



Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУК «Информатика и управление»

КАФЕДРА ИУК4 «Программное обеспечение ЭВМ, информационные технологии»

О Т Ч Е Т

УЧЕБНАЯ ПРАКТИКА

«Ознакомительная практика»

Студент гр. ИУК4-12Б _____ (Ёлгин М.М.)
(подпись) (Ф.И.О.)

Руководитель _____ (Пчелинцева Н.И.)
(подпись) (Ф.И.О.)

Оценка руководителя _____ баллов _____
30-50 (дата)

Оценка защиты _____ баллов _____
30-50 (дата)

Оценка практики _____ баллов _____
(оценка по пятибалльной шкале)

Комиссия: _____ (Пчелинцева Н.И.)
(подпись) (Ф.И.О.)

_____ (Амеличева К.А.)
(подпись) (Ф.И.О.)

_____ (Гагарин Ю.Е.)
(подпись) (Ф.И.О.)

Калуга, 2021

Калужский филиал федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой ИУК4
_____(Гагарин Ю.Е.)
« 08 » сентября 2021 г.

З А Д А Н И Е
на УЧЕБНУЮ, ОЗНАКОМИТЕЛЬНУЮ ПРАКТИКУ

За время прохождения практики студенту необходимо:

1. Согласовать предметную область и тему проекта, закрепить сроки и требования по различным этапам реализации проекта, оформить требования в виде технического задания, утвердить техническое задание, оценить качество составленных документов.
2. Спроектировать структуру разрабатываемого приложения, продумать интерфейс взаимодействия пользователя с системой, оформить результаты работы в виде блок-схем, осуществить выбор библиотек и других технологий разработки.
3. Разработать и реализовать алгоритмы функционирования приложения, структуры, систем передачи информации, технологий обработки информации и интерфейса взаимодействия пользователя с системой, редактировать техническую документацию в соответствии с требованиями ГОСТ.
4. Разработать программное приложение для работы с файлами данных, содержащих информацию о студентах учебной группы
5. Подготовить отчет и защитить результаты практики.

Дата выдачи задания « 08 » сентября 2021 г.

Руководитель практики	_____ Пчелинцева Н.И.
Задание получил студент группы ИУК4-12Б	_____ Ёлгин М.М.

Оглавление

ВВЕДЕНИЕ	4
1. ПОДГОТОВКА К ВЫПОЛНЕНИЮ ЗАДАНИЯ	5
1.1. Анализ поставленной задачи	5
1.2. Выбор метода решения задачи	6
1.3. Технические требования	6
2. ИССЛЕДОВАТЕЛЬСКИЙ ЭТАП.....	7
2.1. Принцип работы приложения.....	7
2.2. Фильтрация ввода	7
2.3. Структура и организация кода программы	8
2.4. Программная организация структуры данных и меню	8
2.5. Программная реализация алгоритмов.....	9
2.6. Тестирование и отладка.....	14
ЗАКЛЮЧЕНИЕ	19
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	20
ПРИЛОЖЕНИЕ	22

ВВЕДЕНИЕ

Целью учебной практики является знакомство с основами будущей профессии, получение сведений о специфике избранной специальности, овладение первичными профессиональными умениями и навыками:

- по постановке задачи на разработку требований к подсистемам и контроль их качества,
 - по применению методов оценивания временной и емкостной сложности программного обеспечения,
 - разработке требований к системе в целом,
- а также подготовка обучающегося к осознанному и углубленному практическому изучению учебных дисциплин.

Для достижения поставленной цели решаются следующие задачи:

- осуществить выбор библиотек, среды разработки, системы контроля версий, обосновать соответствующий выбор,
- подготовить доклад с использованием средств визуализации (презентации, графики, блок-схемы, UML-диаграммы и пр.), в котором будет представлен отчёт студента о проделанной работе, общая информация о разработанном проекте, указаны его преимущества, недостатки и перспективы дальнейшего развития.

1. ПОДГОТОВКА К ВЫПОЛНЕНИЮ ЗАДАНИЯ

1.1. Анализ поставленной задачи

Необходимо используя одномерные массивы составить список учебной группы. Для каждого студента указать имя, фамилию, отчество, год рождения, группу, оценки, по шести предметам, то есть базу данных для учебной группы.

Для выполнения этого задания возможно использовать множество массивов с данными группы, но такая организация данных будет сложна в обработке, а также усложнит процесс создания и поддержки программы. Использовать для хранения данных структуры гораздо проще и практичнее.

Необходимо обеспечить ввод информации пользователем:

- С клавиатуры. Для обеспечения корректного ввода данных пользователем желательно сделать проверку вводимой информации. Для более быстрого и комфортно использования программы можно добавить проверку при вводе пользователем каждого символа с клавиатуры
- Из уже созданного текстового файла
- Из уже созданного бинарного файла. Возможно ввести свой формат файла

Также необходимо обеспечить вывод информации из базы данных:

- На экран в виде таблицы
- В файл. Будет использоваться текстовый файл, как самый надёжный и долговечный тип файлов
- На экран по запросу: вывести фамилии и группы студентов, получивших не более двух 4 (остальные оценки - 5)

Необходим и другой функционал по работе с программой и записями в базе данных:

- Перевод содержимого текстового файла в бинарный файл
- Добавление записи

- Изменение записи
- Удаление записи
- Сортировка. Сортировка по полям имя, фамилия, отчество в алфавитном порядке; по полям год рождения, группа в порядке их возрастания или убывания
- Выход из программы

Для организации удобного и простого интерфейса с функционалом базы данных нужно создать меню, содержащее в себе другие подменю с возможностью быстро выбирать и воспользоваться нужными пунктами.

1.2. Выбор метода решения задачи

Для решения поставленной задачи решено использовать высокоуровневый язык программирования C++. Для разработки программы использовалась среда разработки Microsoft Visual Studio Community 2019.

1.3. Технические требования

- 1) Персональный компьютер типа IBM PC, под управлением русифицированной версии операционной системы MS Windows XP/ Vista.
- 2) Процессор Intel Pentium 4 / Celeron.
- 3) Оперативная память 256 / 512 Мбайт.
- 4) Жёсткий диск (винчестер) оптимально 120 Гбайт.

Протестирована работоспособность программы в операционных системах Windows 10, Windows 11.

2. ИССЛЕДОВАТЕЛЬСКИЙ ЭТАП

2.1. Принцип работы приложения

Работа приложения заключается в обработке данных студенческой группы. При запуске программы выводится меню. Выбирая пункт меню, пользователь может добавить запись, изменить запись полностью или частично, удалять запись студента, сортировать группу студентов по разным полям, вывести данные на экран в виде таблицы. Возможно вывести фамилии и группы студентов, получивших не более двух 4 (остальные оценки – 5) на экран. Также доступен вывод данных в текстовый файл и перевод текстового файла в бинарный файл. Бинарный файл можно создать с разными расширениями или вообще без расширения. Есть возможность перевести данные учебной группы в текстовый файл в виде распечатки таблицы.

После добавления или загрузки данных их можно сортировать:

- По алфавиту
 - По имени
 - По фамилии
 - По отчеству
- По году рождения
 - Сверху старше
 - Сверху младше
- По группе
 - Сверху номер группы больше
 - Сверху номер группы меньше

2.2. Фильтрация ввода

Во время ввода данных с клавиатуры производится проверка каждого введённого символа, то есть пользователь может ввести только строку, состоящую из определённых символов, разрешённой длины. Если будет введён символ не являющийся разрешённым, то символ не выведется на

экран. Пользователь может быстро перемещаться по интерфейсу приложения и переходить из одного пункта меню в другие, благодаря обработки символов прямо из потока ввода. Поэтому для ввода номера пункта меню не нужно нажимать кнопку ввода. Также каждое поле имеет разные ограничения по количеству символов вводимой информации.

2.3. Структура и организация кода программы

Для каждой представленной возможности программы была написана функция, осуществляющая её. Некоторые функции, например, вывод строки, используются при выводе таблицы на экран, при редактировании записи и при выводе таблицы в текстовый файл. Вывод данных в разные выходные потоки возможен благодаря передачи ссылки на поток в функцию вывода строки таблицы. Многие функции в программе используют другие базовые функции, решающие более конкретные задачи. Благодаря такому подходу и такой архитектуре удалось значительно уменьшить размер программы и количество времени, затраченное на её разработку.

2.4. Программная организация структуры данных и меню

Для хранения данных и удобного применения операций над ними было принято решение использовать массив, содержащий структуры. Структура Student содержит информацию о студенте: имя, фамилия, отчество, год рождения, группа, оценки по шести предметам. Оценки хранятся в массиве, состоящем из шести элементов.

