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:

Рауба Арсений

Студент 1 курса 8 группы

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 №9

Разработать функцию, которая формирует стек **Stack**, включив в него по одному разу элементы, которые входят в один из стеков **Stack1** и **Stack2**, но не входят в другой.

#include <iostream>

#include "myStack.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "Rus");

int choice; char x;

Stack1\* myStk1 = new Stack1;

Stack2\* myStk2 = new Stack2; //выделение памяти для стека

Stack3\* myStk3 = new Stack3;

myStk1 = NULL;

myStk2 = NULL;

myStk3 = NULL;//инициализация первого элемента

for (;;)

{

cout << "Выберите команду:" << endl;

cout << "1 - Добавление элемента в стек №1" << endl;

cout << "2 - Добавление элемента в стек №2" << endl;

cout << "3 - Извлечение элемента из стека №1" << endl;

cout << "4 - Извлечение элемента из стека №2" << endl;

cout << "5 - Запись в третий стек" << endl;

cout << "6 - Запись в файл" << endl;

cout << "7 - Чтение из файла" << endl;

cout << "8 - Выход" << endl;

cin >> choice;

switch (choice)

{

case 1: cout << "Введите элемент: " << endl;

cin >> x;

push1(x, myStk1); break;

case 2: cout << "Введите элемент: " << endl;

cin >> x;

push2(x, myStk2); break;

case 3: x = popStack1(myStk1);

if (x != -1)

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

break;

case 4: x = popStack2(myStk2);

if (x != -1)

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

break;

case 5:

push3(myStk3, myStk1, myStk2);

while (myStk3 != NULL) {

cout << "qq: ->> " << myStk3 << endl;

}

break;

case 6:

toFile(myStk1);

break;

case 7:

fromFile(myStk1);

break;

case 8: return 0;

break;

}

}

return 0;

}

#include <iostream>

#include <fstream>

#include "myStack.h"

using namespace std;

void push1(int x, Stack1\*& myStk1) //Добавление элемента х в стек

{

Stack1\* e1 = new Stack1;

cout << "->" << x << endl;//выделение памяти для нового элемента

e1->data1 = x; //запись элемента x в поле data

e1->next = myStk1; //перенос вершины на следующий элемент

myStk1 = e1; //сдвиг вершины на позицию вперед

}

void push2(int y, Stack2\*& myStk2) //Добавление элемента y в стек

{

Stack2\* e2 = new Stack2;

cout << "->" << y << endl; //выделение памяти для нового элемента

e2->data2 = y; //запись элемента y в поле data

e2->next = myStk2; //перенос вершины на следующий элемент

myStk2 = e2; //сдвиг вершины на позицию вперед

}

void push3(Stack3\*& myStk3, Stack1\*& myStk1, Stack2\*& myStk2)

{

int i = 0, j = 0;

bool flag = true;

Stack1\* e = myStk1;

Stack2\* d = myStk2;

Stack3\* c = myStk3;

while (e != NULL) { // подсчет элементов 1 стека

i++;

e = e->next;

}

e = myStk1;

while (d != NULL) { // подсчет элементов 2 стека

j++;

d = d->next;

}

d = myStk2;

for (int a = 0; a < i; a++) {

for (int b = 0; b < j; b++) {

if (e->data1 == d->data2) {

flag = false;

}

}

if (flag == true) {

Stack3\* c = new Stack3;

c->data3 = e->data1;

c->next = myStk3;

myStk3 = c;

d = d->next;

d = myStk2;

}

e = e->next;

flag = true;

}

e = myStk1;

d = myStk2;

c = myStk3;

while (c != NULL) { // вывод

cout << c->data3;

cout << "->>";

c = c->next;

}

c = myStk3;

}

char popStack1(Stack1\*& myStk1) //Извлечение элемента из стека №1

{

if (myStk1 == NULL)

{

cout << "Стек пуст!" << endl;

