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:

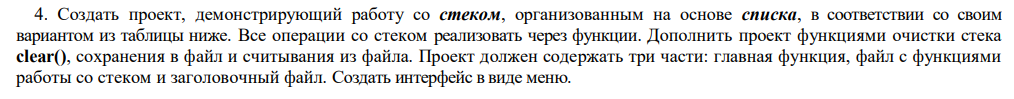
Студент 1 курса 8 группы

Лужецкий Владислав Константинович

Преподаватель: асс. Андронова М.В.

2024, Минск

Основной вариант(6):





//lab7.cpp

#include <iostream>

#include <fstream>

#include "myStack.h"

using namespace std;

void push(char 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

e->before = nullptr; //перенос вершины на предыдущий 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 << e->data << " ";

e = e->next;

}

cout << endl;

}

void clear(Stack\*& myStk)

{

while (myStk != nullptr)

{

Stack\* temp = myStk;

myStk = myStk->next;

delete temp;

}

cout << "\nСтек пуст.\n";

}

void counterEqualLeftAndRight(Stack\*& myStk)

{

int counter = 0; // Счетчик для подсчета соответствующих пар

// Проверяем, что текущий элемент и его следующие два элемента не являются нулевыми

while (myStk != nullptr && myStk->next != nullptr && myStk->next->next != nullptr)

{

if (myStk->data == myStk->next->next->data)

{

counter++; // Увеличиваем счетчик, если значения текущего элемента и элемента через один равны

}

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

}

cout << counter << endl; // Выводим значение счетчика

}

//myStack.h

#pragma once

struct Stack

{

char data; //информационный элемент

Stack\* next; //указатель на следующий элемент

Stack\* before; //указатель на предыдущий элемент

};

void show(Stack\*& myStk);

char pop(Stack\*& myStk);

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

void toFile(Stack\*& myStk);

void fromFile(Stack\*& myStk);

void clear(Stack\*& myStk);

void counterEqualLeftAndRight(Stack\*& myStk);

//myStack.cpp

#include <iostream>

#include "myStack.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "Rus");

int choice; char x;

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

myStk = NULL; //инициализация первого элемента

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 - Очистка стека" << endl;

cout << "7 - Подсчёт количества элементов стека, у которых равные соседи" << 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 7:

cout << "Количество элементов стека, у которых равные соседи: ";

counterEqualLeftAndRight(myStk);

break;

case 0:

return 0;

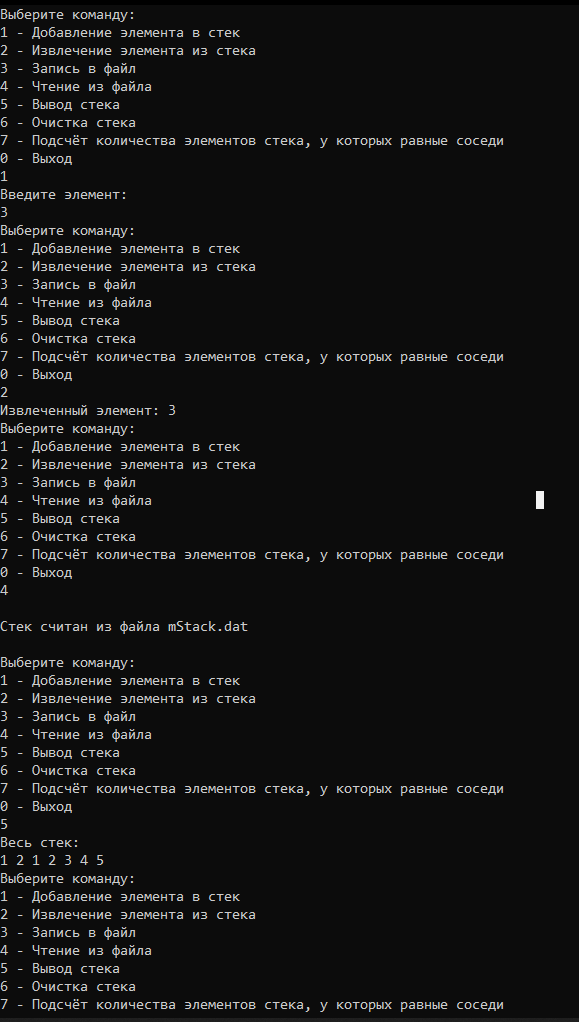
break;

