

Министерство образования Российской Федерации
Пензенский государственный университет
Кафедра «Вычислительная техника»

Пояснительная записка

к курсовой работе
по курсу «Программирование»
на тему «База данных по столицам»

Выполнил:

студент группы 23ВВ1

Куликов Илья Александрович

Принял:

Токарев А.Н.

Хол

13.06.24

Токарев

Пенза 2022

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет Вычислительной техники

Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"



Зав. кафедрой ВТ

М.А. Митрохин

«__» ____ 20__ г

ЗАДАНИЕ

на курсовое проектирование по курсу

Программирование

Студенту Куликову Илье Александровичу Группа 23ВВВ1

Тема проекта: *Разработка программы сложной структуры методом нисходящего программирования.*

Исходные данные (технические требования) на проектирование
*База данных студентам
Создание новой записи, добавление новой записи, вывод всех записей,
поиск по фамилии, сортировка записей, удаление записей,
изменение записей, загрузка записей из файла.*

Обязательные требования к программе:

1. Многомодульность.
2. Использование сложных типов данных – структур, списков, файлов.
3. Режим работы видеосистемы – текстовый / графический.
4. Устройство ввода информации – клавиатура, мышь.
5. Пользовательский интерфейс должен быть построен на основе меню и панели инструментов.
6. Наличие заставки.
7. Операционная система MS Windows.
8. Язык программирования – Си и Ассемблер.
9. Среда разработки ПО – Microsoft Visual Studio.

Объем работы по курсу

1. Расчетная часть

Разработка программы.

2. Графическая часть

Схема данных, схема ресурсов системы, схема работы системы,
иерархическая структура программы, схема взаимодействия программы.

3. Экспериментальная часть

Отладка программы.

Срок выполнения проекта по разделам

В соответствии с графиком.

Дата выдачи задания " 29 " февраль

Дата защиты проекта " " _____

Руководитель 

Задание получил 29 " февраль 2024г.

Студент 

Оглавление

1. Введение.....	5
2. Основная часть	7
2.1 Постановка задачи и обязательные требования.....	7
2.1.1 Многомодульность.....	7
2.1.2 Использование сложных типов данных.....	7
2.1.3 Устройство ввода информации	7
2.1.4 Пользовательский интерфейс.....	7
2.1.5 Операционная система	7
2.1.6 Язык программирования	7
2.2 Выбор решения.....	7
2.2.1 Определение необходимых модулей программы.....	7
2.2.2 Определение структуры файла базы данных.....	8
2.3 Описание разработки программы	10
2.4 Отладка и тестирование.....	11
2.5 Описание программы	12
2.5.1 Разработка приложения	12
2.5.2 Разработка функции на языке Assembler	14
2.6 Руководство пользователя	15
3. Заключение.....	18
4. Список используемых источников	19
Приложение А.....	20
Приложение А.1	20
Приложение А.2	20
Приложение А.3	22
Приложение А.4	22
Приложение А.5	30
Приложение В.....	35
Приложение С.....	Error! Bookmark not defined.

1. Введение

С развитием и совершенствованием ЭВМ тесно связано появление средств программирования. Для машин первого поколения они составлялись в машинных кодах. Это был трудоемкий процесс, поскольку программисту самому приходилось распределять память под программу, исходные данные и результаты. Разобраться в такой программе, модифицировать ее было практически невозможно. Еще в период перехода к машинам второго поколения (50-е годы) возникла необходимость в создании больших и сложных программ. Процесс их составления в машинных кодах резко снижал эффективность использования ЭВМ. Этот период характеризовался появлением первых алгоритмических языков программирования. Они отличались наглядной формой реализации алгоритма, использованием привычной математической символики, ограниченным набором ключевых слов. Основное их достоинство - универсальность.

Работа с программой, выполненной на алгоритмическом языке, очень упрощалась за счет относительной простоты написания, возможности модифицирования. Совершенствование вычислительной техники, а именно увеличение объема памяти и быстродействия машин, делало программирование на алгоритмических языках все более распространенным и перспективным. В настоящее время существует большое количество алгоритмических языков, которым присущи как общие, так и отличительные черты. Это Фортран, Бейсик, Паскаль и др. На их примере можно наглядно увидеть те характерные особенности, которые присущи программированию на алгоритмических языках вообще.

Базы данных — это совокупность структур, предназначенных для хранения больших объемов информации и программных модулей, осуществляющих управление данными, их выборку, сортировку и другие подобные действия.

Информация базы данных хранится в одной или нескольких таблицах. Любая таблица с данными состоит из набора однотипных записей,

расположенных друг за другом. Они представляют собой строки таблицы, которые можно добавлять, удалять или изменять.

Каждая запись является набором именованных полей, или ячеек, которые могут хранить самую разнообразную информацию, начиная от даты рождения и заканчивая подробным описанием кулинарного рецепта. Однотипные поля разных записей образуют столбец таблицы.

Выполнение данной курсовой работы даст мне необходимый опыт при разработке программ со сложной структурой. Создание базы данных поможет мне в понимании возможностей языка С, поможет разобраться в его сильных и слабых сторонах.

2. Основная часть

2.1 Постановка задачи и обязательные требования

Необходимо разработать программу «База данных по студентам», содержащие следующие поля: id студента, ФИО, Дата рождения, Группа, Курс, Оценки.

Также база данных должна отвечать следующим требованиям:

2.1.1 Многомодульность

Программа должна быть поделена на логические модули. Это упростит дальнейшее совершенствование программы, а также позволит локализовать ошибки.

2.1.2 Использование сложных типов данных

Программа должна иметь сложные типы данных, такие как: массивы, структур, объекты и т.д. Для разработки данной базы данных необходимо подробно изучить принципы работы с массивами, структурами.

2.1.3 Устройство ввода информации

Клавиатура/мышь. Необходимо научиться анализировать события, возникающие от работы с клавиатурой или мышью. Необходимо однозначно идентифицировать и выполнять те или иные действия в зависимости от действий пользователя. Сделать проверку на корректность.

2.1.4 Пользовательский интерфейс

Пользовательский интерфейс должен быть построен на основе меню или панели инструментов. Необходимо реализовать пользовательское меню, с заставкой.

2.1.5 Операционная система

MS Windows. При разработке использовать среду MS Windows с системой программирования MS Visual Studio.

2.1.6 Язык программирования

Для создания базы данных нужно использовать язык Си, а также язык Ассемблер для реализации ассемблерной вставки в программу.

2.2 Выбор решения

2.2.1 Определение необходимых модулей программы

Разработанная программа состоит из 4 модулей:

1. Functions.h – заголовочный файл в котором записаны прототипы функций
2. Functions.cpp – модуль в котором реализованы основные функции для работы с базой данных

3. Dataa.h – заголовочный файл, где описан class.
4. Dataa.cpp – модуль, в котором прописаны функции выше названного класса.