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

}

else

{

Stack1\* e = myStk1; //переменная для хранения адреса элемента

char x = myStk1->data1; //запись элемента из поля data в x

if (myStk1)

myStk1 = myStk1->next; //перенос вершины

delete e;

return x;

}

}

char popStack2(Stack2\*& myStk2) //Извлечение элемента из стека №2

{

if (myStk2 == NULL)

{

cout << "Стек пуст!" << endl;

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

}

else

{

Stack2\* d = myStk2; //переменная для хранения адреса элемента

char y = myStk2->data2; //запись элемента из поля data в y

if (myStk2)

myStk2 = myStk2->next; //перенос вершины

delete d;

return y;

}

}

void toFile(Stack1\*& myStk1) //Запись в файл

{

Stack1\* e = myStk1;

Stack1 buf;

ofstream frm("Stack1.txt");

while (e)

{

buf = \*e;

frm.write((char\*)&buf, sizeof(Stack1));

e = e->next;

}

frm.close();

cout << "Стек записан\n";

}

void fromFile(Stack1\*& myStk1) //Считывание из файла

{

Stack1 buf, \* p = nullptr, \* e = nullptr;

ifstream frm("Stack1.txt");

frm.seekg(0);

frm.read((char\*)&buf, sizeof(Stack1));

while (!frm.eof())

{

push1(buf.data1, e);

frm.read((char\*)&buf, sizeof(Stack1));

}

frm.close();

while (e != NULL)

{

buf.data1 = popStack1(e);

push1(buf.data1, p);

myStk1 = p;

}

cout << "\nСтек считан\n\n";

}

#pragma once

struct Stack1

{

int data1; //информационный элемент

Stack1\* next; //указатель на следующий элемент

};

struct Stack2

{

int data2;

Stack2\* next;

};

struct Stack3

{

int data3;

Stack3\* next;

};

char popStack1(Stack1\*& myStk1);

char popStack2(Stack2\*& myStk2);

