

<https://github.com/nioan10>

## СОДЕРЖАНИЕ

СОДЕРЖАНИЕ.....	2
АНОТАЦИЯ.....	3
ВВЕДЕНИЕ.....	4
1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ .....	6
1.1 Структурное программирование .....	6
1.2 Процедурное программирование .....	6
1.4 Ссылки .....	7
1.5 Работа с памятью. Указатели.....	7
1.6 Работа с файлами .....	8
1.7 Динамические структуры данных .....	9
1.8 Линейный список .....	9
1.9 Функции.....	10
1.10 Объектно-ориентированное программирование .....	10
1.11 Конструкторы и деструкторы .....	11
1.12 Классы и дружественные функции .....	12
1.12.1 Описание классов и их методов.....	12
1.13 Методы шифрования и дешифрования .....	13
2 ПРАКТИЧЕСКАЯ ЧАСТЬ.....	15
2.1 Структура программы .....	15
2.2 Алгоритм решения задачи.....	21
2.3 Программная реализация задания.....	23
2.3.1 Пример выполнения задания .....	23
2.3.2 Отображение списка студентов .....	25
2.3.3 Изменение данных о студенте .....	26
2.3.3 Добавление информации о новом студенте .....	28
2.3.4 Удаление данных о студенте.....	30
2.3.5 Шифрование данных .....	32
2.4 Руководство пользователя .....	34
2.8 Системные требования .....	37
ЗАКЛЮЧЕНИЕ .....	38
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ.....	39
ПРИЛОЖЕНИЕ А Исходный код программы .....	40

## АНОТАЦИЯ

Тема курсовой работы: Разработка программы информационного поиска студентов по заданным критериям с возможностью шифрования данных

Выполнил:

Отчёт по Курсовой работе состоит из нескольких частей: введения, двух глав, заключения и списка используемых источников.

Во введении рассмотрены цели, задачи работы и ее актуальность.

Первая глава относится к теоретической части. В ней разбираются основные термины, определения, которые используются в программировании языка C++

Во второй главе, относящейся к практической части, приведены листинги кода, представлена программа, выполняющая поставленную задачу, а приведены блок схемы и разобраны алгоритмы ее работы.

В заключение определены основные выводы, к которым пришли в ходе выполнения Курсовой Работы.

Ключевые слова: C++, объектно-ориентированное программирование, шифрование и дешифрование, шифрование данных, проектирование базы информационного поиска студентов, поиск по заданным критериям.

## ВВЕДЕНИЕ

Задачей курсовой работы является: для текстового файла, отсортировать всю группу студентов по увеличению года поступления в ВУЗ, с поиском среди лиц определённого пола.

Курсовая работа по дисциплине "Языки программирования" предназначена для углубления и закрепления знаний и навыков, полученных в ходе изучения данной дисциплины. Она представляет собой выполнение сложного задания по разработке, проектированию и тестированию программного обеспечения, а также оформлению соответствующей документации.

В результате изучения дисциплины обучающийся должен:

Знать:

- 1) языки, системы и инструментальные средства программирования в профессиональной деятельности;
- 2) математический аппарат, математические пакеты, программные комплексы;
- 3) общие принципы построения и использования современных языков программирования высокого уровня.

Уметь:

- 1) использовать языки, системы и инструментальные средства программирования в профессиональной деятельности;
- 2) строить алгоритм решения задачи, проводить его анализ и реализовывать в современных программных комплексах;
- 3) работать с интегрированной средой разработки программного обеспечения.

Владеть:

- 1) языками программирования, системами и инструментальными средствами программирования в профессиональной деятельности;
- 2) навыками разработки, документирования, тестирования и отладки программ на языке программирования высокого уровня;

- 3) основными методами разработки алгоритмов и программ;
- 4) методами создания структур данных, используемые для представления типовых информационных объектов.

Задачи курсовой работы:

- 1) выполнить анализ предоставленных данных, в соответствии с заданием;
- 2) сделать вывод и определить необходимые данные и структуры, нужные для реализации программы;
- 3) создать рабочий алгоритм для решения конкретных задач индивидуального варианта и целостной работы программы;
- 4) разработать элементы, указанные в пункте два;
- 5) создать контрольные данные, необходимые для проверки работоспособности программы;
- 6) скорректировать созданное программное обеспечение на основе контрольных данных, подготовленных в предыдущем пункте.

# **1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

## **1.1 Структурное программирование**

Структурное программирование представляет собой подход, основанный на системном подходе при создании и использовании программного обеспечения для ЭВМ. Он базируется на нескольких простых принципах: алгоритм и программа должны быть составлены поэтапно, сложная задача должна быть разбита на более простые части, каждая из которых имеет один вход и один выход, и логика алгоритма и программы должна основываться на минимальном числе достаточно простых базовых управляющих структур.

Принципы структурного программирования включают использование трех базовых управляющих конструкций (последовательность, ветвление, цикл), вложенных друг в друга произвольным образом, а также оформление повторяющихся фрагментов программы в виде подпрограмм (процедур и функций). Конструкции должны иметь один вход и один выход, и разработка программы должна вестись пошагово методом "сверху вниз".

## **1.2 Процедурное программирование**

Процедурное программирование представляет собой метод программирования на императивном языке, который позволяет объединять последовательно выполняемые операторы в более крупные единицы кода, называемые подпрограммами. Они создаются с помощью механизмов языка программирования. Процедурное программирование соответствует архитектуре традиционных ЭВМ.

Основными особенностями процедурного программирования являются:

- использование предопределенных функций, которые обычно встроены в язык программирования более высокого уровня;
- использование локальных переменных, которые ограничены областью действия и могут быть использованы только в том методе, где они определены;

- использование глобальных переменных, которые могут быть использованы в любой функции;
- модульность, которая позволяет группировать системы и выполнять задачи поочередно;
- передача параметров в функции, подпрограммы или процедуры.

## 1.4 Ссылки

Ссылка — это псевдоним для другой переменной. Они объявляются при помощи символа `&`. При инициализации ссылка, должна быть проинициализирована, ровно один раз.

Ссылка при определении сразу же инициализируется. Инициализация ссылки производится следующим образом:

```
int i = 0;
```

```
int& iref = i;
```

Физически `iref` представляет собой постоянный указатель на `int` - переменную типа `int* const`.

При передаче объекта в функцию по ссылке, функция получает псевдоним объекта, что позволяет изменять его внутри функции. Однако ссылки не могут ссылаться на другие ссылки или на поле битов, и массивы ссылок или указателей на ссылки не могут быть созданы. Ссылки могут быть использованы для возврата результата из функции, при этом возвращается не указатель на объект или его значение, а сам объект.

## 1.5 Работа с памятью. Указатели

Указатели - это одно из ключевых понятий языка программирования Си, которые позволяют хранить адреса участков памяти, где хранятся данные. При объявлении указателя необходимо указать тип переменной, на которую он будет указывать, и использовать знак `*` перед именем переменной. Чтобы получить адрес переменной, необходимо использовать оператор `&` перед ее именем. В свою очередь, знак `*` перед указателем в программе обозначает значение ячейки памяти, на которую указывает указатель. Важно помнить, что

запись по неправильному адресу может привести к сбою программы. Для обозначения недействительного указателя используется константа NULL. При изменении значения указателя на n он сдвигается к n-ому следующему числу данного типа. Для вывода указателя используется формат %p.

## 1.6 Работа с файлами

Файл – это именованная область ячеек памяти, в которой хранятся данные одного типа. Файл имеет следующие характерные особенности: уникальное имя; однотипность данных; произвольная длина, которая ограничивается только емкостью диска. Для работы с файлом в языке C++ необходима ссылка на файл. Для определения такой ссылки существует структура FILE, описанная в заголовочном файле stdio.h. Данная структура содержит все необходимые поля для управления файлами, например, текущий указатель буфера, текущий счетчик байтов, базовый адрес буфера ввода-вывода, номер файла.

Функция открытия файла.

При открытии файла (потока) в программу возвращается указатель на поток (файловый указатель), являющийся указателем на объект структурного типа FILE. Этот указатель идентифицирует поток во всех последующих операциях.

Функция закрытия файла.

Открытые на диске файлы после окончания работы с ними рекомендуется закрыть явно. Это является хорошим тоном в программировании.

Функция переименования файла.

Функция переименовывает файл; первый параметр – старое имя файла, второй – новое. Возвращает 0 при неудачном выполнении.

Функция контроля конца файла

Для контроля достижения конца файла есть функция feof. `int feof(FILE * filename);`



## **1.7 Динамические структуры данных**

Динамические структуры данных - это способ хранения и организации информации в памяти компьютера, который позволяет создавать и изменять структуры в процессе выполнения программы. Они предоставляют возможность эффективно управлять большим объемом данных, а также удобны для решения различных задач, связанных с обработкой и анализом информации.

Одним из примеров динамических структур данных является динамический массив. В отличие от статического массива, динамический массив может изменять свой размер во время выполнения программы, что делает его более гибким и удобным в использовании.

Еще одним примером динамической структуры данных является связный список. Он состоит из узлов, каждый из которых содержит информацию и указатель на следующий узел. Благодаря этому связный список может легко изменяться в процессе выполнения программы, добавлять или удалять узлы в любой момент времени.

## **1.8 Линейный список**

Линейно связанный список - это одна из разновидностей динамических структур данных, которая представляет собой набор узлов, связанных между собой указателями. Каждый узел содержит данные и указатель на следующий узел. Таким образом, линейно связанный список позволяет эффективно хранить и обрабатывать последовательность данных.

Одной из основных преимуществ линейно связанного списка является его гибкость. Он может изменяться в процессе выполнения программы, что позволяет легко добавлять или удалять элементы из списка. Это делает его особенно удобным для работы со списками неизвестной заранее длины или для решения задач, где требуется динамическое изменение списка.

Кроме того, линейно связанный список может быть использован для реализации других структур данных, таких как стек или очередь. Например,

чтобы реализовать стек, необходимо добавлять новые элементы в начало списка и удалять их оттуда же.

## **1.9 Функции**

Функции в программировании позволяют определить блок инструкций и дать ему имя, которое можно использовать для вызова из разных частей программы. Аргументы, передаваемые в функцию, должны соответствовать типам параметров, указанным в определении функции. Хотя длина функции не ограничена, наилучшей практикой является использование функций, каждая из которых решает только одну задачу. Если возможно, сложные алгоритмы следует разбивать на более простые функции для упрощения кода. Функции могут быть перегружены, если они имеют разное число или тип параметров, но используют одно и то же имя.

Определение функции содержит объявление и тело функции, которые заключаются в фигурные скобки и содержат объявления переменных, операторы и выражения. Пример полного определения функции представлен ниже.

## **1.10 Объектно-ориентированное программирование**

Объектно-ориентированное программирование – это подход, при котором вся программа рассматривается как набор взаимодействующих друг с другом объектов. При этом нам важно знать их характеристики.

Объектно-ориентированное программирование (ООП) - это методология программирования, которая позволяет описывать программные системы в виде набора объектов, которые взаимодействуют друг с другом, передавая сообщения и вызывая методы. Каждый объект имеет свойства и методы, которые определяют его поведение и состояние. ООП позволяет создавать модульный, гибкий и масштабируемый код, который легко поддерживать и расширять.

Основными принципами ООП являются наследование, инкапсуляция и полиморфизм. Наследование позволяет создавать новые классы на основе уже

существующих, перенимая их свойства и методы. Инкапсуляция означает, что данные и методы, которые работают с этими данными, должны быть объединены в единый объект, чтобы обеспечить безопасность и избежать конфликтов. Полиморфизм позволяет использовать один и тот же метод для разных типов данных, что делает код более гибким и удобным для использования.

ООП применяется в различных областях, включая разработку веб-приложений, мобильных приложений, игр, научных и инженерных приложений и многих других. ООП позволяет создавать сложные системы с высокой степенью абстракции и модульности, что делает его одним из наиболее популярных подходов в программировании.

### **1.11 Конструкторы и деструкторы**

Конструктор – это функция-член, имя которой совпадает с именем класса, инициализирующая переменные-члены, распределяющая память для их хранения.

Деструктор – это функция-член, имя которой представляет собой имя класса, предназначенная для уничтожения переменных.

При создании объектов одной из наиболее широко используемых операций которую вы будете выполнять в ваших программах, является инициализация элементов данных объекта. Чтобы упростить процесс инициализации элементов данных класса, C++ использует специальную функцию, называемую конструктором, которая запускается для каждого создаваемого вами объекта. Также C++ обеспечивает функцию, называемую деструктором, которая запускается при уничтожении объекта.

Конструктор представляет собой метод класса, который облегчает вашим программам инициализацию полей при создании объекта класса. Конструктор имеет такое же имя, как и сам класс. Конструктор не имеет возвращаемого значения. Конструкторы относят к интерфейсу класса, чтобы с их помощью можно было создавать объекты данного класса из внешней части программы.

Таким образом, деструктор не может быть перегружен и должен существовать в классе в единственном экземпляре. Деструктор вызывается автоматически при уничтожении объекта.

## **1.12 Классы и дружественные функции**

Классы в C++ представляют собой тип данных, который содержит поля (переменные) и методы (функции), связанные с этим типом данных. Классы позволяют создавать объекты, которые могут хранить данные и выполнять операции над этими данными. Однако иногда возникает необходимость предоставить доступ к закрытым членам класса извне. В этом случае может пригодиться дружественная функция.

Дружественная функция - это функция, которая имеет доступ к закрытым членам класса, но не является его членом. Дружественные функции объявляются внутри класса, но определяются за его пределами. Они могут быть полезны, например, для обработки данных, которые не могут быть обработаны членами класса, или для доступа к закрытым членам класса из других функций.

Для того, чтобы объявить функцию дружественной в C++, необходимо использовать ключевое слово "friend" перед ее объявлением внутри класса. После этого функция будет иметь доступ к закрытым членам класса.

### **1.12.1 Описание классов и их методов**

Классы в языке программирования C++ — это абстракция, которая описывает методы и свойства, ещё не существующих объектов. Объекты — конкретное представление абстракции, которые имеют свои свойства и методы. Свойства — это любые данные, которыми можно характеризовать объект класса. Методы — это функции, выполняющие различные действия над данными (свойствами) класса. Поле класса в объектно-ориентированном программировании — переменная, описание которой создает программист при создании класса. Все данные объекта хранятся в его полях.

Для разработки данного проекта понадобятся следующие классы. В таблице ниже представлены и описаны поля и методы класса, используемые в проекте.

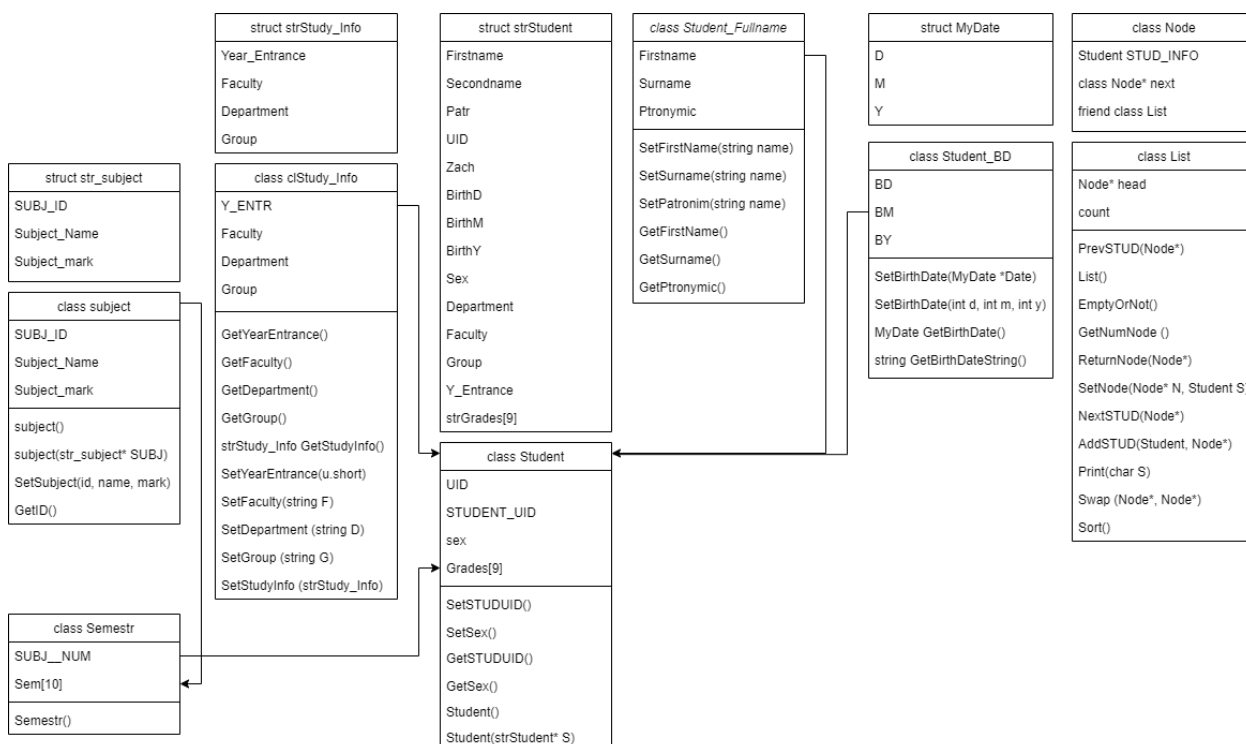


Рисунок 1.1 – Диаграмма классов и структур, используемых в курсовой работе, а также наследование классов друг от друга

### 1.13 Методы шифрования и дешифрования

Шифрование данных - это процесс преобразования информации из исходной формы в зашифрованную форму, которая недоступна для чтения без специального ключа или пароля. Целью шифрования является защита данных от несанкционированного доступа, так как только авторизованные пользователи, имеющие ключ или пароль, могут расшифровать данные и прочесть их содержимое.