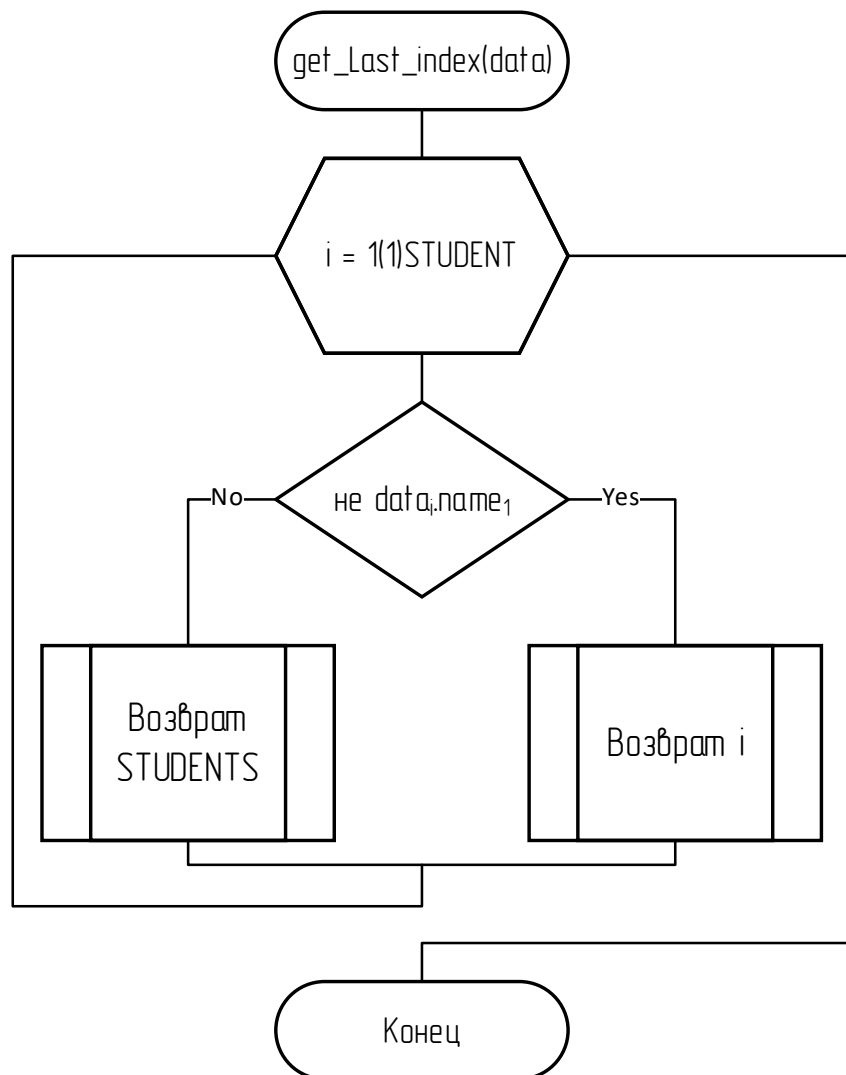
Student
+name
+sur_name
+last_name
+year_of_birth
+group
+marks

В главной функции `main()` было организовано общее меню, используя оператор `switch()`, в каждом `case` выводится подменю, где выбирается нужный пункт и осуществляется вызов соответствующей функции.

2.5. Программная реализация алгоритмов

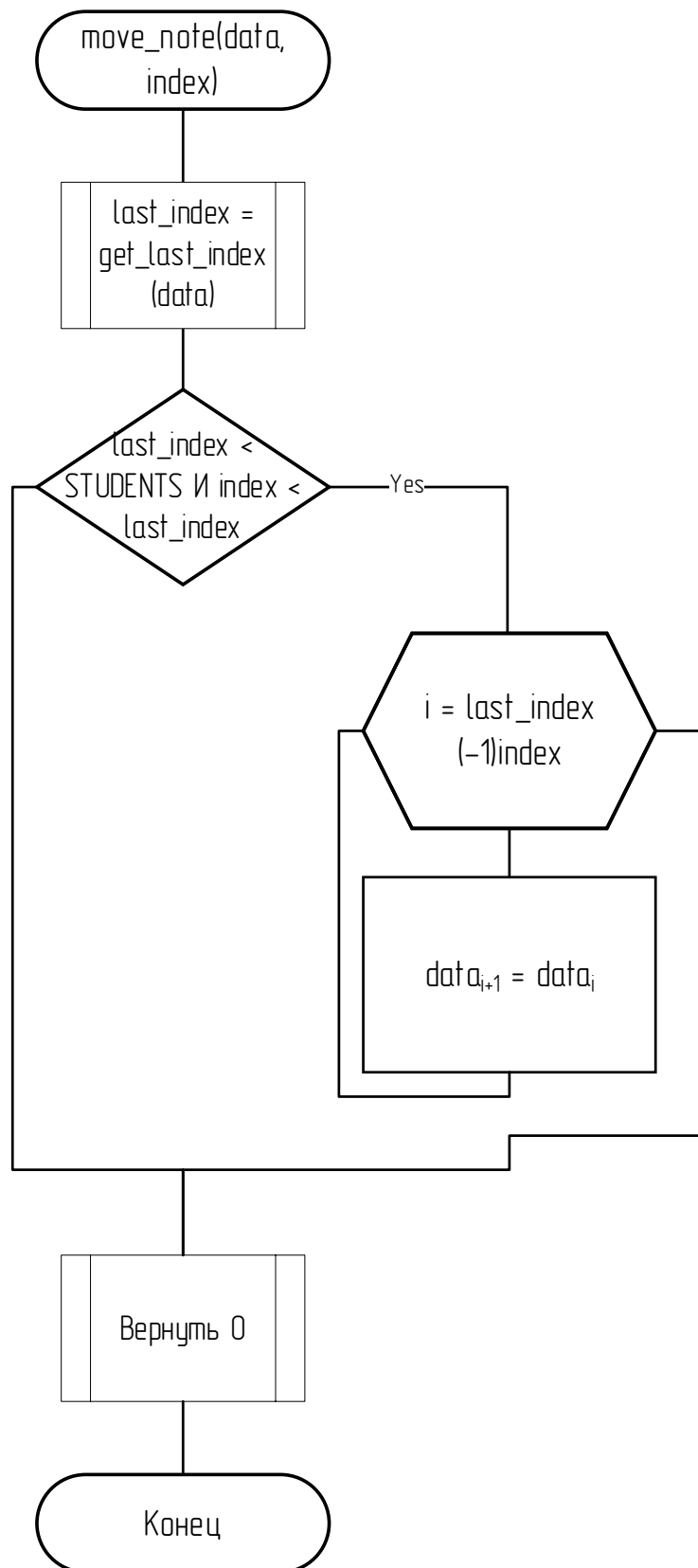
Рассмотрим часть используемых функций.

1. Получение номера последнего ученика, то есть количество учеников



Функция получает массив со структурами и проходит по массиву снизу вверх, проверяя пустое ли `name`, если да, то возвращает индекс этой структуры. Если же не найдётся структуры с пустым `name`, то вернуть максимально возможное количество студентов.

2. Смещение всех записей начиная с номера выбранной записи до конца записей



В функцию передаются массив структур – данные учебной группы и индекс структуры, начиная с которой нужно сместить записи. Каждая запись

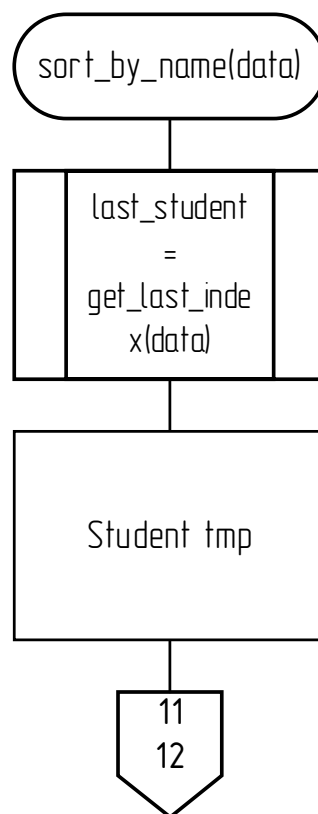
начиная с последней перезаписывает следующую, пока не дойдёт до записи с номером index.

