МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО «СЕВЕРО-КАВКАЗСКИЙ ГОРНО-МЕТАЛЛУРГИЧЕСКИЙ ИНСТИТУТ (ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ)»

Факультет: информационных технологий и электронной техники

Кафедра: Автоматизированной обработки информации

Направление подготовки: 09.03.01 Информатика и вычислительная техника

Дисциплина: Использование СУБД для создания программных систем и их компонентов

Группа: ИВБ-19-1

Студент: Иванов А.А.

**КУРСОВАЯ РАБОТА**

Тема:

Программа для ведения базы данных, содержащей информацию о студентах

.

«К защите допускаю»  
Руководитель:

Томаев Мурат Хасанбекович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Оценка при защите

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Владикавказ 2021

ФГБОУ ВО «СЕВЕРО-КАВКАЗСКИЙ ГОРНО-МЕТАЛЛУРГИЧЕСКИЙ ИНСТИТУТ (ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ)»

Кафедра «Информатика и вычислительная техника»

ЗАДАНИЕ

на курсовую работу по дисциплине «Использование СУБД для создания программных систем и их компонентов»

Студент **Иванов А.А.** Группа **ИВБ-19-1**

**1. Тема курсовой работы:** Программа для ведения базы данных, содержащей информацию о студентах

**2. Исходные данные:**

2.1 Требование к характеру выходных данных;

2.3 Требование к пользовательскому интерфейсу;

2.2 Необходимая дополнительная информация.

**3. Требования к оформлению:**

3.1. Пояснительная записка должна быть оформлена в редакторе в соответствии с требованиями ГОСТ.

3.2. В пояснительной записке должны содержаться следующие разделы:

1. Введение

2. Постановка задачи

2. Общий раздел (Описание предметной области рассматриваемого объекта)

3. Специальный раздел (Реализация приложения)

4. Заключение

5. Список литературы

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Подпись)

**ОТЗЫВ**

на курсовую работу студента

Кафедра «Информатика и вычислительная техника»

Дисциплина «Использование СУБД для создания программных систем и их компонентов»

Тема: Для реляционной базы «Табель учета рабочего времени» создать «Table-valued» функцию MS SQL Server на основе сборки .NET выполняющую формирование и вывод статистической информации об количестве фактически отработанных часов, количестве опозданий и объеме переработок для каждого сотрудника за выбранный календарный период

Студент (Ф.И.О) Иванов А.А. гр. ИВБ 19-1

Руководитель курсовой работы **Томаев Мурат Хасанбекович**

|  |  |  |
| --- | --- | --- |
| **№** | **Критерии, при назначении хотя бы одного из которых работа оценивается на «неудовлетворительно»** | **Баллы** |
| 1 | Тема и (или) содержание работы не относится к предмету дисциплины |  |
| 2 | Соответствие заявленной темы приказу на закрепление тем |  |
| 3 | Неструктурированный план курсовой работы |  |

**Рейтинг работы**

|  |  |  |
| --- | --- | --- |
| **№** | **Наименования показателя** | **Баллы** |
| **1** | **Содержательная часть** |  |
| 1.1 | Аналитический обзор методов решения поставленной задачи |  |
| 1.2 | Наличие корректной формальной постановки задачи |  |
| 1.3 | Наличие описания алгоритма решения поставленной задачи |  |
| 1.4 | Программная реализация (листинг) |  |
| 1.5 | Трассировка листинга |  |
| 1.6 | Полнота списка используемой литературы |  |
| **Итого по содержательной части (max 60 баллов)** | |  |
| **2** | **Оформление и информационное сопровождение работы** |  |
| 2.1 | Качество оформления, язык, стиль и грамматический уровень работы |  |
| 2.2 | Правильность оформления программной и графической документации |  |
| **Итого по оформлению и информационному сопровождению работы (max 20 баллов)** | |  |
| **3** | **Защита курсовой работы** |  |
| 3.1 | Структура и содержание доклада |  |
| 3.2 | Ответы на вопросы |  |
| **Итого по защите курсовой работы (max 20 баллов)** | |  |
| **ИТОГО по рейтингу работы (max 100 баллов):** | |  |

***Дополнительные замечания*** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка (прописью)

Руководитель курсовой работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись)*

