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 10 группы

Жамойдо Артём Игоревич

Преподаватель: асс. Андронова М.В.

Минск, 2024

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

**Вариант 4**

**deleteKLast(int k)** – функция удаления **К** последних элементов списка.

#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;

};

// Объявление функции удаления К последних элементов списка

void deleteKLast(int k, Address\*\* phead, Address\*\* plast);

// Функция для отображения меню и возврата выбора пользователя

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. Запись в файл" << endl;

cout << "6. Чтение из файла" << endl;

cout << "7. Удаление k последних элементов" << endl;

cout << "8. Выход" << endl;

cout << endl;

do

{

cout << "Ваш выбор: ";

cin.sync();

gets\_s(s);

cout << endl;

c = atoi(s);

} while (c < 0 || c > 8);

return c;

}

// Функция для добавления элемента в конец списка

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

{

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;

}

}

// Функция для создания элемента и ввода его значений с клавиатуры

Address\* setElement()

{

Address\* temp = new Address();

if (!temp)

{

cerr << "Ошибка выделения 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;

}

// Функция для вывода списка на экран

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

{

Address\* t = \*phead;

while (t)

{

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

t = t->next;

}

cout << "" << endl;

}

// Функция для поиска имени в списке

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

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

}

// Функция для удаления имени из списка

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

{

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 записи списка в файл

void writeToFile(Address\*\* phead)

{

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);

}

// Функция для чтения списка из файла

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

{

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);

}

// Функция для удаления К последних элементов списка

void deleteKLast(int k, Address\*\* phead, Address\*\* plast)

{

Address\* current = \*plast;

for (int i = 0; i < k && current; i++)

{

if (current == \*phead)

{

\*phead = current->next;

if (\*phead)

(\*phead)->prev = NULL;

delete current;

current = \*phead;

}

else

{

current = current->prev;

delete current->next;

current->next = NULL;

\*plast = current;

}

}

}

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); // Удаление имени

}

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); // Поиск по имени

}

break;

case 5: writeToFile(&head);

break;

case 6: readFromFile(&head, &last);

break;

case 7:

{

int k;

cout << "Введите количество последних элементов для удаления: ";

cin >> k;

cin.ignore();

deleteKLast(k, &head, &last);

}

break;

case 8: exit(0); // Выход

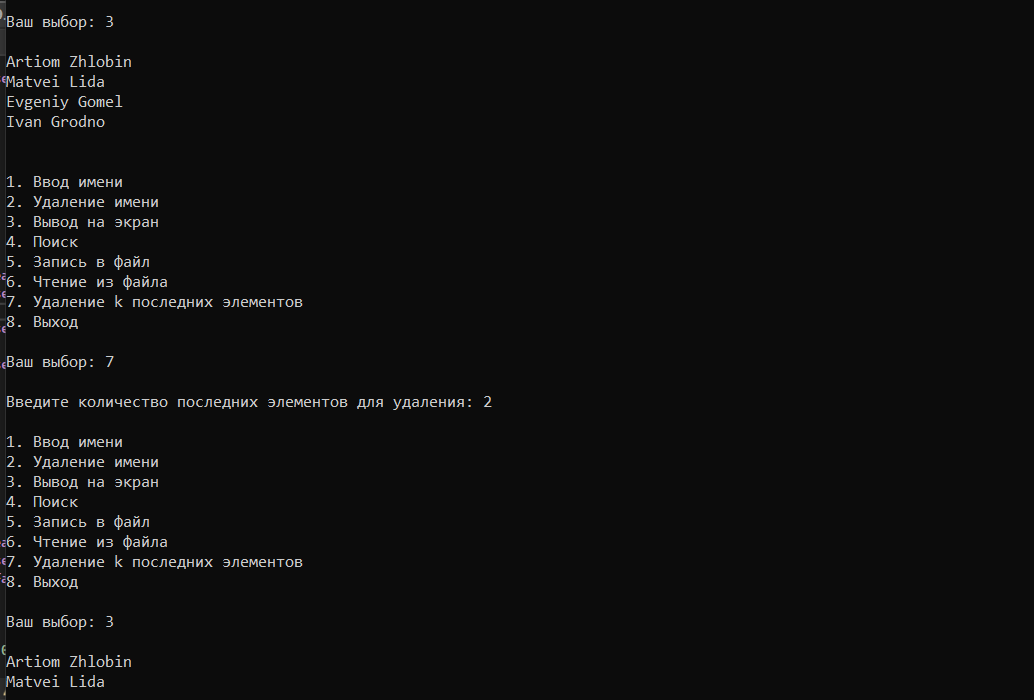
default: exit(1);

