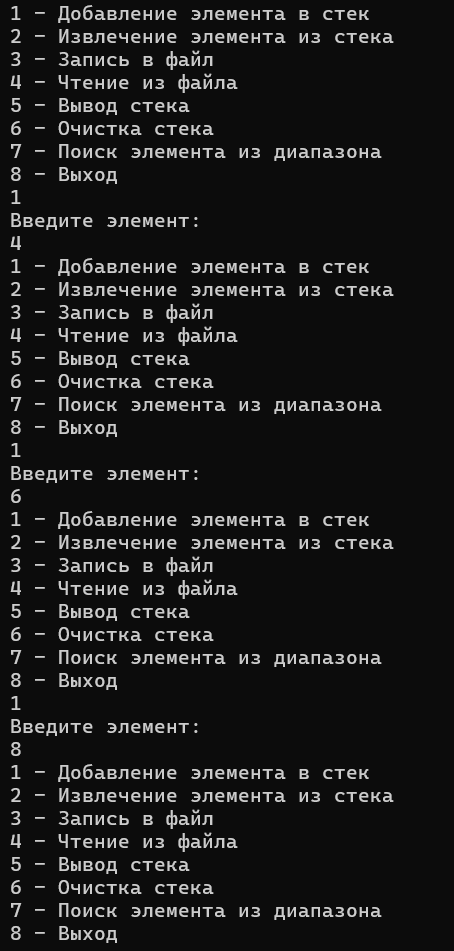
void push(int x, Stack\*& myStk); // функция ввода

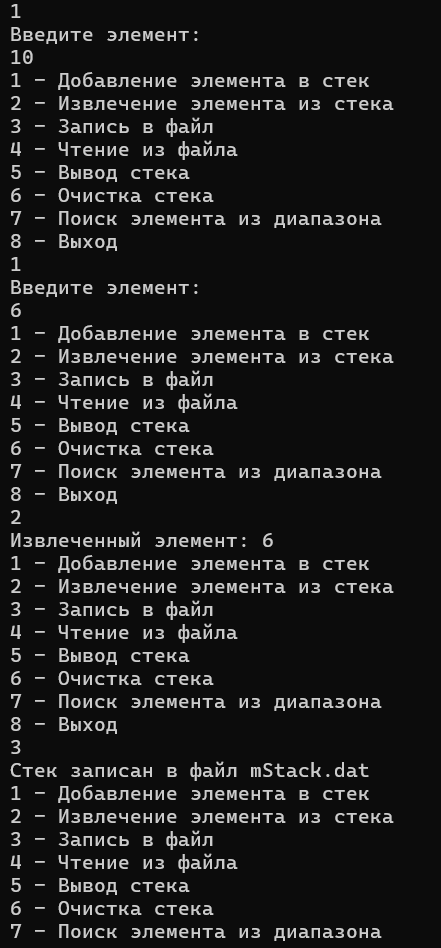
void toFile(Stack\*& myStk); // функция записи в файл

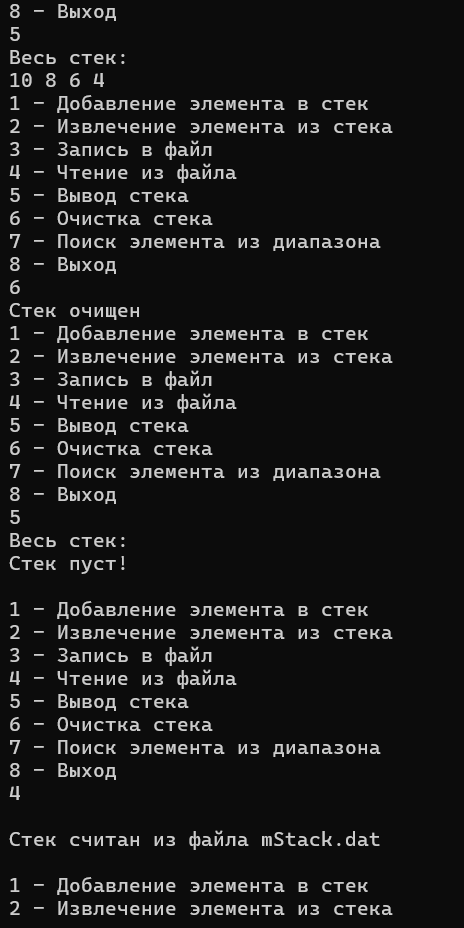
void fromFile(Stack\*& myStk); // функция чтения из файла в стек