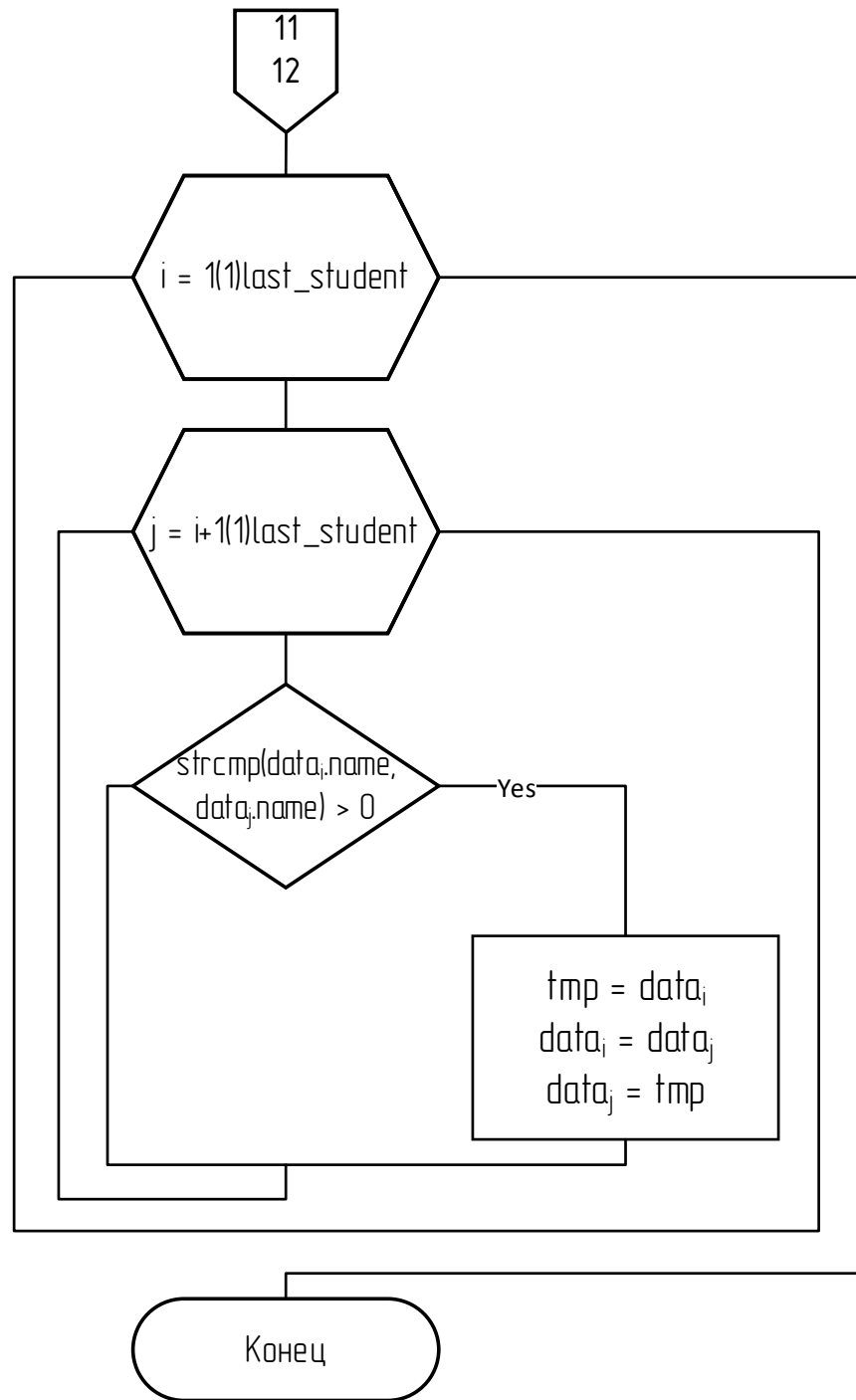
3. Функции сортировки базы данных

Данные функции сортируют записи:

- По алфавиту
 - По имени
 - По фамилии
 - По отчеству
- По году рождения
 - Сверху старше
 - Сверху младше
- По группе
 - Сверху номер группы больше
 - Сверху номер группы меньше

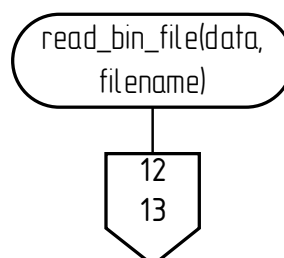
Рассмотрим реализацию одной из функций сортировки

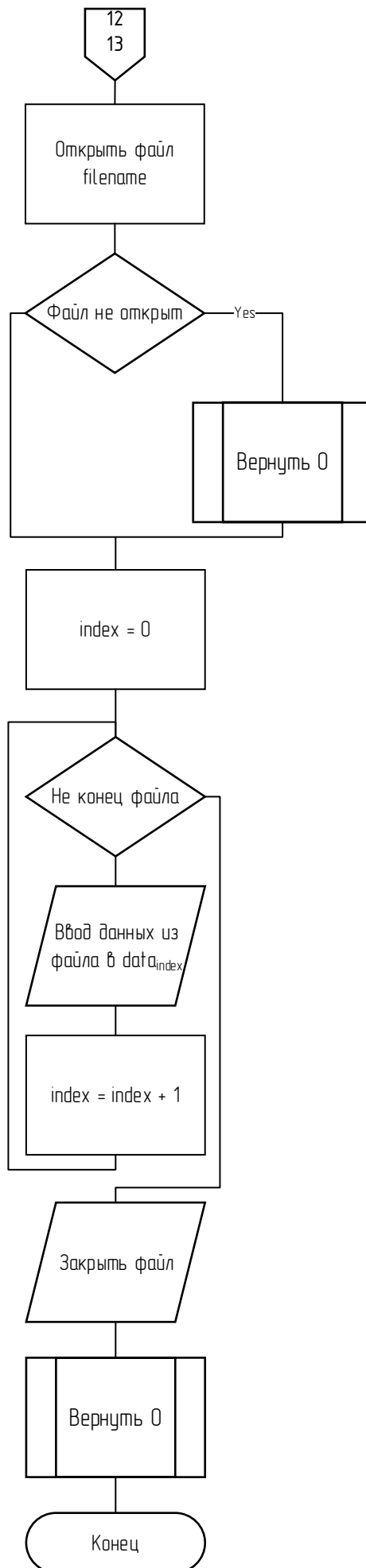




В функции сортировки по алфавиту используется встроенная функция языка C++ `strcmp(str1, str2)`, принимающая две строки и сравнивающая их.

4. Чтение данных из бинарного файла





2.6. Тестирование и отладка

При запуске программы выводится меню:

```
1| Ввод информации
2| Вывод данных
3| Перевод данных
4| Сортировка
0| Выход
Введите номер пункта меню:
```

Ввод данных:

```
ВВОД ДАННЫХ
=====
1| Добавить запись
2| Ввод информации из уже созданного текстового файла
3| Ввод информации из уже созданного бинарного файла
4| Изменение записей
5| Удаление записей
0| Назад
Введите номер пункта меню:
```

Добавить запись:

```
=====
|| ДОБАВИТЬ ЗАПИСЬ ||
=====
Имя:Иван
Фамилия:иванов
Отчество:Иванович
Год рождения:2000
Группа:72
Оценки:
>> Предмет 1:5
>> Предмет 2:5
>> Предмет 3:5
>> Предмет 4:4
>> Предмет 5:4
>> Предмет 6:4
Вернуться в меню (0-Нет / 1-Да):
```

Повторное добавление записи:

```
=====
|| ДОБАВИТЬ ЗАПИСЬ ||
=====
Введите номер ученика (1 - 2):1
Запись с таким номером уже существует. Хотите сместить все записи после неё или заменить запись (1-Смещение / 2-Замена):
СМЕЩЕНИЕ ЗАПИСИ
Имя:Глеб
Фамилия:Суворов
Отчество:
Год рождения:1994
Группа:71
Оценки:
>> Предмет 1:4
>> Предмет 2:4
>> Предмет 3:2
>> Предмет 4:5
>> Предмет 5:5
>> Предмет 6:5
Вернуться в меню (0-Нет / 1-Да):
```

Ввод информации из уже созданного текстового файла:

```
=====
|| ВВОД С ИНФОРМАЦИИ ИЗ УЖЕ СОЗДАННОГО ТЕКСТОВОГО ФАЙЛА ||
=====
Введите имя файла:text
Данные из файл "text.txt" считаны
Для продолжения нажмите любую клавишу . . .
```

Ввод информации из уже созданного бинарного файла:

```
=====
|| ВВОД С ИНФОРМАЦИИ ИЗ УЖЕ СОЗДАННОГО БИНАРНОГО ФАЙЛА ||
=====
Введите имя файла:data.dat
Данные из файл "data.dat" считаны
Для продолжения нажмите любую клавишу . . .
```

Изменение записей:

```
=====
|| ИЗМЕНИТЬ ЗАПИСЬ ||
=====
Введите номер ученика (1 - 7):2
=====
|| ИЗМЕНИТЬ ЗАПИСЬ ||
=====
Оценки:
№ | Имя | Фамилия | Отчество | Год рождения | Группа | Предмет 1 | Предмет 2 | Предмет 3 | Предмет 4 | Предмет 5 | Предмет 6 |
2 | Василиса | Шевцова | Макаровна | 2000 | 72 | 5 | 4 | 5 | 4 | 5 | 5 |

Изменить:
1| Имя
2| Фамилия
3| Отчество
4| Год рождения
5| Группа
6| Оценки
7| Всё
0| Назад
Введите номер пункта меню:

=====
|| ИЗМЕНИТЬ ЗАПИСЬ ||
=====
Оценки:
№ | Имя | Фамилия | Отчество | Год рождения | Группа | Предмет 1 | Предмет 2 | Предмет 3 | Предмет 4 | Предмет 5 | Предмет 6 |
2 | Василиса | Шевцова | Макаровна | 2000 | 72 | 5 | 4 | 5 | 4 | 5 | 5 |

Изменить:
1| Имя
2| Фамилия
3| Отчество
4| Год рождения
5| Группа
6| Оценки
7| Всё
0| Назад
Введите номер пункта меню:
Год рождения:1999

=====
|| ИЗМЕНИТЬ ЗАПИСЬ ||
=====
Оценки:
№ | Имя | Фамилия | Отчество | Год рождения | Группа | Предмет 1 | Предмет 2 | Предмет 3 | Предмет 4 | Предмет 5 | Предмет 6 |
2 | Василиса | Шевцова | Макаровна | 1999 | 72 | 5 | 4 | 5 | 4 | 5 | 5 |

Изменить:
1| Имя
2| Фамилия
3| Отчество
4| Год рождения
5| Группа
6| Оценки
7| Всё
0| Назад
Введите номер пункта меню: █
```

Удаление записей:

```
=====
|| УДАЛИТЬ ЗАПИСЬ ||
=====
Введите номер ученика (1 - 7):2
ЗАПИСЬ УДАЛЕНА
Вернуться в меню (0-Нет / 1-Да):
```