}

}

return 0;

}

****

Доп. задачи

1. В приложении 2 приведен проект, в котором реализован проект с использованием *двусвязного* *списка*. На основе данного проекта разработать функции, которые предлагается создать в данном приложении.

**Вариант 4**

#include <iostream>

#include <string>

#include <Windows.h>

using namespace std;

const unsigned int AUTHOR\_SIZE = 50;

const unsigned int TITLE\_SIZE = 100;

const unsigned int PUBLISHER\_SIZE = 50;

const unsigned int SECTION\_SIZE = 30;

const unsigned int ORIGIN\_SIZE = 20;

struct book

{

string author;

string title;

string publisher;

string section;

string origin;

bool available;

book\* next;

book\* prev;

};

// Функция для добавления книги в конец списка

void insertBook(book\* e, book\*\* phead, book\*\* plast)

{

book\* p = \*plast;

if (\*plast == NULL)

{

e->next = NULL;

e->prev = NULL;

\*plast = e;

\*phead = e;

}

else

{

p->next = e;

e->next = NULL;

e->prev = p;

\*plast = e;

}

}

// Функция для создания новой книги

book\* setBook()

{

book\* temp = new book();

if (!temp)

{

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

return NULL;

}

cout << "Введите автора: ";

cin.ignore();

getline(cin, temp->author);

cout << "Введите название: ";

getline(cin, temp->title);

cout << "Введите издательство: ";

getline(cin, temp->publisher);

cout << "Введите раздел библиотеки: ";

getline(cin, temp->section);

cout << "Введите происхождение книги (покупка, кража, подарок): ";

getline(cin, temp->origin);

cout << "Книга доступна в данный момент? (1 - Да, 0 - Нет): ";

cin >> temp->available;

temp->next = NULL;

temp->prev = NULL;

return temp;

}

// Функция для вывода списка книг

void outputLibrary(book\*\* phead, book\*\* plast)

{

book\* current = \*phead;

if (current == NULL)

{

cout << "Библиотека пуста!" << endl;

}

while (current)

{

cout << "Автор: " << current->author << endl;

cout << "Название: " << current->title << endl;

cout << "Издательство: " << current->publisher << endl;

cout << "Раздел: " << current->section << endl;

cout << "Происхождение: " << current->origin << endl;

cout << "Доступна в данный момент: " << (current->available ? "Да" : "Нет") << endl;

current = current->next;

}

cout << "" << endl;

}

// Функция для поиска книги по автору

void findBookByAuthor(string author, book\*\* phead)

{

book\* current = \*phead;

bool found = false;

while (current)

{

if (current->author == author)

{

found = true;

cout << "Книга найдена: " << current->title << " (" << current->publisher << ")" << endl;

}

current = current->next;

}

if (!found)

{

cout << "Книги данного автора не найдено." << endl;

}

}

// Функция для удаления списка книг

void deleteList(book\*\* phead, book\*\* plast)

{

book\* current = \*phead;

book\* next = NULL;

while (current != NULL)

{

next = current->next;

delete current;

current = next;

}

\*phead = NULL;

\*plast = NULL;

cout << "Список успешно удален!" << endl;

}

// Функция для подсчета количества элементов в списке книг

void countList(book\*\* phead, book\*\* plast)

{

book\* current = \*phead;

int count = 0;

while (current != NULL)

{

count++;

current = current->next;

}

cout << "Число элементов в списке: " << count << endl;

}

int main()

{

book\* libraryHead = NULL;

book\* libraryLast = NULL;

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int choice;

string author;

cout << "\n1. Добавить книгу" << endl;

cout << "2. Вывести каталог" << endl;

cout << "3. Найти книгу по автору" << endl;

cout << "4. Удалить список" << endl;

cout << "5. Подсчитать количество элементов в списке" << endl;

cout << "6. Выход" << endl;

for (;;)