Существует множество алгоритмов шифрования данных, которые различаются по уровню безопасности и сложности. Наиболее распространенные алгоритмы шифрования включают в себя AES, RSA, DES, Blowfish и многие другие. Каждый алгоритм использует свой уникальный ключ или пароль для зашифровки данных.

Дешифровка данных - это процесс преобразования зашифрованных данных в исходный текст с помощью ключа или пароля. Целью дешифровки является получение доступа к конфиденциальной информации, защищенной шифрованием.

Для дешифровки данных необходимо использовать тот же алгоритм и ключ, что и для шифрования. Если ключ или пароль не верны, то дешифрование невозможно. Для успешной дешифровки данных также необходимо иметь доступ к программе или инструменту, который поддерживает используемый алгоритм шифрования. Дешифрование – процесс, обратный процессу шифрованию.

Шифрование данных с помощью OpenSSL — это один из самых распространенных и надежных способов защиты конфиденциальной информации. OpenSSL - это библиотека криптографических функций, которая предоставляет набор инструментов для шифрования данных.

Для шифрования данных с помощью OpenSSL необходимо использовать ключ и алгоритм шифрования. OpenSSL поддерживает множество алгоритмов шифрования, включая AES, DES, Blowfish, RC4 и другие. Ключ используется для зашифровки данных и расшифровки их при необходимости.

## 2 ПРАКТИЧЕСКАЯ ЧАСТЬ

### 2.1 Структура программы

Исходными данными для программы является информация о группе студентов из N человек, где запись о студенте содержит следующие данные (таблица 2.1.1).

Таблица 2.1.1 – Структура и типы данных

Основная структура данных	Типы данных
Ф.И.О. студента	Student_Fullname (класс имени)
Имя	Символьный массив из 30 элементов
Фамилия	string(строковый)
Отчество	string(строковый)
Пол студента	string(строковый)
Число, месяц, год рождения	Student_BD(дата)
Информация о ВУЗе	clStudy_Info(класс информации)
Год поступления в институт	Unsigned short (целочисленный)
Институт	Символьный массив из 30 элементов
Кафедра	string(строковый)
Группа	string(строковый)
Номер зачетной книжки	string(строковый)
Название предмета	string(строковый)
Оценка	Unsigned short(целочисленный)
Информация о сессиях	Semestr[9][10] (массив)

В Курсовой работе допустима информация по 9 сессиям с 10 предметами максимум, которые вводятся в массив данных. Все данные подвластны изменениям. Количество сессий зависит от года поступления.

Для выполнения задачи курсовой работы, разработаны соответствующие классы. В данных классах предусмотрены методы, с помощью которых решаются задач, поставленных в условии курсовой работы.

В представленной ниже таблице продемонстрированы роли используемых классов (таблица 2.1.2).

Таблица 2.1.2 – Классы

Класс	Назначение
Student_Fullname	Класс, который хранит информацию о полном имени студента (Фамилия, Имя, Отчество)

Продолжение таблицы 2.1.2 – Классы.

subject	Класс содержит информацию о конкретном предмете, т.е о названии предмета и оценке за него.
Semestr	Класс, который содержит данные об одной сессии студента, т.е об одном семестре в виде 10 предметов, по которым сдаются экзамены.
Student_BD	Содержит информацию о дне рождения студент, который обучается в ВУЗе
Node	Класс, является элементом списка, который используется в индивидуальном задании
List	Класс, который хранит информацию о списке студентов и является односвязным списком, используемым в индивидуальном задании
clStudy_Info	Содержит информацию о Студенте, а именно информацию о кафедре, институте, группе обучения, а также году поступления в ВУЗ
Student	Содержит полную информацию о конкретном студенте, включая информацию, которая хранится в классах родителей

Для разработки данного проекта понадобятся следующие классы. В таблице ниже представлены и описаны поля и методы класса, используемые в проекте (таблица 2.1.3).

Таблица 2.1.3 – Поля классов

Классы	Поле класса	
	Тип данных	Название и характеристика
Student_Fullname	string	Firstname – Имя Surname – Фамилия Ptronumic - Отчество
Student_BD	Unsigned short	BD – День рождения BM – Месяц рождения BY – Год рождения
subject	Unsigned short	SUBJ_ID – Уникальный ID предмета
	string	Subject_Name – Наименование предмета
	Unsigned short	Subject_mark – Оценка за предмет
Semestr	Unsigned short	SUBJ_NUM – количество предметов с семестре, которые уходят на экзамен
	Subject*[]	Sem[10] – массив состоящий из элементов класса subject, в количестве 10 штук на сессию, который содержит наименование предмета и оценку за него



Продолжение таблицы 2.1.3

Student	int	UID – уникальный целочисленный номер студента
	string	STUDENT_UID – номер зачётной книжки определённого формата sex – пол студента
	Semestr	Grades[9] – хранение результатов Сессий студента
clStudy_Info	string	Faculty – наименование выпускающей кафедры Department – наименование института обучения Group – наименование группы обучения
	Unsigned short	Year_Entrance – год поступления в ВУЗ
Node	Student	STUD_INFO – информация о студенте в конкретном узле линейно связанного списка
	Class Node*	Next – адрес на следующий узел линейно связанного списка
List	Node*	Head – содержит информацию об адресе на головной элемент списка
	int	Count – содержит информацию о количестве узлов в списке, в рамках конкретной задачи, о количестве студентов.

Далее методы классов, представленные в таблице 2.1.4.

Таблица 2.1.4 Методы классов

Класс	Методы класса			
	Название	Назначение	Аргументы (их тип)	Тип возвращаемого значения
Student_BD	SetBirthDate	Установка даты рождения с помощью, заранее созданной, структуры	MyDate*	-
	SetBirthDate	Установка даты рождения студента с помощью целочисленных значений	Int d, m, y	-

Продолжение таблицы 2.1.4 Методы класса

	GetBirthDate	Возврат значения даты рождения студентов в структуру	-	MyDate (структура)
	GetBirthDateString	Возврат значения даты рождения студента в виде строки	-	String (строковый)
subject	Subject	Конструктор по умолчанию, который создаёт предмет-заглушку	-	-
	Subject	Конструктор для предмета, который создаёт предмет на основе структуры	Str_subject	-
	SetSubject	Задаёт новый предмет на основе трёх переменных	Int String Unsigned short	-
	GetID	Возвращает ID предмета, который используется в базе данных предметов	-	Unsigned short
Student_Fullname	SetFirstName	Устанавливает имя студента из строки	String	-
	SetSurname	Задаёт фамилию студента из введённой строки	String	-
	SetPatronim	Задаёт отчество студента из введённой строки	string	-
	GetFirstName	Возвращает строковое значение имени студента	-	String

Продолжение таблицы 2.1.4 Методы классов

	GetSurname	Возвращает строковое значение фамилии студента	-	string
	GetPronymic	Возвращает строковое значение отчества студента	-	string
Student	SetSTUDUID	Устанавливает значение зачётной книжки из строки	String	-
	SetSex	Устанавливает значение пола из строки	string	-
	GetSTUDUID	Возвращает значение зачётной книжки студента	-	String
	GetSex	Возвращает строковое значение пола студента	-	string
	Student	Конструктор, создающий студента заглушку	-	-
	Student	Конструктор, создающий студента, на основе структуры	StrStudent	-
Semestr	Semestr()	Конструктор, создающий массив из 10 классов subject для хранения данных по сессии	-	-
Node	-	-	-	-
List	PrevSTUD	Возвращает адрес предыдущего узла списка	Node*	Node*

Продолжение таблицы 2.1.4. Методы классов

	List	Конструктор создания массива списка	-	-
	EmptyOrNot	Метод, который определяет пустой ли список или нет и возвращает значения 0 или 1	-	bool
	GetNumNode	Возвращает значение равное количеству узлов в списке	-	int
	ReturnNode	Возвращает значение, находящееся по адресу узла, и выводит их в класс	Node*	Student
	SetNode	Устанавливает значение, полученное из класса в узел	Node* Student	-
	NextSTUD	Возвращает значение следующего узла списка	Node*	Node*
	AddSTUD	Добавляет в список нового студента, на основе информации класса	Student Node* (пустой узел, для прогона по списку)	-
	Print	Выводит укороченную информацию со списком студентов, на основе полученного значения пола	string	-
	Swap	Меняет местами две ячейки списка, на основе полученных адресов	Node*, Node*	-

Продолжение таблицы 2.1.4 Методы классов

	Sort	Выполняет сортировку списка студентов на основе индивидуального задания	-	-
	GetFirst()	Возвращает адрес первого элемента списка	-	Node*

## 2.2 Алгоритм решения задачи

В индивидуальном задании Курсовой Работы предлагается отсортировать группу студентов по увеличению года поступления в ВУЗ, с поиском среди лиц определённого пола.

Алгоритм сортировки выполняет работу по следующему принципу. Получается адрес первого элемента списка, создаётся дополнительная переменная флага. Программа входит в цикл, условие опущенного флага, то есть, когда флаг будет поднят на момент завершения цикла, программа покинет его. Флаг поднимается и алгоритм начинает новый цикл, с условием завершения при достижении конца списка. Создаются две динамические ячейки с текущим студентом списка и следующим за ним. Если год поступления в ВУЗ у первого студента больше, чем у второго студента, то происходит замена двух студентов местами в списке, а после флаг опускается. Программа заканчивает проход по списку, и, если обнаруживаются иные пары, удовлетворяющие условию, происходит замена студентов. Алгоритм повторяется до того момента, пока за весь проход внешнего цикла флаг ни разу не будет опущен.