Вывод данных:

ВЫВОД ДАННЫХ


```
=====
1| Вывод данных на экран
2| Вывод данных в текстовый файл
3| Распечатка информации данных по запросу
0| Назад
Введите номер пункта меню:
```

Вывод данных на экран:

```
=====
|| ВЫВОД ДАННЫХ НА ЭКРАН ||
=====
Оценки:
№ | Имя | Фамилия | Отчество | Год рождения | Группа | Предмет 1 | Предмет 2 | Предмет 3 | Предмет 4 | Предмет 5 | Предмет 6 |
1| Владимир | Козлов | Дмитриевич | 1994 | 71 | 5 | 5 | 5 | 5 | 5 | 5 |
2| Артём | Черняев | Филиппович | 1999 | 51 | 4 | 4 | 3 | 5 | 4 | 5 |
3| Николай | Назаров | Дмитриевич | 2003 | 11 | 5 | 4 | 4 | 5 | 5 | 4 |
4| Анна | Семина | | 2002 | 31 | 3 | 4 | 4 | 5 | 4 | 4 |
5| Андрей | Астафьев | Егорович | 2001 | 32 | 5 | 4 | 4 | 5 | 4 | 4 |
6| Артём | Шаповалов | | 2000 | 72 | 3 | 3 | 3 | 4 | 3 | 4 |
Для продолжения нажмите любую клавишу . . .
```

Вывод данных в текстовый файл:

```
=====
|| ВЫВОД ДАННЫХ В ТЕКСТОВЫЙ ФАЙЛ ||
=====
Введите имя файла:text2
Файл "text2.txt" записан
Для продолжения нажмите любую клавишу . . .
```

 text2.txt – Блокнот

Файл Изменить Формат

Владимир
Козлов
Дмитриевич
71
1994
5
5
5
5
5
5

Артём
Черняев
Филиппович
51
1999
4
4
3
5
4
5

Распечатка данных по запросу:

```
=====
|| ВЫВОД СТУДЕНТОВ, ПОЛУЧИВШИХ НЕ БОЛЕЕ ДВУХ 4 (ОСТАЛЬНЫЕ ОЦЕНКИ - 5) НА ЭКРАН ||
=====
```

```
| Фамилия      | Группа |
| Козлов       | 71     |
```

Для продолжения нажмите любую клавишу . . .

Перевод данных:

ПЕРЕВОД ДАННЫХ

- ```
=====
1| Перевод базы данных в текстовый файл
2| Перевод содержимого текстового файла в бинарный файл
0| Назад
```

Введите номер пункта меню:

## Перевод базы данных в текстовый файл:

```
=====
|| ПЕРЕВОД БАЗЫ ДАННЫХ В ВИДЕ ТАБЛИЦЫ В ТЕКСТОВЫЙ ФАЙЛ ||
=====
```

Введите имя текстового файла:table

Файл "table.txt" записан

Для продолжения нажмите любую клавишу . . .

table.txt – Блокнот

Файл Изменить Формат Вид Справка

|   |          |           |            | Оценки:      |        |           |           |           |           |           |           |
|---|----------|-----------|------------|--------------|--------|-----------|-----------|-----------|-----------|-----------|-----------|
| № | Имя      | Фамилия   | Отчество   | Год рождения | Группа | Предмет 1 | Предмет 2 | Предмет 3 | Предмет 4 | Предмет 5 | Предмет 6 |
| 1 | Владимир | Козлов    | Дмитриевич | 1994         | 71     | 5         | 5         | 5         | 5         | 5         | 5         |
| 2 | Артём    | Черняев   | Филиппович | 1999         | 51     | 4         | 4         | 3         | 5         | 4         | 5         |
| 3 | Николай  | Назаров   | Дмитриевич | 2003         | 11     | 5         | 4         | 4         | 5         | 5         | 4         |
| 4 | Анна     | Семина    |            | 2002         | 31     | 3         | 4         | 4         | 5         | 4         | 4         |
| 5 | Андрей   | Астафьев  | Егорович   | 2001         | 32     | 5         | 4         | 4         | 5         | 4         | 4         |
| 6 | Артём    | Шаповалов |            | 2000         | 72     | 3         | 3         | 3         | 4         | 3         | 4         |

## Перевод содержимого текстового файла в бинарный файл:

```
=====
|| ПЕРЕВОД ДАННЫХ ТЕКСТОВОГО ФАЙЛА В БИНАРНЫЙ ФАЙЛ ||
=====
```

Введите имя текстового файла:text

Введите имя бинарного файла:data.dat

Файл "data.dat" записан

Для продолжения нажмите любую клавишу . . .

## Сортировка:

СОРТИРОВКА ДАННЫХ

- ```
=====
1| Сортировка по алфавиту
2| Сортировка по году рождения
3| Сортировка по группам
0| Назад
```

Введите номер пункта меню:

Сортировка по алфавиту:

```
=====
|| СОРТИРОВКА ПО АЛФАВИТУ ||
=====
1| Сортировать по имени
2| Сортировать по фамилии
3| Сортировать по отчеству
0| Назад
Введите номер пункта меню:
```

Сортировка по имени:

```
=====
|| ВЫВОД ДАННЫХ НА ЭКРАН ||
=====
Оценки:
№ | Имя | Фамилия | Отчество | Год рождения | Группа | Предмет 1 | Предмет 2 | Предмет 3 | Предмет 4 | Предмет 5 | Предмет 6 |
1 | Андрей | Астафьев | Егорович | 2001 | 32 | 5 | 4 | 4 | 5 | 4 | 4 |
2 | Анна | Семина | | 2002 | 31 | 3 | 4 | 4 | 5 | 4 | 4 |
3 | Артём | Черняев | Филиппович | 1999 | 51 | 4 | 4 | 3 | 5 | 4 | 5 |
4 | Артём | Шаповалов | | 2000 | 72 | 3 | 3 | 3 | 4 | 3 | 4 |
5 | Владимир | Козлов | Дмитриевич | 1994 | 71 | 5 | 5 | 5 | 5 | 5 | 5 |
6 | Николай | Назаров | Дмитриевич | 2003 | 11 | 5 | 4 | 4 | 5 | 5 | 4 |
Для продолжения нажмите любую клавишу . . .
```

Сортировка по полям фамилия и отчество работает аналогично.

Сортировка по году рождения:

```
=====
|| СОРТИРОВКА ПО ГОДУ РОЖДЕНИЯ ||
=====
1| Сверху старше
2| Сверху младше
0| Назад
Введите номер пункта меню:
Сортировка завершена
Для продолжения нажмите любую клавишу . . .
```

```
=====
|| ВЫВОД ДАННЫХ НА ЭКРАН ||
=====
Оценки:
№ | Имя | Фамилия | Отчество | Год рождения | Группа | Предмет 1 | Предмет 2 | Предмет 3 | Предмет 4 | Предмет 5 | Предмет 6 |
1 | Владимир | Козлов | Дмитриевич | 1994 | 71 | 5 | 5 | 5 | 5 | 5 | 5 |
2 | Артём | Черняев | Филиппович | 1999 | 51 | 4 | 4 | 3 | 5 | 4 | 5 |
3 | Артём | Шаповалов | | 2000 | 72 | 3 | 3 | 3 | 4 | 3 | 4 |
4 | Андрей | Астафьев | Егорович | 2001 | 32 | 5 | 4 | 4 | 5 | 4 | 4 |
5 | Анна | Семина | | 2002 | 31 | 3 | 4 | 4 | 5 | 4 | 4 |
6 | Николай | Назаров | Дмитриевич | 2003 | 11 | 5 | 4 | 4 | 5 | 5 | 4 |
Для продолжения нажмите любую клавишу . . .
```

Сортировка по группам работает аналогично.

ЗАКЛЮЧЕНИЕ

В ходе выполнения задания на учебную, ознакомительную практику была создана и протестирована программа, обрабатывающая данные учебной группы студентов. Были получены навыки работы со структурными типами данных, функциями, потоками, текстовыми и бинарными файлами средствами языка программирования C++. Во время создания такой базы данных мною также были получены навыки работы с системой управления версиями Git, а также навыки работы в системе управления проектами и версиями кода GitHub.

