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»

 Выполнила:

Студентка1 курса 6 группы

Альшевская Алина Михайловна

Преподаватель: асс. Андронова М.В.

2023, Минск

Вариант 1

Задание 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.

****

#pragma once

struct MyStack{

int data;

MyStack\* next;

};

//прототипы

void push\_to\_stack(MyStack\*& stack, int element);

void print\_stack(MyStack\*& stack);

void to\_file(MyStack\* stack);

void clear\_stack(MyStack\*& stack);

void devide\_stack();

//

//

#include "Header.h"

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

void push\_to\_stack(MyStack\*& stack, int element)//функция для вставки элементов в стек

{

MyStack\* new\_elem = new MyStack;//память для элемента

new\_elem->data = element;//вставить элемент

new\_elem->next = stack;//переместить вершину стека на следующую позицию

stack = new\_elem;//переместить вершину вперед

}

void print\_stack(MyStack\*& stack)//функция для печати стека

{

MyStack\* current = stack;//создать указатель для ссылки на стек

if (current == NULL) {

cout << "Стек пуст" << endl;

return;

}

else {

cout << "Стек: ";

while (current != nullptr) {//цикл для печати всех элементов

cout << current->data << ' ';

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

}

cout << endl;

}

}

void clear\_stack(MyStack\*& stack)//функция для очистки всего стека

{

if (stack == NULL) {//если стек пуст

cout << "Стек уже пуст" << endl;

return;

}

else {//если нет

MyStack\* temp = stack;//для хранения значения

while (temp != NULL) {

MyStack\* next\_elem = temp->next;//переместить указатель

delete temp;//удалить элемент

temp = next\_elem;

stack = temp;//переместить указатель в основной функции

}

cout << "Стек очищен" << endl;

}

}

void to\_file(MyStack\* stack)//функция для записи информации в файл

{

ofstream to\_file("file.txt");//открыть файл для записи информации

MyStack\* current = stack;

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

to\_file << ' ';

}

else {

while (current != NULL) {//цикл для вставки всех элементов

to\_file << current->data << ';';

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

}

}

to\_file.close();//закрыть файл

}

void devide\_stack()

{

ifstream from\_file("file.txt");//открыть файл для чтения

MyStack\* pos\_stack = new MyStack;//память для стека

pos\_stack = NULL;//установить указатель на первый элемент

MyStack\* neg\_stack = new MyStack;

neg\_stack = NULL;

string from\_str;

cin.ignore();//очистить буфер

getline(from\_file, from\_str);//получить строку из файла

int str\_length = from\_str.length();

int amm\_of\_values = 0;

for (int i = 0; i < str\_length; i++) {//цикл для подсчета количества значений в исходном стеке

if (from\_str[i] == ';') {

amm\_of\_values++;

}

}

int digit\_num = 0;

for (int i = 0; i < amm\_of\_values; i++) {//цикл для обработки всех значений

string curr\_value;

while (from\_str[digit\_num] != ';') {//цикл для получения текущего значения

curr\_value += from\_str[digit\_num];

digit\_num++;

}

digit\_num++;//перепрыгнуть через знак разделителя

int value = stoi(curr\_value);//преобразовать строку в int

if (value >= 0) {//если это четные числа

MyStack\* new\_pos = new MyStack;//память для элемента

new\_pos->data = value;

new\_pos->next = pos\_stack;//переместить вершину

pos\_stack = new\_pos;

}

else {

//все аналогично

MyStack\* new\_neg = new MyStack;

new\_neg->data = value;

new\_neg->next = neg\_stack;

neg\_stack = new\_neg;

}

}

from\_file.close();//закрыть файл

//печать четного стека

MyStack\* pos\_current = pos\_stack;

if (pos\_current == NULL) {

cout << "В исходном стеке нет положительных элементов" << endl;

}

else {

cout << "Положителный стек: ";

while (pos\_current != NULL) {

cout << pos\_current->data << ' ';

pos\_current = pos\_current->next;