**ОГЛАВЛЕНИЕ**

[**ВВЕДЕНИЕ** 5](#_Toc70490548)

[**ИСПОЛЬЗУЕМЫЕ ФУНКЦИИ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ** 7](#_Toc70490549)

[**ОПИСАНИЕ ПРОГРАММНОЙ РЕАЛИЗАЦИИ** 8](#_Toc70490550)

[**ПРОТОКОЛЫ, ПОДТВЕРЖДАЮЩИЕ РАБОТОСПОСОБНОСТЬ ПРОГРАММЫ** 18](#_Toc70490551)

[**ЗАКЛЮЧЕНИЕ** 20](#_Toc70490552)

[**СПИСОК ЛИТЕРАТУРЫ** 21](#_Toc70490553)

# **ВВЕДЕНИЕ**

Объектно-ориентированное программирование (ООП) — парадигма программирования, в которой основными концепциями являются понятия объектов и классов.

Объектно-ориентированные языки программирования пользуются в последнее время большой популярностью среди программистов, так как они позволяют использовать преимущества объектно-ориентированного подхода не только на этапах проектирования и конструирования программных систем, но и на этапах их реализации, тестирования и сопровождения.

Наиболее распространенным объектно-ориентированным языком программирования, безусловно, является C++. Свободно распространяемые коммерческие системы программирования C++ существуют практически на любой платформе.

Язык C++ был создан в 1983г. его разработчиком являлся Бъярн Страуструп из лаборатории Белл. Однако до 1987г. новый язык находился в стадии доработки.

С++ является языком программирования общего назначения. Именно этот язык хорошо известен своей эффективностью, экономичностью, и переносимостью.

Использование С++ в качестве инструментального языка позволяет получать быстрые и компактные программы. Во многих случаях программы, написанные на С++, сравнимы по скорости с программами, написанными на языке ассемблера.

Целью курсовой работы является разработка программы на С++ для обработки данных, представленных в виде таблиц базы данных. Программа должна корректно обрабатывать данные, представленные в виде таблиц с информацией о дисциплинах, преподавателях и студентах.

Программа должна быть создана так, чтобы любой пользователь без труда мог разобраться во всех её функциях и возможностях. Следовательно, можно определить основные функции программы:

1. запоминать вводимые значения, показывать базу данных;

2. добавлять, редактировать данные;

3. осуществлять поиск в базе;

4. выдавать информацию по различным запросом;

5. удалять и сохранять изменения в базе;

6. сохранять всю информацию, а также иметь доступ к открытию информации.

# **ИСПОЛЬЗУЕМЫЕ ФУНКЦИИ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ**

**char** **\*strcpy(char** **\****strDestination***,** **const** **char** **\****strSource***) –** функция копирует строку strSource в массив strDestination.

int memcmp(const void \* memptr1, const void \* memptr2, size\_t num) – сравнивает первые num байтов блока памяти указателя memptr1с первыми num байтами блока памяти memptr2. Возвращаемое значение 0 если блоки равны, и значение отличное от 0, если блоки не равны.

Void \* malloc(size\_t sizemem) – выделяет блок памяти, размером байт sizemem и возвращает указатель на начало блока. Содержание выделенного блока памяти не инициализируется, оно остается с неопределёнными значениями.

# **ОПИСАНИЕ ПРОГРАММНОЙ РЕАЛИЗАЦИИ**

**Abiturent.h**

#pragma once

#include "Date.h"

#include "Achievement.h"

class Abiturent

{

public:

int id;

char surname[30];

char name[30];

char middlename[30];

Date date;

char fac[30];

char group[30];

int curs;

Achievement discipline[10];

int disciplinepoint;

Abiturent();

static void find(int fieldNum, void\* key);

static void qsort(int direction,int direction2);

static void read(char filePath[]);

static void write();

static void add(Abiturent& abiturent);

static void remove(int index);

static void update(int index, int fieldNum, void\* key);

static void foreach();

friend ostream& operator<<(ostream& out, Abiturent& abiturent);

};

**Abiturent.cpp**

#include "Abiturent.h"

#include "Achievement.h"

#include <iostream>

#include<string>

#include <vector>

#include <iostream>

#include <algorithm>

using namespace std;

static char\* filePath;

static vector<Abiturent> items;

int fieldAsc(const void\* a, const void\* b)

{

Abiturent a\_ = \*((Abiturent\*)a);

Abiturent b\_ = \*((Abiturent\*)b);

return a\_.curs < b\_.curs ? -1 : a\_.curs == b\_.curs ? 0 : 1;

}

int fieldDesc(const void\* a, const void\* b)

{

Abiturent a\_ = \*((Abiturent\*)a);

Abiturent b\_ = \*((Abiturent\*)b);

return a\_.curs > b\_.curs ? -1 : a\_.curs == b\_.curs ? 0 : 1;

}

typedef int (\*qsortFunc)(const void\* a, const void\* b);

Abiturent::Abiturent()

{

id = -1;

strcpy(surname, "-");

strcpy(name, "-");

strcpy(middlename, "-");

date = Date();

strcpy(fac, "-");

strcpy(group, "-");

curs = 0;

disciplinePoint = 0;

}

void Abiturent::find(int fieldNum, void\* key)

{

if (fieldNum == 1) {

char\* key\_ = static\_cast<char\*>(key);

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

if (strcmp(items[i].surname, key\_) == 0)

cout << items[i] << endl;

}

}

else if (fieldNum == 2) {

char\* key\_ = static\_cast<char\*>(key);

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

if (strcmp(items[i].name, key\_) == 0)

cout << items[i] << endl;

}

}

else if (fieldNum == 3) {

char\* key\_ = static\_cast<char\*>(key);

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

if (strcmp(items[i].middlename, key\_) == 0)

cout << items[i] << endl;

}

}

else if (fieldNum == 4) {

char\* key\_ = static\_cast<char\*>(key);

Date key\_\_(key\_);

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

if (items[i].date == key\_\_)

cout << items[i] << endl;

}

}

else if (fieldNum == 5) {

int key\_ = atoi(static\_cast<char\*>(key));

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

if (items[i].ege == key\_)

cout << items[i] << endl;

}

}

else if (fieldNum == 6) {

char\* key\_ = static\_cast<char\*>(key);

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

if (strcmp(items[i].specialty, key\_) == 0)

cout << items[i] << endl;

}

}

else if (fieldNum == 7) {

char\* key\_ = static\_cast<char\*>(key);

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

for (int j = 0; j < items[i].achievementsCount; j++)

{

if (strcmp(items[i].achievements[j].title, key\_) == 0)

cout << items[i] << endl;

}

}

}

}

void Abiturent::qsort(int direction,int direction2)

{

if (direction == 8) {

}

qsortFunc qsortFuncs[2] = {

fieldAsc, fieldDesc

};

std::qsort(&items[0], items.size(), sizeof(Abiturent), qsortFuncs[--direction, direction2]);

}

void Abiturent::read(char path[])

{

filePath = path;

FILE\* f = fopen(filePath, "rb");

fseek(f, 0, SEEK\_END);

int sz = (int)((double)ftell(f) / sizeof(Abiturent));

if (sz > 0) {

items.resize(sz);

fseek(f, 0, SEEK\_SET);

fread(&items[0], sizeof(Abiturent), sz, f);

fclose(f);

}

}

void Abiturent::write()

{

FILE\* f = fopen(filePath, "w+b");

fwrite(&items[0], sizeof(Abiturent), items.size(), f);

fflush(f);

fclose(f);

}

int getNextIndex() {

int max = -1;

for (int i = 0; i < items.size(); i++)

{

if (items[i].id > max)

max = items[i].id;

}

max++;

return max;

}

void Abiturent::add(Abiturent& abiturent)

{

abiturent.id = getNextIndex();

items.push\_back(abiturent);

Abiturent::write();

}

int getRealIndex(int index)

{

int realIndex = 0;

for (size\_t i = 0; i < items.size(); i++)

{

if (items[i].id == index)

realIndex = i;

}

return realIndex;

}

void Abiturent::remove(int index)

{

items.erase(items.begin() + getRealIndex(index));

Abiturent::write();

}

void Abiturent::update(int index, int fieldNum, void\* key)

{

if (fieldNum == 1) {

char\* key\_ = static\_cast<char\*>(key);

strcpy(items[getRealIndex(index)].surname, key\_);