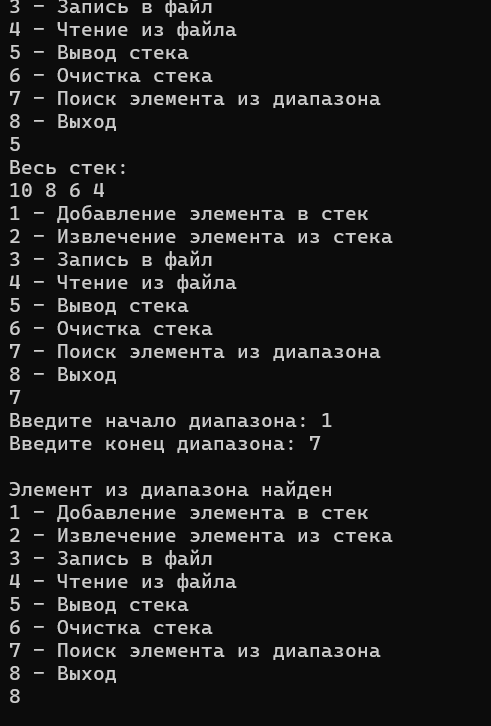
void clear(Stack\*& myStk); // функция очистки

void find(Stack\*& myStk); // функция поиска в стека из заданного диапазона









Доп. Задачи

Вариант 2



dop2.cpp

#include <iostream>

#include <Windows.h>

#include "myStack2.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "Rus");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int choice; int x;

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

myStk = NULL; //инициализация первого элемента

do

{

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 - Поиск элемента" << endl;

cout << "8 - Выход" << endl;

cin >> choice;

// анализируем 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: find(myStk); break;

}

} while (choice != 8);

return 0;

}

myStack2.cpp

#include <iostream>

#include <fstream>

#include "myStack2.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 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 << "Стек пуст!" << endl;

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

}

else

{

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

int 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 в стек

{

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

{

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) // функция очистки

{

Stack\* e = myStk;

if (e == NULL)

{

cout << "Стек пуст!" << endl;

return;

}

else

{

while (e != NULL)

{

Stack\* temp = e; // сохраняем текущий элемент

e = e->next; // переходим к следующему элементу

delete temp; // освобождаем память текущего элемента

}

myStk = NULL;

cout << "Стек очищен";

}

cout << endl;

}

void find(Stack\*& myStk) // функция поиска и удаления первого отрицательного элемента в стеке

{

bool f = true;

Stack\* prev = NULL;

Stack\* e = myStk;

while (e != NULL)

{

if (e->data < 0 && f)

{

if (prev == NULL) // если отрицательный элемент - первый в стеке

{

myStk = e->next;

}

else

{

prev->next = e->next;

}

int x = e->data;

delete e;

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

f = false;

break; // выходим из цикла после удаления элемента

}

prev = e;

e = e->next;

}

if (f)

{

cout << endl << "Отрицательных элементов не найдено" << endl; // Если ни один элемент не нашелся

}

}

myStack2.h

struct Stack

{

int data; //информационный элемент

Stack\* next; //указатель на следующий элемент

};

void show(Stack\*& myStk); // функция вывода

char pop(Stack\*& myStk); // функция извлечения элемента

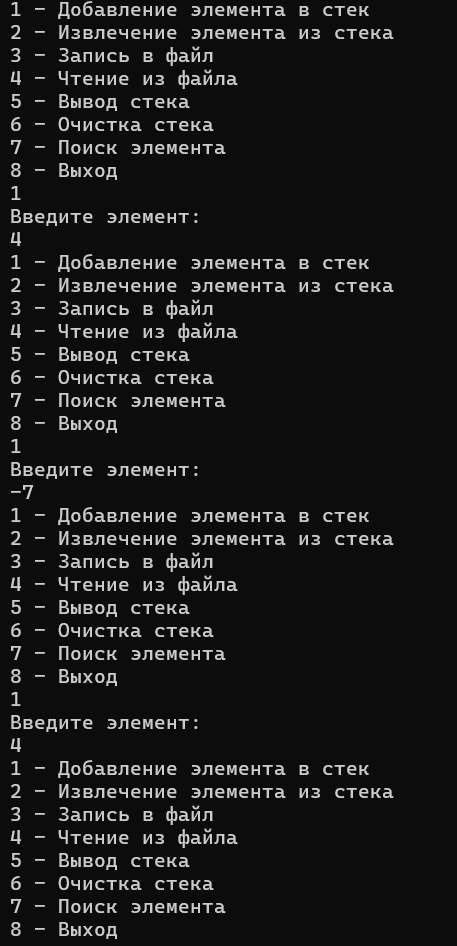
void push(int x, Stack\*& myStk); // функция ввода

