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 8 группы

Статько Герман Вячеславович

Преподаватель: асс. Андронова М.В.

Минск, 2024

**Дополнить программу функцией в соответствии со своим вариантом из таблицы, представленной ниже.**

**Вариант 12**

****

#include <iostream>

#include <fstream>

#include <Windows.h>

using namespace std;

const unsigned int NAME\_SIZE = 30;

const unsigned int CITY\_SIZE = 20;

// Структура для элемента списка

struct Address

{

int age;

char name[NAME\_SIZE];

char city[CITY\_SIZE];

Address\* next;

Address\* prev;

};

// Функция для создания нового элемента списка

Address\* setElement();

// Функция для вставки элемента в конец списка

void insert(Address\* e, Address\*\* phead, Address\*\* plast);

// Функция для вывода списка на экран

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

// Функция для удаления элемента по имени

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

// Функция для поиска элемента по имени

void find(char name[NAME\_SIZE], Address\*\* phead);

// Функция для удаления первого встречающегося элемента с заданным возрастом

void deleteX(Address\*\* phead, Address\*\* plast, int x);

int main()

{

// Установка кодировки для корректного ввода и вывода на кириллице в консоли Windows

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_CTYPE, "rus");

Address\* head = NULL;

Address\* last = NULL;

char choice;

// Вывод меню пользователю

cout << "1 - Добавление информации" << endl

<< "2 - Удаление информации" << endl

<< "3 - Вывод всех элементов" << endl

<< "4 - Поиск" << endl

<< "5 - Удаление первого встречающегося элемента с заданным возрастом" << endl

<< "6 - Выход" << endl;

do

{

cout << "?: ";

cin >> choice;

cin.ignore();

switch (choice)

{

case '1':

// Добавление нового элемента в конец списка

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

break;

case '2':

// Удаление элемента по имени

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':

// Вывод всех элементов списка

outputList(&head, &last);

break;

case '4':

// Поиск элемента по имени

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':

// Удаление первого встречающегося элемента с заданным возрастом

int x;

cout << "Введите возраст: ";

cin >> x;

deleteX(&head, &last, x);

break;

case '6':

cout << "Выход..." << endl;

break;

default:

cout << "Неправильный выбор" << endl;

break;

}

} while (choice != '6');

}

// Функция для вставки элемента в конец списка

void insert(Address\* e, Address\*\* phead, Address\*\* plast)

{

Address\* p = \*plast;

if (\*plast == NULL)

{

// Если список пуст, новый элемент становится началом и концом списка

e->next = NULL;

e->prev = NULL;

\*plast = e;

\*phead = e;

}

else

{

// Иначе добавляем новый элемент в конец списка

p->next = e;

e->next = NULL;

e->prev = p;

\*plast = e;

}

}

// Функция для создания нового элемента списка

Address\* setElement()

{

Address\* temp = new Address();

if (!temp)

{

cerr << "Ошибка выделения памяти";

return NULL;

}

// Ввод данных для нового элемента

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();

cout << "Введите возраст: ";

cin >> temp->age;

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

cin.clear();

temp->next = NULL;

temp->prev = NULL;

return temp;

}

// Функция для вывода списка на экран

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

{

Address\* t = \*phead;

while (t)

{

// Вывод информации об элементе списка

cout << "Имя: " << t->name << endl

<< "Город: " << t->city << endl

<< "Возраст: " << t->age << endl << endl;

t = t->next;

}

cout << endl;

}

// Функция для удаления элемента по имени

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

{

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)

{

// Если удаляемый элемент - начало списка

\*phead = t->next;

if (\*phead)

(\*phead)->prev = NULL;

else

\*plast = NULL;

}

else

{

// Иначе удаляем элемент из середины или конца списка

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

if (t != \*plast)

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

else

\*plast = t->prev;

}

delete t;

cout << "Элемент удален" << endl << endl;

}

}

// Функция для поиска элемента по имени

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

{

Address\* t = \*phead;

while (t)

{

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

break;

t = t->next;

}

if (!t)

{

cerr << "Имя не найдено" << endl;

}

else

{

// Вывод найденного элемента

cout << "Имя: " << t->name << endl

<< "Город: " << t->city << endl

<< "Возраст: " << t->age << endl << endl;

}

}

// Функция для удаления первого встречающегося элемента с заданным возрастом

void deleteX(Address\*\* phead, Address\*\* plast, int x)

{

Address\* t = \*phead;

while (t)

{

if (t->age == x)

break;

t = t->next;

}

if (!t)

{

cerr << "Человек с таким возрастом не найден" << endl;

}

else

{

if (\*phead == t)

{

// Если удаляемый элемент - начало списка

\*phead = t->next;

if (\*phead)

(\*phead)->prev = NULL;

else

\*plast = NULL;

}

else

{

// Иначе удаляем элемент из середины или конца списка

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

if (t != \*plast)

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

else

\*plast = t->prev;

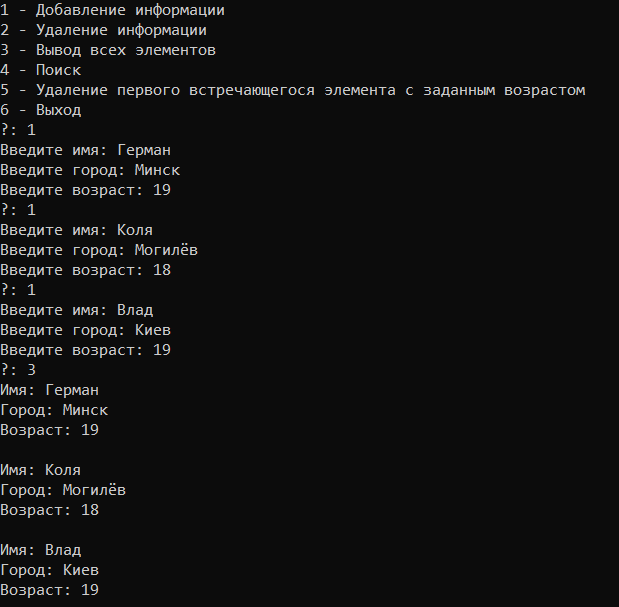
}

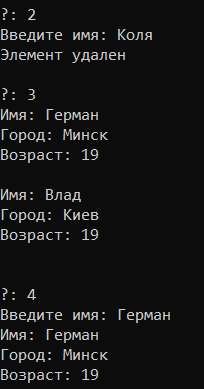
delete t;

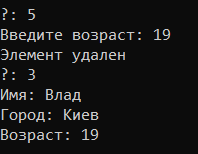
cout << "Элемент удален" << endl;

}

}

****

****

****

**Доп. Задания**

**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 для элемента списка

struct Address

{

int age;

char name[NAME\_SIZE];

char city[CITY\_SIZE];

Address\* next;

Address\* prev;

};

// Функция для создания нового элемента списка

Address\* setElement();

// Функция для вставки элемента в конец списка

void insert(Address\* e, Address\*\* phead, Address\*\* plast);

// Функция для вывода списка на экран

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

// Функция для удаления элемента по имени

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

// Функция для поиска элемента по имени

void find(char name[NAME\_SIZE], Address\*\* phead);

// Функция для удаления первого встречающегося элемента с заданным возрастом

void countX(Address\*\* phead, Address\*\* plast, int x);

int main()

{

// Установка кодировки для корректного ввода и вывода на кириллице в консоли Windows

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_CTYPE, "rus");

Address\* head = NULL;

Address\* last = NULL;

char choice;

// Вывод меню пользователю

cout << "1 - Добавление информации" << endl

<< "2 - Удаление информации" << endl

<< "3 - Вывод всех элементов" << endl

<< "4 - Поиск" << endl

<< "5 - Подсчет элементов с заданным возрастом" << endl

<< "6 - Выход" << endl;

do

{

cout << "?: ";

cin >> choice;

cin.ignore();

switch (choice)

{

case '1':

// Добавление нового элемента в конец списка

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

break;

case '2':

// Удаление элемента по имени

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':

// Вывод всех элементов списка

outputList(&head, &last);

break;

case '4':

// Поиск элемента по имени

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':

// Удаление первого встречающегося элемента с заданным возрастом

int x;

cout << "Введите возраст: ";

cin >> x;

countX(&head, &last, x);

break;

case '6':

cout << "Выход..." << endl;

break;

default:

cout << "Неправильный выбор" << endl;

break;

}

} while (choice != '6');

}

// Функция для вставки элемента в конец списка

void insert(Address\* e, Address\*\* phead, Address\*\* plast)

{

Address\* p = \*plast;

if (\*plast == NULL)

{

// Если список пуст, новый элемент становится началом и концом списка

e->next = NULL;

e->prev = NULL;

\*plast = e;

\*phead = e;

}

else

{

// Иначе добавляем новый элемент в конец списка

p->next = e;

e->next = NULL;

e->prev = p;

\*plast = e;

}

}

// Функция для создания нового элемента списка

Address\* setElement()

{

Address\* temp = new Address();

if (!temp)

{

cerr << "Ошибка выделения памяти";

return NULL;

}

// Ввод данных для нового элемента

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();

cout << "Введите возраст: ";

cin >> temp->age;

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

cin.clear();

temp->next = NULL;

temp->prev = NULL;

return temp;

}

// Функция для вывода списка на экран

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

{

Address\* t = \*phead;

while (t)

{

// Вывод информации об элементе списка

cout << "Имя: " << t->name << endl

<< "Город: " << t->city << endl

<< "Возраст: " << t->age << endl << endl;

t = t->next;

}

cout << endl;

}

// Функция для удаления элемента по имени

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

{

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)

{

// Если удаляемый элемент - начало списка

\*phead = t->next;

if (\*phead)

(\*phead)->prev = NULL;

else

\*plast = NULL;

}

else

{

// Иначе удаляем элемент из середины или конца списка

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

if (t != \*plast)

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

else

\*plast = t->prev;

}

delete t;

cout << "Элемент удален" << endl << endl;

}

}

// Функция для поиска элемента по имени

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

{

Address\* t = \*phead;

while (t)

{

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

break;

t = t->next;

}

if (!t)

{

cerr << "Имя не найдено" << endl;

}

else

{

// Вывод найденного элемента

cout << "Имя: " << t->name << endl

<< "Город: " << t->city << endl

<< "Возраст: " << t->age << endl << endl;

}

}

// Функция для подсчета всех элементов с заданным значением x

void countX(Address\*\* phead, Address\*\* plast, int x)

{

int count = 0;

Address\* t = \*phead;

while (t)

{

if (t->age == x)

count++;

t = t->next;

}

if (count == 0)

{

cout << "Людей с таким возрастом не найдено" << endl << endl;

}

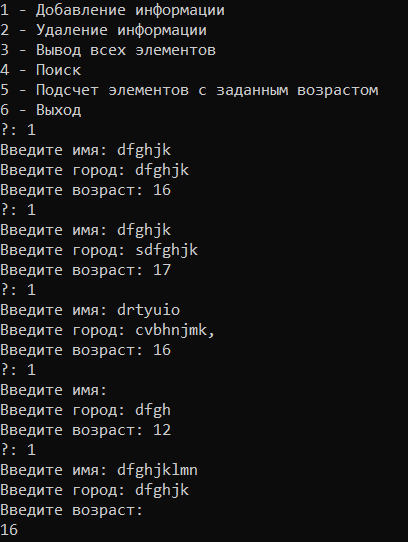
else

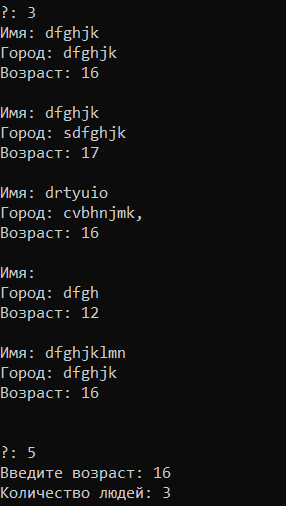
{

cout << "Количество людей: " << count << endl << endl;

}

}

****

****

****

**Main.cpp**

#include "List.h"

#include <Windows.h>

int main() {

setlocale(LC\_ALL, "Russian");

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

int choice;

Object L1 = Create();

do {

std::cout << "Выберите вариант: " << std::endl;

std::cout << "1. Добавить государство" << std::endl;

std::cout << "2. Вывести список государств" << std::endl;

std::cout << "3. Найти государство" << std::endl;

std::cout << "4. Удалить государство" << std::endl;

std::cout << "5. Очистить список государств" << std::endl;

std::cout << "6. Выйти" << std::endl;

std::cin >> choice;

switch (choice) {

case 1: {

char Name[50];

char Capital[50];

int Population;

double Area;

char PresidentLastName[50];

std::cout << "Название: ";

std::cin >> Name;

std::cout << "Столица: ";

std::cin >> Capital;

std::cout << "Население: ";

std::cin >> Population;

std::cout << "Площадь: ";

std::cin >> Area;

std::cout << "Фамилия президента: ";

std::cin >> PresidentLastName;

L1.Insert(Name, Capital, Population, Area, PresidentLastName);

break;

}

case 2: {

L1.PrintList();

break;

}

case 3: {

char name[50];

std::cout << "Введите название государства: ";

std::cin >> name;

Element\* e = L1.Search(name);

if (e == nullptr) {

std::cout << "Государство не найдено" << std::endl;

}

else {

std::cout << "Найдено государство:" << std::endl;

std::cout << "Название: " << e->Data.Name << std::endl;

std::cout << "Столица: " << e->Data.Capital << std::endl;

std::cout << "Население: " << e->Data.Population << std::endl;

std::cout << "Площадь: " << e->Data.Area << " км^2" << std::endl;

std::cout << "Президент: " << e->Data.PresidentLastName << std::endl;

}

break;

}

case 4: {

char name[50];

std::cout << "Введите название государства для удаления: ";

std::cin >> name;

bool rc = L1.Delete(name);

if (rc) {

std::cout << "Государство удалено" << std::endl;

}

else {

std::cout << "Государство не найдено или не удалено" << std::endl;

}

break;

}

case 5: {

if (L1.DeleteList()) {

std::cout << "Список государств очищен" << std::endl;

}

else {

std::cout << "Ошибка при очистке списка" << std::endl;

}

break;

}

default:

break;

}

} while (choice != 6);

return 0;

}

**List.h**

#pragma once

#include <iostream>

#include <cstring>

// Структура для хранения информации о государстве

struct Country {

char Name[50];

char Capital[50];

int Population;

double Area;

char PresidentLastName[50];

};

// Структура элемента списка государств

struct Element {

Element\* Prev;

Element\* Next;

Country Data;

// Конструктор элемента списка

Element(Element\* prev, const char\* name, const char\* capital, int population, double area, const char\* presidentLastName, Element\* next);

// Методы получения указателей на предыдущий и следующий элементы

Element\* GetNext();

Element\* GetPrev();

};

// Структура управления списком государств

struct Object {

Element\* Head;

// Конструктор объекта управления списком

Object();

// Деструктор объекта управления списком

~Object();

// Методы получения указателей на первый и последний элементы списка

Element\* GetFirst();

Element\* GetLast();

// Методы для работы с элементами списка

bool Insert(const char\* name, const char\* capital, int population, double area, const char\* presidentLastName);

bool Delete(Element\* e);

Element\* Search(const char\* name);

bool Delete(const char\* name);

bool DeleteList();

// Метод для вывода списка государств

void PrintList();

};

// Функция создания объекта управления списком государств

Object Create();

**List.cpp**

#include "List.h"

// Реализация конструктора элемента списка

Element::Element(Element\* prev, const char\* name, const char\* capital, int population, double area, const char\* presidentLastName, Element\* next)

: Prev(prev), Next(next) {

strcpy\_s(Data.Name, name);

strcpy\_s(Data.Capital, capital);

Data.Population = population;

Data.Area = area;

strcpy\_s(Data.PresidentLastName, presidentLastName);

}

// Метод получения указателя на следующий элемент

Element\* Element::GetNext() {

return Next;

}

// Метод получения указателя на предыдущий элемент

Element\* Element::GetPrev() {

return Prev;

}

// Конструктор объекта управления списком

Object::Object() {

Head = nullptr;

}

// Деструктор объекта управления списком

Object::~Object() {

DeleteList();

}

// Метод получения указателя на первый элемент списка

Element\* Object::GetFirst() {

return Head;

}

// Метод получения указателя на последний элемент списка

Element\* Object::GetLast() {

Element\* e = Head;

Element\* rc = e;

while (e != nullptr) {

rc = e;

e = e->GetNext();

}

return rc;

}

// Метод вставки нового элемента в начало списка

bool Object::Insert(const char\* name, const char\* capital, int population, double area, const char\* presidentLastName) {

if (Head == nullptr) {

Head = new Element(nullptr, name, capital, population, area, presidentLastName, nullptr);

}

else {

Head = (Head->Prev = new Element(nullptr, name, capital, population, area, presidentLastName, Head));

}

return true;

}

// Метод поиска элемента по названию государства

Element\* Object::Search(const char\* name) {

Element\* rc = Head;

while (rc != nullptr && strcmp(rc->Data.Name, name) != 0) {

rc = rc->Next;

}

return rc;

}

// Метод удаления элемента из списка

bool Object::Delete(Element\* e) {

if (e != nullptr) {

if (e->Next != nullptr) {

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

}

if (e->Prev != nullptr) {

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

}

else {

Head = e->Next;

}

delete e;

return true;

}

return false;

}

// Метод удаления элемента по названию государства

bool Object::Delete(const char\* name) {

return Delete(Search(name));

}

// Метод удаления всех элементов списка

bool Object::DeleteList() {

while (Head != nullptr) {

Element\* p = Head;

Head = Head->Next;

delete p;

}

return true;

}

// Метод вывода списка государств в консоль

void Object::PrintList() {

Element\* e = Head;

while (e != nullptr) {

std::cout << "Название: " << e->Data.Name << std::endl;

std::cout << "Столица: " << e->Data.Capital << std::endl;

std::cout << "Население: " << e->Data.Population << std::endl;

std::cout << "Площадь: " << e->Data.Area << " км^2" << std::endl;

std::cout << "Президент: " << e->Data.PresidentLastName << std::endl;

std::cout << "--------------------------" << std::endl;

e = e->GetNext();

}

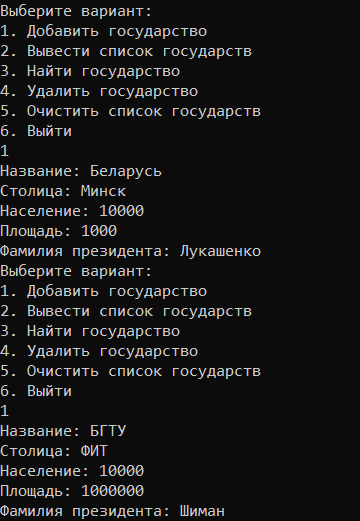
}

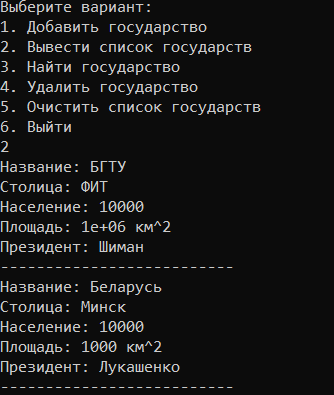
// Функция создания объекта управления списком государств

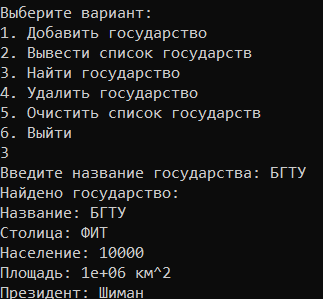
Object Create() {

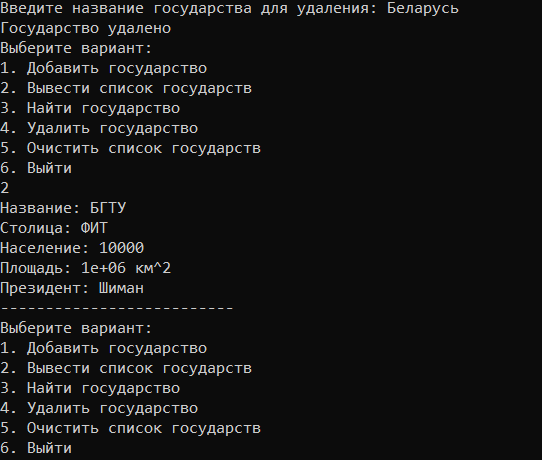
return Object();

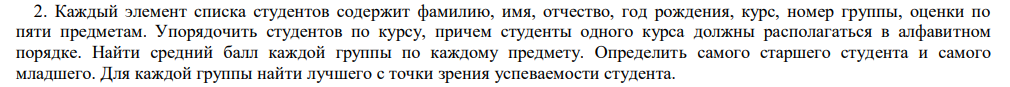
}

****

****

****

****

****

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#define SIZE 50

using namespace std;

struct Student {

char surname[SIZE];

char name[SIZE];

char patronymic[SIZE];

int day;

int month;

int year;

int course;

int group;

float math;

float physics;

float english;

float OAIP;

float history;

Student\* next;

Student\* prev;

};

struct AverageScore {

float math;

float physics;

float english;

float OAIP;

float history;

};

void insert(Student\* e, Student\*\* phead, Student\*\* plast); //размещение

Student\* setElement(); //ввод

void outputList(Student\*\* phead, Student\*\* plast); //вывод

void studentOld(Student\*\* phead, Student\*\* plast); // Поиск старшего

void studentYoung(Student\*\* phead, Student\*\* plast); // Поиск младшего

void outputFirstCourse(Student\*\* phead, Student\*\* plast); //Первый курс

void outputSecondCourse(Student\*\* phead, Student\*\* plast); // Второй курс

void outputThirdCourse(Student\*\* phead, Student\*\* plast); //Третий курс

void outputFourthCourse(Student\*\* phead, Student\*\* plast); //Четвёртый курс

void sortStudentsByCourse(Student\*\* phead); // сортировка

void calculateAverageScores(Student\*\* phead); // расчёт среднего значения

int main() {

setlocale(LC\_ALL, "rus");

Student\* head = nullptr;

Student\* last = nullptr;

int choice;

do {

cout << "1. Ввод данных" << endl;

cout << "2. Поиск старшего студента" << endl;

cout << "3. Поиск младшего студента" << endl;

cout << "4. Отсортировать студентов" << endl;

cout << "5. Вывод всего" << endl;

cout << "6. Первый курс" << endl;

cout << "7. Второй курс" << endl;

cout << "8. Третий курс" << endl;

cout << "9. Четвёртый курс" << endl;

cout << "10. Расчёт среднего балла по группам" << endl;

cout << "11. Выход" << endl;

cin >> choice;

switch (choice)

{

case 1:

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

break;

case 2:

studentOld(&head, &last);

break;

case 3:

studentYoung(&head, &last);

break;

case 4:

sortStudentsByCourse(&head);

break;

case 5:

outputList(&head, &last);

break;

case 6:

outputFirstCourse(&head, &last);

break;

case 7:

outputSecondCourse(&head, &last);

break;

case 8:

outputThirdCourse(&head, &last);

break;

case 9:

outputFourthCourse(&head, &last);

break;

case 10:

calculateAverageScores(&head);

break;

}

} while (choice != 11);

return 0;

}

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

{

if (\*plast == nullptr) {

\*plast = e;

\*phead = e;

}

else {

(\*plast)->next = e;

e->prev = \*plast;

\*plast = e;

}

}

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

{

Student\* temp = new Student();

if (!temp) {

cerr << "Ошибка выделения памяти";

return nullptr;

}

cout << "Введите фамилию: ";

cin >> temp->surname;

cout << "Введите имя: ";

cin >> temp->name;

cout << "Введите отчество: ";

cin >> temp->patronymic;

cout << "Введите дату рождения: " << endl;

cin >> temp->day >> temp->month >> temp->year;

cout << "Введите номер курса: ";

cin >> temp->course;

cout << "Введите номер группы: ";

cin >> temp->group;

cout << "Введите оценки по предметам: " << endl;

cout << "Математика: ";

cin >> temp->math;

cout << "Физика: ";

cin >> temp->physics;

cout << "Английский язык: ";

cin >> temp->english;

cout << "ОАиП: ";

cin >> temp->OAIP;

cout << "История Беларуси: ";

cin >> temp->history;

temp->next = nullptr;

temp->prev = nullptr;

return temp;

}

void outputList(Student\*\* phead, Student\*\* plast) // выводим всех студентов

{

Student\* t = \*phead;

if (t == nullptr) {

cout << "Список пуст!" << endl;

return;

}

while (t != nullptr)

{

cout << "--------------------------------------------------" << endl;

cout << "Фамилия:" << t->surname << endl;

cout << "Имя:" << t->name << endl;

cout << "Отчество:" << t->patronymic << endl;

cout << "Дата рождения:" << t->day << "." << t->month << "." << t->year << endl;

cout << "Курс:" << t->course << endl;

cout << "Группа:" << t->group << endl;

cout << "Оценки:" << endl;

cout << "Математика:" << t->math << endl;

cout << "Физика: " << t->physics << endl;

cout << "Английский язык: " << t->english << endl;

cout << "ОАиП:" << t->OAIP << endl;

cout << "История Беларуси: " << t->history << endl;

t = t->next;

}

}

void studentOld(Student\*\* phead, Student\*\* plast) // ищем старшего студента

{

int old = 0;

Student\* tmp = \*phead;

old = tmp->year;

while (tmp)

{

if (tmp->year < old)

{

old = tmp->year;

}

tmp = tmp->next;

}

Student\* t = \*phead;

while (t)

{

if (t->year == old)

{

cout << "Старший студент = " << t->surname << endl;

}

t = t->next;

}

}

void studentYoung(Student\*\* phead, Student\*\* plast) // ищем младшего студента

{

int young = 0;

Student\* tmp = \*phead;

young = tmp->year;

while (tmp)

{

if (tmp->year > young)

{

young = tmp->year;

}

tmp = tmp->next;

}

Student\* t = \*phead;

while (t)

{

if (t->year == young)

{

cout << "Младший студент = " << t->surname << endl;

}

t = t->next;

}

}

void sortStudentsByCourse(Student\*\* phead) // Сротировка студентов по курсу и фамилии

{

Student\* current = \*phead;

int swapped;

Student\* temp;

if (current == NULL || current->next == NULL)

{

return;

}

do {

swapped = 0;

current = \*phead;

while (current->next != nullptr)

{

// Если курс следующего студента меньше текущего или фамилия следующего при однаковом курсе раньше в алфавитном порядке то меняем их местами

if (current->course > current->next->course || (current->course == current->next->course && strcmp(current->surname, current->next->surname) > 0))

{

temp = current->next;

current->next = temp->next;

temp->next = current;

if (current->prev != NULL)

{

current->prev->next = temp;

}

temp->prev = current->prev;

current->prev = temp;

if (current == \*phead)

{

\*phead = temp;

}

if (current->next != NULL)

{

current->next->prev = current;

}

swapped = 1;

}

else

{

current = current->next;

}

}

} while (swapped);

}

void outputFirstCourse(Student\*\* phead, Student\*\* plast) // первый курс

{

Student \*t = \*phead;

float av = 0, temp;

char bestud[SIZE];

// среднее значение

if (t == NULL)

{

cout << "Список пуст!" << endl;

}

while (t)

{

if (t->course == 1)

{

temp = (t->math + t->physics + t->history + t->OAIP + t->english)/5;

cout << "Фамилия:" << t->surname << endl;

cout << "Имя:" << t->name << endl;

cout << "Отчество:" << t->patronymic << endl;

cout << "Дата рождения:" << t->day << "." << t->month << "." << t->year << endl;

cout << "Курс:" << t->course << endl;

cout << "Группа:" << t->group << endl;

cout << "Оценки:" << endl;

cout << "Математика:" << t->math << endl;

cout << "Физика:" << t->physics << endl;

cout << "Английский язык:" << t->english << endl;

cout << "ОАиП:" << t->OAIP << endl;

cout << "История Беларуси:" << t->history << endl<<endl;

t = t->next;

if (t != nullptr && temp > av)

{

av = temp;

strcpy(bestud, t->surname); // Копируем фамилию лучшего студента в массив

}

}

else

{

t = t->next;

}

}

cout << endl << "Лучший студент: " << bestud<<endl;

}

void outputSecondCourse(Student\*\* phead, Student\*\* plast) // второй курс

{

Student\* t = \*phead;

float av = 0, temp;

char bestud[SIZE];

if (t == NULL)

{

cout << "Список пуст!" << endl;

}

while (t)

{

if (t->course == 2)

{

temp = (t->math + t->physics + t->history + t->OAIP + t->english) / 5;

cout << "Фамилия:" << t->surname << endl;

cout << "Имя:" << t->name << endl;

cout << "Отчество:" << t->patronymic << endl;

cout << "Дата рождения:" << t->day << "." << t->month << "." << t->year << endl;

cout << "Курс:" << t->course << endl;

cout << "Группа:" << t->group << endl;

cout << "Оценки:" << endl;

cout << "Математика:" << t->math << endl;

cout << "Физика:" << t->physics << endl;

cout << "Английский язык:" << t->english << endl;

cout << "ОАиП:" << t->OAIP << endl;

cout << "История Беларуси:" << t->history << endl;

t = t->next;

if (t != nullptr && temp > av)

{

av = temp;

strcpy(bestud, t->surname); // Копируем фамилию лучшего студента в массив

}

}

else

t = t->next;

}

}

void outputThirdCourse(Student\*\* phead, Student\*\* plast) //третий курс

{

Student\* t = \*phead;

float av = 0, temp;

char bestud[SIZE];

if (t == NULL)

{

cout << "Список пуст!" << endl;

}

while (t)

{

if (t->course == 3)

{

temp = (t->math + t->physics + t->history + t->OAIP + t->english) / 5;

cout << "Фамилия:" << t->surname << endl;

cout << "Имя:" << t->name << endl;

cout << "Отчество:" << t->patronymic << endl;

cout << "Дата рождения:" << t->day << "." << t->month << "." << t->year << endl;

cout << "Курс:" << t->course << endl;

cout << "Группа:" << t->group << endl;

cout << "Оценки:" << endl;

cout << "Математика:" << t->math << endl;

cout << "Физика:" << t->physics << endl;

cout << "Английский язык:" << t->english << endl;

cout << "ОАиП:" << t->OAIP << endl;

cout << "История Беларуси:" << t->history << endl;

t = t->next;

if (t != nullptr && temp > av)

{

av = temp;

strcpy(bestud, t->surname); // Копируем фамилию лучшего студента в массив

}

}

else

t = t->next;

}

}

void outputFourthCourse(Student\*\* phead, Student\*\* plast) //четвёртый курс

{

Student\* t = \*phead;

float av = 0, temp;

char bestud[SIZE];

if (t == NULL)

{

cout << "Список пуст!" << endl;

}

while (t)

{

if (t->course == 4)

{

temp = (t->math + t->physics + t->history + t->OAIP + t->english) / 5;

cout << "Фамилия:" << t->surname << endl;

cout << "Имя:" << t->name << endl;

cout << "Отчество:" << t->patronymic << endl;

cout << "Дата рождения:" << t->day << "." << t->month << "." << t->year << endl;

cout << "Курс:" << t->course << endl;

cout << "Группа:" << t->group << endl;

cout << "Оценки:" << endl;

cout << "Математика:" << t->math << endl;

cout << "Физика:" << t->physics << endl;

cout << "Английский язык:" << t->english << endl;

cout << "ОАиП:" << t->OAIP << endl;

cout << "История Беларуси:" << t->history << endl;

t = t->next;

if (t != nullptr && temp > av)

{

av = temp;

strcpy(bestud, t->surname); // Копируем фамилию лучшего студента в массив

}

}

else

t = t->next;

}

}

void calculateAverageScores(Student\*\* phead) // Расчет средних баллов по каждому предмету для каждой группы

{

const int MAX\_GROUPS = 10;

int numStudents[MAX\_GROUPS] = { 0 }; // Массив для хранения количества студентов в каждой группе

float math[MAX\_GROUPS] = { 0 }, physics[MAX\_GROUPS] = { 0 }, totalEnglish[MAX\_GROUPS] = { 0 }, totalOAIP[MAX\_GROUPS] = { 0 }, totalHistory[MAX\_GROUPS] = { 0 };

// Проходим по списку студентов и суммируем оценки по каждому предмету для каждой группы

Student\* current = \*phead;

while (current != nullptr)

{

int groupIndex = current->group - 1; // Индекс группы в массиве

numStudents[groupIndex]++;

math[groupIndex] += current->math;

physics[groupIndex] += current->physics;

totalEnglish[groupIndex] += current->english;

totalOAIP[groupIndex] += current->OAIP;

totalHistory[groupIndex] += current->history;

current = current->next;

}

// Вычисляем средний балл по каждому предмету для каждой группы

for (int i = 0; i < MAX\_GROUPS; i++)

{

if (numStudents[i] != 0) {

float averageMath = math[i] / numStudents[i];

float averagePhysics = physics[i] / numStudents[i];

float averageEnglish = totalEnglish[i] / numStudents[i];

float averageOAIP = totalOAIP[i] / numStudents[i];

float averageHistory = totalHistory[i] / numStudents[i];

// Выводим результаты для каждой группы

cout << "Группа " << i + 1 << ":" << endl;

cout << "Средний балл по Математике: " << averageMath << endl;

cout << "Средний балл по Физике: " << averagePhysics << endl;

cout << "Средний балл по Английскому языку: " << averageEnglish << endl;

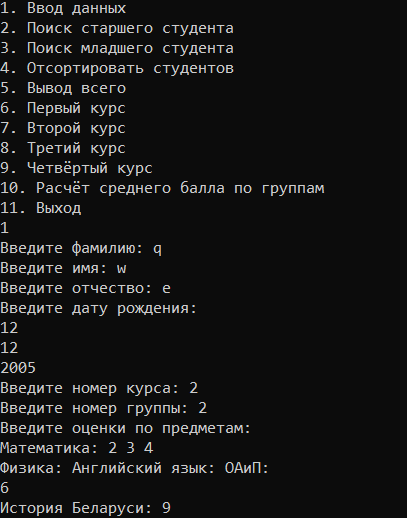
cout << "Средний балл по ОАиПу: " << averageOAIP << endl;

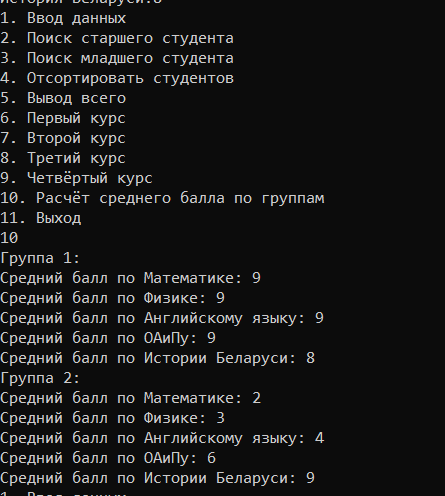
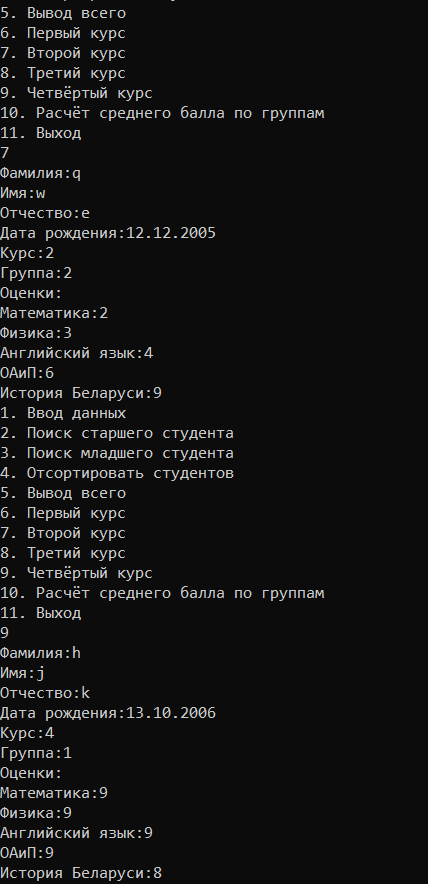
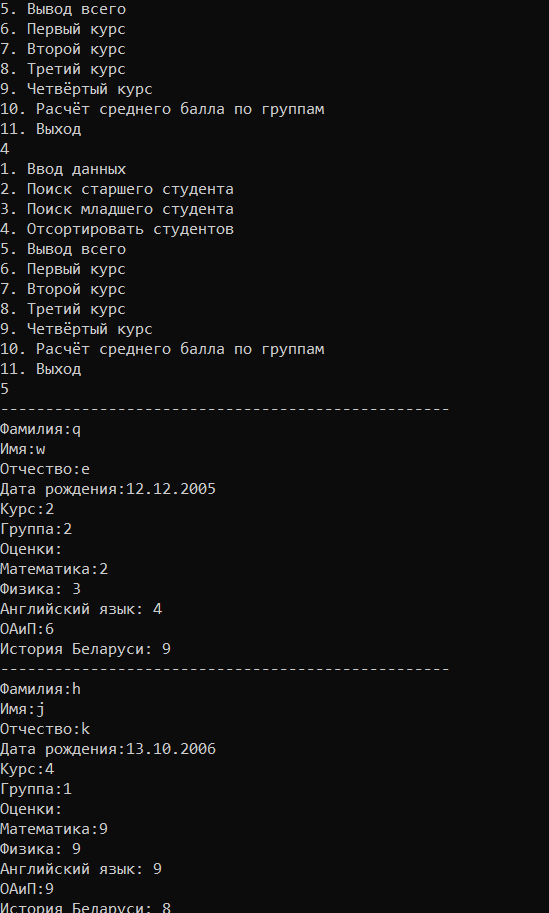
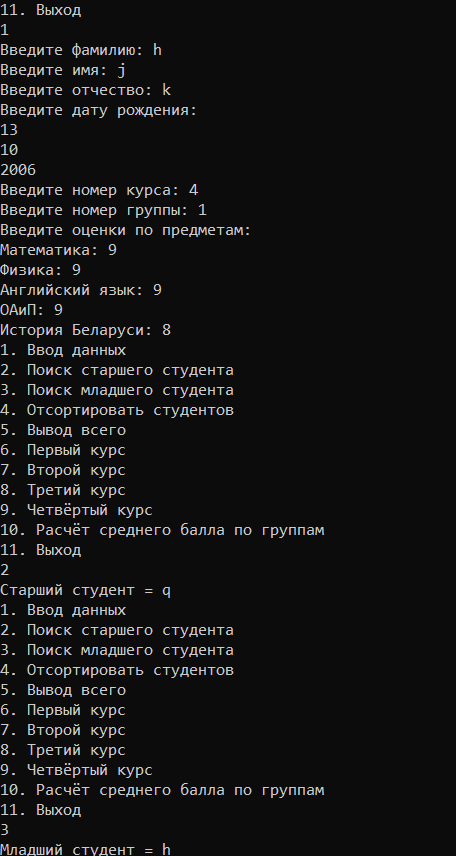
cout << "Средний балл по Истории Беларуси: " << averageHistory << endl;

}

}

}

****

****