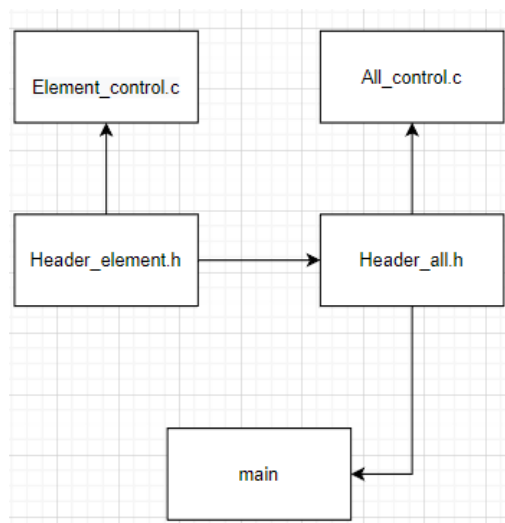


Рисунок 1 — Схема модулей программы

2.2.2 Определение структуры файла базы данных

Для структурирования данных внутри файла была выбрана следующая структура данных:

```

struct Initials {
    int id;
    string surname, name, patr;
};

struct Date {
    int day, month, year;
};

struct Group {
    string group;
    int course;
};

struct Grades {
    int math, progr, phys;
    double srball;
};

class Data {
private:
    Initials initials;
    Date date;
    Group group;
    Grades grades;
public:
    Data();
    Data(Initials initials_, Date date_, Group group_, Grades grades_);
    ~Data();

    void Print();
    void DataEntry(Initials initials_, Date date_, Group group_, Grades grades_);
    void DataEntryId(Initials initials_);

    Initials GetInitials() { return initials; };
    Date GetDate() { return date; };
    Group GetGroup() { return group; };
    Grades GetGrades() { return grades; };

    Data& operator = (Data d_o);
};
  
```

Рисунок 2 — структура данных

База данных реализована с помощью класса и его функций. Программа умеет делать следующее:

- 1) Сортировка – пользователь выбираем один из 5 видов сортировок.
- 2) Поиск и фильтрация – пользователь выбирает способ, по которому будет вестись поиск и вводит данные для поиска.
- 3) Добавление элемента – пользователь пишет данные для студента после чего он добавляется в базу данных
- 4) Удаление элемента – пользователь пишет id определенного элемента после чего данный элемент удаляется из базы данных
- 5) Редактирование элемента – пользователь выбирает определенный элемент по id после чего редактирует его данные.
- 6) Сохранение базы данных – пользователь может сохранить базу данных
- 7) Загрузка базы данных – пользователь загружает существующую базу данных

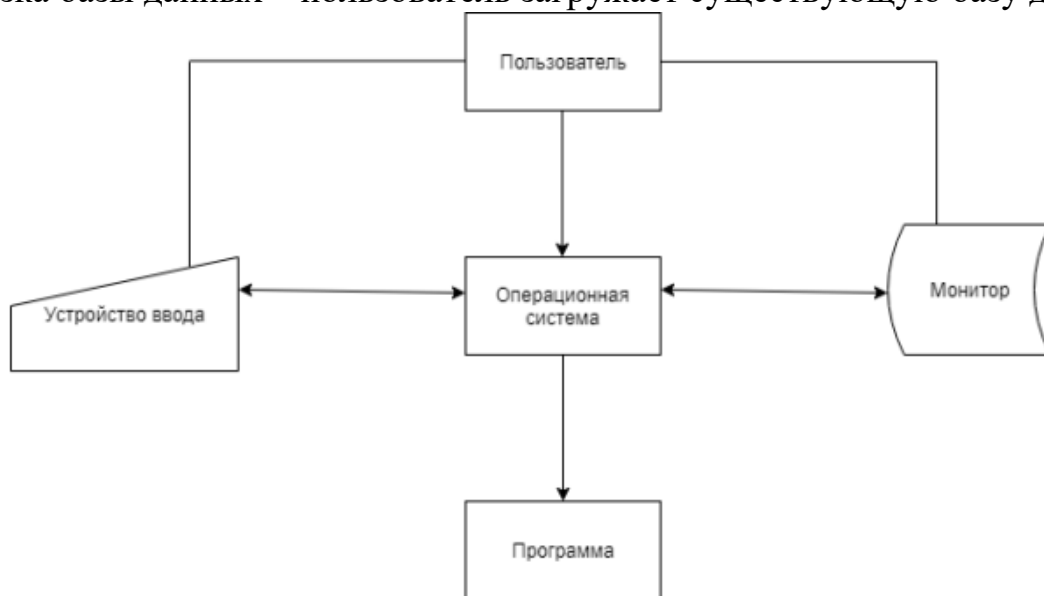


рисунок 3 - Схема данных

2.3 Описание разработки программы

Для написания данной программы будет использован язык программирования Си. Этот язык является распространённым языком программирования. При разработке языка Си был принят компромисс между низким уровнем языка ассемблера и высоким уровнем других языков. В языке Си предусмотрено много операций, непосредственно исполняемых большинством микропроцессоров; в то же время он даёт программисту возможность выражать свои алгоритмы и данные наиболее подходящими средствами. Язык Си обеспечивает возможности структурирования данных. Он позволяет разрабатывать большие, но структурированные программы, предоставляя возможность отдельной разработки подпрограмм.

В качестве среды программирования был выбран программный продукт Microsoft Visual Studio 2019.

При запуске программы на экран выводится заставка-приветствие, затем выводится и меня и предлагает выбор необходимого пункта, но если не создана или не загружена база данных, то выбор остальных функций не будет доступен. Основные действия базы данных происходят в функции main. Она отвечает за вызов функции заставки и вызов функции основного меню.

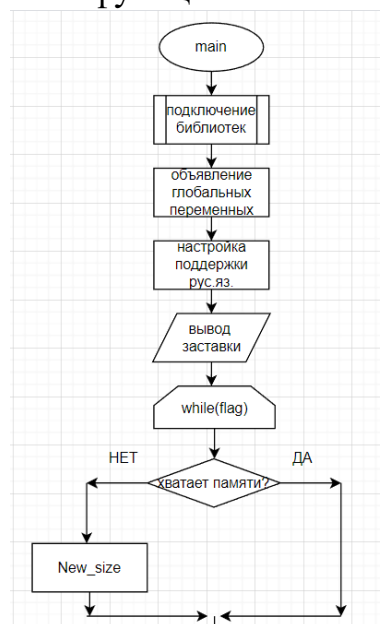


рисунок 4 - Схема данных

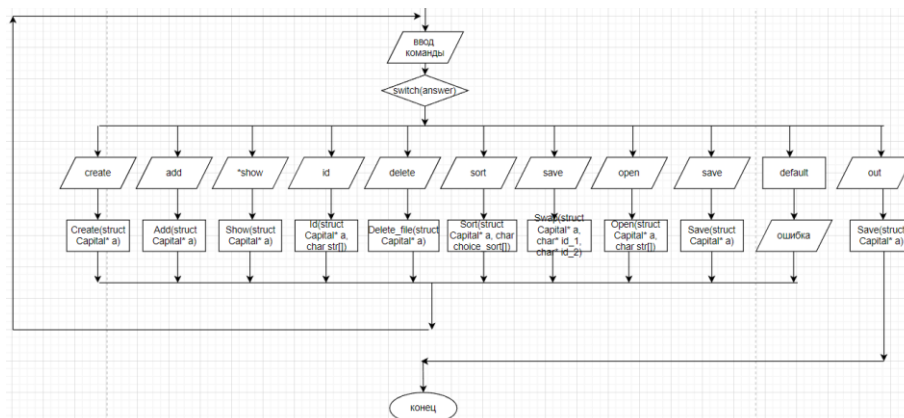


рисунок 5 – блок-схема функции main()

2.4 Отладка и тестирование

В качестве среды разработки была выбрана программа Visual Studio 2019. Программа предоставляет все средства необходимые при разработке и отладке разрабатываемых модулей и программ. Для отладки использовались такие инструменты как точка остановки, выполнение кода по шагам, трассировка, анализ содержимого локальных и глобальных переменных, анализ содержимого памяти.

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. В ходе тестирования было выявлено и исправлено множество проблем, связанных с обработкой класса, были найдены утечки памяти, системой поиска.

2.5 Описание программы

2.5.1 Разработка приложения

При запуске программы и вывода приветствия на экран программа готова к выполнению команд, введенных пользователем.