Работа над этим проектом сильно расширила и улучшила мои навыки программирования на высокоуровневом языке C++.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Моделирование информационных ресурсов [Электронный ресурс]: учебно-методический/ Составитель Огнев Э.Н. - Кемерово : Кемеровский государственный университет культуры и искусств, 2013. - 36 с. : ил., табл. - URL: <http://biblioclub.ru/index.php?page=book&id=274218>
2. Коваленко, Ю.В. Информационно-поисковые системы [Электронный ресурс]: учебно-методическое пособие / Ю.В. Коваленко, Т.А. Сергиенко. — Омск: Омская юридическая академия, 2017. — 38 с.— Режим доступа: <http://www.iprbookshop.ru/66817.html>
3. Маюрникова, Л. А. Основы научных исследований в научно-технической сфере [Электронный ресурс]: учебно-методическое пособие / Л. А. Маюрникова, С. В. Новосёлов. — Кемерово: Кемеровский технологический институт пищевой промышленности, 2009. — 123 с. — Режим доступа: <http://www.iprbookshop.ru/14381.html>
4. Вайнштейн, М. З. Основы научных исследований [Электронный ресурс]: учебное пособие / М. З. Вайнштейн, В. М. Вайнштейн, О. В. Кононова. — Йошкар-Ола: Марийский государственный технический университет, Поволжский государственный технологический университет, ЭБС АСВ, 2011. — 216 с. — Режим доступа: <http://www.iprbookshop.ru/22586.html>
5. Мокий, М.С. Методология научных исследований[Текст]: учебник / М.С. Мокий, А.Л. Никифоров, В.С. Мокий. - М.: Юрайт, 2015. - 255 с.
6. Рогов, В.А. Методика и практика технических экспериментов[Текст]: учеб.пособие / В.А. Рогов, А.В. Антонов, Г.Г. Поздняк. – М.: Академия, 2005. – 288 с.
7. Щербаков, А. Интернет-аналитика [Электронный ресурс]: поиск и оценка информации в web-ресурсах: практическое пособие / А. Щербаков. - М.: Книжный мир, 2012. - 78 с. - URL: <http://biblioclub.ru/index.php?page=book&id=89693>.
8. Моделирование систем[Текст]: учебник для вузов / С.И. Дворецкий, Ю.Л. Муромцев, В.А. Погонин, А.Г. Схиртладзе. – М.: Академия, 2009. – 320 с.
9. Порсев, Е. Г. Организация и планирование экспериментов [Электронный ресурс]: учебное пособие / Е. Г. Порсев.— Новосибирск : Новосибирский государственный технический университет, 2010. — 155 с. — Режим доступа: <http://www.iprbookshop.ru/45415.html>
10. Герберт Шилдт «С++ для начинающих» Издательский дом «Вильямс» 2005.
11. Роберт Седжвик «Алгоритмы на С++. Анализ структуры данных. Сортировка. Поиск.
12. А.Д. Хомоненко «Программирование на С++»
13. Бьерн Страуструп «Язык программирования С++»
14. Роберт Лафоре «Объектно-ориентированное программирование в С++»
15. Белева, Л. Ф. Программирование на языке С++ : учебное пособие / Л. Ф. Белева. — Саратов : Ай Пи Эр Медиа, 2018. — 81 с. — ISBN 978-5-4486-

0253-5. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: <https://www.iprbookshop.ru/72466.html>

ПРИЛОЖЕНИЕ

```
#include <iostream>
#include <iomanip>
#include <conio.h>
#include <windows.h>
#include <fstream>
using namespace std;

const char ALPHABET[] =
"ABCDEFGHIJKLMNOPQRSTUVWXYZАБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯabcdefghijklmnopqrstuvwxyz
opqrstuvwxyzабвгдеёжзийклмнопрстуфхцчшщъыьэюя";

enum Setup_const {
    FILE_NAME_LEN = 30,
};

enum Data_const {
    STUDENTS = 31,
    COLUMNS = 11,
    MAX_NAME_LEN = 18,
    MAX_SURNAME_LEN = 18,
    MAX_LAST_NAME_LEN = 21,
    MAX_GROUP = 99,
    GROUP_WIDTH = 6,
    MARK_COUNT = 6,
    MAX_MARK = 5,
    MAX_MARK_LEN = 1,
};

struct Student {
    char name[MAX_NAME_LEN]{};
    char sur_name[MAX_SURNAME_LEN]{};
    char last_name[MAX_LAST_NAME_LEN] = "";
    int year_of_birth{ 0 };
    int group{ 0 };
    int marks[MARK_COUNT]{};
};

int centred(const int, int, ofstream&);
int centred(const char[], int, ostream&);
int input_fast_menu(int);
bool is_in_digit(char);
int input_int(int, int);
```

```

bool is_in_mas(char, const char arr[]);
void print_header(const char str[]);
void print_line(const int);
int add_note(Student data[]);
int edit_note(Student data[]);
int delete_note(Student data[]);
int get_last_index(Student data[]);
int move_note(Student data[], int);
void print_hat(ostream&);
void output_row(Student data[], int, ostream&);
int output_display(Student data[]);
void output_on_request(Student data[]);
void sort_by_name(Student data[]);
void sort_by_sur_name(Student data[]);
void sort_by_last_name(Student data[]);
void sort_by_year_of_birth_up(Student data[]);
void sort_by_year_of_birth_down(Student data[]);
void sort_by_group_up(Student data[]);
void sort_by_group_down(Student data[]);
int sort_alpha_menu(Student data[]);
int sort_year_of_birth_menu(Student data[]);
int sort_group_menu(Student data[]);
int write_text_file(Student data[], int, char filename[]);
int write_text_file_menu(Student data[]);
int counvert_bin_file(char in_filename[], char out_filename[]);
int convert_to_bin_file_menu(Student data[]);
int write_file_table_menu(Student data[]);
int read_text_file_menu(Student data[]);
int read_text_file(Student data[]);
int read_bin_file_menu(Student data[]);
int write_file_table(char filename[], Student data[], int);

int main() {
    system("chcp 1251 >> null");

    bool exit_flag = false;

    enum Menu_const {
        INPUT = 1,
        OUTPUT = 2,
        TRANSLATION = 3,
        SORT = 4,
        EXIT = 0
    };
};

```

```

Student data[STUDENTS]{};

do {
    system("cls");
    cout << "1| Ввод информации" << endl;
    cout << "2| Вывод данных" << endl;
    cout << "3| Перевод данных" << endl;
    cout << "4| Сортировка" << endl;
    cout << "0| Выход" << endl;

    cout << "Введите номер пункта меню: ";
    int ask = input_fast_menu(4);

    system("cls");
    switch (ask) {
    case INPUT:
        cout << "ВВОД ДАННЫХ" << endl;
        print_line(60);
        cout << "1| Добавить запись" << endl;
        cout << "2| Ввод информации из уже созданного текстового файла" <<
endl;
        cout << "3| Ввод информации из уже созданного бинарного файла" <<
endl;
        cout << "4| Изменение записей" << endl;
        cout << "5| Удаление записей" << endl;
        cout << "0| Назад" << endl;
        cout << "Введите номер пункта меню: ";
        ask = input_fast_menu(5);
        switch (ask) {
        case 1:
            add_note(data);
            break;
        case 2:
            read_text_file_menu(data);
            break;
        case 3:
            read_bin_file_menu(data);
            break;
        case 4:
            edit_note(data);
            break;
        case 5:
            delete_note(data);

```



```

        break;
case 0:
default:
        break;
}
break;

case OUTPUT:
    cout << "ВЫВОД ДАННЫХ" << endl;
    print_line(60);
    cout << "1| Вывод данных на экран" << endl;
    cout << "2| Вывод данных в текстовый файл" << endl;
    cout << "3| Распечатка информации данных по запросу" << endl;
    cout << "0| Назад" << endl;
    cout << "Введите номер пункта меню: ";
    ask = input_fast_menu(3);

switch (ask) {
case 1:
    output_display(data);
    break;
case 2:
    write_text_file_menu(data);
    break;
case 3:
    output_on_request(data);
    break;
case 0:
default:
    break;
}
break;

case TRANSLATION:
    cout << "ПЕРЕВОД ДАННЫХ" << endl;
    print_line(60);
    cout << "1| Перевод базы данных в текстовый файл" << endl;
    cout << "2| Перевод содержимого текстового файла в бинарный файл"
<< endl;
    cout << "0| Назад" << endl;
    cout << "Введите номер пункта меню: ";
    ask = input_fast_menu(2);

switch (ask) {

```

```

case 1:
    write_file_table_menu(data);
    break;
case 2:
    convert_to_bin_file_menu(data);
    break;
case 0:
default:
    break;
}
break;
case SORT:
    cout << "СОРТИРОВКА ДАННЫХ" << endl;
    print_line(60);
    cout << "1| Сортировка по алфавиту" << endl;
    cout << "2| Сортировка по году рождения" << endl;
    cout << "3| Сортировка по группам" << endl;
    cout << "0| Назад" << endl;
    cout << "Введите номер пункта меню: ";
    ask = input_fast_menu(3);

switch (ask) {
case 1:
    sort_alpha_menu(data);
    break;
case 2:
    sort_year_of_birth_menu(data);
    break;
case 3:
    sort_group_menu(data);
    break;
case 0:
default:
    break;
}
break;
case EXIT:
    system("cls");
    cout << "Выйти из программы? (1-Да / 0-Нет) ";
    ask = input_fast_menu(1);
    if (ask == 1) {
        exit_flag = true;
    }
    break;

```

```

    }
} while (!exit_flag);

system("cls");
cout << "Database by Mikhail Yolgin" << endl;
cout << "          v1.0" << endl;

return 0;
}

#pragma region basic
bool is_in_digit(char str) {
    if (str >= int('0') && str <= int('9')) return 1;
    return 0;
}

int input_fast_menu(int end) {
    /*
    Функция не работает с отрицательными значениями!
    */
    char ch{};
    int num{}, i{}, tmp{}, begin = 0;
    while (ch != '\r' || num < begin || num > end) {
        // ввод символов и проверка
        ch = _getch();
        if (ch != '\r' && is_in_digit(ch) && +ch - 48 <= end) {
            num = ch - 48; // в ASCII цифры начинаются с 48, поэтому отнимаем
от char 48 и получаем число
            i++;
            break;
        }
    }
    return num;
}

int input_int(int begin, int end) {
    /*
    Функция не работает с отрицательными значениями!
    */
    char ch{};
    int num{}, i{}, length{}, tmp{};
    if (abs(end) > abs(begin)) {
        tmp = end;
    }

