Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа 9

По дисциплине «Основы алгоритмизации и программирования»

На тему «Двухсвязные списки»

 Выполнила:

Студентка1 курса 6 группы

Альшевская Алина Михайловна

Преподаватель: асс. Андронова М.В.

2023, Минск

Вариант 1



#include <fstream>

#include <iostream>

using namespace std;

const unsigned int NAME\_SIZE = 30;

const unsigned int CITY\_SIZE = 20;

struct Address

{

char name[NAME\_SIZE];

char city[CITY\_SIZE];

Address\* next;

Address\* prev;

};

//-----------------------------------------------------------

int menu(void)

{

char s[80]; int c;

cout << endl;

cout << "1. Ввод имени" << endl;

cout << "2. Удаление имени" << endl;

cout << "3. Вывод на экран" << endl;

cout << "4. Поиск" << endl;

cout << "5. Удаление дубликатов" << endl;

cout << "6. Выход" << endl;

cout << endl;

do {

cout << "Ваш выбор: ";

cin.sync();

gets\_s(s);

cout << endl;

c = atoi(s);

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

return c;

}

//-----------------------------------------------------------

void insert(Address\* e, Address\*\* phead, Address\*\* plast) //Добавление в конец списка

{

Address\* p = \*plast;

if (\*plast == NULL)

{

e->next = NULL;

e->prev = NULL;

\*plast = e;

\*phead = e;

return;

}

else

{

p->next = e;

e->next = NULL;

e->prev = p;

\*plast = e;

}

}

//-----------------------------------------------------------

Address\* setElement() // Создание элемента и ввод его значений с клавиатуры

{

Address\* temp = new Address();

if (!temp)

{

cerr << "Ошибка выделения памяти памяти";

return NULL;

}

cout << "Введите имя: ";

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

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

cin.clear();

cout << "Введите город: ";

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

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

cin.clear();

temp->next = NULL;

temp->prev = NULL;

return temp;

}

//-----------------------------------------------------------

void outputList(Address\*\* phead, Address\*\* plast) //Вывод списка на экран

{

Address\* t = \*phead;

while (t)

{

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

t = t->next;

}

cout << "" << endl;

}

//-----------------------------------------------------------

void find(char name[NAME\_SIZE], Address\*\* phead) // Поиск имени в списке

{

Address\* t = \*phead;

while (t)

{

if (!strcmp(name, t->name)) break;

t = t->next;

}

if (!t)

cerr << "Имя не найдено" << endl;

else

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

}

//-----------------------------------------------------------

void delet(char name[NAME\_SIZE], Address\*\* phead, Address\*\* plast) // Удаление имени

{

struct Address\* t = \*phead;

while (t)

{

if (!strcmp(name, t->name)) break;

t = t->next;

}

if (!t)

cerr << "Имя не найдено" << endl;

else

{

if (\*phead == t)

{

\*phead = t->next;

if (\*phead)

(\*phead)->prev = NULL;

else

\*plast = NULL;

}

else

{

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

if (t != \*plast)

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

else

\*plast = t->prev;

}

delete t;

cout << "Элемент удален" << endl;

}

}

// Функция для удаления одного элемента

void deleteNode(Address\*\* head, Address\* del) {

if (\*head == NULL || del == NULL)

return;

if (\*head == del)

\*head = del->next;

if (del->next != NULL)

del->next->prev = del->prev;

if (del->prev != NULL)

del->prev->next = del->next;

delete del;

}

// Функция для удаления всех дубликатов

void deleteDouble(Address\*\* head) {

if ((\*head) == nullptr) {

cout << "В списке нет элементов. Удалять нечего." << endl;

return;

}

Address\* current = \*head;

Address\* next;

while (current != nullptr) {

next = current->next;

while (next != nullptr) {

if ((current->city == next->city) && (current->name == next->name)) {

Address\* duplicate = next;

next = next->next;

deleteNode(head, duplicate);

}

else {

next = next->next;

}

}

current = current->next;

}

\*head = current; // Установка оригинального указателя и списка, как в функции

cout << "Все дубликаты удалены!" << endl;

}

// Главная функция

int main(void) {

Address\* head = nullptr;

Address\* last = nullptr;

setlocale(LC\_CTYPE, "Rus"); // Установка локали

while (true) {

switch (menu()) {

case 1: insert(setElement(), &head, &last);

break;

case 2: {

char dname[NAME\_SIZE];

cout << "Введите имя: ";

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

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

cin.sync();

delet(dname, &head, &last);