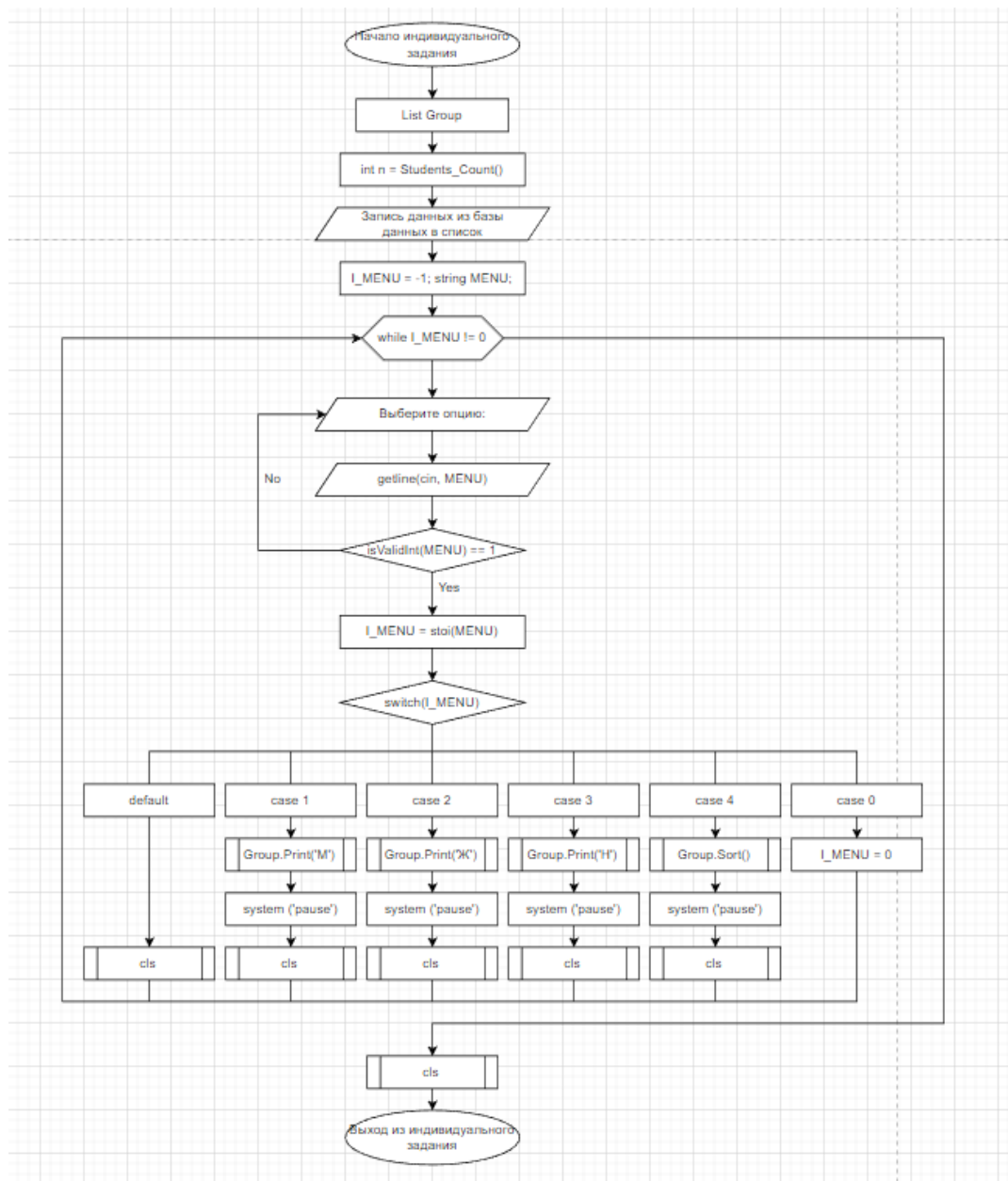


Рисунок 2.1— схема алгоритма работы меню индивидуального задания

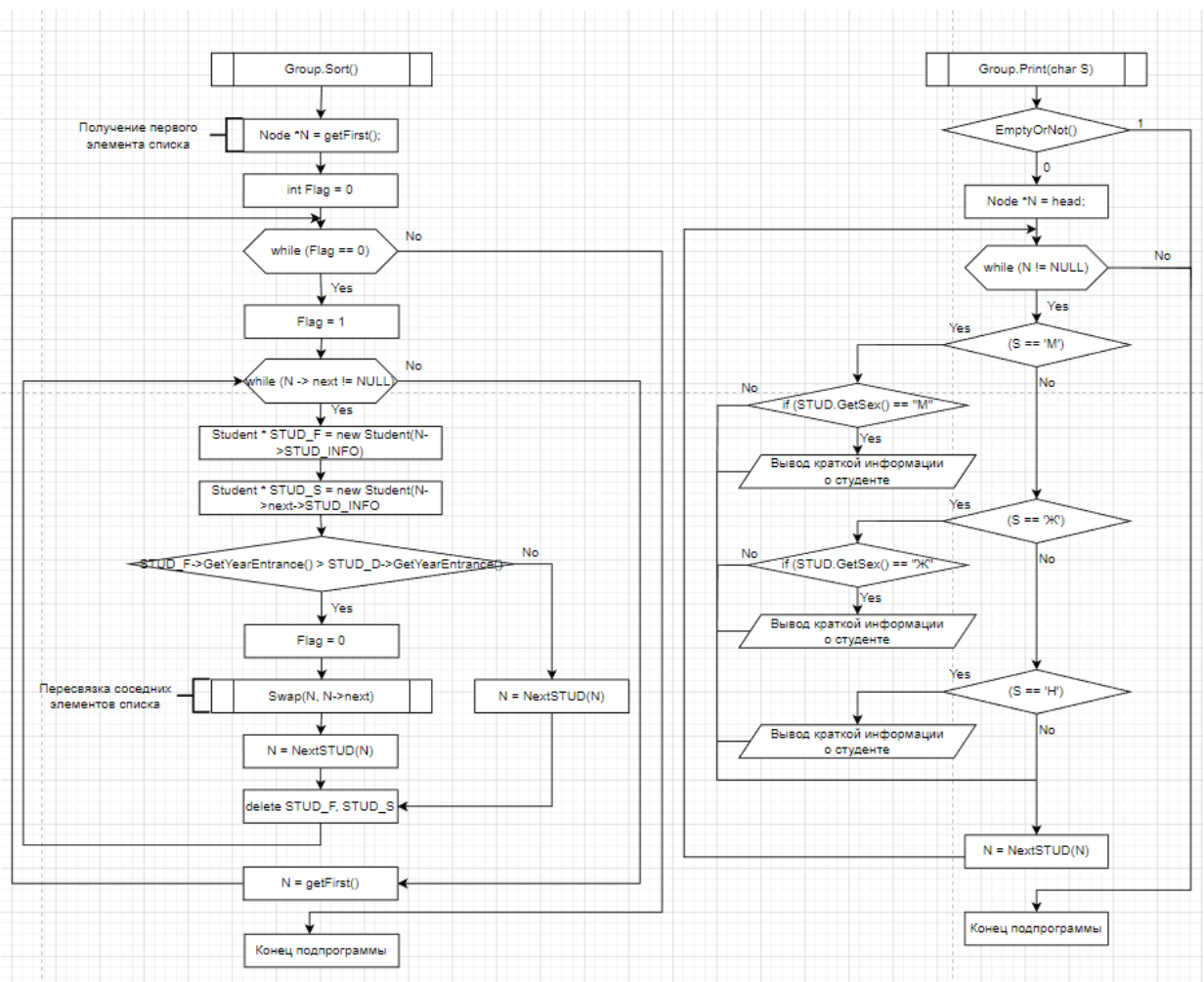


Рисунок 2.2 – схема алгоритма работы дополнительных подпрограмм в индивидуальном задании

## 2.3 Программная реализация задания

### 2.3.1 Пример выполнения задания

В рамках индивидуального задания пользователь может осуществить сортировку студентов, находящихся в базе данных, на основе года поступления в ВУЗ, с учётом выбора определённого пола. При переходе в индивидуальное задание пользователю предоставлена возможность выбора ряда функций.

Создан линейный список из предоставленных студентов. Выберите одну из следующих опций

1. Отобразить список из студентов Мужского пола
2. Отобразить список студентов Женского пола
3. Отобразить полный список студентов
4. Отсортировать список студентов по году поступления в ВУЗ
0. Покинуть раздел индивидуального задания

Введите операцию:

*Рисунок 3.1.1 – меню индивидуального задания*

При выборе пунктов 1-3 на экран меню будет выводиться краткая информация о студентах ВУЗа. При выборе пункта 4, произойдёт сортировка студентов. Сортировка выполняется на основе алгоритма, приведённого выше, как работает отображение элементов, также приведено в алгоритме выполнения задания индивидуального варианта.

0. Покинуть раздел индивидуального задания  
Введите операцию: 1  
0 Фамилия студента: Иванов  
Имя студента: Николай  
Отчество студента: Игоревич  
Год поступления студента в ВУЗ: 2021. Пол студента: М  
-----  
0 Фамилия студента: Керн  
Имя студента: Владислав  
Отчество студента: Афансиевич  
Год поступления студента в ВУЗ: 2022. Пол студента: М  
-----  
0 Фамилия студента: Соловьев  
Имя студента: Артем  
Отчество студента: Петрович  
Год поступления студента в ВУЗ: 2022. Пол студента: М  
-----  
0 Фамилия студента: Петров  
Имя студента: Дмитрий  
Отчество студента: Сергеевич  
Год поступления студента в ВУЗ: 2022. Пол студента: М  
-----  
0 Фамилия студента: Заглушкин  
Имя студента: Вячеслав  
Отчество студента: Сергеевич  
Год поступления студента в ВУЗ: 2022. Пол студента: М  
-----  
Для продолжения нажмите любую клавишу . . .

*Рисунок 3.1.2 – отображение студентов мужского пола, после сортировки студентов по году поступления в ВУЗ*



### 2.3.2 Отображение списка студентов

Список студентов ВУЗа		
UID студента	Группа студента	ФИ Студента
0	ББББ-10-22	Вячеслав.Заглушкин
1	ББББ-10-22	Александра.Иванова
2	БИБИ-01-22	Дмитрий.Петров
3	ББББ-11-22	Елена.Смирнова
4	БИСК-01-21	Владилена.Меризе
5	ББББ-11-22	Екатерина.Белова
6	ББББ-10-22	Артем.Соловьев
7	ББББ-11-22	Анастасия.Михайлова
8	БИСК-01-21	Николай.Иванов
9	ББББ-11-22	Анастасия.Петрова
10	ББББ-10-22	Владимир.Смирнов
Выберите операцию взаимодействия:		
1 - Вывод информации о студенте по UID		
2 - Добавление нового студента в Базу Данных		
3 - Редактирование информации о студенте по UID		
4 - Удаление информации о студенте по UID		
0 - Возврат в предыдущее меню		
Введите операцию взаимодействия:		

Рисунок 3.2 – полный список студентов со списком взаимодействия

В данном меню представлена реализация одного из пунктов Курсовой работы, а именно отображение полного списка студентов, которые занесены в базу данных студентов. Каждый студент имеет свой уникальный идентификатор (UID), который выдаётся студенту при занесении в базу данных. Во втором поле каждого студента хранится информация о группе, в которой обучается студент. В третье и последнем поле содержится информация о самом студенте, то есть Имя и Фамилия студента.

В нижней зоне содержится информация, которой пользователь может пользоваться для взаимодействия с базой данных студента. В список включены все основные функции, которые используются в программе и необходимы в реализации Курсовой работы, а именно вывод информации о студенте, редактирование информации о студенте, добавление нового студента, удаление существующего студента, а также возврат в предыдущее меню. Определение операции происходит путём ввода целочисленного значения, соответствующего номеру команды, в нижнее поле «Введите

операцию взаимодействия:». Если выбрать пункт (1), и в следующем поле ввести UID, происходит очистка экрана. Выводится информация об Студенте.

Информация о студенте ВУЗа		
UID студента	Зачётная книжка	
3	01Г2468	
Фамилия студента	Имя студента	Отчество студента
Смирнова	Елена	Александровна
Институт	Кафедра	Группа
ИУИТ	КБ-4	ББББ-11-22
Пол студента	Год поступления	
Ж	2022	
Сведения по оценкам за сессии (по семестрам)		
1 семестр		
Наименование дисциплины	Оценка	
Английский_язык	3	
Математика	5	
Программирование	5	
Физика	5	
Экономика	5	
Культурология	5	
Теория_Систем	5	
Операционные_Системы	5	
Химия	5	
Логика_Жизни	5	
Для продолжения нажмите любую клавишу . . .		

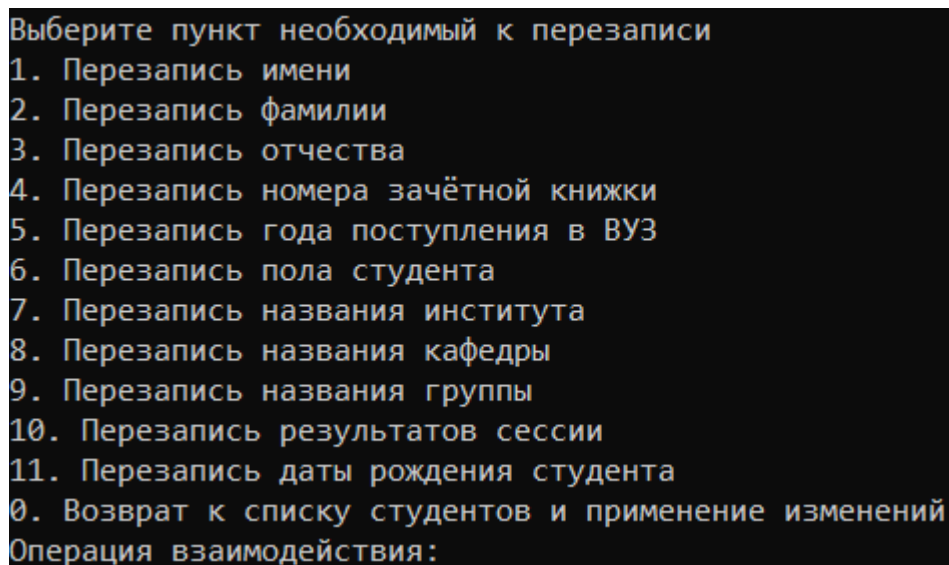
Рисунок 3.3 – информация об одном студенте

Данный экран выводит информацию об одном студенте вуза. В данном окне отображаются все основные поля, которые указаны в вводных данных в курсовой работе. Сверху выводится информацию по UID, номеру зачётной книжки, ФИО студента, институту, выпускающей кафедры и группы обучения, пол и год поступления в ВУЗ. Ниже приводится информация по сессиям за каждый семестр.

### 2.3.3 Изменение данных о студенте

Если в меню со списком студентов выбрать третий пункт (3), произойдёт переход к изменению данных о студенте. Пере пользователем выводится

меню, в котором можно выбрать, какой параметр будет изменяться. Допустимо изменение нескольких полей, все изменения будут сохранены и заменены позже при выходе из окна редактирования.



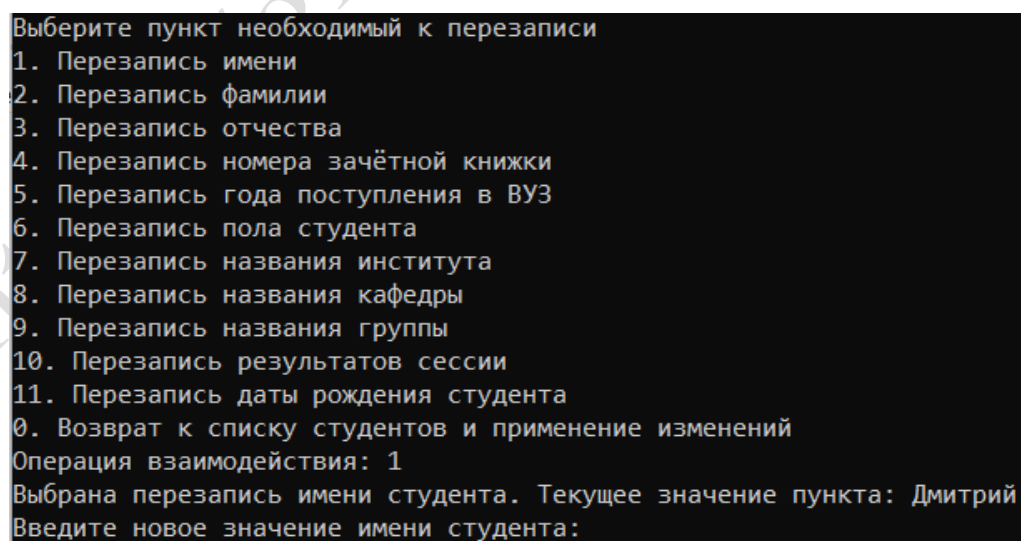
Выберите пункт необходимый к перезаписи

1. Перезапись имени
2. Перезапись фамилии
3. Перезапись отчества
4. Перезапись номера зачётной книжки
5. Перезапись года поступления в ВУЗ
6. Перезапись пола студента
7. Перезапись названия института
8. Перезапись названия кафедры
9. Перезапись названия группы
10. Перезапись результатов сессии
11. Перезапись даты рождения студента
0. Возврат к списку студентов и применение изменений

Операция взаимодействия:

*Рисунок 3.4 – окно редактирование полей студента*

Выбор поля для редактирования осуществляется на основе вводимого с клавиатуры целочисленного значения около строки «Операция взаимодействия». При выборе любого из значений, кроме «Перезапись результатов сессии» (пункт 10). На экран выводится информация о текущем значении данного поля, а также предоставляется возможность к редактированию.



Выберите пункт необходимый к перезаписи

1. Перезапись имени
2. Перезапись фамилии
3. Перезапись отчества
4. Перезапись номера зачётной книжки
5. Перезапись года поступления в ВУЗ
6. Перезапись пола студента
7. Перезапись названия института
8. Перезапись названия кафедры
9. Перезапись названия группы
10. Перезапись результатов сессии
11. Перезапись даты рождения студента
0. Возврат к списку студентов и применение изменений

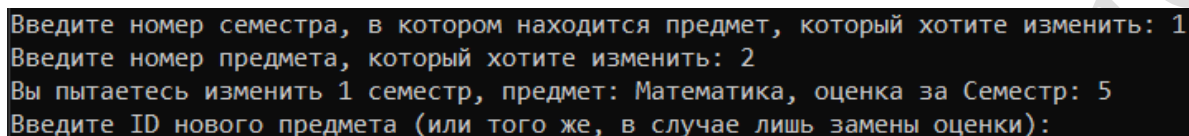
Операция взаимодействия: 1

Выбрана перезапись имени студента. Текущее значение пункта: Дмитрий

Введите новое значение имени студента:

*Рисунок 3.5 – окно редактирования имени студента*

При выборе поля «Перезапись результатов сессии» (пункт 10), пользователю предоставляется возможность выбрать семестр, который будет отредактирован, дальше предоставлена возможность определить номер предмета, который будет скорректирован. После определения поля редактирования, пользователю предоставлена возможность заменить предмет, путём ввода UID предмета из отдельной базы данных, а также простановка новой оценки для предмета.



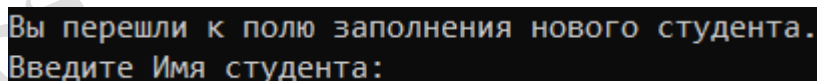
Введите номер семестра, в котором находится предмет, который хотите изменить: 1  
Введите номер предмета, который хотите изменить: 2  
Вы пытаетесь изменить 1 семестр, предмет: Математика, оценка за Семестр: 5  
Введите ID нового предмета (или того же, в случае лишь замены оценки):

*Рисунок 3.6 – окно редактирования предмета в семестре*

После изменения нужных данных, в том числе, изменение данных о предмете и оценках в случае, если подобное необходимо, пользователю предоставлена возможность покинуть поле редактирования. После выхода все данные будут перезаписаны в базе данных.

### **2.3.3 Добавление информации о новом студенте**

Один из пунктов взаимодействия с базой данных студентов в основном меню, где располагается весь список студентов, предоставляет возможность пользователю добавить нового студента в базу данных. При выборе пункта два (2) в меню взаимодействий, пользователю открывается окно, в котором он может ввести все данные о новом студенте.



