

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА – Российский технологический университет»  
РТУ МИРЭА**

**Институт кибербезопасности и цифровых технологий**

*(наименование института, филиала)*

**Кафедра КБ-2 «Прикладные информационные технологии»**

*(наименование кафедры)*

**КУРСОВАЯ РАБОТА**

*(указать вид работы)*

**«ЯЗЫКИ ПРОГРАММИРОВАНИЯ»**

*(указать дисциплину в соответствии с учебным планом)*

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

**Студент** 1 **курса, учебной группы** БББО-01-21    
Мысливец Леонид Владимирович

Руководитель курсовой работы:

|  |  |  |
| --- | --- | --- |
| Серов В.А. | доцент, к.т.н. |  |
| *(Фамилия И.О.)* | *(Должность, звание, ученая степень)* | *(Подпись руководителя)* |

Рецензент (при наличии):

|  |  |  |
| --- | --- | --- |
|  |  |  |
| *(Фамилия И.О.)* | *(Должность, звание, ученая степень)* | *(Подпись руководителя)* |

Работа представлена к защите:

|  |  |  |
| --- | --- | --- |
| « » июня 2022 г. |  | ( Серов В.А., доцент ) |
|  | *Подпись* | *Расшифровка, должность* |

Допущен к защите:

|  |  |  |
| --- | --- | --- |
| « » июня 2022 г. |  | ( Серов В.А., доцент ) |
|  | *Подпись* | *Расшифровка, должность* |



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА – Российский технологический университет»  
РТУ МИРЭА**

**Институт кибербезопасности и цифровых технологий**

*(наименование института, филиала)*

**Кафедра КБ-2 «Прикладные информационные технологии»**

*(наименование кафедры)*

**УТВЕРЖДАЮ**Заведующий кафедрой КБ-2

  / Трубиенко О.В.

« » июня 2022 г.

**ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ**

*(указать вид работы)*

**«ЯЗЫКИ ПРОГРАММИРОВАНИЯ»**

*(указать дисциплину в соответствии с учебным планом)*

**Студент** 1 **курса, учебной группы** БББО-01-21    
Мысливец Леонид Владимирович

*(фамилия, имя и отчество)*

**1. Тема:** Разработка программы информационного поиска студентов по заданным критериям с возможностью шифрования данных.

**2. Исходные данные:** Ф.И.О. студента, число, месяц, год рождения, год поступления в институт, факультет (институт), кафедра, группа, номер зачетной книжки, названия предметов и оценки по каждому предмету в каждой сессии; конкретное задание, вариант которого определяется 2-мя последними цифрами зачетной книжки (шифр).

**3. Перечень вопросов, подлежащих разработке, и обязательного графического материала:** массив, структура, файл, функции, классы, объекты, инкапсуляция, полиморфизм и другие элементы ООП.

Срок представления к защите курсового проекта (работы) до «15» мая 2022 г.

Задание на выполнение курсовой работы выдал:

|  |  |  |
| --- | --- | --- |
| « » 2022 г. |  | ( Серов В.А. ) |
|  | *Подпись* | *ФИО* |

Задание на курсовой проект получил:

|  |  |  |
| --- | --- | --- |
| « » 2022 г. |  | ( Мысливец Л.В. ) |
|  | *Подпись* | *ФИО* |

СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 1](#_Toc105370126)

[ВВЕДЕНИЕ 2](#_Toc105370127)

[1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 3](#_Toc105370128)

[1.1 Структурное программирование 3](#_Toc105370129)

[1.2 Процедурное программирование 3](#_Toc105370130)

[1.3 Модульное программирование 4](#_Toc105370131)

[1.4 Ссылки 5](#_Toc105370132)

[1.5 Работа с памятью. Указатели 5](#_Toc105370133)

[1.6 Работа с файлами 6](#_Toc105370134)

[1.7 Динамические структуры данных 7](#_Toc105370135)

[1.8 Линейный список 7](#_Toc105370136)

[1.9 Функции 7](#_Toc105370137)

[1.10 Объектно-ориентированное программирование 8](#_Toc105370138)

[1.10.1 Элементы ООП 9](#_Toc105370139)

[1.11 Конструкторы и деструкторы 9](#_Toc105370140)

[1.12 Классы и дружественные функции 10](#_Toc105370141)

[1.12.1 Описание классов и их методов 11](#_Toc105370142)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 12](#_Toc105370143)

АНОТАЦИЯ

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

Выполнил: Мысливец Леонид Владимирович, БББО-01-21

Руководитель: Серов В.А., доцент, к.т.н.