}

break;

case 3: outputList(&head, &last);

break;

case 4: {

char fname[NAME\_SIZE];

cout << "Введите имя: ";

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

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

cin.sync();

find(fname, &head);

}

break;

case 5:

deleteDouble(&head);

outputList(&head, &last);

break;

case 6: exit(0); break;

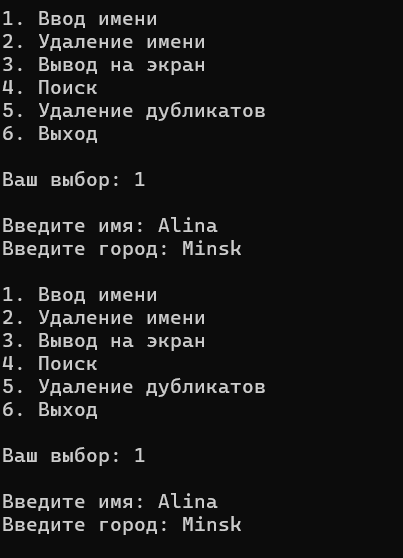
default: exit(1);

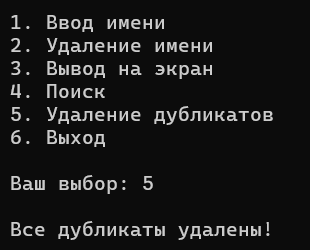
}

}

return 0;

}





Дополнительные задания

Задание 1

В [приложении 2](#prilogenie2) приведен проект, в котором реализован проект с использованием *двусвязного* *списка*. На основе данного проекта разработать функции, которые предлагается создать в данном приложении.

#include "List.h"

#include <iostream>

using namespace std;

bool Object::Insert(void\* data) // Вставка в начало

{

bool rc = 0;

if (Head == NULL)

{

Head = new Element(NULL, data, Head);

rc = true;

}

else

{

Head = (Head->Prev = new Element(NULL, data, Head));

rc = true;

}

return rc;

}

//-----------------------------------------------------------

Element\* Object::Search(void\* data) // Найти заданный элемент

{

Element\* rc = Head;

while ((rc != NULL) && (rc->Data != data))

rc = rc->Next;

return rc;

}

//-----------------------------------------------------------

void Object::PrintList(void(\*f)(void\*)) // Вывод

{

Element\* e = Head;

while (e != NULL)

{

f(e->Data);

e = e->GetNext();

};

}

//-----------------------------------------------------------

void Object::PrintList(void(\*f)(void\*), Element\* e)

{

f(e->Data);

}

//-----------------------------------------------------------

bool Object::Delete(Element\* e) // Удалить по ссылке

{

bool rc = 0;

if (rc = (e != NULL))

{

if (e->Next != NULL)

e->Next->Prev = e->Prev;

if (e->Prev != NULL)

e->Prev->Next = e->Next;

else

Head = e->Next;

delete e;

}

return rc;

}

//-----------------------------------------------------------

bool Object::Delete(void\* data) // Удалить по значению

{

return Delete(Search(data));

};

//-----------------------------------------------------------

Element\* Object::GetLast()

{

Element\* e = Head, \* rc = e;

while (e != NULL)

{

rc = e;

e = e->GetNext();

}

return rc;

}

int Object::CountList() {

Element\* e = Head;

int cnt = 0;

while (e != NULL)

{

cnt++;

e = e->GetNext();

}

Head = nullptr;

return cnt;

}

//-----------------------------------------------------------

bool Object::DeleteList(int size) {

Element\* e = Head;

for (int i = 0; i < size; i++) {

bool rc = 0;

if (rc = (e != NULL))

{

if (e->Next != NULL)

e->Next->Prev = e->Prev;

if (e->Prev != NULL)

e->Prev->Next = e->Next;

else

Head = e->Next;

delete e;

}

}

return true;

}

//-----------------------------------------------------------

Object Create()

{

return \*(new Object());

}

struct Person

{

string name;

string capital;

int amm\_of\_people;

double square;

string president\_surename;

Person\* next;

};

void print(void\* b) //Функция используется при выводе

{

Person\* a = (Person\*)b;

cout << a->name << " " << a->capital << a->amm\_of\_people << a->square << a->president\_surename << endl;

}

int main()

{

setlocale(LC\_ALL, "Russian");

Person a1 = { "Belarus","Minsk",9000000,10000,"Lukashenko" };

Person a2 = { "Russia","Moscow",162000000,1000000,"Putin" };

