Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа № 9

По дисциплине «Основы алгоритмизации и программирования»

На тему «Двусвязные списки»

Выполнил:

Кулешов Артём

Студент 1 курса 8 группы

Преподаватель: асс. Андронова М.В.

Минск, 2024

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

Вариант №5

#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;

};

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

Address\* setElement();

void appendToEnd(Address\* data, Address\*\* phead, Address\*\* plast);

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

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

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

void writeToFile(Address\*\* phead);

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

int main(void)

{

Address\* head = NULL;

Address\* last = NULL;

setlocale(LC\_CTYPE, "Rus");

char s[80]; int choice;

cout << endl;

do {

cout << "1. Ввод имени" << endl;

cout << "2. Удаление имени" << endl;

cout << "3. Вывод на экран" << endl;

cout << "4. Добавление имени в конец списка" << endl;

cout << "5. Поиск" << endl;

cout << "6. Выход" << endl;

cout << endl;

cin >> choice;

switch (choice)

{

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:

appendToEnd(setElement(), &head, &last); // Используем appendToEnd для добавления элемента в конец списка

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();

find(fname, &head);

break;

case 6:

exit(0);

break;

default:

cout << "Error." << endl;

break;

}

} while (choice != 6);

return 0;

}

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

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

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 appendToEnd(Address\* data, Address\*\* phead, Address\*\* plast) {

Address\* new\_node = new Address;

if (!new\_node) {

cerr << "Ошибка выделения памяти" << endl;

return;

}

\*new\_node = \*data; // Копируем данные

new\_node->next = nullptr; // Устанавливаем указатель как nullptr

new\_node->prev = \*plast;

if (\*plast) {

(\*plast)->next = new\_node;

}

\*plast = new\_node; // Обновляем указатель

if (!\*phead) {

\*phead = new\_node;

}

}

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

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

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 writeToFile(Address\*\* phead) //Запись в файл

{

struct Address\* t = \*phead;

FILE\* fp;

errno\_t err = fopen\_s(&fp, "mlist", "wb");

if (err)

{

cerr << "Файл не открывается" << endl;

exit(1);

}

cout << "Сохранение в файл" << endl;

while (t)

{

fwrite(t, sizeof(struct Address), 1, fp);

t = t->next;

}

fclose(fp);

}

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

void readFromFile(Address\*\* phead, Address\*\* plast) //Считывание из файла

{

struct Address\* t;

FILE\* fp;

errno\_t err = fopen\_s(&fp, "mlist", "rb");

if (err)

{

cerr << "Файл не открывается" << endl;

exit(1);

}

while (\*phead)

{

\*plast = (\*phead)->next;

delete\* phead;

\*phead = \*plast;

}

\*phead = \*plast = NULL;

cout << "Загрузка из файла" << endl;

while (!feof(fp))

{

t = new Address();

if (!t)

{

cerr << "Ошибка выделения памяти" << endl;

return;

}

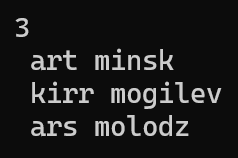
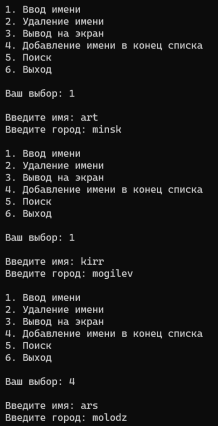
if (1 != fread(t, sizeof(struct Address), 1, fp)) break;

insert(t, phead, plast);

}

fclose(fp);

}



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<Windows.h>

#include <iostream>

int main() {

setlocale(LC\_ALL, "Russian");

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

int choice;

Object L1 = Create();

do {

std::cout<<"Выберитевариант: "<<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<<"Фамилияпрезидента: ";

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<<"Государствоненайдено"<<std::endl;

}

else {

std::cout<<"Найденогосударство:"<<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 <<"Списокгосударствочищен"<< std::endl;

}

else {

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

}

break;

}

default:

break;

}

} while (choice != 6);

return 0;

}

#pragmaonce

#include<iostream>

#include<cstring>

// Структура для хранения информации о государстве

structCountry {

char Name[50];

char Capital[50];

int Population;

doubleArea;

charPresidentLastName[50];

};

// Структура элемента списка государств

structElement {

Element\* Prev;

Element\* Next;

Country Data;

// Конструкторэлементасписка

Element(Element\* prev, constchar\* name, constchar\* capital, intpopulation, doublearea, constchar\* presidentLastName, Element\* next);

// Методы получения указателей на предыдущий и следующий элементы

Element\* GetNext();

Element\* GetPrev();

};

// Структура управления списком государств

structObject {

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(constchar\* name, constchar\* capital, intpopulation, doublearea, constchar\* presidentLastName);

bool Delete(Element\* e);

Element\* Search(constchar\* name);

bool Delete(constchar\* name);

bool DeleteList();

// Метод для вывода списка государств

void PrintList();

};

// Функция создания объекта управления списком государств

Object Create();

#include"List.h"

// Реализация конструктора элемента списка

Element::Element(Element\* prev, constchar\* name, constchar\* capital, intpopulation, doublearea, constchar\* 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

boolObject::Insert(constchar\* name, constchar\* capital, intpopulation, doublearea, constchar\* presidentLastName) {

if (Head == nullptr) {

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

}

else {

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

}

returntrue;

}

// Метод поиска элемента по названию государства

Element\* Object::Search(constchar\* name) {

Element\* rc = Head;

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

rc = rc->Next;

}

return rc;

}

// Методудаленияэлементаизсписка

boolObject::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;

}

deletee;

returntrue;

}

returnfalse;

}

// Метод удаления элемента по названию государства

boolObject::Delete(constchar\* name) {

return Delete(Search(name));

}

// Метод удаления всех элементов списка

boolObject::DeleteList() {

while (Head != nullptr) {

Element\* p = Head;

Head = Head->Next;

delete p;

}

returntrue;

}

// Метод вывода списка государств в консоль

voidObject::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();

}

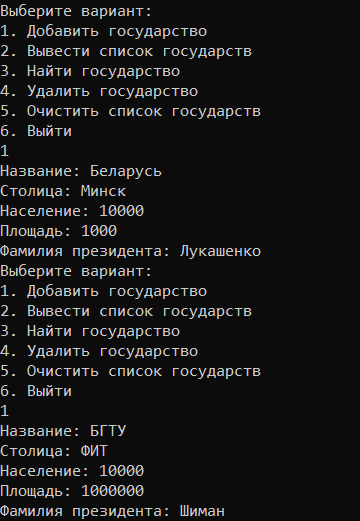
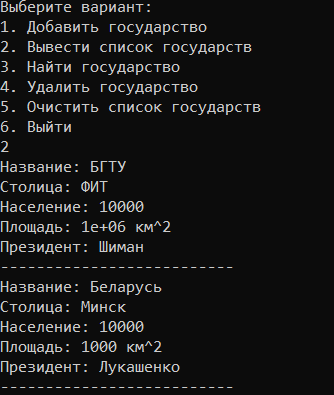
}

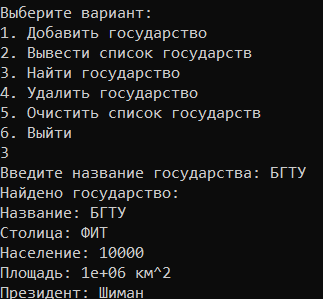
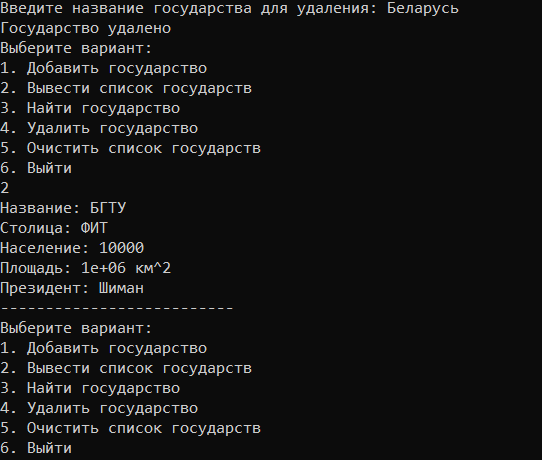
// Функция создания объекта управления списком государств

Object Create() {

returnObject();

}

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>

#defineSIZE 50

usingnamespace std;

structStudent {

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;

};

structAverageScore {

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); // Поискстаршего

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

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

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

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

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

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

void calculateAverageScores(Student\*\* phead); // расчётсреднегозначения

int main() {

setlocale(LC\_ALL, "rus");

Student\* head = nullptr;

Student\* last = nullptr;

int choice;

do {

cout <<"1. Вводданных"<< 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) //Добавлениевконецсписка

{

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 = newStudent();

if (!temp) {

cerr <<"Ошибка выделения памяти";

returnnullptr;

}

cout <<"Введите фамилию: ";

cin >> temp->surname;

cout <<"Введитеимя: ";

cin >> temp->name;

cout <<"Введитеотчество: ";

cin >> temp->patronymic;

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

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

cout <<"Введитеномеркурса: ";

cin >> temp->course;

cout <<"Введитеномергруппы: ";

cin >> temp->group;

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

cout <<"Математика: ";

cin >> temp->math;

cout <<"Физика: ";

cin >> temp->physics;

cout <<"Английскийязык: ";

cin >> temp->english;

cout <<"ОАиП: ";

cin >> temp->OAIP;

cout <<"ИсторияБеларуси: ";

cin >> temp->history;

temp->next = nullptr;

temp->prev = nullptr;

return temp;

}

void outputList(Student\*\* phead, Student\*\* plast) // выводимвсехстудентов

{

Student\* t = \*phead;

if (t == nullptr) {

cout <<"Списокпуст!"<< 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 <<"Английскийязык: "<< t->english << endl;

cout <<"ОАиП:"<< t->OAIP << endl;

cout <<"ИсторияБеларуси: "<< t->history << endl;

t = t->next;

}

}

void studentOld(Student\*\* phead, Student\*\* plast) // ищемстаршегостудента

{

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 <<"Старшийстудент = "<< t->surname << endl;

}

t = t->next;

}

}

void studentYoung(Student\*\* phead, Student\*\* plast) // ищеммладшегостудента

{

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 <<"Младшийстудент = "<< t->surname << endl;

}

t = t->next;

}

}

void sortStudentsByCourse(Student\*\* phead) // Сротировкастудентовпокурсуифамилии

{

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) // первыйкурс

{

Student \*t = \*phead;

float av = 0, temp;

char bestud[SIZE];

// среднеезначение

if (t == NULL)

{

cout <<"Списокпуст!"<< 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 <<"Английскийязык:"<< t->english << endl;

cout <<"ОАиП:"<< t->OAIP << endl;

cout <<"ИсторияБеларуси:"<< t->history << endl<<endl;

t = t->next;

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

{

av = temp;

strcpy(bestud, t->surname); // Копируемфамилиюлучшегостудентавмассив

}

}

else

{

t = t->next;

}

}

cout << endl <<"Лучшийстудент: "<< bestud<<endl;

}

void outputSecondCourse(Student\*\* phead, Student\*\* plast) // второйкурс

{

Student\* t = \*phead;

float av = 0, temp;

char bestud[SIZE];

if (t == NULL)

{

cout <<"Списокпуст!"<< 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 <<"Английскийязык:"<< t->english << endl;

cout <<"ОАиП:"<< t->OAIP << endl;

cout <<"ИсторияБеларуси:"<< t->history << endl;

t = t->next;

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

{

av = temp;

strcpy(bestud, t->surname); // Копируемфамилиюлучшегостудентавмассив

}

}

else

t = t->next;

}

}

void outputThirdCourse(Student\*\* phead, Student\*\* plast) //третийкурс

{

Student\* t = \*phead;

float av = 0, temp;

char bestud[SIZE];

if (t == NULL)

{

cout <<"Списокпуст!"<< 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 <<"Английскийязык:"<< t->english << endl;

cout <<"ОАиП:"<< t->OAIP << endl;

cout <<"ИсторияБеларуси:"<< t->history << endl;

t = t->next;

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

{

av = temp;

strcpy(bestud, t->surname); // Копируемфамилиюлучшегостудентавмассив

}

}

else

t = t->next;

}

}

void outputFourthCourse(Student\*\* phead, Student\*\* plast) //четвёртыйкурс

{

Student\* t = \*phead;

float av = 0, temp;

char bestud[SIZE];

if (t == NULL)

{

cout <<"Списокпуст!"<< 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 <<"Английскийязык:"<< t->english << endl;

cout <<"ОАиП:"<< t->OAIP << endl;

cout <<"ИсторияБеларуси:"<< t->history << endl;

t = t->next;

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

{

av = temp;

strcpy(bestud, t->surname); // Копируемфамилиюлучшегостудентавмассив

}

}

else

t = t->next;

}

}

void calculateAverageScores(Student\*\* phead) // Расчетсреднихбалловпокаждомупредметудлякаждойгруппы

{

constint MAX\_GROUPS = 10;

int numStudents[MAX\_GROUPS] = { 0 }; // Массивдляхраненияколичествастудентоввкаждойгруппе

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; // Индексгруппывмассиве

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;

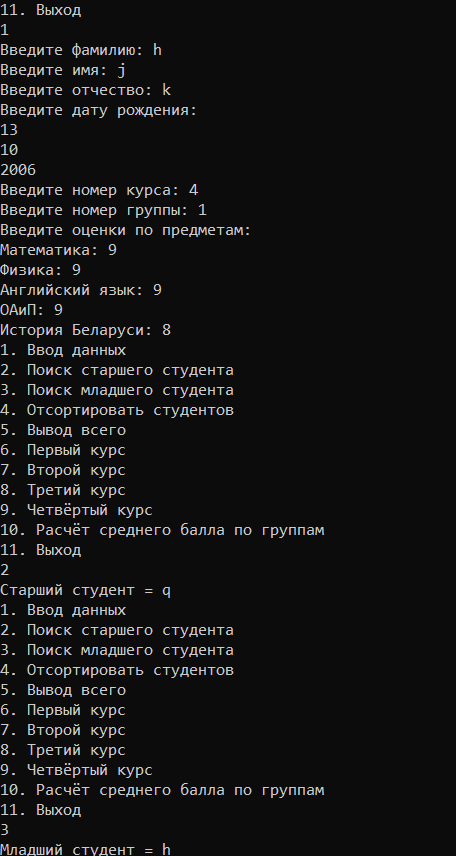
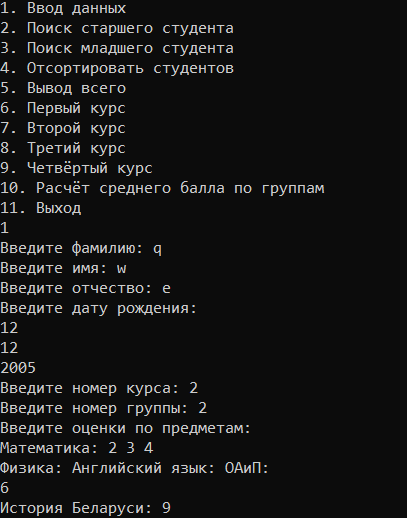
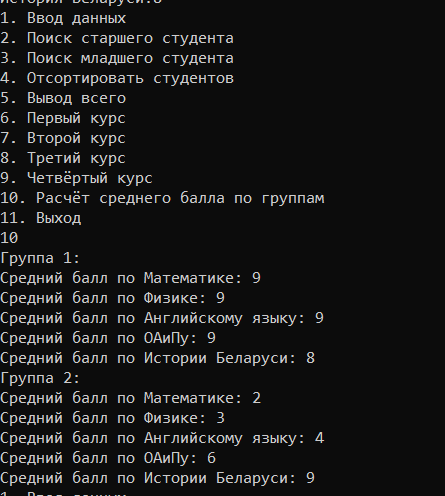
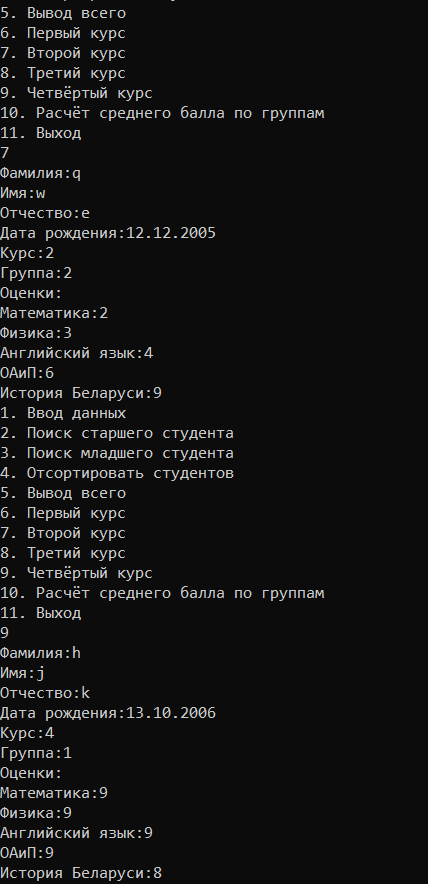
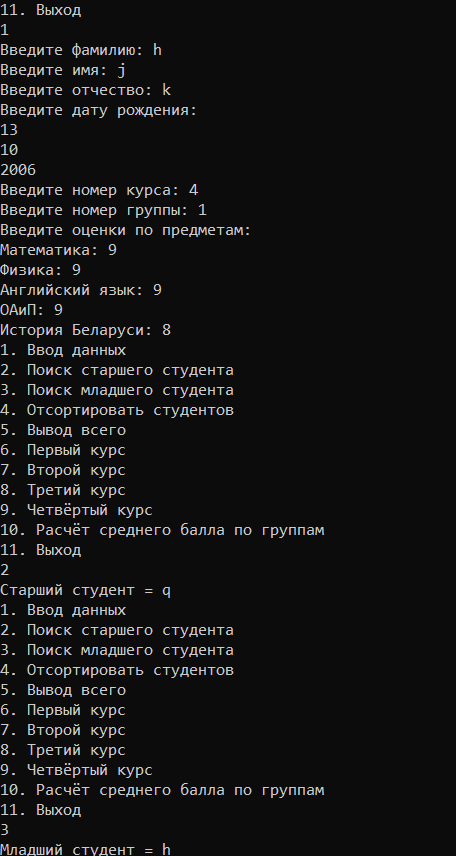
cout <<"Средний балл по ОАиПу: "<< averageOAIP << endl;

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

}

}

}

**CountX (int x)** – функция подсчёта числа элементов списка с заданным значением **x**.

#include<iostream>

#include<fstream>

#include<Windows.h>

usingnamespace std;

constunsignedint NAME\_SIZE = 30;

constunsignedint CITY\_SIZE = 20;

// Структура для элемента списка

structAddress

{

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(charname[NAME\_SIZE], Address\*\* phead, Address\*\* plast);

// Функция для поиска элемента по имени

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

// Функция для удаления первого встречающегося элемента с заданным возрастом

void countX(Address\*\* phead, Address\*\* plast, intx);

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 <<"Введитеимя: ";

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 <<"Введитеимя: ";

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<<"Неправильныйвыбор"<<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 = newAddress();

if (!temp)

{

cerr<<"Ошибкавыделенияпамяти";

returnNULL;

}

// Ввод данных для нового элемента

cout<<"Введитеимя: ";

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(charname[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(charname[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, intx)

{

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 <<"Количестволюдей: "<< count << endl << endl;

}

}

