# Учреждение образования

# «Белорусский государственный технологический университет»

Кафедра информационных систем и технологий

Отчёт по лабораторной работе №8

Вариант 2

Выполнил:

студент 1 курса 2 группы Самцевич Алексей

Минск 2020

Задание №1: Создать очередь для целых чисел и функции для ввода, вывода и удаления элементов очереди. Найти количество элементов между максимальным и минимальным элементами очереди.

Код:

Главный файл:

////

//// main.cpp

//// laba\_8.1

////

//// Created by Alex Samtsevich on 19.05.2020.

//// Copyright © 2020 Alex Samtsevich. All rights reserved.

////

#include <iostream>

#include <fstream>

#include "functions.hpp"

using namespace std;

int main() {

list \*first = NULL;

int choice = 0;

while (choice != 7) {

menu();

cout << "Выберите нужный пункт: ";

cin >> choice;

switch (choice) {

case 1:

insert(first);

printList(first);

break;

case 2:

if (!IsEmpty(first)) {

if (del(first)) {

printList(first);

}

else {

cout << "Число не найдено" << endl;

}

}

else {

cout << "Список пуст" << endl;

}

break;

case 3:

search(first);

break;

case 4:

avarage(first);

break;

case 5:

toFile(first);

break;

case 6:

fromFile(first);

break;

case 7:

minMax(first);

break;

default:

cout << "Неправильный выбор" << endl;

menu();

break;

}

}

cout << "Конец" <<endl;

return 0;

}

Заголовочный файл:

//

// functions.hpp

// laba\_8.1

//

// Created by Alex Samtsevich on 19.05.2020.

// Copyright © 2020 Alex Samtsevich. All rights reserved.

//

#ifndef functions\_hpp

#define functions\_hpp

#include <stdio.h>

struct list {

float number;

list \*next;

};

void insert(list \*&);

float del(list \*&);

void search(list \*p);

int IsEmpty(list \*);

void printList(list \*);

void menu(void);

void avarage(list \*);

void toFile(list \*&p);

void fromFile(list \*&p);

void insertFromFile(list \*&p, float value);

void minMax(list \*p);

#endif /\* functions\_hpp \*/

Файл с функциями:

//

// functions.cpp

// laba\_8.1

//

// Created by Alex Samtsevich on 19.05.2020.

// Copyright © 2020 Alex Samtsevich. All rights reserved.

//

#include "functions.hpp"

#include <iostream>

#include <fstream>

using namespace std;

void menu(void) {

cout << "\n 1 - Ввод числа" << endl;

cout << " 2 - Удаление числа" << endl;

cout << " 3 - Поиск позиции числа" << endl;

cout << " 4 - Вычисление среднего значения" << endl;

cout << " 5 - Записать в файл" << endl;

cout << " 6 - Прочитать из файла" << endl;

cout << " 7 - Расстояние между максимальным и минимальным элементами" << endl;

cout << " 8 - Выход" << endl;

}

void insert(list \*&p) {

float value;

cout << "Введите свое число: " << endl;

cin >> value;

list \*newP = new list;

if (newP != NULL) {

newP->number = value;

newP->next = p;

p = newP;

}

else {

cout << "Операция добавления не выполнена" << endl;

}

}

void insertFromFile(list \*&p, float value) {

list \*newP = new list;

if (newP != NULL) {

newP->number = value;

newP->next = p;

p = newP;

}

else {

cout << "Операция добавления не выполнена" << endl;

}

}

float del(list \*&p) {

float value;

cout << "Введите удаляемое число: " << endl;

cin >> value;

list \*previous, \*current, \*temp;

if (value == p->number) {

temp = p;

p = p->next;

delete temp;

return value;

}

else {

previous = p;

current = p->next;

while (current!= NULL && current->number != value) {

previous = current;

current = current->next;

}

if (current!= NULL) {

temp = current;

previous->next = current->next;

free(temp);

return value;

}

}

return 0;

}

int IsEmpty(list \*p) {

return p == NULL;

}

void printList(list \*p) {

if (p == NULL) {

cout<<"Список пуст"<<endl;

}

else {

cout<<"Список:"<<endl;

while (p!= NULL) {

cout << " -> " << p->number;

p = p->next;

}

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

}

}

void search(list \*p) {

int i = 0;

float searchNumber;

cout << "Введите искомое число: ";

cin >> searchNumber;

while (p != NULL) {

if (p->number == searchNumber) {

cout << "Искомое число имеет позицию: " << i << endl;

break;

}

else {

p = p->next;

i++;

}

}

}

void avarage(list \*p) {

float sm = 0, average;

int count = 0;

if (p == NULL) {

cout << "Список пуст" << endl;

}

else {

while (p!= NULL) {

if (p->number > 0) {

sm = sm + (p->number);

count++;

}

p = p->next;

}

average = sm / count;

cout << "Среднее значение = " << average << endl;

}

}

void toFile(list \*&p) {

list \*temp = p;

list buf;

ofstream frm("mList.dat");

if (frm.fail()) {

cout << "\n Ошибка открытия файла";

exit(1);

}

while (temp) {

buf = \*temp;

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

temp = temp->next;

}

frm.close();

cout << "Список записан в файл mList.dat\n";

}

void fromFile(list \*&p) {

list buf;

list \*first = nullptr;

ifstream frm("mList.dat");

if (frm.fail()) {

cout << "\n Ошибка открытия файла";

exit(1);

}

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

while (!frm.eof()) {

insertFromFile(first, buf.number);

cout << " -> " << buf.number;

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

}

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

frm.close();

p = first;

cout << "Список считан из файла mList.dat\n";

}

void minMax(list \*p) {

float numberNow;

int positionNow = 0;

float maxNumber = -100;

int maxPosition = 0;

float minNumber = 100;

int minPosition = 0;

int avaragePosition;

if (p == NULL) {

cout<<"Список пуст"<<endl;

}

else {

while (p != NULL) {

numberNow = p->number;

if(numberNow > maxNumber) {

maxNumber = numberNow;

maxPosition = positionNow;

}

if(numberNow < minNumber) {

minNumber = numberNow;

minPosition = positionNow;

}

positionNow += 1;

p = p->next;

}

avaragePosition = abs(maxPosition - minPosition);

cout << "Расстояние между максимальным и минимальным элементами: " << avaragePosition << endl;

}

}

Скриншот:

