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»

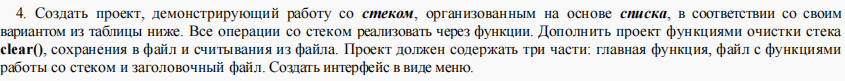
Выполнил:

Студент 1 курса 6 группы

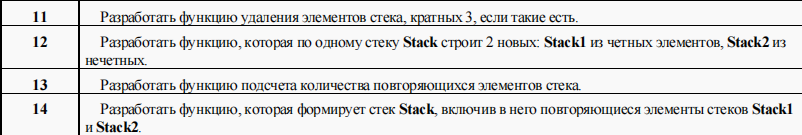
Романов Игорь Вячеславович

Преподаватель: асс. Андронова М.В.

2024, Минск



Вариант 12 (основной)



**stack.cpp (функция с меню)**

#include "stack.h"

using namespace std;

int main() {

setlocale(LC\_ALL, "Rus");

int choice; char x;

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

myStack = NULL; //инициализация первого элемента

do {

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 - Разбить Stack на Stack1 из четных и на Stack2 из нечетных элементов" << endl;

cout << "7 - Очистить стек" << endl;

cout << "0 - Выход" << endl;

cin >> choice;

switch (choice) {

case 1: {

cout << "Введите элемент: " << endl;

cin >> x;

inputStack(x, myStack);

break;

}

case 2: {

x = getElement(myStack);

if (x != -1)

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

break;

}

case 3: {

toFile(myStack);

break;

}

case 4: {

fromFile(myStack);

break;

}

case 5: {

cout << "Весь стек: " << endl;

displayStack(myStack);

break;

}

case 6: {

breakdownStack(myStack);

break;

}

case 7: {

if (!clearStack(myStack)) {

cout << "Стек пуст!" << endl;

}

else {

cout << "Стек очищен" << endl;

}

break;

}

case 0: {

return 0;

break;

}

default: {

cout << "Выбран неверный вариант!" << endl;

}

}

} while (choice != 0);

return 0;

}

**stack\_functions.cpp (функция с функциями программы)**

#include "stack.h"

void inputStack(int x, Stack\*& myStack) { //Добавление элемента х в стек

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

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

element->next = myStack; //перенос вершины на следующий элемент

myStack = element; //сдвиг вершины на позицию вперед

}

int getElement(Stack\*& myStack) {

if (myStack == NULL) {

cout << "Стек пуст!" << endl;

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

}

else {

Stack\* element = myStack; // element - переменная для хранения адреса элемента

int x = myStack->data; // запись элемента из поля data в переменную x

myStack = myStack->next; // перенос вершины

delete element; // освобождение памяти

return x;

}

}

void toFile(Stack\*& myStack) {

ofstream ofile("myStack.txt");

if (ofile.fail()) {

cout << "\nОшибка открытия файла";

exit(1);

}

Stack\* temp = myStack; // Создаем временный указатель для обхода стека

while (temp != NULL) {

ofile << temp->data << " "; // Записываем данные элемента в файл

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

}

ofile.close();

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

}

void fromFile(Stack\*& myStack) {

ifstream ifile("myStack.txt");

int temp;

if (ifile.fail()) {

cout << "\nОшибка открытия файла";

exit(1);

}

while (ifile >> temp) {

inputStack(temp, myStack); // Используем ссылку на указатель myStack для добавления элемента в стек

}

ifile.close();

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

}

void displayStack(Stack\*& myStack) { // Вывод стека

Stack\* temp = myStack;

if (temp == NULL)

cout << "Стек пуст!" << endl;

while (temp != NULL) {

cout << temp->data << " ";

temp = temp->next;

}

cout << endl;

}

int clearStack(Stack\*& myStack) {

if (myStack == NULL) { //если стек пуст

return 0;