Работа состоит из введения, двух глав, заключения и списка используемых источников.

Работа включает в себя 66 страниц, 22 рисунка, использовано 5 источника, содержит 2 приложения, исходный код программы содержит 1600 строк, написанных самостоятельно.

Во введении рассмотрены цели, задачи работы и ее актуальность.

В первой главе, относящейся к теоретической части, рассмотрены и приведены основные термины, понятия и определения из языка программирования С++.

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 заключении приведены основные выводы, полученные в хотя выполнения работы.

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

ВВЕДЕНИЕ

Задачей курсовой работы является: Разбить группу на 3 части, с поиском среди лиц определенного пола: отличников, хорошистов, троечников за все время обучения.

Цель курсовой работы по дисциплине «Языки программирования» состоит в закреплении и углублении знаний и навыков, полученных при изучении дисциплины. Курсовая работа предполагает выполнение задания повышенной сложности по проектированию, разработке и тестированию программного обеспечения, а также оформлению сопутствующей документации.

В результате изучения дисциплины обучающийся должен:

Знать:

1) языки, системы и инструментальные средства программирования в профессиональной деятельности;

2) математический аппарат, математически 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) проанализировать исходные данные, указанные в задании;

2) определить данные, структуры, классы, методы и функции, необходимые для выполнения работы согласно варианту;

3) разработать соответствующей алгоритм решения конкретной задачи;

4) реализовать элементы, описанные в пункте 2;

5) подготовить контрольные данные для тестирования программного обеспечения;

6) отладить разработанное программное обеспечение на основе контрольных данных, подготовленных в предыдущем пункте.

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Структурное программирование

Структурное программирование воплощает принципы системного подхода в процессе создания и эксплуатации программного обеспечения ЭВМ. В основу структурного программирования положены следующие достаточно простые положения:

1) алгоритм и программа должны составляться поэтапно;

2) сложная задача должна разбиваться на достаточно простые части, каждая из которых имеет один вход и один выход;

3) логика алгоритма и программы должна опираться на минимальное число достаточно простых базовых управляющих структур.

Принципы структурного программирования:

1) программа построена из трёх базовых управляющих конструкций: последовательность, ветвление, цикл;

2) в программе базовые управляющие конструкции вложены друг в друга произвольным образом;

3) повторяющиеся фрагменты программы были оформлены в виде подпрограмм (процедур и функций);

4) перечисленные конструкции должны имеют один вход и один выход;

5) разработка программы ведётся пошагово, методом «сверху вниз».

1.2 Процедурное программирование

Процедурное программирование — программирование на императивном языке, при котором последовательно выполняемые операторы можно собрать в подпрограммы, то есть более крупные целостные единицы кода, с помощью механизмов самого языка. Процедурное программирование является отражением архитектуры традиционных ЭВМ.

Особенности процедурного программирования:

1) предопределенные функции. Предопределенная функция — это инструкция, идентифицируемая именем. Обычно предопределенные функции встроены в языки программирования более высокого уровня, но они получены из библиотеки или реестра, а не из программы;

2) локальная переменная. Локальная переменная — это переменная, которая объявлена, в основной структуре метода и ограничена локальной областью действия, которую она задает. Локальная переменная может использоваться только в том методе, в котором она определена;

3) глобальная переменная. Глобальная переменная — это переменная, которая объявляется вне любой другой функции, определенной в коде. Благодаря этому глобальные переменные могут использоваться во всех функциях, в отличие от локальной переменной;

4) модульность: Модульность — это когда две разные системы имеют под рукой две разные задачи, но сгруппированы вместе, чтобы сначала выполнить более крупную задачу. В этом случае каждая группа систем будет выполнять свои собственные задачи один за другим, пока все задачи не будут выполнены;

5) передача параметров: Передача параметров — это механизм, используемый для передачи параметров в функции, подпрограммы или процедуры.

1.3 Модульное программирование

Модуль в программировании — это фрагмент кода, имеющий определенное функциональное значение и характеризующийся логической завершенностью.

Модульное программирование — это способ создания программы посредством объединения модулей в единую структуру.