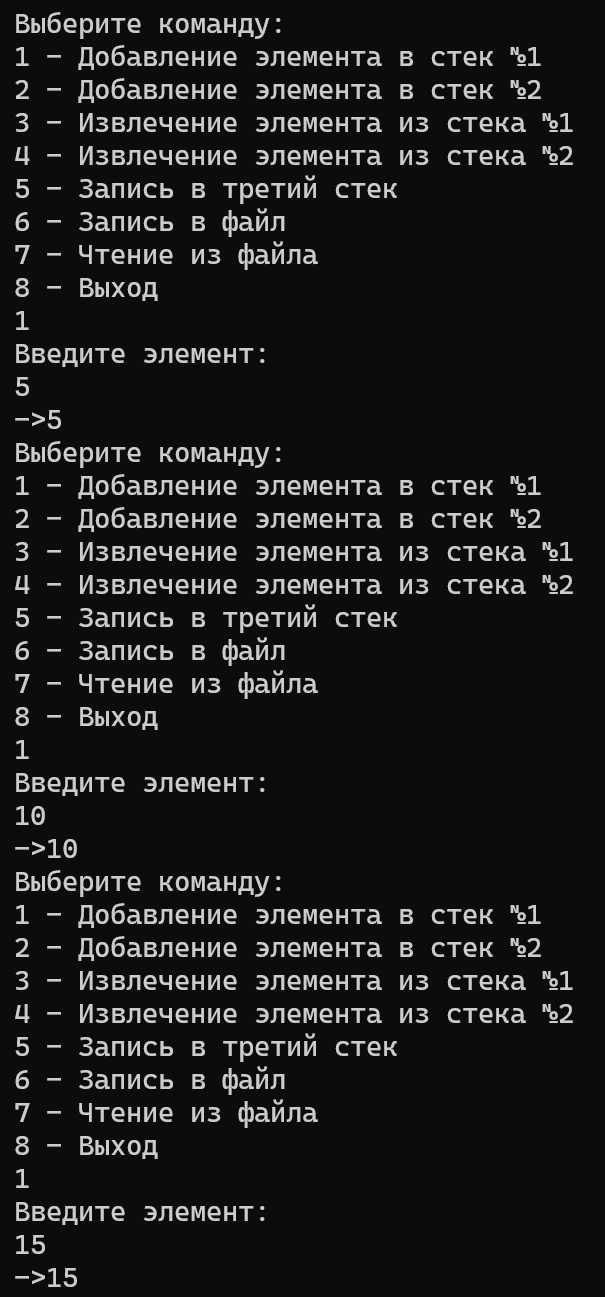
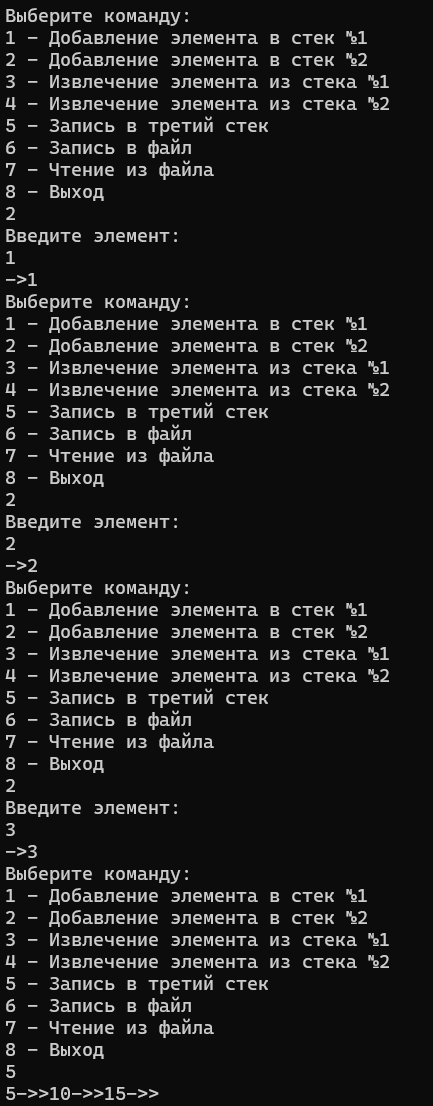
void push1(int x, Stack1\*& myStk1);

void push2(int y, Stack2\*& myStk2);

void push3(Stack3\*& myStk3, Stack1\*& myStk1, Stack2\*& myStk2);

void toFile(Stack1\*& myStk1);

void fromFile(Stack1\*& myStk1);

****

**Доп.Задачи**

Вариант №2

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

#include<iostream>

#include"myStack.h"

Using namespace std;

int main()

{

setlocale(LC\_ALL, "Rus");

intchoice; int x;

Stack\* myStk = newStack; myStk = NULL;

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 - Удаление первого отрицательного элемента"<<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 0:

return 0;

break;

}

}

return 0;

}

#include<iostream>

#include<fstream>

#include"myStack.h"

usingnamespacestd;

voidpush(intx, Stack\*&myStk) //Добавление элемента х в стек

{

Stack\* e = newStack; //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

}

charpop(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;

}

}

voidtoFile(Stack\*&myStk) //Запись в файл

{

Stack\* e = myStk;

Stackbuf;

ofstreamfrm("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";

}

voidfromFile(Stack\*&myStk) //Считывание из файла

{

Stackbuf, \* p = nullptr, \* e = nullptr;

ifstreamfrm("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<<pop(e) <<" ";

cout<<endl;

}

void clear(Stack\*&myStk) {

boolflag = false;

Stack\* e = myStk;

Stack\* previous, \* current, \* temp;

if (e == NULL) {

cout<<"Список пуст"<<endl;

return;

}

if (e->data< 0) {

temp = e;

myStk = e->next;

deletetemp;