}

while (myStack != NULL) { //пока стек не пуст

Stack\* temp = myStack; //временная переменная для текущего элемента стека

myStack = myStack->next; //переключаемся на следующей элемент стека

delete temp; //удаялем текущий элемент

}

return 1;

}

//разбиение стека на стек1 четных и на стек2 из нечетных элементов

void breakdownStack(Stack\*& myStack) {

Stack\* Stack1 = new Stack; //создание доп.стеков

Stack1 = NULL;

Stack\* Stack2 = new Stack;

Stack2 = NULL;

int element;

while (myStack != NULL) {

element = getElement(myStack);

if (element % 2 == 0) { //если элемент четный, то добавить в 1

inputStack(element, Stack1);

}

else { //иначе добавить в 2

inputStack(element, Stack2);

}

}

cout << "Стек1 из нечетных элементов: "; displayStack(Stack1);

cout << "\nСтек2 из нечетных элементов: "; displayStack(Stack2); cout << endl;

}

**stack.h (заголовочный файл)**

#include <iostream>

#include <fstream>

using namespace std;

struct Stack

{

int data; //информационный элемент

Stack\* next; //указатель на следующий элемент

};

void displayStack(Stack\*& myStack);

int getElement(Stack\*& myStack);

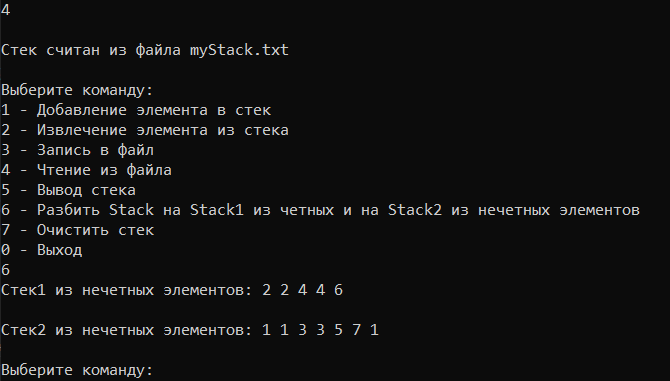
void inputStack(int x, Stack\*& myStack);

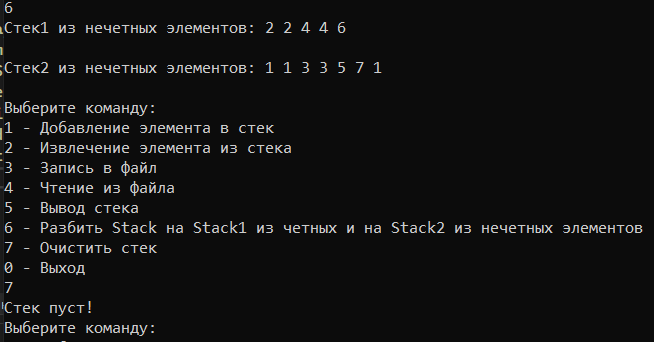
void toFile(Stack\*& myStack);

void fromFile(Stack\*& myStack);

void breakdownStack(Stack\*& myStack);

int clearStack(Stack\*& MyStack);





Доп. задания

Вариант 10



**stack.cpp (функция с меню)**

#include "stack.h"

using namespace std;

int main() {

setlocale(LC\_ALL, "Rus");

int choice; char x;

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

myStack = NULL; //инициализация первого элемента

do {

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 - Определить, есть ли числа в веденном диапазоне чисел" << endl;

cout << "7 - Очистить стек" << endl;

cout << "0 - Выход" << endl;

cin >> choice;

switch (choice) {

case 1: {

cout << "Введите элемент: " << endl;

cin >> x;

inputStack(x, myStack);

break;

}

case 2: {

x = getElement(myStack);

if (x != -1)

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

break;

}

case 3: {

toFile(myStack);

break;

}

case 4: {

fromFile(myStack);

break;

}

case 5: {

cout << "Весь стек: " << endl;

displayStack(myStack);

break;