```

```

else {
    tmp = begin;
}
while (tmp != 0) {
    tmp /= 10;
    length++;
}
while (ch != '\r' || num < begin || num > end || ch == '-') {
    // ввод символов и проверка
    ch = _getch();
    if (ch != '\r' && is_in_digit(ch) && num * 10 + ch - 48 <= end) {
        num *= 10;
        num += ch - 48; // в ASCII цифры начинаются с 48, поэтому отнимаем
от char 48 и получаем число
        i++;
        cout << ch;
    }
    else {
        if (ch != '\r' && ch == '-' && i == 0) {
            i++;
            num = 0;
            cout << ch;
        }
    }
    // стирание символов в консоли
    if (ch == '\b' && i > 0) {
        i--;
        num /= 10;
        cout << '\b' << ' ' << '\b';
    }
}
return num;
}

bool is_in_mas(char ch, const char arr[]) {
    if (strlen(arr) == 0) return 1;
    for (int i = 0; i < strlen(arr); i++) {
        if (ch == arr[i]) return 1;
    }
    return 0;
}

void input_str(char str[], const int LENGTH, const char
additional_chars[], const int to_remove_count) {

```

```

char ch{};
int i = -1 * to_remove_count;
while (ch != '\r') {
    ch = _getch();
    if (i < LENGTH - 1) {
        if (ch != '\r' && (is_in_mas(ch, ALPHABET) || is_in_mas(ch,
additional_chars))) {
            str[i] = ch;
            i++;
            cout << ch;
        }
    }
    // Стирание по символам
    if (ch == '\b' && i > 0) {
        i--;
        str[i] = 0;
        cout << '\b' << ' ' << '\b';
    }
}
str[i] = 0;
}

void input_str(char str[], const int LENGTH) {
    char ch{};
    int i{};
    while (ch != '\r') {
        ch = _getch();
        if (i < LENGTH - 1) {
            if (ch != '\r' && is_in_mas(ch, ALPHABET)) {
                str[i] = ch;
                i++;
                cout << ch;
            }
        }
        // Стирание по символам
        if (ch == '\b' && i > 0) {
            i--;
            str[i] = 0;
            cout << '\b' << ' ' << '\b';
        }
    }
    str[i] = 0;
}

```

```

void to_title(char str[]) {
    //toupper()
    char high[60 * 2] =
"ABCDEFGHIJKLMNOPQRSTUVWXYZАБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ";
    char low[60 * 2] =
"abcdefghijklmnopqrstuvwxyzабвгдеёжзийклмнопрстуфхцчшщъыьэюя";
    if (str[0] != 0) {
        for (int i = 0; i < 60; i++) {
            if (str[i] == low[i]) {
                str[i] = high[i];
                break;
            }
        }
    }
}

int centred(const int output, int length, ostream& out) {
    int len{ 0 };
    int tmp = output;
    while (tmp != 0) {
        tmp /= 10;
        len += 1;
    };
    char spacing[30] = " ";
    for (int i = 1; i < length - len; i++) {
        spacing[i] = ' ';
    }
    out << " " << output << spacing;
    return 0;
}

int centred(const char output[], int length, ostream& out) {
    int len = strlen(output);
    char spacing[30] = " ";
    for (int i = 1; i < length - len; i++) {
        spacing[i] = ' ';
    }
    out << " " << output << spacing;
    return 0;
}

void print_header(const char str[]) {
    int length = strlen(str);
    for (int i = 0; i < length + 6; i++) cout << '=';
}

```

```

    cout << endl;
    cout << "|| " << str << " ||" << endl;
    for (int i = 0; i < length + 6; i++) cout << '=';
    cout << endl;
}

void print_line(const int length) {
    for (int i = 0; i < length; i++) cout << '=';
    cout << endl;
}

#pragma endregion basic

#pragma region database_operation
int get_last_index(Student data[]) {
    for (int i = 0; i < STUDENTS; i++) {
        if (!data[i].name[0]) return i;
    }
    return STUDENTS;
}

int move_note(Student data[], int index) {
    int last_index = get_last_index(data);
    if (last_index < STUDENTS - 1 && index < last_index) {
        for (int i = last_index; i >= index; i--) {
            data[i + 1] = data[i];
        }
    }
    return 0;
}

void input_row(Student data[], int index) {
    char tmp_name[MAX_NAME_LEN]{};
    cout << "Имя:";
    while (tmp_name[0] == 0) input_str(tmp_name, MAX_NAME_LEN);
    to_title(tmp_name);
    strcpy_s(data[index].name, tmp_name);
    cout << endl;

    char tmp_surname[MAX_SURNAME_LEN]{};
    cout << "Фамилия:";
    while (tmp_surname[0] == 0) input_str(tmp_surname, MAX_SURNAME_LEN);
    to_title(tmp_surname);
    strcpy_s(data[index].sur_name, tmp_surname);
    cout << endl;
}

```

```

cout << "Отчество:";
input_str(data[index].last_name, MAX_LAST_NAME_LEN);
to_title(data[index].last_name);
cout << endl;

cout << "Год рождения:";
data[index].year_of_birth = input_int(1900, 2021);
cout << endl;

cout << "Группа:";
data[index].group = input_int(1, MAX_GROUP);
cout << endl;

cout << "Оценки:" << endl;
for (int i{}; i < MARK_COUNT; i++) {
    cout << ">>\tПредмет " << i + 1 << ":";
    data[index].marks[i] = input_int(1, MAX_MARK);
    cout << endl;
}
cout << endl;
}

void print_hat(ostream& out) {
    out << setw(MAX_NAME_LEN + 1 + MAX_NAME_LEN + 1 + MAX_LAST_NAME_LEN + 1 +
13 + GROUP_WIDTH + 1 + 13) << "Оценки:" << endl;
    out << "№ |";
    centred("Имя", MAX_NAME_LEN + 1, out);
    out << "|";
    centred("Фамилия", MAX_SURNAME_LEN + 1, out);
    out << "|";
    centred("Отчество", MAX_LAST_NAME_LEN + 1, out);
    out << "|";
    centred("Год рождения", 13, out);
    out << "|";
    centred("Группа", GROUP_WIDTH + 1, out);
    out << "|";
    for (int i{}; i < MARK_COUNT; i++) {
        char str_subject[10] = "Предмет ";
        str_subject[8] = char(i + 1 + 48);
        centred(str_subject, 10, out);
        out << "|";
    }
}

```



```

    out << endl;
}

void output_row(Student data[], int index, ostream& out) {
    out << setw(2) << right << index + 1 << "|";
    centred(data[index].name, MAX_NAME_LEN + 1, out);
    out << "|";
    centred(data[index].sur_name, MAX_SURNAME_LEN + 1, out);
    out << "|";
    if (strcmp(data[index].last_name, "") != 0 &&
        strcmp(data[index].last_name, "-") != 0) {
        centred(data[index].last_name, MAX_LAST_NAME_LEN + 1, out);
    }
    else {
        centred(" ", MAX_LAST_NAME_LEN + 1, out);
    }
    out << "|";
    if (data[index].year_of_birth != 0) {
        centred(data[index].year_of_birth, 13, out);
    }
    else {
        centred(" ", 13, out);
    }
    out << "|";
    if (data[index].group != 0) {
        centred(data[index].group, GROUP_WIDTH + 1, out);
    }
    else {
        centred(" ", GROUP_WIDTH + 1, out);
    }
    out << "|";
    for (int i = 0; i < MARK_COUNT; i++) {
        if (data[index].marks[i] != 0) {
            centred(data[index].marks[i], 10, out);
        }
        else {
            centred(" ", 10, out);
        }
    }
    out << "|";
}

out << endl;
}

```

```

int add_note(Student data[]) {
    int continue_flag{};
    do {
        int student_index{};
        system("cls");
        print_header("ДОБАВИТЬ ЗАПИСЬ");
        if (get_last_index(data) != 0) {
            cout << "Введите номер ученика (1 - " << get_last_index(data) + 1
<< "):";
            if (get_last_index(data) < STUDENTS - 1) {
                student_index = input_int(0, get_last_index(data) + 1);
            }
            else {
                student_index = input_int(0, get_last_index(data));
            }
            if (student_index == 0) return 0;
            cout << endl;
        }
        else {
            student_index = 1;
        }

        int index = student_index - 1;
        if (data[index].name[0]) {
            cout << "Запись с таким номером уже существует. Хотите сместить
все записи после неё или заменить запись (1-Смещение / 2-Замена): ";
            int ask{};
            ask = input_fast_menu(get_last_index(data) + 1);
            cout << endl;
            switch (ask) {
                case 1:
                    cout << "СМЕЩЕНИЕ ЗАПИСИ" << endl;
                    if (get_last_index(data) < STUDENTS - 1) {
                        move_note(data, index);
                        input_row(data, index);
                    }
                    else {
                        cout << "Выбрана последняя запись, смещение
невозможно" << endl;
                    }
                    break;
                case 2:
                    cout << "ЗАМЕНА ЗАПИСИ" << endl;
                    print_hat(cout);