Команда	Действие программы
1 - Создание новой базы данных	Программа вызывает функцию DataEntry(d, _size)
2 - Добавление новой записи	Программа вызывает функцию AddData(d, _size)
3 - Вывод всех записей	Программа вызывает функцию Print(d, _size)
4 - Поиск по фильтру	Выводит новое меню с выбором вариантов поиска
5 - Сортировка записей	Выводит новое меню с выбором вариантов сортировки
6 - Удаление записей	Программа вызывает функцию DeleteData(d, _size)
7 - Изменение записей	Программа вызывает функцию DataChange(d, _size)
8 - Загрузка БД из файла	Программа вызывает функцию DataReading(d, _size, fileName);
9 - Сохранение БД в файл	Программа вызывает функцию SaveData(d, _size, fileName);
0 - Выход из программы	Программа завершает работу

Таблица 1 — Описание состояний программы в основном меню

При вводе цифры 4 появляется ещё одно меню:

Команда	Действие программы
1 – Поиск по ID	Программа вызывает функцию FindId(d, _size)
2 – Поиск по фамилии	Программа вызывает функцию FindSurname(d, _size)
3 – Поиск по дате рождения	Программа вызывает функцию FindDate(d, _size)
4 - Поиск по группе	Программа вызывает функцию FindGroup(d, _size)
5 - Сортировка курсу	Программа вызывает функцию FindCourse(d, _size)
0 – Вернуться в главное меню	Возвращает предыдущее меню

Таблица 1 — Описание состояний программы меню поиска

При вводе цифры 5 появляется ещё одно меню:

Команда	Действие программы
1 – Сортировка по id в обратном порядке	Программа вызывает функцию SortIdBack(d, _size)
2 – Сортировка по дате рождения	Программа вызывает функцию SortDate(d, _size)
3 - Сортировка по дате рождения в обратном порядке	Программа вызывает функцию SortDateBack(d, _size)
4 – Сортировка по успеваемости	Программа вызывает функцию SortProgress(d, _size)
5 – Сортировка по успеваемости в обратном порядке	Программа вызывает функцию SortProgressBack(d, _size)
0 – Вернуться в главное меню	Возвращает предыдущее меню

Таблица 3 — Описание состояний программы меню сортировки

На рисунке 4 показана иерархическая структура программы

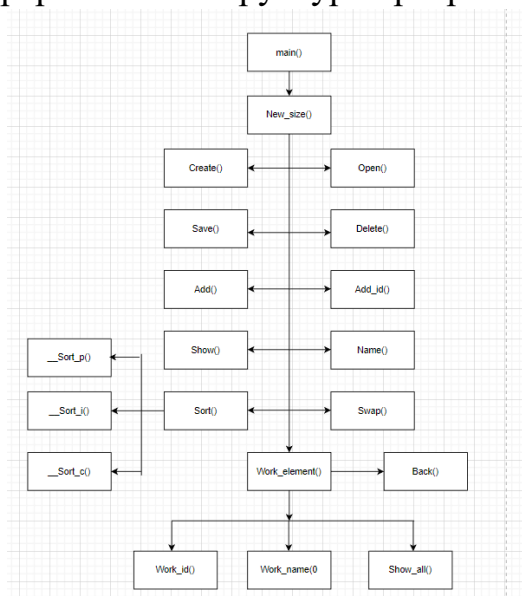


Рисунок 4 — структура данных

2.5.2 Разработка функции на языке Assembler

В своей работе я использовал ассемблер для нахождения id элемента, чтобы вывести его данные на экран.

```
Initials initials;
Date date;
Group group;
Grades grades;
int _id = 0;
int count = 0;
int srav = 0;

cout << "Введите id: ";
cin >> _id;

for (int i = 0; i < n; i++)
{
    int k = d[i].GetInitials().id;
    _asm
    {
        mov eax, k;
        mov ebx, _id;
        mov ecx, 0;

        cmp eax, ebx;
        jne End;

        result:
        mov ecx, 1;
        jmp End;

        End:
        mov srav, ecx;
    }

    if (srav == 1) {
        d[i].Print();
        count++;
    }
}
```

Рисунок 5 — вставка на assembler

В данной вставке и задаю 2 значения для сравнения, это введенный id и id, который на данный момент в классе на i месте. Затем, сравнивая их, я прыгаю на конец, если они не равны, и на указатель result, если верно. В result делаю ecx = 1, а затем в конце я перемещаю ecx в нужную переменную, а после уже снаружи вставки сравниваю, если = 1, то мы выводим данное значение.

Код функции представлен выше.

2.6 Руководство пользователя

Программа предназначена для хранения информации о студентах и их данных. Пользователь может добавлять студентов, редактировать записи уже имеющиеся, находить определенных студентов, сортировать данные студентов, удалять данные, сохранять базу данных студентов, создавать или открывать уже существующие.

Изначально мы видим список команд в нашей базе данных.

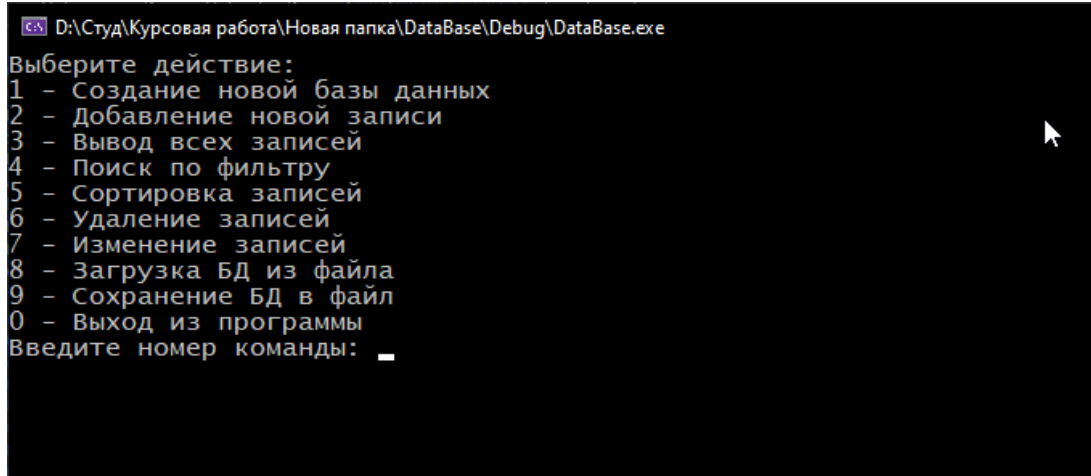


Рисунок 6 — главное меню

До ввода базы данных нельзя ничего делать.

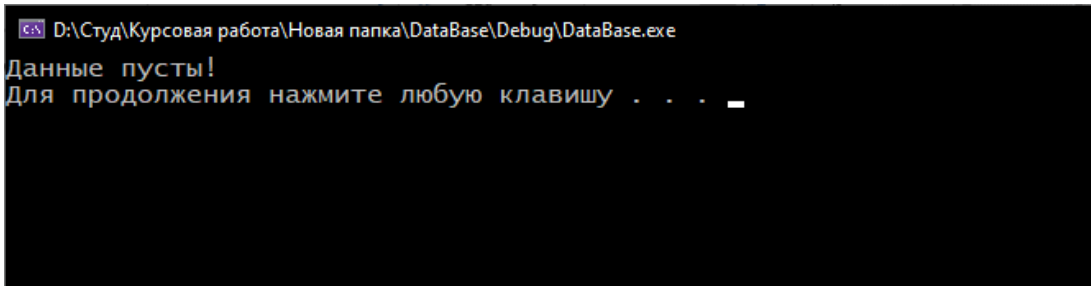


Рисунок 7 — выбор пункта без БД

Для создания новой базы данных нужно ввести 1, а затем сколько всего будет записей в ней. После этого надо будет ввести записи и всё.