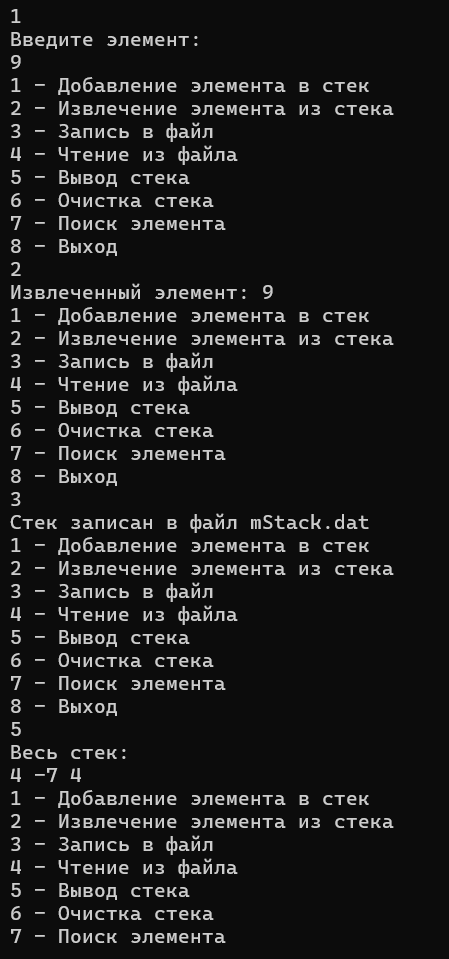
void toFile(Stack\*& myStk); // функция записи в файл

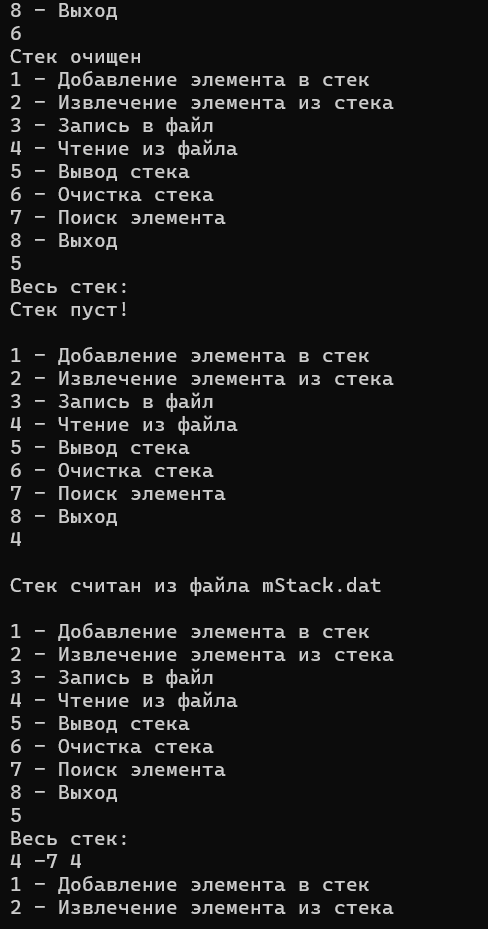
void fromFile(Stack\*& myStk); // функция чтения из файла в стек

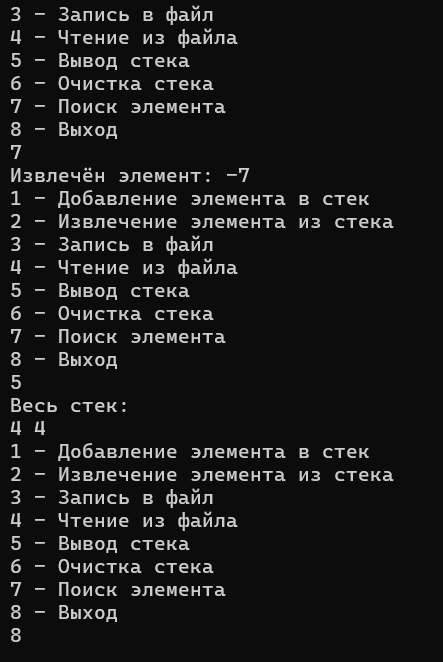
void clear(Stack\*& myStk); // функция очистки

void find(Stack\*& myStk); // функция поиска и удаления первого отрицательного элемента в стеке









Вариант 3



dop3.cpp  
 #include <iostream>

#include <Windows.h>

#include "myStack3.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "Rus");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int choice; int x;

