|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  «МИРЭА – Российский технологический университет»  РТУ МИРЭА |

Институт комплексной безопасности и специального приборостроения

Кафедра КБ-2 «Прикладные информационные технологии»

КУРСОВОЙ ПРОЕКТ (РАБОТА)

по дисциплине «Языки программирования»

(наименование дисциплины)

Тема курсового проекта Разработка защищенной файловой системы на основе ООП

Студент группы БББО-05-20 Кутьин Захар Сергеевич

(учебная группа, фамилия, имя, отчество студента) (подпись студента)

Руководитель курсового проекта ст.пр. Мерсов А.А.

(должность, звание, ученая степень) (подпись руководителя)

Рецензент (при наличии)

(должность, звание, ученая степень) (подпись рецензента)

Работа представлена к защите « » 2021 г.

Допущен к защите « » 2021 г.

|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  «МИРЭА – Российский технологический университет»  РТУ МИРЭА |

Институт комплексной безопасности и специального приборостроения

Кафедра КБ-2 «Прикладные информационные технологии»

Утверждаю

Заведующий кафедрой

(Фамилия, Имя, Отчество) (подпись).

« » 2021 г.

ЗАДАНИЕ

на выполнение курсового проекта (работы) по дисциплине

«Языки программирования»

Студент Кутьин Захар Сергеевич Группа БББО-05-20

Тема: Разработка защищенной файловой системы на основе ООП в соответствие с вариантом: разбить группу на 2 части, с поиском среди лиц с указанием интервала года

рождения.: 1) по алфавиту от А до П; 2) по алфавиту от Р до Я. Каждую часть отсортировать в порядке увеличения среднего бала за все время обучения.

Перечень вопросов, подлежащих разработке и обязательного графического материала:

реализация загрузки данных с клавиатуры, реализация функции записи и чтения информации виз файл(а), использование динамической памяти, реализация функции добавления, удаления или изменения записей в файле, использование конструкторов и деструкторов, использование друзей классов, реализация наследования классов, реализация перегрузки операторов, использование виртуальных функций, реализация шифрования и дешифрования файла.

Срок представления к защите курсового проекта (работы) до « » 2021 г.

Задание на выполнение курсовой проект (работу) выдал Мерсов А.А.

(Ф.И.О. руководителя) (Подпись руководителя)

« » 2021 г.

Задание на курсовой проект (работу) получил Кутьин З.С.

(Ф.И.О. исполнителя) (Подпись обучающегося)

Оглавление

[Введение 4](#_Toc74065976)

[Исходные данные 5](#_Toc74065977)

[Сведения из теории. 6](#_Toc74065978)

[Перечень используемых структур, классов 13](#_Toc74065979)

[Блок-схема программы 44](#_Toc74065980)

[Руководство пользователя 45](#_Toc74065981)

[Обработка исключительных ситуаций. 58](#_Toc74065982)

[Контрольные данные 65](#_Toc74065983)

[Скриншоты выполнения работы по тестовым данным 66](#_Toc74065984)

[Анализ полученных результатов 68](#_Toc74065985)

[Вывод 70](#_Toc74065986)

[Список литературы 71](#_Toc74065987)

[Листинг 72](#_Toc74065988)

# Введение

В настоящее время в высших учебных заведениях обучается большое количество студентов. Однако для хранения информации о них требуется большое количество бумажных ресурсов, что довольно неразумно в настоящее время. Поэтому для удобства работы с большим количеством информации, были поставлены следующие цели и задачи для разработки файловой базы студентов:

**Цель курсовой работы:**

Закрепленить и углубить знания и навыки при работе с языком программирования C++, при разработке и тестировании ПО.

**Задачи курсовой работы:**

1. Изучить литературу и документацию по разработке в среде программирования Visual Studio при использовании языка программирования C++.
2. Разработать файловую базу данных на основе элементов ООП.
3. Разработать программу, на работу которой не будут влиять:
   1. Неправильно введенные данные
   2. Случайно нажатые клавиши
4. Реализовать при разработке следующие пункты:
   1. Изменение, удаление, добавление любых данных
   2. Ввод/вывод информации на экран
   3. Осуществление задания согласно варианту
5. Оформленить документацию по разработке и использованию ПО.

# Исходные данные

1. Информация о группе студентов из N человек, где запись о студенте содержит следующие данные:
   1. Ф. И. О. студента.
   2. Число, месяц, год рождения.
   3. Пол.
   4. Год поступления в институт.
   5. Факультет, кафедра.
   6. Группа.
   7. Номер студенческого билета.
   8. Названия предметов и оценки по каждому предмету в каждой сессии (максимально 9 сессий по 10 предметов в каждом семестре).
2. Конкретное задание, вариант которого определяется 2-мя последними цифрами зачетной книжки:

Вариант 97.Разбить группу на 2 части, с поиском среди лиц с указанием интервала года рождения: 1) по алфавиту от А до П; 2) по алфавиту от Р до Я. Каждую часть отсортировать в порядке увеличения среднега6о бала за все время обучения.

# Сведения из теории.

**Директивы препроцессора.**

Директива #define определяет идентификатор и последовательность символов, которой будет замещаться данный идентификатор при его обнаружении в тексте программы.

При подключении директивы #include, препроцессор копирует содержимое подключаемого файла в текущий файл сразу после строки с #include.

**Динамические массивы. Указатели.**

Указатель – это переменная, в которой хранится адрес другой переменной или участка памяти. Основная информация об указателях:

1. при объявлении указателя надо указать тип переменных, на которых он будет указывать, а перед именем поставить знак \*;

2. знак & перед именем переменной обозначает ее адрес;

3. знак \* перед указателем в рабочей части программы (не в объявлении) обозначает значение ячейки, на которую указывает указатель;

4. для обозначения недействительного указателя используется константа nullptr.

Использовать динамическое выделение памяти – значит сначала определить, сколько чисел в массиве, а потом выделить ровно столько памяти, сколько нужно.

Динамическими называются массивы, размер которых неизвестен на этапе написания программы. Существуют стандартные способы выделения и освобождения памяти в языке Си ++ (с помощью операторов new и delete).

**Файлы.**

Файл – это именованная область ячеек памяти, в которой хранятся данные одного типа.

Двоичный файл – файл, данные которого представлены в бинарном виде. При записи в двоичный файл символы и числа записываются в виде последовательности байт (в своем внутреннем двоичном представлении в

памяти компьютера). Для работы с двоичными файлами используется стандартная библиотека.

Имя файла – указатель на строку символов, представляющую собой допустимое имя файла.

Поток – это абстрактное понятие, относящееся к любому переносу данных от источника к приемнику.

Для потоковой работы с файлом используется библиотека fstream.

Классы библиотеки fstream:

• ifstream – открыть файл на чтение

• ofstream – открыть файл на запись

Флаги для открытия файлов:

• ios::binary – открыть файл бинарно

• ios::in – открыть файл на считывание

• ios::out – открыть файл на запись

• ios::app – открыть файл на запись данных в его конец

Флаги позиции смещения:

• ios::beg – начало файла

• ios::cur – текущее положение

• ios::end – конец файла

Для работы с файлами используются следующие стандартные функции:

• open() – метод открытия файла.

• сlose() – метод закрытия файла.

• int remove(const char \*filename) – функция удаления файла. Эта функция удаляет с диска файл, указатель на который хранится в файловой переменной filename.

• int rename(const char \*oldfilename, const char \*newfilename) – функция переименования файла. Первый параметр – старое имя файла, второй – новое.

Класс ifstream - предоставляет возможности для чтения файлов. Открыть файл можно двумя способами: вызвав метод open() или указав путь к нему в конструкторе. Считывание из бинарного файла производить лучше всего с

помощью метода read(), в который передается имя переменной, в которую будет произведено считывание, и количество байт для считывания.

Класс ofstream - предоставляет возможности для записи файлов. Открыть файл можно двумя способами: вызвав метод open() или указав путь к нему в конструкторе. Запись в бинарный файл производить лучше всего с помощью метода write(), в который передается имя переменной, из которой будут считаны данные для записи, и количество байт для записи. get() и getline() – методы записи строк (отличаются тем, что функция get() вместо того, чтобы читать и отбрасывать символ новой строки, оставляет этот символ во входной очереди).

**Элементы ООП.**

Наследование - наличие в языке ООП механизма, позволяющего объектам класса наследовать характеристики более простых и общих типов. Наследование обеспечивает как требуемый уровень общности, так и необходимую специализацию.

Полиморфизм - дословный перевод с греческого "много форм". В С++ полиморфизм реализуется с помощью виртуальных функций, которые позволяют в рамках всей иерархии классов иметь несколько версий одной и той же функции. Решение о том, какая именно версия должна выполняться в данный момент, определяется на этапе выполнения программы и носит название позднего связывания.

**Классы.**

Основное отличие С++ от С состоит в том, что в С++ имеются классы. С точки зрения языка С классы в С++ - это структуры, в которых вместе с данными определяются функции. Это и есть инкапсуляция в терминах ООП.

Класс (class) – это тип, определяемый пользователем, включающий в себя данные и функции, называемые методами или функциями-членами класса.

Данные класса – это то, что класс знает.

Функции – члены (методы) класса - это то, что класс делает.

Область видимости public разрешает доступ к полям и методам класса из любой части программы, в которой известен этот объект (общедоступный).

Область видимости private разрешает доступ к полям и методам класса только из методов этого класса. Объекты с такой областью видимости называют частными.

Конструктор – это метод класса, имя которого совпадает с именем класса. Конструктор вызывается автоматически после выделения памяти для переменной и обеспечивает инициализацию компонент – данных. Для класса может быть объявлено несколько конструкторов, различающихся числом и типами параметров.

Деструктор - деструктор вызывается перед освобождением памяти, занимаемой объектной переменной, и предназначен для выполнения дополнительных действий, связанных с уничтожением объектной переменной, например, для освобождения динамической памяти, закрытия, уничтожения файлов и т.п.

Один класс может быть дружественным другому классу. Это откроет всем членам первого класса доступ к закрытым членам второго класса. Такой класс называется дружественным классом. Для определения дружественных классов используется ключевое слово friend.

**Перегрузка операций.**

Для перегрузки операции для класса в С++ используется следующий синтаксис:

<Тип> operator <операция> (<входные параметры>)

{

<операторы>;

}

где < Тип > - тип, возвращаемый функцией;

operator - ключевое слово;

< операция > - перегружаемая операция.

В языке С++ имеются следующие ограничения на перегрузку операций:

· С++ не различает префиксную и постфиксную формы ++ и - -;

· переопределяемая операция должна присутствовать в языке (например, нельзя определить операцию с символом #);

· нельзя переопределить операторы, заданные следующими символами . \* :: ? ;

· переопределённые операции сохраняют свой изначальный приоритет.

**Наследование классов, механизм виртуальных функций.**

Наследование - механизм создания производного класса из базового, т.е. к существующему классу можно что-либо добавить или изменять его каким-либо образом для создания нового (производного) класса.

При определении производного класса не наследуются из базового:

1. конструкторы;

2. деструкторы;

3. операторы new, определенные пользователем;

4. операторы присвоения, определенные пользователем;

5. отношения дружественности.

Виртуальная функция объявляется в базовом или в производном классе и, затем, переопределяется в наследуемых классах. Совокупность классов (подклассов), в которых определяется и переопределяется виртуальная функция, называется полиморфическим кластером, ассоциированным с некоторой виртуальной функцией. В пределах полиморфического кластера сообщение связывается с конкретной виртуальной функцией-методом во время выполнения программы.

Обычную функцию-метод можно переопределить в наследуемых классах. Однако без атрибута virtual такая функция-метод будет связана с сообщением на этапе компиляции. Атрибут virtual гарантирует позднее связывание в пределах полиморфического кластера.

**Некоторые криптографические элементы.**

CryptoAPI — интерфейс программирования приложений, который обеспечивает разработчиков Windows-приложений стандартным набором функций для работы с криптопровайдером. CryptoAPI поддерживает работу с асимметричными и симметричными ключами, то есть позволяет шифровать и расшифровывать данные. Набор поддерживаемых криптографических алгоритмов зависит от конкретного криптопровайдера.

Любой сеанс работы с CryptoAPI начинается с инициализации (получения контекста). Инициализация выполняется при помощи функции CryptAcquireContext.

Криптопровайдер – это независимый модуль, содержащий библиотеку криптографических функций со стандартизованным интерфейсом.

Базовая функция шифрования данных имеет следующее объявление:

BOOL CryptEncrypt(HCRYPTKEY hKey,

HCRYPTHAS hHash,

BOOL Final,

DWORD dwFlags,

BYTE\* pbData,

DWORD\* pdwDataLen,

DWORD dwBufLen);

Эта функция может обрабатывать данные блоками. То есть достаточно передавать массив данных поблочно, специальным образом отметив лишь последний блок данных. В данной функции для этого служит третий параметр Final.

Базовая функция десшифровывания имеет следующее описание:

BOOL CryptDecrypt(HCRYPTKEY hKey,

HCRYPTHASH hHash,

BOOL Final,

DWORD dwFlags,

BYTE\* pbData,

DWORD\* pdwDataLen);

Первым параметром данной функции передается инициализированный контекст сессионного ключа, применяемого для расшифровывания данных. Параметры pbData и pdwDataLen используются точно так же, как и у

CryptEncrypt и представляют собой ссылку на входной/выходной массив данных и длину этого массива данных.

Генерация ключей и обмен ключами

Для генерации ключей в CryptoAPI предусмотрены две функции –CryptGenKey и CryptDeriveKey. Первая из них генерирует ключи случайным образом, а вторая – на основе пользовательских данных.

Создание хэша из пользовательских данных:

Функция CryptHashData добавляет данные в указанный хэш-объект . Перед вызовом этой функции необходимо вызвать CryptCreateHash для создания дескриптора хэш-объекта.

Синтаксис:

BOOL CryptHashData(

HCRYPTHASH hHash,

const BYTE \*pbData,

DWORD dwDataLen,

DWORD dwFlags

);

Параметры фунции:

· hHash - Дескриптор хеш-объекта.

· pbData - Указатель на буфер, содержащий данные, которые нужно добавить к хеш-объекту.

· dwDataLen - Количество байтов данных, которые нужно добавить. Он должен быть равен нулю, если установлен флаг CRYPT\_USERDATA.

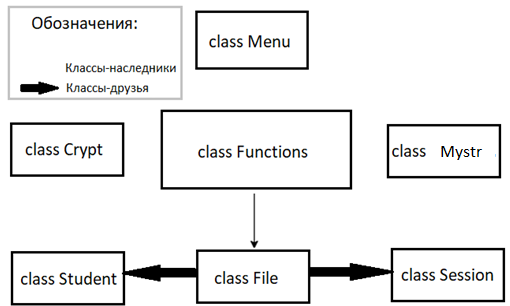
· dwFlags – как правило данный параметр не используется

Если функция завершается успешно, возвращаемое значение - ИСТИНА.

В случае сбоя функции возвращается значение FALSE.

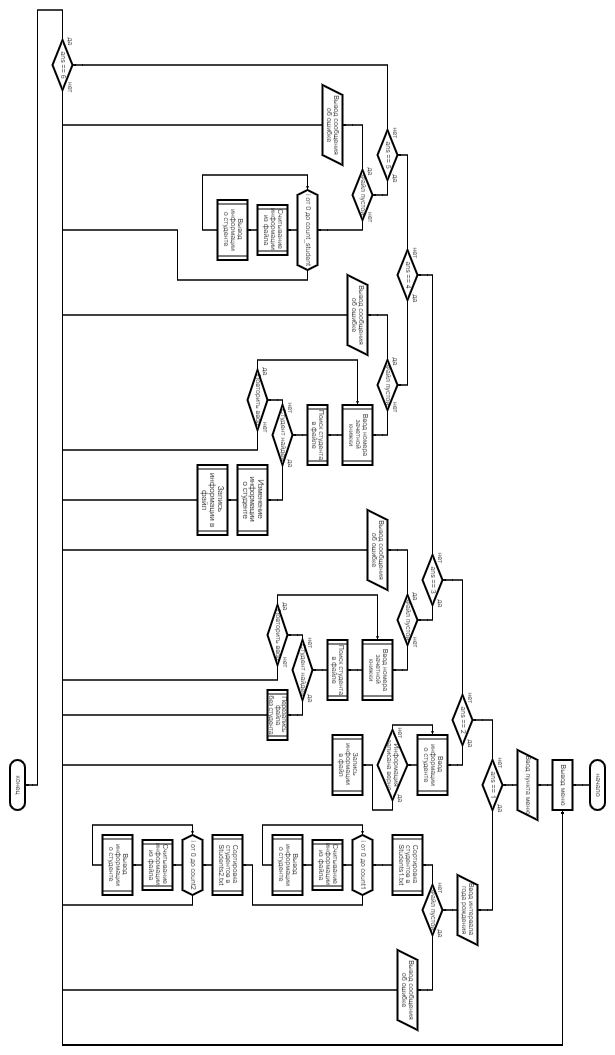
# Перечень используемых структур, классов