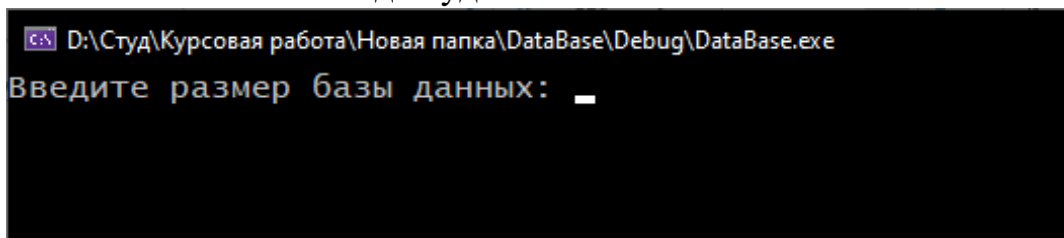


Рисунок 8 — ввод новой БД

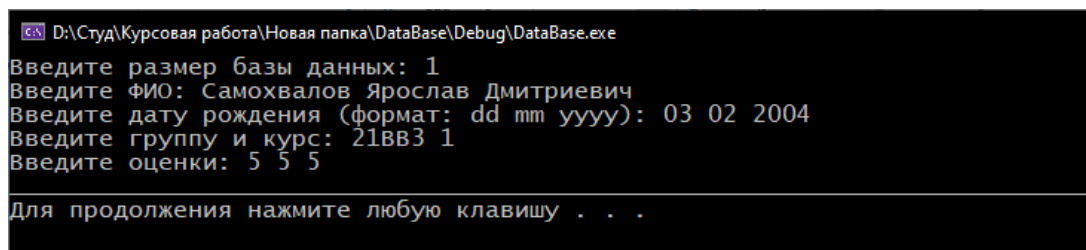


Рисунок 9 — ввод новой БД

Для ввода нового элемента на ввести 2 и ввести данные по студенту
Если пользователь хочет посмотреть все элементы бд, то надо ввести 3

```
D:\Студ\Курсовая работа\Новая папка\DataBase\Debug\DataBase.exe

Номер студента в базе данных: 1
ФИО: Головинов Владислав Русланович
Дата рождения: 10.11.2003
Учебная группа: 21ВВ3, 1 курс
Оценки:
Математика: 5
Программирование: 5
Физика: 4
Минимальный балл: 4,000000

Номер студента в базе данных: 2
ФИО: Самохвалов Ярослав Дмитриевич
Дата рождения: 3.2.2004
Учебная группа: 21ВВ3, 1 курс
Оценки:
Математика: 5
Программирование: 5
Физика: 5
Минимальный балл: 5,000000

Для продолжения нажмите любую клавишу . . .
```

Рисунок 10 — вывод всех данных

Если пользователь хочет отсортировать БД, то он может выбрать сортировку

```
D:\Студ\Курсовая работа\Новая папка\DataBase\Debug\DataBase.exe

Выберите дальнейшее действие:
1 - Сортировать по id в обратном порядке
2 - Сортировать по дате рождения
3 - Сортировать по дате рождения в обратном порядке
4 - Сортировать по успеваемости
5 - Сортировать по успеваемости в обратном порядке
0 - Вернуться в главное меню
Введите номер команды: _
```

Рисунок 11 — меню сортировки

А уже после этого при выборе способа, он увидит результат

```
D:\Студ\Курсовая работа\Новая папка\DataBase\Debug\DataBase.exe

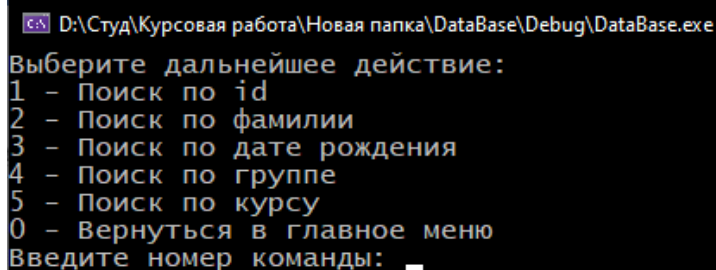
Номер студента в базе данных: 2
ФИО: Самохвалов Ярослав Дмитриевич
Дата рождения: 3.2.2004
Учебная группа: 21ВВ3, 1 курс
Оценки:
Математика: 5
Программирование: 5
Физика: 5
Минимальный балл: 5,000000

Номер студента в базе данных: 1
ФИО: Головинов Владислав Русланович
Дата рождения: 10.11.2003
Учебная группа: 21ВВ3, 1 курс
Оценки:
Математика: 5
Программирование: 5
Физика: 4
Минимальный балл: 4,000000

Для продолжения нажмите любую клавишу . . .
```

Рисунок 12 — сортировка по ID в обратном порядке

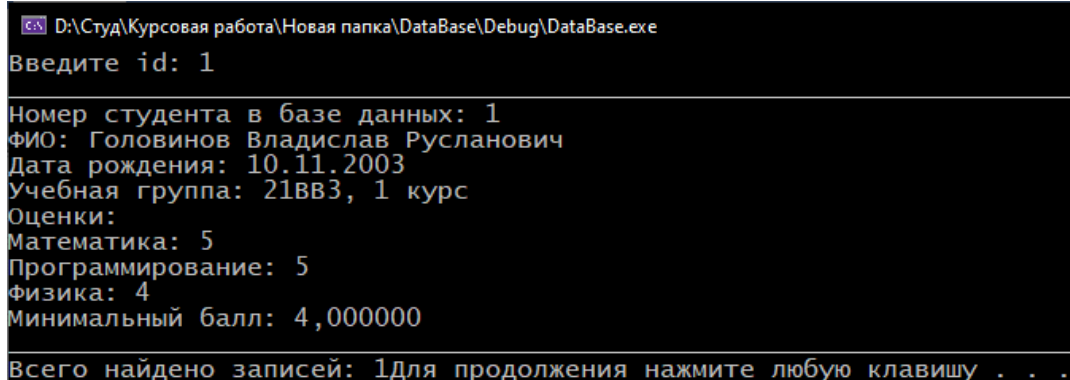
Также пользователь может искать определенные записи в БД, для этого выбираем соответствующий пункт.



```
D:\Студ\Курсовая работа\Новая папка\DataBase\Debug\DataBase.exe
Выберите дальнейшее действие:
1 - Поиск по id
2 - Поиск по фамилии
3 - Поиск по дате рождения
4 - Поиск по группе
5 - Поиск по курсу
0 - Вернуться в главное меню
Введите номер команды: _
```

Рисунок 13 — меню поиска

А после этого выбрать вариант поиска.



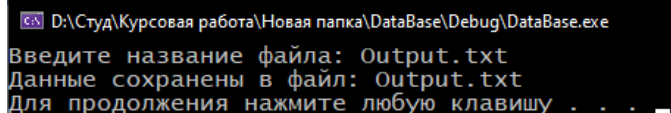
```
D:\Студ\Курсовая работа\Новая папка\DataBase\Debug\DataBase.exe
Введите id: 1

Номер студента в базе данных: 1
ФИО: Головинов Владислав Русланович
Дата рождения: 10.11.2003
Учебная группа: 21BV3, 1 курс
Оценки:
Математика: 5
Программирование: 5
Физика: 4
Минимальный балл: 4,000000

Всего найдено записей: 1Для продолжения нажмите любую клавишу . . .
```