}

case 6: {

if (!defineElement(myStack)) {

cout << "Нет чисел в стеке, попадающих в введенный диапазон" << endl;

}

else {

cout << "Есть числа в стеке, попадающие в веденные диапазон" << endl;

}

break;

}

case 7: {

if (!clearStack(myStack)) {

cout << "Стек пуст!" << endl;

}

else {

cout << "Стек очищен" << endl;

}

break;

}

case 0: {

return 0;

break;

}

default: {

cout << "Выбран неверный вариант!" << endl;

}

}

} while (choice != 0);

return 0;

}

**stack\_functions.cpp (функция с функциями программы)**

#include "stack.h"

void inputStack(int x, Stack\*& myStack) { //Добавление элемента х в стек

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

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

element->next = myStack; //перенос вершины на следующий элемент

myStack = element; //сдвиг вершины на позицию вперед

}

int getElement(Stack\*& myStack) {

if (myStack == NULL) {

cout << "Стек пуст!" << endl;

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

}

else {

Stack\* element = myStack; // element - переменная для хранения адреса элемента

int x = myStack->data; // запись элемента из поля data в переменную x

myStack = myStack->next; // перенос вершины

delete element; // освобождение памяти

return x;

}

}

void toFile(Stack\*& myStack) {

ofstream ofile("myStack.txt");

if (ofile.fail()) {

cout << "\nОшибка открытия файла";

exit(1);

}

Stack\* temp = myStack; // Создаем временный указатель для обхода стека

while (temp != NULL) {

ofile << temp->data << " "; // Записываем данные элемента в файл

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

}

ofile.close();

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

}

void fromFile(Stack\*& myStack) {

ifstream ifile("myStack.txt");

int temp;

if (ifile.fail()) {

cout << "\nОшибка открытия файла";

exit(1);

}

while (ifile >> temp) {

inputStack(temp, myStack); // Используем ссылку на указатель myStack для добавления элемента в стек

}

ifile.close();

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

}

void displayStack(Stack\*& myStack) { // Вывод стека

Stack\* temp = myStack;

if (temp == NULL)

cout << "Стек пуст!" << endl;

while (temp != NULL) {

cout << temp->data << " ";

temp = temp->next;

}

cout << endl;

}

int clearStack(Stack\*& myStack) {

if (myStack == NULL) { //если стек пуст

return 0;

}

while (myStack != NULL) { //пока стек не пуст

Stack\* temp = myStack; //временная переменная для текущего элемента стека

myStack = myStack->next; //переключаемся на следующей элемент стека

delete temp; //удаялем текущий элемент

}

return 1;

}

//определение элемента в границе диапазона чисел

int defineElement(Stack\*& myStack) {

int a, b; //левая и правая границы диапазона чисел

cout << "Введите диапазон чисел [a, b]: ";

cin >> a >> b;

int element;

while (myStack != NULL) {

element = getElement(myStack); //извлечение элемента

if (a <= element && element <= b) { //если элемент попадает в диапазон чисел

return 1;

}

}

return 0;

}

**stack.h (заголовочный файл)**

#include <iostream>

#include <fstream>

using namespace std;

struct Stack

{

int data; //информационный элемент

Stack\* next; //указатель на следующий элемент

};

void displayStack(Stack\*& myStack);

int getElement(Stack\*& myStack);

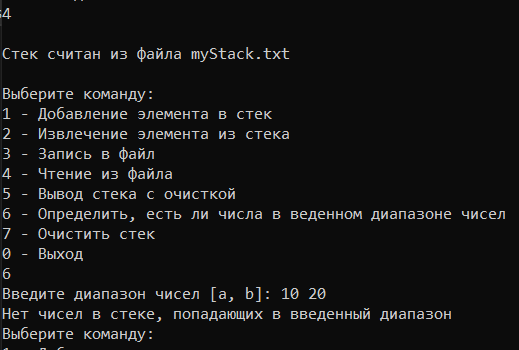
void inputStack(int x, Stack\*& myStack);