}

cout << endl;

}

//печать нечетного стека

MyStack\* neg\_current = neg\_stack;

if (neg\_current == NULL) {

cout << "В исходном стеке нет отрицательных элементов" << endl;

}

else {

cout << "Отрицательный стек: ";

while (neg\_current != NULL) {

cout << neg\_current->data << ' ';

neg\_current = neg\_current->next;

}

cout << endl;

}

}

//

//

#include <iostream>

#include "Header.h"

using namespace std;

void main()//главная функция

{

setlocale(LC\_ALL, "Russian");

MyStack\* stack = new MyStack;//память для стека

stack = NULL;//инициализация первого элемента

char choise;

do {

cout << "1-Добавить элементы в стек" << endl;

cout << "2-Печать стека" << endl;

cout << "3-Очистить стек" << endl;

cout << "4-Очистить консоль" << endl;

cout << "5-Разделить стек на два" << endl;

cout << "6-Выход" << endl;

cin >> choise;

int choise\_sw = choise - '0';//преобразование char в int

switch (choise\_sw) {

case 1:int size;

cout << "Введите, сколько элементов вы хотели бы в стеке: ";

cin >> size;

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

int element;

cout << "Введите элемент: ";

cin >> element;

push\_to\_stack(stack, element);

}

to\_file(stack);

break;

case 2:print\_stack(stack); break;

case 3:clear\_stack(stack); break;

case 4:system("cls"); break;

case 5:devide\_stack(); break;

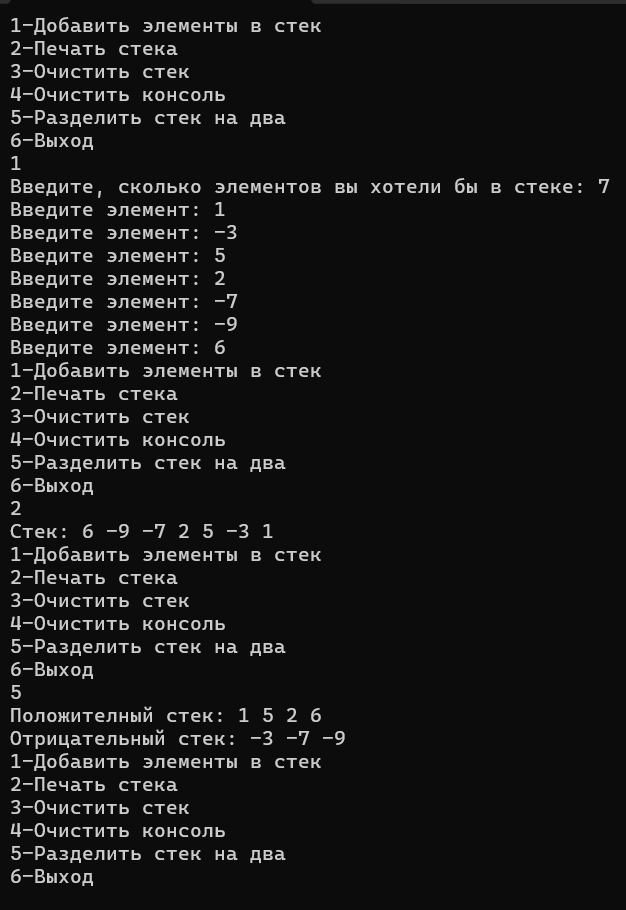
case 6:exit(0); break;

default:cout << "Неверный ввод! Попробуйте снова : "; cin >> choise; cout << endl;

}

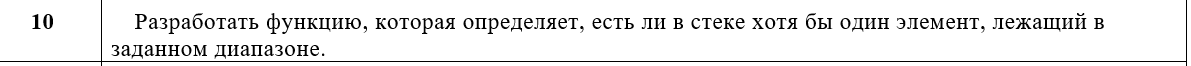
} while (true);