**Схема классов:**



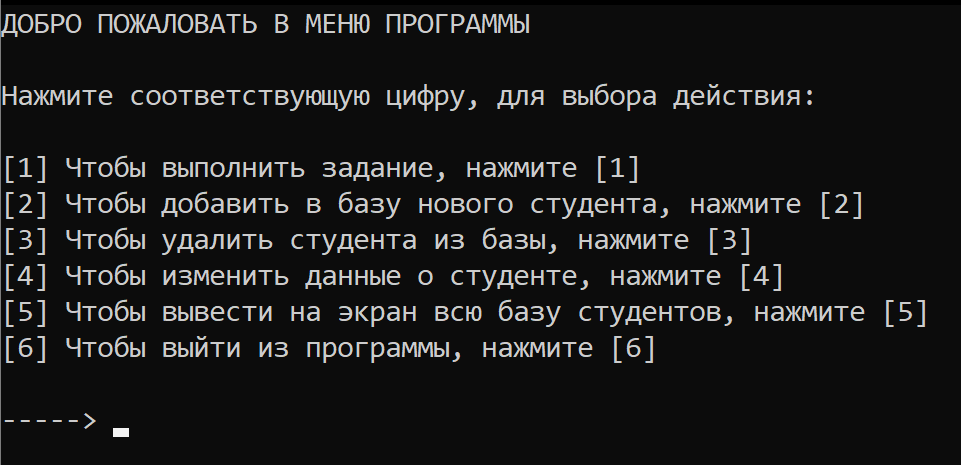
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Тип объекта** | **Название** | **Описание** | **Элементы** | **Описание элементов** |
| **Структура** | **Sub** | Структура для описания предмета в какой-либо сессии студента, содержит имя предмета и оценку | **char\* name** | Указатель на массив символов, содержащих имя предмета |
| **int\* mark** | Указатель на целочисленную переменную, хранящую оценку по этому предмету |
| **Класс** | **Functions** | Основной класс программы. Он наследуется другими классами в программе, содержит функции вывода га экран, а также методы, позволяющие правильно работать в программе в любых ситуациях. | **void print()** | Перегружаемая функция для вывода информации в консоль. Виды функции: **void print (char\* val)** – служит для вывода массива символов; **void print(int val)** – служит для вывода целочисленной информации; **void print(const char val[])** – служит для вывода массива символов.  **Val –** целочисленная переменная или указатель на массив символов, или строковая констаната, параметр, который будет выведен на экран. |
| **void cleanCin()** | Служебная функция, служит для сброса флагов ввода, а также для перемещения указателя в конец потока ввода. Предотвращает неправильную работу программы. |
| **void checkEmptyStr(char\* in, int len)** | Функция для проверки ввода (введены ли символы, пустая ли строка) и ввода символов  **char\* in –** указатель на массив символов, который проверяется функцией.  **int len —** целочисленная переменная, размер массива **in**. |
| **bool checkMinus** **(char\* in, int len)** | Функция, проверяющая, есть ли в начале/конце массива символ «-». Если символ «-» есть в начале/конце массива возвращает **false**, иначе **true.**  **char\* in –** указатель на массив символов, который проверяет функция.  **int len –** целочисленная переменная, размер массива **in**. |
| **void backMenu()** | Функция, которая ждет нажатия клавиши ENTER для выхода в меню |
| **void checkValue(int value)** | Функция, которая проверяет, что ввели число.  **int value –** значение, которое проверяет функция. |
| **bool checkMasChar(char\* line, int n)** | Функция, которая проверяет, что ввели символы.  Если были введены числа или буквы латинского алфавита возвращает **false**, иначе **true**.  **char\* line –** указатель на массив символов, который проверяет функция.  **int n –** целочисленная переменная, размер массива **line.** |
| **Класс** | **Student** | Класс для хранения основной информации о студенте, является дочерним классов Functions. Является дружественным классом File | **Private:**  **char\* surname** | Указатель на массив символов для хранения фамилии студента |
| **char\* name** | Указатель на массив символов для хранения имени студента |
| **char\* middlename** | Указатель на массив символов для хранения отчества студента |
| **int\* day** | Указатель на целочисленную переменную для хранения дня рождения студента |
| **int\* month** | Указатель на целочисленную переменную для хранения месяца рождения студента |
| **int\* year** | Указатель на целочисленную переменную для хранения года рождения студента |
| **int\* yearOfAdmission** | Указатель на целочисленную переменную для хранения года зачисления студента в университет |
| **char\* sex** | Указатель на массив символов для хранения пола студента |
| **char\* faculty** | Указатель на массив символов для хранения факультета, на котором обучается студент |
| **char\* departament** | Указатель на массив символов для хранения кафедры, на которой обучается студент. |
| **char\* group** | Указатель на массив символов для хранения группы, в которой обучается студент |
| **char\* recordBookNumber** | Указатель на массив символов для хранения номера зачетной книжки студента |
| **void Set(char\* in, int len)** | Метод для присвоения значений полям класса.  **char\* in –** указатель на массив символов, который будет присвоен полю класса.  **int len –** целочисленная переменная, длина массива **in.** |
| **bool checkBirthDay()** | Метод для ввода даты рождения студента. Также функция проверяет, что в дате рождения нет лишних символов. Если все проверки выполнены и дата рождения записалась возвращает **true**, иначе **false**. |
| **void checkAdmissionYear()** | Метод для ввода года приема студента в университет и проверки корректности ввода. |
| **bool checkRecordBookNumber()** | Метод для проверки существования студента с таким же номером зачетной книжки. Если не существует возвращает **true**, в противном случае **false**. |
| **bool checkDate(int day, int month, int year)** | Метод для проверки корректного ввода даты рождения. Если корректно возвращает **true**, в противном случае **false**.  **int day –** целочисленная переменная, отвечающая за день.  **int month –**целочисленная переменная, отвечающая за месяц.  **int year –**целочисленная переменная, отвечающая за год. |
| **char checkSex()** | Метод для проверки корректности введенного пола студента. Возвращает значение пола студента. |
| **Public:**  **Student()** | Конструктор класса. Вызывается при создании объекта. Создает указатели на все необходимые переменные, описанные в области **private.** |
| **~Student()** | Деструктор класса, удаляет всю информацию, хранящуюся в объекте класса. Вызывается при уничтожении класса. |
| **void Set()** | Метод для записи всей информации о студенте в поля класса. |
| **bool Edit()** | Метод для изменения информации о студенте. В момент вызова, когда пользователь выберет «11», возвращает **true**, иначе **false**. |
| **Класс** | **Session** | Класс для хранения информации о сессиях студента, является дочерним классов Functions. Является дружественным классом File | **Private:**  **int\* session\_count** | Указатель на целочисленную переменную, хранящую количество сессий студента. |
| **int\* sub\_count** | Указатель на динамический массив целых чисел, хранящий количество предметов в каждой сессии. |
| **Sub subject[9][10]** | Статический двумерный массив структур Sub. |
| **void setSessionCount()** | Функция, устанавливающая количество сессий. |
| **void setSubjCount()** | Функция, устанавливающая количество предметов в сессии. |
| **void setSubjects()** | Функция, устанавливающая названия предметов и оценку за них. |
| **Public:**  **Session()** | Конструктор класса Session. Вызывается при создании объекта. Создает указатели на все необходимые переменные, описанные в области **private.** |
| **~Session()** | Деструктор класса, удаляет всю информацию, хранящуюся в объекте класса. Вызывается при уничтожении класса. |
| **void setSession()** | Метод для ввода информации о сессиях студента. |
| **bool editSession()** | Метод для изменения информации о сессиях или предметах в сессиях студента. Возвращет **true**, если изменения информации о сессии и предметах прошли успешно, возвращает **false**, если в процессе изменения произогли какие-либо ошибки. |
| **Класс** | **File** | Класс, наследник Functions, предназначен для работы с файлом | **Private:**  **Student\* edit\_student** | Указатель на объект класса **Student** |
| **Session\* edit\_session** | Указатель на объект класса **Session** |
| **int\* file\_length** | Указатель на переменную целого типа, хранящую размер файла. |
| **int pos** | Указатель на переменную целого типа, хранящую порядковый номер студента в файле |
| **int count1** | Целочисленная переменная, хранящая количество студентов, записанных в файл и имеющих фамилию, начинающуюся от А до П. |
| **int count2** | Целочисленная переменная, хранящая количество студентов, записанных в файл и имеющих фамилию, начинающуюся от Р до Я. |
| **int\* count** | Указатель на переменную целого типа, хранящую количество студентов, записанных в файл. |
| **int\* sum** | Указатель на переменную целого типа, в которой хранится общее количество предметов во всех сессиях у студента. |
| **char\* rec\_book\_num** | Указатель на массив символов, в которой записывается номер зачетной книжки студента, информацию о котором нужно удалить или изменить. |
| **bool findStudent()** | Метод для поиска информации о студенте по номеру его зачётной книжки. Возвращает значение **false**, если поиск не был осуществлен, и **true** в противном случае. |
| **void getStudent(const char\* name\_file)** | Метод для считывания информации о студенте из файла.  **const char\* name\_file –** указатель на const массив символов, в котором указывается имя файла, из которого будет считываться информация. |
| **void fileWriter(const char\* name\_file)** | Метод для записи информации о студентах в файл.  **const char\* name\_file –** указатель на const массив символов, в котором указывается имя файла, в который будет записываться информация. |
| **bool isThereStudent()** | Метод для подсчета количества студентов, информация о которых записана в файл. Возвращает **false** если файл пустой, в противном случае **true**. |
| **void studentsPrint(int rez)** | Функция, которая выводит на экран информацию о студентах.  **int rez** – целочисленная переменная,отвечает за то, какую информацию выводить:  1 – вывод краткой информации о студенте;  2 – вывод полной информации о студенте без оценок;  3 – вывод оценок студента. |
| **Public:**  **File()** | Конструктор класса File. Вызывается при создании объекта. Создает указатели на все необходимые переменные, описанные в области **private.** |
| **~File()** | Деструктор класса, удаляет всю информацию, хранящуюся в объекте класса. Вызывается при уничтожении класса. |
| **void addStudent()** | Метод для добавления информации о новом студенте в программу. |
| **void editStudent()** | Метод для изменения информации о студенте. |
| **void delStudent()** | Метод для удаления информации о студенте из программы. |
| **void printChoice (int rez)** | Функция, которая в зависимости от выбора пользователя выводит информацию о студентах.  **int rez** – целочисленная переменная,отвечает за то, какую информацию выводить:  1 – вывод краткой информации о студенте;  2 – вывод полной информации о студенте без оценок;  3 – вывод краткой информации о студенте и его оценок;  4 - вывод полной информации о студенте и его оценок. |
| **void task(int up, int down)** | Функция, отвечающая за выполнение задания. Записывает в Students1.txt студентов, фамилия которых начинается от А до П, а в Students2.txt студентов, фамилия которых начинается от Р до Я. После сортировки студентов в этих фалйлах выводит студентов на экран.  **int up –** целочисленная переменная, отвечает за верхний предел года рождения для вывода студентов на экран.  **int down –** целочисленная переменная, отвечает за нижний предел года рождения для вывода студентов на экран. |
| **void sort(const char\* file\_name, int count)** | Функция сортировки студентов по возрастанию среднего балла за все сессии в файле.  **const char\* file\_name –** указатель на const массив символов, в котором указывается имя файла, в котором будет происходить сортировка студентов.  **int count –** целочисленная переменная, количество студентов, которые будут отсортированы в файле. |
| **bool min\_sr(Session& t1, Session& t2)** | Функция, которая получает на вход две сессии, если средний балл первой меньше возвращает **true**, в противном случае **false.**  **Session& t1 –** адрес объекта класса Session, сессия студента 1;  **Session& t2 –** адрес объекта класса Session, сессия студента 1. |
| **Класс** | **Mystr** | Класс для обработки массива символов. | **private:**  **char\* data** | Указатель на динамический массив символов. |
| **public:**  **Mystr(const char in[])** | Конструктор класса с параметром. Записывает параметр в **data**.  **const char in[] –** const массив символов, который перезаписывается в **data**. |
| **~Mystr()** | Деструктор класса |
| **void operator += (const char other[])** | Метод перегрузки оператора **+=**, в результате своего выполнения происходит конкатенация строки **data** и строки, которая подается параметром.  **Const char other[] –** const массив символов, который будет добавлен к массиву символов **data**. |
| **char\* Get()** | Метод для получения данных из переменной **data**. Возвращает значение **data**. |
| **Класс** | **Crypt** | Класс для шифровки, дешифровки файла с данными | **Private:**  **char\* Gen\_pass()** | Метод для генерации случайного 16-значного пароля. Возвращает 16-значный пароль. |
| **Public:**  **void Encrypt()** | Метод для осуществления шифрования файла. |
| **void Decrypt()** | Метод для осуществления дешифрования файла. |
| **Класс** | **Menu** | Класс, который отвечает за навигацию в программе и отрисовку меню, является наследником от класса **Functions** | **Menu()** | Конструктор класса. Вызывается при создании объекта. Создает указатели на все необходимые переменные, описанные в области **private.** |
| **~Menu()** | Деструктор класса, удаляет динамическую переменную **data**. Вызывается при уничтожении класса. |
| **bool hub()** | Метод для вывода пунктов меню. Если пользователь введет «6», возвращает **false**, иначе возвращает **true**. |
| **File\* file** | Указатель на объект класса **File.** |
| **int\* ans** | Указатель на целочисленную переменную, которая отвечает за то, в какой пункт меню перейдет пользователь. Вводится с клавиатуры.  1 – перейти в меню выполнения задания;  2 - перейти в меню добавления студента;  3 - перейти в меню удаления студента;  4 - перейти в меню редактирования информации о студенте;  5 - перейти в меню выода информации о студентах;  6 – выйти из программы. |

# Блок-схема программы

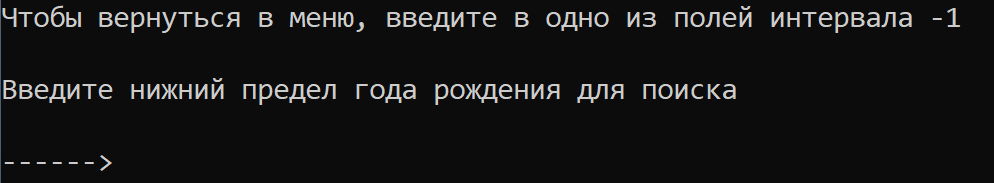


# Руководство пользователя

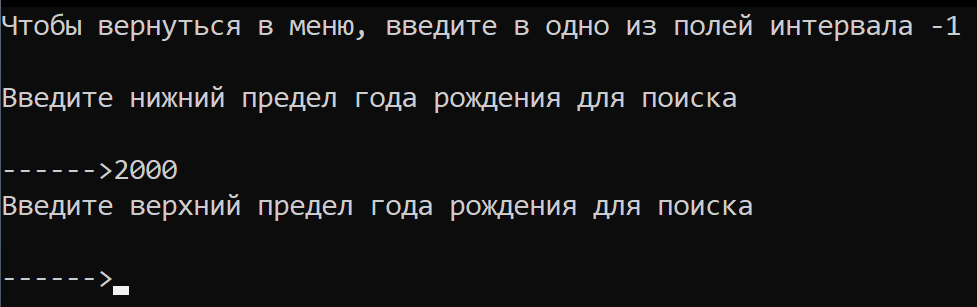
При запуске программына экране появляется главное меню.

**** Для дальнейшей работы с базой данных следует выбрать один из пунктов меню [1-6].

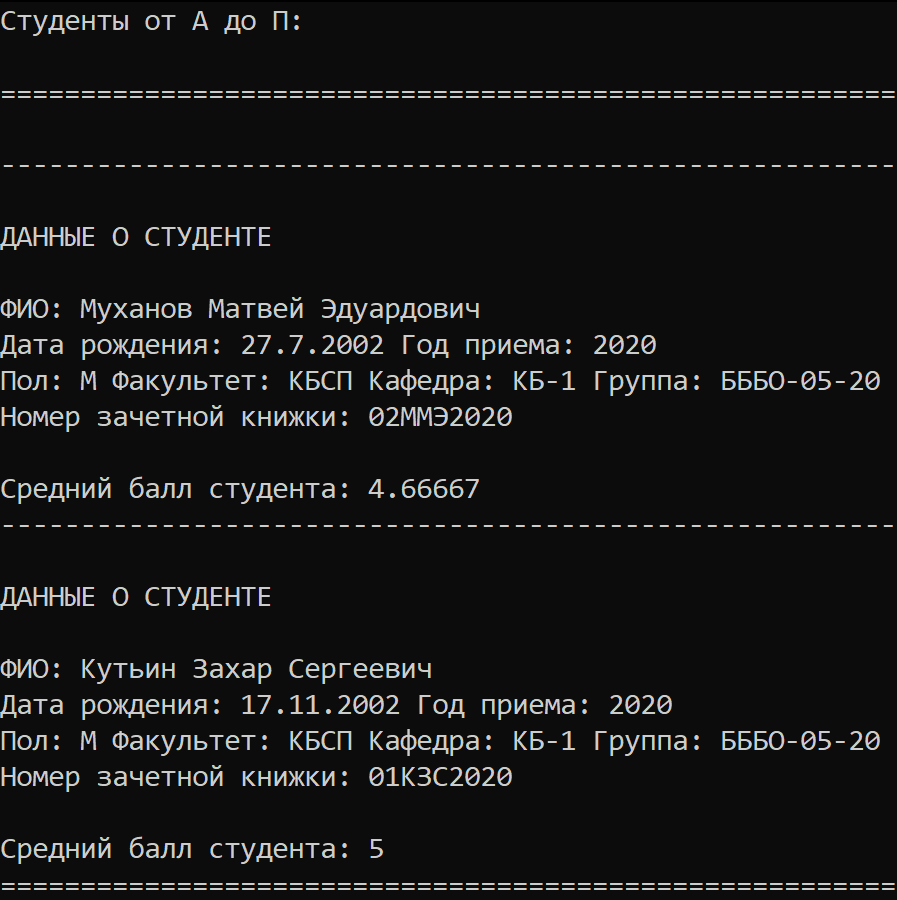
1. При выборе пункта [1] откроется меню задания.

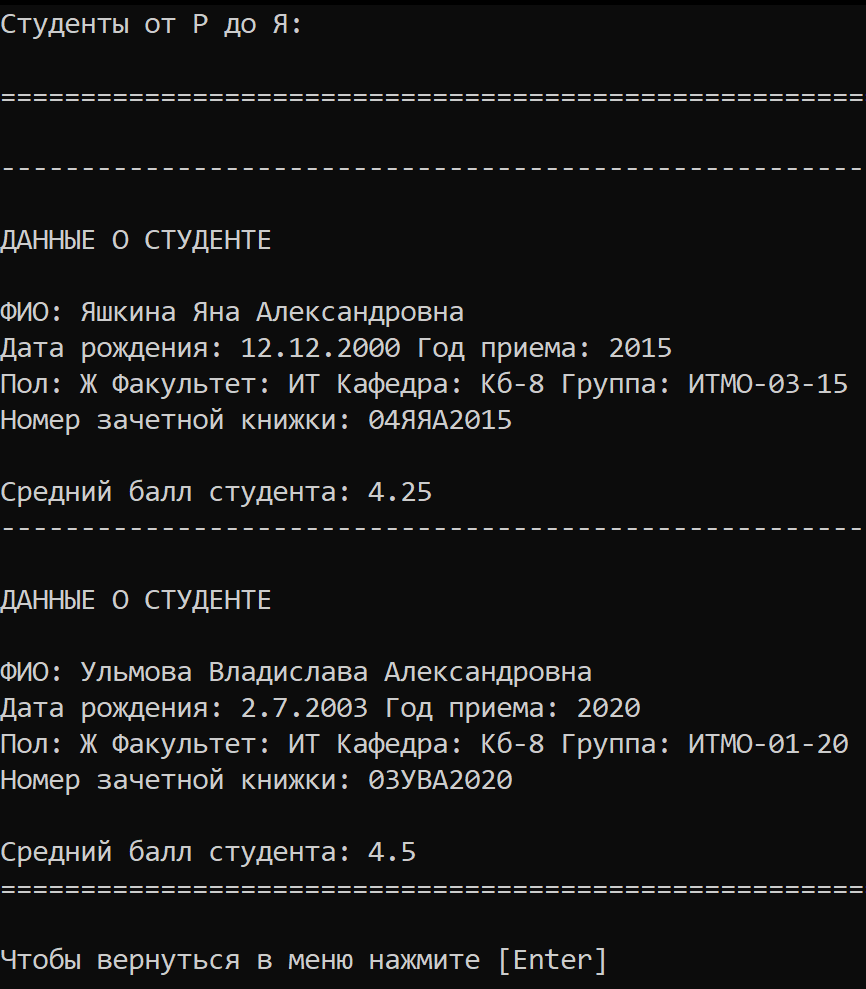
****

Для выхода в главное меню нужно ввести «-1», для выполнения задания по варианту нужно ввести нижний предел года рождения для поиска. После его ввода появится поле для ввода верхнего предела года рождения.

****

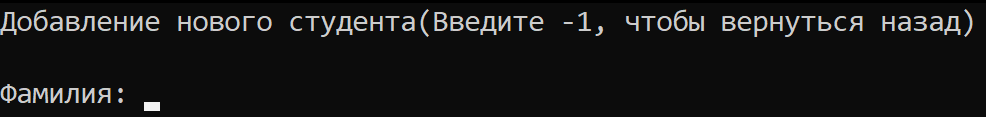
Чтобы вернуться в меню, нужно ввести «-1». Чтобы вывести студентов в заданном диапазоне, задать значение верхнего предела года рождения. После этого программа выводит студентов: сначала тех, у кого фамилия начинается от А до П, затем тех, у кого фамилия начинается от Р до Я, при этом каждая группа отсортирована в порядке увеличения среднего балла за все сессии.

****

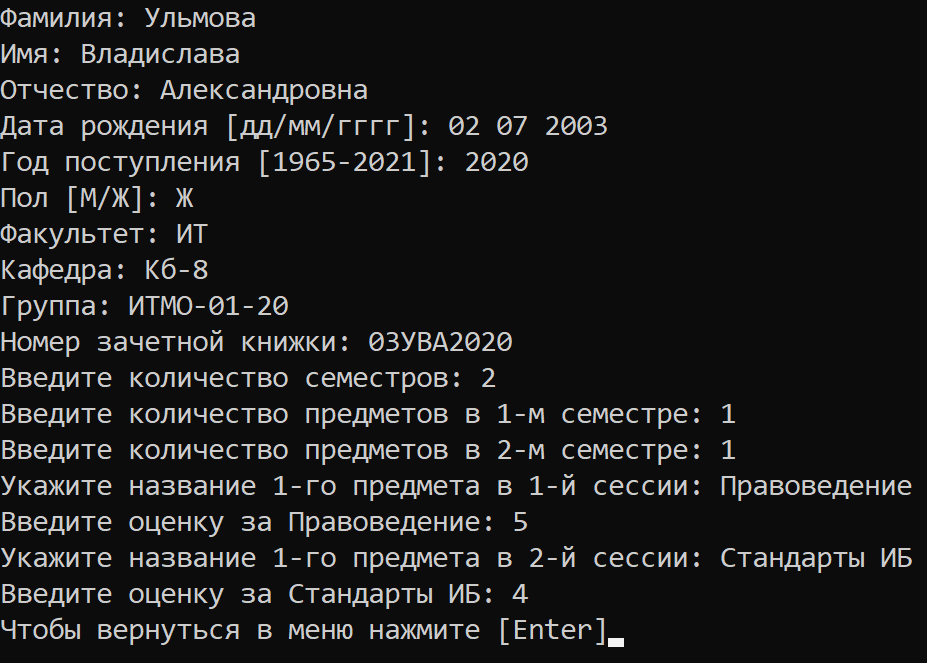
****

Для возвращения в главное меню нужно нажать [Enter].

1. При выборе пункта [2] откроется меню добавления студента в базу данных.

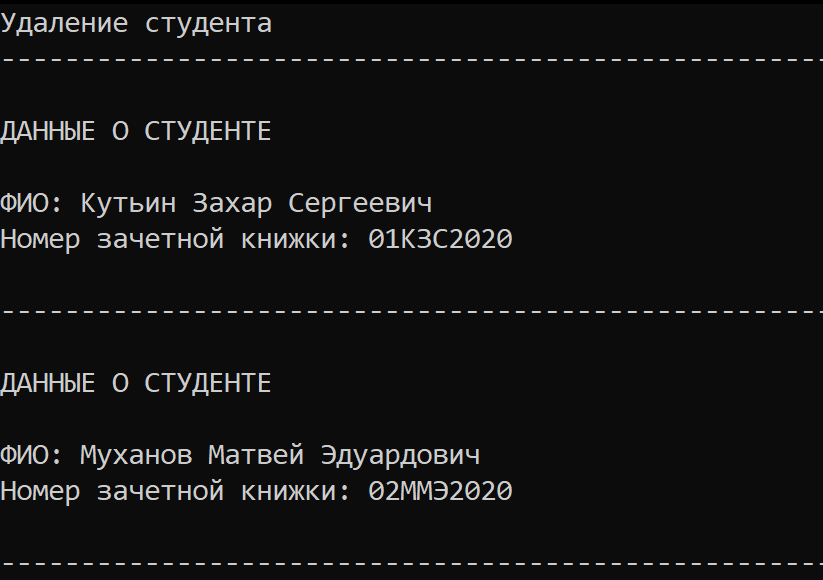


Для выхода в главное меню нужно ввести «-1», для добавления студента в базу данных нужно ввести фамилию. После ввода фамилии последовательно появятся поля для ввода имени, отчества, даты рождения, года поступления, пола, факультета, кафедры, группы, номера зачетной книжки, количества семестров, количества предметов в каждом семестре, названия предметов, оценки за предмет.

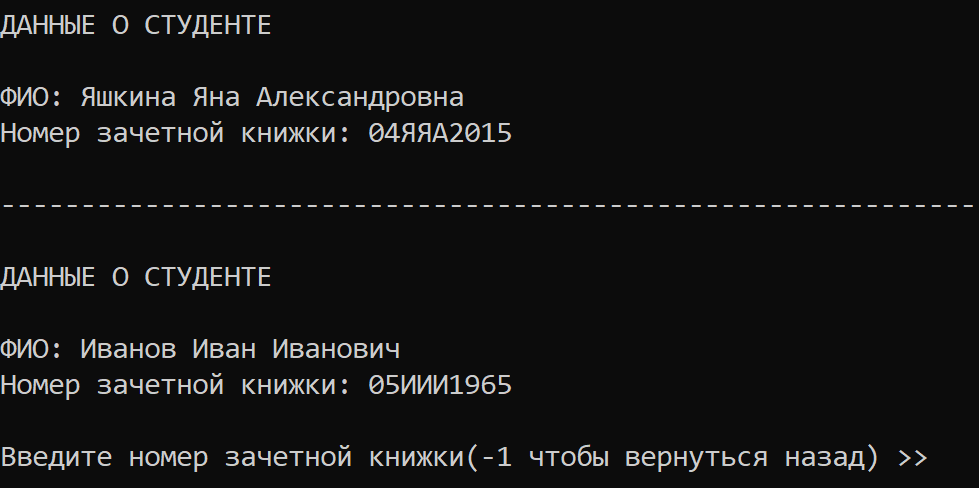


После ввода всех данных нужно нажать [Enter], чтобы вернуться в главное меню.

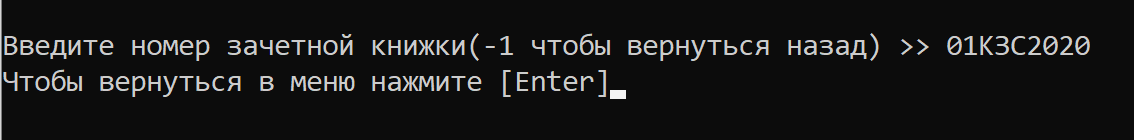
1. При выборе пункта 3 откроется меню удаления студента.



На экране появятся все студенты, которые находятся в файле, с номерами их зачетных книжек. Чтобы удалить студента из базы данных нужно ввести в поле номер его зачетной книжки. Для того чтобы вернуться в главное меню, нужно ввести «-1».

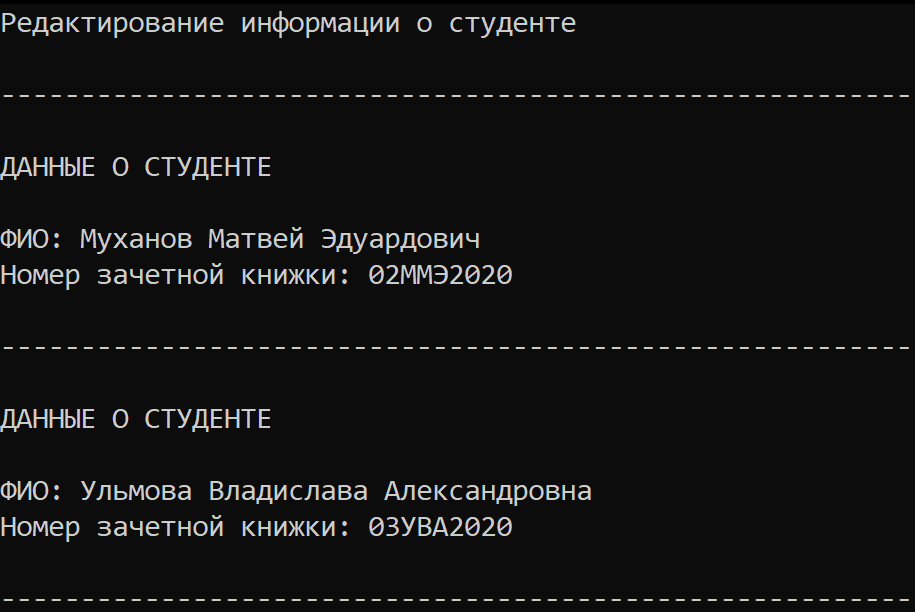


После ввода корректной записи зачетной книжки, студент будет удален из базы данных.