}

else {

previous = e;

current = e->next;

while (current != NULL&&current->data> 0) {

previous = current;

current = current->next; // перейти к следующему

}

if (current != NULL) {

temp = current;

previous->next = current->next;

deletetemp;

}

}

cout<<"Первый положительный элемент удален"<<endl;

// Печать значений после удаления

e = myStk;

while (e != NULL) {

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

e = e->next;

}

cout<<"-->NULL"<<endl;

}

struct Stack

{

intdata; //информационный элемент

Stack\* next; //указатель на следующий элемент

};

voidshow(Stack\*&myStk);

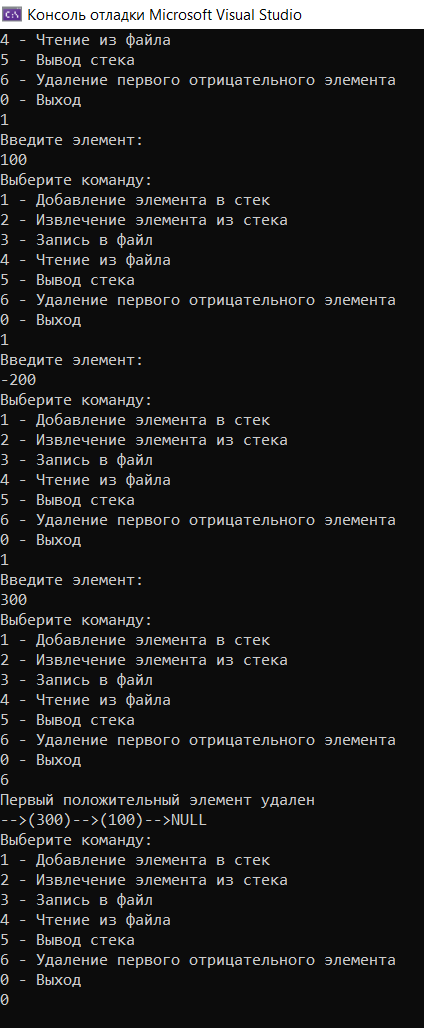
charpop(Stack\*&myStk);

voidpush(intx, Stack\*&myStk);

voidtoFile(Stack\*&myStk);

voidfromFile(Stack\*&myStk);

voidclear(Stack\*&myStk);



Вариант №3

Разработать функцию, которая удаляет первый положительный элемент, если такой есть.

#include<iostream>

#include"myStack.h"

using namespace std;

Int main()

{

setlocale(LC\_ALL, "Rus");

int choice; int x;

Stack\* myStk = new Stack; myStk = NULL;

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 - Удаление первого положительного элемента"<<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:

find\_same(myStk);

break;

case 0:

return 0;

break;

}

}

return 0;

}

#include<iostream>

#include<fstream>

#include"myStack.h"

usingnamespacestd;

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

{

Stack\* e = newStack; //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

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;

Stackbuf;

ofstreamfrm("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

{

Stackbuf, \* p = nullptr, \* e = nullptr;

ifstreamfrm("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<<pop(e) <<" ";

cout<<endl;

}

Void clear(Stack\*&myStk) {

Bool flag = false;

Stack\* e = myStk;

Stack\* previous, \* current, \* temp;

if (e == NULL) {

cout<<"Список пуст"<<endl;

return;

}

if (e->data> 0) {

temp = e;

myStk = e->next;

deletetemp;

}

else {

previous = e;

current = e->next;

while (current != NULL&&current->data< 0) {

previous = current;

current = current->next; // перейти к следующему

}

if (current != NULL) {

temp = current;

previous->next = current->next;

deletetemp;

}

}

cout<<"Первый положительный элемент удален"<<endl;

// Печать значений после удаления

e = myStk;

while (e != NULL) {

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

e = e->next;

}

cout<<"-->NULL"<<endl;