Рисунок 14 — поиск по ID

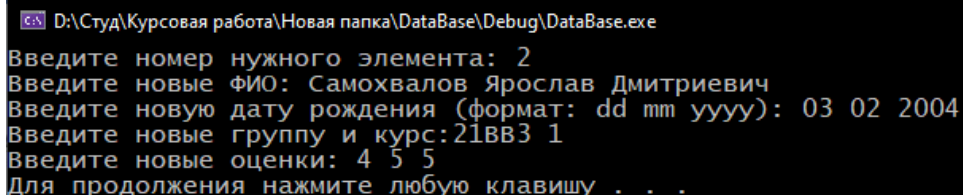
Для сохранения или загрузки БД нужно выбрать соответствующий пункт и ввести имя файла.



```
D:\Студ\Курсовая работа\Новая папка\DataBase\Debug\DataBase.exe
Введите название файла: Output.txt
Данные сохранены в файл: Output.txt
Для продолжения нажмите любую клавишу . . . _
```

Рисунок 15 — сохранение бд в файл

Для работы с определенным элементом выбираем нужный пункт и изменяем данные в записи.



```
D:\Студ\Курсовая работа\Новая папка\DataBase\Debug\DataBase.exe
Введите номер нужного элемента: 2
Введите новые ФИО: Самохвалов Ярослав Дмитриевич
Введите новую дату рождения (формат: dd mm уууу): 03 02 2004
Введите новые группу и курс: 21BV3 1
Введите новые оценки: 4 5 5
Для продолжения нажмите любую клавишу . . .
```

Рисунок 16 — редактирование записи в БД

3. Заключение

При выполнении данной курсовой работы были получены навыки разработки многомодульных программ. Получены знания и навыки по работе со структурами и классами. Получены базовые навыки программирования на языке Си и Ассемблера. Получены навыки отладки и тестирования программ.

В рамках данной курсовой была написана база данных по студентам. В рамках выполнения курсовой работы была написана база данных. Программа представляет небольшой, но достаточный список возможностей.

В дальнейшем программу можно улучшить, изменив интерфейс, добавив псевдографику, шифрование данных и отдельные аккаунты, возможность сортировать по другим параметрам.

4. Список используемых источников

- 1) О. А. Калашников - Ассемблер - это просто.
- 2) Библиотека MSDN
- 3) К.Р. Ирвин – Язык ассемблер для микропроцессоров Intel
- 4) Стивен Прата – Язык программирования С
- 5) К. Поляков. Хранение и обработка данных.
- 6) Керниган Б. Ритчи Д. Язык программирования С. 1985 г.

Приложение А

Листинги программ

Приложение А.1

Файл Dataa.h

```
#pragma once

#include "locale.h"
#include <iostream>
#include <string>
#include <fstream>
#include <Windows.h>

using namespace std;

struct Initials {
    int id;
    string surname, name, patr;
};

struct Date {
    int day, month, year;
};

struct Group {
    string group;
    int course;
};

struct Grades {
    int math, progr, phys;
    double srball;
};

class Data {
private:
    Initials initials;
    Date date;
    Group group;
    Grades grades;

public:
    Data();
    Data(Initials initials_, Date date_, Group group_, Grades grades_);
    ~Data();

    void Print();
    void DataEntry(Initials initials_, Date date_, Group group_, Grades grades_);
    void DataEntryId(Initials initials_);

    Initials GetInitials() { return initials; };
    Date GetDate() { return date; };
    Group GetGroup() { return group; };
    Grades GetGrades() { return grades; };

    Data& operator = (Data d_o);
};
```

Приложение А.2

Файл Dataa.cpp

```
#include "Dataa.h"

Data::Data()
{
    initials.id = 0;
    initials.surname = "";
    initials.name = "";
    initials.patr = "";

    date.day = 0;
    date.month = 0;
    date.year = 0;
}
```

```

        group.group = "";
        group.course = 0;

        grades.math = 0;
        grades.progr = 0;
        grades.phys = 0;
        grades.srball = 0;
    }

Data::Data(Initials initials_, Date date_, Group group_, Grades grades_)
{
    initials.id = initials_.id;
    initials.surname = initials_.surname;
    initials.name = initials_.name;
    initials.patr = initials_.patr;

    date.day = date_.day;
    date.month = date_.month;
    date.year = date_.year;

    group.group = group_.group;
    group.course = group_.course;

    grades.math = grades_.math;
    grades.progr = grades_.progr;
    grades.phys = grades_.phys;
    grades.srball = (grades_.math + grades_.progr + grades_.phys) / 3;
}

Data::~Data()
{
}

void Data::Print()
{
    SetConsoleCP(1251);
    cout <<

"
<< endl;
    cout << "Номер студента в базе данных: " << initials.id << endl;
    cout << "ФИО: " << initials.surname << " " << initials.name << " " << initials.patr << endl;
    cout << "Дата рождения: " << date.day << "." << date.month << "." << date.year << endl;
    cout << "Учебная группа: " << group.group << ", " << group.course << " курс" << endl;
    cout << "Оценки: " << endl
        << "Математика: " << grades.math << " " << endl
        << "Программирование: " << grades.progr << " " << endl
        << "Физика: " << grades.phys << " " << endl
        << "Минимальный балл: " << grades.srball << " " << endl;
}

void Data::DataEntry(Initials initials_, Date date_, Group group_, Grades grades_)
{
    initials.id = initials_.id;
    initials.surname = initials_.surname;
    initials.name = initials_.name;
    initials.patr = initials_.patr;

    date.day = date_.day;
    date.month = date_.month;
    date.year = date_.year;

    group.group = group_.group;
    group.course = group_.course;

    grades.math = grades_.math;
    grades.progr = grades_.progr;
    grades.phys = grades_.phys;
    grades.srball = (grades_.math + grades_.progr + grades_.phys) / 3;
}

void Data::DataEntryId(Initials initials_)
{
    initials.id = initials_.id;
}

Data& Data::operator=(Data d_o)
{
    this->initials.id = d_o.initials.id;
    this->initials.surname = d_o.initials.surname;

```

```

    this->initials.name = d_o.initials.name;
    this->initials.patr = d_o.initials.patr;

    this->date.day = d_o.date.day;
    this->date.month = d_o.date.month;
    this->date.year = d_o.date.year;

    this->group.group = d_o.group.group;
    this->group.course = d_o.group.course;

    this->grades.math = d_o.grades.math;
    this->grades.progr = d_o.grades.progr;
    this->grades.phys = d_o.grades.phys;
    this->grades.srball = d_o.grades.srball;

    return *this;
}

```

Приложение A.3

Файл Functions.h

```

#include "Dataa.h"

void DataEntry(Data* (&d), int& n);
void DataReading(Data* (&d), int& n, string fileName);
void Copy(Data* d_n, Data* d_o, int n);
void Print(Data* d, int n);
void DataChange(Data* d, int n);
void AddData(Data* (&d), int& n);
void DeleteData(Data* (&d), int& n);
void SaveData(Data* d, int n, string fileName);
void FindId(Data* d, int n);
void FindSurname(Data* d, int n);
void FindDate(Data* d, int n);
void FindGroup(Data* d, int n);
void FindCourse(Data* d, int n);
void SortId(Data* d, int n);
void SortIdBack(Data* d, int n);
void SortDate(Data* d, int n);
void SortDateBack(Data* d, int n);
void SortProgress(Data* d, int n);
void SortProgressBack(Data* d, int n);

```

Приложение A.4

Файл Functions.cpp

```

#pragma once
#include "Functions.h"
void DataEntry(Data* (&d), int& n)
{
    Initials initials;
    Date date;
    Group group;
    Grades grades;

    cout << "Введите размер базы данных: ";
    cin >> n;

    d = new Data[n];

    for (int i = 0; i < n; i++) {
        initials.id = i+1;
        cout << "Введите ФИО: ";
        cin >> initials.surname >> initials.name >> initials.patr;

        cout << "Введите дату рождения (формат: dd mm yyyy): ";
        cin >> date.day >> date.month >> date.year;

        cout << "Введите группу и курс: ";
        cin >> group.group >> group.course;

        cout << "Введите оценки: ";
        cin >> grades.math >> grades.progr >> grades.phys;
        grades.srball = (grades.math + grades.progr + grades.phys) / 3;

        d[i].DataEntry(initials, date, group, grades);
    }
}