В основе модульного программирования лежат три основных концепции:

Основные концепции модульного программирования:

1) каждый модуль имеет единственную точку входа и выхода;

2) размер модуля по возможности должен быть минимизирован;

3) вся система построена из модулей;

4) каждый модуль не зависит от того, как реализованы другие модули.

Использование модульного программирования позволяет упростить тестирование программы и обнаружение ошибок. Аппаратно-зависимые подзадачи могут быть строго отделены от других подзадач, что улучшает мобильность создаваемых программ.

1.4 Ссылки

Ссылка — это псевдоним для другой переменной. Они объявляются при помощи символа &. Ссылки должны быть проинициализированы при объявлении, причем только 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.

Поскольку ссылка — это псевдоним, то при передаче объекта в функцию по ссылке внутри нее объект можно изменять. Ссылки не могут ссылаться на другие ссылки или на поле битов. Не может быть массивов ссылок или указателей на ссылку. Ссылка может использоваться для возврата результата из функции. Возвратить результат по ссылке — значит возвратить не указатель на объект и не его значение, а сам этот объект.

1.5 Работа с памятью. Указатели

Указатели являются одним из основных понятий языка Си. В такие переменные можно записывать адреса любых участков памяти, на чаще всего – адрес начального элемента динамического массива.

Важно знать, что:

1) указатель – это переменная, в которой записан адрес другой переменной;

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

3) знак & перед именем переменной обозначает ее адрес;

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

5) нельзя записывать по указателю, который указывает непонятно куда – это вызывает сбой программы, поскольку что-то стирается в памяти;

6) для обозначения недействительного указателя используется константа NULL;

7) при изменении значения указателя на n он в самом деле сдвигается к n-ому следующему числу данного типа, то есть для указателей на целые числа на n\*sizeof(int) байт;

8) указатель печатаются по формату %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

Часто в серьезных программах надо использовать данные, размер и структура которых должны меняться в процессе работы. Динамические массивы здесь не выручают, поскольку заранее нельзя сказать, сколько памяти надо выделить – это выясняется только в процессе работы. Например, надо проанализировать текст и определить, какие слова и в каком количество в нем встречаются, причем эти слова нужно расставить по алфавиту. В таких случаях применяют данные особой структуры, которые представляют собой отдельные элементы, связанные с помощью ссылок.

1.8 Линейный список

В простейшем случае каждый узел содержит всего одну ссылку. Для определенности будем считать, что решается задача частотного анализа текста – определения всех слов, встречающихся в тексте и их количества. В этом случае область данных элемента включает строку (длиной не более 40 символов) и целое число.

1.9 Функции

Функция определяет действия, которые выполняет программа. Функции позволяют выделить набор инструкций и придать ему имя. А затем многократно по присвоенному имени вызывать в различных частях программы. По сути функция — это именованный блок кода.

Функцию можно вызвать из любого количества мест в программе. Значения, передаваемые функции, являются аргументами, типы которых должны быть совместимы с типами параметров в определении функции.

Длина функции практически не ограничена, однако для максимальной эффективности кода целесообразно использовать функции, каждая из которых выполняет одиночную, четко определенную задачу. Сложные алгоритмы лучше разбивать на более короткие и простые для понимания функции, если это возможно. Функции могут быть перегружены, что означает, что разные версии функции могут использовать одно и то же имя, если они отличаются числом и типом формальных параметров.

Определение функции состоит из объявления и тела функции, заключенных в фигурные скобки, которые содержат объявления переменных, операторы и выражения. В следующем примере показано полное определение функции.

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.

Основные задачи ООП — структурировать код, повысить его читабельность и ускорить понимание логики программы. Косвенно выполняются и другие задачи: например, повышается безопасность кода и сокращается его дублирование.

Такой подход помогает строить сложные системы более просто и естественно благодаря тому, что вся предметная область разбивается на объекты и каждый из них слабо связан с другими объектами. Слабая связанность возникает вследствие соблюдения трех принципов: инкапсуляции, наследования и полиморфизма.