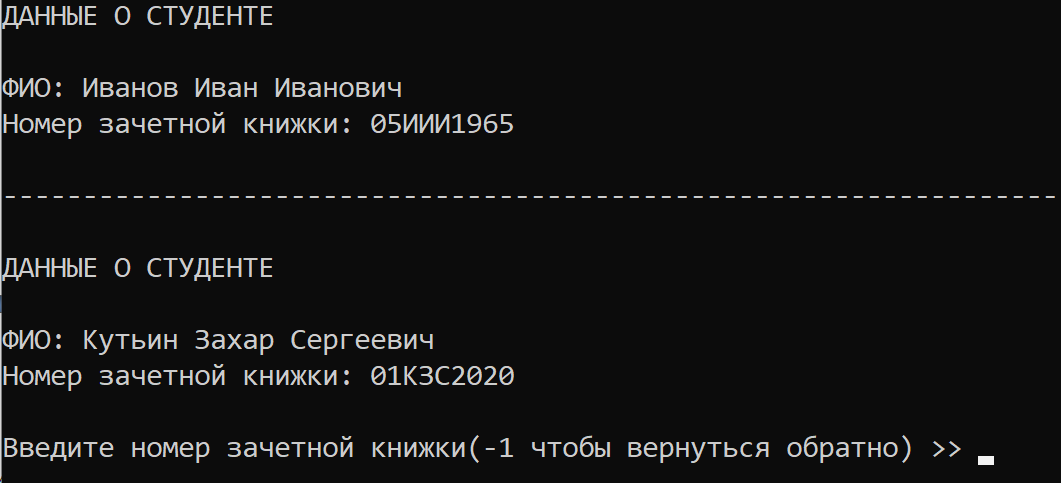


Для возвращения в главное меню нужно нажать [Enter].

1. При выборе пункта 4 откроется меню изменения любых данных о студенте.

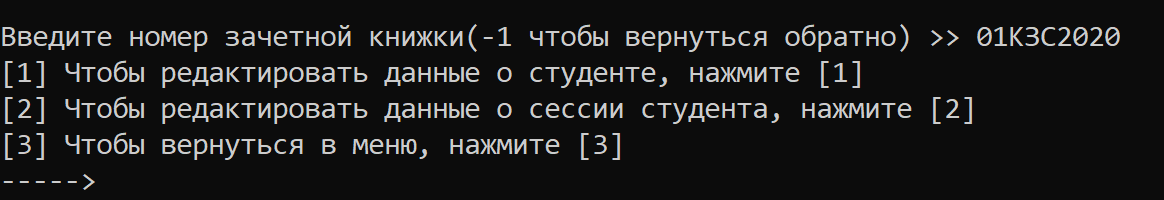


На экране появятся все студенты, которые находятся в файле, с номерами их зачетных книжек. Чтобы изменить данные о студенте в базе данных нужно ввести в поле номер его зачетной книжки. Для того чтобы вернуться в главное меню, нужно ввести «-1».

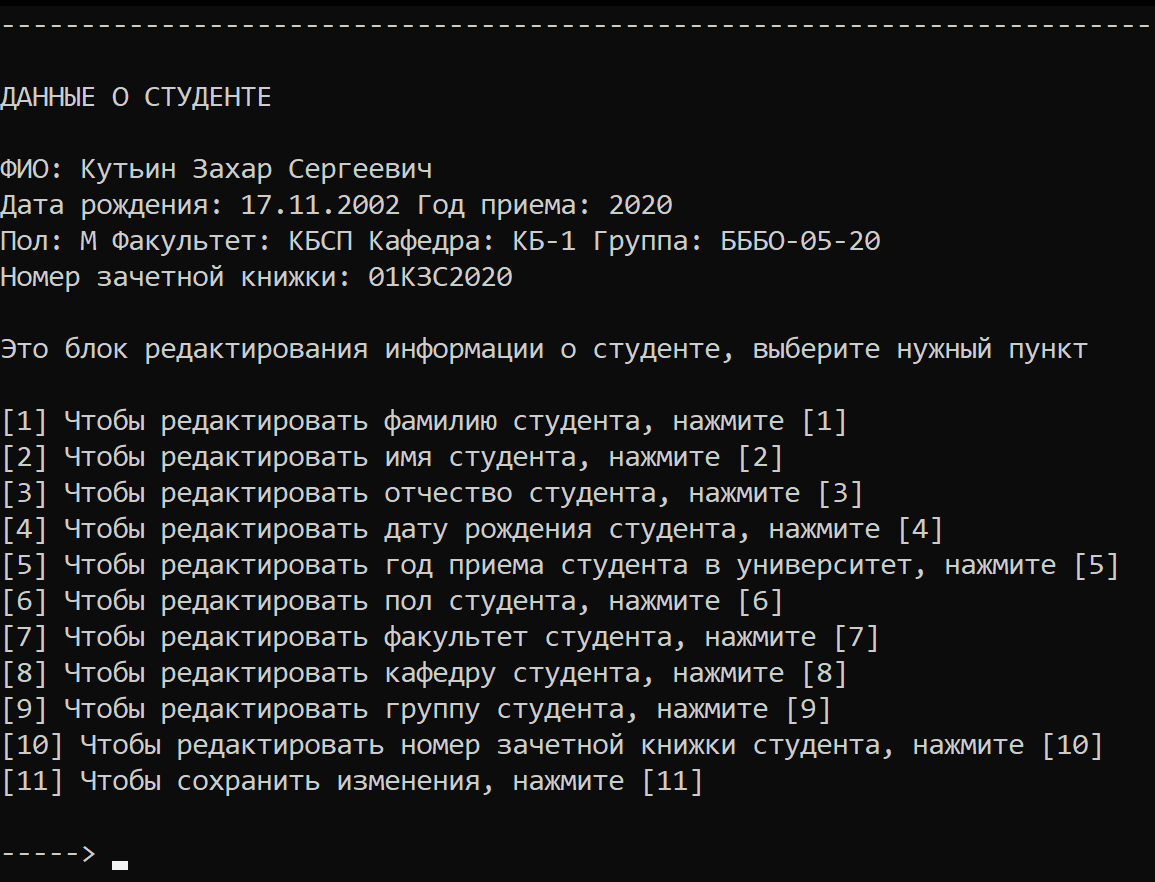


После ввода зачетной книжки появится меню, где необходимо выбрать:

Изменить данные о студенте или изменить данные о его сессии. Чтобы вернуться в главное меню нужно ввести [3] и нажать [Enter].



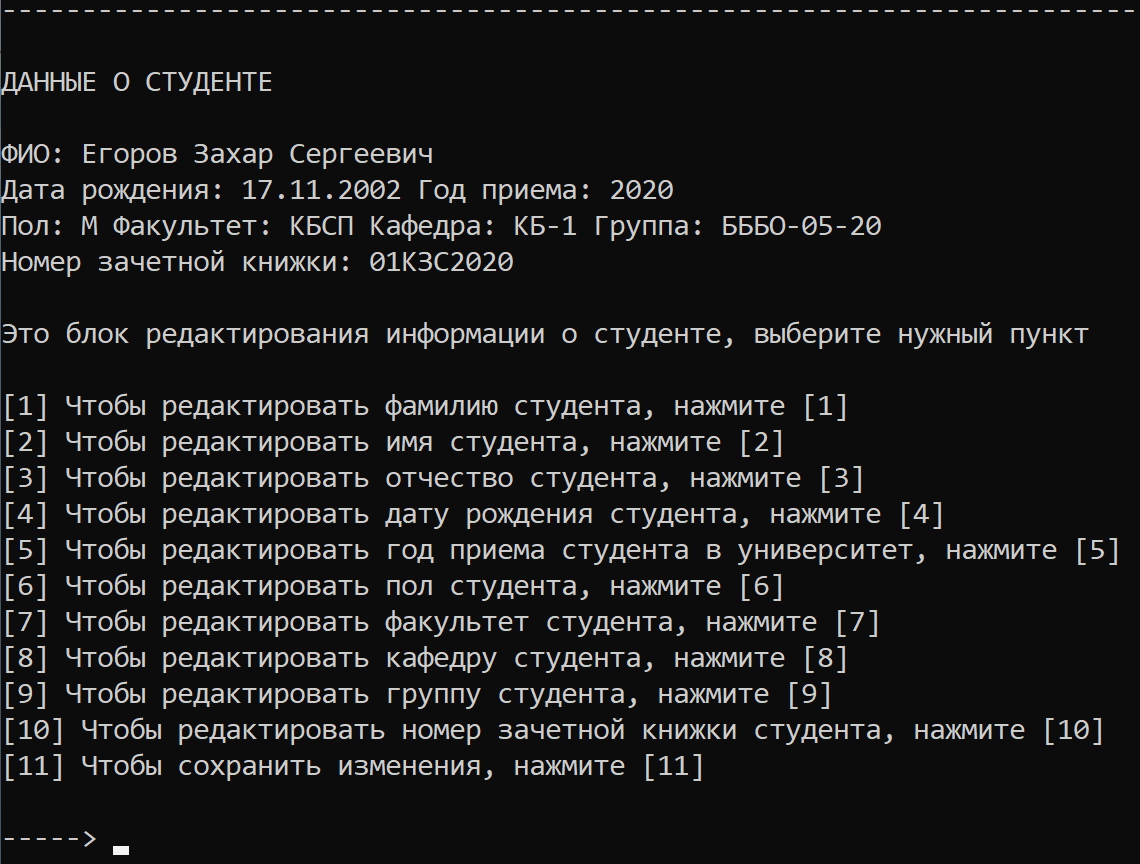
* 1. При выборе пункта 1 откроется меню изменения данных о студенте



Затем необходимо выбрать какие именно данные нужно изменить. При выборе одного из пунктов [1-10], появится поле для ввода изменения данных. Например, для фамилии:



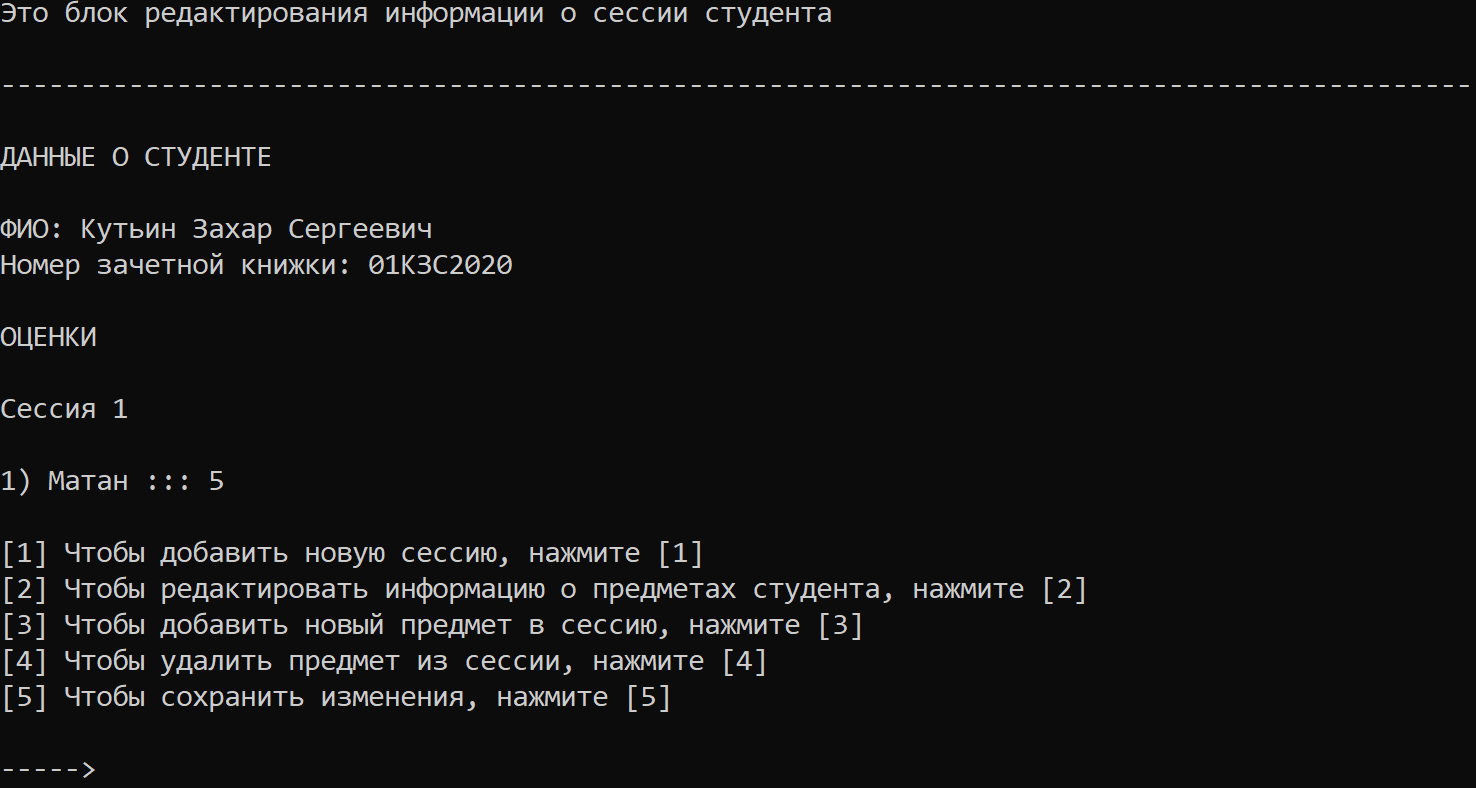
После ввода изменений и нажатия [Enter] данные о студенте поменяются автоматически.



Затем можно выбрать еще один из пунктов для изменения данных о студенте. Чтобы сохранить все изменения в файле, нужно ввести «11».

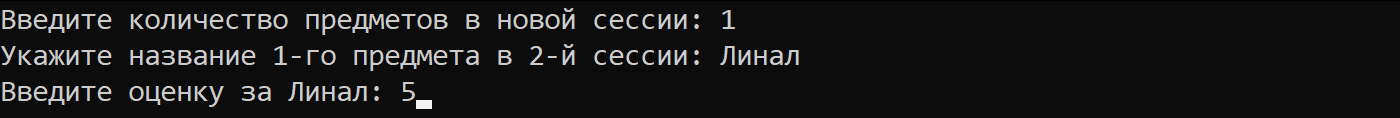
И для выхода в главное меню нажать [Enter].

* 1. При выборе пункта 2 откроется меню изменения данных о сессии студента.

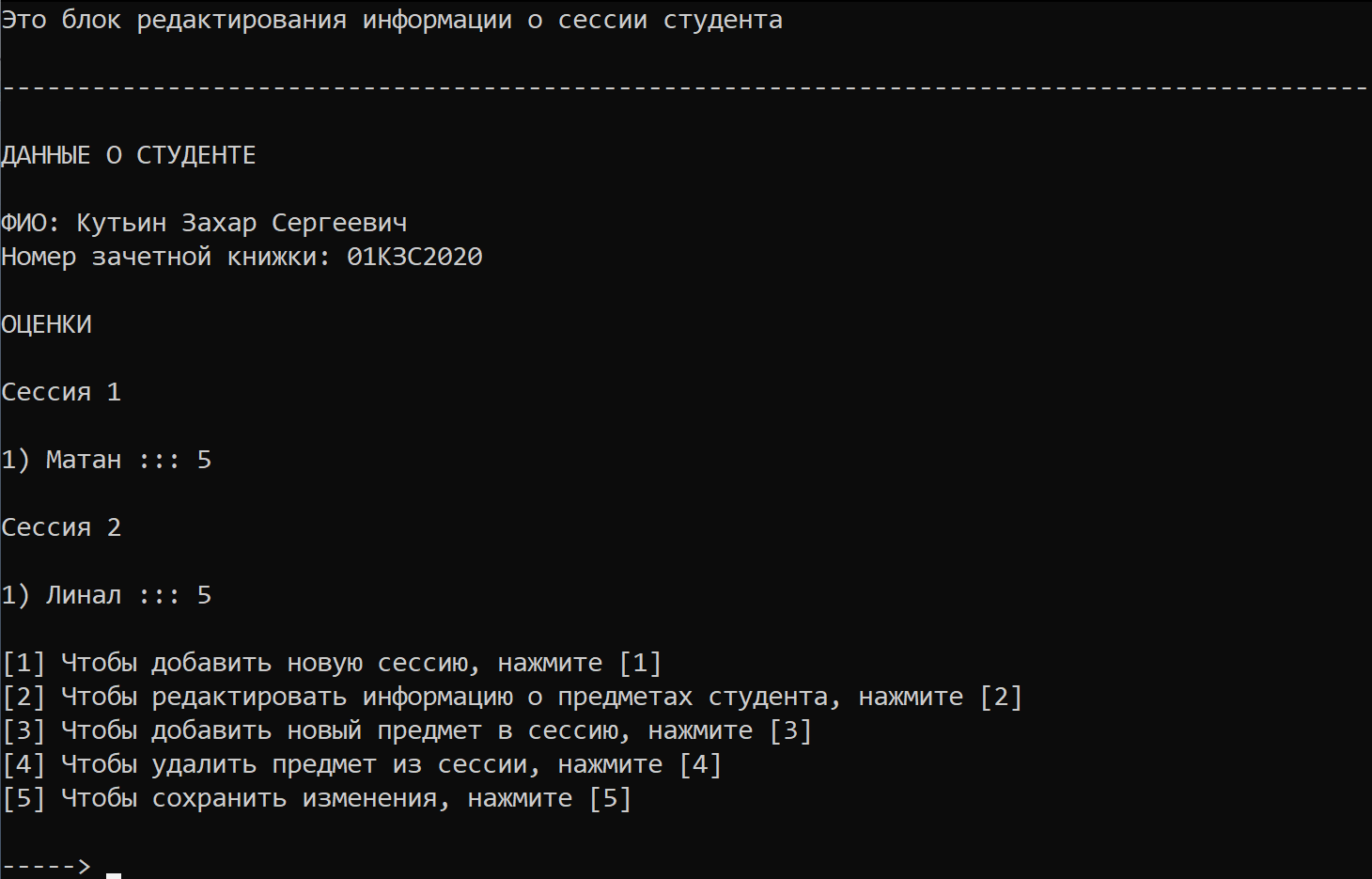


Затем необходимо выбрать какую именно информацию о сессиях нужно изменить.

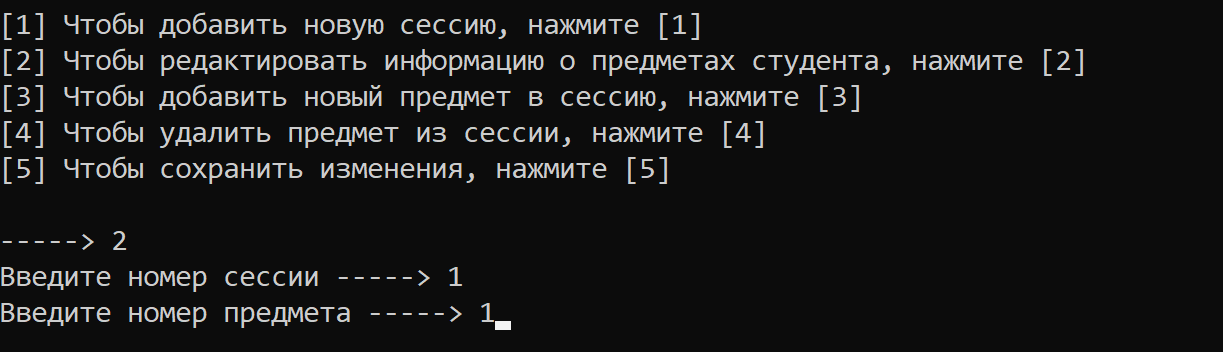
* + 1. При выборе пункта 1 появится меню добавления новой сессии. Нужно ввести количество предметов в новой сессии, названия предметов и оценку за каждый из них.

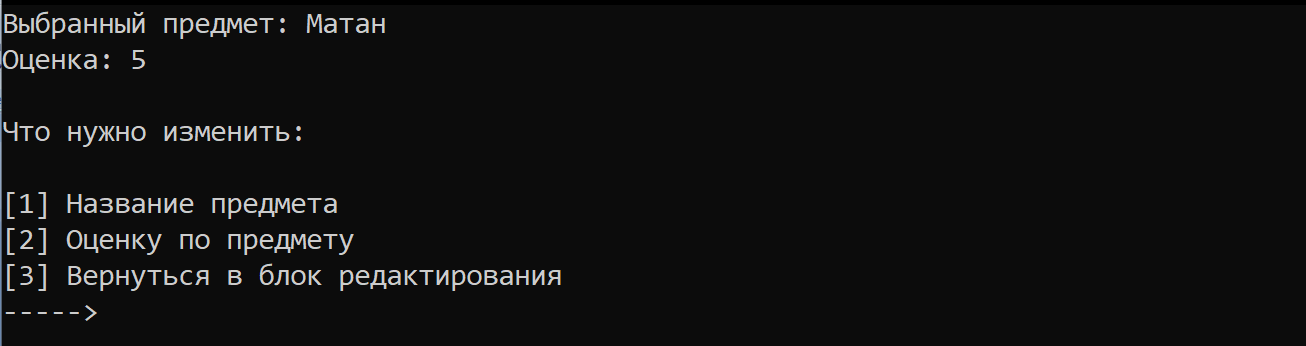


После этого у студента добавится новая сессия.