Person a3 = { "Poland","Warsaw",20000000,1000,"Duda" };

bool rc;

Object L1 = Create(); // создание списка с именем L1

L1.Insert(&a1);

L1.Insert(&a2);

L1.Insert(&a3);

L1.PrintList(print);

Element\* e = L1.Search(&a3);

Person\* aa = (Person\*)e->Data;

cout << "Найден элемент= " << aa->name << endl;

rc = L1.Delete(&a2);

if (rc == true) cout << "Удален элемент" << endl;

cout << "Список:" << endl;

L1.PrintList(print);

cout << "Количество эдементов: " << L1.CountList() << endl;

int size = L1.CountList();

L1.DeleteList(size);

cout << "Лист удален" << endl;

cout << "Лист: " << endl;

L1.PrintList(print);

return 0;

}

List.h :

#pragma once

#include <iostream>

// Структура Element представляет узел в списке

struct Element

{

Element\* Prev; // Указатель на предыдущий элемент в списке

Element\* Next; // Указатель на следующий элемент в списке

void\* Data; // Указатель на данные, которые хранит элемент

// Конструктор для Element

Element(Element\* prev, void\* data, Element\* next)

{

Prev = prev;

Data = data;

Next = next;

}

// Метод для получения следующего элемента в списке

Element\* GetNext()

{

return Next;

};

// Метод для получения предыдущего элемента в списке

Element\* GetPrev()

{

return Prev;

};

};

// Структура Object для управления списком

struct Object

{

Element\* Head; // Указатель на начало списка

// Конструктор для Object

Object()

{

Head = NULL;

};

// Метод для получения первого элемента списка

Element\* GetFirst()

{

return Head;

};

// Методы, которые будут реализованы позже

Element\* GetLast(); // Получить последний элемент списка

Element\* Search(void\* data); // Найти первый элемент по данным

bool Insert(void\* data); // Добавить элемент в начало

bool InsertEnd(void\* data); // Добавить в конец

bool Delete(Element\* e); // Удалить по адресу элемента

bool Delete(void\* data); // Удалить первый по данным

bool DeleteList(int size); // Очистить список

void PrintList(void(\*f)(void\*));

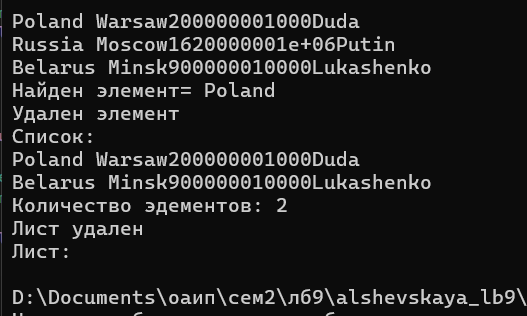
void PrintList(void(\*f)(void\*), Element\*);

int CountList();

bool DeleteDouble();

};

Object Create();



Задание 2

Каждый элемент списка студентов содержит фамилию, имя, отчество, год рождения, курс, номер группы, оценки по пяти предметам. Упорядочить студентов по курсу, причем студенты одного курса должны располагаться в алфавитном порядке. Найти средний балл каждой группы по каждому предмету. Определить самого старшего студента и самого младшего. Для каждой группы найти лучшего с точки зрения успеваемости студента.

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

// Структура DATA представляет студента

struct DATA {

string surename; // Фамилия студента

string name; // Имя студента

string patronimic; // Отчество студента

int b\_year; // Год рождения студента

int course; // Курс студента

int grup\_num; // Номер группы студента

string marks; // Оценки студента

DATA\* prev; // Указатель на предыдущего студента в списке

DATA\* next; // Указатель на следующего студента в списке

};

// Прототипы функций

void insert(int size, DATA\*\* head);

void print\_data(DATA\* head);

void sort\_course(DATA\* head);

void find\_oldest\_and\_youngest(DATA\* head);

void find\_average(DATA\* head);

void main()

{

setlocale(LC\_ALL, "Russian");

DATA\* head = nullptr;

char choise;

do {

// Меню для пользователя

cout << "1-Ввести данные" << endl;

cout << "2-Вывести данные" << endl;

cout << "3-Средний балл" << endl;

cout << "4-Самый старый и молодой" << endl;

cout << "5-Очистить" << endl;

cout << "6-Выход" << endl;

cin >> choise;

// Проверка ввода

while (!(choise >= '1' && choise <= '6')) {

cout << "Неверный ввод! Попробуйте снова: ";

cin >> choise;

