

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
федеральное государственное автономное образовательное учреждение высшего образования  
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ

КУРСОВАЯ РАБОТА (ПРОЕКТ)  
ЗАЩИЩЕНА С ОЦЕНКОЙ  
РУКОВОДИТЕЛЬ

доц., канд. техн. наук  
должность, уч. степень, звание

подпись, дата

А.В. Туманова  
инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
К КУРСОВОМУ ПРОЕКТУ

РАЗРАБОТКА ПРОГРАММЫ «УЧЕТ УСПЕВАЕМОСТИ  
СТУДЕНТОВ»

по дисциплине: ОСНОВЫ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4134к

подпись, дата

И.В.Иванов  
инициалы, фамилия

Санкт-Петербург 2022

## Содержание

1. ПОСТАНОВКА ЗАДАЧИ .....	3
2. ОПИСАНИЕ СТРУКТУР ДАННЫХ .....	4
3. ОПИСАНИЕ ПРОГРАММЫ И СОЗДАННЫХ ФУНКЦИЙ .....	5
4. ОПИСАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА. ....	10
5. РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ.....	11
ЗАКЛЮЧЕНИЕ.....	20
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ.....	21
ПРИЛОЖЕНИЕ.....	22

## **1. ПОСТАНОВКА ЗАДАЧИ**

1.1. Задачей курсового проекта является разработка программы «расписание поездов» с использованием заданных структур данных, которая позволяет вводить информацию, хранить её в файле, осуществлять поиск, модификацию, сортировку и удаление данных.

1.2. Тип хранимой информации и задание на поиск приводятся в каждом варианте.

Вариант 1.

Предметная область – «Учет успеваемости студентов». Данные о студенте хранятся в структуре STUDENT, содержащей следующие поля:

- фамилия и инициалы;
- номер группы;
- успеваемость.

Задание на поиск: найти студентов, чей средний балл не меньше указанного пользователем значения.

## 2. ОПИСАНИЕ СТРУКТУР ДАННЫХ

Данные о студенте хранятся в структуре STUDENT.

```
struct STUDENT // структура данных
{
    string name; // имя и инициалы студента
    string group; // номер группы
    vector <int> marks; // массив оценок студента
};
```

Ограничения, накладываемые на выбранные типы данных:

### Имя и инициалы

Могут быть написаны на русском или на английском, начинаться с заглавной буквы и состоять только из букв. Буква, идущая после пробела (инициалы) также должна быть заглавной. Строка не должна превышать 30 символов.

### Номер группы

Формат номера: число от 0000 до 9999. Номер группы должен только из четырех цифр. Число должно быть написано слитно.

### Массив оценок

Оценки хранятся в векторе. Ввод оценок осуществляется в переменную типа string, после чего переводится в массив со значениями типа int. Формат: числа от 2 до 5.

База данных хранится на диске в виде текстового файла формата .txt.

Фамилия, инициалы и группа студента являются уникальными.

Во время работы программы данные хранятся в линейном однонаправленном списке, после выполнения программы изменения, внесенные в список, сохраняются в файл по желанию пользователя. Данные одного студента, расположенные в одной строке, разделяются символом «|».

### 3. ОПИСАНИЕ ПРОГРАММЫ И СОЗДАННЫХ ФУНКЦИЙ

Программа реализована на языке C++ в виде консольного приложения. Средой разработки является Visual Studio. В главной функции main() реализовано меню пользователя, в котором каждому действию соответствует определенная цифра. Реализованы следующие функции для работы с данными: добавление, редактирование, удаление записи, сортировка, поиск, загрузка данных из файла и сохранение в текстовый файл, вывод на экран. Описание функций основано на меню пользователя. Функции, напрямую задействованные в меню пользователя, являются основными. А все остальные функции, которые вызываются в ходе выполнения программы, являются вспомогательными.

#### Основные функции.

##### Добавление записи

```
void add_stud(Student* stud);
```

Функция выполняет добавление записи в конец списка. Перед вызовом функции вводятся данные и выполняется проверка формата введенных данных (фамилия и инициалы, номер группы, массив оценок), т.е. вызываются функции проверки. Если введены некорректные данные, пользователю предлагается ввести их повторно. Также выполняется проверка на уникальность студента. Если пользователь ввел не уникального студента, выводится соответствующее сообщение.

##### Редактирование записи

```
void edit_info();
```

Функция выполняет редактирование записи в списке. Пользователь вводит ФИО и группу студента, данные которого хочет изменить. Осуществляется проверка корректности ввода, если ввод некорректен, то пользователю предлагается снова ввести данные. Вызывается функция поиска. Если такого студента нет в списке, то выводится соответствующее сообщение. Иначе пользователю предоставляется выбор, какое поле он хочет изменить (ФИО/группа/оценки). Выбор осуществляется вводом числа (1/2/3). Если ввод некорректен, то пользователю предлагается повторно ввести число. Далее

пользователь вводит выбранное поле. Осуществляется проверка корректности введённого поля. Если поле введено некорректно, то запрашивается повторный ввод. После успешного ввода выбранное поле заменяется на введённое, остальные поля остаются без изменений.

### **Удаление записи**

```
void del_stud();
```

Функция выполняет удаление записи из списка. Пользователь вводит ФИО и группу студента, данные которого будут удалены. Осуществляется проверка на корректный ввод. Вызывается функция поиска. Если пользователь ввёл студента, которого нет в списке, будет выведено соответствующее сообщение. Иначе данные о студенте удаляются из списка.

### **Поиск записи**

```
Student* search();
```

Функция выполняет поиск записи в списке. Пользователь вводит ФИО и группу студента. Если ввод некорректен, то пользователю предлагается повторно ввести число. Далее вводится соответствующее поле. Осуществляется проверка корректности введённого поля. Если поле введено некорректно, пользователю предлагается ввести поле повторно. Осуществляется проверка наличия в списке студента с выбранным полем, если запись не найдена, то выводится соответствующее сообщение. В результате выводится строка, в которой содержатся все данные о заданном студенте.

### **Вывод на экран**

```
void show();
```

Функция выводит на экран содержимое списка в виде таблицы.

### **Сортировка**

```
void sort_group();
```

Функция выполняет сортировку списка по номеру группы в порядке убывания. Функция меняет указатель на голову и следующий элемент списка, учитывая номер группы. Номер группы с большим первым значением перемещается вверх списка.

## Поиск по заданию

```
void search_task();
```

Функция находит студентов, чей средний балл не меньше указанного пользователем значения. Пользователь вводит средний балл (число типа float). Подходящие под условия студенты хранятся в векторе. Проходимся по всем элементам списка, до тех пор, пока в списке не останется элементов. Далее выводится результат всех соответствующих студентов. Если таких студентов нет, выводится соответствующее сообщение.