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

myStk = NULL; //инициализация первого элемента

do

{

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 - Поиск элемента" << endl;

cout << "8 - Выход" << endl;

cin >> choice;

// анализируем 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: find(myStk); break;

}

} while (choice != 8);

return 0;

}

myStack3.cpp

#include <iostream>

#include <fstream>

#include "myStack3.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 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 << "Стек пуст!" << endl;

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

}

else

{

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

int 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 в стек

{

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

{

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) // функция очистки

{

Stack\* e = myStk;

if (e == NULL)

{

cout << "Стек пуст!" << endl;

return;

}

else

{

while (e != NULL)

{

Stack\* temp = e; // сохраняем текущий элемент

e = e->next; // переходим к следующему элементу

delete temp; // освобождаем память текущего элемента

}

myStk = NULL;

cout << "Стек очищен";

}

cout << endl;

}

void find(Stack\*& myStk) // функция поиска и удаления первого положительного элемента в стеке

{

bool f = true; // для ограничения поиска чисел

Stack\* prev = NULL;

Stack\* e = myStk;

while (e != NULL)

{

if (e->data > 0 && f)

{

if (prev == NULL) // если положительный элемент - первый в стеке

{

myStk = e->next;

}

else

{

prev->next = e->next;

}

int x = e->data;

delete e;

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

f = false;

break; // выходим из цикла после удаления элемента

}

prev = e;

e = e->next;

}

if (f)

{

cout << endl << "Положительных элементов не найдено" << endl; // Если ни один элемент не нашелся

}

}

myStack3.h

struct Stack

{

int data; //информационный элемент

Stack\* next; //указатель на следующий элемент

};

void show(Stack\*& myStk); // функция вывода

char pop(Stack\*& myStk); // функция извлечения элемента

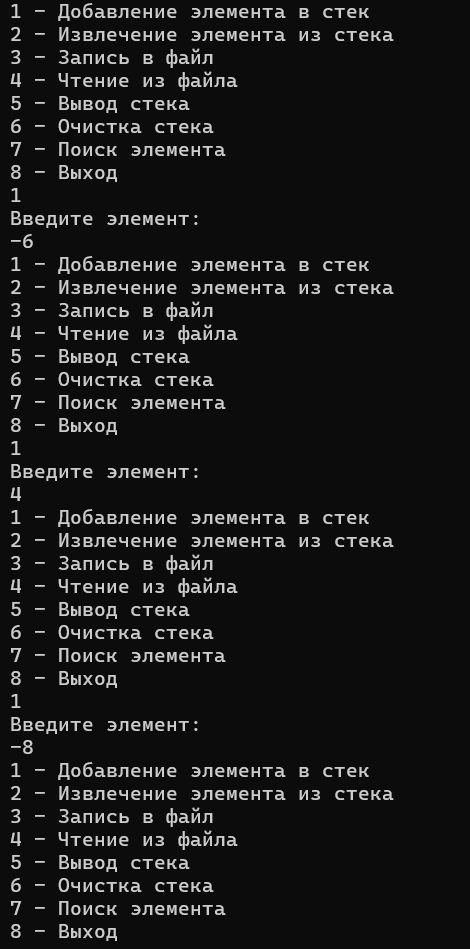
void push(int x, Stack\*& myStk); // функция ввода

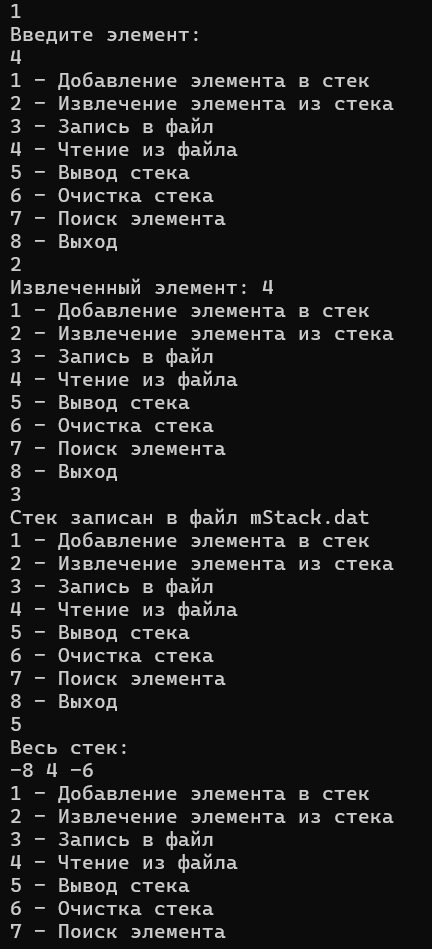
void toFile(Stack\*& myStk); // функция записи в файл