}

int choise\_sw = choise - '0'; // Преобразование char в int

int size;

switch (choise\_sw) {

case 1:

cout << "Сколько студентов вы хотите добавить: ";

cin >> size;

insert(size, &head);

break;

case 2:

sort\_course(head);

print\_data(head); break;

case 3:find\_average(head); break;

case 4:find\_oldest\_and\_youngest(head); break;

case 5:system("cls"); break;

}

} while (choise != '6');

}

void insert(int size, DATA\*\* head) // Функция для вставки новой информации

{

for (int i = 0; i < size; i++) {

cout << "Студент номер " << i << " :" << endl;

DATA\* NewNode = new DATA(); // Выделение памяти для нового узла

string str\_tmp;

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

// Получение информации для узла

cout << "Введите фамилию: ";

getline(cin, str\_tmp);

NewNode->surename = str\_tmp;

cout << "Введите имя: ";

getline(cin, str\_tmp);

NewNode->name = str\_tmp;

cout << "Введите отчество: ";

getline(cin, str\_tmp);

NewNode->patronimic = str\_tmp;

int int\_tmp;

cout << "Введите год рождения: ";

cin >> int\_tmp;

NewNode->b\_year = int\_tmp;

cout << "Введите курс: ";

cin >> int\_tmp;

NewNode->course = int\_tmp;

cout << "Введите номер группы: ";

cin >> int\_tmp;

NewNode->grup\_num = int\_tmp;

char temp\_mark;

string temp\_mark\_str;

for (int i = 0; i < 5; i++) { // Цикл для ввода всех оценок

cout << "Введите оценку:";

cin >> temp\_mark;

temp\_mark\_str += temp\_mark;

if (i != 4) {

temp\_mark\_str += ';';

}

}

// Вставка узла в исходный список

NewNode->marks = temp\_mark\_str;

NewNode->next = (\*head);

NewNode->prev = nullptr;

if ((\*head) != nullptr)

(\*head)->prev = NewNode;

(\*head) = NewNode;

system("cls");

}

}

void sort\_course(DATA\* head) // Функция для сортировки списка по курсу

{

bool swaped;

DATA\* ptr;

DATA\* lptr = nullptr;

if (head == nullptr) {

return;

}

// Сортировка пузырьком

do {

swaped = false;

ptr = head;

while (ptr->next != nullptr) {

if (ptr->course > ptr->next->course) {

swap(ptr, ptr->next);

swaped = true;

}

ptr = ptr->next;

}

} while (swaped);

}

void find\_oldest\_and\_youngest(DATA\* head) // Функция для поиска самого старого студента

{

DATA\* old\_ptr = head;

int oldest = old\_ptr->b\_year;

while (old\_ptr != nullptr) {

if (old\_ptr->b\_year < oldest) {

oldest = old\_ptr->b\_year;

}

old\_ptr = old\_ptr->next;

}

cout << "Самый старый студент родился в " << oldest << " году" << endl;

DATA\* young\_ptr = head;

int youngest = young\_ptr->b\_year;

while (young\_ptr != nullptr) {

if (young\_ptr->b\_year > youngest) {

youngest = young\_ptr->b\_year;

}

young\_ptr = young\_ptr->next;

}

cout << "Самый молодой студент родился в " << youngest << " году" << endl;

}

void find\_average(DATA\* head) // Функция для поиска среднего балла

{

DATA\* ptr = head;

int size = 0;

while (ptr != nullptr) { // Вычисление размера списка

size++;

ptr = ptr->next;

}

float whole\_marks = size \* 5;

ptr = head; // Установка указателя на начало

float sum = 0;

while (ptr != nullptr) { // Суммирование всех оценок

string temp = ptr->marks;

int temp\_length = temp.length();

for (int i = 0; i < temp\_length; i++) { // Суммирование всех оценок

if (temp[i] != ';') {

sum += temp[i] - '0';

}

}

ptr = ptr->next;

}

// Вычисление среднего значения

float average = sum / whole\_marks;

cout << "Средний балл: " << average << endl;

}

void print\_data(DATA\* head) // Функция для вывода данных из списка

{

system("cls");

while (head != nullptr) {

cout << "Фамилия: " << head->surename << endl;

cout << "Имя: " << head->name << endl;

cout << "Отчество: " << head->patronimic << endl;

cout << "Год рождения: " << head->b\_year << endl;

cout << "Курс: " << head->course << endl;

cout << "Номер группы: " << head->grup\_num << endl;

cout << "Оценки: " << head->marks << endl;

head = head->next;

}

cout << endl << endl;

