МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

Институт искусственного интеллекта
Кафедра проблем управления

# КУРСОВАЯ РАБОТА

по дисциплине **Основы программирования систем управления**

**Тема курсовой работы:** «Программа управления базой данных»

**Студент группы КВБО-13-24:** Мишучков Е. В. _____

**Руководитель курсовой работы:** Новоженин М. Б. _____

Работа представлена к защите: «___» _____ 2025 г.

Допущен к защите «___» _____ 2025 г.

Москва 2025

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

Институт искусственного интеллекта
Кафедра проблем управления

**ЗАДАНИЕ**

**на выполнение курсовой работы** по дисциплине

«Основы программирования систем управления»

Утверждаю

Заведующий кафедрой ПУ

_____ Романов М.П.

Подпись

«___» _____ 2025 г.

Студент: Мишучков Е. В. Группа: КВБО-13-24

**Тема курсовой работы:** «Программа управления базой данных»

Исходные данные:

1. Язык программирования Си/Си++
2. База данных в виде файла на диске компьютера
3. Динамическое распределение и освобождение памяти
4. Структуры

**Перечень вопросов, подлежащих обработке, и обязательного графического материала:**

1. Создание базы данных в виде файла на жестком диске
2. Реализация операций доступа к базе данных в соответствии с вариантом задания
3. Блок-схема алгоритма программы

**Срок представления к защите курсовой работы: до** «___» ____ 2025 г.

**Задание на курсовую работу выдал _____ Новоженин М.Б.**

«___» _____ 2025 г.

**Задание на курсовую работу получил _____ Мишучков Е. В.**

# СОДЕРЖАНИЕ



## ВВЕДЕНИЕ

Управление данными - важная часть любого сложного технического проекта, позволяющая систематизировать информацию, упрощать ее восприятие и анализ. Управление данными присутствует везде: от анализа потребительского спроса на продукт до выдачи паспорта и внесения гражданина в государственный реестр.

Программы управления базами данных помогают облегчить сотрудникам работу с большими объемами процессов, операций, сведений и других направлений на основе общих свойств значительной части разрозненных данных, их быстрой выборке, сортировке и видоизменению.

Цель моей курсовой работы создать подобную программу, работающую со списком студентов ВУЗа.

## АНАЛИЗ ПОСТАВЛЕННОЙ ЗАДАЧИ

Программа должна быть написана на C++. Должны присутствовать изменение, сортировка и поиск элементов базы данных.

База данных сама по себе состоит из списка студентов ВУЗа, имеющих:

- ФИО
- Дату рождения
- учебную группу
- количество баллов за экзамен
- экзамены
- даты экзаменов
- результат зачета
- зачеты
- даты зачетов

Для экзаменов и зачетов удобно создать отдельные структуры, каждая из которых будет хранить название мероприятия, дату проведения и результат студента. Каждое из мероприятий будет храниться в векторе своего типа внутри области данных о студенте. ФИО, дата рождения и учебная группа — в виде отдельных переменных.

```
struct Exam // Структура одного экзамена конкретного студента
{
    int grade;
    string subject;
    string date;
};

struct Test // Структура одного зачета конкретного студента
{
    string subject;
    bool passed;
    string date;
};
```

Каждый студент будет представлять собой экземпляр класса Студенты ~ Students.

```
class Students // Класс студентов
{
public:

    string return_group() { ... }
    string return_fio() { ... }
    string return_birth() { ... }
    string return_tests(int r) { ... }
    string return_exams(int r) { ... }

    void set_fio(string s1, string s2, string s3) { ... }
    void set_group(string s) { ... }
    void set_birth(string b) { ... }
    void set_exam(int grade, string subject, string date) { ... }
    void set_test(bool passed, string subject, string date) { ... }

    int r_test() { ... }
    int r_ex() { ... }
    int passed_ex(int gr) { ... }

    int passed_tests() { ... }

private: // все переменные приватны, доступ к ним можно получить только через функции

    string name;
    string surname;
    string last_name;
    string birth;
    string group;


    Exam exams[MAX_EXAMS]; // массив всех экзаменов конкретного студента
    int exam_c = 0;

    Test tests[MAX_TESTS]; // массив всех зачетов конкретного студента
    int test_c = 0;
};
```

В классе в общей доступности будут находиться функции, а в приватной — внутренние переменные и массивы.