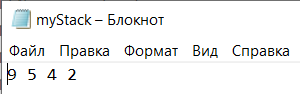
void toFile(Stack\*& myStack);

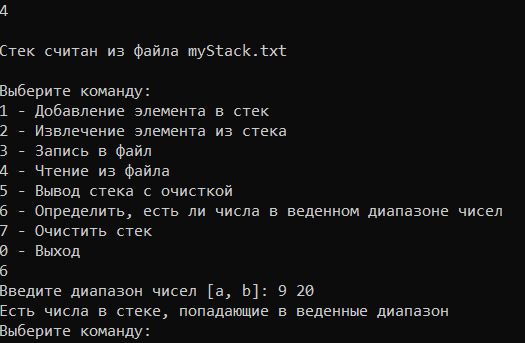
void fromFile(Stack\*& myStack);

int defineElement(Stack\*& myStack);

int clearStack(Stack\*& MyStack);

****

****

****

Вариант 11



**stack.cpp (функция с меню)**

#include "stack.h"

using namespace std;

int main() {

setlocale(LC\_ALL, "Rus");

int choice; char x;

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

myStack = NULL; //инициализация первого элемента

do {

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 из стека числа, кратные 3" << endl;

cout << "7 - Очистить стек" << endl;

cout << "0 - Выход" << endl;

cin >> choice;

switch (choice) {

case 1: {

cout << "Введите элемент: " << endl;

cin >> x;

inputStack(x, myStack);

break;

}

case 2: {

x = getElement(myStack);

if (x != -1)

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

break;

}

case 3: {

toFile(myStack);

break;

}

case 4: {

fromFile(myStack);

break;

}

case 5: {

cout << "Весь стек: " << endl;

displayStack(myStack);

break;

}

case 6: {

defineElement(myStack);

cout << "Обновленный стек: ";

displayStack(myStack);

break;

}

case 7: {

if (!clearStack(myStack)) {

cout << "Стек пуст!" << endl;

}

else {

cout << "Стек очищен" << endl;

}

break;

}

case 0: {

return 0;

break;

}

default: {

cout << "Выбран неверный вариант!" << endl;

}

}

} while (choice != 0);

return 0;

}

**stack\_functions.cpp (функция с функциями программы)**

#include "stack.h"

void inputStack(int x, Stack\*& myStack) { //Добавление элемента х в стек

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

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

element->next = myStack; //перенос вершины на следующий элемент

myStack = element; //сдвиг вершины на позицию вперед

}

int getElement(Stack \* &myStack) {

if (myStack == NULL) {

cout << "Стек пуст!" << endl;

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

} else {

Stack\* element = myStack; // element - переменная для хранения адреса элемента

int x = myStack->data; // запись элемента из поля data в переменную x

myStack = myStack->next; // перенос вершины

delete element; // освобождение памяти

return x;

}

}

void toFile(Stack\*& myStack) {

ofstream ofile("myStack.txt");

if (ofile.fail()) {

cout << "\nОшибка открытия файла";

exit(1);

}

Stack\* temp = myStack; // Создаем временный указатель для обхода стека

while (temp != NULL) {

ofile << temp->data << " "; // Записываем данные элемента в файл

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

}

ofile.close();

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

}

void fromFile(Stack\*& myStack) {

ifstream ifile("myStack.txt");

int temp;

if (ifile.fail()) {

cout << "\nОшибка открытия файла";

exit(1);

}

while (ifile >> temp) {

inputStack(temp, myStack); // Используем ссылку на указатель myStack для добавления элемента в стек

}

ifile.close();

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

}

void displayStack(Stack\*& myStack) { // Вывод стека

Stack\* temp = myStack;

if (temp == NULL)

cout << "Стек пуст!" << endl;

while (temp != NULL) {

cout << temp->data << " ";

temp = temp->next;

}

cout << endl;