{

cout << "Ваш выбор: ";

cin >> choice;

switch (choice)

{

case 1:

insertBook(setBook(), &libraryHead, &libraryLast);

break;

case 2:

outputLibrary(&libraryHead, &libraryLast);

break;

case 3:

cout << "Введите автора для поиска книги: ";

cin.ignore();

getline(cin, author);

findBookByAuthor(author, &libraryHead);

break;

case 4:

deleteList(&libraryHead, &libraryLast);

break;

case 5:

countList(&libraryHead, &libraryLast);

break;

case 6:

cout << "До свидания!" << endl;

exit(0);

break;

default:

cout << "Некорректный выбор, повторите пожалуйста." << endl;

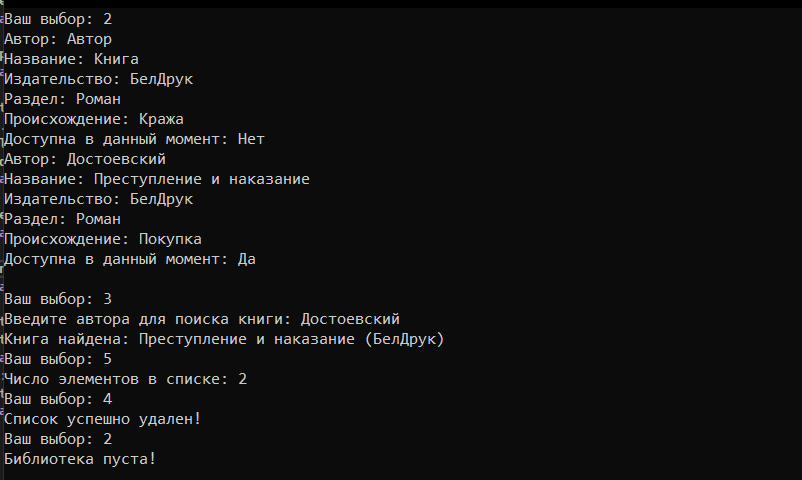
break;

}

}

return 0;

}



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

**Вариант 8**

#include <iostream>

#include <string.h>

const unsigned int NAME\_SIZE = 30;

const unsigned int CITY\_SIZE = 20;

using namespace std;

struct address

{

string name;

string sername;

string klass;

int scoreSub; // Количество предметов

address\* next;

address\* prev;

};

// Функция для добавления элемента в список

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

{

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;

}

}

// Функция для создания нового элемента

address\* setelement()

{

address\* temp = new address();

if (!temp)

{

cerr << "Ошибка выделения памяти памяти";

return NULL;

}

cout << "Введите имя: ";

cin >> temp->name;

cout << "Введите фамилию: ";

cin >> temp->sername;

cout << "Введите класс: ";

cin >> temp->klass;

cout << "Введите количество предметов: ";

cin >> temp->scoreSub;

temp->next = NULL;

temp->prev = NULL;

return temp;

}

// Функция для вывода списка на экран

void outputlist(address\*\* phead, address\*\* plast)

{

address\* t = \*phead;

if (t == NULL)

{

cout << "Список пуст." << endl;

}

while (t)

{

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

cout << "Имя: " << t->name << endl;

cout << "Класс: " << t->klass << endl;

cout << "Количество предметов: " << t->scoreSub << endl;

t = t->next;

}

cout << "" << endl;

}

// Функция для поиска по имени в списке

void find(string name, address\*\* phead)

{

address\* current = \*phead;

bool found = false;

while (current)

{

if (current->name == name)

{

found = true;

cout << "Фамилия: " << current->sername << endl;

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

cout << "Класс: " << current->klass << endl;

cout << "Количество предметов: " << current->scoreSub << endl;

}

current = current->next;

}

}

// Функция для подсчета количества элементов в списке

void countlist(address\*\* phead, address\*\* plast)

{

address\* t = \*phead;

int i = 0;

while (t != NULL)

{

i++;

t = t->next;

}

cout << "Количество элементов: " << i << endl;

}

// Функция для удаления всех элементов в списке

void deleteList(address\*\* phead, address\*\* plast)

{

address\* current = \*phead;

address\* next = NULL;

while (current != NULL)