Все зачеты и экзамены будут храниться в массиве типа структуры заданной константной длины.

На диске должен быть создан файл с данными каждого студента.

Для удобства считывания данные в файле будут записываться по общему образцу:
*ФИО, учебная группа, дата рождения, оценка за экзамен №i, название*

*экзамена №i, дата проведения экзамена №i, результат (+/-) зачета №j, название зачета №j, дата проведения зачета №j*.

- 0 <= i <= 5
- 0 <= j <= 10
- Все данные разделяются пробелом.

*Горбунова Катя Алекс. 05.07.2006 КВБО-13-24 99 Мат.Ан. 1.07.24 + Фи. 30.06.24 + Ин.Яз. 29.06.24 + Дис.Мат. 25.06.24*

*Мишучков Егор Валерьевич 23.12.2005 КВБО-13-24 + Ин.Яз 19.09.25*

*Авельянов Илья Дмитриевич 05.06.2006 КВБО-13-24 60 Физ-ра 01.09.24*

*Левин Артем Иванович 03.05.2003 КВБО-13-24 39 Физ. 10.12.25 90 Инф. 10.11.25 + Ин.Яз. 20.11.24*

*Абаноков Идар Оглы 09.03.2005 КВБО-13-24 - Ин.Яз 19.09.25*

*Солженикина Алина Юрьевна 03.09.2006 ККСО-9-11 90 О.П. 08.08.25*

Для работы с кириллицей необходимо подключить библиотеку **<Windows.h>** и установить кодировку Windows-1251.

```
int main()
{
    SetConsoleCP(1251); // для обработки русских символов
    SetConsoleOutputCP(1251);
```

Для работы с файлами подключается библиотека **<fstream>**.

Программа должна считывать файл и создавать в динамической памяти объект каждого студента. Это будет представлено в виде вектора из экземпляров класса **Students**, элементы которого могут быть удалены, изменены и отсортированы.

Класс и структуры будут находиться в отдельном заголовочном файле.

Для работы экземпляра будут созданы функции инициализации переменных и данных структур, функции передачи внутренних данных для внешней проверки.

Программа должна иметь значительный функционал влияния на базу данных:

- Ввод имени файла с базой данных
- Ввод, корректировку и удаление информации о студенте
- Поиск информации в базе данных по заданным фамилии, имени и отчеству студента
- Поиск информации о студентах, имеющих академические задолженности
- Поиск информации о студентах, сдавших сессию
- Сортировку базы данных по фамилиям студентов в алфавитном порядке
- Вывод на экран всей информации из базы данных
- Вывод на экран найденной информации из базы данных

В качестве пользовательского интерфейса удобно использовать консольное меню, в котором введенная цифра будет означать одну из перечисленных функций.

*~~ s.txt ~~*
*0 - exit*
*2 - search for..*
*3 - add*
*4 - info*
*5 - sort*
*8 - -->file*
*Enter:*

## РАЗРАБОТКА ПРОГРАММЫ

Программа будет состоять из бесконечного цикла, ожидающего ввода 0 (окончания работы). На каждой итерации программа принимает значение от пользователя: выбор функции или ввод данных. Для функций, требующих более одной итерации главного цикла, вводятся переменные-флаги, показывающие, что операция не завершена.

В начале работы пользователь выбирает файл для считывания.

Создается объект для открытия файла

```
ifstream file_read; // объект для чтения файла
ofstream file_write; // объект для записи в файл
```

Создается вектор типа **Students**.

```
vector<Students> data2; // вектор всех экземпляров класса студентов
```

Построчно берутся данные, разделенные пробелом, добавляются в новый экземпляр, экземпляр добавляется в вектор.

```
string line;
while (getline(file_read, line)) {
    data2.push_back(set_person(line));
}
```

Функция получает строку данных о студенте. Далее в экземпляр класса передает ФИО, группу, дату рождения, экзамены и тесты.