}

else if (fieldNum == 2) {

char\* key\_ = static\_cast<char\*>(key);

strcpy(items[getRealIndex(index)].name, key\_);

}

else if (fieldNum == 3) {

char\* key\_ = static\_cast<char\*>(key);

strcpy(items[getRealIndex(index)].middlename, key\_);

}

else if (fieldNum == 4) {

char\* key\_ = static\_cast<char\*>(key);

items[getRealIndex(index)].date = Date(key\_);

}

else if (fieldNum == 5) {

char\* key\_ = static\_cast<char\*>(key);

strcpy(items[getRealIndex(index)].secondaryschool, key\_);

}

else if (fieldNum == 6) {

char\* key\_ = static\_cast<char\*>(key);

strcpy(items[getRealIndex(index)].specialty, key\_);

}

else if (fieldNum == 7) {

int key\_ = atoi(static\_cast<char\*>(key));

items[getRealIndex(index)].ege = key\_;

}

Abiturent::write();

}

void Abiturent::foreach()

{

for (int i = 0; i < items.size(); i++)

cout << items[i] << endl;

}

ostream& operator<<(ostream& out, Abiturent& abiturent)

{

out << "{\n"

<< " ID: " << abiturent.id << "\n"

<< " Фамилия: " << abiturent.surname << "\n"

<< " Имя: " << abiturent.name << "\n"

<< " Отчество: " << abiturent.middlename << "\n"

<< " Дата поступления: " << abiturent.date.toString() << "\n"

<< " Факультет: " << abiturent.fac << "\n"

<< " Группа: " << abiturent.group << "\n"

<< " Курс: " << abiturent.curs << "\n"

<< " Список дисциплин: [\n";

for (int i = 0; i < abiturent.disciplinePoint; i++)

{

out << " {\n"

<< " Название: " << abiturent.discipline[i].name << "\n"

<< " Оценка: " << abiturent.discipline[i].point << "\n"

<< (i == abiturent.disciplinePoint - 1 ? " }\n" : " },\n");

}

out << "\n ]\n}\n";

return out;

}

**Achievement.h**

#pragma once

class Achievement

{

public:

int id;

char name[50];

int point;

};

**Date.h**

#pragma once

#include <vector>

#include <iostream>

#include <string>

using namespace std;

class Date

{

public:

int day;

int month;

int year;

Date();

Date(int day, int month, int year);

Date(const Date& date);

Date(char\* strdate);

char\* toString();

bool operator<(const Date& date) const {

if (year < date.year)

return true;

else if (year == date.year)

{

if (month < date.month)

return true;

else if (month == date.month)

{

if (day < date.day)

return true;

}

}

return false;

}

bool operator>(const Date& date) const {

if (year > date.year)

return true;

else if (year == date.year)

{

if (month > date.month)

return true;

else if (month == date.month)

{

if (day > date.day)

return true;

}

}

return false;

}

bool operator==(const Date& date) const {

return this->year == date.year &&

this->month == date.month &&

this->day == date.day;

}

Date& operator=(const Date& date)

{

day = date.day;

month = date.month;

year = date.year;

return \*this;

}

};

**Date.cpp**

#include "Date.h"

Date::Date() :

day(1), month(1), year(1970)

{}

Date::Date(int day, int month, int year) :

day(day), month(month), year(year)

{}

Date::Date(const Date& datetime) :

day(datetime.day), month(datetime.month), year(datetime.year)

{}

Date::Date(char\* strdate)

{

static const char seps[] = ".";

// format: dd.mm.yyyy

char\*\* tokens = new char\* [3];

char\* token = strtok(strdate, seps);

for (int i = 0; i < 3; i++)

{

tokens[i] = token;

token = strtok(NULL, seps);

}

this->day = stoi(tokens[0]);

this->month = stoi(tokens[1]);

this->year = stoi(tokens[2]);

}

char\* Date::toString()

{

char\* buf = (char\*)malloc(sizeof(char) \* 11);

sprintf(buf, "%02d.%02d.%04d",

this->day,

this->month,

this->year);

return buf;

}

**main.cpp**

#include "Abiturent.h"

#include <iostream>

#include<string>

using namespace std;

int main()

{

setlocale(0, "");

cout << "Введите путь к файлу: ";

char filePath[50];

cin >> filePath;