Вы перешли к полю заполнения нового студента.  
Введите Имя студента:

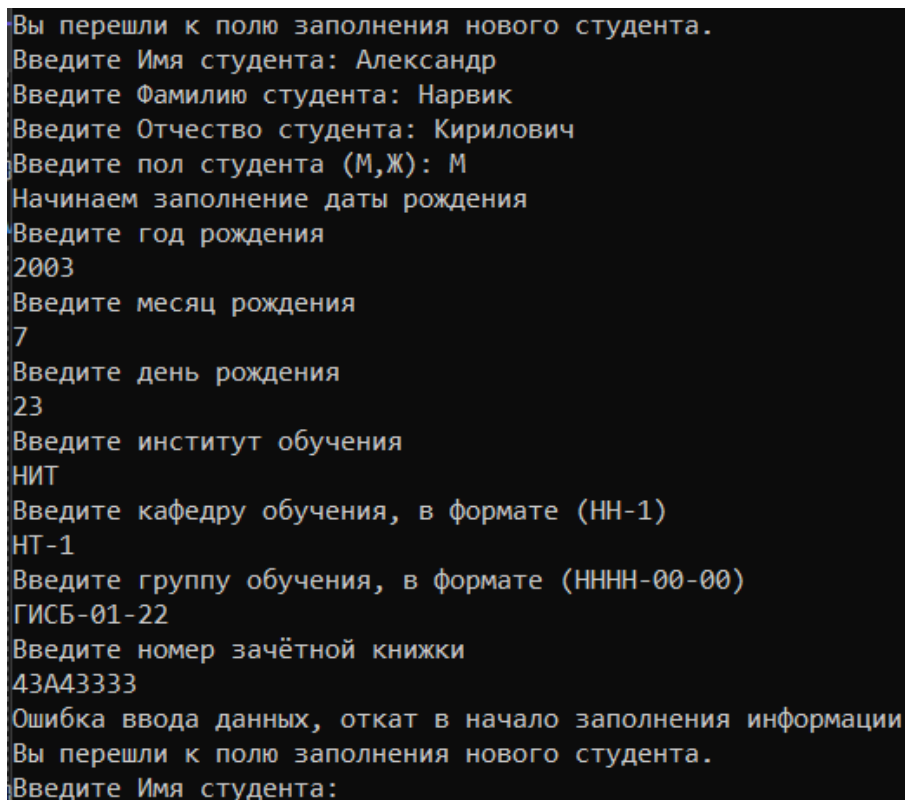
*Рисунок 3.6 – окно создание нового студента в базе данных*

После перехода к созданию нового студента, пользователю следует в поочерёдно появляющихся полях ввести все основные данные о студенте, так как Фамилия, Имя, Отчества, дата рождения, год поступления в ВУЗ, пол, наименование института, выпускающей кафедры и группы обучения, а также оценки за все сессии, которые уже были с момента поступления в ВУЗ, иными словами за 1, 3, 5 и так далее семестров, в зависимости от года поступления.

```
Вы перешли к полю заполнения нового студента.  
Введите Имя студента: Александр  
Введите Фамилию студента: Нарвик  
Введите Отчество студента: Кирилович  
Введите пол студента (М,Ж): М  
Начинаем заполнение даты рождения  
Введите год рождения  
2003  
Введите месяц рождения  
12  
Введите день рождения  
7  
Введите институт обучения  
НИТ  
Введите кафедру обучения, в формате (НН-1)  
НТ-1  
Введите группу обучения, в формате (НННН-00-00)  
КИКБ-01-22  
Введите номер зачётной книжки  
45A5677  
Введите год поступления в ВУЗ  
2022  
Идёт заполнение семестра 1  
Введите UID предмета  
23  
Введите оценку по данному предмету  
3  
Введите UID предмета  
24  
Введите оценку по данному предмету  
5  
Введите UID предмета  
54  
Введите оценку по данному предмету  
3  
Введите UID предмета  
55  
Введите оценку по данному предмету  
6
```

*Рисунок 3.7 – ввод данных при заполнении студента ВУЗа*

Большая часть данных проходит проверку на допустимость к вводу в базу данных и имеет строгий формат ввода. Такие поля, как номер зачётной книжки, название группы обучения, выпускающей кафедры, института обучения – имеют строгий формат ввода. Остальные поля проходят проверку на корректность ввода данных. В случае ошибочного ввода данных, выводится сообщение об ошибке и происходит откат в заполнении данных.



```
Вы перешли к полю заполнения нового студента.  
Введите Имя студента: Александр  
Введите Фамилию студента: Нарвик  
Введите Отчество студента: Кирилович  
Введите пол студента (М,Ж): М  
Начинаем заполнение даты рождения  
Введите год рождения  
2003  
Введите месяц рождения  
7  
Введите день рождения  
23  
Введите институт обучения  
НИТ  
Введите кафедру обучения, в формате (НН-1)  
НТ-1  
Введите группу обучения, в формате (НННН-00-00)  
ГИСБ-01-22  
Введите номер зачётной книжки  
43А43333  
Ошибка ввода данных, откат в начало заполнения информации  
Вы перешли к полю заполнения нового студента.  
Введите Имя студента:
```

*Рисунок 3.7 – ввод ошибочных данных при вводе студента ВУЗа*

Как только все поля заполнены, данные за сессии введены, происходит возврат в меню списка студентов, где уже будет находится добавленный студент. Далее, всю информацию о нём можно отсмотреть с помощью уже отмеченных функций взаимодействия: отобразить полную информацию, отредактировать часть данных о студенте, удалить студента.

#### **2.3.4 Удаление данных о студенте**

Пунктом 4 в меню взаимодействия в меню списка студентов, пользователю предоставлена возможность удаления ненужного пользователя из списка студентов. Удаление студента из базы данных реализовано, путём перезаписи всех остальных студентов в отдельный файл, кроме удаляемого студента, а после возврат в стандартный файл.

Список студентов ВУЗа		
UID студента	Группа студента	ФИ Студента
0	ББББ-10-22	Вячеслав.Заглушкин
1	ББББ-10-22	Александра.Иванова
2	БИБИ-01-22	Дмитрий.Петров
3	ББББ-11-22	Елена.Смирнова
4	БИСК-01-21	Владилена.Меризе
5	ББББ-11-22	Екатерина.Белова
6	ББББ-10-22	Артем.Соловьев
7	ББББ-11-22	Анастасия.Михайлова
8	БИСК-01-21	Николай.Иванов
9	ББББ-11-22	Анастасия.Петрова
10	ББББ-10-22	Владимир.Смирнов

Выберите операцию взаимодействия:

1 - Вывод информации о студенте по UID  
2 - Добавление нового студента в Базу Данных  
3 - Редактирование информации о студенте по UID  
4 - Удаление информации о студенте по UID  
0 - Возврат в предыдущее меню

Введите операцию взаимодействия: 4  
Выбран пункт удаления информации о студенте по UID: 10

Рисунок 3.8 – ввод информации по удалению студента из базы данных

Список студентов ВУЗа		
UID студента	Группа студента	ФИ Студента
0	ББББ-10-22	Вячеслав.Заглушкин
1	ББББ-10-22	Александра.Иванова
2	БИБИ-01-22	Дмитрий.Петров
3	ББББ-11-22	Елена.Смирнова
4	БИСК-01-21	Владилена.Меризе
5	ББББ-11-22	Екатерина.Белова
6	ББББ-10-22	Артем.Соловьев
7	ББББ-11-22	Анастасия.Михайлова
8	БИСК-01-21	Николай.Иванов
9	ББББ-11-22	Анастасия.Петрова

Выберите операцию взаимодействия:

1 - Вывод информации о студенте по UID  
2 - Добавление нового студента в Базу Данных  
3 - Редактирование информации о студенте по UID  
4 - Удаление информации о студенте по UID  
0 - Возврат в предыдущее меню

Введите операцию взаимодействия:

Рисунок 3.9 – список студентов после удаления



Удаление из списка студентов происходит путём ввода Уникального Номера студента (UID), который указан в первой колонке в списке студентов. После удаления студента, пользователю отображается новый список студентов, где уже удалён нужный студент.

### 2.3.5 Шифрование данных

В программе базы данных реализована функция шифрования и дешифрования данных о студентах ВУЗа. Шифровка данных происходит при выходе из программы с сохранением кода дешифрования. При запуске программы происходит автоматическая дешифровка базы данных, которыми позже можно спокойно оперировать.

Для шифрования и дешифровки данных используется библиотека OpenSSL и алгоритм AES. Реализация функция шифровки и дешифровки можно увидеть ниже.

```
void Crypt()
{
    system("openssl\\bin\\openssl.exe rand -hex -out key.txt 64");
    system("openssl\\bin\\openssl.exe enc -aes-256-cbc -pbkdf2 -pass file:key.txt -in 1.txt -out 1.txt.enc");
    if (remove("1.txt") != 0) {
        cout << "Ошибка удаления файла базы данных." << endl;
    }
    system("openssl\\bin\\openssl.exe rsautl -encrypt -pubin -inkey rsa.public -in key.txt -out key.txt.enc");
    if (remove("key.txt") != 0) {
        cout << "Ошибка удаления файла базы пароля базы данных." << endl;
    }
}
```

*Рисунок 3.10 – функция шифрования данных о студенте*

Шифрование данных реализовано с помощью пяти команд, занесённых в отдельную функцию, срабатывающих поочерёдно. Первой отработывает команда «`rand -hex -out key.txt 64`», которая генерирует уникальный код шифрования длиной 64 символа, который заносится в файл с наименованием `key.txt`. В дальнейшем данный код используется для шифрования базы данных студентов. Следующей командой происходит шифрование самой базы данных студентов, уже с использованием созданного ключа шифровки. Код команды: «`enc -aes-256-cbc -pbkdf2 -pass file: key.txt -in 1.txt -out 1.txt.enc`». Шифрование происходит с помощью алгоритма AES, где `key.txt` – файл с ключом шифрования, `1.txt` – файл базы данных студентов, `1.txt.enc` – зашифрованный файл базы данных.



После шифрования базы данных студентов, происходит удаление незашифрованных данных. В случае ошибки при удалении файла на дисплей выводится информация об ошибке. Последней отработывающей командой является: «`rsautl -encrypt -pubin -inkey rsa.public -in key.txt -out key.txt.enc`», где `rsa.public` – публичный ключ шифрования RSA, с помощью которого происходит шифрование ключа шифрования базы данных `key.txt` в `key.txt.enc`. Далее происходит удаление незашифрованного файла с паролем и, в случае возникновения ошибки при удалении, выводится соответствующее сообщение о подобном.

```
void Decrypt()
{
    system("openssl\\bin\\openssl.exe rsautl -decrypt -inkey rsa.private -in key.txt.enc -out key.txt");
    if (remove("key.txt.enc") != 0) {
        cout << "Ошибка удаления зашифрованного файла с паролем." << endl;
    }
    system("openssl\\bin\\openssl.exe enc -aes-256-cbc -d -pbkdf2 -pass file:key.txt -in 1.txt.enc -out 1.txt");
    if (remove("key.txt") != 0) {
        cout << "Ошибка удаления файла пароля." << endl;
    }
    if (remove("1.txt.enc") != 0) {
        cout << "Ошибка удаления зашифрованной базы данных." << endl;
    }
}
```

Рисунок 3.11 – функция дешифрования данных

Функция дешифрования создана по аналогичному шифрованию алгоритму и состоит из пяти последовательно срабатывающих команд, из которых три являются системными командами. Первой отработывает команда «`rsautl -decrypt -inkey rsa.private -in key.txt.enc -out key.txt`» - где `rsa.private` – приватный ключ RSA, с помощью которого дешифровывается файл с паролем `key.txt.enc` в `key.txt`, готовый к дешифровке базы данных. Далее происходит удаление файла с паролем. В случае возникновения ошибки на данном этапе удаления, на дисплей выводится информация об ошибке удаления файла.

Следом отработывает команда: «`enc -aes-256-cbc -d -pbkdf2 -pass file:key.txt -in 1.txt.enc -out 1.txt`», с помощью которой происходит дешифрование данных, где `1.txt.enc` – зашифрованная база данных студентов, `1.txt` – дешифрованная база студентов. Дешифровка происходит согласно ключу, полученному ранее, которые хранятся в файле `key.txt`. Как только дешифровка завершена происходит последовательное удаление файла с ключом шифрования и самой зашифрованной базы данных. В случае возникновения ошибок на любом из

этих этапов, на экран выведется сообщения с соответствующей ошибкой при удалении.

## **2.4 Руководство пользователя**

Программа доступная пользователю представляет собой каскад из функций, которые вызываются с помощью меню взаимодействия. Так при запуске программы на экран выводится базовая информация о базе данных ВУЗа, а именно количество студентов в базе данных и количество предметов, доступных к изучению. Пользователю предоставлена возможность к заходу в базу данных студентов или выходу из программы, путём ввода «1» или «0» для соответствующих функций.

При вводе «1», пользователю открывается следующее меню, с выбором действия в базе данных ВУЗа, а именно, переход к списку предметов, переход к списку студентов, переход к индивидуальному заданию или возврат в предыдущее меню.

При вводе в данном меню «1» - пользователю открывается список всех предметов, которые преподают в ВУЗе (0).

При вводе в меню «2» - пользователю открывается список всех студентов, которые занесены в базу данных. Данный раздел будет описан ниже (1)

При вводе в меню «3» - пользователя переводит к разделу выполнения индивидуального задания варианта. Данный раздел будет описан ниже (2)

При вводе в меню «4» - пользователь возвращается в предыдущее меню.

(0) – Как только пользователь перешёл в данный раздел, программа выводит весь список предметов, которые хранятся в базе данных ВУЗа, разбивая данные в две колонки. В первой колонке выводится уникальный номер предмета UID, а во второй само название предмета. В данном меню пользователь может увидеть весь список предметов, а также добавить новый предмет с помощью ввода в меню взаимодействия «1», а после само название предмета в соответствующей строке. При вводе в меню взаимодействия «0» - программа возвращает окно предыдущего меню.

(1) – при переходе к списку Студентов в базе данных, программа открывает и выводит весь список студентов занесённых в базу данных, разбивая информация в три колонки. В первую выводится уникальный номер студента UID, во второй отображается группа, в которой студент обучается, в третьей выводится Имя и Фамилия соответствующего студента. В нижней части окна, отображается меню взаимодействия со списком студентов.

При вводе в меню взаимодействия «1», пользователь должен ввести UID студента, информацию о котором хочет получить. После ввода UID, программа отобразит полную информацию о студенте в отдельном окне. Возврат к списку студентов происходит путём нажатия любой клавиши.

При вводе в меню взаимодействия «2», пользователю предоставлена возможность к вводу нового студента в базу данных. Программа очищает экран и выводит поля для заполнения информации о новом студенте. В случае допущения ошибки при вводе данных, которые не соответствуют формату ввода или содержат некорректные значения, программа возвращается к списку студентов, давая возможность пользователю ввести информацию повторно. В случае корректного заполнения всех полей, программа возвращает пользователю экран списка студентов, где уже будет отображаться новый студент.

При вводе в меню взаимодействия «3», пользователь получает возможность к изменению данных о конкретном студенте, введя UID в следующее поле. Экран очищается и выводится список пунктов, которые могут быть изменены. При вводе конкретных значений, на экран выводится текущее значение и предоставляется возможность к изменению поля студента. «1» - отвечает за изменение имени, «2» - за фамилию, «3» - за отчество студента, «4» - номер зачётной книжки студента, «5» - год поступления в вуз, «6» - перезапись пола студента, «7» - изменение наименования института, «8» - перезапись названия выпускающей кафедры, «9» - смена названия группы обучения, «10» - перезапись результатов сессии, где пользователь может выбрать конкретный предмет и конкретную оценку для изменения, следуя

инструкция на экране, «11» - перезапись даты рождения студента. При вводе «0» - программа вернёт в пользователя в предыдущее окно.

При выборе пункта «4», то есть ввода 4 в поле взаимодействия, пользователь способен удалить ненужного студента из базы данных. Достаточно ввести значение UID студента. После этого программа отработает и уберёт данного студента из базы данных, выводя новый список студентов.

При вводе «0» - программа возвращается в предыдущее меню.

(2) – при выборе пункта «3» в меню, пользователя переносит в отдельное меню индивидуального задания. Во время данного перехода программа формирует линейный список из студентов, которые хранятся в базе данных. В небольшом меню пользователю предоставлена возможность к выбору соответствующих пунктов индивидуального задания курсовой.

При вводе «1» - программа отобразит краткую характеристику по студентам только мужского пола, в том виде, в котором они закреплены в линейно-связанном списке. Как только программа завершит команды, при нажатии любой клавиши, пользователя вернут к списку пунктов индивидуального задания.

При вводе «2» - программа отобразит краткую характеристику по студентам только женского пола, в том виде, в котором они закреплены в линейно-связанном списке. Как только программа завершит команды, при нажатии любой клавиши, пользователя вернут к списку пунктов индивидуального задания.