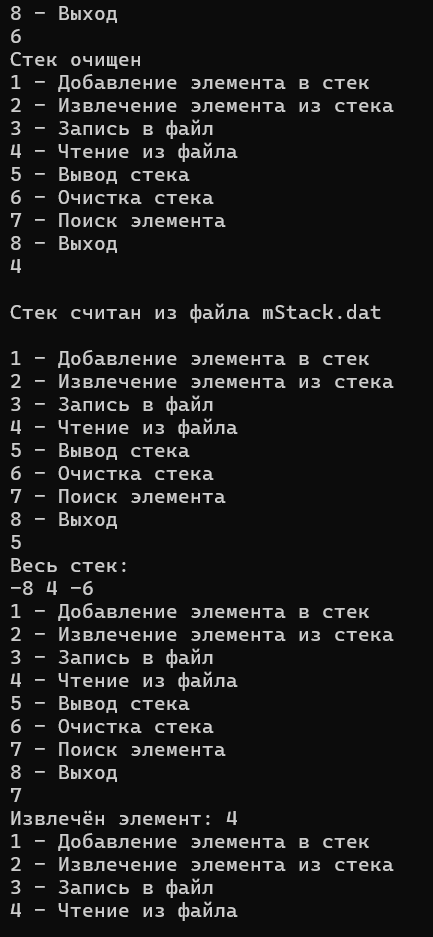
void fromFile(Stack\*& myStk); // функция чтения из файла в стек

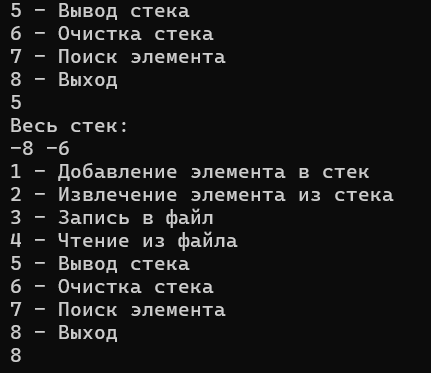
void clear(Stack\*& myStk); // функция очистки

void find(Stack\*& myStk); // функция поиска и удаления первого положительного элемента в стеке









Вариант 7



dop7.cpp

#include <iostream>

#include <Windows.h>

#include "myStack7.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "Rus");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int choice; int x;

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

myStk = NULL; //инициализация первого элемента

do

{

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 - Поиск элемента" << endl;

cout << "8 - Выход" << endl;

cin >> choice;

// анализируем 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: find(myStk); break;

}

} while (choice != 8);

return 0;

}

myStcak7.cpp

#include <iostream>

#include <fstream>

#include "myStack7.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 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 << "Стек пуст!" << endl;

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

}

else

{

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

int 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 в стек

{

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

{

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) // функция очистки

{

Stack\* e = myStk;

if (e == NULL)

{

cout << "Стек пуст!" << endl;

return;

}

else

{

while (e != NULL)

{

Stack\* temp = e; // сохраняем текущий элемент

e = e->next; // переходим к следующему элементу

delete temp; // освобождаем память текущего элемента

}

myStk = NULL;

cout << "Стек очищен";

}

cout << endl;

}

void find(Stack\*& myStk) // функция поиска и удаления первого элемента больше 100 в стеке

{

bool f = true; // для ограничения поиска чисел

Stack\* prev = NULL;

Stack\* e = myStk;

while (e != NULL)

{

if (e->data > 100 && f)

{

if (prev == NULL) // если нужный элемент - первый в стеке

{

myStk = e->next;

}

else

{

prev->next = e->next;

}

int x = e->data;

delete e;

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

f = false;

break; // выходим из цикла после удаления элемента

}

prev = e;

e = e->next;

}

if (f)

{

cout << endl << "нужных элементов не найдено" << endl; // Если ни один элемент не нашелся

}

}

myStack7.h

struct Stack

{

int data; //информационный элемент

Stack\* next; //указатель на следующий элемент

};

void show(Stack\*& myStk); // функция вывода

char pop(Stack\*& myStk); // функция извлечения элемента

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

void toFile(Stack\*& myStk); // функция записи в файл

void fromFile(Stack\*& myStk); // функция чтения из файла в стек

void clear(Stack\*& myStk); // функция очистки

void find(Stack\*& myStk); // функция поиска и удаления первого элемента больше 100 в стеке