Считанная строка передается в функцию **set_person()**, разбивается на слова. Создается новый объект класса **Students**. С помощью функций, передающих данные из строки, экземпляр устанавливает значения внутренних переменных.

```
Students set_person(string line) { // создание объекта класса Students на основе строки line, взятой из файла или терминала

    string word;
    vector<string> allWords; // вектор для хранения всех слов строки

    Students* person = new Students();
    stringstream str1(line); // разбиение строки на слова


    while (str1 >> word) {
        allWords.push_back(word); // добавление слов в вектор
    }


    // Если длина меньше 5, то есть ФИО, дня рождения и группы
    if (allWords.size() < 5) { ... }


    person->set_fio(allWords[0], allWords[1], allWords[2]);
    person->set_birth(allWords[3]);
    person->set_group(allWords[4]);

    try { // попытка пройтись по оставшимся строковым элементам с шагом 3
        for (int i = 5; i < allWords.size(); i = i + 3)
        {
            if (allWords[i] == "+" || allWords[i] == "-") {
                if (allWords[i] == "+") person->set_test(true, allWords[i + 1], allWords[i + 2]);
                if (allWords[i] == "-") person->set_test(false, allWords[i + 1], allWords[i + 2]);
            }
            else {
                int grade = stoi(allWords[i]);
                if (grade >= 1 && grade <= 100) {

                    person->set_exam(grade, allWords[i + 1], allWords[i + 2]);
                }
            }
        }
    }

    catch (...) { ... }

    allWords.clear();
    return *person; // возвращение созданного экземпляра
}
```

Пример функции **set_exam()**, получающей часть строки для ее передачи в структуру данных об экзамене.

```
void set_exam(int grade, string subject, string date)
{
    if (exam_c >= MAX_EXAMS) { // проверка, если счетчик стал больше= максимальному количеству экзаменов в массиве
        cout << "limit: exam index" << endl;
        return; // прерывание функции
    }

    exams[exam_c].date = date;
    exams[exam_c].grade = grade;
    exams[exam_c].subject = subject;


    exam_c++;
}
```

Далее идет выбор операции.
Второй идет поиск по базе данных. Он осуществляется по учебной группе, Фамилии ИО, по наличию задолжностей или сдачи сессии.

```
getline(cin, for_s);
stringstream str(for_s); // разделяем строку на слова
vector<string> s;

string name, fi, name2, f2, i2;
while (str >> name) {
    s.push_back(name);
}

cout << endl << S << endl << endl;

if (s.size() == 2) // В зависимости от длины s и ее наполения происходит поиск по ФИО, группе или зачету|задолжности
{
```

Если строка состоит из двух слов, то поиск осуществляется по Фамилии ИО.

```
if (s[0] == name2 && s[1][0] == f2[0] && s[1][1] == i2[0]) {
    cout << cc << ". ";
    cc++;
    cout << print_data(data2[i], 1) << endl;
    search.push_back(data2[i]); // Добавление в новый вектор выбранных экземпляров
    ind.push_back(i); // Добавление в новый вектор индексов выбранных экземпляров основного вектора data2
    cout << endl;
    flag1 = 1;
```

Если строка является “+” или “-”, то речь идет о сданной сессии или задолжностях.

```
else if (s[0] == "+") {
    int cc = 1;
    for (int i = 0; i < data2.size(); i++)
    {
        // если кол-во сданных экзаменов и зачетов совпадает с общим кол-вом
        if (data2[i].passed_ex(45) == data2[i].r_ex() && data2[i].passed_tests() == data2[i].r_test()) {
            cout << cc << ". ";
            cc++;
            cout << print_data(data2[i], 1) << endl;
            search.push_back(data2[i]); // Добавление в новый вектор выбранных экземпляров
            ind.push_back(i); // Добавление в новый вектор индексов выбранных экземпляров основного вектора data2
            cout << endl;

            flag1 = 1;
```

В ином случае происходит поиск по учебной группе.

При нахождении подходящего студента его экземпляр добавляется в дополнительный вектор **search**, служащий для повторного отображения появившегося списка с нумерацией для влияния на данные под выбранным номером.

|----------|

1. *Горбунова Катя Алекс. | 05.07.2006 | КВБО-13-24 | 99_Мат.Ан._1.07.24 | (+)_Фи._30.06.24 (+)_Ин.Яз._29.06.24 (+)_Дис.Мат._25.06.24*
2. *Авельянов Илья Дмитриевич | 05.06.2006 | КВБО-13-24 | 60_Физ-ра_01.09.24 |*
3. *Солженикина Алина Юрьевна | 03.09.2006 | ККСО-9-11 | 90_О.П._08.08.25 |*

|----------|

*DELETE*

*Choose number:*

Новый вектор **ind** добавляет индекс элемента начального вектора **data2**, благодаря чему пользователь в дальнейшем может заменять или удалять выведенный экземпляр.

```
cout << endl << " DELETE" << endl;
cout << " Choose number: ";
int c;
cin >> c;
if (search.size() > c - 1) {

    data2.erase(data2.begin() + ind[c - 1]); // удаление выбронного студента из основного вектора
    search.erase(search.begin() + c - 1); // удаление выбронного студента из побочного вектора
    ind.erase(ind.begin() + c - 1); // удаление индекса студента основного вектора
}
```