При вводе «3» - программа отобразит краткую характеристику по студента любого пола, в том виде, в котором они закреплены в линейно-связанном списке. Как только программа завершит команды, при нажатии любой клавиши, пользователя вернут к списку пунктов индивидуального задания.

При вводе «4» - программа выполняет сортировку списка, в соответствии с заданием курсовой работы, а именно по году поступления в

ВУЗ. Как только программа завершит команды, при нажатии любой клавиши, пользователя вернут к списку пунктов индивидуального задания.

При вводе «0» - программа вернёт пользователя в предыдущее меню.

## **2.5 Системные требования**

Язык программирования: C++.

Операционная система: Linux Mint 20.1 выше или аналог, Windows 10.

ОЗУ: 1 Гб и более.

Свободное место на диске: 200 Мб и более.

## ЗАКЛЮЧЕНИЕ

В процессе выполнения курсовой работы была написана программа, по сортировке группы студентов по увеличению года поступления в ВУЗ, с поиском по лицам определённого пола, а также создание самой базы данных студентов и функции её редактирования.

В процессе выполнения работы были закреплены базовые навыки программирования, полученные при изучении дисциплины "Языки программирования", а также успешно применены на практике теоретические выкладки, используемые в процессе написания программы.

Особое внимание было уделено работе с динамической памятью и файлами, что позволило создать функциональный и гибкий интерфейс для пользователей. Кроме того, были рассмотрены и применены на практике методы симметричного и ассиметричного шифрования и дешифрования файлов, что повышает безопасность работы с базой данных.

Результаты тестирования продукта показали его высокую корректность и способность справляться с поставленной задачей.

Данная разработка имеет высокий потенциал для дальнейшего развития и внедрения в различные информационные системы. Она может быть использована в учебных заведениях, организациях, занимающихся управлением информацией о студентах, а также в других сферах, где требуется надежное и безопасное хранение и управление данными.

## СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Мерсов А. А. Основы объектно-ориентированного программирования на языке C++ [Электронный ресурс]: практикум / А. А. Мерсов, А. М. Русаков, В. В. Филатов. — М.: РТУ МИРЭА, 2021. — Электрон. опт. диск (ISO)
2. Страуструп Б. Программирование. Принципы и практика использования C++. – Litres, 2022.
3. Лафоре Р. Объектно-ориентированное программирование в C++. 4-е изд., перераб. и доп //Санкт-Петербург.: Питер. – 2022.
4. Стенли Липпман Язык программирования C++: полное руководство / Липпман Стенли, ЛажойеЖози. – Саратов: Профобразование, 2023. 1104 с.
5. Шилдт, Герберт C++ для начинающих. Шаг за шагом / Герберт Шилдт. - М.: ЭКОМ Паблишерз, 2020. - 640 с.
6. Страуструп, Б. Язык программирования C++: Специальное издание / Б. Страуструп; Пер. с англ. Н.Н. Мартынов. — М.: БИНОМ, 2017. — 1136 с.

## ПРИЛОЖЕНИЕ А Исходный код программы

### Hedear.h

```
#pragma once
#include <iostream>
#include <fstream>
#include <windows.h>
#include <string>
#include <fstream>
#include <iostream>

#define cls system("cls")
using namespace std;

void Crypt();
void Decrypt();
int Menu_Start();
int Menu_Menu();
int Menu_Subject_List();
int Menu_Student_List();
int Menu_Student_Info(int NUM);
int Menu_New_Student();
int Menu_New_Subject();
int Menu_Changing_Info(int NUM);
int Menu_Delete_User(int NUM);
int IndividualTask();
```

### Student\_FN.cpp

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

class Student_Fullname {
public:
    void SetFirstName(string name) {
        Firstname = name;
    }
    void SetSurname(string name) {
        Surname = name;
    }
    void SetPatronym(string name) {
        Pronymic = name;
    }
    string GetFirstName() {
        return Firstname;
    }
    string GetSurname() {
        return Surname;
    }
    string GetPronymic() {
        return Pronymic;
    }
protected:
    string Firstname;
    string Surname;
    string Pronymic;
};
```

### Grades.cpp

```
#include "Header.h"

struct strStudy_Info {
    unsigned short Year_entrance = 2017;
    string Faculty;
    string Department;
    string Group;
};
```



```

struct str_subject {
    unsigned short SUBJ_ID = 0;
    string Subject_Name;
    unsigned short Subject_mark = 5;
};

class clStudy_Info {
protected:
    unsigned short Y_ENTR;
    string Faculty;
    string Department;
    string Group;
public:
    unsigned short GetYearEntrance() {
        return Y_ENTR;
    }
    string GetFaculty() {
        return Faculty;
    }
    string GetDepartment() {
        return Department;
    }
    string GetGroup() {
        return Group;
    }
    strStudy_Info GetStudyInfo() {
        strStudy_Info ST_INFO;
        ST_INFO.Department = GetDepartment();
        ST_INFO.Faculty = GetFaculty();
        ST_INFO.Group = GetGroup();
        ST_INFO.Year_entrance = GetYearEntrance();
        return ST_INFO;
    }
    void SetYearEntrance(unsigned short Y_ENT) {
        Y_ENTR = Y_ENT;
    }
    void SetFaculty(string F) {
        Faculty = F;
    }
    void SetDepartment(string D) {
        Department = D;
    }
    void SetGroup(string G) {
        Group = G;
    }
    void SetStudyInfo(strStudy_Info* INFO) {
        SetYearEntrance(INFO->Year_entrance);
        SetFaculty(INFO->Faculty);
        SetDepartment(INFO->Department);
        SetGroup(INFO->Group);
    }
};

class subject {
protected:
    unsigned short SUBJ_ID;
public:
    string Subject_Name;
    unsigned short Subject_mark;
    subject() {
        SUBJ_ID = 000; Subject_Name = "Предмет заглушка"; Subject_mark = 5;
    };
    subject(str_subject* SUBJ) {
        SetSubject(SUBJ->SUBJ_ID, SUBJ->Subject_Name, SUBJ->Subject_mark);
    };
    void SetSubject(unsigned short subj_id, string subject_name, unsigned short subject_mark) {
        SUBJ_ID = subj_id; Subject_Name = subject_name; Subject_mark = subject_mark;
    }
    unsigned short GetID() {
        return SUBJ_ID;
    }
};

```

```

struct strSemestr {
    unsigned short SUBJ = 10;
    str_subject Sem[10];
};

class Semestr : public subject {
public:
    unsigned short SUBJ_NUM = 10;
    subject Sem[10];
    Semestr() {
        for (int i = 0; i < SUBJ_NUM; i++) {
            Sem[i].Subject_Name = "Заглушка";
            Sem[i].Subject_mark = 5;
        }
    }
};

```

## Student\_Info.cpp

```

#include "Header.h"
#include "Student_FN.cpp"
#include "Grades.cpp"

struct MyDate {
    unsigned short D;
    unsigned short M;
    unsigned short Y;
};

class Student_BD {
public:
    void SetBirthDate(MyDate* Date) {
        SetBirthDate(Date->D, Date->M, Date->Y);
    }
    void SetBirthDate(int d, int m, int y) {
        BD = d; BM = m; BY = y;
    }
    MyDate GetBirthDate() {
        MyDate SBirthDate;
        SBirthDate.D = BD;
        SBirthDate.M = BM;
        SBirthDate.Y = BY;
        return SBirthDate;
    }
    string GetBirthDateString() {
        string BirthDateString = to_string(BD) + '.' + to_string(BM) + '.' + to_string(BY);
        return BirthDateString;
    }
protected:
    unsigned short BD;
    unsigned short BM;
    unsigned short BY;
};

struct strStudent {
    string Firstname, Secondname, Patr;
    unsigned short UID = 0;
    string Zach;
    unsigned short BirthD = 1, BirthM = 1, BirthY = 2000;
    string Sex;
    string Department, Faculty, Group;
    unsigned short Y_Entrance = 2012;
    strSemestr strGrades[9];
};

class Student : public Student_Fullname, public Student_BD, public Semestr, public clStudy_Info {
public:
    int UID;
    void SetSTUDUID(string uid) {
        STUDENT_UID = uid;
    }
    void SetSex(string S) {

```

```

        sex = S;
    }
    string GetSTUDUID() {
        return STUDENT_UID;
    }
    string GetSex() {
        return sex;
    }
protected:
    string STUDENT_UID;
    string sex;
public:
    Semestr Grades[9];
    Student() {
        UID = 0;
        SetSTUDUID("00A0000"); SetFirstName("Заглушко"); SetSurname("Заглушкин");
        SetPatronim("Заглушкович");
        SetBirthDate(1, 1, 2004);
        SetSex("М");
        SetDepartment("HHH");
        SetFaculty("HH-0");
        SetGroup("HHHH-00-00");
        SetYearEntrance(2022);
    }
    Student(strStudent* S) {
        UID = S->UID;
        SetSTUDUID(S->Zach);
        SetFirstName(S->Firstname);
        SetSurname(S->Secondname);
        SetPatronim(S->Patr);
        SetBirthDate(S->BirthD, S->BirthM, S->BirthY);
        SetSex(S->Sex);
        SetDepartment(S->Department);
        SetFaculty(S->Faculty);
        SetGroup(S->Group);
        SetYearEntrance(S->Y_Entrance);
    }
};

```

## Курс.cpp

```

#pragma once
#include "Student_Info.cpp"
#pragma once
#include "Student_Info.cpp"

int isValidInt(string num) {
    if (num == "" || num.length() > 9)
        return 0;
    num.append("a");

    if (num.find_first_not_of("0123456789") != num.length() - 1)
        return 0;
    return 1;
}

int isValidSex(string num) {
    if (num == "" || num.length() != 1)
        return 0;
    num.append("a");
    if (num.find_first_not_of("МЖ") != num.length() - 1)
        return 0;
    return 1;
}

int isValidSubject(string name) {
    if (name == "" || name.length() > 60)
        return 0;
    name.append("*");
    if (name.find_first_not_of("АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯабвгдеёжзийклмнопрстуфхцчшщъыьэюя -") !=
        name.length() - 1)
        return 0;
    return 1;
}

```

```

};

int isValidSurName(string name) {
    if (name == "" || name.length() > 40)
        return 0;
    name.append("*");
    if (name.find_first_not_of("АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯабвгдеёжзийклмнопрстуфхцчшщъыьэюя") != name.length() - 1)
        return 0;
    return 1;
}

int isValidNamePatronim(string name) {
    if (name == "" || name.length() > 30)
        return 0;
    name.append("*");
    if (name.find_first_not_of("АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯабвгдеёжзийклмнопрстуфхцчшщъыьэюя") != name.length() - 1)
        return 0;
    return 1;
}

int isValidMark(string name) {
    if (name == "" || name.length() > 1)
        return 0;
    name.append("*");
    if (name.find_first_not_of("2345") != name.length() - 1)
        return 0;
    return 1;
}

int isValidFaculty(string name) {
    if (name == "" || name.length() > 6)
        return 0;
    name.append("*");
    if (name.find_first_not_of("АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ") != name.length() - 1)
        return 0;
    return 1;
}

int isValidKaf(string name) {
    if (name == "" || name.length() > 5 || name.length() < 4)
        return 0;
    name.append("*");
    if (name.find_first_not_of("АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ-1234567890") != name.length() - 1)
        return 0;
    if (name.find('.') != 2)
        return 0;
    if (name.find_first_not_of("АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ") != 2)
        return 0;
    if (name.find_first_of("1234567890") != 3)
        return 0;
    if (name.find_last_of("АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ") == 4)
        return 0;
    return 1;
}

int isValidZach(string name) {
    if (name == "" || name.length() != 7)
        return 0;
    name.append("*");
    if (name.find_first_not_of("АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ1234567890") != name.length() - 1)
        return 0;
    if (name.find_first_not_of("1234567890") != 2 && name.find_first_not_of("1234567890") < name.length()) {
        cout << name.find_first_not_of("1234567890");
        return 0;
    }
    if (name.find_last_of("0123456789") != 6 && name.find_first_not_of("1234567890") < name.length() - 1) {
        cout << name.find_last_of("0123456789");
        return 0;
    }
    return 1;
}

int isValidGroup(string name) {

```

```

if (name == "" || name.length() != 10)
    return 0;
name.append("*");
if (name.find_first_not_of("АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ1234567890-") != name.length() - 1)
    return 0;
if (name.find('-') != 4)
    return 0;
if (name.rfind('-') != 7)
    return 0;
if (name.find_last_of("АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ") != 3 &&
name.find_last_of("АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ") < name.length() - 1)
    return 0;
if (name.find_first_of("1234567890") != 5)
    return 0;
if (name.find_last_of("1234567890") != 9 && name.find_last_of("1234567890") < name.length() - 1)
    return 0;
return 1;
}

int isValidBirth(int BD, int BM, int BY) {
    if ((BY < 1970) || BY > 2023)
        return 0;
    if ((BM < 1) || (BM > 12))
        return 0;
    switch (BM)
    {
        case 1: case 3: case 5: case 7: case 8: case 10: case 12: {
            if ((BD > 31) || (BD < 0))
                return 0;
        }
        case 4: case 6: case 9: case 11: {
            if ((BD > 30) || (BD < 0))
                return 0;
        }
        case 2: {
            if (((BD > 29) || (BD < 0)) && (BY % 4 == 0))
                return 0;
            if (((BD > 28) || (BD < 0)) && ((BY % 4 != 0) || (BY == 2000)))
                return 0;
        }
    }
    return 1;
}

int isValidEntrance(int Year) {
    if (Year < 2019)
        return 0;
    if (Year > 2023)
        return 0;
    return 1;
}

void rus() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
}

int Subjects_Count() {
    char CUR_SUBJ; unsigned short Subject_count, count = 0;
    fstream file;
    file.open("subject_DB.txt", ios::in | ios::app);
    while (!file.eof()) {
        file >> CUR_SUBJ;
        if (CUR_SUBJ == '+') {
            count++;
        }
    }
    Subject_count = count / 2;
    return Subject_count;
}

int Students_Count() {
    char CUR_STD; unsigned short count = 0;
    fstream file;
    file.open("1.txt", ios::in | ios::app);

```