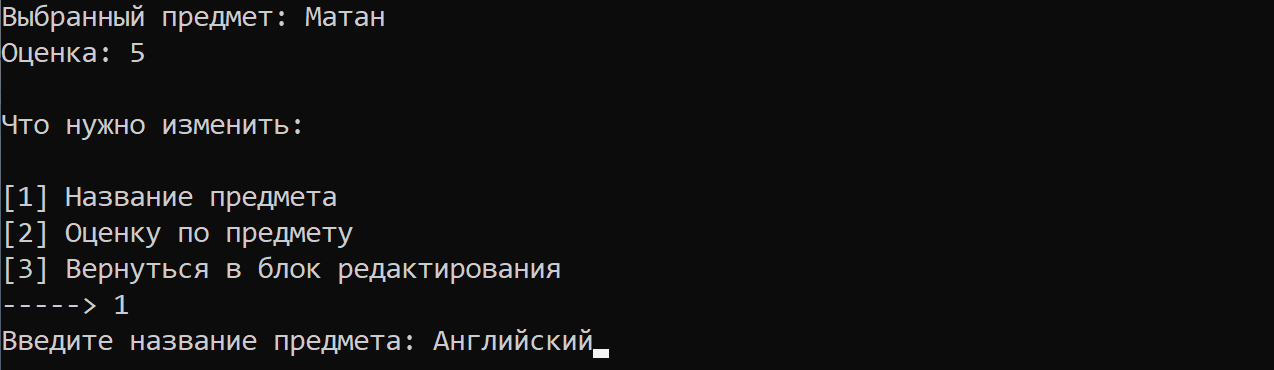


* + 1. При выборе пункта 2 появится, нужно ввести номер сессии, в которой будет отредактирована информация о предметах. Затем ввести номер предмета

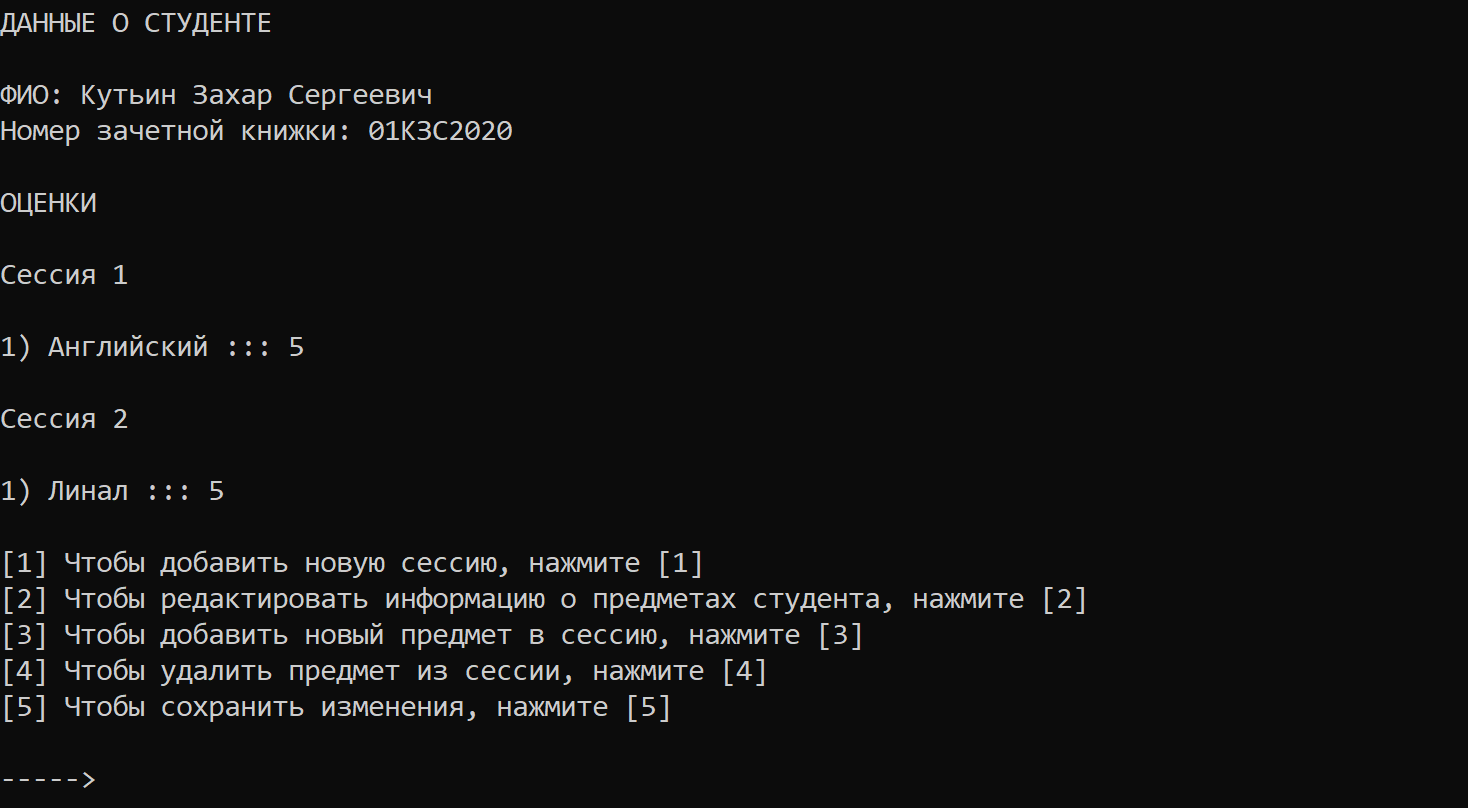
После этого откроется меню редактирования информации о предмете.



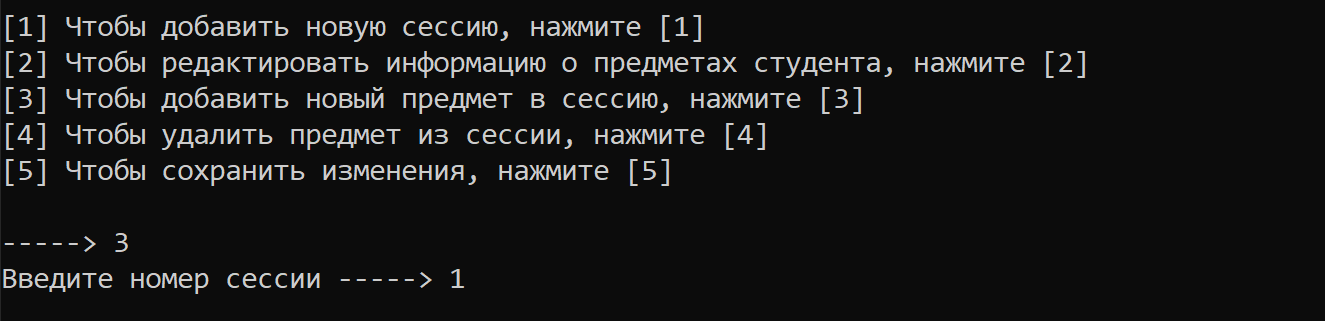
Затем необходимо выбрать какие именно данные нужно изменить. При выборе одного из пунктов [1-2], появится поле для ввода изменения данных. Например, для названия предмета:



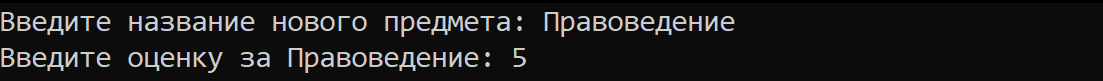
После ввода изменений и нажатия [Enter] данные о предмете поменяются автоматически. Для возвращения в меню изменения информации о сессии нужно ввести «3».



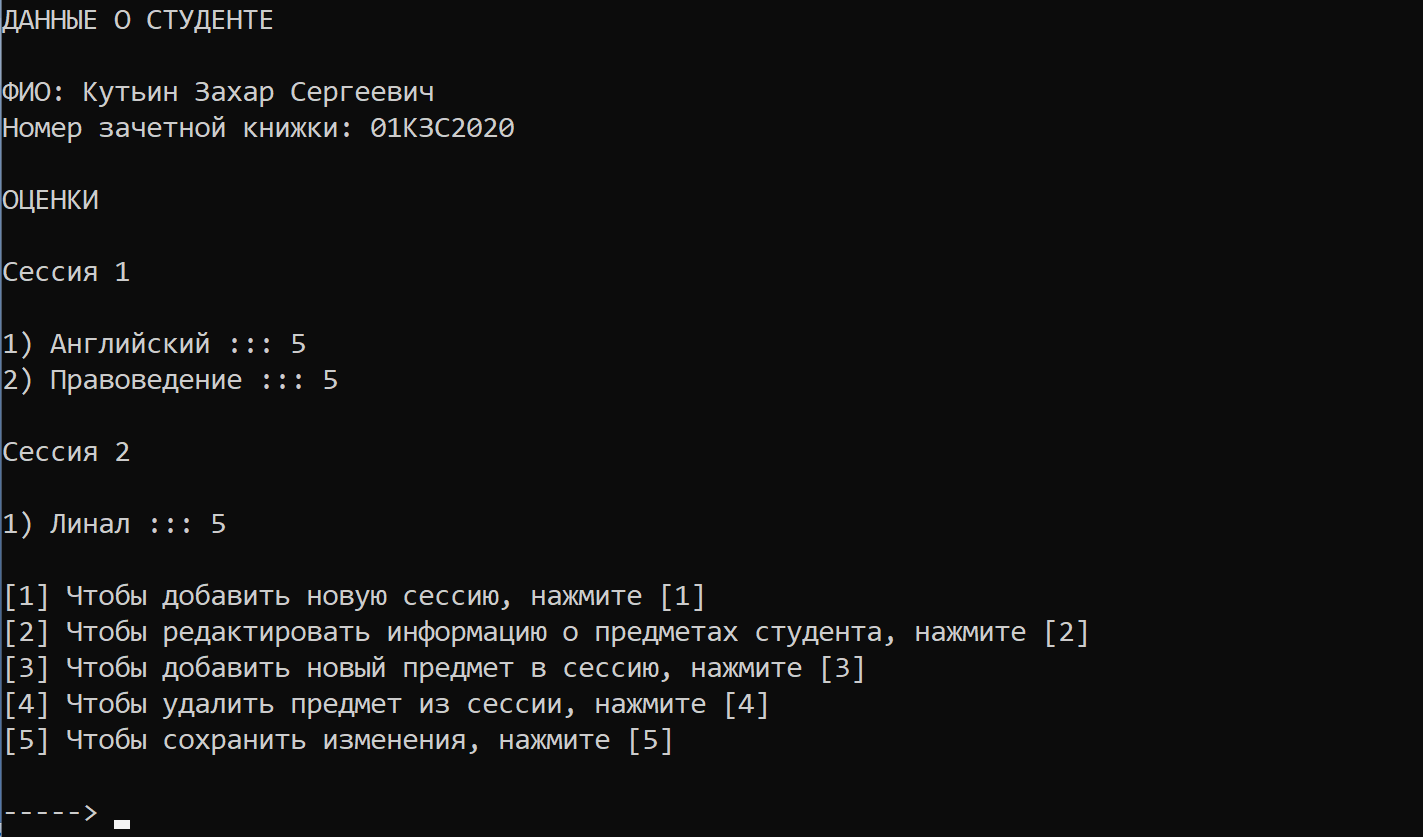
* + 1. При выборе пункта 3, нужно ввести номер сессии, в которую будет добавлен новый предмет.



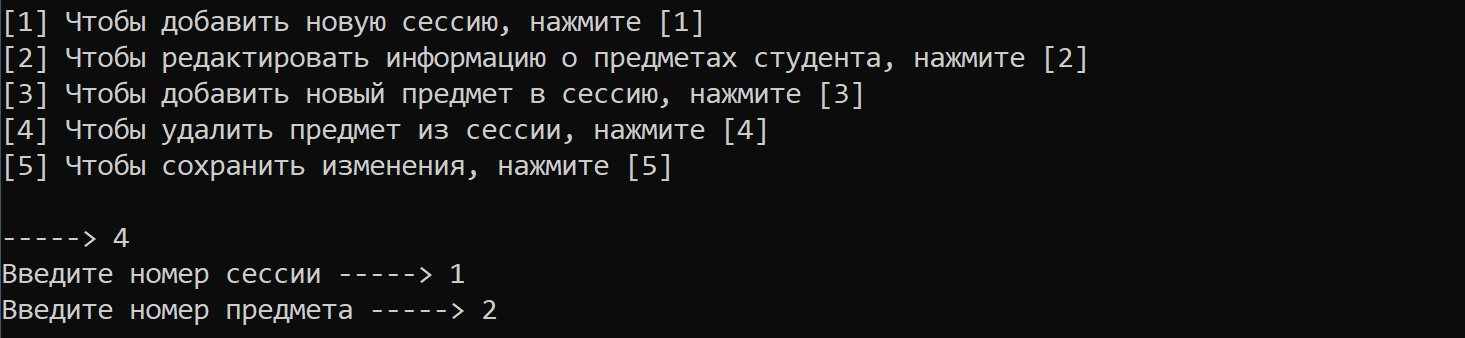
Затем станут доступны поля для ввода названия нового предмета и оценки за него.

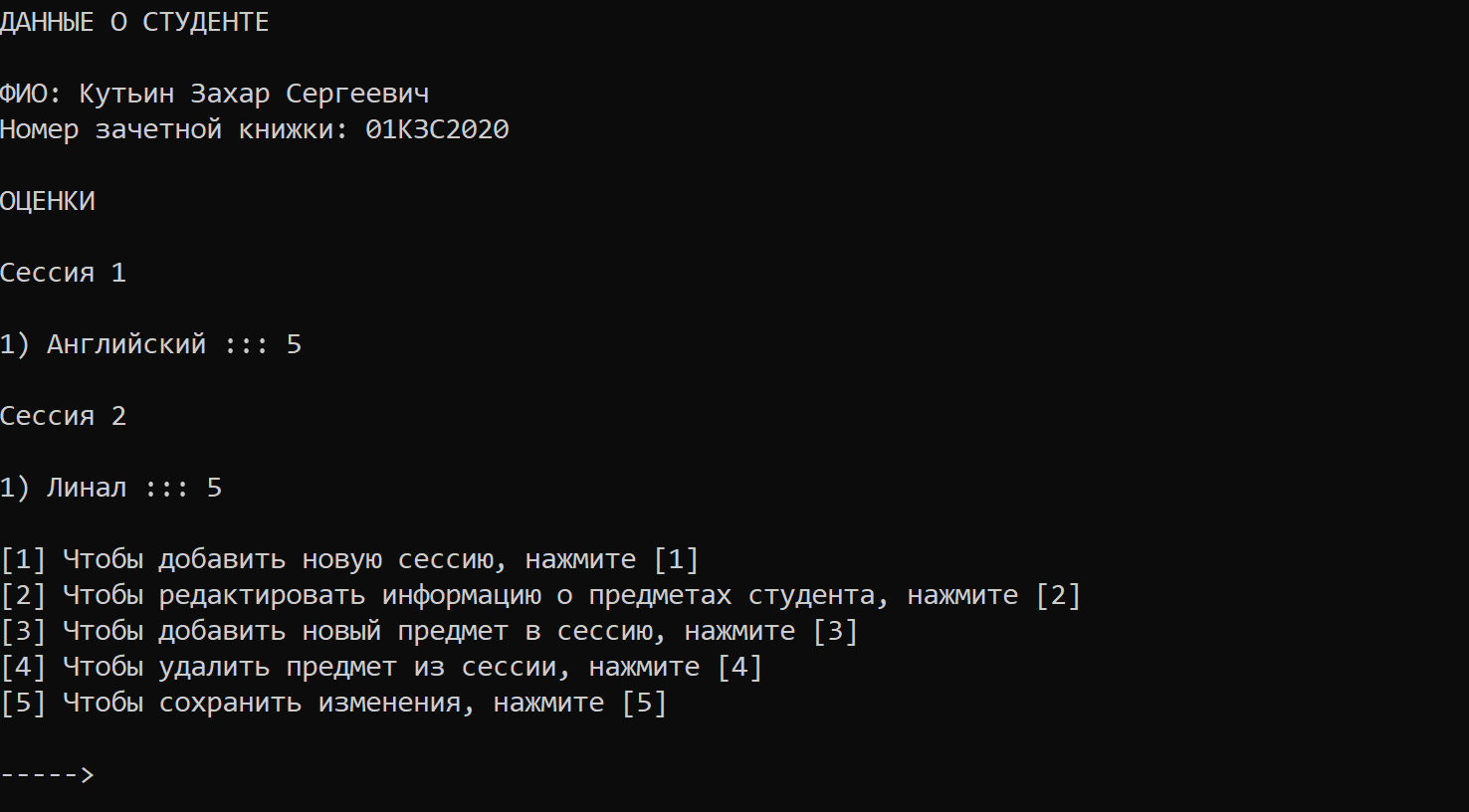


После ввода данных и нажатия [Enter] новый предмет будет добавлен в выбранную сессию.



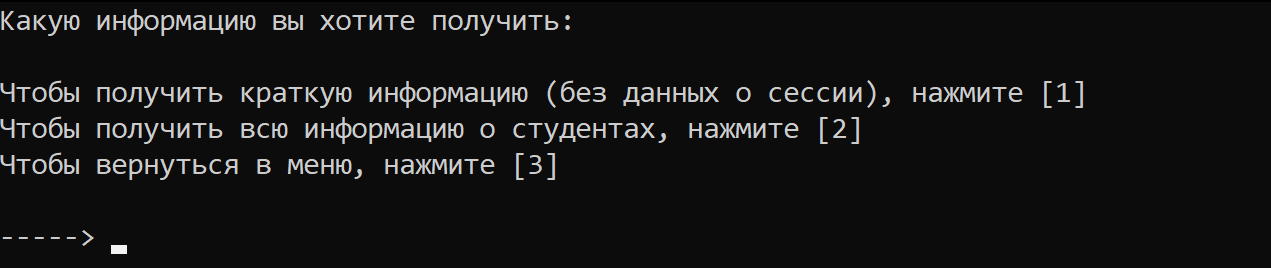
* + 1. При выборе пункта 4 появятся поля для ввода номера сессии и номера предмета, который затем будет удален.



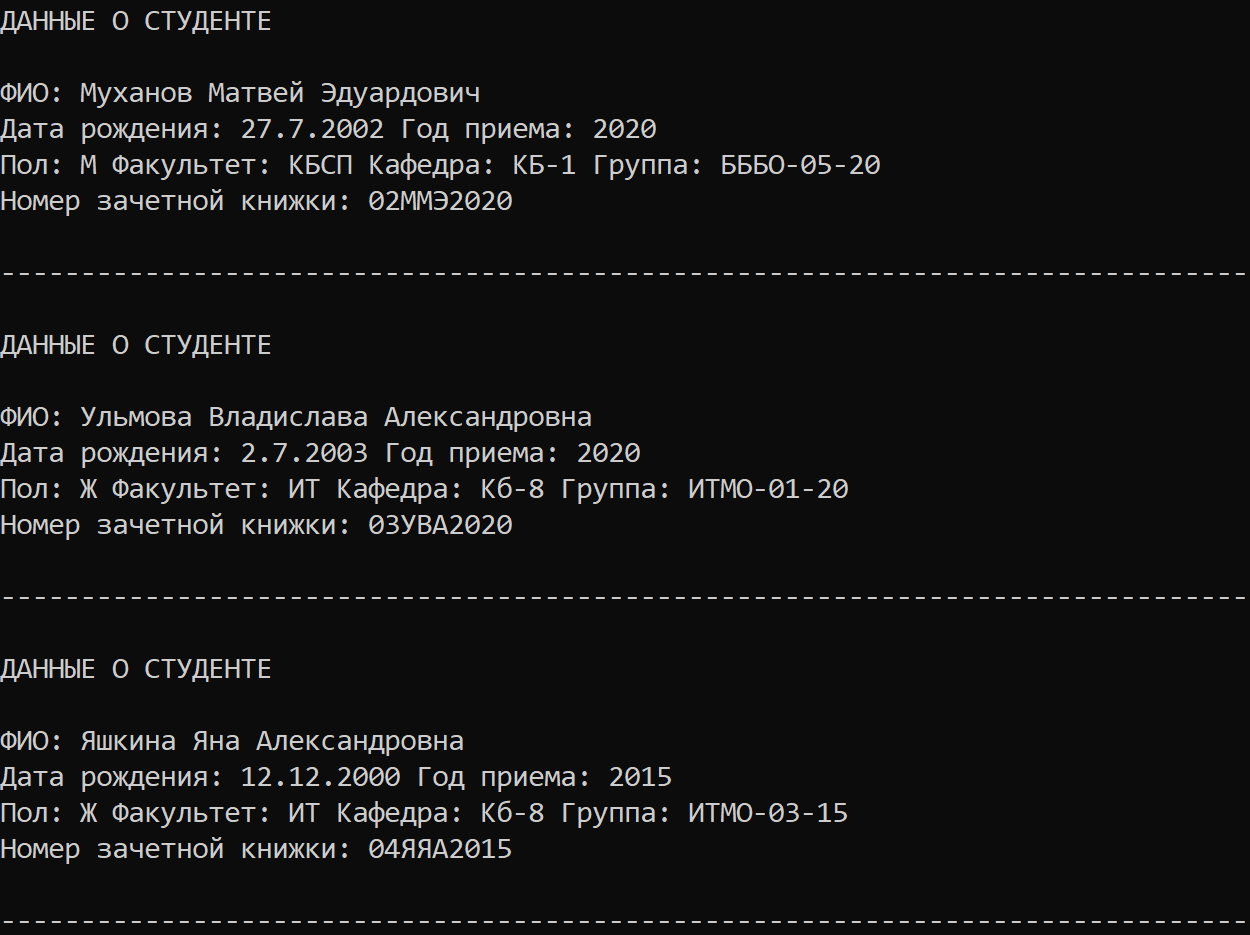


* + 1. При выборе пункта 5 все изменения сохраняться в файле. И после нажатия [Enter] будет осуществлен выход в главное меню.
  1. При выборе пункта 3 будет осуществлен выход в главное меню.

1. При выборе пункта 5 откроется меню вывода данных о студенте.

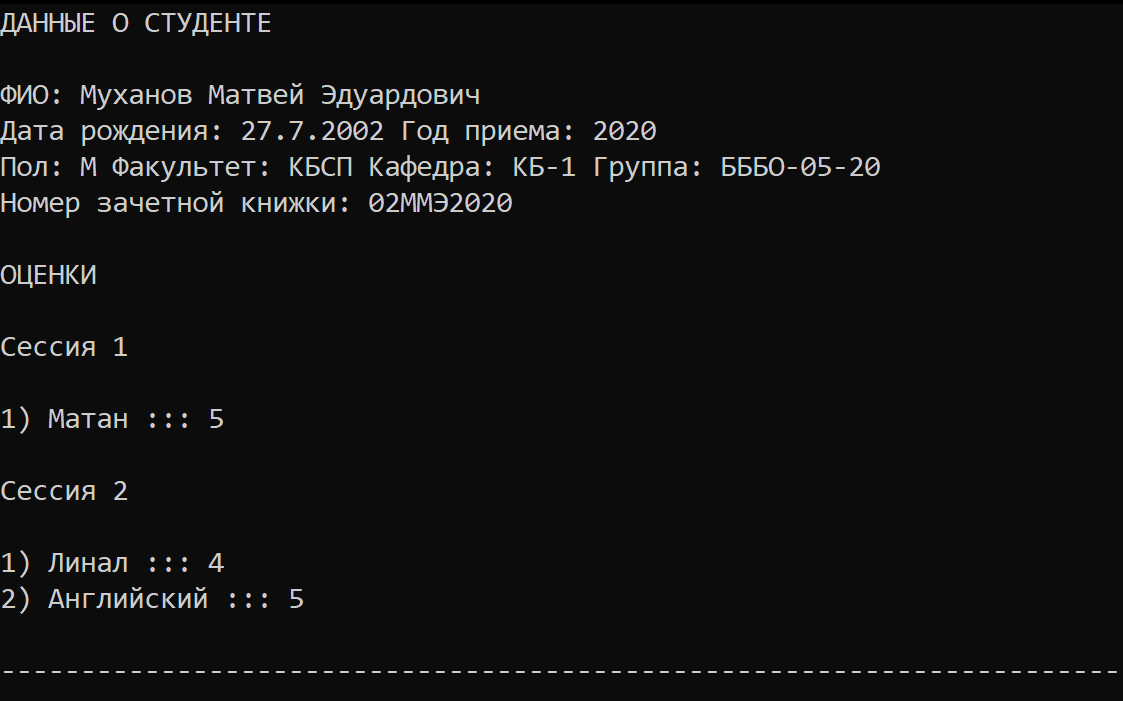


* 1. При выборе пункта 1 будет выведена информация обо всех студентах без данных о сессии (на скриншоте представлена лишь часть студентов).