}

Дополнительный вариант 12

#include <iostream>

#include <fstream>

using namespace std;

const unsigned int NAME\_SIZE = 30;

const unsigned int CITY\_SIZE = 20;

struct Address

{

char name[NAME\_SIZE];

char city[CITY\_SIZE];

Address\* next;

Address\* prev;

};

//-----------------------------------------------------------

int menu(void)

{

char s[80]; int c;

cout << endl;

cout << "1. Ввод имени" << endl;

cout << "2. Удаление имени" << endl;

cout << "3. Вывод на экран" << endl;

cout << "4. Поиск" << endl;

cout << "5. Удаление первого встречающегося введённого элемента" << endl;

cout << "6. Выход" << endl;

cout << endl;

do {

cout << "Ваш выбор: ";

cin.sync();

gets\_s(s);

cout << endl;

c = atoi(s);

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

return c;

}

//-----------------------------------------------------------

void insert(Address\* e, Address\*\* phead, Address\*\* plast) //Добавление в конец списка

{

Address\* p = \*plast;

if (\*plast == NULL)

{

e->next = NULL;

e->prev = NULL;

\*plast = e;

\*phead = e;

return;

}

else

{

p->next = e;

e->next = NULL;

e->prev = p;

\*plast = e;

}

}

//-----------------------------------------------------------

Address\* setElement() // Создание элемента и ввод его значений с клавиатуры

{

Address\* temp = new Address();

if (!temp)

{

cerr << "Ошибка выделения памяти памяти";

return NULL;

}

cout << "Введите имя: ";

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

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

cin.clear();

cout << "Введите город: ";

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

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

cin.clear();

temp->next = NULL;

temp->prev = NULL;

return temp;

}

//-----------------------------------------------------------

void outputList(Address\*\* phead, Address\*\* plast) //Вывод списка на экран

{

Address\* t = \*phead;

while (t)

{

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

t = t->next;

}

cout << "" << endl;

}

//-----------------------------------------------------------

void find(char name[NAME\_SIZE], Address\*\* phead) // Поиск имени в списке

{

Address\* t = \*phead;

while (t)

{

if (!strcmp(name, t->name)) break;

t = t->next;

}

if (!t)

cerr << "Имя не найдено" << endl;

else

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

}

//-----------------------------------------------------------

void delet(char name[NAME\_SIZE], Address\*\* phead, Address\*\* plast) // Удаление имени

{

struct Address\* t = \*phead;

while (t)

{

if (!strcmp(name, t->name)) break;

t = t->next;

}

if (!t)

cerr << "Имя не найдено" << endl;

else

{

if (\*phead == t)

{

\*phead = t->next;

if (\*phead)

(\*phead)->prev = NULL;

else

\*plast = NULL;

}

else

{

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

if (t != \*plast)

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

else

\*plast = t->prev;

}

delete t;

cout << "Элемент удален" << endl;

}

}

void deleteX(int X, Address\*\* phead) // Функция для удаления первого встреченного элемента с заданным номером

{

int check\_number = 0; // Счётчик для проверки номера элемента

Address\* current = \*phead;

while (current != nullptr) {

if (check\_number == X) {

if (\*phead == current) {

\*phead = current->next;

}

if (current->next != nullptr) {

current->next->prev = current->prev; // Обновить указатель на предыдущий элемент следующего узла

}

if (current->prev != nullptr) {

current->prev->next = current->next; // Обновить указатель на следующий элемент предыдущего узла

}

// Освобождение занятой памяти

Address\* temp = current;

current = current->next;

delete temp;

cout << "Выполнено!" << endl;

return;

}

check\_number++;

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

}

cout << "Такого номера или элемента нет" << endl;

}

//-----------------------------------------------------------

int main(void)

{

Address\* head = NULL;

Address\* last = NULL;

setlocale(LC\_CTYPE, "Rus");

while (true)

{

switch (menu())

{

case 1: insert(setElement(), &head, &last);

break;

case 2: {

char dname[NAME\_SIZE];

cout << "Введите имя: ";

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

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

cin.sync();

delet(dname, &head, &last);

}

break;

case 3: outputList(&head, &last);

break;

case 4: {

char fname[NAME\_SIZE];

cout << "Введите имя: ";

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

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

cin.sync();

find(fname, &head);

}

break;

case 5:int X;

cout << "Введите номер элемента, который хотите удалить: ";

cin >> X;

deleteX(X, &head);

break;

case 6:exit(0); break;

default: exit(1);

}

}

return 0;

}