}

struct Stack

{

intdata; //информационный элемент

Stack\* next; //указатель на следующий элемент

};

voidshow(Stack\*&myStk);

charpop(Stack\*&myStk);

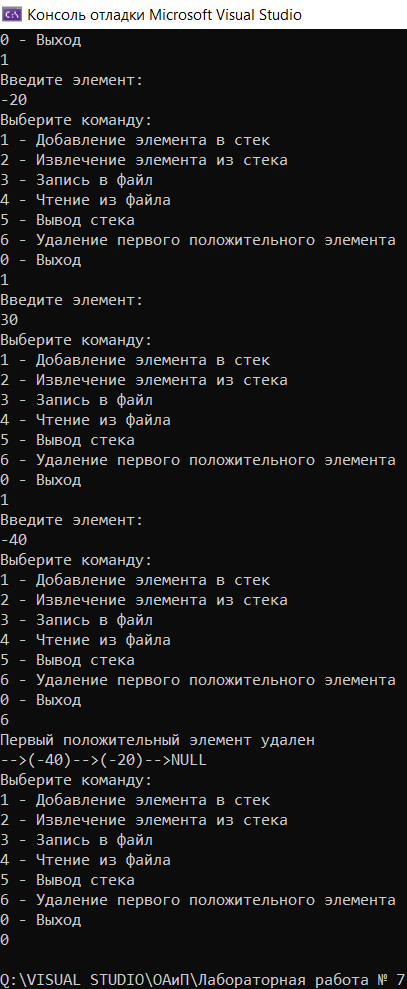
voidpush(intx, Stack\*&myStk);

voidtoFile(Stack\*&myStk);

voidfromFile(Stack\*&myStk);

voidclear(Stack\*&myStk);

voidfind\_same(Stack\*&myStk);



Вариант №4

Разработать функцию, которая определяет, есть ли в стеке элемент, равный следующему за ним элементу.

#include<iostream>

#include"myStack.h"

usingnamespacestd;

intmain()

{

setlocale(LC\_ALL, "Rus");

intchoice; int x;

Stack\* myStk = newStack; myStk = NULL;

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 - Есть ли в стеке элемент, равный следующему за ним элементу?"<<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 0:

return 0;

break;

}

}

return 0;

}

#include<iostream>

#include<fstream>

#include"myStack.h"

usingnamespacestd;

voidpush(intx, Stack\*&myStk) //Добавление элемента х в стек

{

Stack\* e = newStack; //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

}

charpop(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;

}

}

voidtoFile(Stack\*&myStk) //Запись в файл

{

Stack\* e = myStk;

Stackbuf;

ofstreamfrm("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";

}

voidfromFile(Stack\*&myStk) //Считывание из файла

{

Stackbuf, \* p = nullptr, \* e = nullptr;

ifstreamfrm("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";

}

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

{

Stack\* e = myStk;

if (e == NULL)

cout<<"Стек пуст!"<<endl;

while (e != NULL)

cout<<pop(e) <<" ";

cout<<endl;

}

voidclear(Stack\*&myStk) { // Сравнивание следующих элементов

boolflag = false; intbuf;

Stack\* e = myStk;

if (e == NULL) {

cout<<"Список пуст"<<endl;

}

while (e->next != NULL) {

if (e->data == e->next->data) {

flag = true;

}

e = e->next;

}

if (flag == true) cout<<"Eсть в стеке элемент, равный следующему за ним элементу."<<endl;

if (flag != true) cout<<"Eсть в стеке нет элемента, равного следующему за ним элементу."<<endl;

}

structStack

{

intdata; //информационный элемент

Stack\* next; //указатель на следующий элемент

};

voidshow(Stack\*&myStk);

charpop(Stack\*&myStk);

voidpush(intx, Stack\*&myStk);

voidtoFile(Stack\*&myStk);

voidfromFile(Stack\*&myStk);

voidclear(Stack\*&myStk);

voidfind\_same(Stack\*&myStk);