{

next = current->next;

delete current;

current = next;

}

\*phead = NULL;

\*plast = NULL;

cout << "Список успешно удален!" << endl;

}

int main()

{

address\* head = NULL;

address\* last = NULL;

setlocale(LC\_CTYPE, "rus");

int choice;

char s[80]{};

// Меню программы

cout << endl;

cout << "1. Ввод элемента" << endl;

cout << "2. Вывод на экран" << endl;

cout << "3. Поиск" << endl;

cout << "4. Количество элементов" << endl;

cout << "5. Очищение списка" << endl;

cout << "6. Выход" << endl;

cout << endl;

for (;;)

{

cout << "Ваш выбор: ";

cin >> choice;

switch (choice)

{

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

break;

case 2: outputlist(&head, &last);

break;

case 3:

{

char fname[NAME\_SIZE];

cout << "Введите имя: ";

cin >> fname;

find(fname, &head);

}

break;

case 4: countlist(&head, &last);

break;

case 5: deleteList(&head, &last);

break;

case 6: exit(0);

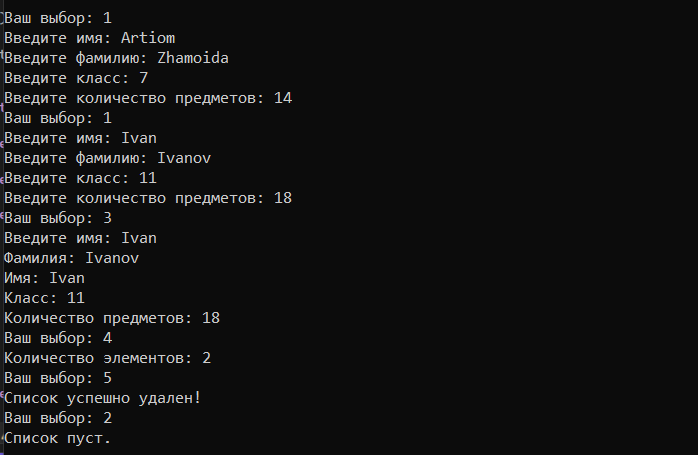
default: exit(1);

}

}

return 0;

}

****

3. 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 <algorithm>

#include <map>

using namespace std;

struct item

{

int value;

char fname[30] = {};

char sname[30] = {};

char thname[30] = {};

int year;

int course;

int group;

int marks[5];

float middle\_mark = 0;

item\* next;

item\* prev;

};

item\* plist = NULL;

item\* p;

item\* plist1 = NULL;

item\* p1;

int count1 = 0; // Подсчет всех элементов

int counter = 1; // Для функции вывода

// Добавление студента

void add()

{

cout << endl;

char fname[50];

char sname[50];

char thname[50];

cout << "Введите Имя студента : ";

cin >> fname >> sname >> thname;

int year;

cout << "Введите год рождения : ";

cin >> year;

int group, course, marks[5]{};

cout << "Введите курс и номер группы : ";

cin >> course >> group;

for (int i = 0; i < 5; i++)

{

cout << "Введите оценку номер " << i + 1 << " : ";

cin >> marks[i];

}

p = new item;

for (int i = 0; i < strlen(fname); i++)

{

p->fname[i] = fname[i];

}

for (int i = 0; i < strlen(sname); i++)

{

p->sname[i] = sname[i];

}

for (int i = 0; i < strlen(thname); i++)

{

p->thname[i] = thname[i];

}

p->year = year;

p->course = course;

p->group = group;

for (int i = 0; i < 5; i++)

{

p->marks[i] = marks[i];

p->middle\_mark += marks[i];

}

p->middle\_mark /= 5;

p->next = plist;

p->prev = NULL;

if (p->next != NULL)

{

item\* previ;

previ = p;

p = p->next;

p->prev = previ;

p = previ;

}

plist = p;

count1++;

cout << "Студент был успешно добавлен в начало списка." << endl;

cout << endl;

}

// Сортировка списка по курсам

void Sort()

{

cout << endl;

int val = count1 - 1;

for (int i = 0; i < count1; i++)

{

item\* host1 = p;

item\* host2 = p->next;

for (int j = 0; j < val; j++)

{

if (host1->course < host2->course)