Для возвращения в главное меню нужно нажать [Enter].

* 1. При выборе пункта 2 будет выведена вся информация обо всех студентах (на скриншоте представлена лишь часть студентов).



Для возвращения в главное меню нужно нажать [Enter].

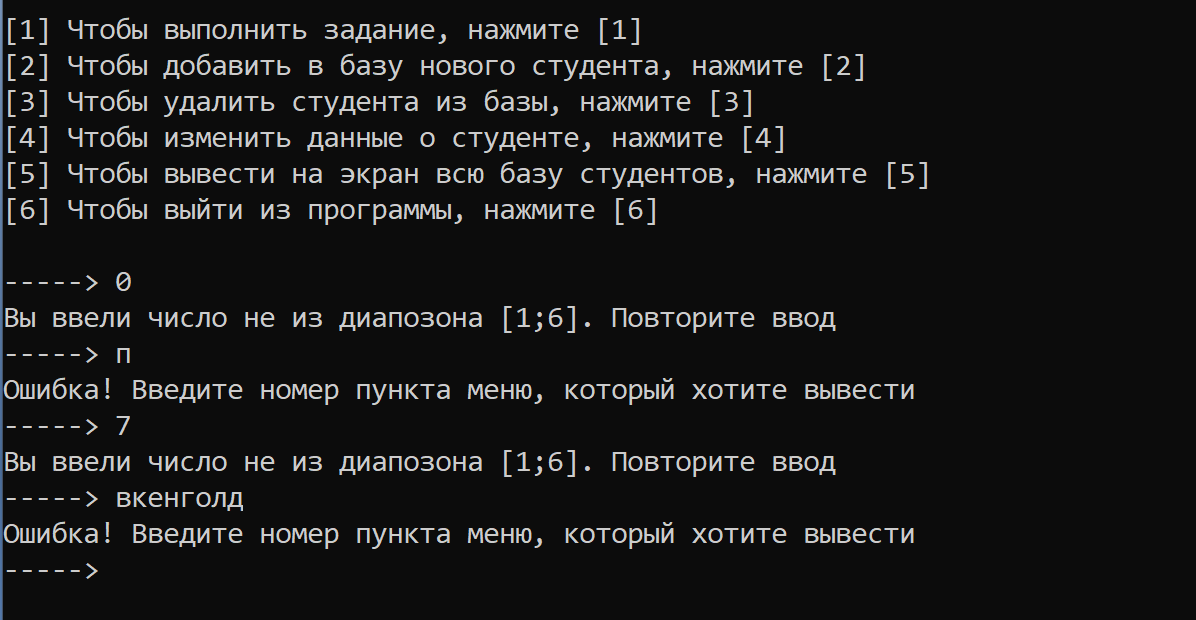
* 1. При выборе пункта 3 будет осуществлен выход в главное меню.

1. При выборе пункта 6 будет осуществлен выход из программы.

## Обработка исключительных ситуаций.

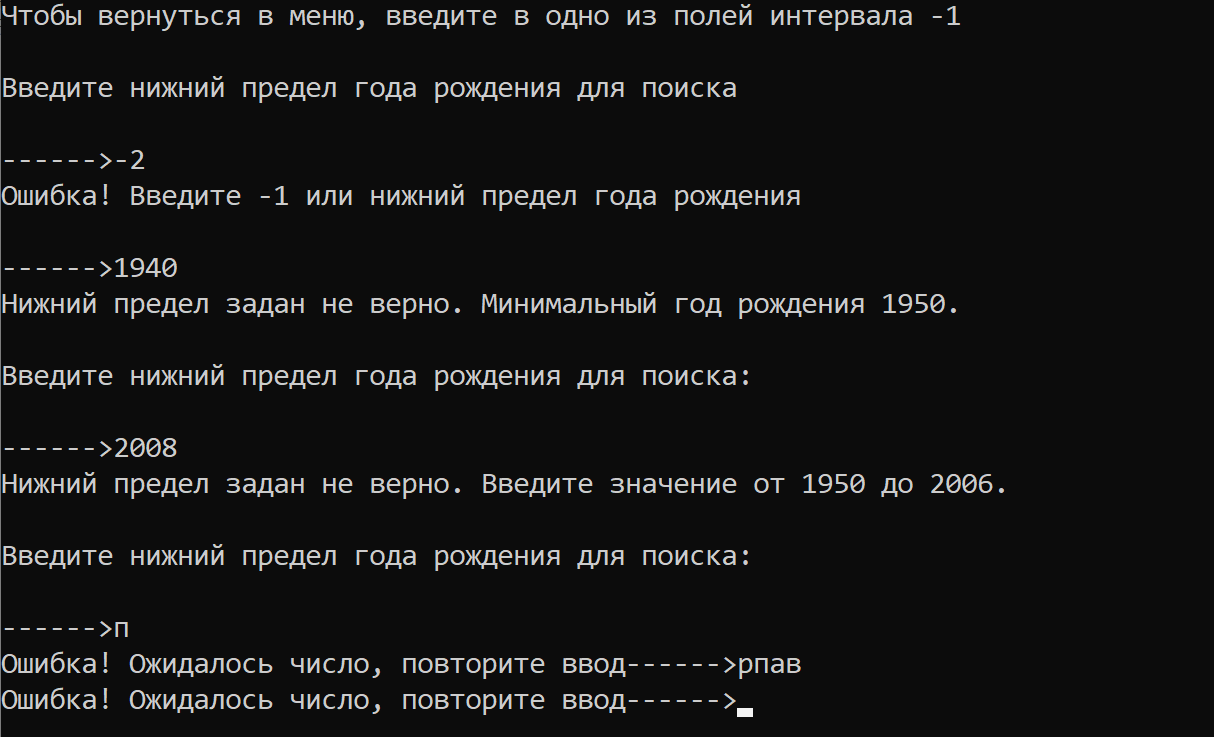
1. Введен неправильный пункт меню.

После ввода неправильного пункта меню (числа, не входящего в диапазон [1-6] или символа, или строки символов) программа запросит ввести верный пункт меню.

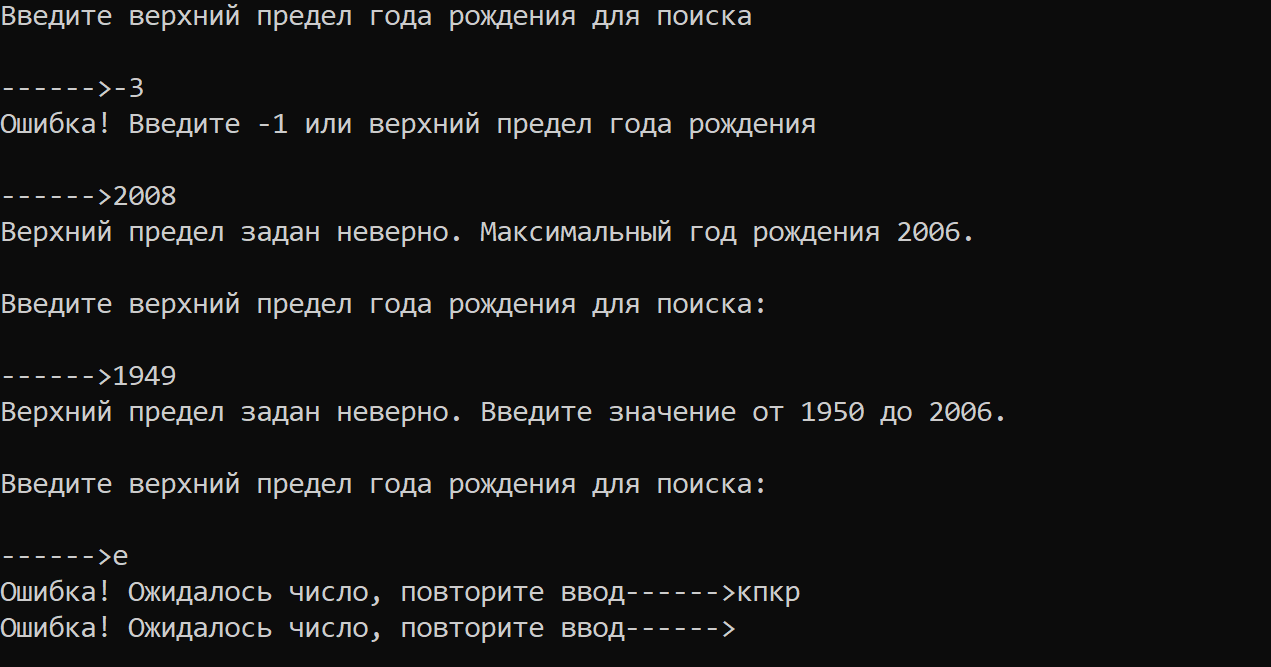


1. Введен неправильный интервал года рождения для поиска в меню задания.

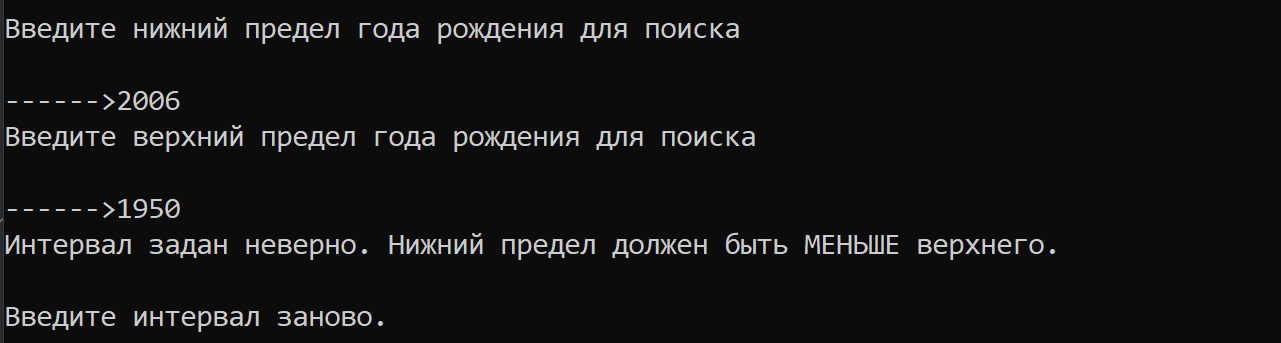
Если при вводе нижнего предела будет введено число меньше 1950, появится ошибка о том, что минимальный год рождения в базе данных – 1950 год. Если будет введено число больше 2006, программа укажет, что предел задан неверно и нужно повторить ввод. Если будет введено отрицательное число или символ, или строка символов, программа укажет на ошибку и запросит ввести год заново.



Если при вводе верхнего предела будет введено число больше 2006, появится ошибка о том, что максимальный год рождения в базе данных – 2006 год. Если будет введено число меньше 1950, программа укажет, что предел задан неверно и нужно повторить ввод. Если будет введено отрицательное число или символ, или строка символов, программа укажет на ошибку и запросит ввести год заново.

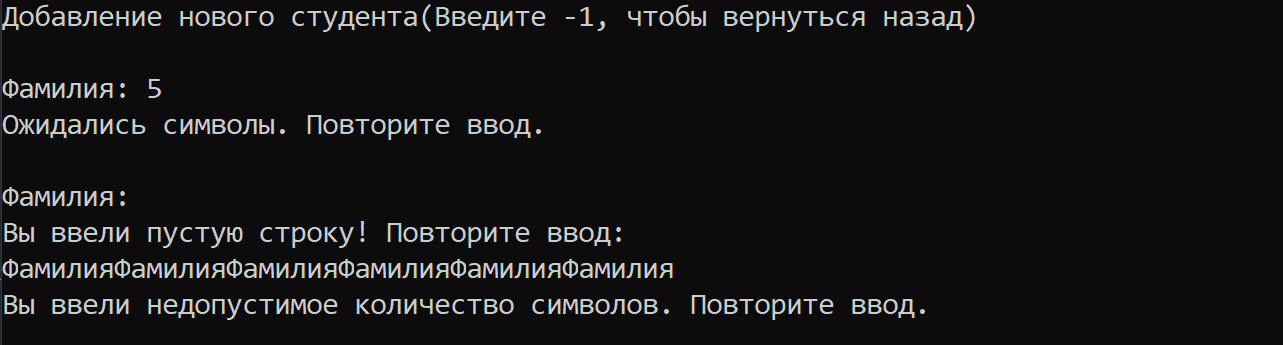


Если интервал будет задан не верно, то есть верхний предел будет меньше нижнего, программа укажет на эту ошибку и запросит ввести интервал заново.



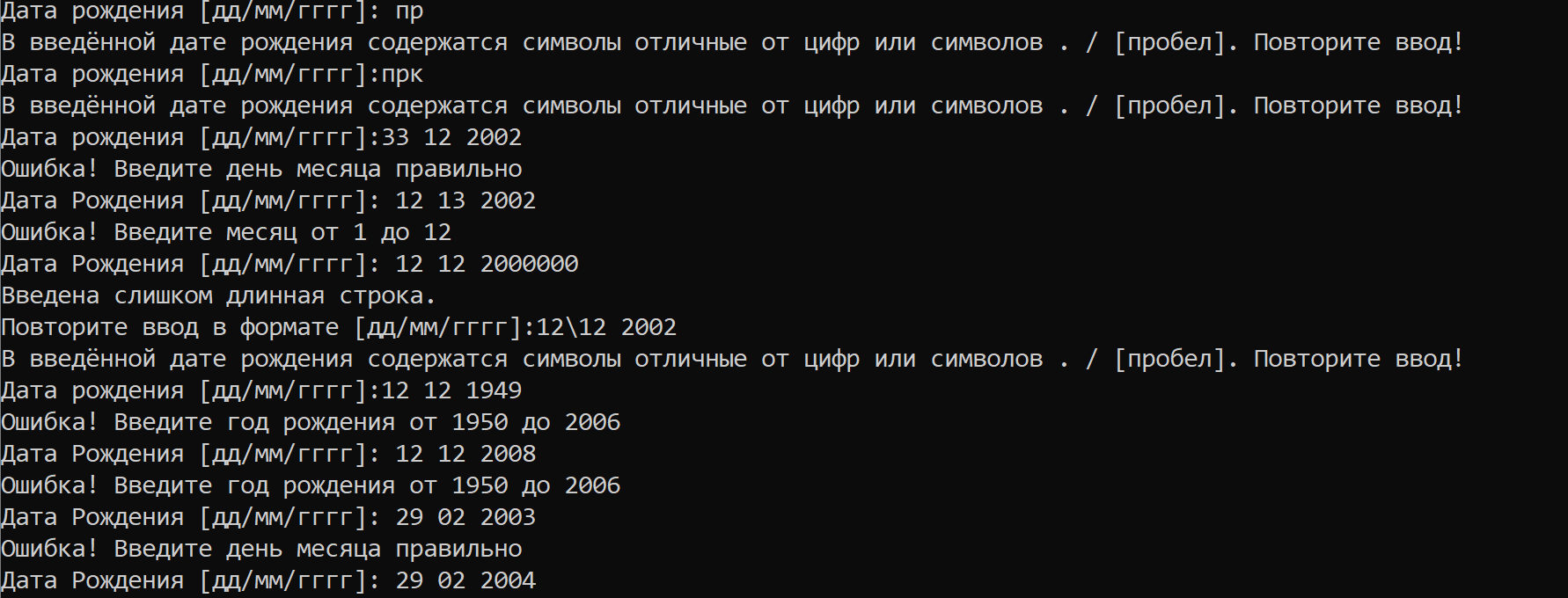
1. Ошибки, возникающие при добавлении нового студента.

При вводе цифр в поля фамилия, имя, отчества на экране появится соответствующая ошибка, и программа запросит ввод данных заново. При вводе количества символов больше допустимого значения или пустой строки в поля фамилия, имя, отчество, факультет, кафедра, группа, номер зачетной книжки, название предмета. Например, для фамилии:



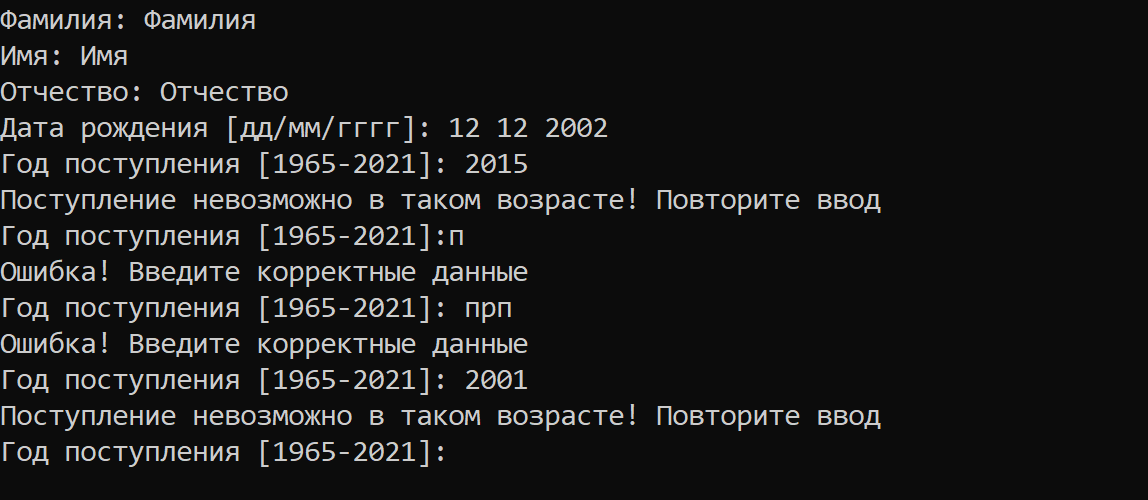
При вводе символов в поле года рождения выведется соответствующая ошибка. При вводе некорректной даты рождения (число дней месяца или число месяца, или число года превышают значения в действительности) также выведется соответствующая ошибка.

При вводе года рождения больше 2006 или меньше 1950 программа укажет, что максимальный год рождения 2006, а минимальный 1950. После вывода любой из ошибок программа запросит ввести данные заново.

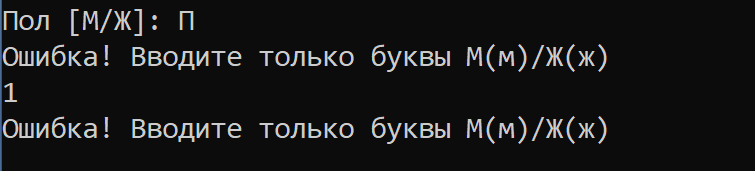


При вводе символов в поле года поступления выведется соответствующая ошибка. При вводе некорректного года поступления (год поступления меньше года рождения, разница между годом рождения и поступления меньше 15) также выведется соответствующая ошибка.

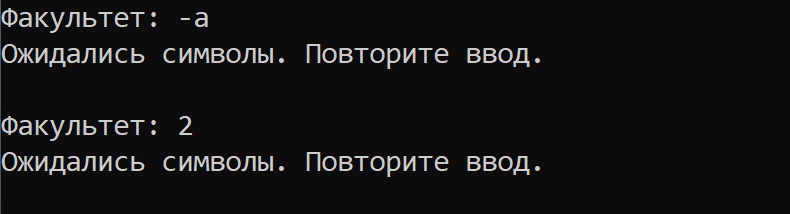
После вывода любой из ошибок программа запросит ввести данные заново.



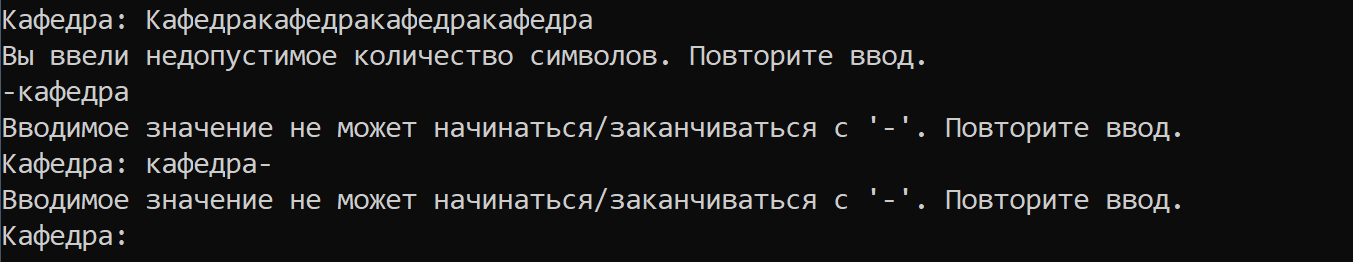
При вводе цифр или символов отличных от «М(м)» и «Ж(ж)» в поле ввода пол на экране появится соответствующая ошибка, и программа попросит ввести данные заново.



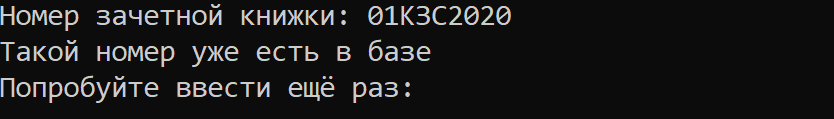
При вводе цифр и недопустимых символов в поле факультет на экране появится ошибка, и программа запросит ввод заново.



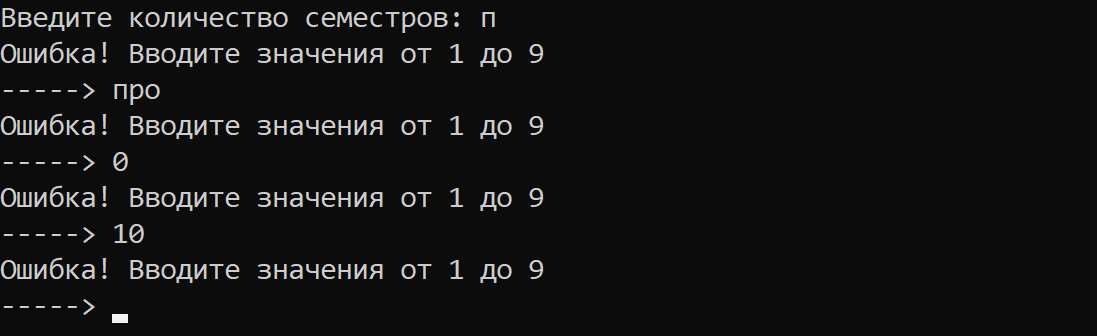
При вводе в поля кафедра, группа, номер зачетной книжки значения, начинающегося/заканчивающегося с « - » на экране появиться ошибка, и программа запросит ввод заново.