}



Дополнительные варианты

Вариант 10



#include <iostream>

#include <Windows.h>

#include "Header.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;

}

//

//

#include <iostream>

#include <fstream>

#include "Header.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

}

}

//

//

struct Stack

{

int data;

Stack\* next;

};

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

char pop(Stack\*& myStk);

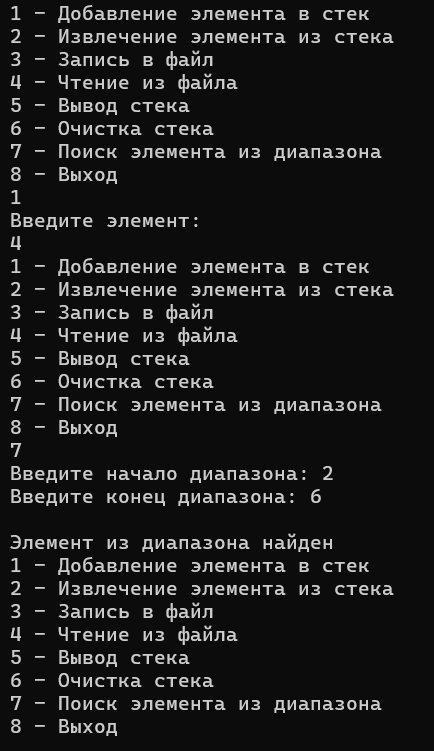
void push(int x, Stack\*& myStk);

void toFile(Stack\*& myStk);

void fromFile(Stack\*& myStk);

void clear(Stack\*& myStk);

void find(Stack\*& myStk);



Вариант 2

#include <iostream>

#include <Windows.h>

#include "Header.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;

}

//

//

#include <iostream>

#include <fstream>

#include "Header.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

}

}

//

//

struct Stack

{

int data; //информационный элемент

Stack\* next; //указатель на следующий элемент

};

void show(Stack\*& myStk);

char pop(Stack\*& myStk);

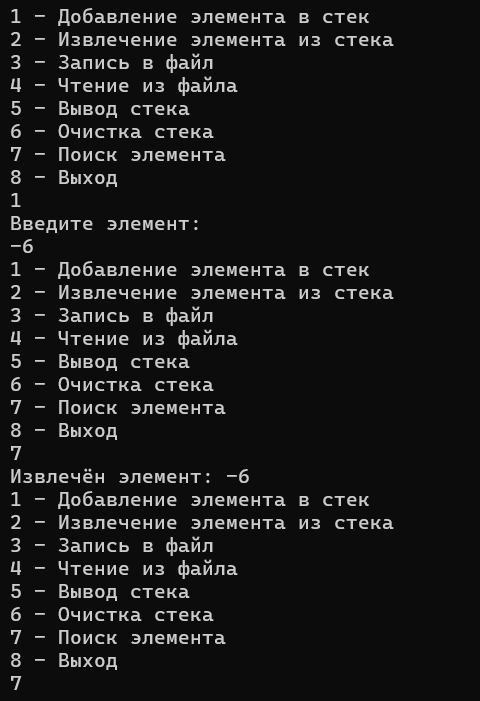
void push(int x, Stack\*& myStk);

void toFile(Stack\*& myStk);

void fromFile(Stack\*& myStk);

void clear(Stack\*& myStk);

void find(Stack\*& myStk);



Вариант 3

#include <iostream>

#include <Windows.h>

#include "Header.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;

}

//

//

#include <iostream>

#include <fstream>

#include "Header.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

}

}

//

//

struct Stack

{

int data; //информационный элемент

Stack\* next; //указатель на следующий элемент

};

void show(Stack\*& myStk);

char pop(Stack\*& myStk);

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

void toFile(Stack\*& myStk);

void fromFile(Stack\*& myStk);

void clear(Stack\*& myStk);

void find(Stack\*& myStk);