{

swap(host1->fname, host2->fname);

swap(host1->sname, host2->sname);

swap(host1->thname, host2->thname);

swap(host1->course, host2->course);

swap(host1->group, host2->group);

swap(host1->year, host2->year);

swap(host1->marks, host2->marks);

swap(host1->middle\_mark, host2->middle\_mark);

}

host1 = host1->next;

host2 = host2->next;

}

val--;

}

cout << "Сортировка проведена успешно!" << endl;

cout << endl;

}

// Вывод всех студентов

void write()

{

cout << endl;

item\* host = p;

for (int i = 0; i < count1; i++)

{

cout << "Имя : " << host->fname << " " << host->sname << " " << host->thname << endl;

cout << "Год рождения : " << host->year << " Курс : " << host->course << " Группа : " << host->group << endl;

cout << "Оценки : " << host->marks[0] << " " << host->marks[1] << " " << host->marks[2] << " " << host->marks[3] << " " << host->marks[4] << endl;

host = host->next;

}

cout << endl;

}

// Поиск лучшего студента по успеваемости в каждой группе

void better\_student()

{

cout << endl;

// Создаем карту для хранения лучших студентов по группам

map<int, pair<item\*, float>> best\_students; // Ключ: номер группы, Значение: пара лучшего студента и его среднего балла

item\* host = p;

for (int i = 0; i < count1; i++)

{

// Если текущий студент лучше текущего лучшего студента в группе, заменяем его

if (best\_students.find(host->group) == best\_students.end() || host->middle\_mark > best\_students[host->group].second)

{

best\_students[host->group] = make\_pair(host, host->middle\_mark);

}

host = host->next;

}

// Выводим лучших студентов по группам

for (const auto& entry : best\_students)

{

cout << "Лучший студент в группе " << entry.first << " : " << entry.second.first->fname << " " << entry.second.first->sname << " " << entry.second.first->thname << endl;

}

cout << endl;

}

// Определение самого старшего и самого младшего студента

void max\_min\_year()

{

cout << endl;

item\* host = p;

item\* eldest = nullptr;

item\* youngest = nullptr;

int max\_age = -1;

int min\_age = INT\_MAX;

for (int i = 0; i < count1; i++)

{

if (host->year > max\_age)

{

max\_age = host->year;

youngest = host;

}

if (host->year < min\_age)

{

min\_age = host->year;

eldest = host;

}

host = host->next;

}

if (eldest)

{

cout << "Самый старший студент : " << eldest->fname << " " << eldest->sname << " " << eldest->thname << endl;

}

if (youngest)

{

cout << "Самый младший студент : " << youngest->fname << " " << youngest->sname << " " << youngest->thname << endl;

}

cout << endl;

}

// Расчет среднего балла по группам

void middle\_groups()

{

cout << endl;

map<int, pair<float, int>> group\_avg; // Хранит средний балл и количество студентов в каждой группе

item\* host = p;

for (int i = 0; i < count1; i++)

{

group\_avg[host->group].first += host->middle\_mark;

group\_avg[host->group].second++;

host = host->next;

}

cout << "Средний балл по группам:" << endl;

for (auto& entry : group\_avg)

{

cout << "Группа " << entry.first << " : " << entry.second.first / entry.second.second << endl;

}

cout << endl;

}

void main()

{

setlocale(LC\_CTYPE, "Rus");

int c;

do

{

cout << "Меню" << endl;

cout << "1 - Добавление студента." << endl;

cout << "2 - Вывод всех студентов." << endl;

cout << "3 - Сортировка в порядке курсов." << endl;

cout << "4 - Лучший студент по успеваемости." << endl;

cout << "5 - Самый старший и младший студент." << endl;

cout << "6 - Средний балл по группам." << endl;

cout << "7 - Выход." << endl;

cout << "Ваш выбор: ";

cin >> c;

switch (c)

{

case 1:

add();

break;

case 2:

write();

break;

case 3:

Sort();

break;

case 4:

better\_student();

break;

case 5:

max\_min\_year();

break;

case 6:

middle\_groups();

break;

case 7:

cout << "Конец" << endl;

break;

default:

cout << "Некорректный запрос, повторите пожалуйста." << endl;

break;

}

} while (c != 7);

}