Abiturent::read(filePath);

Abiturent::foreach();

int tempInt1, tempInt2;

char tempChar = '+';

char tempStr[256];

char tempChar2 = '+';

while (true)

{

cout << "\n1. Добавление\n";

cout << "2. Удаление\n";

cout << "3. Редактирование\n";

cout << "4. Поиск\n";

cout << "5. Сортировка\n";

cout << "6. Выход\n";

cin >> tempInt1;

if (tempInt1 == 1) {

do {

Abiturent abiturent;

cout << "\nФамилия: ";

cin >> abiturent.surname;

cout << "Имя: ";

cin >> abiturent.name;

cout << "Отчество: ";

cin >> abiturent.middlename;

cout << "Дата поступления: ";

cin >> tempStr;

Date date(tempStr);

abiturent.date = date;

cout << "Факультет: ";

cin >> abiturent.fac;

cout << "Группа: ";

cin >> abiturent.group;

cout << "Текущий курс: ";

cin >> abiturent.curs;

cout << "Список дисциплин: ";

int dostCount = 0;

do {

cout << "\nНаименование предмета: ";

cin >> abiturent.discipline[dostCount].name;

cout << "оценка: ";

cin >> abiturent.discipline[dostCount].point;

dostCount++;

cout << "\n\nДобавить дисциплину? (no/yes): ";

cin >> tempChar2;

cout << "\n";

} while (tempChar2 != 'no');

abiturent.disciplinePoint = dostCount;

Abiturent::add(abiturent);

cout << abiturent;

cout << "\n\nДобавить еще студента? (no/yes): ";

cin >> tempChar;

cout << "\n";

} while (tempChar != 'no');

}

else if (tempInt1 == 2) {

cout << "\nВведите номер элемента для удаления: ";

cin >> tempInt1;

Abiturent::remove(tempInt1);

Abiturent::foreach();

}

else if (tempInt1 == 3) {

cout << "\nВведите номер студента для редактирования: ";

cin >> tempInt1;

cout << "\nВыберите поле для редактироания:\n";

cout << "1. Фамилия\n";

cout << "2. Имя\n";

cout << "3. Отчество\n";

cout << "4. Дата поступления\n";

cout << "5. Текущий курс\n";

cout << "6. Группа\n";

cout << "7. Дисциплины\n";

cin >> tempInt2;

if (tempInt2 != 8)

{

cout << "\nОшибка: ";

cin >> tempStr;

}

Abiturent::update(tempInt1, tempInt2, (void\*)tempStr);

cout << "\nРезультат редактирования:\n";

Abiturent::foreach();

}

else if (tempInt1 == 4) {

cout << "\nВыберите критерий поиска:\n";

cout << "1. Фамилия\n";

cout << "2. Имя\n";

cout << "3. Отчество\n";

cout << "4. Дата поступления\n";

cout << "5. Курс\n";

cout << "6. Группа\n";

cout << "7. Дисциплины\n";

cin >> tempInt1;

cout << "\nВведите что искать: ";

cin >> tempStr;

cout << "\nРезультат поиска:\n";

Abiturent::find(tempInt1, (void\*)tempStr);

}

else if (tempInt1 == 5) {

cout << "\nВыберите критерии сортировки:\n";

cout << "1. Фамилия\n";

cout << "2. Имя\n";

cout << "3. Отчество\n";

cout << "4. Дата поступления\n";

cout << "5. Курс\n";

cout << "6. Группа\n";

cout << "7. Дисциплины\n";

cout << "8. Средняя успеваймость\n";

cin >> tempInt1;

cout << "\nВыберите направление сортировки:\n";

cout << "1. По возрастанию\n";

cout << "2. По убыванию\n 1\n";

cin >> tempInt2;

Abiturent::qsort(tempInt1, tempInt2);

Abiturent::foreach();

}

else if (tempInt1 == 6) {

break;

}

}

return 0;