Для повторного удаления элемента из выведенного списка создается переменная-флаг **flag2**. Пользователю предлагается выбор продолжения работы с данным списком через ввод согласия на продолжение процесса или его прекращения.

```
cout << " Continue?" << endl;
cout << "y/n: ";

cin >> g;
if (g == 'n') {
    flag1 = 0;
    ind.clear();
    search.clear();
    flag2 = 0;
}
else if (g == 'y') // использование флага2, чтобы в новой итерации цикла снова перескочить к данному кейсу
{
    cout << " --> ok. continue" << endl;
    flag2 = 0;
}
else
{
    cout << " No answer. Retry" << endl;
    flag2 = 1;
}
```

Изменение информации о студенте происходит за счет создания нового экземпляра класса с введенными данными и замены старого. Данные вводятся сообразно с файловым представлением.

```
if (search.size() > c - 1) { // Если индекс подходит под размер вектора

    cout << print_data(search[c - 1], 2) << endl;

    cout << "Repeit with changes" << endl;
    cin.ignore();

    string k;
    getline(cin, k);
    Students New = set_person(k); // создание экземпляра класса на основе полученных измененных данных

    // проверка на существование ФИО, дня рождения и группы
    if (New.return_fio() != "" && New.return_birth() != "" && New.return_group() != "")
        data2[ind[c - 1]] = New;
}
```

Список студентов может быть отсортирован в алфавитном порядке благодаря построчному сравнению ФИО студентов из вектора экземпляров. Сортировка происходит в функции, получающей вектор **a** из всех объектов **Students**. Создается вектор **b** типа **Students**, добавляющий в себя элементы в алфавитном порядке. На каждой итерации цикла находится строка, ближайшая к началу алфавита; экземпляр добавляется в **b**, удаляется из **a**. Цикл повторяется, пока **b** не получит все элементы **a**.

```
vector<Students> alph(vector<Students> a) // функция сортировки по ФИО
{
    string s1, s2;
    vector<Students> b;
    int j;
    int c = a.size();
    while (b.size() != c) // сравнение строк, добавление в вектор b и удаление из вектора a
    {
        j = 0;
        for (int i = 1; i < a.size(); i++) {
            s1 = a[i].return_fio();
            s2 = a[j].return_fio();
            if (s1 < s2) j = i;
        }
        b.push_back(a[j]);
        a.erase(a.begin() + j);
    }

    return b;
}
```

Сохранение файла осуществляется через построчную перезапись данных всех элементов вектора **data2** типа **Students** в файл.

```
case 8: {
    if (flag1 == 1 || flag2 == 1) {
        cout << " Now impossible to save. Complete the previous operation" << endl; // Если
        flag1 = 0;
        flag2 = 0;
        break;
    }
    file_write.open(f);

    if (file_write) // проверка открытия файла
    {
        for (int i = 0; i < data2.size(); i++)
        {
            file_write << print_data(data2[i], 2) << endl; // Сохранение в файл построчно
        }
        cout << " --> Saved" << endl;
    }
```

## ВЫВОД

Мною была успешно разработана программа по управлению базой данных студентов на языке C++. Софт представляет собой консольное приложение с интерактивным вводом для пользователя, которое обладает следующими функциями:

- Чтение файла
- Поиск студентов по ФИО, учебной группе, наличию задолжностей или сданной сессии
- Добавление новых студентов в базу данных
- Вывод всей информации в консоль
- Сортировка студентов по ФИО
- Изменение данных о студенте
- Удаление информации о студенте
- Запись полученной базы данных в файл

За время выполнения курсовой работы я изучил базовый синтаксис языка C++, познакомился с работой с памятью и файлами, объектно-ориентированным программированием и модульным программированием. Спроектировал архитектуру программы. Код читаемый, с комментариями.

Все поставленные задачи реализованы. Курсовая работа выполнена в полном объеме.