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:

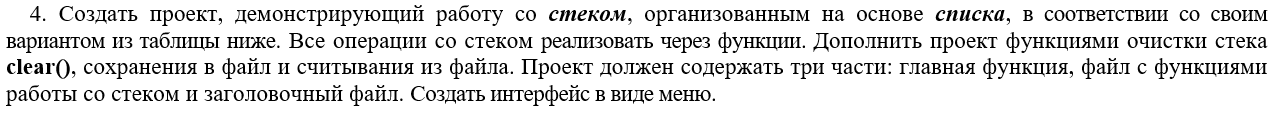
Федорович Вадим

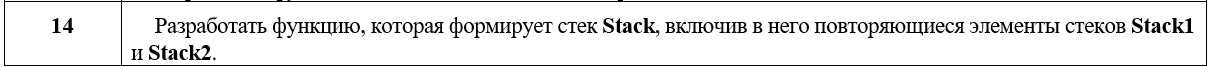
Студент 1 курса 8 группы

Преподаватель: асс. Андронова М.В.

Минск, 2024

Основное задание:





//task\_1.cpp

#include <iostream>

#include "myStack.h"

using namespace std;

//Разработать функцию, которая формирует стек Stack, включив в него повторяющиеся элементы стеков Stack1 и Stack2.

int main()

{

setlocale(LC\_ALL, "Rus");

int choice; int x,y;

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

Stack\_2\* myStk\_2 = new Stack\_2; myStk\_2 = NULL;

Stack\_3\* myStk\_3 = new Stack\_3; myStk\_3 = NULL;

for (;;)

{

cout << "Выберите команду:" << endl;

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

cout << "2 - Добавление элемента в стек 2" << endl;

cout << "3 - Извлечение элемента из стека" << endl;

cout << "4 - Запись в файл" << endl;

cout << "5 - Чтение из файла" << endl;

cout << "6 - Вывод стека" << endl;

cout << "7 - 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:

cout << "Введите элемент: " << endl;

cin >> y;

push\_2(y, myStk\_2);

break;

case 3:

x = pop(myStk);

if (x != -1)

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

break;

case 4:

toFile(myStk);

break;

case 5:

fromFile(myStk);

break;

case 6:

cout << "Весь стек: " << endl;

show(myStk);

break;

case 7:

clear(myStk, myStk\_2, myStk\_3);

break;

case 0:

return 0;

break;

}

}

return 0;

}

//mySatk.cpp

#include <iostream>

#include <fstream>

#include "myStack.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

}

void push\_2(int y, Stack\_2\*& myStk\_2) //Добавление элемента y в стек

{

Stack\_2\* r = new Stack\_2;

r->data\_2 = y;

r->next = myStk\_2;

myStk\_2 = r;

}

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;

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

{

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 << pop(e) << " ";

cout << endl;

}

void clear(Stack\*& myStk, Stack\_2\*& myStk\_2, Stack\_3\*& myStk\_3) { // Сравнивание следующих элементов

Stack\* e = myStk;int i = 0, j = 0;

Stack\_2\* d = myStk\_2;

Stack\_3\* c = myStk\_3;

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

i++;

e = e->next;

}

e = myStk;

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

j++;

d = d->next;

}

d = myStk\_2;

for (int a = 0; a < i; a++) { // проверка на равность

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

if (e->data == d->data\_2) { // Запись в 3 стек

Stack\_3\* c = new Stack\_3;

c->data\_3 = e->data;

c->next = myStk\_3;

myStk\_3 = c;

}

d = d->next;

}

d = myStk\_2;

e = e->next;

}

e = myStk;

d = myStk\_2;

c = myStk\_3;

while (c != NULL) { // Вывод

cout << c->data\_3;

cout << "-->";

c = c->next;

}

cout << "--NULL";

c = myStk\_3;

}

// myStak.h struct Stack

{

int data; //информационный элемент

Stack\* next; //указатель на следующий элемент

};

struct Stack\_2

{

int data\_2;

Stack\_2\* next;

};

struct Stack\_3

{

int data\_3;

Stack\_3\* next;

};

void show(Stack\*& myStk);

char pop(Stack\*& myStk);

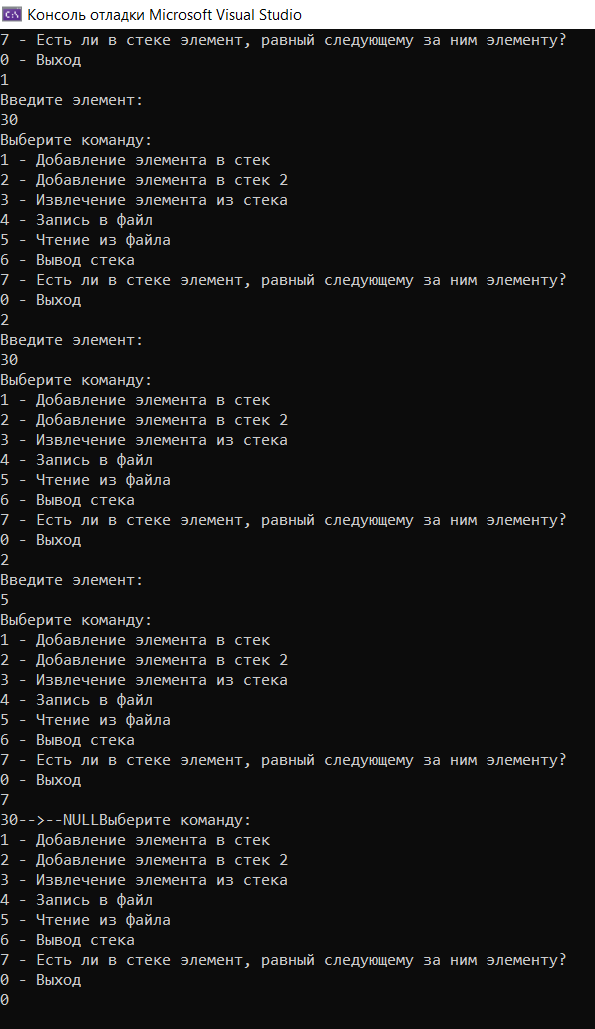
void push(int x, Stack\*& myStk);