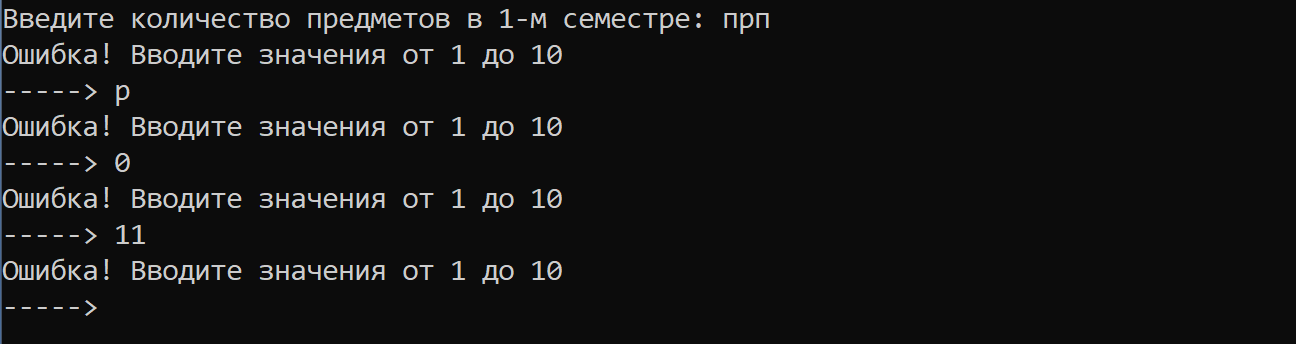
При вводе в поле номер зачетной книжки уже существующего номера программа укажет на ошибку и попросит ввести значение заново.



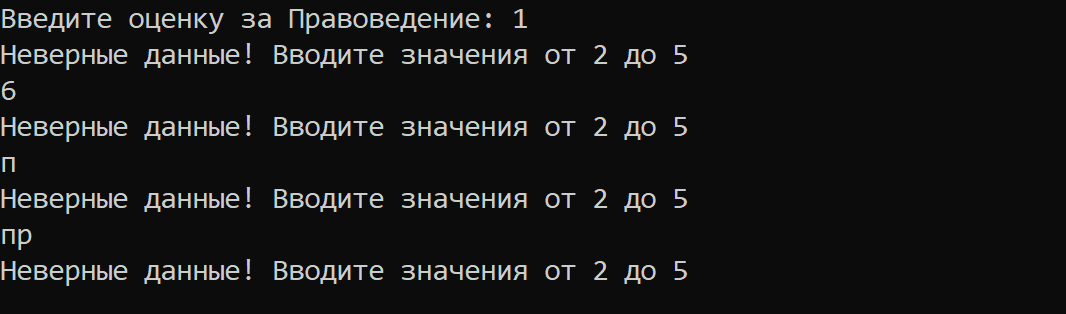
При вводе цифр не из диапазона [1-9] или при вводе символов в поле количество семестров на экране появится ошибка, и программа запросит ввод заново.



При вводе цифр не из диапазона [1-10] или при вводе символов в поле количество предметов в N семестре на экране появится ошибка, и программа запросит ввод заново.

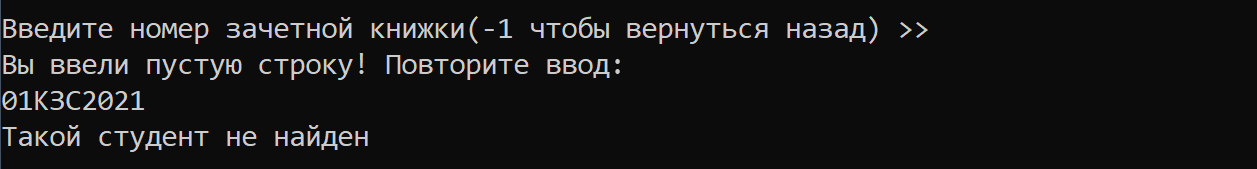


При вводе цифр не из диапазона [2-5] или при вводе символов в поле оценка за предмет на экране появится ошибка, и программа запросит ввод заново.



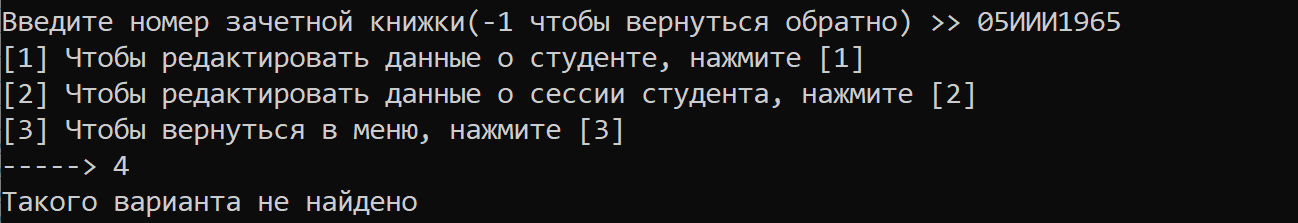
1. Ошибки, возникающие при удалении студента

При вводе пустой строки на экране появится ошибка и программа запросит ввод заново. При вводе номера зачетной книжки не существующей в базе на экране появится соответсвующая ошибка, после чего снова выведутся все студенты из базы данных, и программа запросит ввод заново.



1. Ошибки, возникающие при изменении данных о студенте.

Проверка номера зачетной книжки аналогична проверки из пункта 4. При вводе числа не из диапазона пунктов меню будет выводится соответствующая ошибка.



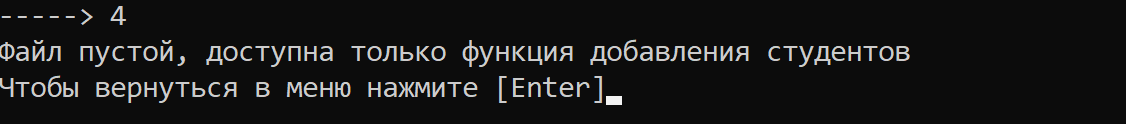
Проверки при изменении данных о студенте и его сессиях аналогичны проверкам из пункта 3.

1. Ошибки, возникающие при выводе студентов.

При вводе числа не из диапазона [1-3] осуществиться возврат в главное меню.

1. Если файл пустой.

Если в базе студентов нет ни одной записи, то будет доступна только функция добавления студентов. При выборе каких-либо других пунктов, программа укажет, что файл пуст.



# Контрольные данные

ДАННЫЕ О СТУДЕНТЕ №1

ФИО: Кутьин Захар Сергеевич

Дата рождения: 17.11.2002 Год приема: 2020

Пол: М Факультет: КБСП Кафедра: КБ-1 Группа: БББО-05-20

Номер зачетной книжки: 01КЗС2020

ОЦЕНКИ

Сессия 1

1) Матан ::: 5

ДАННЫЕ О СТУДЕНТЕ №2

ФИО: Муханов Матвей Эдуардович

Дата рождения: 27.7.2002 Год приема: 2020

Пол: М Факультет: КБСП Кафедра: КБ-1 Группа: БББО-05-20

Номер зачетной книжки: 02ММЭ2020

ОЦЕНКИ

|  |  |
| --- | --- |
| Сессия 1  1) Матан ::: 5 | Сессия 2  1) Линал ::: 4  2) Английский ::: 5 |

ДАННЫЕ О СТУДЕНТЕ №3

ФИО: Ульмова Владислава Александровна

Дата рождения: 2.7.2003 Год приема: 2020

Пол: Ж Факультет: ИТ Кафедра: Кб-8 Группа: ИТМО-01-20

Номер зачетной книжки: 03УВА2020

ОЦЕНКИ

|  |  |
| --- | --- |
| Сессия 1  1) Правоведение ::: 5 | Сессия 2  1) Стандарты ИБ ::: 4 |

ДАННЫЕ О СТУДЕНТЕ №4

ФИО: Яшкина Яна Александровна

Дата рождения: 12.12.2000 Год приема: 2015

Пол: Ж Факультет: ИТ Кафедра: Кб-8 Группа: ИТМО-03-15

Номер зачетной книжки: 04ЯЯА2015

ОЦЕНКИ

|  |  |  |
| --- | --- | --- |
| Сессия 1  1) Матан ::: 5 | Сессия 2  1) Линал ::: 3  2) Английский ::: 4 | Сессия 3  1) Правоведение ::: 5 |

ДАННЫЕ О СТУДЕНТЕ №5

ФИО: Иванов Иван Иванович

Дата рождения: 23.6.1950 Год приема: 1965

Пол: М Факультет: КБСП Кафедра: Кб-2 Группа: БАСО-04-65

Номер зачетной книжки: 05ИИИ1965

ОЦЕНКИ

|  |  |  |
| --- | --- | --- |
| Сессия 1  1) Матан ::: 5 | Сессия 2  1) Физкультура ::: 5 | Сессия 3  1) АСА ::: 4 |

## Скриншоты выполнения работы по тестовым данным

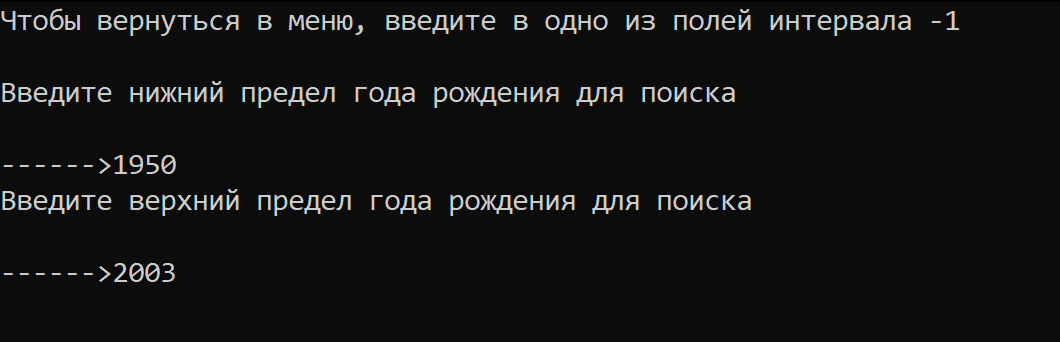
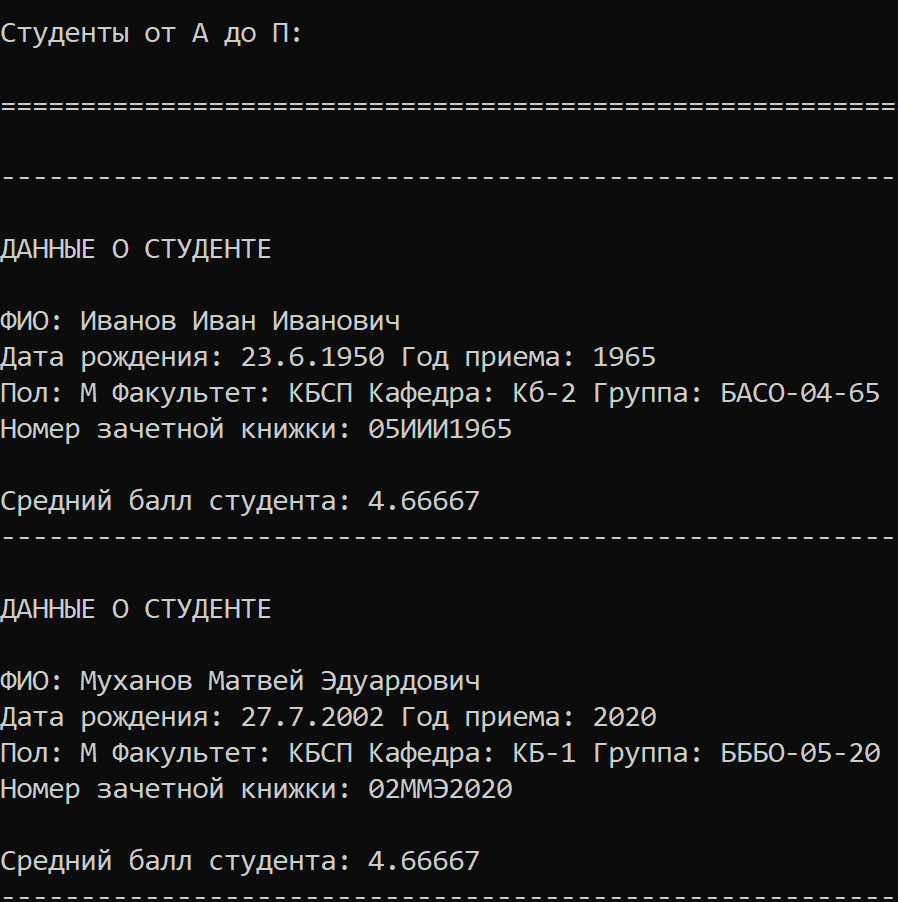
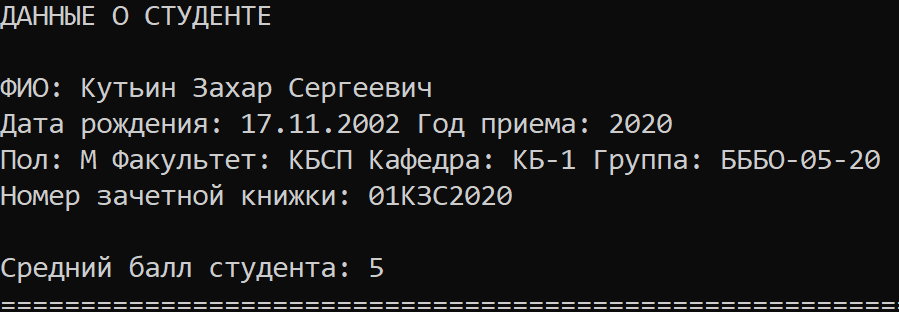


Рис. 1 Ввод интервала года рождения для вывода студентов по этому интервалу.



****Рис. 2-3 Вывод студентов с фамилиями, начинающимися от А до П, которые отсортированы в порядке увеличения среднего балла за все сессии.

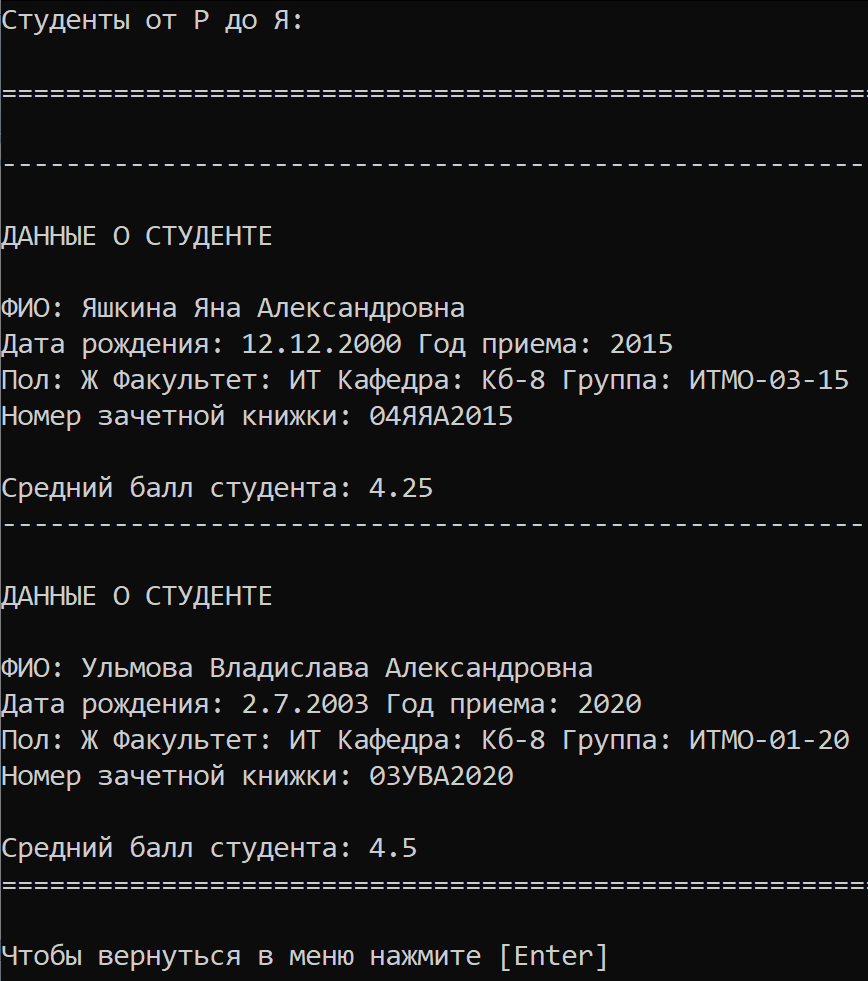


Рис. 3 Вывод студентов с фамилиями, начинающимися от Р до Я, которые отсортированы в порядке увеличения среднего балла за все сессии.

## Анализ полученных результатов

Убедимся, что программа работает исправно. Для начала сверим диапазон года рождения, который ввели с клавиатуры, с годами рождения студентов, выведенных на экран программой.

Вводимый диапазон: 1950-2003.

Года рождения студентов, которые были выведены программой:

|  |  |
| --- | --- |
| ФИО: Кутьин Захар Сергеевич  Год рождения: 2002 | ФИО: Муханов Матвей Эдуардович  Год рождения: 2002 |
| ФИО: Ульмова Владислава Александровна  Год рождения: 2003 | ФИО: Яшкина Яна Александровна  Год рождения: 2000 |
| ФИО: Иванов Иван Иванович  Год рождения: 1950 |  |

Как можно заметить, студенты были правильно выведены в заданном диапазоне. Далее проверим, что студенты были верно распределены по первым буквам фамилии. Рассмотрим, в каких группах программа вывела студентов:

|  |  |
| --- | --- |
| От А до П: | От Р до Я: |
| Иванов Иван Иванович | Яшкина Яна Александровна |
| Муханов Матвей Эдуардович | Ульмова Владислава Александровна |
| Кутьин Захар Сергеевич |  |

Программа верно распределила студентов на две группы, по первой букве фамилии. И наконец, стоит проверить, правильно ли программа отсортировала студентов по среднему баллу в порядке возрастания.

|  |  |
| --- | --- |
| От А до П   1. Иванов Иван Иванович: (5+5+4) / 3 = 4,66667 2. Муханов Матвей Эдуардович: (5+4+5) / 3 = 4,66667 3. Кутьин Захар Сергеевич: (5) / 1 = 5 | От Р до Я   1. Яшкина Яна Александровна: (5+3+4+5) / 4 = 4,25 2. Ульмова Владислава Александравна: (5+4) / 2 = 4,5 |

Таким образом, программа верно выполняет задание согласно варианту по тестовым данным.

# Вывод

При написании и выполнении курсовой работы, были закреплены навыки и знания, полученные при изучении дисциплины «Языки программирования» и языка программирования C++. В итоге на основе ООП была разработана программа, результат которой подтверждает задание по курсовой работе согласно варианту.

# Список литературы

1. Прата С. Язык программирования С. Лекции и упражнения. – Litres, 2021.
2. Лафоре Р. Объектно-ориентированное программирование в C++, 4 – е издание :[пер. с англ.]. – Издательский дом" Питер", 2018.
3. Страуструп Б. Программирование. Принципы и практика использования C++. – Litres, 2021.
4. Ravesli | Программирование для начинающих // URL: https://ravesli.com/
5. Microsoft | Cryptography Functions // URL: https://docs.microsoft.com/en-us/windows/win32/seccrypto/cryptography-functions

# Листинг

#include <iostream>

#include <fstream>

#include <ctime>

#include <string.h>

#include <windows.h>

#include <wincrypt.h>

#define PATH "Students.txt"

#define PATH\_enc "Students.txt.enc"

#define PATH1 "Students1.txt"

#define PATH2 "Students2.txt"

#define PATH\_NEW "Students.new.txt"

#define RUS SetConsoleCP(1251); SetConsoleOutputCP(1251);

#pragma comment(lib, "crypt32.lib")

using namespace std;

struct Sub {

char\* name = new char[21]();

int\* mark = new int();

};

class Mystr {

private:

char\* data = nullptr;

public:

Mystr(const char in[]) {

data = new char[strlen(in) + 1]();

for (int i = 0; i < strlen(in); i++) {

\*(data + i) = in[i];

}

data[strlen(data)] = '\0';

}

~Mystr() {

delete[] data;

}

void operator += (const char other[]) {

char\* temp = new char[strlen(data) + strlen(other) + 1]();

int i = 0;

for (; i < strlen(data); i++) {

\*(temp + i) = \*(data + i);

}

for (int j = 0; j < strlen(other); j++) {

\*(temp + i + j) = \*(other + j);

}

temp[strlen(temp)] = '\0';

delete[] data;

data = new char[strlen(temp) + 1]();

for (int i = 0; i < strlen(temp); i++) {

\*(data + i) = \*(temp + i);

}

data[strlen(data)] = '\0';

}

char\* Get() {

return data;

}

};

class Crypt {

private:

char\* Gen\_pass() {

srand(time(NULL));

char\* pass = new char[17];

for (int i = 0; i < 16; ++i)

{

switch (rand() % 3) {

case 0:

pass[i] = rand() % 10 + '0';

break;

case 1:

pass[i] = rand() % 26 + 'A';

break;

case 2:

pass[i] = rand() % 26 + 'a';

}

}

pass[16] = '\0';

return pass;

}

public:

void Encrypt() {

Mystr PATH\_ENC(PATH);

PATH\_ENC += ".enc";

ifstream File;

File.open(PATH, ios::binary);

ofstream File\_enc;

File\_enc.open(PATH\_ENC.Get(), ios::binary | ios::app);

File\_enc.seekp(0, ios::beg);

int file\_length;

File.seekg(0, ios::end);

file\_length = File.tellg();

File.seekg(0, ios::beg);

char\* szPassword = Gen\_pass();

int dwLength = strlen(szPassword);

File\_enc.write((char\*)&dwLength, sizeof(dwLength));

File\_enc.write((char\*)szPassword, dwLength + 1);

HCRYPTPROV hProv;

HCRYPTKEY hKey;

HCRYPTHASH hHash;

if (!CryptAcquireContext(&hProv, NULL, NULL, PROV\_RSA\_AES, CRYPT\_VERIFYCONTEXT))

{

cout << "Error during CryptAcquireContext!";

}

if (!CryptCreateHash(hProv, CALG\_MD5, 0, 0, &hHash))

{

cout << "Error during CryptCreateHash!";

}

if (!CryptHashData(hHash, (BYTE\*)szPassword, (DWORD)dwLength, 0))

{

cout << "Error during CryptHashData!";

}

if (!CryptDeriveKey(hProv, CALG\_RC4, hHash, CRYPT\_EXPORTABLE, &hKey))

{

cout << "Error during CryptDeriveKey!";

}

size\_t enc\_length = 8;

DWORD dwBlockLen = 1000 - 1000 % enc\_length;

DWORD dwBufferLen = 0;

if (enc\_length > 1)

{

dwBufferLen = dwBlockLen + enc\_length;