```

```

        cout
    }
}
void SaveData(Data* d, int n, string fileName)
{
    ofstream record(fileName);

    if (record) {
        record << n << endl;

        for (int i = 0; i < n; i++) {
            record << d[i].GetInitials().id << endl;
            record << d[i].GetInitials().surname << " " << d[i].GetInitials().name << " " <<
d[i].GetInitials().patr << endl;
            record << d[i].GetDate().day << " " << d[i].GetDate().month << " " <<
d[i].GetDate().year << endl;
            record << d[i].GetGroup().group << " " << d[i].GetGroup().course << endl;
            record << d[i].GetGrades().math << " " << d[i].GetGrades().progr << " " <<
d[i].GetGrades().phys << endl;

            if (i < n - 1)
                record << endl;
        }
    }
    else
        cout << "Ошибка открытия файла!" << endl;

    cout << "Данные сохранены в файл: " << fileName << endl;

    record.close();
}
void DeleteData(Data* (&d), int& n)
{
    int _n;
    Data* buf = new Data[n];

    cout << "Введите номер нужного элемента: ";
    cin >> _n;
    _n--;

    if (_n >= 0 && _n < n) {
        Copy(buf, d, n);
        int q = 0;
        n--;
        d = new Data[n];

        for (int i = 0; i <= n; i++) {
            if (i != _n) {
                d[q] = buf[i];
                q++;
            }
        }

        cout << "Данные удалены!" << endl;
    }
    else
        cout << "Номер введен не верно!" << endl;

    Initials initials;

    for (int i = 0; i < n; i++)
    {
        initials.id = i+1;
        d[i].DataEntryId(initials);
    }

    delete[] buf;
}
void AddData(Data* (&d), int& n)
{
    Initials initials;
    Date date;
    Group group;

```

```

    Grades grades;

    Data* buf = new Data[n];
    int size = n, new_size = ++n;

    Copy(buf, d, size);

    d = new Data[new_size];

    Copy(d, buf, size);

    initials.id = new_size;

    cout << "Введите новые ФИО: ";
    cin >> initials.surname >> initials.name >> initials.patr;

    cout << "Введите новую дату рождения (формат: dd mm yyyy): ";
    cin >> date.day >> date.month >> date.year;

    cout << "Введите новые группы и курс: ";
    cin >> group.group >> group.course;

    cout << "Введите новые оценки: ";
    cin >> grades.math >> grades.progr >> grades.phys;
    grades.srball = (grades.math + grades.progr + grades.phys) / 3;

    d[size].DataEntry(initials, date, group, grades);

    cout << "Данные добавлены!" << endl;

    delete[] buf;
}
void Print(Data* d, int n)
{
    SortId(d, n);
    for (int i = 0; i < n; i++)
    {
        d[i].Print();
    }
    cout
}
<< endl;
}
void DataChange(Data* d, int n)
{
    Initials initials;
    Date date;
    Group group;
    Grades grades;
    int _n;

    cout << "Введите номер нужного элемента: ";
    cin >> _n;
    initials.id = _n;
    _n--;

    if (_n >= 0 && _n < n) {
        cout << "Введите новые ФИО: ";
        cin >> initials.surname >> initials.name >> initials.patr;

        cout << "Введите новую дату рождения (формат: dd mm yyyy): ";
        cin >> date.day >> date.month >> date.year;

        cout << "Введите новые группы и курс: ";
        cin >> group.group >> group.course;

        cout << "Введите новые оценки: ";
        cin >> grades.math >> grades.progr >> grades.phys;
        grades.srball = (grades.math + grades.progr + grades.phys) / 3;

        d[_n].DataEntry(initials, date, group, grades);
    }
    else
        cout << "Номер введен не верно!" << endl;
}
void DataReading(Data* (&d), int& n, string fileName)

```

```

{
    setlocale(LC_ALL, "ru");
    ifstream reading(fileName);

    if (reading)
    {
        Initials initials;
        Date date;
        Group group;
        Grades grades;

        reading >> n;

        d = new Data[n];

        for (int i = 0; i < n; i++) {
            SetConsoleCP(1251);
            reading >> initials.id >> initials.surname >> initials.name >> initials.patr;
            reading >> date.day >> date.month >> date.year;
            reading >> group.group >> group.course;
            reading >> grades.math >> grades.progr >> grades.phys;
            grades.srball = (grades.math + grades.progr + grades.phys) / 3;

            d[i].DataEntry(initials, date, group, grades);
            SetConsoleCP(1251);
        }

        cout << "Данные считаны!" << endl;
    }
    else
        cout << "Ошибка открытия файла!" << endl;
    reading.close();
}

void Copy(Data* d_n, Data* d_o, int n)
{
    for (int i = 0; i < n; i++)
        d_n[i] = d_o[i];
}

void FindId(Data* d, int n)
{
    Initials initials;
    Date date;
    Group group;
    Grades grades;
    int _id = 0;
    int count = 0;
    int srav = 0;

    cout << "Введите id: ";
    cin >> _id;

    for (int i = 0; i < n; i++)
    {
        int k = d[i].GetInitials().id;
        __asm
        {
            mov eax, k;
            mov ebx, _id;
            mov ecx, 0;

            cmp eax, ebx;
            jne End;

            result:
                mov ecx, 1;
                jmp End;

            End:
                mov srav, ecx;
        }

        if (srav == 1) {
            d[i].Print();
            count++;
        }
    }
}

```

```

        }
    }
    cout
    "
    << endl;
        cout << "Всего найдено записей: " << count;
    }
    void FindSurname(Data* d, int n)
    {
        Initials initials;
        Date date;
        Group group;
        Grades grades;
        string _surname = "";
        int count = 0;

        cout << "Введите фамилию: ";
        cin >> _surname;

        for (int i = 0; i < n; i++)
        {
            if (d[i].GetInitials().surname == _surname)
            {
                d[i].Print();
                count++;
            }
        }
        cout
        "
        << endl;
            cout << "Всего найдено записей: " << count;
        }
        void FindDate(Data* d, int n)
        {
            Initials initials;
            Date date;
            Group group;
            Grades grades;
            int _day, _month, _year;
            int count = 0;

            cout << "Введите год рождения (dd mm yyyy): ";
            cin >> _day >> _month >> _year;

            for (int i = 0; i < n; i++)
            {
                if (d[i].GetDate().day == _day && d[i].GetDate().month == _month && d[i].GetDate().year ==
                _year)
                {
                    d[i].Print();
                    count++;
                }
            }
            cout
            "
            << endl;
                cout << "Всего найдено записей: " << count;
            }
            void FindGroup(Data* d, int n)
            {
                Initials initials;
                Date date;
                Group group;
                Grades grades;
                string _group = "";
                int count = 0;

                cout << "Введите группу: ";
                cin >> _group;

                for (int i = 0; i < n; i++)
                {
                    if (d[i].GetGroup().group == _group)
                    {
                        d[i].Print();
                        count++;
                    }
                }
            }
        }
    }
}