## Сохранение базы данных в файл

```
void in_to_base();
```

Функция выполняет сохранение БД в текстовый файл. Перед выполнением происходит проверка на существование файла, в случае его отсутствия будет создан новый пустой файл для записи. Осуществляется проверка открытия файла, если файл открыть не удалось, то выводится соответствующее сообщение. Иначе в файл записываются данные о каждом студенте через разделяющий символ «|» между полями одного студента и через символ перевода каретки между разными студентами. Запись данных происходит до тех пор, пока не будет достигнут конец списка.

Формат записи данных в файл:

Фамилия\_И\_О|Номер\_группы|Массив\_оценок

Иванов А А|2433|54354

## Загрузить базу данных из файла

```
int out_from_base();
```

Функция выполняет выгрузку БД из текстового файла и сохранение в список. Производится проверка открытия файла, если файл открыть не удалось, то выводится соответствующее сообщение. Считывание происходит с помощью функции `getline` до тех пор, пока не будет достигнут конец файла. Номер группы переводится из `string` в `int` с помощью вызова соответствующей функции, в которой предварительно осуществляется проверка на корректность записанного в файле числа. Пустые строки в файле игнорируются.

Осуществляется проверка корректности ФИО, номера группы, их уникальности, и корректность массива оценок. Если все данные корректны, то вызывается функция добавления записи. Иначе считывается следующая строка из файла.

### **Вспомогательные функции.**

#### **Ввод имени и инициалов студента**

```
string get_name();
```

Функция запрашивает пользователя ввести фамилию и инициалы до тех пор, пока ввод не будет корректен. То есть пройдена проверка ввода ФИО, и ввод удовлетворяет ограничениям.

#### **Ввод номера группы**

```
string get_group();
```

Функция запрашивает пользователя ввести номер группы до тех пор, пока ввод не будет осуществлён корректно. То есть пройдена проверка на корректность ввода группы, и ввод удовлетворяет ограничениям.

#### **Ввод массива оценок**

```
vector <int> get_marks();
```

Функция запрашивает пользователя ввести количество оценок. Далее оценки студента до тех пор, пока ввод не будет осуществлён корректно. То есть пройдена проверка корректности массива оценок, и ввод удовлетворяет ограничениям.

#### **Проверка на корректность ввода ФИО студента**

```
bool check_correct_name(string target);
```

Функция проверяет длину строки, возвращает результат регулярного выражения, с помощью которого происходит проверка (первая заглавная буква, далее строчные буквы, пробел, заглавная буква, пробел, заглавная буква), и ложь в противном случае.

#### **Проверка на корректность ввода группы студента**

```
bool check_correct_group(string target);
```



Функция проверяет корректность строки в соответствии с ограничениями, описанными в пункте описание структур данных. Возвращает истину, если пункт назначения корректен, и ложь в противном случае.

### **Проверка на корректность ввода оценок студента**

```
bool check_form_marks(vector<int>& marks);
```

Функция проверяет корректность формата массива оценок. Возвращает истину, если оценки – это целые числа в пределах 2-5, и ложь в противном случае.

### **Перевод из строки в вектор**

```
vector<int> str_to_vec(string n);
```

Функция переводит данные типа string в вектор.

### **Проверка на длину строки**

```
bool check_length(string n, int length);
```

Функция проверяет длину строки, если она меньше length и больше нуля, возвращает истину, в противном случае ложь.

### **Проверка уникальности студента в списке**

```
bool check_uniq(Student* stud);
```

Функция просматривает весь список и возвращает истину, если ФИО и номер группы уникальный, т.е. не встречается в списке, и возвращает ложь, если не уникальны, т.е. студент с таким ФИО и номером группы уже содержится в списке.

### **Подсчет суммы оценок**

```
int sum_marks();
```

Функция проходит по всему вектору оценок и суммирует их.

### **Сравнение структур**

```
bool operator==(Student* s);
```

Функция сравнивает имя, инициалы и группу студента, если все соответствует, возвращает истину, в противном случае ложь.

#### **4. ОПИСАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА.**

При запуске программы на экран выводится консольное приложение с меню пользователя, в котором пользователь может увидеть список действий и выбрать нужный ему вариант. При нажатии на клавиатуре на определенную цифру выполняется соответствующая функция. Пользователю будет предлагаться ввод выбора варианта действий до тех пор, пока не будет выбрана команда выхода из программы. Если пользователь введёт недопустимые данные, т.е. не цифру одного из вариантов меню, то ему будет предложено заново их ввести, это будет происходить до первого корректного ввода команды. При выборе определённой команды программа обращается к соответствующей функции.

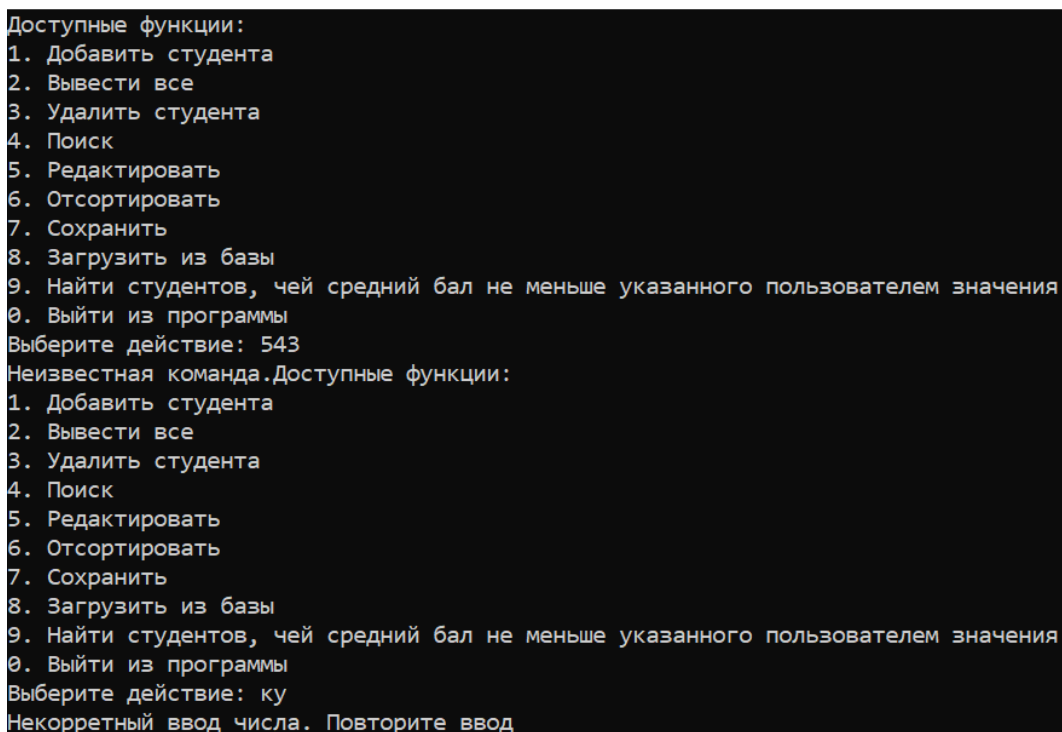
Варианты, предлагаемые пользователю:

- 1 – Добавление-add\_stud()
- 2 - Вывод на экран-show()
- 3 – Удаление-del\_stud()
- 4 - Поиск записи-search()
- 5 – Редактирование-edit\_stud()
- 6 - Сортировка по группе-sort()
- 7 - Сохранение БД в файл-in\_to\_base()
- 8 - Загрузка БД из файла-out\_from\_base()
- 9 - Поиск студентов по заданию-search\_task()
- 0 - Выход из программы

## 5. РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Результаты тестирования представлены в виде скриншотов с комментариями.