}

else

{

dwBufferLen = dwBlockLen;

}

int count = 0;

bool final = false;

while (count != file\_length) {

if (file\_length - count < dwBlockLen) {

dwBlockLen = file\_length - count;

final = true;

}

BYTE\* temp = new BYTE[dwBufferLen]();

File.read((char\*)temp, dwBlockLen);

if (!CryptEncrypt(hKey, NULL, final, 0, temp, &dwBlockLen, dwBufferLen))

{

cout << "Error during CryptEncrypt. \n";

}

File\_enc.write((char\*)temp, dwBlockLen);

count = count + dwBlockLen;

}

if (hHash)

{

if (!(CryptDestroyHash(hHash)))

cout << "Error during CryptDestroyHash";

}

if (hKey)

{

if (!(CryptDestroyKey(hKey)))

cout << "Error during CryptDestroyKey";

}

if (hProv)

{

if (!(CryptReleaseContext(hProv, 0)))

cout << "Error during CryptReleaseContext";

}

File.close();

File\_enc.close();

if (remove(PATH) != 0) {

cout << "ERROR -- ошибка при удалении файла\n";

}

}

void Decrypt() {

Mystr PATH\_ENC(PATH);

PATH\_ENC += ".enc";

ofstream File;

File.open(PATH, ios::binary | ios::app);

ifstream File\_enc;

File\_enc.open(PATH\_ENC.Get(), ios::binary);

int file\_length;

File\_enc.seekg(0, ios::end);

file\_length = File\_enc.tellg();

File\_enc.seekg(0, ios::beg);

if (file\_length == -1 || file\_length == 0) {

return;

}

int dwLength;

File\_enc.read((char\*)&dwLength, sizeof(dwLength));

char\* szPassword = new char[dwLength];

File\_enc.read((char\*)szPassword, dwLength + 1);

HCRYPTPROV hProv;

HCRYPTKEY hKey;

HCRYPTHASH hHash;

if (!CryptAcquireContext(&hProv, NULL, NULL, PROV\_RSA\_AES, CRYPT\_VERIFYCONTEXT))

{

cout << "Error during CryptAcquireContext!";

}

if (!CryptCreateHash(hProv, CALG\_MD5, 0, 0, &hHash))

{

cout << "Error during CryptCreateHash!";

}

if (!CryptHashData(hHash, (BYTE\*)szPassword, (DWORD)dwLength, 0))

{

cout << "Error during CryptHashData!";

}

if (!CryptDeriveKey(hProv, CALG\_RC4, hHash, CRYPT\_EXPORTABLE, &hKey))

{

cout << "Error during CryptDeriveKey!";

}

size\_t enc\_length = 8;

DWORD dwBlockLen = 1000 - 1000 % enc\_length;

DWORD dwBufferLen = 0;

if (enc\_length > 1)

{

dwBufferLen = dwBlockLen + enc\_length;

}

else

{

dwBufferLen = dwBlockLen;

}

int count = sizeof(dwLength) + strlen(szPassword) + 1;

bool final = false;

while (count != file\_length) {

if (file\_length - count < dwBlockLen) {

dwBlockLen = file\_length - count;

final = true;

}

BYTE\* temp = new BYTE[dwBlockLen];

File\_enc.read((char\*)temp, dwBlockLen);

if (!CryptDecrypt(hKey, 0, final, 0, temp, &dwBlockLen))

{

cout << "Error during CryptEncrypt. \n";

}

File.write((char\*)temp, dwBlockLen);

count = count + dwBlockLen;

}

if (hHash)

{

if (!(CryptDestroyHash(hHash)))

cout << "Error during CryptDestroyHash";

}

if (hKey)

{

if (!(CryptDestroyKey(hKey)))

cout << "Error during CryptDestroyKey";

}

if (hProv)

{

if (!(CryptReleaseContext(hProv, 0)))

cout << "Error during CryptReleaseContext";

}

File.close();

File\_enc.close();

if (remove(PATH\_ENC.Get()) != 0) {

cout << "ERROR -- ошибка при удалении файла\n";

}

}

};

class Functions {

public:

virtual void studentsPrint(int val) = 0;

void print(char\* val) {

cout << val;

}

void print(int val) {

cout << val;

}

void print(const char val[]) {

cout << val;

}

void cleanCin() {

cin.seekg(0, ios::end);

cin.clear();

}

void checkEmptyStr(char\* in, int len) {

char\* buf = new char[len+1]();

cleanCin();

cin.get(buf, len+1);

cleanCin();

bool flag = false;

int count = 0;

int k = 0;

while ((\*(buf + k)) != '\0') {

count++;

if (count >= len) {

k = 0;

count = 0;

flag = false;

print("Вы ввели недопустимое количество символов. Повторите ввод.\n");

cleanCin();

cin.get(buf, len+1);

cleanCin();

continue;

}

else {

flag = true;

}

k++;

}

cleanCin();

if (strlen(buf) == 0) {

print("Вы ввели пустую строку! Повторите ввод:\n");

checkEmptyStr(in, len);

}

if (flag){

for (int i = 0; i < len; i++) {

in[i] = \*(buf + i);

}

delete[] buf;

}

}

bool checkMinus(char\* in, int len) {

if ((in[0] == '-') || (in[strlen(in)-1] == '-')) {

print("Вводимое значение не может начинаться/заканчиваться с '-'. Повторите ввод.\n");

return false;

}

else {

return true;

}

}

void backMenu() {

char temp[2];

print("Чтобы вернуться в меню нажмите [Enter]");

cleanCin();

cin.get(temp, 2);

cleanCin();

}

int checkValue(int value) {

int \_value = value;

while (true) {

if (cin.fail()) {

cin.clear();

cin.ignore(32767, '\n');

print("Ошибка! Ожидалось число, повторите ввод------>");

cin >> value;

\_value = value;

continue;

}

else

break;

}

return \_value;

}

bool checkMasChar(char\* line, int len) {

int i = 0;

int x = 1;

while (line[i] != '\0') {

if (!((line[i] >= 'а' && line[i] <= 'я') || (line[i] >= 'А' && line[i] <= 'Я'))) {

i = 0;

x = 0;

print("Ожидались символы. Повторите ввод. \n\n");

break;

}

i++;

}

if (x == 0) {

return false;

}

else {

return true;

}

}

};

class Student : Functions {

friend class File;

public:

Student() {

surname = new char[31]();

name = new char[31]();

middlename = new char[31]();

day = new int(0);

month = new int(0);

year = new int(0);

yearOfAdmission = new int(0);

sex = new char[2]();

faculty = new char[25]();

departament = new char[25]();

group = new char[11]();

recordBookNumber = new char[21]();

}

~Student() {

delete surname;

delete name;

delete middlename;

delete day;

delete month;

delete year;

delete yearOfAdmission;

delete faculty;

delete departament;

delete group;

delete recordBookNumber;

}

void Set() {

print("Фамилия: ");

Set(surname, 31);

if (!strcmp(surname, "-1")) {

return;

}

while (!(checkMasChar(surname, 31))) {

print("Фамилия: ");

Set(surname, 31);

}

print("Имя: ");

Set(name, 31);

while (!(checkMasChar(name, 31))) {

print("Имя: ");

Set(name, 31);

}

print("Отчество: ");

Set(middlename, 31);

while (!(checkMasChar(middlename, 31))) {

print("Отчество: ");

Set(middlename, 31);

}

print("Дата рождения [дд/мм/гггг]: ");

while (!checkBirthDay()) {

print("Дата Рождения [дд/мм/гггг]: ");

};

print("Год поступления [1965-2021]: ");

checkAdmissionYear();

print("Пол [М/Ж]: ");

\*sex = checkSex();

\*(sex + 1) = '\0';

print("Факультет: ");

Set(faculty, 25);

while (!(checkMasChar(faculty, 25))) {

print("Факультет: ");

Set(faculty, 25);

}

print("Кафедра: ");

Set(departament, 25);

while (!(checkMinus(departament, 25))) {

print("Кафедра: ");

Set(departament, 25);

}

print("Группа: ");

Set(group, 11);

while (!(checkMinus(group, 11))) {

print("Группа: ");

Set(group, 11);

}

print("Номер зачетной книжки: ");

checkEmptyStr(recordBookNumber, 21);

cleanCin();

while (!(checkMinus(recordBookNumber, 21))) {

print("Номер зачетной книжки: ");

Set(recordBookNumber, 21);

}

while (!checkRecordBookNumber()) {

print("Такой номер уже есть в базе\nПопробуйте ввести ещё раз: ");

checkEmptyStr(recordBookNumber, 21);

cleanCin();

}

}

bool Edit() {

int ans;

cin >> ans;

cleanCin();

switch (ans) {

case 1:

print("Фамилия: ");

Set(surname, 31);

while (!(checkMasChar(surname, 31))) {

print("Фамилия: ");

Set(surname, 31);

}

break;

case 2:

print("Имя: ");

Set(name, 31);

while (!(checkMasChar(name, 31))) {

print("Имя: ");

Set(name, 31);

}

break;

case 3:

print("Отчество: ");

Set(middlename, 31);

while (!(checkMasChar(middlename, 31))) {

print("Отчество: ");

Set(middlename, 31);

}

break;

case 4:

print("Дата рождения [дд/мм/гггг]: ");

while (!checkBirthDay()) {

print("Дата рождения [дд/мм/гггг]: ");

};

break;

case 5:

print("Год Поступления [1980-2021]: ");

checkAdmissionYear();

break;

case 6:

print("Пол [м/ж]: ");

\*sex = checkSex();

\*(sex + 1) = '\0';

break;

case 7:

print("Факультет: ");

Set(faculty, 25);

while (!(checkMasChar(faculty, 25))) {

print("Факультет: ");

Set(faculty, 31);

}

break;

case 8:

print("Кафедра: ");

Set(departament, 25);

while (!(checkMinus(departament, 25))) {

print("Кафедра: ");

Set(departament, 25);

}

break;

case 9:

print("Группа: ");

Set(group, 11);

while (!(checkMinus(group, 11))) {

print("Группа: ");

Set(group, 11);

}

break;

case 10:

print("Номер зачетной книжки: ");

checkEmptyStr(recordBookNumber, 21);

cleanCin();

while (!(checkMinus(recordBookNumber, 21))) {

print("Номер зачетной книжки: ");

Set(recordBookNumber, 21);

}

while (!checkRecordBookNumber()) {

print("Такой номер зачетной книжки уже существует\n");

print("Введите Номер зачетной книжки: ");

checkEmptyStr(recordBookNumber, 21);

cleanCin();

}

break;

case 11: return true;

default: {

print("Введен неверный вариант\n");

Edit();

}

}

return false;

}

private:

char\* surname = nullptr;

char\* name = nullptr;

char\* middlename = nullptr;

int\* day = nullptr;

int\* month = nullptr;

int\* year = nullptr;

int\* yearOfAdmission = nullptr;

char\* sex = nullptr;

char\* faculty = nullptr;

char\* departament = nullptr;

char\* group = nullptr;

char\* recordBookNumber = nullptr;

void studentsPrint(int val) override {

return;

}

void Set(char\* in, int len) {

checkEmptyStr(in, len);

cleanCin();

}

bool checkBirthDay() {

char\* temp = new char[12]();

\*day = 0;

\*month = 0;

\*year = 0;

cin.get(temp, 12);

cleanCin();

bool flag = true;

int counter1 = 0;

for (int i = 0; \*(temp + i) != '\0'; i++) {

counter1++;

flag = true;

while (flag) {

if (counter1 >= 11) {

cout << "Введена слишком длинная строка.\nПовторите ввод в формате [дд/мм/гггг]:";

cleanCin();

cin.get(temp, 12);

cleanCin();

flag = true;

i = 0;

counter1 = 0;

}

if ((\*(temp + i) >= '0' && \*(temp + i) <= '9') || \*(temp + i) == ' ' || \*(temp + i) == '.' || \*(temp + i) == '/' || \*(temp + i) == '\0' || \*(temp + i) == '\n') {

flag = false;

}

else {

cout << "В введённой дате рождения содержатся символы отличные от цифр или символов . / [пробел]. Повторите ввод!\nДата рождения [дд/мм/гггг]:";

cleanCin();

cin.get(temp, 12);

cleanCin();

flag = true;

i = 0;

counter1 = 0;

}

}

}

for (int i = 0; \*(temp + i) != '\0'; i++) {

if (\*(temp + i) >= 48 && \*(temp + i) <= 57 && ((i >= 0 && i <= 1) || (i >= 3 && i <= 4) || (i >= 6 && i <= 9))) {

switch (i) {

case 0: case 1:

\*day = \*day \* 10 + \*(temp + i) - 0x30;

break;

case 3: case 4:

\*month = \*month \* 10 + \*(temp + i) - 0x30;

break;

case 6: case 7: case 8: case 9:

\*year = \*year \* 10 + \*(temp + i) - 0x30;

break;

}

}

}

delete[] temp;

if (checkDate(\*day, \*month, \*year)) return true;

else return false;

}

void checkAdmissionYear() {

cin >> \*yearOfAdmission;

while (\*yearOfAdmission <= \*year || \*yearOfAdmission - \*year < 15 || \*yearOfAdmission < 1965 || \*yearOfAdmission > 2021) {

if (cin.fail()) {

cin.clear();

cin.ignore(32767, '\n');

print("Ошибка! Введите корректные данные\nГод поступления [1965-2021]: ");

cin >> \*yearOfAdmission;

continue;

}

if (\*yearOfAdmission - \*year <= 15 || \*yearOfAdmission < 1965 || \*yearOfAdmission > 2021 || \*yearOfAdmission <= \*year) {

print("Поступление невозможно в таком возрасте! Повторите ввод\nГод поступления [1965-2021]:");

cin.ignore(32767, '\n');

cin >> \*yearOfAdmission;

}

}

cleanCin();

}

bool checkRecordBookNumber() {

int\* length = new int(0);

int\* len\_file = new int(0);

char\* buf = new char[21];

Crypt crypt;

crypt.Decrypt();

ifstream File;

File.open(PATH, ios::binary);

File.seekg(0, ios::end);

\*len\_file = File.tellg();

File.seekg(0, ios::beg);

while (\*length != \*len\_file) {

File.seekg(171, ios::cur);

File.read(buf, 21);

if (!strcmp(buf, recordBookNumber)) {

File.close();

crypt.Encrypt();

return false;

}

File.seekg(2290, ios::cur);

\*length += 2482;

}

File.close();

crypt.Encrypt();

return true;

}

bool checkDate(int day, int month, int year) {

if (day != 0 && month != 0 && year != 0) {

if (year >= 1950 && year <= 2006) {

if (month >= 1 && month <= 12) {

switch (month) {

case 1: case 3: case 5: case 7: case 8: case 10: case 12:

if (day >= 1 && day <= 31) {

return true;

}

break;

case 2:

if (year % 4 != 0 || year % 100 == 0 && year % 400 != 0) {

if (day >= 1 && day <= 28) {

return true;

}

}

else {

if (day >= 1 && day <= 29) {

return true;

}

}

break;

case 4: case 6: case 9: case 11:

if (day >= 1 && day <= 30) {

return true;

}

break;

default:

print("Ошибка! Повторите ввод\n");

break;

}

print("Ошибка! Введите день месяца правильно\n");

}

else {

print("Ошибка! Введите месяц от 1 до 12\n");

}

}

else {

print("Ошибка! Введите год рождения от 1950 до 2006\n");

}

}

return false;

}

char checkSex() {

char value;

while (true) {

cin >> value;

if (value == 'М' || value == 'Ж' || value == 'ж' || value == 'м') {

cleanCin();

return value;

}

print("Ошибка! Вводите только буквы М(м)/Ж(ж) \n");

cleanCin();

}

}

};

class Session : Functions {

friend class File;

public:

Session() {

session\_count = new int(0);

sub\_count = nullptr;

}

~Session() {

delete session\_count;

delete sub\_count;

}

void setSession() {

setSessionCount();

setSubjCount();

setSubjects();

}

bool editSession() {

int pos = -1;

int\* temp\_2 = nullptr;

int sum = 0;

int ans, num = -1, ses;

cin >> ans;

cleanCin();

if (ans == 1) {

if (\*session\_count < 9) {

system("cls");

print("Введите количество предметов в новой сессии: ");

cin >> sub\_count[\*session\_count];

cleanCin();

for (int j = 0; j < sub\_count[\*session\_count]; j++) {

subject[\*session\_count][j].name = new char[21]();

subject[\*session\_count][j].mark = new int(0);

print("Укажите название ");

print(j + 1);

print("-го предмета в ");

print(\*session\_count + 1);

print("-й сессии: ");

checkEmptyStr(subject[\*session\_count][j].name, 21);

cleanCin();

print("Введите оценку за ");

cout << subject[\*session\_count][j].name;

print(": ");

int buf;

while (true) {

cleanCin();

cin >> buf;

cleanCin();

if (buf >= 2 && buf <= 5) {

\*subject[\*session\_count][j].mark = buf;

break;

}

print("Неверные данные! Вводите значения от 2 до 5\n");

}

}

\*(session\_count) += 1;

}

else {

print("Достигнуто максимальное количество сессий\n");

backMenu();

}

}

else if (ans == 2 || ans == 3 || ans == 4) {

print("Введите номер сессии -----> ");

cin >> ses;

if (!(ses != 0 && ses <= \*session\_count)) {

print("Номер такой сессии не найден, повторите ввод\n");

backMenu();

return false;

}

ses -= 1;

if (ans == 2 || ans == 4)

{

print("Введите номер предмета -----> ");

cin >> num;

if (!(num <= sub\_count[ses] && num != 0)) {

print("Номер такого предмета не найден, повторите ввод\n");

backMenu();

return false;

}

num -= 1;

}

if (ans == 3) {

if (\*(sub\_count + ses) == 10) {

print("Достигнуто максимальное кол-во предметов\n");

backMenu();

return false;

}

}

system("cls");

if (ans == 2) {

print("Выбранный предмет: ");

cout << subject[ses][num].name << "\nОценка: " << \*subject[ses][num].mark << "\n";

print("\nЧто нужно изменить:\n\n[1] Название предмета\n[2] Оценку по предмету\n[3] Вернуться в блок редактирования\n");

print("-----> ");

int ans1;

cin >> ans1;

switch (ans1) {

case 1:

print("Введите название предмета: ");

checkEmptyStr(subject[ses][num].name, 21);

break;

case 2:

print("Введите оценку: ");

int buf;

while (true) {

cleanCin();

cin >> buf;

cleanCin();

if (buf >= 2 && buf <= 5) {

\*subject[ses][num].mark = buf;

break;

}

print("Неверные данные! Вводите значения от 2 до 5\n");

}

break;

case 3:

return false;

}

}

if (ans == 3) {

print("Введите название нового предмета: ");

checkEmptyStr(subject[ses][sub\_count[ses]].name, 21);

cleanCin();

print("Введите оценку за ");

cout << subject[ses][sub\_count[ses]].name;

print(": ");

int buf;

while (true) {

cleanCin();

cin >> buf;

cleanCin();

if (buf >= 2 && buf <= 5) {

\*subject[ses][sub\_count[ses]].mark = buf;

break;

}

print("Неверные данные! Вводите значения от 2 до 5\n");

}

sub\_count[ses] += 1;

}

if (ans == 4) {

if (sub\_count[ses] > 1) {

subject[ses][num].name = subject[ses][sub\_count[ses] - 1].name;

subject[ses][num].mark = subject[ses][sub\_count[ses] - 1].mark;

}

else {

subject[ses][num].name = (char\*)"\0";

subject[ses][num].mark = 0;

(\*(session\_count))--;

}

sub\_count[ses]--;

}

}

else if (ans == 5) {

return true;

}

else {

print("Такого варианта не найдено\n");

backMenu();

return false;

}

return false;

}

private:

int\* session\_count = nullptr;

int\* sub\_count = nullptr;

Sub subject[9][10];

void studentsPrint(int val) override {

return;

}

void setSessionCount() {

print("Введите количество семестров: ");

int value;

cin >> value;

while (value < 1 || value > 9) {

if (cin.fail()) {

cin.clear();

cin.ignore(32767, '\n');

print("Ошибка! Вводите значения от 1 до 9 \n-----> ");

cin >> value;

continue;

}

if (value < 1 || value > 9) {

print("Ошибка! Вводите значения от 1 до 9\n-----> ");

cin.ignore(32767, '\n');

cin >> value;

}

}

\*session\_count = value;

}

void setSubjCount() {

sub\_count = new int[9]();

bool flag = true;

for (int i = 0; i < \*session\_count; i++) {

bool flag = true;

print("Введите количество предметов в ");

print(i + 1);

print("-м семестре: ");

int\* buf = new int();

cin >> \*buf;

while (\*buf < 1 || \*buf > 10) {

if (cin.fail()) {

cin.clear();

cin.ignore(32767, '\n');

print("Ошибка! Вводите значения от 1 до 10 \n-----> ");

cin >> \*buf;

continue;

}

if (\*buf < 1 || \*buf > 10) {

print("Ошибка! Вводите значения от 1 до 10\n-----> ");

cin.ignore(32767, '\n');

cin >> \*buf;

}

}

\*(sub\_count + i) = \*buf;

}

}

void setSubjects() {

for (int i = 0; i < \*session\_count; i++) {

for (int j = 0; j < sub\_count[i]; j++) {

subject[i][j].name = new char[21]();

subject[i][j].mark = new int(0);

print("Укажите название ");

print(j + 1);

print("-го предмета в ");

print(i + 1);

print("-й сессии: ");

checkEmptyStr(subject[i][j].name, 21);

cleanCin();

print("Введите оценку за ");

cout << subject[i][j].name;

print(": ");

int buf;

while (true) {

cleanCin();

cin >> buf;

cleanCin();

if (buf >= 2 && buf <= 5) {

\*subject[i][j].mark = buf;

break;

}

print("Неверные данные! Вводите значения от 2 до 5\n");

}

}

}

}

};

class File : Functions {

public:

File() {

file\_length = new int(0);

count = new int(0);

sum = new int(0);

rec\_book\_num = new char[21]();

edit\_student = new Student();

edit\_session = new Session();

}

~File() {

delete file\_length;

delete count;

delete sum;

}

void addStudent() {

Student\* student = new Student;

Session\* session = new Session;

student->Set();

if (!strcmp(student->surname, "-1")) {

delete student;

delete session;

return;

}

session->setSession();

Crypt crypt;

crypt.Decrypt();

ofstream file(PATH, ios::binary | ios::app);

file.write(student->surname, 31);

file.write(student->name, 31);

file.write(student->middlename, 31);

file.write((char\*)student->day, 4);

file.write((char\*)student->month, 4);

file.write((char\*)student->year, 4);

file.write((char\*)student->sex, 1);

file.write((char\*)student->yearOfAdmission, 4);

file.write(student->faculty, 25);

file.write(student->departament, 25);

file.write(student->group, 11);

file.write(student->recordBookNumber, 21);

file.write((char\*)session->session\_count, 4);

for (int i = 0; i < 9; i++) {

file.write((char\*)(&\*(session->sub\_count + i)), 4);