```

```

        }
    }
    cout
    "
    << endl;
        cout << "Всего найдено записей: " << count;
    }
    void FindCourse(Data* d, int n)
    {
        Initials initials;
        Date date;
        Group group;
        Grades grades;
        int _course = 0;
        int count = 0;

        cout << "Введите курс обучения: ";
        cin >> _course;

        for (int i = 0; i < n; i++)
        {
            if (d[i].GetGroup().course == _course)
            {
                d[i].Print();
                count++;
            }
        }
        cout
        "
        << endl;
            cout << "Всего найдено записей: " << count;
        }
        void SortId(Data* d, int n)
        {
            Initials initials;
            Date date;
            Group group;
            Grades grades;
            Data* buf = new Data[n];
            Copy(buf, d, n);

            for (int i = 0; i < n; i++)
            {
                for (int j = 0; j < n - 1 - i; j++)
                {
                    if (buf[j].GetInitials().id > buf[j + 1].GetInitials().id)
                    {
                        swap(buf[j], buf[j + 1]);
                    }
                }
            }
            for (int i = 0; i < n; i++)
            {
                d[i] = buf[i];
            }

            delete[] buf;
        }
        void SortIdBack(Data* d, int n)
        {
            Initials initials;
            Date date;
            Group group;
            Grades grades;
            Data* buf = new Data[n];
            Copy(buf, d, n);

            for (int i = 0; i < n; i++)
            {
                for (int j = 0; j < n - 1 - i; j++)
                {
                    if (buf[j].GetInitials().id < buf[j + 1].GetInitials().id)
                    {
                        swap(buf[j], buf[j + 1]);
                    }
                }
            }
        }
    }
}

```

```

    }

    for (int i = 0; i < n; i++)
    {
        buf[i].Print();
    }

    delete[] buf;
}

void SortDate(Data* d, int n)
{
    Initials initials;
    Date date;
    Group group;
    Grades grades;
    Data* buf = new Data[n];
    Copy(buf, d, n);

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n - 1 - i; j++)
        {
            if (buf[j].GetDate().day < buf[j + 1].GetDate().day)
            {
                swap(buf[j], buf[j + 1]);
            }
        }
    }

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n - 1 - i; j++)
        {
            if (buf[j].GetDate().month < buf[j + 1].GetDate().month)
            {
                swap(buf[j], buf[j + 1]);
            }
        }
    }

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n - 1 - i; j++)
        {
            if (buf[j].GetDate().year < buf[j + 1].GetDate().year)
            {
                swap(buf[j], buf[j + 1]);
            }
        }
    }

    for (int i = 0; i < n; i++)
    {
        buf[i].Print();
    }

    delete[] buf;
}

void SortDateBack(Data* d, int n)
{
    Initials initials;
    Date date;
    Group group;
    Grades grades;
    Data* buf = new Data[n];
    Copy(buf, d, n);

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n - 1 - i; j++)
        {
            if (buf[j].GetDate().day > buf[j + 1].GetDate().day)
            {
                swap(buf[j], buf[j + 1]);
            }
        }
    }

```

```

        }
    }

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n - 1 - i; j++)
        {
            if (buf[j].GetDate().month > buf[j + 1].GetDate().month)
            {
                swap(buf[j], buf[j + 1]);
            }
        }
    }

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n - 1 - i; j++)
        {
            if (buf[j].GetDate().year > buf[j + 1].GetDate().year)
            {
                swap(buf[j], buf[j + 1]);
            }
        }
    }

    for (int i = 0; i < n; i++)
    {
        buf[i].Print();
    }

    delete[] buf;
}

void SortProgress(Data* d, int n)
{
    Initials initials;
    Date date;
    Group group;
    Grades grades;
    Data* buf = new Data[n];
    Copy(buf, d, n);

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n - 1 - i; j++)
        {
            if (buf[j].GetGrades().srball < buf[j + 1].GetGrades().srball)
            {
                swap(buf[j], buf[j + 1]);
            }
        }
    }

    for (int i = 0; i < n; i++)
    {
        buf[i].Print();
    }

    delete[] buf;
}

void SortProgressBack(Data* d, int n)
{
    Initials initials;
    Date date;
    Group group;
    Grades grades;
    Data* buf = new Data[n];
    Copy(buf, d, n);

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n - 1 - i; j++)
        {
            if (buf[j].GetGrades().srball > buf[j + 1].GetGrades().srball)
            {
                swap(buf[j], buf[j + 1]);
            }
        }
    }

```

```

    }
}

for (int i = 0; i < n; i++)
{
    buf[i].Print();
}

delete[] buf;
}

```

```
#pragma once
#include "Functions.h"

int _stateMenu, _act1, _act2, _act3;
void Menu()
{
    cout << "Выберите действие: " << endl
        << "1 - Создание новой базы данных" << endl
        << "2 - Добавление новой записи" << endl
        << "3 - Вывод всех записей" << endl
        << "4 - Поиск по фильтру" << endl
        << "5 - Сортировка записей" << endl
        << "6 - Удаление записей" << endl
        << "7 - Изменение записей" << endl
        << "8 - Загрузка БД из файла" << endl
        << "9 - Сохранение БД в файл" << endl
        << "0 - Выход из программы" << endl
        << "Введите номер команды: ";
    cin >> _stateMenu;
}
void Menu2()
{
    cout << "Выберите дальнейшее действие: " << endl
        << "1 - Поиск по id" << endl
        << "2 - Поиск по фамилии" << endl
        << "3 - Поиск по дате рождения" << endl
        << "4 - Поиск по группе" << endl
        << "5 - Поиск по курсу" << endl
        << "0 - Вернуться в главное меню" << endl
        << "Введите номер команды: ";
    cin >> _act2;
}
void Menu3()
{
    cout << "Выберите дальнейшее действие: " << endl
        << "1 - Сортировать по id в обратном порядке" << endl
        << "2 - Сортировать по дате рождения" << endl
        << "3 - Сортировать по дате рождения в обратном порядке" << endl
        << "4 - Сортировать по успеваемости" << endl
        << "5 - Сортировать по успеваемости в обратном порядке" << endl
        << "0 - Вернуться в главное меню" << endl
        << "Введите номер команды: ";
    cin >> _act3;
}

int main()
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    cout <<
endl
    << "|
| << endl
    << "|
| << endl
    << "|
| << endl
    << "|
| << endl
```

Курсовая работа
по дисциплине 'Программирование'
на тему 'База данных по студентам'

```

" |
endl
" << endl
" << endl
" << endl
" |
endl;

```

Выполнил: студент группы 23BV1 Куликов И.А

Принял: Токарев А.Н.

```

Sleep(5000);
system("cls");
Menu();
string fileName;

int _size = 0;
Data* d = new Data[_size];

while (_stateMenu != 0)
{
    switch (_stateMenu)
    {
        case 1:
            system("cls");

            DataEntry(d, _size);

            system("pause");
            system("cls");
            Menu();
            break;

        case 2:
            system("cls");

            if (_size != 0)
                AddData(d, _size);
            else
                cout << "Данные пусты!" << endl;

            system("pause");
            system("cls");
            Menu();
            break;

        case 3:
            system("cls");

            if (_size != 0)
                Print(d, _size);
            else
                cout << "Данные пусты!" << endl;

            system("pause");
            system("cls");
            Menu();
            break;

        case 4:
            system("cls");

            if (_size != 0)
            {
                Menu2();
                while (_act2 != 0)
                {
                    switch (_act2)
                    {
                        case 1:
                            system("cls");

                            if (_size != 0)
                                FindId(d, _size);
                            else
                                cout << "Данные пусты!" << endl;

                            system("pause");
                            system("cls");
                            Menu2();
                            break;

```

```

        case 2:
            system("cls");

            if (_size != 0)
                FindSurname(d, _size);
            else
                cout << "Данные пусты!" << endl;

            system("pause");
            system("cls");
            Menu2();
            break;
        case 3:
            system("cls");

            if (_size != 0)
                FindDate(d, _size);
            else
                cout << "Данные пусты!" << endl;

            system("pause");
            system("cls");
            Menu2();
            break;
        case 4:
            system("cls");

            if (_size != 0)
                FindGroup(d, _size);
            else
                cout << "Данные пусты!" << endl;

            system("pause");
            system("cls");
            Menu2();
            break;
        case 5:
            system("cls");

            if (_size != 0)
                FindCourse(d, _size);
            else
                cout << "Данные пусты!" << endl;

            system("pause");
            system("cls");
            Menu2();
            break;
        default:
            cout << "Пункт меню введен не верно!" << endl;
            system("cls");
            Menu2();
            break;
    }
}
else
    cout << "Данные пусты!" << endl;

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

    if (_size != 0)
    {
        Menu3();

        while (_act3 != 0)
        {
            switch (_act3)
            {
            case 1:
                system("cls");

                if (_size != 0)
                    SortIdBack(d, _size);
                else