```

while (!file.eof()) {
    file >> CUR_STD;
    if (CUR_STD == '#') {
        count++;
    }
}
return count-1;
}

int Menu_Start() {
    int STD_NUM = Students_Count(); int PRG_NUM = Subjects_Count();
    string MENU_TEMP; int MENU_NUM = -1;
    for (; MENU_NUM != 0;) {
        cout << '|'; cout.width(60); cout.fill('-'); cout << '|' << endl;
        cout << '|'; cout.width(59); cout.fill(' '); cout << left << " Добро пожаловать в базу данных студентов ВУЗа"; cout << '|' << endl;
        cout << '|'; cout.width(60); cout.fill('-'); cout << right << '|' << endl;
        cout << '|'; cout.width(59); cout.fill(' '); cout << left << " Наименование ВУЗа: РТУ МИРЭА"; cout << '|' << endl;
        cout << '|'; cout.width(60); cout.fill('-'); cout << right << '|' << endl;
        cout << '|'; cout.width(28); cout.fill(' '); cout << left << " Кол-во студентов: " << STD_NUM << '|';
        cout.width(28); cout.fill(' '); cout << left << ' ' << '|' << endl;
        cout << '|'; cout.width(27); cout << left << " Кол-во предметов: " << PRG_NUM << '|';
        cout.width(28); cout.fill(' '); cout << " " << left << '|' << endl;
        cout << '|'; cout.width(60); cout.fill('-'); cout << right << '|' << endl;
        cout << '|'; cout.width(59); cout.fill(' '); cout << left << " Выберите операцию взаимодействия: "; cout << '|' << endl;
        cout << '|'; cout.width(59); cout.fill(' '); cout << left << " 1 - Войти в базу данных"; cout << '|' << endl;
        cout << '|'; cout.width(59); cout.fill(' '); cout << left << " 0 - Выйти из базы данных"; cout << '|' << endl;
        cout << '|'; cout.width(60); cout.fill('-'); cout << right << '|' << endl;
        cout << "Введите операцию взаимодействия: "; getline(cin, MENU_TEMP);
        if (isValidInt(MENU_TEMP)) {
            MENU_NUM = stoi(MENU_TEMP);
            switch (MENU_NUM)
            {
                case 1: cout << "Выполняем вход в базу данных студентов"; system("cls"); Menu_Menu(); break;
                case 0: {cout << "Выход из базы данных студентов\n"; return 0;}
                default: cout << "Ошибка ввода\n"; Sleep(1000); system("cls"); break;
            }
        }
        else { cout << "Ошибка ввода\n"; cls; }
    }
}

int Menu_Menu() {
    string MENU_TEMP; int NUM_MENU = -1, I_MENU = 1;
    while (I_MENU == 1) {
        cout << '|'; cout.width(60); cout.fill('-'); cout << '|' << endl;
        cout << '|'; cout.width(59); cout.fill(' '); cout << "База данных студентов ВУЗа" << '|' << endl;
        cout << '|'; cout.width(60); cout.fill('-'); cout << '|' << endl;
        cout << '|'; cout.width(59); cout.fill(' '); cout << left << " Выберите операцию взаимодействия: "; cout << '|' << endl;
        cout << '|'; cout.width(59); cout.fill(' '); cout << left << " 1 - Переход к базе предметов"; cout << '|' << endl;
        cout << '|'; cout.width(59); cout.fill(' '); cout << left << " 2 - Открыть список студентов"; cout << '|' << endl;
        cout << '|'; cout.width(59); cout.fill(' '); cout << left << " 3 - Выполнение индивидуального задания"; cout << '|' << endl;
        cout << '|'; cout.width(59); cout.fill(' '); cout << left << " 4 - Назад в меню"; cout << '|' << endl;
        cout << '|'; cout.width(60); cout.fill('-'); cout << right << '|' << endl;
        cout << "Введите операцию взаимодействия: "; getline(cin, MENU_TEMP);
        if (isValidInt(MENU_TEMP)) {
            NUM_MENU = stoi(MENU_TEMP);
            switch (NUM_MENU)
            {
                case 1: {cout << "Переход к заполнению полей нового предмета"; Sleep(1000); system("cls"); Menu_Subject_List(); break;}
                case 2: {cout << "Переход к списку студентов"; Sleep(1000); system("cls"); Menu_Student_List(); break;}
                case 3: {cout << "Переход к выполнению индивидуального задания"; system("cls"); Sleep(1000); IndividualTask(); break;}
                case 4: {cout << "Возврат в меню"; Sleep(500); system("cls"); I_MENU = 0; break;}
                default: {cout << "Ошибка ввода"; Sleep(1000); system("cls"); break;}
            }
        }
        else { cout << "Ошибка ввода"; cls; }
    }
    return 0;
}

int Menu_Subject_List() {
    string MENU_TEMP, temp; int NUM_MENU; string CUR_SUBJ; unsigned short CUR_ID; unsigned short subj_num = Subjects_Count(); int
    I_MENU = -1;
    while (I_MENU != 0) {
        fstream file;

```

```

file.open("subject_DB.txt", ios::out | ios::in | ios::app);
cout << '|'; cout.width(80); cout.fill('-'); cout << '|' << endl;
cout << '|'; cout.width(79); cout.fill(' '); cout << "База данных предметов по программам ВУЗа" << '|' << endl;
cout << '|'; cout.width(80); cout.fill('-'); cout << '|' << endl;
cout << '|'; cout.width(15); cout.fill(' '); cout << left << "UID предмета" << '|'; cout.width(63); cout.fill(' '); cout << "Наименование
предмета" << '|' << endl;
cout << '|'; cout.width(80); cout.fill('-'); cout << '|' << endl;
char temp1 = '-';
for (int i = 0; i < subj_num; i++) {
    file >> temp1;
    while (temp1 != '+') {
        file.seekp(-1, ios::cur);
        file >> CUR_ID;
        file >> temp1;
    }
    file >> temp1; char SUBJ_temp = ' '; CUR_SUBJ = ' ';
    while (temp1 != '+') {
        file.seekp(-1, ios::cur);
        file >> SUBJ_temp;
        CUR_SUBJ += SUBJ_temp;
        file >> temp1;
    }
    cout << '|'; cout.width(15); cout.fill(' '); cout << left << CUR_ID << '|'; cout.width(63); cout.fill(' '); cout << CUR_SUBJ << '|' << endl;
}
file.close();
cout << '|'; cout.width(80); cout.fill('-'); cout << right << '|' << endl;
cout << '|'; cout.width(79); cout.fill(' '); cout << "Выберите операцию с базой предметов ВУЗа:" << '|' << endl;
cout << '|'; cout.width(80); cout.fill('-'); cout << '|' << endl;
cout << '|'; cout.width(79); cout.fill(' '); cout << "0 - Возврат в предыдущее меню" << '|' << endl;
cout << '|'; cout.width(79); cout.fill(' '); cout << "1 - Ввести новый предмет в базу данных" << '|' << endl;
cout << '|'; cout.width(80); cout.fill('-'); cout << '|' << endl;
cout << "Введите операцию взаимодействия: "; getline(cin, MENU_TEMP);
if (isValidInt(MENU_TEMP)) {
    NUM_MENU = stoi(MENU_TEMP);
    switch (NUM_MENU)
    {
        case 1: { Menu_New_Subject(); cls; break; };
        case 0: { cout << "Возврат к меню базы данных"; Sleep(500); system("cls"); I_MENU = 0; break; };
        default: { cout << "Ошибка ввода"; Sleep(500); system("cls"); break; };
    }
}
}
return 0;
}

int Menu_Student_List() {
    string MENU_TEMP; int NUM_MENU; string UID; unsigned short UID_I, I_MENU = -1;
    while (I_MENU != 0) {
        fstream file;
        file.open("1.txt", ios_base::in | ios_base::out | ios_base::app);
        cout << '|'; cout.width(80); cout.fill('-'); cout << '|' << endl;
        cout << '|'; cout.width(79); cout.fill(' '); cout << "Список студентов ВУЗа" << '|' << endl;
        cout << '|'; cout.width(80); cout.fill('-'); cout << '|' << endl;
        cout << '|'; cout.width(15); cout.fill(' '); cout << "UID студента" << '|'; cout.width(20); cout.fill(' '); cout << "Группа студента" << '|';
        cout.width(42); cout << "ФИ Студента" << '|' << endl;
        cout << '|'; cout.width(80); cout.fill('-'); cout << '|' << endl;
        string STR, STR1, UID_S;
        for (int i = 0; i < Students_Count(); i++) {
            file >> STR;
            STR1 = STR;
            UID_S = STR1;
            UID_S.erase(UID_S.find('+'), UID_S.length());
            if (i < 10) {
                STR.erase(0, 2);
            }
            if (i >= 10) {
                STR.erase(0, 3);
            }
            STR.erase(STR.find('.'), STR.length());
            STR.erase(STR.rfind('.'), STR.length());
            STR1.erase(0, STR1.find('.') + 1);
            STR1.erase(STR1.find('.'), STR1.length()); STR1.erase(0, STR1.rfind('.') + 1);
            cout << '|'; cout.width(15); cout.fill(' '); cout << UID_S; cout << '|'; cout.width(20); cout.fill(' '); cout << STR1; cout << '|'; cout.width(42);
            cout.fill(' '); cout << STR; cout << '|' << endl;
        }
    }
}

```

```

cout << '|'; cout.width(80); cout.fill('-'); cout << '|' << endl;
cout << '|'; cout.width(79); cout.fill(' '); cout << left << " Выберите операцию взаимодействия: "; cout << '|' << endl;
cout << '|'; cout.width(79); cout.fill(' '); cout << "1 - Вывод информации о студенте по UID" << '|' << endl;
cout << '|'; cout.width(79); cout.fill(' '); cout << "2 - Добавление нового студента в Базу Данных" << '|' << endl;
cout << '|'; cout.width(79); cout.fill(' '); cout << "3 - Редактирование информации о студенте по UID" << '|' << endl;
cout << '|'; cout.width(79); cout.fill(' '); cout << "4 - Удаление информации о студенте по UID" << '|' << endl;
cout << '|'; cout.width(79); cout.fill(' '); cout << "0 - Возврат в предыдущее меню" << '|' << endl;
cout << '|'; cout.width(80); cout.fill('-'); cout << right << '|' << endl;
cout << "Введите операцию взаимодействия: "; getline(cin, MENU_TEMP);
if (isValidInt(MENU_TEMP)) {
    NUM_MENU = stoi(MENU_TEMP);
    switch (NUM_MENU) {
        case 1: {cout << "Выбран вывод информации о студенте по UID: ";
            getline(cin, UID);
            if (isValidInt(UID)) {
                UID_I = stoi(UID);
                int n = Students_Count();
                cls; Menu_Student_Info(UID_I);
            }
            else {
                cout << "Ошибка ввода"; Sleep(1000); cls; break;
            }
            break;
        }
        case 2: {
            cout << "Выбрано добавление нового студента в Базу Данных"; Sleep(1000); cls; Menu_New_Student(); break;
        }
        case 3: {
            cout << "Выбрано редактирование информации о студенте по UID : "; getline(cin, UID);
            if (isValidInt(UID)) {
                UID_I = stoi(UID);
                int n = Students_Count();
                cls; Menu_Changing_Info(UID_I);
            }
            else {
                cout << "Ошибка ввода"; Sleep(1000); cls; break;
            }
            break;
        }
        case 4: {cout << "Выбран пункт удаления информации о студенте по UID: "; getline(cin, UID);
            if (isValidInt(UID)) {
                UID_I = stoi(UID);
                int n = Students_Count();
                if (UID_I + 1 <= n) {
                    cls; Menu_Delete_User(UID_I);
                }
                else {
                    cout << "Ошибка ввода"; Sleep(1000); cls; break;
                }
            }
            else {
                cout << "Ошибка ввода"; Sleep(1000); cls; break;
            }
            break;
        }
        case 0: {cout << "Возврат в предыдущее меню"; Sleep(1000); cls; I_MENU = 0; break; }
        default: {cout << "Ошибка ввода"; Sleep(1000); cls; break; }
    }
}
else { cout << "Ошибка ввода"; cls; }
}
return 0;
};

Student* ReturnInfo(int Student_Num) {
    fstream file;
    file.open("1.txt", ios_base::in | ios_base::out | ios_base::app);
    string Student_String, STD_NUM; unsigned short UID; string STD_TEMP;
    file >> Student_String;
    STD_NUM = Student_String; STD_NUM.erase(STD_NUM.find('+'), STD_NUM.length()); UID = stoi(STD_NUM);
    while (UID != Student_Num) {
        if (file.eof()) {
            Menu_Student_List();
            break;
        }
    }
}

```



```

file >> Student_String;
STD_NUM = Student_String; STD_NUM.erase(STD_NUM.find('+'), STD_NUM.length()); UID = stoi(STD_NUM);
}
Student_String.erase(0, Student_String.find('+') + 1);
Student* clsStudent = new Student;
MyDate* BirthDateS = new MyDate; unsigned short D, M, Y;
STD_TEMP = Student_String; STD_TEMP.erase(STD_TEMP.find('.'), STD_TEMP.length());
clsStudent->SetFirstName(STD_TEMP);
STD_TEMP = Student_String; STD_TEMP.erase(0, STD_TEMP.find('.') + 1); STD_TEMP.erase(STD_TEMP.find('.'), STD_TEMP.length());
clsStudent->SetSurname(STD_TEMP);
STD_TEMP = Student_String; STD_TEMP.erase(0, STD_TEMP.find('.') + 1); STD_TEMP.erase(0, STD_TEMP.find('.') + 1);
STD_TEMP.erase(STD_TEMP.find('.'), STD_TEMP.length());
clsStudent->SetPatronym(STD_TEMP);
Student_String.erase(0, Student_String.find('.') + 1);
STD_TEMP = Student_String; STD_TEMP.erase(STD_TEMP.find('.'), STD_TEMP.length());
clsStudent->SetDepartment(STD_TEMP);
STD_TEMP = Student_String; STD_TEMP.erase(0, STD_TEMP.find('.') + 1); STD_TEMP.erase(STD_TEMP.find('.'), STD_TEMP.length());
clsStudent->SetFaculty(STD_TEMP);
STD_TEMP = Student_String; STD_TEMP.erase(0, STD_TEMP.find('.') + 1); STD_TEMP.erase(0, STD_TEMP.find('.') + 1);
STD_TEMP.erase(STD_TEMP.find('.'), STD_TEMP.length());
clsStudent->SetGroup(STD_TEMP);
Student_String.erase(0, Student_String.find('.') + 1);
STD_TEMP = Student_String;
STD_TEMP.erase(STD_TEMP.find('.'), STD_TEMP.length());
D = stoi(STD_TEMP);
STD_TEMP = Student_String; STD_TEMP.erase(0, STD_TEMP.find('.') + 1); STD_TEMP.erase(STD_TEMP.find('.'), STD_TEMP.length());
M = stoi(STD_TEMP);
STD_TEMP = Student_String; STD_TEMP.erase(0, STD_TEMP.find('.') + 1); STD_TEMP.erase(0, STD_TEMP.find('.') + 1);
STD_TEMP.erase(STD_TEMP.find('.'), STD_TEMP.length());
Y = stoi(STD_TEMP);
BirthDateS->D = D; BirthDateS->M = M; BirthDateS->Y = Y;
clsStudent->SetBirthDate(BirthDateS);
Student_String.erase(0, Student_String.find('.') + 1);
STD_TEMP = Student_String; STD_TEMP.erase(STD_TEMP.find('>'), STD_TEMP.length());
clsStudent->SetYearEntrance(stoi(STD_TEMP));
Student_String.erase(0, Student_String.find('>') + 1);
STD_TEMP = Student_String; STD_TEMP.erase(STD_TEMP.find('<'), STD_TEMP.length());
clsStudent->SetSTUDUID(STD_TEMP);
Student_String.erase(0, Student_String.find('<') + 1);
STD_TEMP = Student_String; STD_TEMP.erase(STD_TEMP.find('!'), STD_TEMP.length());
clsStudent->SetSex(STD_TEMP);
Student_String.erase(0, Student_String.find('!') + 1);
for (int i = 1; i < (2023 - clsStudent->GetYearEntrance()) * 2; i++) {
    STD_TEMP = Student_String;
    STD_TEMP.erase(STD_TEMP.find(')'), STD_TEMP.length());
    string SBJ_TEMP;
    STD_TEMP.erase(STD_TEMP.find(')'), STD_TEMP.length());
    int SBJ_NUM1 = stoi(STD_TEMP);
    clsStudent->Grades[i-1].SUBJ_NUM = SBJ_NUM1;
    Student_String.erase(0, Student_String.find('!') + 1);
    STD_TEMP = Student_String;
    for (int j = 0; j < SBJ_NUM1; j++) {
        SBJ_TEMP = STD_TEMP;
        SBJ_TEMP.erase(SBJ_TEMP.find('.'), SBJ_TEMP.length());
        clsStudent->Grades[i-1].Sem[j].Subject_Name = SBJ_TEMP;
        SBJ_TEMP = STD_TEMP;
        SBJ_TEMP.erase(0, SBJ_TEMP.find('.') + 1); SBJ_TEMP.erase(SBJ_TEMP.find('+'), SBJ_TEMP.length());
        clsStudent->Grades[i-1].Sem[j].Subject_mark = stoi(SBJ_TEMP);
        STD_TEMP.erase(0, STD_TEMP.find('+') + 1);
    }
    Student_String.erase(0, Student_String.find('!') + 1);
}
return clsStudent;
}

int Menu_Student_Info(int Student_Num) {
    Student* STUD_INFO = ReturnInfo(Student_Num);
    cout << '|'; cout.width(80); cout.fill('-'); cout << '|' << endl;
    cout << '|'; cout.width(79); cout.fill(' '); cout << "Информация о студенте ВУЗа" << '|' << endl;
    cout << '|'; cout.width(80); cout.fill('-'); cout << '|' << endl;
    cout << '|'; cout.width(12); cout.fill(' '); cout << "UID студента" << '|'; cout.width(21); cout.fill(' '); cout << "Зачётная книжка" << '|' << endl;
    cout << '|'; cout.width(12); cout.fill(' '); cout << Student_Num << '|'; cout.width(21); cout.fill(' '); cout << STUD_INFO->GetSTUDUID() << '|' << endl;
    cout << '|'; cout.width(80); cout.fill('-'); cout << '|' << endl;
}

```