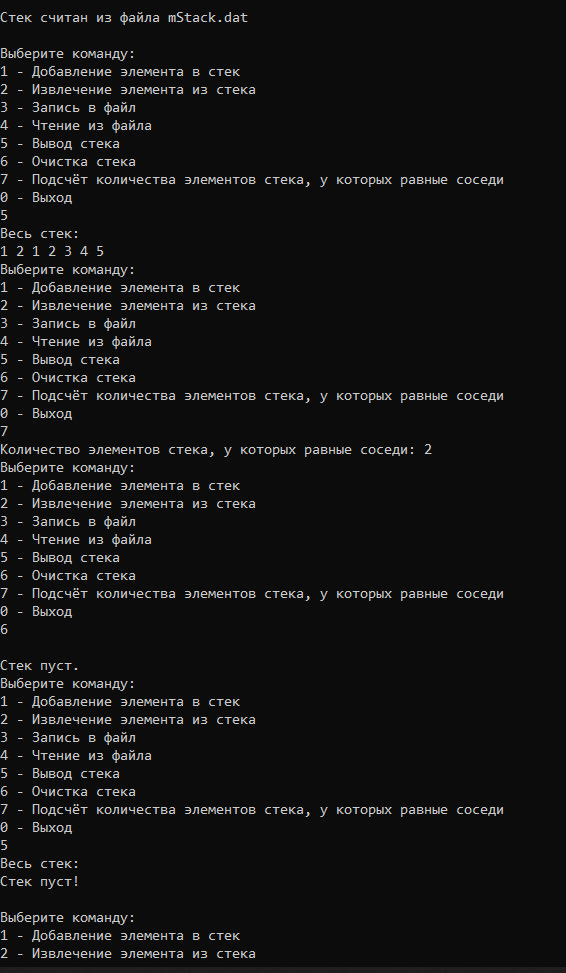
}

}

return 0;

}





Доп 1:

//все допы отличаются только функцией(я буду скидывать только эту функцию)



void counterEqualThatAndNext(Stack\*& myStk)

{

int counter = 0; // Счетчик для подсчета пар

// Проверяем, что текущий элемент и следующий элемент не являются нулевыми

while (myStk != nullptr && myStk->next != nullptr)

{

if (myStk->data == myStk->next->data)

{

counter++; // Увеличиваем счетчик, если значения текущего и следующего элементов равны

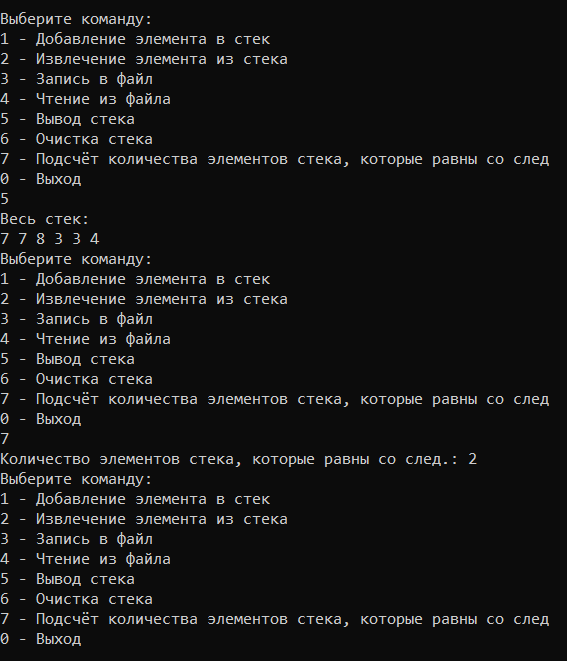
}

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

}

cout << counter << endl; // Выводим значение счетчика

}



Доп 2:



void DeleteFirstNegativeElement(Stack\*& myStk)

{

Stack\* prev = nullptr; // Указатель на предыдущий элемент

Stack\* current = myStk; // Указатель на текущий элемент

// Проходим по элементам стека

while (current != nullptr)

{

if (current->data < 0)

{

cout << "Удаленный элемент: " << current->data << endl;

if (prev == nullptr)

{

// Если предыдущий элемент равен nullptr, значит текущий элемент - первый в стеке

// Обновляем указатель myStk на следующий элемент

myStk = current->next;

}

else

{

// Иначе, текущий элемент не является первым в стеке

// Обновляем ссылку предыдущего элемента на следующий элемент, пропуская текущий элемент

prev->next = current->next;

}

delete current; // Удаляем текущий элемент

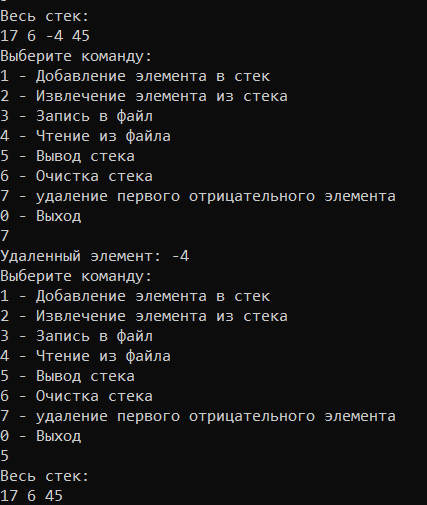
return;

}

prev = current; // Обновляем указатель на предыдущий элемент

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

}

}

Доп 3:



void DeleteFirstPositiveElement(Stack\*& myStk)

{

Stack\* prev = nullptr; // Указатель на предыдущий элемент

Stack\* current = myStk; // Указатель на текущий элемент

// Проходим по элементам стека

while (current != nullptr)

{

if (current->data > 0)

{

cout << "Удаленный элемент: " << current->data << endl;

if (prev == nullptr)

{

// Если предыдущий элемент равен nullptr, значит текущий элемент - первый в стеке

// Обновляем указатель myStk на следующий элемент

myStk = current->next;

}

else

{

// Иначе, текущий элемент не является первым в стеке

// Обновляем ссылку предыдущего элемента на следующий элемент, пропуская текущий элемент

prev->next = current->next;

}

delete current; // Удаляем текущий элемент

return;

}

prev = current; // Обновляем указатель на предыдущий элемент

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

}

}