Вводим цифру, соответствующую пункту меню, чтобы выбрать действие (если же пользователь введёт некорректное значение, программа ещё раз попросит ввести число).



```
Доступные функции:
1. Добавить студента
2. Вывести все
3. Удалить студента
4. Поиск
5. Редактировать
6. Отсортировать
7. Сохранить
8. Загрузить из базы
9. Найти студентов, чей средний бал не меньше указанного пользователем значения
0. Выйти из программы
Выберите действие: 543
Неизвестная команда.Доступные функции:
1. Добавить студента
2. Вывести все
3. Удалить студента
4. Поиск
5. Редактировать
6. Отсортировать
7. Сохранить
8. Загрузить из базы
9. Найти студентов, чей средний бал не меньше указанного пользователем значения
0. Выйти из программы
Выберите действие: ку
Некорретный ввод числа. Повторите ввод
```

Рисунок 1 – представлена проверка на ввод выбора пункта меню.

Вводим цифру 1 и добавляем студента в список. Также происходит проверка на корректность всех введенных данных.

```

1
Введите фамилию и инициалы студента (в формате Фамилия И О)
звоищ вв
Неправильный формат имени.
Неккоректный ввод данных. Повторите ввод
Введите фамилию и инициалы студента (в формате Фамилия И О)
Орлова С А
Введите номер группы студента (в формате 1234)
к433
Неправильный формат группы.
Неккоректный ввод данных. Повторите ввод
Введите номер группы студента (в формате 1234)
3422
Введите кол-во вводимых оценок
4
Введите оценки студента (от 2 до 5)
3
4
в
Некорретный ввод числа. Повторите ввод
9
2
Неправильный формат оценок.
Введены некорректные значения. Повторите ввод всех оценок
4
4
3
5
Доступные функции:
1. Добавить студента
2. Вывести все
3. Удалить студента
4. Поиск
5. Редактировать
6. Отсортировать
7. Сохранить
8. Загрузить из базы
9. Найти студентов, чей средний бал не меньше указанного пользователем значения
0. Выйти из программы
Выберите действие: 2

|      Ученик      |      Группа      |      Оценки      |
|-----|-----|-----|
|      Орлова С А      |      3422      |      4435      |
|-----|-----|-----|

Доступные функции:
1. Добавить студента
2. Вывести все
3. Удалить студента
4. Поиск
5. Редактировать
6. Отсортировать
7. Сохранить
8. Загрузить из базы
9. Найти студентов, чей средний бал не меньше указанного пользователем значения
0. Выйти из программы
Выберите действие: 1
Введите фамилию и инициалы студента (в формате Фамилия И О)
Орлова С А
Введите номер группы студента (в формате 1234)
3422
Введите кол-во вводимых оценок
4
Введите оценки студента (от 2 до 5)
5
4
3
2
Данный студент уже есть

```

Рисунок 2 – демонстрируется проверка ввода ФИО, формата номера группы и формата массива оценок. Также видно проверку на уникальность студента по ФИО и номеру группы. Продемонстрирована функция добавления записи в список.

Вводим цифру 4, после вводим имя студента и номер группы. Выводится все данные о введенном студенте.

Ученик	Группа	Оценки
Орлова С А	3422	4435
Петров М П	2339	232
Белова Н Л	1453	3455
Зорин М А	5601	54555

Доступные функции:  
1. Добавить студента  
2. Вывести все  
3. Удалить студента  
4. Поиск  
5. Редактировать  
6. Отсортировать  
7. Сохранить  
8. Загрузить из базы  
9. Найти студентов, чей средний бал не меньше указанного пользователем значения  
0. Выйти из программы  
Выберите действие: 4  
Введите имя и группу студента, которого хотите найти  
Введите фамилию и инициалы студента (в формате Фамилия И О)  
Белова Н Л  
Введите номер группы студента (в формате 1234)  
1453  

Белова Н Л	1453	3455
------------	------	------

Рисунок 3 – продемонстрирована работа функции вывода списка и функции поиска по ФИО и группе.

Вводим цифру 4, после вводим имя студента и номер группы. В случае отсутствия такого студента появляется сообщение «Студент не найден».

Ученик	Группа	Оценки
Орлова С А	3422	4435
Петров М П	2339	232
Белова Н Л	1453	3455
Зорин М А	5601	54555

Доступные функции:  
1. Добавить студента  
2. Вывести все  
3. Удалить студента  
4. Поиск  
5. Редактировать  
6. Отсортировать  
7. Сохранить  
8. Загрузить из базы  
9. Найти студентов, чей средний бал не меньше указанного пользователем значения  
0. Выйти из программы  
Выберите действие: 4  
Введите имя и группу студента, которого хотите найти  
Введите фамилию и инициалы студента (в формате Фамилия И О)  
Смирнов Д Е  
Введите номер группы студента (в формате 1234)  
2311  
Студент не найден

Рисунок 4 – показана работа функции поиска при отсутствии искомого студента.

Вводим цифру 6, после перед нами появляется отсортированный список.

Ученик	Группа	Оценки
Орлова С А	3422	4435
Петров М П	2339	232
Белова Н Л	1453	3455
Зорин М А	5601	54555

Доступные функции:  
1. Добавить студента  
2. Вывести все  
3. Удалить студента  
4. Поиск  
5. Редактировать  
6. Отсортировать  
7. Сохранить  
8. Загрузить из базы  
9. Найти студентов, чей средний бал не меньше указанного пользователем значения  
0. Выйти из программы  
Выберите действие: 6  
Сортировка по группам выполнена  
Доступные функции:  
1. Добавить студента  
2. Вывести все  
3. Удалить студента  
4. Поиск  
5. Редактировать  
6. Отсортировать  
7. Сохранить  
8. Загрузить из базы  
9. Найти студентов, чей средний бал не меньше указанного пользователем значения  
0. Выйти из программы  
Выберите действие: 2

Ученик	Группа	Оценки
Зорин М А	5601	54555
Орлова С А	3422	4435
Петров М П	2339	232
Белова Н Л	1453	3455

Рисунок 5 – сортировка списка по номеру группы.

Вводим цифру 5, после вводим имя студента и номер группы. Далее можем выбрать то, что мы хотим изменить (Имя, Номер группы, Оценки)

Ученик	Группа	Оценки
Зорин М А	5601	54555
Орлова С А	3422	4435
Петров М П	2339	232
Белова Н Л	1453	3455

Доступные функции:  
1. Добавить студента  
2. Вывести все  
3. Удалить студента  
4. Поиск  
5. Редактировать  
6. Отсортировать  
7. Сохранить  
8. Загрузить из базы  
9. Найти студентов, чей средний бал не меньше указанного пользователем значения  
0. Выйти из программы  
Выберите действие: 5  
Введите имя и группу студента, данные которого хотите изменить  
Введите фамилию и инициалы студента (в формате Фамилия И О)  
Петров М П  
Введите номер группы студента (в формате 1234)  
2339  
Какую именно информацию вы хотите изменить?  
1. Имя  
2. Группа  
3. Оценки  
0. Выход  
2  
Введите номер группы студента (в формате 1234)  
4021  
Какую именно информацию вы хотите изменить?  
1. Имя  
2. Группа  
3. Оценки  
0. Выход  
3  
Введите кол-во вводимых оценок  
4  
Введите оценки студента (от 2 до 5)  
2  
3  
3  
3  
Какую именно информацию вы хотите изменить?  
1. Имя  
2. Группа  
3. Оценки  
0. Выход  
0  
Доступные функции:  
1. Добавить студента  
2. Вывести все  
3. Удалить студента  
4. Поиск  
5. Редактировать  
6. Отсортировать  
7. Сохранить  
8. Загрузить из базы  
9. Найти студентов, чей средний бал не меньше указанного пользователем значения  
0. Выйти из программы  
Выберите действие: 2

Ученик	Группа	Оценки
Зорин М А	5601	54555
Орлова С А	3422	4435
Петров М П	4021	2333
Белова Н Л	1453	3455

Рисунок 6 – демонстрируется работа функции редактирования записи.

Редактируется номер группы и оценки студента.

Вводим цифру 3, после вводим имя студента и номер группы. После чего удаляется данный студент.

Ученик	Группа	Оценки
Зорин М А	5601	54555
Орлова С А	3422	4435
Петров М П	4021	2333
Белова Н Л	1453	3455

Доступные функции:  
1. Добавить студента  
2. Вывести все  
3. Удалить студента  
4. Поиск  
5. Редактировать  
6. Отсортировать  
7. Сохранить  
8. Загрузить из базы  
9. Найти студентов, чей средний бал не меньше указанного пользователем значения  
0. Выйти из программы  
Выберите действие: 3  
Введите фамилию и инициалы студента (в формате Фамилия И О)  
Орлов Е Е  
Введите номер группы студента (в формате 1234)  
3422  
Студент не найден  
Доступные функции:  
1. Добавить студента  
2. Вывести все  
3. Удалить студента  
4. Поиск  
5. Редактировать  
6. Отсортировать  
7. Сохранить  
8. Загрузить из базы  
9. Найти студентов, чей средний бал не меньше указанного пользователем значения  
0. Выйти из программы  
Выберите действие: 3  
Введите фамилию и инициалы студента (в формате Фамилия И О)  
Орлова С А  
Введите номер группы студента (в формате 1234)  
3422  
Доступные функции:  
1. Добавить студента  
2. Вывести все  
3. Удалить студента  
4. Поиск  
5. Редактировать  
6. Отсортировать  
7. Сохранить  
8. Загрузить из базы  
9. Найти студентов, чей средний бал не меньше указанного пользователем значения  
0. Выйти из программы  
Выберите действие: 2

Ученик	Группа	Оценки
Зорин М А	5601	54555
Петров М П	4021	2333
Белова Н Л	1453	3455

Доступные функции:

Рисунок 7 – показана работа функции удаления записи.

Вводим цифру 7, тем самым происходит сохранение введенных данных в файл.



Ученик	Группа	Оценки
Галкин Ф К	4567	45
Щекин Л Р	2102	3435
Грибова М А	3547	32434

Доступные функции:

1. Добавить студента
2. Вывести все
3. Удалить студента
4. Поиск
5. Редактировать
6. Отсортировать
7. Сохранить
8. Загрузить из базы
9. Найти студентов, чей
0. Выйти из программы

Выберите действие: 7  
Список сохранён

base – Блокнот

Файл Правка Формат Вид Справка

Галкин Ф К|4567|45;  
Щекин Л Р|2102|3435;  
Грибова М А|3547|32434;|

Стр 3, стлб 24

Рисунок 8 – демонстрируется работа функции загрузки базы данных в файл. Представлен список в виде таблицы в программе и то, как он загрузился в файл.

Вводим цифру 8, тем самым выводим данные из базы(т.е. из нашего файла).

Выберите действие: 8  
Данные полученные с файла

Зорин Д Л	6210	5455
-----------	------	------

:Добавлен

Белкин К О\2222\534		
---------------------	--	--

:Неправильный формат имени.

Петрова М П	4024	2323
-------------	------	------

:Добавлен

Галкин В Л	2450	35455
------------	------	-------

:Добавлен

ЛастоваСА	4024	2323
-----------	------	------

:Неправильный формат имени.

Смирнов А П	1227	2343
-------------	------	------

:Добавлен

Сидорова П П		45
--------------	--	----

:Длина строки должна быть меньше 30 символов и больше нуля!

Орлов К Г	A402	44435
-----------	------	-------

:Неправильный формат группы.

База данных загружена  
Доступные функции:  
1. Добавить студента  
2. Вывести все  
3. Удалить студента  
4. Поиск  
5. Редактировать  
6. Отсортировать  
7. Сохранить  
8. Загрузить из базы  
9. Найти студентов, чей средний  
0. Выйти из программы  
Выберите действие: 2

file – Блокнот

Файл Правка Формат Вид Справка

Зорин Д Л|6210|5455  
Белкин К О\2222\534  
Петрова М П|4024|2323  
Галкин В Л|2450|35455  
ЛастоваСА|4024|2323  
Смирнов А П|1227|2343  
Сидорова П П||45|  
Орлов К Г|A402|44435

Стр 7, стлб 17

Ученик	Группа	Оценки
Зорин Д Л	6210	5455
Петрова М П	4024	2323
Галкин В Л	2450	35455
Смирнов А П	1227	2343

Рисунок 9 – показана работа функции загрузки базы данных из файла. В файле записаны некоторые некорректные данные. В результате работы функции в список записываются только корректные данные, а записи, не удовлетворяющие проверкам, игнорируются.

Вводим цифру 9, после чего вводим средний балл, после которого нас интересуют студенты.

```
Введите среднее значение оценок
4.3
Результат:
| Иванов И И | 3333 | 4555 | 4.75 |
|-----|-----|-----|-----|
| Антонов А А | 4444 | 44455 | 4.4 |
|-----|-----|-----|-----|
Доступные функции:
1. Добавить студента
2. Вывести все
3. Удалить
4. Поиск
5. Редактировать
6. Отсортировать
7. Сохранить
8. Загрузить из базы
9. Найти студентов, чей средний балл не меньше указанного пользователем значения
0. Выйти
```

Рисунок 10 – демонстрируется поиск студента, чей средний балл не меньше введенного значения.

## ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта были закреплены теоретические и практические знания, полученные на лекционных, лабораторных и практических занятиях по курсу, а также получены навыки разработки, отладки и тестирования программы на алгоритмическом языке программирования. Также закреплены на практике знания по работе с однонаправленными линейными списками. Разработана программа «расписание поездов» на языке C++. Она выполняет все поставленные задачи.

В программе использовались операторы динамического выделения памяти, поэтому были предприняты меры по обнаружению возможных утечек памяти.

Достоинства программы:

- в программе осуществляются различные проверки ввода данных
- программа проста и понятна в использовании
- программа работает корректно и проходит все тесты
- при наличии некорректных значения в БД, они выводятся не будут
- предусмотрено предотвращение утечек памяти

Недостатки программы:

- при редактировании и удалении из БД программа не выводит, что изменилось, нужно воспользоваться функцией вывода всех данных
- сортировка осуществляется только по номеру группы
- нет проверки на пустоту списка

## **СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. Туманова А.В. основы программирования: Методические указания к выполнению курсового проекта / СПбГУАП. СПб., 2019.
2. Ключарёв А.А., Матяш В.А., Щёкин С.В., Структуры и алгоритмы обработки данных: Учебное пособие / СПбГУАП. СПб., 2004.
3. Демидович, Е.М. Основы алгоритмизации и программирования. Язык СИ [Текст]: учебное пособие / Е. М. Демидович. - 2-е изд., испр. и доп. - СПб.: БХВ - Петербург, 2008. - 440 с.

## ПРИЛОЖЕНИЕ

### Приложение 1 (текст программы)

```
#define _CRTDBG_MAP_ALLOC //Включение учета утечек памяти
#define _CRT_SECURE_NO_WARNINGS
#include <locale>
#include <stdlib.h>
#include <crtdbg.h>
#ifdef _DEBUG
#ifdef DBG_NEW
#define DBG_NEW new ( _NORMAL_BLOCK , __FILE__ , __LINE__ )
#define newDBG_NEW
#else
#define newDBG_NEW
#endif
#endif

#include <iostream>
#include <iomanip>
#include <cctype>
#include <string>
#include <cstring>
#include <windows.h>
#include <fstream>
#include <vector>
#include <regex> //регулярные выражения

using namespace std;
bool flag = false;
vector<int> str_to_vec(string n) { //перевод из строки в вектор
    vector<int> out; //вектор который хранит вектор значений
    for (int i = 0; i < n.size(); i++) {
        out.push_back(n[i] - '0'); //занося в вектор наши числа
    }
    return out;
}

bool check_length(string n, int length) //Проверка на длину
{
    if (n.length() > length || n.length() == 0)
    {
        cout << "Длина строки должна быть меньше " << length << " символов и больше нуля!" << endl;
        return false;
    }
    return true;
}

string vec_to_string(vector<int> arr) { //перевод из вектора в строку
    string out; //вектор который хранит результат
    for (int i = 0; i < arr.size(); i++)
```

```

        out += arr[i] + '0'; //записываем в строку числа из вектора

    return out;
}

int get_num() { //ввод целого числа с проверкой ввода
    int num;
    string num1;

    do {
        std::cout << "Введите число:\n";
        std::string input;
        std::getline(std::cin, input);
        if (input.length() != 1) {
            cout << "Введите через enter\n";
            continue;
        }
        else {
            char a = input[0];
            if (isdigit(a)) {
                num = stoi(input);
                if (cin.fail()) {
                    cout << "Некорректный ввод числа. Повторите ввод\n";
                    cin.clear();
                    cin.ignore(cin.rdbuf()->in_avail());
                }
                else
                    break;
            }
        }
    } while (true);

    return num;
}

struct Student { //структура данных
    string name; //имя студента
    string group; //группа студента
    vector<int> marks; //массив оценок студента
    double tempAver;
    Student(string Name, string Group, vector<int> Marks) { //конструктор структуры
        name = Name;
        group = Group;
        marks = Marks;
    }
}

int sum_marks() { //посчитать сумму всех оценок
    int res = 0;
    for (int i = 0; i < marks.size(); i++) //суммируем все оценки
        res += marks[i];
    return res;
}

```

```

    }
    friend ostream& operator<<(ostream& out, Student* stud) { //функция вывода данных
структуры //перегрузка оператора вывода

        if (flag == false) {
            cout << "| " << setw(13) << stud->name << " | " << setw(14) << stud->group << " | "
                << setw(14) << vec_to_string(stud->marks) << " |" << endl;
            cout << "|-----|-----|-----|" << endl;
        }
        else {
            cout << "| " << setw(13) << stud->name << " | " << setw(14) << stud->group << " | "
                << setw(14) << vec_to_string(stud->marks) << " |" << setw(14) << stud->tempAver << "
| " << endl;
            cout << "|-----|-----|-----|-----|" << endl;
        }

        return out;
    }
};
class StudsData { //класс для хранения и обработки списка структур
private:
    struct Node { //структура узла
        Student* stud = nullptr; //указатель на структуру
        Node* next = nullptr; //указатель на следующий элемент списка
    };

    Node* head; //указатель на начальный элемент
    int count; //кол-во элементов в списке

    bool check_correct_name(string target) { //проверка на корректность ввода имени студента
        if (!check_length(target, 30)) //проверяем длину
            return false;

        bool res;
        bool res2;
        regex rx(R"(([A-ZА-Я]{1})([a-za-я]+)\s([A-ZА-Я]+)\s([A-ZА-Я]+))"); //создаем
регулярное выражение
        regex rx2(R"(([A-ZА-Яa-za-я]{1})([a-za-я]+)\s([A-ZА-Яa-za-я]+)\s([A-ZА-Яa-za-я]+))");

        res = regex_match(target.cbegin(), target.cend(), rx); //проверяем совпадает ли результат
рег. выражения со всей строкой
        res2 = regex_match(target.cbegin(), target.cend(), rx2);
        if (!res && !res2) {
            cout << "Неправильный формат данных имени.\n";
            return 0;
        }
        if (res) {
            return res;
        }
        else {
            return res2;

```



```

    }
}

bool check_correct_group(string target) { //проверка на корректность ввода группы
студента
    if (!check_length(target, 30)) //проверка на длину
        return false;

    bool res;

    regex rx(R"(([d]{4}))"); //создаем регулярное выражение //

    res = regex_match(target.cbegin(), target.cend(), rx); //проверяем совпадает ли результат
рег. выражения со всей строкой

    if (!res)
        cout << "Неправильный формат данных группы.\n";

    return res;
}

bool check_form_marks(vector<int>& marks) { //проверка на корректность ввода оценок
студента
    static int left = 2; //минимальное число оценки
    static int right = 5; //максимальное число оценки

    int temp;
    for (int i = 0; i < marks.size(); i++) {
        if (marks[i] > 5 || marks[i] < 2) { //сравниваем с мин и макс числами
            cout << "Неправильный формат данных оценок\n";
            return false;
        }
    }
    return true;
}

bool check_uniq(Student* stud) { //проверка на уникальность элемента в списке
    Node* temp = head;
    Student* s;
    for (int i = 0; i < count; i++) { //проходимся по всем элементам списка
        s = temp->stud;
        if (s->name == stud->name && s->group == stud->group) //сравниваем уникальность
на имя и группу
            return false;
        temp = temp->next;
    }
    return true;
}

string get_name() { //функция для получения имени студента с проверкой ввода
    string name; //переменная хранящая результат
    bool res;

```

```

regex rx(R"(([A-ZА-Яа-za-я]{1})([a-za-я]+)\s([A-ZА-Яа-za-я]+)\s([A-ZА-Яа-za-я]+))");
while (true) {
    std::cout << "Введите фамилию и инициалы студента (в формате Фамилия А А)\n";
    cin.clear();//очищаем флаги и буфер
    cin.ignore(cin.rdbuf()->in_avail());
    getline(std::cin, name);
    if (check_correct_name(name)) {
        res= regex_match(name.cbegin(), name.cend(), rx);
        for (int i = 0; i < name.size(); i++) {
            if (name[i] == ' ') {
                name[i + 1] = toupper(name[i + 1]);
            }
        }
        if (res) {
            name[0] = toupper(name[0]);
        }
        break;
    }
    else
        cout << "Некорректный ввод данных. Повторите ввод\n";
}

return name;
}

string get_group() { //функция для получения группы студента с проверкой ввода

    string group;//переменная хранящая результат
    while (true) {
        std::cout << "Введите номер группы студенты (в формате 1234)\n";
        cin.clear();//очищаем флаги и буфер
        cin.ignore(cin.rdbuf()->in_avail());
        getline(cin, group);
        if (check_correct_group(group))//проверка на корректность ввода
            break;
        else
            cout << "Некорректный ввод данных. Повторите ввод\n";
    }

    return group;
}

vector<int> get_marks() { //функция для получения оценок студента с проверкой ввода

    int size;//кол-во оценок
    std::cout << "Введите кол-во вводимых оценок\n";
    bool correct = 0;//флаг означающий корректность ввода массива оценок
    size = get_num();//получаем число с проверкой ввода

    cout << "Введите оценки студента (от 2 до 5)\n";
    vector<int> marks;

```

```

marks.resize(size); //изменяем размер вектора, чтобы можно было добавлять данные с
помощью индекса
correct = 0;
while (!correct) {
    cin.clear(); //очищаем флаги и буфер
    cin.ignore(cin.rdbuf()->in_avail());
    for (int i = 0; i < size; i++) {
        marks[i] = get_num(); //получаем число с проверкой ввода
        bool flag = true;
        if (marks[i] > 5 || marks[i] < 2) {
            flag = false;
            cout << "неверно. Введите оценки студента (от 2 до 5) ";
            while (!flag) {
                marks[i] = get_num();
                if (marks[i] < 5 || marks[i] > 2) {
                    flag = true;
                }
            }
        }

        continue;
    }

    //if (check_form_marks(marks)) //проверка корректности массива оценок
    // correct = 1;
    //else
    // cout << "Введены некорректные значения. Повторите ввод всех оценок\n";
    break;
}

return marks;
}

public:
StudsData() { //конструктор класса для хранения списка
    head = nullptr;
    count = 0;
}

Student* input() { //функция для ввода данных структуры

    string name = get_name(); //получаем имя с проверкой ввода

    string group = get_group(); //получаем группу с проверкой ввода

    vector<int> marks = get_marks(); //получаем оценки с проверкой ввода

    return new Student(name, group, marks);
}

```

```

void add_stud(Student* stud) { //функция для добавления студента в список

    if (!check_uniq(stud)) { //проверка на уникальность студента
        cout << "Данный студент уже есть\n";
        delete stud;
        return;
    }

    Node* node = new Node; //создаем новый узел и в него заносим нашу структуру
    node->stud = stud;

    if (head == nullptr) { //если нет элементов в списке
        head = node;
    }
    else { //если в списке 1 и более элемент
        Node* temp = head;
        for (int i = 0; i < count - 1; i++) { //находим самый последний элемент
            temp = temp->next;
        }
        temp->next = node; //заносим в next последнего элемента новый узел
    }
    count++; //увеличиваем кол-во элементов
}

void del_stud() { //функция для удаления студента из списка
    if (count == 0) {
        cout << "Список пуст" << endl;
        return;
    }
    Student* stud = search(); //ищем студента в списке

    if (stud == nullptr || count == 0) { //если студент не был найден или список пуст
        cout << "Студент не найден\n";
        return;
    }

    if (count == 1) { //если в списке всего один студент
        delete head;
        head = nullptr;
        count--;
    }
    else { //если в списке 1 и более элементов
        Node* prev = nullptr; //указатель хранящий пройденный элемент списка
        Node* temp = head; //устанавливаем указатель на начало списка
        for (int i = 0; i < count; i++) { //проходимся по списку
            if (stud == temp->stud) { //если нашли совпадение
                if (temp == head) { //если элемент явл-ся первым в списке
                    head = head->next;
                }
                else { //если эл-т не явл-ся началом списка
                    prev->next = temp->next;
                }
            }
        }
    }
}

```

```

        count--; //уменьшаем кол-во элементов
        cout << "Студент удален\n";
        delete temp->stud; //освобождаем память
        delete temp;
        break;
    }
    prev = temp; //сохраняем пройденный элемент
    temp = temp->next; //переставляем унк-ль на след элемент
}
}
}

```

Student\* search() { //функция для поиска студента в списке

```

    if (count == 0) {
        cout << "Список пуст" << endl;
        return 0;
    }
    string name;
    vector<int> arr;

```

```

    Node* temp = head;
    Student* w;

```

```

    int ks{ };

```

```

    cout << ("Введите фамилию \n\n");

```

```

    cin.clear(); //очищаем флаги и буфер
    cin.ignore(cin.rdbuf()->in_avail());
    getline(std::cin, name);

```

```

    w = temp->stud;
    name[0] = toupper(name[0]);
    int it = 1;
    while (name[it] != '\0')
    {
        if (name[it] == ' ')
        {
            name[++it] = toupper(name[it]);
        }
        else
        {
            it++;
        }
    }
    if (name.size() <= w->name.size() && name.size() >= 3 && !(cin.fail()))
    {
        for (int i = 0; i < count; i++)
        {
            int j{ };
            int k{ };

```

```

int rr{};
w = temp->stud;
while (true)
{
    if (name.size() < j)
    {
        break;
    }
    else if (rr == name.size() / 3)
    {
        break;
    }
    if (name[j] == w->name[k])
    {
        j++;
        k++;
    }
    else
    {
        rr++;
        k++;
        j++;
    }
    if (k == name.size() && rr == 0)
    {
        std::cout << i << " | " << w->name << " | " << w->group << " | " << endl;
        arr.push_back(i);
        ks++;
        break;
    }
    else if (j == name.size() && rr != 0)
    {
        std::cout << i << " | " << w->name << " | " << w->group << " | " << endl;
        arr.push_back(i);
        ks++;

        break;
    }
}
temp = temp->next;
}
}
else
{
    std::cout << "введите символы больше 3 штук\n";
    return nullptr;
}
}
if (ks == 0) {
    return nullptr;
}
}
int i;
bool fl = false;

```

```

while (true)
{
    cout <<"выберите номер студента " << endl;
    i= get_num();
    for (int ki = 0; ki < arr.size(); ki++)
    {
        if (i == arr[ki])
            fl = true;
    }
    if (!fl)
        std::cout << "данной опции не существует\n\n";
    else
        break;
}
temp = head;
for (int j = 0; j <= i; j++) {
    w = temp->stud;

    if (j == i) {
        return w;
    }
    temp = temp->next;
}
}

void edit_info() { //функция для редактирования данных о студенте в списке
    if (count == 0) {
        cout << "Список пуст" << endl;
        return;
    }
    cout << "Введите имя и группу студента, данные которого хотите изменить\n";
    Student* stud = search(); //ищем студента в списке

    if (stud == nullptr) { //если не нашли студента
        cout << "Студент не найден\n";
        return;
    }
    string temp1;
    Node* ku = head;

    int choice = -1; //номер выбора
    string temp; //строковая переменная для хранения полученных данных
    vector<int> vec; //массив для хранения оценок
    int k = 0;
    while (choice != 0) {
        Student s = *stud; //переменная для хранения структуры
        cout << "Какую именно информацию вы хотите изменить?\n";
        cout << "1. Имя\n2. Группа\n3. Оценки\n0. Выход\n";
        cin >> choice;
        switch (choice) {
            case 0:
                break;

```

```

case 1:
    temp = get_name();
    //изменяем имя студента на новое
    temp1 = stud->name;
    stud->name = temp;
    //изменяем группу студента на новую
    Student* s;
    for (int i = 0; i < count; i++) { //проходимся по всем элементам списка
        s = ku->stud;
        if (s->name == stud->name && s->group == stud->group) { //сравниваем
уникальность на имя и группу

            k++;
        }
        ku = ku->next;

    }
    if (k != 1) {
        cout << "имя не было изменено, данный студент уже есть" << endl;
        stud->name = temp1;
    }
    k = 0;
    ku = head;
    break;
case 2:
    temp = get_group();

    temp1 = stud->group;
    stud->group = temp;
    //изменяем группу студента на новую

    for (int i = 0; i < count; i++) { //проходимся по всем элементам списка
        s = ku->stud;
        if (s->name == stud->name && s->group == stud->group) { //сравниваем
уникальность на имя и группу

            k++;
        }
        ku = ku->next;

    }
    if (k != 1) {
        cout << "группа не была изменена, данный студент уже есть" << endl;
        stud->group = temp1;
    }
    k = 0;
    ku = head;

    break;
case 3:
    vec = get_marks();
    stud->marks = vec; //изменяем оценки студента на новые

```



```

        break;
    default:
        cout << "Неизвестная команда. Повторите ввод\n";

    }

}

}

}

void sort_group() { //сортировка студентов в списке по группам
    Student* temp, * next;
    if (count == 0) {
        cout << "Список пуст" << endl;
        return;
    }
    for (int i = 0; i < count; i++) { //сортируем студентов в массиве по группам
        Node* t = head;
        for (int j = 0; j < count - i - 1; j++) {
            temp = t->stud;
            next = t->next->stud;
            if (temp->group < next->group) { //сравниваем номер группы
                Student s = *temp;
                *temp = *next;
                *next = s;
            }
            t = t->next;
        }
    }
    cout << "Сортировка по группам выполнена\n";
}

void in_to_base() { //загрузка базы данных в файл
    if (count == 0) {
        cout << "Список пуст" << endl;
        return ;
    }
    ofstream base("base.txt"); //создаем файл
    if (!base.is_open()) //проверка на возможность открыть файл для записи
    {
        cout << "Не удалось записать в файл\n";
        return;
    }

    else
    {
        base.clear(); //чистим буфер от ошибок

        Node* temp = head;
        for (int i = 0; i < count; i++) //проходимся по списку
        {

```

```

        base << temp->stud->name << "|" << temp->stud->group << "|" <<
vec_to_string(temp->stud->marks) << ";" << endl; //записываем
        temp = temp->next; //идем дальше
    }
}
cout << "Список сохранён" << endl;
base.close(); //закрываем файл
}

```

```

void out_from_base() { //загружаем данные из файла в нашу базу данных
    ifstream inf("base.txt");
    int ind = 0; //индекс для строковой переменной
    string str; //строковая переменная, куда мы будем сохранять строки
    string name, group; //переменные, куда будем записывать имя и группу
    vector<int> marks; //массив, куда будем записывать оценки
    Student* s; //указатель на структуру

    string temp; //вспомогательная переменная для хранения данных

    if (!inf.is_open()) //проверка на возможность открыть файл для записи
    {
        cout << "Не удалось открыть файл!" << endl;
        return;
    }

    if (inf.peek() == EOF) { //проверка на пустоту файла
        cout << "Файл пуст!\n";
        return;
    }

    cout << "Данные полученные с файла\n";
    while (!inf.eof()) //пока не конец файла
    {
        getline(inf, str); //получаем одну строку из файла

        if (str.size() == 0)
            continue;

        while (ind < str.size() && str[ind] != '|') //до знака | записываем имя студента
        {
            temp += str[ind]; //записываем посимвольно в нашу переменную
            ind++;
        }
        name = temp; //вносим полученное значение в переменную имени

        ind++;
        temp.clear(); //очищаем переменную от старых данных для записи новых
        while (ind < str.size() && str[ind] != '|') //до знака | записываем группы студента
        {
            temp += str[ind]; //записываем посимвольно в нашу переменную
            ind++;
        }
    }
}