```

        cout << '|'; cout.width(21); cout.fill(' '); cout << "Фамилия студента" << '|'; cout.width(21); cout.fill(' '); cout << "Имя студента" << '|';
        cout.width(35); cout.fill(' '); cout << "Отчество студента" << '|' << endl;
        cout << '|'; cout.width(21); cout.fill(' '); cout << STUD_INFO->GetSurname() << '|'; cout.width(21); cout.fill(' '); cout << STUD_INFO-
>GetFirstName() << '|'; cout.width(35); cout.fill(' '); cout << STUD_INFO->GetPronymic() << '|' << endl;
        cout << '|'; cout.width(80); cout.fill('-'); cout << '|' << endl;
        cout << '|'; cout.width(21); cout.fill(' '); cout << "Институт" << '|'; cout.width(21); cout.fill(' '); cout << "Кафедра" << '|'; cout.width(35);
        cout.fill(' '); cout << "Группа" << '|' << endl;
        cout << '|'; cout.width(21); cout.fill(' '); cout << STUD_INFO->GetDepartment() << '|'; cout.width(21); cout.fill(' '); cout << STUD_INFO-
>GetFaculty() << '|'; cout.width(35); cout.fill(' '); cout << STUD_INFO->GetGroup() << '|' << endl;
        cout << '|'; cout.width(80); cout.fill('-'); cout << '|' << endl;
        cout << '|'; cout.width(12); cout.fill(' '); cout << "Пол студента" << '|'; cout.width(21); cout.fill(' '); cout << "Год поступления" << '|';
        cout.width(12); cout << "Дата рождения" << endl;
        cout << '|'; cout.width(12); cout.fill(' '); cout << STUD_INFO->GetSex() << '|'; cout.width(21); cout.fill(' '); cout << STUD_INFO-
>GetYearEntrance() << '|'; cout.width(12); cout << STUD_INFO->GetBirthDateString() << endl;
        cout << '|'; cout.width(80); cout.fill('-'); cout << '|' << endl;
        cout << '|'; cout.width(79); cout.fill(' '); cout << "Сведения по оценкам за сессии (по семестрам)" << '|' << endl;
        for (int i = 1; i < (2023 - STUD_INFO->GetYearEntrance())*2; i++) {
            cout << '|'; cout.width(80); cout.fill('-'); cout << '|' << endl;
            cout << '|'; cout.width(71); cout.fill(' '); cout << i << " семестр" << '|' << endl;
            cout << '|'; cout.width(80); cout.fill('-'); cout << '|' << endl;
            cout << '|'; cout.width(69); cout.fill(' '); cout << "Наименование дисциплины" << '|'; cout.width(9); cout.fill(' '); cout << "Оценка" << '|' <<
            endl;
            cout << '|'; cout.width(80); cout.fill('-'); cout << '|' << endl;
            for (int j = 0; j < STUD_INFO->Grades[i-1].SUBJ_NUM; j++) {
                cout << '|'; cout.width(69); cout.fill(' '); cout << STUD_INFO->Grades[i-1].Sem[j].Subject_Name << '|'; cout.width(9); cout.fill(' '); cout <<
                STUD_INFO->Grades[i-1].Sem[j].Subject_mark << '|' << endl;
                cout << '|'; cout.width(80); cout.fill('-'); cout << '|' << endl;
            }
        }
    }
    system("pause"); clr; return 0;
};

int Menu_New_Subject() {
    string TEMP_SUBJ; char CUR_SUBJ; unsigned short Subject_count;
    fstream file;
    file.open("subject_DB.txt", ios::in | ios::app);
    Subject_count = Subjects_Count();
    cout << "Вы перешли в меню ввода нового предмета" << endl;
    cout << "Введите название нового предмета" << endl;
    cout << "Название не должно содержать цифр и не должно быть больше 60 символов" << endl;
    cout << "Название нового предмета: "; getline(cin, TEMP_SUBJ);
    if (isValidSubject(TEMP_SUBJ)) {
        str_subject subject_temp;
        subject_temp.Subject_Name = TEMP_SUBJ;
        subject_temp.Subject_mark = 5;
        subject_temp.SUBJ_ID = Subject_count;
        subject* clsubject = new subject(&subject_temp);
        file << clsubject->GetID() << '+' << clsubject->Subject_Name << '+' << endl;
        file.close();
        delete clsubject;
    }
    clr; return 0;
};

int Menu_New_Student() {
    int I_MENU = -1, flag = 1;
    while (I_MENU != 0) {
        string Name, Surname, Patronim, Sex, Day, Month, Year, Year_ENT, Kaf, Institute, Book, Group, Student_UID, UID_SUBJ, Mark, strfile,
        CUR_SUBJ, TempSTR, TEMP, STUD_STR = "";
        unsigned short D, M, Y, Y_ENT, UID, Mark_ST, STD_NUM = 0;
        fstream num;
        num.open("1.txt", ios_base::in | ios_base::out | ios_base::app);
        for (int i = 0; i < Students_Count(); i++) {
            num >> TEMP;
            TEMP.erase(TEMP.find('+'), TEMP.length());
            STD_NUM = stoi(TEMP);
        }
        fstream file;
        file.open("1.txt", ios_base::in | ios_base::out | ios_base::app);
        file.seekp(0);

        fstream subjects_DB;
        subjects_DB.open("subject_DB.txt", ios_base::out | ios_base::in | ios_base::app);
        cout << "Вы перешли к полю заполнения нового студента. " << endl;
        cout << "Введите Имя студента: ";
    }
}

```

```

getline(cin, Name);
if (!IsValidNamePatronym(Name)) {
    cout << "Ошибка ввода данных, откат в начало заполнения информации" << endl;
    Sleep(1000); flag = 0;
    break;
}
cout << "Введите Фамилию студента: ";
getline(cin, Surname);
if (!IsValidSurName(Surname)) {
    cout << "Ошибка ввода данных, откат в начало заполнения информации" << endl;
    Sleep(1000); flag = 0;
    break;
}
cout << "Введите Отчество студента: ";
getline(cin, Patronim);
if (!IsValidNamePatronym(Patronim)) {
    cout << "Ошибка ввода данных, откат в начало заполнения информации." << endl;
    Sleep(1000); flag = 0;
    break;
}
cout << "Введите пол студента (М,Ж): ";
getline(cin, Sex);
if (IsValidSex(Sex)) {

}
else {
    cout << "Ошибка ввода данных, откат в начало заполнения информации" << endl;
    Sleep(1000); flag = 0;
    break;
}
cout << "Начинаем заполнение даты рождения" << endl;
cout << "Введите год рождения" << endl;
getline(cin, Year);
if (IsValidInt(Year)) {
    Y = stoi(Year);
}
else {
    cout << "Ошибка ввода данных, откат в начало заполнения информации" << endl;
    Sleep(1000); flag = 0;
    break;
}
cout << "Введите месяц рождения" << endl;
getline(cin, Month);
if (IsValidInt(Month)) {
    M = stoi(Month);
}
else {
    cout << "Ошибка ввода данных, откат в начало заполнения информации" << endl;
    Sleep(1000); flag = 0;
    break;
}
cout << "Введите день рождения" << endl;
getline(cin, Day);
if (IsValidInt(Day)) {
    D = stoi(Day);
}
else {
    cout << "Ошибка ввода данных, откат в начало заполнения информации" << endl;
    Sleep(1000); flag = 0;
    break;
}
if (!IsValidBirth(D, M, Y)) {
    cout << "Ошибка ввода данных, откат в начало заполнения информации" << endl;
    Sleep(1000); flag = 0;
    break;
}
cout << "Введите институт обучения" << endl;
getline(cin, Institute);
if (!IsValidFaculty(Institute)) {
    cout << "Ошибка ввода данных, откат в начало заполнения информации" << endl;
    Sleep(1000); flag = 0;
    break;
}
cout << "Введите кафедру обучения, в формате (НН-1)" << endl;
getline(cin, Kaf);

```

```

if (!IsValidKaf(Kaf)) {
    cout << "Ошибка ввода данных, откат в начало заполнения информации" << endl;
    Sleep(1000); flag = 0;
    break;
}
cout << "Введите группу обучения, в формате (НННН-00-00)" << endl;
getline(cin, Group);
if (!IsValidGroup(Group)) {
    cout << "Ошибка ввода данных, откат в начало заполнения информации" << endl;
    Sleep(1000); flag = 0;
    break;
}
cout << "Введите номер зачётной книжки" << endl;
getline(cin, Student_UID);
if (!IsValidZach(Student_UID)) {
    cout << "Ошибка ввода данных, откат в начало заполнения информации" << endl;
    Sleep(1000); flag = 0;
    break;
}
cout << "Введите год поступления в ВУЗ" << endl;
getline(cin, Year_ENT);
if (!IsValidInt(Year_ENT)) {
    Y_ENT = stoi(Year_ENT);
    if (!(Y - Y_ENT > 16) || !IsValidEntrance(Y_ENT)) {
        cout << "Ошибка ввода данных, откат в начало заполнения информации" << endl;
        Sleep(1000); flag = 0;
        break;
    }
}
else {
    cout << "Ошибка ввода данных, откат в начало заполнения информации" << endl;
    Sleep(1000); flag = 0;
    break;
}
unsigned short sem = (2023 - Y_ENT) * 2;
strStudent* s_Student = new strStudent;
s_Student->Firstname = Name; s_Student->Secondname = Surname; s_Student->Patr = Patronim;
s_Student->Sex = Sex; s_Student->BirthD = D; s_Student->BirthM = M; s_Student->BirthY = Y;
s_Student->Department = Institute; s_Student->Faculty = Kaf; s_Student->Group = Group;
s_Student->Zach = Student_UID; s_Student->Y_Entrance = Y_ENT;
subjects_DB >> strfile;
Student* Stud;
Stud = new Student(s_Student);
STUD_STR = to_string(STD_NUM + 1) + "+" + Stud->GetFirstName() + "." + Stud->GetSurname() + "." + Stud->GetPronymic() + "." + Stud->GetDepartment() + "." + Stud->GetFaculty() + "." + Stud->GetGroup() + ":" + Stud->GetBirthDateString() + ":" + to_string(Stud->GetYearEntrance()) + ">" + Stud->GetSTUDUID() + "<" + Stud->GetSex() + "!";
string empty;
for (int i = 1; i < sem; i++) {
    str_subject* Subj = new str_subject;
    cout << "Идёт заполнение семестра " << i << endl;
    cout << "Введите количество предметов за сессию: "; getline(cin, TEMP);
    if (!IsValidInt(TEMP)) {
        cout << "Ошибка ввода данных, откат в начало заполнения информации" << endl;
        Sleep(1000); flag = 0; break;
    }
    int SUBJ_SEM = stoi(TEMP);
    if ((SUBJ_SEM > 10) && (SUBJ_SEM < 1)) {
        cout << "Ошибка ввода данных, откат в начало заполнения информации" << endl;
        Sleep(1000); flag = 0; break;
    }
    STUD_STR += TEMP;
    STUD_STR += '!';
    for (int j = 0; j < SUBJ_SEM; j++) {
        cout << "Введите UID предмета" << endl;
        getline(cin, UID_SUBJ);
        if (!IsValidInt(UID_SUBJ)) {
            cout << "Ошибка ввода данных, откат в начало заполнения информации" << endl;
            Sleep(1000); flag = 0; break;
        }
        if (!IsValidInt(UID_SUBJ)) {
            UID = stoi(UID_SUBJ);
            if (UID > Subjects_Count()) {
                cout << "Ошибка ввода данных. Откат заполнения."; flag = 0;
                Sleep(1000); flag = 0; break;
            }
        }
    }
}

```

```

    }
    UID_SUBJ += '+';
    int temp = -1;
    fstream fsubj;
    fsubj.open("subject_DB.txt", ios_base::out || ios_base::app || ios_base::in);
    while (!fsubj.eof()) {
        fsubj >> strfile;
        int temp = strfile.find(UID_SUBJ);
        if (temp == 0) break;
    }
    if (UID < 10) {
        strfile.replace(0, 2, empty);
        strfile.replace(strfile.length() - 1, 1, empty);
    }
    if ((UID >= 10) && (UID < 100)) {
        strfile.replace(0, 3, empty);
        strfile.replace(strfile.length() - 1, 1, empty);
    }
    if (UID >= 100) {
        strfile.replace(0, 4, empty);
        strfile.replace(strfile.length() - 1, 1, empty);
    }
    Subj->Subject_Name = strfile;
    strfile = "";
    fsubj.close();
}
cout << "Введите оценку по данному предмету" << endl;
getline(cin, Mark);
if (isValidMark(Mark)) {
    Mark_ST = stoi(Mark);
    Subj->Subject_mark = Mark_ST;
}
else {
    cout << "Ошибка ввода данных. Откат заполнения."; flag = 0;
    Sleep(1000); break;
}
STUD_STR += Subj->Subject_Name;
STUD_STR += '.';
STUD_STR += to_string(Subj->Subject_mark);
STUD_STR += '+';
}
delete Subj;
STUD_STR += 'I';
}
STUD_STR += '#';
file << STUD_STR << endl;
file.close();
I_MENU = 0; return 0;
}
};
cls; return 0;
};

```

```

int Menu_Changing_Info(int STD_NUM) {
    string MENU_TEMP; unsigned short MENU; string TEMP;
    Student* STUD_INFO = ReturnInfo(STD_NUM); int I_MENU = -1;
    while (I_MENU != 0) {
        cout << "Выберите пункт необходимый к перезаписи" << endl;
        cout << "1. Перезапись имени" << endl;
        cout << "2. Перезапись фамилии" << endl;
        cout << "3. Перезапись отчества" << endl;
        cout << "4. Перезапись номера зачётной книжки" << endl;
        cout << "5. Перезапись года поступления в ВУЗ" << endl;
        cout << "6. Перезапись пола студента" << endl;
        cout << "7. Перезапись названия института" << endl;
        cout << "8. Перезапись названия кафедры" << endl;
        cout << "9. Перезапись названия группы" << endl;
        cout << "10. Перезапись результатов сессии" << endl;
        cout << "11. Перезапись даты рождения студента" << endl;
        cout << "0. Возврат к списку студентов и применение изменений" << endl;
        cout << "Операция взаимодействия: ";
        getline(cin, MENU_TEMP);
        if (isValidInt(MENU_TEMP)) {
            MENU = stoi(MENU_TEMP);
            switch (MENU) {
                case 1: {

```

```

cout << "Выбрана перезапись имени студента. Текущее значение пункта: " << STUD_INFO->GetFirstName() << endl;
cout << "Введите новое значение имени студента: "; getline(cin, TEMP);
if (isValidNamePatronym(TEMP)) {
    STUD_INFO->SetFirstName(TEMP);
}
break;
}
case 2: {
    cout << "Выбрана перезапись фамилии студента. Текущее значение пункта: " << STUD_INFO->GetSurname() << endl;
    cout << "Введите новое значение имени студента: "; getline(cin, TEMP);
    if (isValidSurName(TEMP)) {
        STUD_INFO->SetSurname(TEMP);
    }
    break;
}
case 3: {
    cout << "Выбрана перезапись отчества студента. Текущее значение пункта: " << STUD_INFO->GetPronymic() << endl;
    cout << "Введите новое значение отчества студента: "; getline(cin, TEMP);
    if (isValidNamePatronym(TEMP)) {
        STUD_INFO->SetPatronym(TEMP);
    }
}
case 4: {
    cout << "Выбрана перезапись номера зачётной книжки. Текущее значение пункта: " << STUD_INFO->GetSTUDUID() << endl;
    cout << "Введите новое значение зачётной книжки: "; getline(cin, TEMP);
    if (isValidZach(TEMP)) {
        STUD_INFO->SetSTUDUID(TEMP);
    }
    break;
}
case 5: {
    cout << "Выбрана перезапись года поступления в ВУЗ: " << STUD_INFO->GetYearEntrance() << endl;
    cout << "Введите новое значение года поступления: "; getline(cin, TEMP);
    if (isValidInt(TEMP)) {
        unsigned short YEAR = stoi(TEMP);
        if (isValidEntrance(YEAR)) {
            STUD_INFO->SetYearEntrance(YEAR);
        }
        break;
    }
}
case 6: {
    cout << "Выбрана перезапись пола студента. Текущее значение: " << STUD_INFO->GetSex() << endl;
    cout << "Введите новое значение пола: "; getline(cin, TEMP);
    if (isValidSex(TEMP)) {
        STUD_INFO->SetSex(TEMP);
    }
    break;
}
case 7: {
    cout << "Выбрана перезапись наименования института. Текущее название: " << STUD_INFO->GetDepartment() << endl;
    cout << "Введите новое наименование института: "; getline(cin, TEMP);
    if (isValidFaculty(TEMP)) {
        STUD_INFO->SetDepartment(TEMP);
    }
    break;
}
case 8: {
    cout << "Выбрана перезапись наименования выпускающей Кафедры. Текущее значение: " << STUD_INFO->GetFaculty() << endl;
    cout << "Введите новое название выпускающей Кафедры: "; getline(cin, TEMP);
    if (isValidKaf(TEMP)) {
        STUD_INFO->SetFaculty(TEMP);
    }
    break;
}
case 9: {
    cout << "Выбрана перезапись наименование группы студента. Текущее значение: " << STUD_INFO->GetGroup() << endl;
    cout << "Введите новое наименование группы обучения: "; getline(cin, TEMP);
    if (isValidGroup(TEMP)) {
        STUD_INFO->SetGroup(TEMP);
    }
    break;
}
case 10: {
    unsigned short sem = (2023 - STUD_INFO->GetYearEntrance()) * 2 - 1; int SEM, SUBJ_INT, SUBJ_UID;

```

