Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа 9

По дисциплине «Основы алгоритмизации и программирования»

На тему «Двусвязные списки»

Выполнил:

Студент 1 курса 6 группы

Романов Игорь Вячеславович

Преподаватель: асс. Андронова М.В.

2024, Минск

3-ий доп к лабораторной работе

Вариант 16



#include <iostream>

#include <fstream>

#include <Windows.h>

using namespace std;

const unsigned int NAME\_SIZE = 30;

const unsigned int CITY\_SIZE = 20;

//Структура, содержащая адрес людей

struct Address {

char name[NAME\_SIZE]; //имя человека

char city[CITY\_SIZE]; //родной город

Address\* next; //следующий человек в списке

Address\* prev; //предыдущий человек в списке

};

//Меню программы

int menu() {

short choice;

cout << endl;

cout << "1. Ввод имени" << endl;

cout << "2. Удаление по имени" << endl;

cout << "3. Подсчет элементов списка с заданным значением" << endl;

cout << "4. Вывод на экран" << endl;

cout << "5. Поиск" << endl;

cout << "6. Выход" << endl;

cout << endl;

do {

cout << "Ваш выбор: ";

cin >> choice;

} while (choice < 0 || choice > 6);

return choice;

}

//Добавление элемента в конец списка

void insertElement(Address\* e, Address\*\* phead, Address\*\* plast) {

Address\* p = \*plast; //Временный указатель на конец списка

if (\*plast == NULL) { //Если список пуст

e->next = NULL;

e->prev = NULL;

\*plast = e; //Устанавливаем начало списка на добавленный элемент

\*phead = e; //Устанавливаем конец списка на добавленный элемент

}

else { //Если в списке есть элементы

p->next = e; //Указываем, что следующий элемент в существующем спике - это добавленный элемент

e->next = NULL; //Следюущий нулевой

e->prev = p; //Предыдущий уже находившийся в списке

\*plast = e; //Конец список перенаправляем на добавленный элемент

}

}

// Создание элемента и ввод его значений с клавиатуры

Address\* setElement() {

Address\* temp = new Address(); //

if (!temp) {

cerr << "Ошибка выделения памяти памяти";

return NULL;

}

cout << "Введите имя: ";

cin.ignore(cin.rdbuf()->in\_avail()); //очистка буфера для корректного ввода

cin.clear();

cin.getline(temp->name, NAME\_SIZE - 1, '\n');

cout << "Введите город: ";

cin.ignore(cin.rdbuf()->in\_avail());

cin.clear();

cin.getline(temp->city, CITY\_SIZE - 1, '\n');

temp->next = NULL;

temp->prev = NULL;

return temp;

}

//Вывод списка на экран

void outputList(Address\*\* phead, Address\*\* plast) {

Address\* t = \*phead; //Указатель на начало списка

while (t) { //Пока указатель не NULL, выводим список

cout << t->name << ' ' << t->city << endl;

t = t->next;

}

cout << "" << endl;

}

// Поиск имени в списке

void findName(char name[NAME\_SIZE], Address\*\* phead) {

Address\* t = \*phead; //Указатель на начало списка

while (t) { //Поиск элемента

if (!strcmp(name, t->name)) { //Если элемент найден, выходим из поиска

break;

}

t = t->next;

}

if (!t) {

cerr << "Имя не найдено" << endl;

}

else {

cout << t->name << ' ' << t->city << endl;

}

}

// Удаление имени

void deleteByName(char name[NAME\_SIZE], Address\*\* phead, Address\*\* plast) {

struct Address\* t = \*phead; //Указатель на начало списка

while (t) {

if (!strcmp(name, t->name)) { //Если элемент найден, выходим из поиска

break;

}

t = t->next;

}

if (!t) {

cerr << "Имя не найдено" << endl;

}

else {

if (\*phead == t) { //Если элемент находится в начале списке

\*phead = t->next;

if (\*phead) {

(\*phead)->prev = NULL;

}

else {

\*plast = NULL;

}

}

else { //Если элемент где-то не в начале

t->prev->next = t->next;

if (t != \*plast) {

t->next->prev = t->prev;

}

else {

\*plast = t->prev;

}

}

delete t;

cout << "Элемент удален" << endl;

}

}

//Подсчет числа элементов списка с заданным значением

int countX(char city[CITY\_SIZE], Address\*\* phead, Address\*\* plast) {

struct Address\* t = \*phead; //Указатель на начало списка

int counter = 0; //счетчик повторяющихся элементов

while (t) {

if (!strcmp(city, t->city)) { //Если элемент найден, увеличиваем счетчик

counter++;

}

t = t->next;

}

return counter;

}

//главная функция

int main() {

Address\* head = NULL; //создание указателя на начало списка

Address\* last = NULL; //создание указателя на конец списка

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

while (true) {

switch (menu()) {

case 1: {

insertElement(setElement(), &head, &last);

break;

}

case 2: {

char dname[NAME\_SIZE];

cout << "Введите имя: ";

cin.ignore(cin.rdbuf()->in\_avail());

cin.sync();

cin.getline(dname, NAME\_SIZE - 1, '\n');

deleteByName(dname, &head, &last);

break;

}

case 3: {

char dcity[CITY\_SIZE];

cout << "Введите город: ";

cin.ignore(cin.rdbuf()->in\_avail());

cin.sync();

cin.getline(dcity, CITY\_SIZE - 1, '\n');

cout << "Найдено совпадений: " << countX(dcity, &head, &last);

break;

}

case 4: {

outputList(&head, &last);

break;

}

case 5: {

char fname[NAME\_SIZE];

cout << "Введите имя: ";

cin.getline(fname, NAME\_SIZE - 1, '\n');

cin.ignore(cin.rdbuf()->in\_avail());

cin.sync();

findName(fname, &head);

break;

}

case 6: {

exit(0);

}

default: {

exit(1);

}

}

}

return 0;

}