```

```

    }
    group = temp;//заносим полученное значение в переменную группы

    ind++;

    temp.clear();//очищаем переменную от старых данных для записи новых
    while (ind < str.size())//до enter записываем оценки студента
    {
        char op;
        op = str[ind];
        bool flag=true;
        if (op == '2' || op == '3' || op == '4' || op == '5') {

            temp += str[ind];//записываем посимвольно в нашу переменную
            ind++;
        }
        else {
            ind++; }
    }

    marks = str_to_vec(temp);//заносим полученное значение в переменную имени с
    преобразованием строки в массив
    s = new Student(name, group, marks);
    cout << s << ":";

    if (check_correct_name(name) && check_correct_group(group) && flag==false) { //если
    полученные данные корректны, то добавляем в список
        cout << "Добавлен\n";
        add_stud(s);
    }
    else
        delete s;
    cout << "\n\n";

    ind = 0;//обнуляем индекс для строки
    temp.clear();//очищаем переменную от старых данных для записи новых
}
cout << "База данных загружена\n" << endl;

inf.close();//закрываем файл
}

void search_task() { //функция для нахождения студентов, чей средний балл не меньше
указанного пользователем значения
    if (count == 0) {
        cout << "Список пуст" << endl;
        return;
    }
    double average;//переменная для хранения среднего значения оценок
    cout << "Введите среднее значение оценок\n";
    do { //ввод с проверкой

```

```

cin >> average;

if (cin.fail()) { //если в вводе произошла ошибка
    cout << "Некорректный ввод числа. Повторите ввод\n";
    cin.clear(); //очищаем флаги и буфер
    cin.ignore(cin.rdbuf()->in_avail());
}
else
    break;
} while (true);

vector<Student*> vec; //вектор для хранения студентов, средний балл которых не
меньше среднего average

Node* temp = head; //устанавливаем указатель на начало списка
double tempAver; //переменная для хранения среднего значения оценок текущего
студента
Student* s; //переменная для хранения структуры текущего студента
for (int i = 0; i < count; i++) { //проходимся по списку
    s = temp->stud; //вносим нашу структуру в переменную s
    double ku = s->sum_marks();
    tempAver = ku / s->marks.size(); //считаем среднее значение оценок текущего
студента
    s->tempAver = tempAver;
    if (tempAver >= average) //если среднее значение оценок студента не меньше
вносим в наш вектор
        vec.push_back(s);
    flag = true;
    temp = temp->next;
}

cout << "Результат:\n";
if (vec.size() == 0) //если ничего не записали в вектор
    cout << "Пусто\n";
for (int i = 0; i < vec.size(); i++) //выводим содержимое вектора
    cout << vec[i];
flag = false;
}

void show() { //функция вывода списка студентов
    if (count == 0) {
        cout << "Список пуст" << endl;
        return;
    }
    Node* temp = head;
    cout << endl << "| " << setw(13) << " Ученик " << "| " << setw(14) << " Группа " << "| "
<< setw(14)
        << " Оценки " << "| " << endl
        << "|-----|-----|-----|" << endl;
    for (int i = 0; i < count; i++) //пока не дошли до конца
    {
        cout << temp->stud;

```

```

        temp = temp->next; //Переставляем указатель на следующий элемент списка
    }
}

~StudsData() { //деструктор списка, который освобождает память из узлов и структур
данных
    Node* temp = head; //устанавливаем указатель на начало списка
    Node* prev; //указатель для хранения пройденного элемента
    for (int i = 0; i < count; i++) {
        prev = temp;
        temp = temp->next;
        delete prev->stud; //освобождаем память из-под узла и структуры
        delete prev;
    }
}

Node* getHead()
{
    return head;
}
};

int main()
{

    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    setlocale(LC_ALL, "Rus");

    StudsData data;

    int choice = -1;
    Student* s;
    int temp;
    while (choice != 0) {
        cout << "Доступные функции: " << endl;
        cout << "1. Добавить студента" << endl;
        cout << "2. Вывести все" << endl;
        cout << "3. Удалить" << endl;
        cout << "4. Поиск" << endl;
        cout << "5. Редактировать" << endl;
        cout << "6. Отсортировать" << endl;
        cout << "7. Сохранить" << endl;
        cout << "8. Загрузить из базы" << endl;
        cout << "9. Найти студентов, чей средний бал не меньше указанного пользователем
значения" << endl;
        cout << "0. Выйти" << endl;
        cout << "\nВыберите действие: ";
        choice = get_num();
    }
}

```

```

switch (choice) {
case 1:
    system("cls");
    s = data.input();
    data.add_stud(s);
    break;
case 2:
    system("cls");
    data.show();
    break;
case 3:
    system("cls");
    data.del_stud();
    break;
case 4:
    system("cls");
    data.show();
    cout << "Введите имя студента, которого хотите найти\n";
    s = data.search();
    if (s != nullptr)
        cout << s;
    else
        cout << "Студент не найден\n";
    break;
case 5:
    system("cls");
    data.show();
    data.edit_info();
    break;
case 6:
    system("cls");
    data.sort_group();

    break;
case 7:
    system("cls");
    data.in_to_base();
    break;
case 8:
    system("cls");
    data.out_from_base();
    break;
case 9:
    system("cls");
    data.search_task();
    break;
case 0:
    system("cls");
    cout << "Выход из программы";
    break;
default:
    cout << "Неизвестная команда" << endl;

```

```

        break;
    }
}

_CrtSetReportMode(_CRT_WARN, _CRTDBG_MODE_FILE);
_CrtSetReportFile(_CRT_WARN, _CRTDBG_FILE_STDOUT);
_CrtSetReportMode(_CRT_ERROR, _CRTDBG_MODE_FILE);
_CrtSetReportFile(_CRT_ERROR, _CRTDBG_FILE_STDOUT);
_CrtSetReportMode(_CRT_ASSERT, _CRTDBG_MODE_FILE);
_CrtSetReportFile(_CRT_ASSERT, _CRTDBG_FILE_STDOUT);
_CrtSetDbgFlag(_CRTDBG_ALLOC_MEM_DF | _CRTDBG_LEAK_CHECK_DF);

return 0;
}

```