```

```

        output_row(data, index, cout);
        input_row(data, index);
        break;
    default:
        return 0;
    }
}
else {
    input_row(data, index);
}

cout << "Вернуться в меню (0-Нет / 1-Да): ";
continue_flag = input_fast_menu(1);
} while (continue_flag == 0);

return 0;
}

int edit_note(Student data[]) {
    int continue_flag{};
    do {
        system("cls");
        print_header("ИЗМЕНИТЬ ЗАПИСЬ");
        if (get_last_index(data) != 0) {
            cout << "Введите номер ученика (1 - " << get_last_index(data) <<
"):";
        }
        else {
            cout << "Нет записей для изменения" << endl;
            system("pause");
            return 0;
        }
        int student_index = input_int(0, get_last_index(data));
        if (student_index == 0) return 0;
        cout << endl;

        int index = student_index - 1;
        if (data[index].name[0]) {
            int ask{};
            do {
                system("cls");
                print_header("ИЗМЕНИТЬ ЗАПИСЬ");
                print_hat(cout);
                output_row(data, index, cout);

```

```

cout << endl;
cout << "Изменить:" << endl;
cout << "1| Имя" << endl;
cout << "2| Фамилия" << endl;
cout << "3| Отчество" << endl;
cout << "4| Год рождения" << endl;
cout << "5| Группа" << endl;
cout << "6| Оценки" << endl;
cout << "7| Всё" << endl;
cout << "0| Назад" << endl;
cout << "Введите номер пункта меню: ";
ask = input_fast_menu(7);
cout << endl;
switch (ask) {
case 1: {
    char tmp_name[MAX_NAME_LEN]{};
    cout << "Имя:";
    while (tmp_name[0] == 0) input_str(tmp_name,
MAX_NAME_LEN);

    to_title(tmp_name);
    strcpy_s(data[index].name, tmp_name);
}

    break;
case 2: {
    char tmp_surname[MAX_SURNAME_LEN]{};
    cout << "Фамилия:";
    while (tmp_surname[0] == 0) input_str(tmp_surname,
MAX_SURNAME_LEN);

    to_title(tmp_surname);
    strcpy_s(data[index].sur_name, tmp_surname);
}

    break;
case 3:
    cout << "Отчество:";
    input_str(data[index].last_name, MAX_LAST_NAME_LEN);
    to_title(data[index].last_name);
    break;
case 4:
    cout << "Год рождения:";
    data[index].year_of_birth = input_int(1900, 2021);
    break;
case 5:
    cout << "Группа:";
    data[index].group = input_int(1, MAX_GROUP);

```

```

        break;
    case 6:
        cout << "Оценки:" << endl;
        for (int i{}; i < MARK_COUNT; i++) {
            cout << ">>\tПредмет " << i + 1 << ":";
            data[index].marks[i] = input_int(1, MAX_MARK);
            cout << endl;
        }
        break;
    case 7:
        input_row(data, index);
        break;
    }
    } while (ask != 0);
}
else {
    cout << "Записи ученика с номером " << index + 1 << " не
существует" << endl;

}
cout << "Вернуться в меню (0-Нет / 1-Да): ";
continue_flag = input_fast_menu(1);
} while (continue_flag == 0);

return 0;
}

int delete_note(Student data[]) {
    int continue_flag{};
    do {
        int student_index{};
        system("cls");
        print_header("УДАЛИТЬ ЗАПИСЬ");
        if (get_last_index(data) != 0) {
            cout << "Введите номер ученика (1 - " << get_last_index(data) <<
"): ";
        }
        else {
            cout << "Нет записей для изменения" << endl;
            system("pause");
            return 0;
        }
        student_index = input_int(0, get_last_index(data));
        if (student_index == 0) return 0;
    }
}

```

```

cout << endl;

int index = student_index - 1;
if (data[index].name[0]) {
    for (int i = index; i < get_last_index(data); i++) {
        data[i] = data[i + 1];
    }
    /*
    dest - pointer to the object to fill
    ch - fill byte
    count - number of bytes to fill
    memset(dest, ch, count)
    */
    memset(&data[get_last_index(data)], 0, sizeof(Student));
    cout << "ЗАПИСЬ УДАЛЕНА" << endl;
}
else {
    cout << "Записи ученика с номером " << student_index << " не
существует" << endl;
}

cout << "Вернуться в меню (0-Нет / 1-Да): ";
continue_flag = input_fast_menu(1);
} while (continue_flag == 0);

return 0;
}

int output_display(Student data[]) {
    system("cls");
    print_header("ВЫВОД ДАННЫХ НА ЭКРАН");
    print_hat(cout);
    // Рисуем 1 строку, даже если get_last_index = 0
    int limit = 1;
    if (get_last_index(data) != 0) limit = get_last_index(data);
    for (int i{}; i < limit; i++) {
        output_row(data, i, cout);
    }

    system("pause");

    return 0;
}

```

```

//вывести фамилии и группы студентов, получивших не более двух 4
//(остальные оценки - 5).
void output_on_request(Student data[]) {
    system("cls");
    print_header("ВЫВОД СТУДЕНТОВ, ПОЛУЧИВШИХ НЕ БОЛЕЕ ДВУХ 4 (ОСТАЛЬНЫЕ
ОЦЕНКИ - 5) НА ЭКРАНЕ");
    bool student_list[STUDENTS]{ 1 };
    cout << endl;
    cout << "|";
    centred("Фамилия", MAX_SURNAME_LEN + 1, cout);
    cout << "|";
    centred("Группа", GROUP_WIDTH + 1, cout);
    cout << "|";
    cout << endl;
    bool flag_printed = false;
    for (int index = 0; index < STUDENTS; index++) {
        student_list[index] = true;
        int sum{};
        for (int i = 0; i < MARK_COUNT; i++) {
            sum += data[index].marks[i];
            if (data[index].marks[i] < 4) {
                student_list[i] = false;
            }
        }
        if (sum < 28) {
            student_list[index] = false;
        }
        if (student_list[index] == true) {
            cout << "|";
            centred(data[index].sur_name, MAX_SURNAME_LEN + 1, cout);
            cout << "|";
            centred(data[index].group, GROUP_WIDTH + 1, cout);
            cout << "|";
            cout << endl;

            flag_printed = true;
        }
    }
    if (!flag_printed) {
        cout << "|";
        centred(" ", MAX_SURNAME_LEN + 1, cout);
        cout << "|";
        centred(" ", GROUP_WIDTH + 1, cout);
        cout << "|";
    }
}

```

```

        cout << endl;
        cout << "Нет подходящих учеников" << endl;
    }
    system("pause");
}
#pragma endregion database_operation

#pragma region sorting
void sort_by_name(Student data[]) {
    int last_student = get_last_index(data);
    Student tmp;
    for (int i = 0; i < last_student; i++) {
        for (int j = i + 1; j < last_student; j++) {
            if (strcmp(data[i].name, data[j].name) > 0) {
                tmp = data[i];
                data[i] = data[j];
                data[j] = tmp;
            }
        }
    }
}

void sort_by_sur_name(Student data[]) {
    int last_student = get_last_index(data);
    Student tmp;
    for (int i = 0; i < last_student; i++) {
        for (int j = i + 1; j < last_student; j++) {
            if (strcmp(data[i].sur_name, data[j].sur_name) > 0) {
                tmp = data[i];
                data[i] = data[j];
                data[j] = tmp;
            }
        }
    }
}

void sort_by_last_name(Student data[]) {
    int last_student = get_last_index(data);
    Student tmp;
    for (int i = 0; i < last_student; i++) {
        for (int j = i + 1; j < last_student; j++) {
            if (strcmp(data[i].last_name, data[j].last_name) > 0 &&
data[j].last_name[0] != 0) {
                tmp = data[i];

```



```

        data[i] = data[j];
        data[j] = tmp;
    }
}
}

void sort_by_year_of_birth_up(Student data[]) {
    int last_student = get_last_index(data);
    Student tmp{};
    for (int i = 0; i < last_student; i++) {
        for (int j = i + 1; j < last_student; j++) {
            if (data[i].year_of_birth > data[j].year_of_birth) {
                tmp = data[i];
                data[i] = data[j];
                data[j] = tmp;
            }
        }
    }
}

void sort_by_year_of_birth_down(Student data[]) {
    int last_student = get_last_index(data);
    Student tmp{};
    for (int i = 0; i < last_student; i++) {
        for (int j = i + 1; j < last_student; j++) {
            if (data[i].year_of_birth < data[j].year_of_birth) {
                tmp = data[i];
                data[i] = data[j];
                data[j] = tmp;
            }
        }
    }
}

void sort_by_group_up(Student data[]) {
    int last_student = get_last_index(data);
    Student tmp{};
    for (int i = 0; i < last_student; i++) {
        for (int j = i + 1; j < last_student; j++) {
            if (data[i].group > data[j].group) {
                tmp = data[i];
                data[i] = data[j];
                data[j] = tmp;
            }
        }
    }
}

```

```

        }
    }
}

void sort_by_group_down(Student data[]) {
    int last_student = get_last_index(data);
    Student tmp{};
    for (int i = 0; i < last_student; i++) {
        for (int j = i + 1; j < last_student; j++) {
            if (data[i].group < data[j].group) {
                tmp = data[i];
                data[i] = data[j];
                data[j] = tmp;
            }
        }
    }
}

