Лабораторная работа 7

По дисциплине «Основы алгоритмизации и программирования»

На тему «Полустатические структуры данных: стеки»

***Задание#1***



*Macros.h*

#include <iostream>

#include <vector>

#include <ctime>

#include <cmath>

#include <iomanip>

#include <stack>

#include <map>

#include <cstring>

#include <string>

#include <chrono>

#include <fstream>

#include <Windows.h>

#define vi cout <<

#define vv cin >>

#define nl cout << "\n";

#define nw cout << "\t";

#define SCOCP1251 SetConsoleOutputCP(1251);

#define SCCP1251 SetConsoleCP(1251);

#define ret return

#define pause system("pause");

using namespace :: std;

*Stackfunctions.h*

#pragma once

#include "Macros.h"

struct Stack {

int data;

Stack\* next;

};

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

int pop(Stack\*& myStk);

void clear(Stack\*& myStk);

void saveToFile(Stack\*& myStk, const char\* fileName);

void readFromFile(Stack\*& myStk, const char\* fileName);

void display(Stack\* myStk);

int removeFirstNegative(Stack\*& myStk);

*Stackfunctions.cpp*

#include "stackFunctions.h"

#include "Macros.h"

// Функция для добавления нового элемента в начало стека

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

// Создание нового узла и заполнение его данными

Stack\* e = new Stack;

e->data = x;

e->next = myStk; // Установлен указатель на следующий элемент (переданный в функцию узел стека)

myStk = e; // Установлен указатель на новый элемент в начале стека

}

// Функция для удаления (извлечения) элемента из стека

int pop(Stack\*& myStk) {

if (myStk == nullptr) {

vi "Стек пуст!"; nl

ret - 1;

}

else {

int x = myStk->data;

Stack\* temp = myStk;

myStk = myStk->next; // Перенос вершины на следующий элемент

delete temp; // Освобождение памяти для удаленного элемента

ret x;

}

}

// Функция для очистки стека

void clear(Stack\*& myStk) {

while (myStk) {

pop(myStk); // Для каждого элемента вызывает функцию, которая удалит его

}

}

// Функция для сохранения стека в файл

void saveToFile(Stack\*& myStk, const char\* fileName) {

ofstream file(fileName);

if (!file.is\_open()) {

vi "Ошибка открытия файла"; nl

ret;

}

Stack\* temp = myStk;

while (temp) {

// Запись данных стека в файл через пробел

file << temp->data << " ";

temp = temp->next;

}

file.close();

vi "Стек записан в файл " << fileName; nl

}

// Функция для считывания стека из файла

void readFromFile(Stack\*& myStk, const char\* fileName) {

// Создание временного стека для прочитанных данных

Stack\* tempStack = nullptr;

ifstream file(fileName);

if (!file.is\_open()) {

vi "Ошибка открытия файла"; nl

ret;

}

int value;

// Считывание данных из файла и добавление их в временный стек

while (file >> value) {

push(value, tempStack);

}

file.close();

// Добавление прочитанных данных в основной стек в обратном порядке (что сохранит первоначальный порядок чисел в файле)

while (tempStack) {

push(pop(tempStack), myStk);

}

vi "Стек считан из файла " << fileName; nl

}

// Функция для вывода содержимого стека

void display(Stack\* myStk) {

if (myStk == nullptr) {

vi "Стек пуст!"; nl

}

else {

vi "Стек: ";

while (myStk) {

vi myStk->data << " "; // Вывод значения текущего элемента

myStk = myStk->next; // Переход к следующему элементу

}

nl

}

}

// Функция для удаления первого отрицательного элемента из стека

int removeFirstNegative(Stack\*& myStk) {

// Инициализация указателей для прохода по стеку

Stack\* current = myStk;

Stack\* prev = nullptr;

// Переменная для хранения значения удаленного элемента (по умолчанию 1, если не найден)

int num = 1;

// Перебор элементов стека

while (current != nullptr) {

// Проверка, является ли текущий элемент отрицательным

if (current->data < 0) {

// Удаление первого отрицательного элемента

// Если текущий элемент - первый в стеке

if (prev == nullptr) {

myStk = current->next; // Перемещение указателя на начало стека на следующий элемент

}

else {

prev->next = current->next; // Пропуск текущего элемента при связывании предыдущего и следующего

}

num = current->data; // Сохранение значения удаленного элемента

delete current; // Освобождение памяти элемента, который удаляется.

ret num; // Возвращение значения удаленного элемента

}

// Переход к следующему элементу

prev = current;

current = current->next;

}

// Если отрицательный элемент не найден, возвращается значение по умолчанию (-1)

ret num;

}

*Task#1.cpp*

#include "Macros.h"

#include "stackFunctions.h"

int main() {

SCOCP1251;

SCCP1251;

Stack\* myStack = nullptr;

int choice;

do {

vi "----------------------"; nl

vi "1. Добавить элемент"; nl

vi "2. Удалить первый отрицательный элемент"; nl

vi "3. Очистить стек"; nl

vi "4. Сохранить стек в файл"; nl

vi "5. Прочитать стек из файла"; nl

vi "6. Вывести стек"; nl

vi "0. Выход"; nl

vi "----------------------"; nl

vi "Выберите действие: ";

vv choice;

switch (choice) {

case 1:

// Добавление элемента в стек

int value;

vi "Введите значение: ";

vv value;

push(value, myStack);

break;

case 2:

// Удаление первого отрицательного элемента

{

int removed = removeFirstNegative(myStack);

if (removed == -1) {

vi "Стек не содержит отрицательных элементов"; nl

}

else {

vi "Первый отрицательный элемент удален: " << removed; nl

}

}

break;

case 3:

// Очистка стека

clear(myStack);

vi "Стек очищен"; nl

break;

case 4:

// Сохранение стека в файл

saveToFile(myStack, "stack.dat");

break;

case 5:

// Чтение стека из файла

readFromFile(myStack, "stack.dat");

break;

case 6:

// Вывод стека

display(myStack);

break;

case 0:

// Завершение программы

clear(myStack); // Очистка памяти перед выходом

vi "Программа завершена"; nl

break;

default:

vi "Неправильный выбор"; nl

}

} while (choice != 0);

pause

ret 0;

}