}

# **ПРОТОКОЛЫ, ПОДТВЕРЖДАЮЩИЕ РАБОТОСПОСОБНОСТЬ ПРОГРАММЫ**

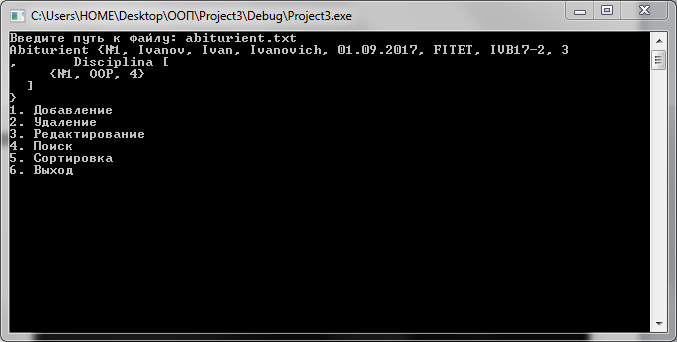


Рис.1 Начальный экран программы

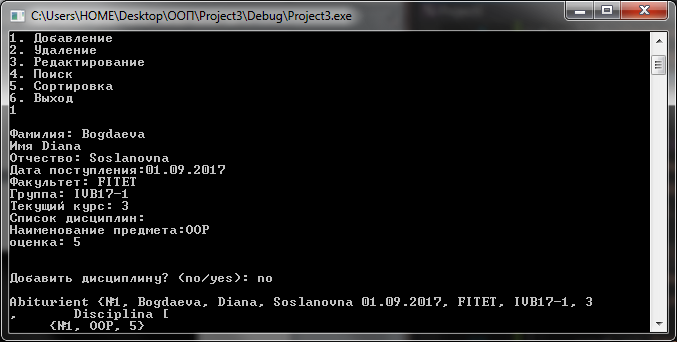


Рис.2 Добавление нового абитуриента.

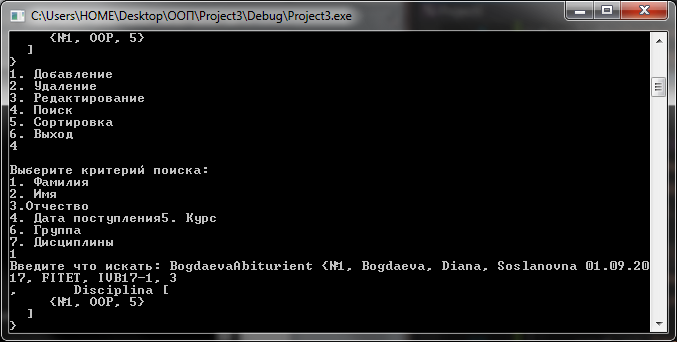


Рис.3 Поиск по фамилии

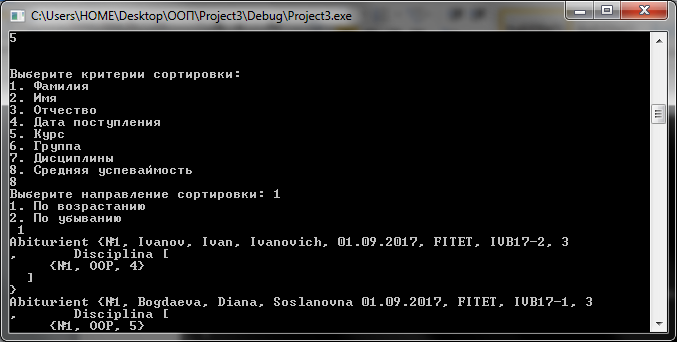


Рис.4 Сортировка средней успеваемости по возрастанию

# **ЗАКЛЮЧЕНИЕ**

В результате выполнения курсовой работы были получены результаты: создан программный продукт на интегрированной среде программирования C++.

Предоставленная программа предназначена для работы с базой данных «Абитуриент».

Свойства программы - быстрый ввод, обработка и поиск данных в данной базе.

Программа включает в себя следующие процедуры: создание новой базы данных, просмотр существующей базы данных, редактирование записей, их добавление и удаление, поиск записей и сортировка записей по данному полю.

# **СПИСОК ЛИТЕРАТУРЫ**

1. Алгоритмы и программы. Язык С++. Е. Конова, Г. Поллак, 2017

2. Объектно-ориентированное программирование в С++. Р. Лафоре, 2004

3. Программирование в C++ Builder, Архангельский А.Я., М.: Бином, 2003.

4. «Работа с базами данных в C++ Builder». А. Д. Хомоненко, С. Е. Ададуров. 2006