```

cout << "Введите номер семестра, в котором находится предмет, который хотите изменить: "; getline(cin, TEMP);
if (isValidInt(TEMP)) {
    SEM = stoi(TEMP); string strfile, empty = "";
    if ((SEM <= sem) && (SEM > 0)) {
        cout << "Введите номер предмета, который хотите изменить: "; getline(cin, TEMP);
        if (isValidInt(TEMP)) {
            SUBJ_INT = stoi(TEMP);
            if ((SUBJ_INT >= STUD_INFO->Grades->SUBJ_NUM) && (SUBJ_INT <1)) {
                cout << "Ошибка ввода"; break;
            }
            cout << "Вы пытаетесь изменить " << SEM << " семестр, предмет: " << STUD_INFO->Grades[SEM - 1].Sem[SUBJ_INT-
1].Subject_Name << ", оценка за Семестр: " << STUD_INFO->Grades[SEM - 1].Sem[SUBJ_INT-1].Subject_mark << endl;
            cout << "Введите ID нового предмета (или того же, в случае лишь замены оценки): "; getline(cin, TEMP);
            if (isValidInt(TEMP)) {
                int UID = stoi(TEMP);
                TEMP += '+';
                int temp = -1;
                fstream fsubj;
                fsubj.open("subject_DB.txt", ios_base::out || ios_base::app || ios_base::in);
                while (!fsubj.eof()) {
                    fsubj >> strfile;
                    int temp = strfile.find(TEMP);
                    if (temp == 0) break;
                }
                if (UID < 10) {
                    strfile.replace(0, 2, empty);
                    strfile.replace(strfile.length() - 1, 1, empty);
                }
                if ((UID >= 10) && (UID < 100)) {
                    strfile.replace(0, 3, empty);
                    strfile.replace(strfile.length() - 1, 1, empty);
                }
                if (UID >= 100) {
                    strfile.replace(0, 4, empty);
                    strfile.replace(strfile.length() - 1, 1, empty);
                }
                STUD_INFO->Grades[SEM - 1].Sem[SUBJ_INT - 1].Subject_Name = strfile;
                cout << "Введите значение оценки: "; getline(cin, TEMP);
                if (isValidMark(TEMP)) {
                    int Mark = stoi(TEMP);
                    STUD_INFO->Grades[SEM - 1].Sem[SUBJ_INT - 1].Subject_mark = Mark;
                }
            }
        }
    }
    break;
}
case 11: {
    string TEMP2, TEMP3; unsigned short d, m, y, count = 0;
    cout << "Выбрана перезапись даты рождения студента. Текущая дата: " << STUD_INFO->GetBirthDateString() << endl;
    cout << "Введите новый день рождения: "; getline(cin, TEMP);
    cout << "Введите новый месяц рождения: "; getline(cin, TEMP2);
    cout << "Введите новый год рождения: "; getline(cin, TEMP3);
    if (isValidInt(TEMP)) {
        d = stoi(TEMP);
        count++;
    }
    if (isValidInt(TEMP2)) {
        m = stoi(TEMP2);
        count++;
    }
    if (isValidInt(TEMP3)) {
        y = stoi(TEMP3);
        count++;
    }
    if (count == 3) {
        if (isValidBirth(d, m, y)) {
            MyDate* S = new MyDate;
            S->D = d; S->M = m; S->Y = y;
            STUD_INFO->SetBirthDate(S);
            delete S;
        }
        else {
            cout << "Недопустимая дата" << endl;
        }
    }
}

```

```

    }
    break;
}
case 0: {
    cout << "Перезапись информации. Возрат к списку студентов";
    I_MENU = 0;
    ofstream TempF("temp.txt");
    fstream file;
    string tempstring, tempuid; unsigned short tempU;
    file.open("1.txt", ios_base::in | ios_base::out | ios_base::app);
    int n = Students_Count();
    for (int i = 0; i < n; i++) {
        file >> tempstring;
        tempuid = tempstring;
        tempuid.erase(tempuid.find('+'), tempuid.length());
        tempU = stoi(tempstring);
        if (tempU != STD_NUM) {
            TempF << tempstring << endl;
        }
        if (tempU == STD_NUM) {
            TempF << STD_NUM << '+' << STUD_INFO->GetFirstName() << '!' << STUD_INFO->GetSurname() << '!' << STUD_INFO->GetPTronymic() << '!' << STUD_INFO->GetDepartment() << '!' << STUD_INFO->GetFaculty() << '!' << STUD_INFO->GetGroup() << '!' << STUD_INFO->GetBirthDateString() << '!' << STUD_INFO->GetYearEntrance() << '>' << STUD_INFO->GetSTUDUID() << '<' << STUD_INFO->GetSex() << '!' << '!' << STUD_INFO->GetYearEntrance() * 2; j++) {
                for (int k = 0; k < 10; k++) {
                    TempF << STUD_INFO->Grades[j - 1].Sem[k].Subject_Name << '!' << STUD_INFO->Grades[j - 1].Sem[k].Subject_mark << '+' << '!' << ']' << endl;
                }
            }
            TempF << '#' << endl;
        }
    }
    file.close();
    TempF.close();
    remove("1.txt");
    ofstream FileS("1.txt");
    fstream TempFileS, filetemp;
    TempFileS.open("temp.txt", ios_base::in | ios_base::out | ios_base::app);
    for (int i = 0; i < n; i++) {
        TempFileS >> tempstring;
        FileS << tempstring << endl;
    }
    cls;
    I_MENU = 0;
    break;
}
default: {
    cout << "В результате ошибки произошёл откат";
    Sleep(1000); cls; }
}
else {
    cout << "Ошибка в вводе"; Sleep(1000); cls; break;
}
}
return 0;
};

int Menu_Delete_User(int STD_NUM) {
    fstream file; int count = Students_Count();
    file.open("1.txt", ios_base::in | ios_base::out | ios_base::app);
    string STUD_UID, STUD_string;
    ofstream temp("temp.txt");
    while (!file.eof()) {
        file >> STUD_string;
        STUD_UID = STUD_string; STUD_UID.erase(STUD_UID.find('+'), STUD_UID.length());
        if (stoi(STUD_UID) != STD_NUM) {
            temp << STUD_string << endl;
        }
    }
    temp.close(); file.close();
    remove("1.txt");
    ofstream Myfile ("1.txt");

```



```

fstream temp1("temp.txt", ios_base::in | ios_base::out | ios_base::app);
for (int i = 0; i < count-1; i++) {
    temp1 >> STUD_string;
    Myfile << STUD_string << endl;
}
Myfile.close(); temp1.close();
remove("temp.txt");
return 0;
}

int isValidIndTask(string S) {
    if (S == "" || S.length() != 1)
        return 0;
    S.append("a");
    if (S.find_first_not_of("12340") != S.length() - 1)
        return 0;
    return 1;
}

class Node {
    Student STUD_INFO;
    class Node* next;
    friend class List;
};

class List {
    Node* head;
    int count = 0;
    Node* PrevSTUD(Node*);
public:
    List() { head = NULL; }
    bool EmptyOrNot() { return head == NULL; }
    int GetNumNode() { return count; }
    Student ReturnNode(Node* N) {
        return N->STUD_INFO;
    }
    void SetNode(Node* N, Student STUD) {
        N->STUD_INFO = STUD;
    }
    Node* NextSTUD(Node*);
    Node* AddSTUD(Student, Node*);
    void Print(char S);
    void Swap(Node*, Node*);
    Node* getFirst() { return head; }
    void Sort();
    ~List() {
        class Node* old = NULL;
        class Node* cur = head;
        while (cur != NULL) {
            old = cur;
            cur = cur->next;
            delete old;
        }
    }
};

void List::Sort() {
    Node* N = getFirst();
    bool Flag = 0;
    while (Flag == 0) {
        Flag = 1;
        while (N->next != NULL) {
            Student* STUD_F = new Student(N->STUD_INFO);
            Student* STUD_S = new Student(N->next->STUD_INFO);
            if (STUD_F->GetYearEntrance() > STUD_S->GetYearEntrance()) {
                Flag = 0;
                Swap(N, N->next);
                N = NextSTUD(N);
            }
            else {
                N = NextSTUD(N);
            }
        }
        N = getFirst();
    }
}

```

```

}
Node* List::AddSTUD(Student STUD, Node* node = NULL) {
    Node* newpart = new Node();
    newpart->STUD_INFO = STUD;
    count++;
    if (node == NULL) {
        if (head == NULL) {
            newpart->next = NULL;
            head = newpart;
        }
        else {
            newpart->next = head;
            head = newpart;
        }
        return newpart;
    }
    newpart->next = node->next;
    node->next = newpart;
    return newpart;
};

Node* List::NextSTUD(Node* Node) {
    if (EmptyOrNot()) return NULL;
    return Node->next;
};

Node* List::PrevSTUD(Node* N) {
    if (EmptyOrNot()) return NULL;
    if (N == head) return NULL;
    Node* P = head;
    while (P->next != N)
        P = P->next;
    return P;
};

void List::Print(char S) {
    if (EmptyOrNot()) {
        return;
    }
    Node* N = head;
    do {
        Student STUD = ReturnNode(N);
        if (S == 'M') {
            if (STUD.GetSex() == "М") {
                cout << STUD.UID << " Фамилия студента: " << STUD.GetSurname() << endl;
                cout << "Имя студента: " << STUD.GetFirstName() << endl;
                cout << "Отчество студента: " << STUD.GetPronymic() << endl;
                cout << "Год поступления студента в ВУЗ: " << STUD.GetYearEntrance() << ". Пол студента: " << STUD.GetSex() << endl;
                cout << "-----" << endl;
            }
        }
        if (S == 'Ж') {
            if (STUD.GetSex() == "Ж") {
                cout << STUD.UID << " Фамилия студента: " << STUD.GetSurname() << endl;
                cout << "Имя студента: " << STUD.GetFirstName() << endl;
                cout << "Отчество студента: " << STUD.GetPronymic() << endl;
                cout << "Год поступления студента в ВУЗ: " << STUD.GetYearEntrance() << ". Пол студента: " << STUD.GetSex() << endl;
                cout << "-----" << endl;
            }
        }
        if (S == 'H') {
            cout << " Фамилия студента: " << STUD.GetSurname() << endl;
            cout << "Имя студента: " << STUD.GetFirstName() << endl;
            cout << "Отчество студента: " << STUD.GetPronymic() << endl;
            cout << "Год поступления студента в ВУЗ: " << STUD.GetYearEntrance() << ". Пол студента: " << STUD.GetSex() << endl;
            cout << "-----" << endl;
        }
        N = NextSTUD(N);
    } while (N != NULL);
};

void List::Swap(Node* node1, Node* node2)
{
    if (node1 == NULL || node2 == NULL) return;
    if (node1 == node2) return;

```

```

if (node2->next == node1) {
    Node* p = node1;
    node1 = node2;
    node2 = p;
}
Node* prev1 = PrevSTUD(node1);
Node* prev2 = PrevSTUD(node2);
Node* next1 = NextSTUD(node1);
Node* next2 = NextSTUD(node2);
if (next1 == node2)
{
    if (prev1 != NULL)
        prev1->next = node2;
    else
        head = node2;
    node2->next = node1;
    node1->next = next2;
    return;
}
if (prev1 != NULL)
    prev1->next = node2;
else
    head = node2;
if (prev2 != NULL)
    prev2->next = node1;
else
    head = node1;
node2->next = next1;
node1->next = next2;
}

int IndividualTask() {
    List Group;
    fstream file;
    file.open("1.txt", ios_base::in | ios_base::out | ios_base::app);
    int n = Students_Count();
    Student** Student_Group;
    string tempUID; unsigned short UID_T;
    Student_Group = (Student**) new Student * [n];
    for (int i = 0; i < n; i++) {
        file >> tempUID;
        tempUID.erase(tempUID.find('+'), tempUID.length());
        UID_T = stoi(tempUID);
        Student_Group[i] = (Student*) new Student;
        Student_Group[i] = ReturnInfo(UID_T);
    }
    for (int i = 0; i < n; i++) {
        Group.AddSTUD(*Student_Group[i]);
    }
    string MENU; int I_MENU = -1;
    while (I_MENU != 0) {
        cout << "Создан линейный список из предоставленных студентов. Выберите одну из следующих опций" << endl;
        cout << "1. Отобразить список из студентов Мужского пола" << endl;
        cout << "2. Отобразить список студентов Женского пола" << endl;
        cout << "3. Отобразить полный список студентов" << endl;
        cout << "4. Отсортировать список студентов по году поступления в ВУЗ" << endl;
        cout << "0. Покинуть раздел индивидуального задания" << endl;
        cout << "Введите операцию: "; getline(cin, MENU);
        if (IsValidIndTask(MENU)) {
            I_MENU = stoi(MENU);
            switch (I_MENU) {
                case 1: {Group.Print('M'); system("pause"); cls; break; };
                case 2: {Group.Print('Ж'); system("pause"); cls; break; };
                case 3: {Group.Print('H'); system("pause"); cls; break; };
                case 4: {Group.Sort(); system("pause"); cls; break; };
                default: {I_MENU = 0; break; };
            }
        }
    }
    cls;
    return -1;
}

void Crypt()
{
    system("openssl\\bin\\openssl.exe rand -hex -out key.txt 64");
}

```

```

system("openssl\\bin\\openssl.exe enc -aes-256-cbc -pbkdf2 -pass file:key.txt -in 1.txt -out 1.txt.enc");
if (remove("1.txt") != 0) {
    cout << "Ошибка удаления файла базы данных." << endl;
}
system("openssl\\bin\\openssl.exe rsautl -encrypt -pubin -inkey rsa.public -in key.txt -out key.txt.enc");
if (remove("key.txt") != 0) {
    cout << "Ошибка удаления файла базы пароля базы данных." << endl;
}
}

void Decrypt()
{
    system("openssl\\bin\\openssl.exe rsautl -decrypt -inkey rsa.private -in key.txt.enc -out key.txt");
    if (remove("key.txt.enc") != 0) {
        cout << "Ошибка удаления зашифрованного файла с паролем." << endl;
    }
    system("openssl\\bin\\openssl.exe enc -aes-256-cbc -d -pbkdf2 -pass file:key.txt -in 1.txt.enc -out 1.txt");
    if (remove("key.txt") != 0) {
        cout << "Ошибка удаления файла пароля." << endl;
    }
    if (remove("1.txt.enc") != 0) {
        cout << "Ошибка удаления зашифрованной базы данных." << endl;
    }
}

int main()
{
    rus();
    Decrypt();
    cout << "Дешифровка" << endl;
    for (; Menu_Start() != 0;) {
        Menu_Start();
    }
    Crypt();
    cout << "Шифровка" << endl;
}

```