```

```

int sort_alpha_menu(Student data[]) {
    system("cls");
    print_header("СОРТИРОВКА ПО АЛФАВИТУ");
    cout << "1| Сортировать по имени" << endl;
    cout << "2| Сортировать по фамилии" << endl;
    cout << "3| Сортировать по отчеству" << endl;
    cout << "0| Назад" << endl;
    cout << "Введите номер пункта меню: ";
    int ask = input_fast_menu(3);
    cout << endl;

    switch (ask) {
        case 1:
            sort_by_name(data);
            break;
        case 2:
            sort_by_sur_name(data);
            break;
        case 3:
            sort_by_last_name(data);
            break;
        default:
            return 0;
            break;
    }
}

```

```

cout << "Сортировка завершена" << endl;
system("pause");

return 0;
}

int sort_year_of_birth_menu(Student data[]) {
    system("cls");
    print_header("СОРТИРОВКА ПО ГОДУ РОЖДЕНИЯ");
    cout << "1| Сверху старше" << endl;
    cout << "2| Сверху младше" << endl;
    cout << "0| Назад" << endl;
    cout << "Введите номер пункта меню: ";
    int ask = input_fast_menu(2);
    cout << endl;

    switch (ask) {
    case 1:
        sort_by_year_of_birth_up(data);
        break;
    case 2:
        sort_by_year_of_birth_down(data);
        break;
    default:
        return 0;
        break;
    }
    cout << "Сортировка завершена" << endl;
    system("pause");

    return 0;
}

int sort_group_menu(Student data[]) {
    system("cls");
    print_header("СОРТИРОВКА ПО ГРУППЕ");
    cout << "1| Сверху номер группы больше" << endl;
    cout << "2| Сверху номер группы меньше" << endl;
    cout << "0| Назад" << endl;
    cout << "Введите номер пункта меню: ";
    int ask = input_fast_menu(2);
    cout << endl;

    switch (ask) {

```

```

case 1:
    sort_by_group_down(data);
    break;
case 2:
    sort_by_group_up(data);
    break;
default:
    return 0;
    break;
}
cout << "Сортировка завершена" << endl;
system("pause");

return 0;
}
#pragma endregion sorting

#pragma region file_operation
int write_text_file(Student data[], int rows, char filename[]) {
    ofstream fout(filename, ios_base::out | ios_base::trunc);
    if (!fout.is_open()) return 1;
    for (int index = 0; index < rows; index++) {
        fout << data[index].name << endl;
        fout << data[index].sur_name << endl;
        if (strcmp(data[index].last_name, "") == 0 ||
            strcmp(data[index].last_name, "-") == 0) {
            fout << "-" << endl;
        }
        else {
            fout << data[index].last_name << endl;
        }
        fout << data[index].group << endl;
        fout << data[index].year_of_birth << endl;
        for (int i = 0; i < MARK_COUNT; i++) {
            fout << data[index].marks[i] << endl;
        }
        fout << endl;
    }
    fout.close();

    return 0;
}

int write_text_file_menu(Student data[]) {

```

```

system("cls");
print_header("ВЫВОД ДАННЫХ В ТЕКСТОВЫЙ ФАЙЛ");
cout << "Введите имя файла:";

char filename[FILE_NAME_LEN + 4] = "";
input_str(filename, FILE_NAME_LEN, "-_+0123456789()", 0);
if (strcmp(filename, "") == 0) return 0;
strcat_s(filename, ".txt");
cout << endl;

switch (write_text_file(data, get_last_index(data), filename)) {
case 0:
    cout << "Файл \"" << filename << "\" записан" << endl;
    break;
case 1:
    cout << "Не удалось открыть файл \"" << filename << "\"" << endl;
    break;
}
system("pause");

return 0;
}

int counvert_bin_file(Student data[], char in_filename[], char
out_filename[]) {
    ifstream fin(in_filename, ios::in);
    ofstream fout(out_filename, ios::binary | ios::out);
    if (!fin.is_open()) return 1;
    if (!fout.is_open()) return 2;

    int index{};
    while (!fin.eof()) {
        Student tmp;
        fin >> tmp.name;
        fin >> tmp.sur_name;
        fin >> tmp.last_name;
        fin >> tmp.group;
        fin >> tmp.year_of_birth;
        for (int i = 0; i < MARK_COUNT; i++) {
            fin >> tmp.marks[i];
        }
        /*
        s      -   pointer to the character string to write
        count   -   number of characters to write
    */
    }
}

```

```

        write(s, count)
        */
        fout.write((char*)&tmp, sizeof(Student));
        index++;
    }

    fin.close();
    fout.close();

    return 0;
}

int convert_to_bin_file_menu(Student data[]) {
    system("cls");
    print_header("ПЕРЕВОД ДАННЫХ ТЕКСТОВОГО ФАЙЛА В БИНАРНЫЙ ФАЙЛ");
    cout << "Введите имя текстового файла:";

    char in_filename[FILE_NAME_LEN + 4] = "";
    input_str(in_filename, FILE_NAME_LEN, "-_+0123456789()", 0);
    if (strcmp(in_filename, "") == 0) return 0;
    cout << endl;

    strcat_s(in_filename, ".txt");
    cout << "Введите имя бинарного файла:";
    char out_filename[FILE_NAME_LEN] = "";
    input_str(out_filename, FILE_NAME_LEN, "-_+0123456789().", 0);
    if (strcmp(out_filename, "") == 0) return 0;
    cout << endl;

    switch (counvert_bin_file(data, in_filename, out_filename)) {
    case 0:
        cout << "Файл \"" << out_filename << "\" записан" << endl;
        break;
    case 1:
        cout << "Не удалось открыть файл \"" << in_filename << "\"" << endl;
        break;
    case 2:
        cout << "Не удалось открыть файл \"" << out_filename << "\"" << endl;
        break;
    }
    system("pause");

    return 0;
}

```

```

int read_text_file(Student data[], char filename[]) {
    ifstream fin(filename, ios::in | ios::binary);
    if (!fin.is_open()) return 1;
    int index{};
    char a{};
    while (!fin.eof()) {
        Student tmp;
        fin >> tmp.name;
        fin >> tmp.sur_name;
        fin >> tmp.last_name;
        fin >> tmp.group;
        fin >> tmp.year_of_birth;
        for (int i = 0; i < MARK_COUNT; i++) {
            fin >> tmp.marks[i];
        }
        data[index] = tmp;
        index++;
    }
    fin.close();

    return 0;
}

int read_text_file_menu(Student data[]) {
    system("cls");
    print_header("ВВОД С ИНФОРМАЦИИ ИЗ УЖЕ СОЗДАННОГО ТЕКСТОВОГО ФАЙЛА");
    cout << "Введите имя файла:";

    char filename[FILE_NAME_LEN + 4] = "";
    input_str(filename, FILE_NAME_LEN, "-_+0123456789()", 0);
    if (strcmp(filename, "") == 0) return 0;
    strcat_s(filename, ".txt");
    cout << endl;

    switch (read_text_file(data, filename)) {
        case 0:
            cout << "Данные из файл \"" << filename << "\" считаны" << endl;
            break;
        case 1:
            cout << "Не удалось открыть файл \"" << filename << "\"" << endl;
            break;
    }
    system("pause");
}

```

```

return 0;
}

int read_bin_file(Student data[], char filename[]) {
    ifstream fin(filename, ios::in | ios::binary);
    if (!fin.is_open()) return 1;
    int index{};
    while (!fin.eof()) {
        fin.read((char*)&data[index], sizeof(Student));
        index++;
    }
    fin.close();

    return 0;
}

int read_bin_file_menu(Student data[]) {
    system("cls");
    print_header("ВВОД С ИНФОРМАЦИИ ИЗ УЖЕ СОЗДАННОГО БИНАРНОГО ФАЙЛА");
    cout << "Введите имя файла:";

    char filename[FILE_NAME_LEN + 4] = "";
    input_str(filename, FILE_NAME_LEN, "-_+0123456789() .", 0);
    if (strcmp(filename, "") == 0) return 0;
    cout << endl;

    switch (read_bin_file(data, filename)) {
        case 0:
            cout << "Данные из файл \"" << filename << "\" считаны" << endl;
            break;
        case 1:
            cout << "Не удалось открыть файл \"" << filename << "\"" << endl;
            break;
    }
    system("pause");

    return 0;
}

int write_file_table(char filename[], Student data[], int rows) {
    ofstream fout(filename, ios_base::out | ios_base::trunc);
    if (!fout.is_open()) return 1;
    print_hat(fout);

```



```

for (int index = 0; index < rows; index++) {
    output_row(data, index, fout);
}

fout.close();

return 0;
}

int write_file_table_menu(Student data[]) {
    system("cls");
    print_header("ПЕРЕВОД БАЗЫ ДАННЫХ В ВИДЕ ТАБЛИЦЫ В ТЕКСТОВЫЙ ФАЙЛ");
    cout << "Введите имя текстового файла:";

    char filename[FILE_NAME_LEN + 4] = "";
    input_str(filename, FILE_NAME_LEN, "-_+0123456789()", 0);
    if (strcmp(filename, "") == 0) return 0;
    cout << endl;
    strcat_s(filename, ".txt");

    switch (write_file_table(filename, data, get_last_index(data))) {
    case 0:
        cout << "Файл \"" << filename << "\" записан" << endl;
        break;
    case 1:
        cout << "Не удалось открыть файл \"" << filename << "\"" << endl;
        break;
    }
    system("pause");

    return 0;
}

#pragma endregion file_operation

```