void push\_2(int y, Stack\_2\*& myStk\_2);

void toFile(Stack\*& myStk);

void fromFile(Stack\*& myStk);

void clear(Stack\*& myStk, Stack\_2\*& myStk\_2, Stack\_3\*& myStk\_3);



Дополнительные задания:



//main.cpp

#include <iostream>

#include "fun.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 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;

}

//fun.cpp

#include <iostream>

#include <fstream>

#include "fun.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

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;

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

{

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 << 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;

delete temp;

}

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;

delete temp;

}

}

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

// Печать значений после удаления

e = myStk;

while (e != NULL) {

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

e = e->next;

}

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

}

//fun.h

struct Stack

{

int data; //информационный элемент

Stack\* next; //указатель на следующий элемент

};

void show(Stack\*& myStk);

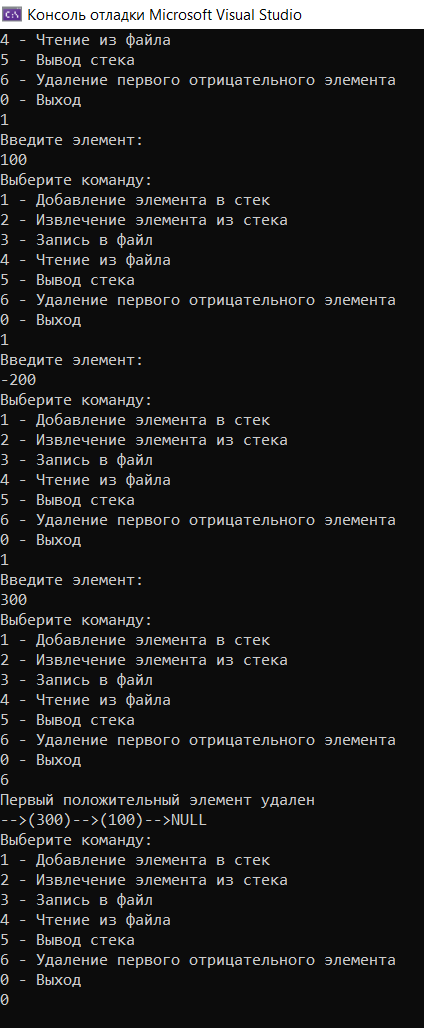
char pop(Stack\*& myStk);

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

void toFile(Stack\*& myStk);

void fromFile(Stack\*& myStk);

void clear(Stack\*& myStk);





//task\_3

#include <iostream>

#include "functions.h"

using namespace std;

//Разработать функцию, которая формирует стек Stack, включив в него повторяющиеся элементы стеков Stack1 и Stack2.

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;

}

//functions.cpp

#include <iostream>

#include <fstream>

#include "functions.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

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;

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

{

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 << 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;

delete temp;

}

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;

delete temp;

}

}

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

// Печать значений после удаления

e = myStk;

while (e != NULL) {

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

e = e->next;

}

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

}

//functions.h

struct Stack

{

int data; //информационный элемент

Stack\* next; //указатель на следующий элемент

};

void show(Stack\*& myStk);

char pop(Stack\*& myStk);

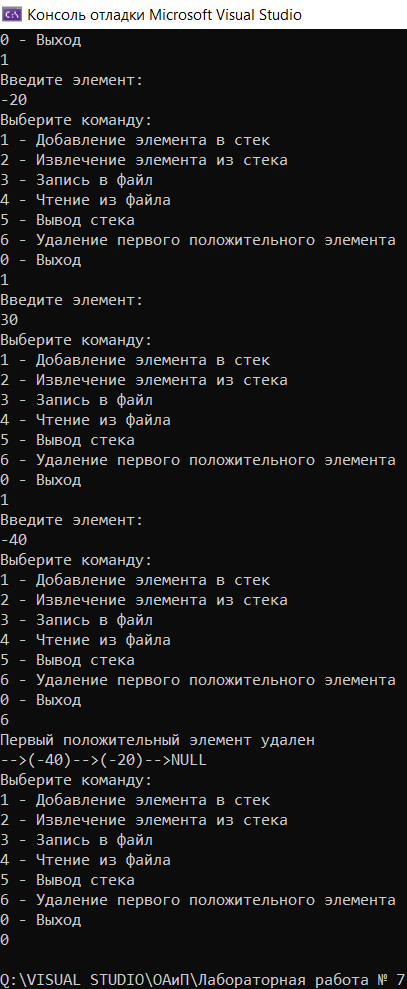
void push(int x, Stack\*& myStk);

void toFile(Stack\*& myStk);

void fromFile(Stack\*& myStk);

void clear(Stack\*& myStk);

void find\_same(Stack\*& myStk);





//task\_1.cpp

#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 - 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;

}

//MyStk.cpp

#include <iostream>

#include <fstream>

#include "myStack.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

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;

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

{

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 << pop(e) << " ";

cout << endl;

}

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

bool flag = false; int buf;

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;

}

//MyStk.cpp

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\_same(Stack\*& myStk);