}

int clearStack(Stack \*& myStack) {

if (myStack == NULL) { //если стек пуст

return 0;

}

while (myStack != NULL) { //пока стек не пуст

Stack\* temp = myStack; //временная переменная для текущего элемента стека

myStack = myStack->next; //переключаемся на следующей элемент стека

delete temp; //удаялем текущий элемент

}

return 1;

}

//удаление элемента кратного 3

void defineElement(Stack\*& myStack) {

Stack\* current = myStack;

Stack\* prev = NULL;

while (current != NULL) {

if (current->data % 3 == 0) { // Проверка на кратность 3

Stack\* temp = current;

current = current->next;

delete temp; // Удаляем текущий элемент

if (prev != NULL) {

prev->next = current; // Обновляем указатель предыдущего элемента

}

else {

myStack = current; // Обновляем указатель на начало стека

}

}

else {

prev = current;

current = current->next; // Переходим к следующему элементу

}

}

}

**stack.h (заголовочный файл)**

#include <iostream>

#include <fstream>

using namespace std;

struct Stack

{

int data; //информационный элемент

Stack\* next; //указатель на следующий элемент

};

void displayStack(Stack\*& myStack);

int getElement(Stack\*& myStack);

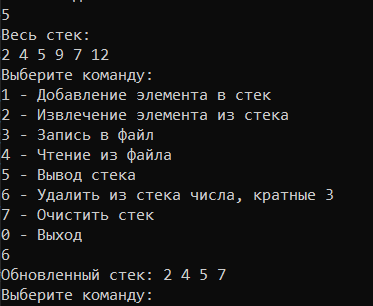
void inputStack(int x, Stack\*& myStack);

void toFile(Stack\*& myStack);

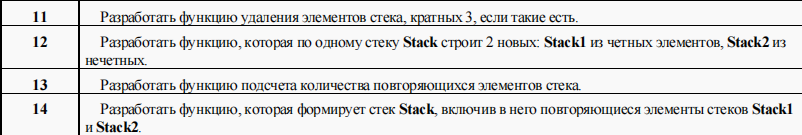
void fromFile(Stack\*& myStack);

void defineElement(Stack\*& myStack);

int clearStack(Stack \*& MyStack);

****

Вариант 13



**stack.cpp (функция с меню)**

#include "stack.h"

using namespace std;

int main() {

setlocale(LC\_ALL, "Rus");

int choice; char x;

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

myStack = NULL; //инициализация первого элемента

do {

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 - Подсчет повторных элементов в стеке" << endl;

cout << "7 - Очистить стек" << endl;

cout << "0 - Выход" << endl;

cin >> choice;

switch (choice) {

case 1: {

cout << "Введите элемент: " << endl;

cin >> x;

inputStack(x, myStack);

break;

}

case 2: {

x = getElement(myStack);

if (x != -1)

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

break;

}

case 3: {

toFile(myStack);

break;

}

case 4: {

fromFile(myStack);

break;

}

case 5: {

cout << "Весь стек: " << endl;

displayStack(myStack);

break;

}

case 6: {

countDuplicates(myStack);

break;

}

case 7: {

if (!clearStack(myStack)) {

cout << "Стек пуст!" << endl;

}

else {

cout << "Стек очищен" << endl;

}

break;

}

case 0: {

return 0;

break;

}

default: {

cout << "Выбран неверный вариант!" << endl;

}

}

} while (choice != 0);

return 0;

}

**stack\_functions.cpp (функция с функциями программы)**

#include "stack.h"

void inputStack(int x, Stack\*& myStack) { //Добавление элемента х в стек

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

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

element->next = myStack; //перенос вершины на следующий элемент

myStack = element; //сдвиг вершины на позицию вперед

}

int getElement(Stack \* &myStack) {

if (myStack == NULL) {

cout << "Стек пуст!" << endl;

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

} else {

Stack\* element = myStack; // element - переменная для хранения адреса элемента

int x = myStack->data; // запись элемента из поля data в переменную x

myStack = myStack->next; // перенос вершины

delete element; // освобождение памяти

return x;