}

for (int i = 0; i < 9; i++) {

for (int j = 0; j < 10; j++) {

file.write((char\*)session->subject[i][j].name, 21);

file.write((char\*)session->subject[i][j].mark, 4);

}

}

delete student;

delete session;

file.close();

crypt.Encrypt();

}

void editStudent() {

ofstream test("Students.new.txt", ios::binary);

test.close();

Crypt crypt;

int ans;

int ans\_2;

if (!isThereStudent()) return;

system("cls");

print("Редактирование информации о студенте\n\n");

printChoice(1);

while (!findStudent()) {

print("Такой студент не найден\n");

print("[1] Чтобы ввести номер другой зачетной книжки, нажмите [1]\n");

print("[2] Чтобы вернуться в блок редактирования, нажмите [2]\n-----> ");

cin >> ans\_2;

while (ans\_2 != 1 && ans\_2 != 2) {

print("[1] Чтобы ввести другого студента\n");

print("[2] Чтобы вернуться в блок редактирования, нажмите [2]\n-----> ");

cin >> ans\_2;

}

if (ans\_2 == 2) return;

}

if (pos\_f == -1) {

return;

}

bool flag = true;

while (flag) {

print("[1] Чтобы редактировать данные о студенте, нажмите [1]\n");

print("[2] Чтобы редактировать данные о сессии студента, нажмите [2]\n");

print("[3] Чтобы вернуться в меню, нажмите [3]\n-----> ");

cin >> ans;

for (int i = 0; i < \*count; i++) {

pos = i;

getStudent(PATH\_enc);

if (i != pos\_f) {

fileWriter(PATH\_NEW);

}

else {

switch (ans) {

case 1:

while (true) {

system("cls");

studentsPrint(2);

print("Это блок редактирования информации о студенте, выберите нужный пункт\n\n");

print("[1] Чтобы редактировать фамилию студента, нажмите [1]\n");

print("[2] Чтобы редактировать имя студента, нажмите [2]\n");

print("[3] Чтобы редактировать отчество студента, нажмите [3]\n");

print("[4] Чтобы редактировать дату рождения студента, нажмите [4]\n");

print("[5] Чтобы редактировать год приема студента в университет, нажмите [5]\n");

print("[6] Чтобы редактировать пол студента, нажмите [6]\n");

print("[7] Чтобы редактировать факультет студента, нажмите [7]\n");

print("[8] Чтобы редактировать кафедру студента, нажмите [8]\n");

print("[9] Чтобы редактировать группу студента, нажмите [9]\n");

print("[10] Чтобы редактировать номер зачетной книжки студента, нажмите [10]\n");

print("[11] Чтобы сохранить изменения, нажмите [11]\n\n");

print("-----> ");

if (edit\_student->Edit()) {

break;

}

}

break;

case 2:

while (true) {

system("cls");

print("Это блок редактирования информации о сессии студента\n\n");

studentsPrint(1);

studentsPrint(3);

print("[1] Чтобы добавить новую сессию, нажмите [1]\n");

print("[2] Чтобы редактировать информацию о предметах студента, нажмите [2]\n");

print("[3] Чтобы добавить новый предмет в сессию, нажмите [3]\n");

print("[4] Чтобы удалить предмет из сессии, нажмите [4]\n");

print("[5] Чтобы сохранить изменения, нажмите [5]\n\n");

print("-----> ");

if (edit\_session->editSession()) {

break;

};

}

break;

case 3:

flag = false;

break;

default:

if (remove("Students.new.txt") != 0) {

print("Ошибка при удалении файла!!!\n");

backMenu();

}

print("Такого варианта не найдено\n");

backMenu();

break;

}

fileWriter(PATH\_NEW);

}

}

if (remove("Students.txt.enc") != 0) {

print("Ошибка при удалении файла!!!\n");

backMenu();

}

if (rename("Students.new.txt", PATH) != 0) {

print("Ошибка при переименовании файла!!!\n");

backMenu();

}

crypt.Encrypt();

flag = false;

}

delete rec\_book\_num;

}

void delStudent() {

fstream file(PATH\_NEW, ios::binary | ios::out);

file.close();

Crypt crypt;

if (!isThereStudent()) return;

pos = -1;

rec\_book\_num = new char[21]();

printChoice(1);

print("Введите номер зачетной книжки(-1 чтобы вернуться назад) >> ");

cleanCin();

checkEmptyStr(rec\_book\_num, 21);

cleanCin();

if (!strcmp(rec\_book\_num, "-1")) {

return;

}

for (int i = 0; i < \*count; i++) {

pos = i;

getStudent(PATH\_enc);

if (!strcmp(rec\_book\_num, edit\_student->recordBookNumber)) {

pos\_f = i;

break;

}

}

int pos\_w = 0, pos\_r = 0;

if (pos\_f != -1) {

for (int i = 0; i < \*count; i++) {

pos = pos\_r;

getStudent(PATH\_enc);

if (i != pos\_f) {

pos = pos\_w;

fileWriter(PATH\_NEW);

pos\_w += 1;

}

pos\_r++;

}

if (remove(PATH\_enc) != 0) {

print("Ошибка при удалении файла!!!\n");

backMenu();

};

if (rename(PATH\_NEW, PATH) != 0) {

print("Ошибка при переименовании файла!!!\n");

backMenu();

}

crypt.Encrypt();

}

else {

print("Такой студент не найден\n");

delStudent();

}

delete rec\_book\_num;

}

void printChoice(int rez) {

if (!isThereStudent()) return;

for (int i = 0; i < \*count; i++) {

pos = i;

getStudent(PATH\_enc);

switch (rez) {

case 1:

studentsPrint(1);

break;

case 2:

studentsPrint(2);

break;

case 3:

studentsPrint(1);

studentsPrint(3);

break;

case 4:

studentsPrint(2);

studentsPrint(3);

break;

}

}

}

void task(int up, int down) {

fstream file(PATH1, ios::binary | ios::out);

file.close();

file.open(PATH2, ios::binary | ios::out);

file.close();

if (!isThereStudent()) {

return;

}

for (int i = 0; i < \*count; i++) {

pos = i;

getStudent(PATH\_enc);

if ((((\*(edit\_student->surname)) >= 'А') && (((\*(edit\_student->surname)) <= 'П') || ((\*(edit\_student->surname)) >= 'а')) && ((\*(edit\_student->surname)) <= 'п'))) {

pos = count1;

count1++;

fileWriter(PATH1);

}

else if ((((\*(edit\_student->surname)) >= 'Р') && (((\*(edit\_student->surname)) <= 'р') || ((\*(edit\_student->surname)) >= 'Я')) && ((\*(edit\_student->surname)) <= 'я'))) {

pos = count2;

count2++;

fileWriter(PATH2);

}

}

print("\n\nСтуденты от А до П: \n\n");

print("============================================================================================\n\n");

sort(PATH1, count1);

for (int i = 0; i < count1; i++) {

pos = i;

getStudent(PATH1);

if ((\*(edit\_student->year) >= down) && (\*(edit\_student->year) <= up)) {

studentsPrint(2);

double sum = 0;

double k = 0;

for (int i = 0; i < \*edit\_session->session\_count; i++) {

for (int j = 0; j < edit\_session->sub\_count[i]; j++) {

sum += \*edit\_session->subject[i][j].mark;

k++;

}

}

cout << "Средний балл студента: " << sum / k << endl;

}

}

print("============================================================================================\n\n");

print("Студенты от Р до Я: \n\n");

print("============================================================================================\n\n");

sort(PATH2, count2);

for (int i = 0; i < count2; i++) {

pos = i;

getStudent(PATH2);

if ((\*(edit\_student->year) >= down) && (\*(edit\_student->year) <= up)) {

studentsPrint(2);

double sum = 0;

double k = 0;

for (int i = 0; i < \*edit\_session->session\_count; i++) {

for (int j = 0; j < edit\_session->sub\_count[i]; j++) {

sum += \*edit\_session->subject[i][j].mark;

k++;

}

}

cout << "Средний балл студента: " << sum / k << endl;

}

}

print("============================================================================================\n\n");

}

void sort(const char\* file\_name, int count) {

int min\_index = -1;

int temp\_pos = -1;

Student\* min\_student = new Student();

Student\* temp\_std = new Student();

Session\* min\_session = new Session();

Session\* temp\_ses = new Session();

for (int i = 0; i < count - 1; i++)

{

min\_index = i;

pos = i;

edit\_student = min\_student;

edit\_session = min\_session;

getStudent(file\_name);

for (int j = i + 1; j < count; j++) {

pos = j;

edit\_student = temp\_std;

edit\_session = temp\_ses;

getStudent(file\_name);

if (!min\_sr(\*min\_session, \*temp\_ses)) {

min\_index = pos;

edit\_student = min\_student;

edit\_session = min\_session;

getStudent(file\_name);

}

}

if (i != min\_index) {

pos = i;

edit\_student = temp\_std;

edit\_session = temp\_ses;

getStudent(file\_name);

edit\_student = min\_student;

edit\_session = min\_session;

fileWriter(file\_name);

pos = min\_index;

edit\_student = temp\_std;

edit\_session = temp\_ses;

fileWriter(file\_name);

}

}

}

bool min\_sr(Session& t1, Session& t2) {

double sr\_1 = 0, sr\_2 = 0;

int count\_1 = 0, count\_2 = 0;

for (int i = 0; i < \*t1.session\_count; i++) {

for (int j = 0; j < t1.sub\_count[i]; j++)

{

sr\_1 += \*t1.subject[i][j].mark;

}

count\_1 += t1.sub\_count[i];

}

for (int i = 0; i < \*t2.session\_count; i++) {

for (int j = 0; j < t2.sub\_count[i]; j++)

{

sr\_2 += \*t2.subject[i][j].mark;

}

count\_2 += t2.sub\_count[i];

}

sr\_1 = sr\_1 / count\_1;

sr\_2 = sr\_2 / count\_2;

if (sr\_1 < sr\_2) {

return true;

}

else {

return false;

}

}

private:

Student\* edit\_student = nullptr;

Session\* edit\_session = nullptr;

int\* file\_length;

int pos = -1;

int pos\_f = -1;

int count1 = 0, count2 = 0;

int\* count;

int\* sum;

char\* rec\_book\_num;

bool findStudent() {

pos\_f = -1;

rec\_book\_num = new char[21]();

print("Введите номер зачетной книжки(-1 чтобы вернуться обратно) >> ");

cleanCin();

checkEmptyStr(rec\_book\_num, 21);

cleanCin();

if (!strcmp(rec\_book\_num, "-1")) {

return true;

}

for (int i = 0; i < \*count; i++) {

pos = i;

getStudent(PATH\_enc);

if (!strcmp(rec\_book\_num, edit\_student->recordBookNumber)) {

pos\_f = i;

return true;

}

}

return false;

}

void getStudent(const char\* name\_file) {

Crypt crypt;

const char\* name;

if (!strcmp(name\_file, "Students.txt.enc")) {

crypt.Decrypt();

name = "Students.txt";

}

else {

name = name\_file;

}

ifstream File;

File.open(name, ios::binary);

File.seekg(0, ios::end);

\*file\_length = File.tellg();

File.seekg(pos \* 2482, ios::beg);

File.read(edit\_student->surname, 31);

File.read(edit\_student->name, 31);

File.read(edit\_student->middlename, 31);

File.read((char\*)edit\_student->day, 4);

File.read((char\*)edit\_student->month, 4);

File.read((char\*)edit\_student->year, 4);

File.read(edit\_student->sex, 1);

File.read((char\*)edit\_student->yearOfAdmission, 4);

File.read(edit\_student->faculty, 25);

File.read(edit\_student->departament, 25);

File.read(edit\_student->group, 11);

File.read(edit\_student->recordBookNumber, 21);

File.read((char\*)edit\_session->session\_count, 4);

edit\_session->sub\_count = new int[9];

for (int i = 0; i < 9; i++) {

File.read((char\*)(&\*(edit\_session->sub\_count + i)), 4);

}

for (int i = 0; i < 9; i++) {

for (int j = 0; j < 10; j++) {

File.read(edit\_session->subject[i][j].name, 21);

File.read((char\*)(edit\_session->subject[i][j].mark), 4);

}

}

File.close();

if (!strcmp(name\_file, "Students.txt.enc")) {

crypt.Encrypt();

}

}

void fileWriter(const char\* name\_file) {

fstream FILE\_NEW;

if (!strcmp(name\_file, PATH)) {

FILE\_NEW.open(name\_file, ios::binary | ios::app | ios::out);

}

else {

FILE\_NEW.open(name\_file, ios::binary | ios::in | ios::out);

FILE\_NEW.seekp(pos \* 2482, ios::beg);

}

FILE\_NEW.write(edit\_student->surname, 31);

FILE\_NEW.write(edit\_student->name, 31);

FILE\_NEW.write(edit\_student->middlename, 31);

FILE\_NEW.write((char\*)edit\_student->day, 4);

FILE\_NEW.write((char\*)edit\_student->month, 4);

FILE\_NEW.write((char\*)edit\_student->year, 4);

FILE\_NEW.write((char\*)edit\_student->sex, 1);

FILE\_NEW.write((char\*)edit\_student->yearOfAdmission, 4);

FILE\_NEW.write(edit\_student->faculty, 25);

FILE\_NEW.write(edit\_student->departament, 25);

FILE\_NEW.write(edit\_student->group, 11);

FILE\_NEW.write(edit\_student->recordBookNumber, 21);

FILE\_NEW.write((char\*)edit\_session->session\_count, 4);

for (int i = 0; i < 9; i++) {

FILE\_NEW.write((char\*)(&\*(edit\_session->sub\_count + i)), 4);

}

for (int i = 0; i < 9; i++)

{

for (int j = 0; j < 10; j++) {

FILE\_NEW.write((char\*)edit\_session->subject[i][j].name, 21);

FILE\_NEW.write((char\*)edit\_session->subject[i][j].mark, 4);

}

}

FILE\_NEW.close();

}

bool isThereStudent() {

fstream file("Students.txt.enc", ios::binary | ios::in);

file.seekg(0, ios::end);

if (file.tellg() == -1 || file.tellg() == 0) {

file.close();

print("Файл пустой, доступна только функция добавления студентов\n");

return false;

}

file.close();

Crypt\* crypt = new Crypt;

crypt->Decrypt();

ifstream File;

File.open(PATH, ios::binary);

File.seekg(0, ios::end);

\*file\_length = File.tellg();

\*count = \*(file\_length) / 2482;

File.close();

crypt->Encrypt();

return true;

}

void studentsPrint(int rez) override {

switch (rez) {

case 1:

print("--------------------------------------------------------------------------------------------\n\nДАННЫЕ О СТУДЕНТЕ\n\n");

cout << "ФИО: " << edit\_student->surname << " " << edit\_student->name << " " << edit\_student->middlename << "\n";

cout << "Номер зачетной книжки: " << edit\_student->recordBookNumber << "\n\n";

break;

case 2:

print("--------------------------------------------------------------------------------------------\n\nДАННЫЕ О СТУДЕНТЕ\n\n");

cout << "ФИО: " << edit\_student->surname << " " << edit\_student->name << " " << edit\_student->middlename << "\n";

cout << "Дата рождения: " << \*edit\_student->day << "." << \*edit\_student->month << "." << \*edit\_student->year << " Год приема: " << \*edit\_student->yearOfAdmission << "\n";

cout << "Пол: " << edit\_student->sex << " Факультет: " << edit\_student->faculty << " Кафедра: " << edit\_student->departament << " Группа: " << edit\_student->group << "\n";

cout << "Номер зачетной книжки: " << edit\_student->recordBookNumber << "\n\n";

break;

case 3:

print("ОЦЕНКИ \n\n");

int sum = 0;

for (int i = 0; i < \*edit\_session->session\_count; i++) {

cout << "Cессия " << i + 1 << "\n\n";

for (int j = 0; j < edit\_session->sub\_count[i]; j++) {

cout << j + 1 << ") " << edit\_session->subject[i][j].name << " ::: " << \*edit\_session->subject[i][j].mark << "\n";

sum++;

}

print("\n");

}

break;

}

}

};

class Menu : Functions {

public:

Menu() {

ans = new int;

file = new File;

}

~Menu() {

delete file;

delete ans;

}

bool hub() {

file = new File;

system("cls");

print("ДОБРО ПОЖАЛОВАТЬ В МЕНЮ ПРОГРАММЫ\n\n");

print("Нажмите соответствующую цифру, для выбора действия: \n\n");

print("[1] Чтобы выполнить задание, нажмите [1]\n");

print("[2] Чтобы добавить в базу нового студента, нажмите [2]\n");

print("[3] Чтобы удалить студента из базы, нажмите [3]\n");

print("[4] Чтобы изменить данные о студенте, нажмите [4]\n");

print("[5] Чтобы вывести на экран всю базу студентов, нажмите [5]\n");

print("[6] Чтобы выйти из программы, нажмите [6]\n\n-----> ");

cin >> \*ans;

while (\*ans < 1 || \*ans>6) {

if (cin.fail()) {

cin.clear();

cin.ignore(32767, '\n');

print("Ошибка! Введите номер пункта меню, который хотите вывести\n-----> ");

cin >> \*ans;

continue;

}

if (\*ans < 1 || \*ans>6) {

print("Вы ввели число не из диапозона [1;6]. Повторите ввод\n-----> ");

cin.ignore(32767, '\n');

cin >> \*ans;

}

}

cleanCin();

switch (\*ans) {

case 1: {

system("cls");

int yearUp, yearDown;

print("Чтобы вернуться в меню, введите в одно из полей интервала -1\n\n");

print("Введите нижний предел года рождения для поиска\n\n");

print("------>");

cin >> yearDown;

yearDown = checkValue(yearDown);

while ((yearDown < 1950) || (yearDown > 2006)) {

if ((yearDown < 1950) && (yearDown > -1)) {

print("Нижний предел задан не верно. Минимальный год рождения 1950.\n\n");

print("Введите нижний предел года рождения для поиска:\n\n");

print("------>");

cin >> yearDown;

yearDown = checkValue(yearDown);

continue;

}

if (yearDown > 2006) {

print("Нижний предел задан не верно. Введите значение от 1950 до 2006.\n\n");

print("Введите нижний предел года рождения для поиска:\n\n");

print("------>");

cin >> yearDown;

yearDown = checkValue(yearDown);

continue;

}

if (yearDown < -1) {

print("Ошибка! Введите -1 или нижний предел года рождения\n\n");

print("------>");

cin >> yearDown;

yearDown = checkValue(yearDown);

continue;

}

if (yearDown == -1) {

backMenu();

break;

}

}

if (yearDown == -1) {

break;

}

print("Введите верхний предел года рождения для поиска\n\n");

print("------>");

cin >> yearUp;

yearUp = checkValue(yearUp);

while ((yearUp < 1950) || (yearUp > 2006)) {

if (yearUp > 2006) {

print("Верхний предел задан неверно. Максимальный год рождения 2006.\n\n");

print("Введите верхний предел года рождения для поиска:\n\n");

print("------>");

cin >> yearUp;

yearUp = checkValue(yearUp);

continue;

}

if ((yearUp < 1950) && (yearUp > -1)) {

print("Верхний предел задан неверно. Введите значение от 1950 до 2006.\n\n");

print("Введите верхний предел года рождения для поиска:\n\n");

print("------>");

cin >> yearUp;

yearUp = checkValue(yearUp);

continue;

}

if (yearUp < -1) {

print("Ошибка! Введите -1 или верхний предел года рождения\n\n");

print("------>");

cin >> yearUp;

yearUp = checkValue(yearUp);

continue;

}

if (yearUp == -1) {

backMenu();

break;

}

}

if (yearUp == -1) {

break;

}

while (yearUp - yearDown < 0) {

print("Интервал задан неверно. Нижний предел должен быть МЕНЬШЕ верхнего.\n\n");

print("Введите интервал заново.\n\n");

print("Введите нижний предел года рождения для поиска:\n\n");

print("------>");

cin >> yearDown;

yearDown = checkValue(yearDown);

while ((yearDown < 1950) || (yearDown > 2006)) {

if (yearDown < 1950) {

print("Нижний предел задан не верно. Минимальный год рождения 1950.\n\n");

print("Введите нижний предел года рождения для поиска:\n\n");

print("------>");

cin >> yearDown;

yearDown = checkValue(yearDown);

continue;

}

if (yearDown > 2006) {

print("Нижний предел задан не верно. Введите значение от 1950 до 2006.\n\n");

print("Введите нижний предел года рождения для поиска:\n\n");

print("------>");

cin >> yearDown;

yearDown = checkValue(yearDown);

continue;

}

if (yearDown < -1) {

print("Ошибка! Введите -1 или нижний предел года рождения\n\n");

print("------>");

cin >> yearDown;

yearDown = checkValue(yearDown);

continue;

}

if (yearDown == -1) {

backMenu();

break;

}

}

if (yearDown == -1) {

break;

}

print("Введите верхний интервал года рождения для поиска:\n\n");

print("------>");

cin >> yearUp;

yearUp = checkValue(yearUp);

while ((yearUp < 1950) || (yearUp > 2006)) {

if (yearUp > 2006) {

print("Верхний предел задан неверно. Максимальный год рождения 2006.\n\n");

print("Введите верхний предел года рождения для поиска:\n\n");

print("------>");

cin >> yearUp;

yearUp = checkValue(yearUp);

continue;

}

if (yearUp < 1950) {

print("Верхний предел задан неверно. Введите значение от 1950 до 2006.\n\n");

print("Введите верхний предел года рождения для поиска:\n\n");

print("------>");

cin >> yearUp;

yearUp = checkValue(yearUp);

continue;

}

if (yearUp < -1) {

print("Ошибка! Введите -1 или верхний предел года рождения\n\n");

print("------>");

cin >> yearUp;

yearUp = checkValue(yearUp);

continue;

}

if (yearUp == -1) {

backMenu();

break;

}

}

if (yearUp == -1) {

break;

}

}

if ((yearDown == -1) || (yearUp == -1)) {

break;

}

file->task(yearUp, yearDown);

backMenu();

break;

}

case 2: {

system("cls");

print("Добавление нового студента(Введите -1, чтобы вернуться назад)\n\n");

file->addStudent();

backMenu();

break;

}

case 3: {

system("cls");

print("Удаление студента\n");

file->delStudent();

backMenu();

break;

}

case 4: {

file->editStudent();

backMenu();

break;

}

case 5: {

system("cls");

print("Какую информацию вы хотите получить:\n\n");

print("Чтобы получить краткую информацию (без данных о сессии), нажмите [1]\n");

print("Чтобы получить всю информацию о студентах, нажмите [2]\n");

print("Чтобы вернуться в меню, нажмите [3]\n\n");

print("-----> ");

cin >> \*ans;

switch (\*ans) {

case 1: {

file->printChoice(2);

backMenu();

break;

}

case 2: {

file->printChoice(4);

backMenu();

break;

}

case 3: {

break;

}

}

break;

}

case 6:

return false;

}

return true;

delete file;

}

private:

File\* file = nullptr;

int\* ans = nullptr;

void studentsPrint(int rez) {

return;

}

};

int main()

{

RUS;

Menu\* menu = new Menu();

while (menu->hub());

delete menu;

return 0;

}