```

```

        cout << "Данные пусты!" << endl;

        system("pause");
        system("cls");
        Menu3();
        break;
    case 2:
        system("cls");

        if (_size != 0)
            SortDate(d, _size);
        else
            cout << "Данные пусты!" << endl;

        system("pause");
        system("cls");
        Menu3();
        break;
    case 3:
        system("cls");

        if (_size != 0)
            SortDateBack(d, _size);
        else
            cout << "Данные пусты!" << endl;

        system("pause");
        system("cls");
        Menu3();
        break;
    case 4:
        system("cls");

        if (_size != 0)
            SortProgress(d, _size);
        else
            cout << "Данные пусты!" << endl;

        system("pause");
        system("cls");
        Menu3();
        break;
    case 5:
        system("cls");

        if (_size != 0)
            SortProgressBack(d, _size);
        else
            cout << "Данные пусты!" << endl;

        system("pause");
        system("cls");
        Menu3();
        break;
    default:
        cout << "Пункт меню введен не верно!" << endl;
        system("cls");
        Menu3();
        break;
    }
}
else
    cout << "Данные пусты!" << endl;

system("pause");
system("cls");
Menu();
break;
case 6:
    system("cls");

    if (_size != 0)
        DeleteData(d, _size);
    else
        cout << "Данные пусты!" << endl;

    system("pause");
    system("cls");
    Menu();

```

```

        break;
    case 7:
        system("cls");

        if (_size != 0)
            DataChange(d, _size);
        else
            cout << "Данные пусты!" << endl;

        system("pause");
        system("cls");
        Menu();
        break;
    case 8:
        system("cls");

        cout << "Введите название файла: ";
        cin >> fileName;

        DataReading(d, _size, fileName);

        system("pause");
        system("cls");
        Menu();
        break;
    case 9:
        system("cls");

        if (_size != 0) {
            cout << "Введите название файла: ";
            cin >> fileName;

            SaveData(d, _size, fileName);
        }
        else
            cout << "Данные пусты!" << endl;

        system("pause");
        system("cls");
        Menu();
        break;
    default:
        cout << "Пункт меню введен не верно!" << endl;
        system("cls");
        Menu();
        break;
    }

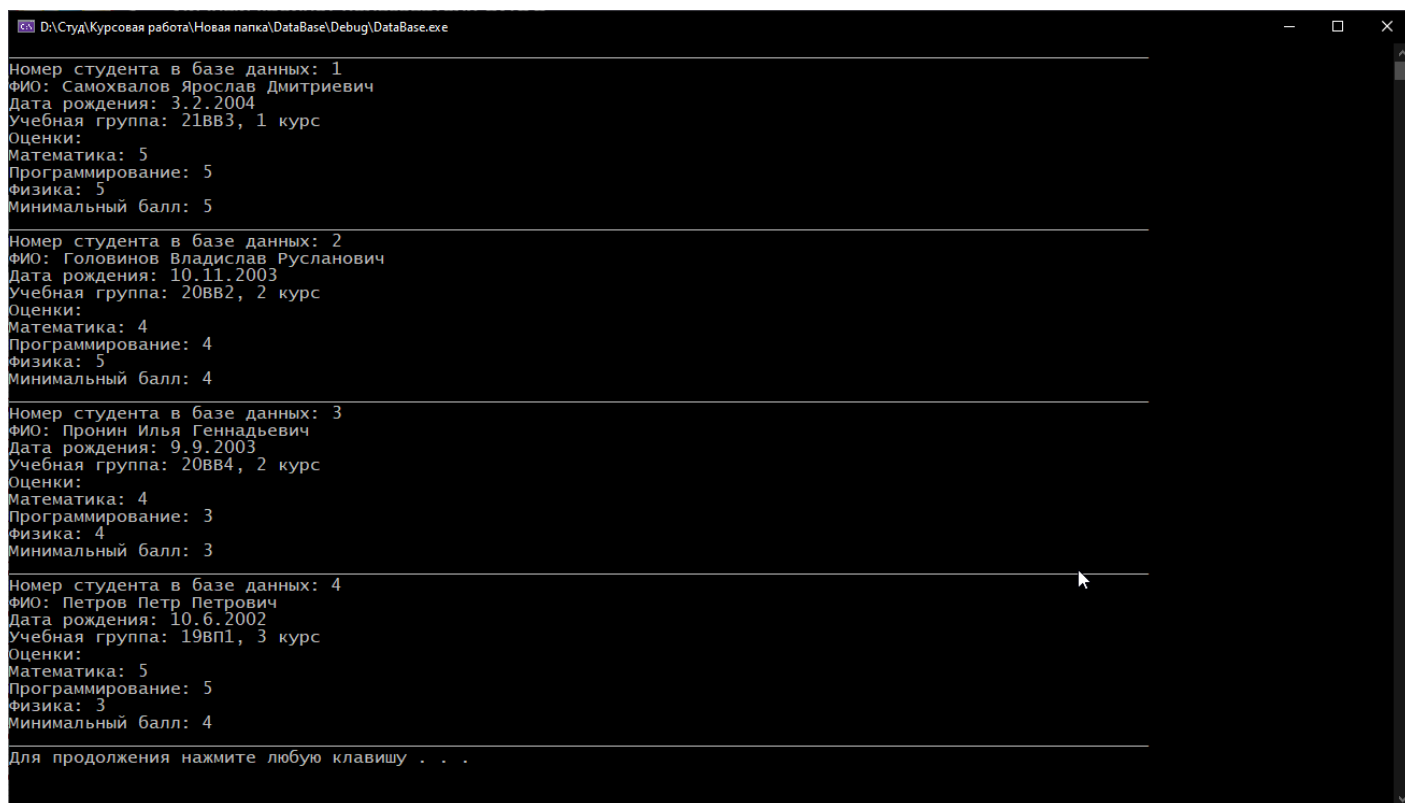
}

system("cls");
cout << "Работа завершена!" << endl;
system("pause");
}

```

Приложение В

Результат работы программы



```
D:\Студ\Курсовая работа\Новая папка\DataBase\Debug\DataBase.exe
Номер студента в базе данных: 1
ФИО: Самохвалов Ярослав Дмитриевич
Дата рождения: 3.2.2004
Учебная группа: 21BV3, 1 курс
Оценки:
Математика: 5
Программирование: 5
Физика: 5
Минимальный балл: 5

Номер студента в базе данных: 2
ФИО: Головинов Владислав Русланович
Дата рождения: 10.11.2003
Учебная группа: 20BV2, 2 курс
Оценки:
Математика: 4
Программирование: 4
Физика: 5
Минимальный балл: 4

Номер студента в базе данных: 3
ФИО: Пронин Илья Геннадьевич
Дата рождения: 9.9.2003
Учебная группа: 20BV4, 2 курс
Оценки:
Математика: 4
Программирование: 3
Физика: 4
Минимальный балл: 3

Номер студента в базе данных: 4
ФИО: Петров Петр Петрович
Дата рождения: 10.6.2002
Учебная группа: 19BP1, 3 курс
Оценки:
Математика: 5
Программирование: 5
Физика: 3
Минимальный балл: 4

Для продолжения нажмите любую клавишу . . .
```

рисунок 17 – Вывод всех элементов