}

}

void toFile(Stack\*& myStack) {

ofstream ofile("myStack.txt");

if (ofile.fail()) {

cout << "\nОшибка открытия файла";

exit(1);

}

Stack\* temp = myStack; // Создаем временный указатель для обхода стека

while (temp != NULL) {

ofile << temp->data << " "; // Записываем данные элемента в файл

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

}

ofile.close();

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

}

void fromFile(Stack\*& myStack) {

ifstream ifile("myStack.txt");

int temp;

if (ifile.fail()) {

cout << "\nОшибка открытия файла";

exit(1);

}

while (ifile >> temp) {

inputStack(temp, myStack); // Используем ссылку на указатель myStack для добавления элемента в стек

}

ifile.close();

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

}

void displayStack(Stack\*& myStack) { // Вывод стека

Stack\* temp = myStack;

if (temp == NULL)

cout << "Стек пуст!" << endl;

while (temp != NULL) {

cout << temp->data << " ";

temp = temp->next;

}

cout << endl;

}

int clearStack(Stack \*& myStack) {

if (myStack == NULL) { //если стек пуст

return 0;

}

while (myStack != NULL) { //пока стек не пуст

Stack\* temp = myStack; //временная переменная для текущего элемента стека

myStack = myStack->next; //переключаемся на следующей элемент стека

delete temp; //удаялем текущий элемент

}

return 1;

}

//подсчет повторных элементов в стеке

void countDuplicates(Stack\*& myStack) {

if (myStack == NULL) {

cout << "Стек пуст!" << endl;

}

else {

Stack\* current = myStack; //изначальный стек

Stack\* check\_duplicates = myStack; //стек для сравнения с изначальным

Stack\* duplicates = {}; //стек с дубликатами

int nDuplicates = 0; //подсчет повторов элементов

while (current != NULL) {

int check\_current = current->data; //извлекаем элемент из текущего стека

while (check\_duplicates != NULL) { //сравнием этот же элемент с каждым элементом стека, кроме него самого

if (check\_current == check\_duplicates->data && current != check\_duplicates) { //если повторяющийся элемент найден

bool isDuplicate = false; //проверяем найденное число со стеком дубликатов

Stack\* temp = duplicates;

while (temp != NULL) {

if (check\_current == temp->data) {

isDuplicate = true;

break;

}

temp = temp->next;

}

if (!isDuplicate) {

nDuplicates++;

inputStack(check\_current, duplicates); //добавляем элемент в стек с повторками

}

break;

}

check\_duplicates = check\_duplicates->next; //переходим к следующему элементу

}

current = current->next; //переходим к следующему элементу в стеке

check\_duplicates = myStack; //возвращаем стек для проверки в исходное положение

}

if (nDuplicates) {

cout << "Найдено дубликатов в стеке: " << nDuplicates << endl;

}

else {

cout << "В стеке нет повторяющихся элементов" << endl;

}

}

}

**stack.h (заголовочный файл)**

#include <iostream>

#include <fstream>

using namespace std;

struct Stack

{

int data; //информационный элемент

Stack\* next; //указатель на следующий элемент

};

void displayStack(Stack\*& myStack);

int getElement(Stack\*& myStack);

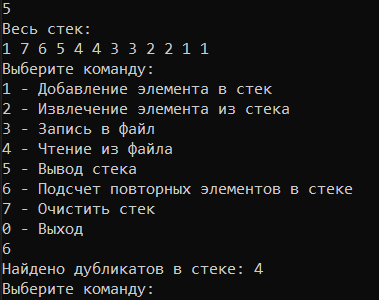
void inputStack(int x, Stack\*& myStack);

void toFile(Stack\*& myStack);

void fromFile(Stack\*& myStack);

void countDuplicates(Stack\*& myStack);

int clearStack(Stack \*& MyStack);

****