1) инкапсуляция – сокрытие поведения объекта внутри него. Объекту «водитель» не нужно знать, что происходит в объекте «машина», чтобы она ехала. Это ключевой принцип ООП;

2) наследование. Есть объекты «человек» и «водитель». У них есть явно что-то общее. Наследование позволяет выделить это общее в один объект (в данном случае более общим будет человек), а водителя — определить, как человека, но с дополнительными свойствами и/или поведением. Например, у водителя есть водительские права, а у человека их может не быть;

3) полиморфизм – это переопределение поведения. Можно снова рассмотреть «человека» и «водителя», но теперь добавить «пешехода». Человек умеет как-то передвигаться, но как именно, зависит от того, водитель он или пешеход. То есть у пешехода и водителя схожее поведение, но реализованное по-разному: один перемещается ногами, другой – на машине.

ООП позволяет упростить сложные объекты, составляя их из более маленьких и простых, поэтому над программой могут работать сотни разработчиков, каждый из которых занят своим блоком. Большинство современных языков программирования — объектно-ориентированные, и, однажды поняв суть, вы сможете освоить сразу несколько языков.

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

Класс представляет составной тип, который может использовать другие типы. Классы и объекты в С++ являются основными концепциями объектно-ориентированного программирования — ООП.

Класс предназначен для описания некоторого типа объектов. То есть класс является планом объекта. А объект представляет конкретное воплощение класса, его реализацию. 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.

Класс может определять переменные и константы для хранения состояния объекта и функции для определения поведения объекта.

На объекты классов, как и на объекты других типов, можно определять указатели. Затем через указатель можно обращаться к членам класса - переменным и методам. Однако если при обращении через обычную переменную используется символ точка, то для для обащения к членам класса через указатель применяется стрелка (->).

Дружественные функции — это функции, которые не являются членами класса, однако имеют доступ к его закрытым членам - переменным и функциям, которые имеют спецификатор private.

Для определения дружественных функций используется ключевое слово friend.

1.12.1 Описание классов и их методов

Классы в языке программировании С++ — 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.

Таблица 1.12.1.1 – Классы

|  |  |
| --- | --- |
| Класс | Назначение |
| Core | Служебный класс, содержащий в себе базовые функции, которые используют классы-наследники |
| Task | Отвечает за выполнение поставленного задания (Вариант) |
| Stud\_marks | Класс, отвечающий за оценки и названия предметов (Все сопутствующие функции для работы с табелем успеваемости) |
| Menu | Отвечает за меню взаимодействия пользователя с программой |
| Student | Хранит персональные данные студента  (ФИО, дата рождения, пол и т.д.) |

1.13 Методы шифрования и дешифрования

Шифрование – это процесс кодирования информации с целью предотвращения несанкционированного доступа.

Дешифрование – процесс, обратный процессу шифрованию.

OpenSSL – криптографическая библиотека с открытым исходным кодом. Библиотека поддерживает почти все низкоуровневые алгоритмы хеширования и шифрования, а также реализует большинство популярных криптографических стандартов, в том числе: позволяет создавать ключи RSA, DH, DSA, шифровать данные.

Для начала реализуем функцию Crypt, для этого поэтапно реализуем следующее:

1) генерацию рандомного пароля;

2) шифрование алгоритмом AES базы данных;

3) удаление не зашифрованной базы данных;

4) запись пароля в файл;

5) шифрование алгоритмом RSA файла с паролем с помощью публичного ключа;

6) удаление не зашифрованного файла с паролем.

Листинг 1.13.1

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | srand(time(NULL));  char\* pass = new char[64];  for (int i = 0; i < 64; ++i) {  switch (rand() % 3) {  case 0:  pass[i] = rand() % 10 + '0';  break;  case 1:  pass[i] = rand() % 26 + 'A';  break;  case 2:  pass[i] = rand() % 26 + 'a';  }  } |

Листинг 1.13.2

|  |  |
| --- | --- |
| 1  -  2  3 | string command = "openssl\\bin\\openssl.exe enc -aes-256-cbc -salt -in file.txt -out file.txt.enc -pass pass:";  command += pass;  system(command.c\_str()); |

Листинг 1.13.3

|  |  |
| --- | --- |
| 1  2  3 | if (remove("file.txt") != 0) {  cout << "[ERROR] - deleting file" << endl;  } |

Листинг 1.13.4

|  |  |
| --- | --- |
| 1  2  3  4 | ofstream file;  file.open("key.txt", ios::binary);  file.write(pass, 65);  file.close(); |

Листинг 1.13.5

|  |  |
| --- | --- |
| 1  -  2 | command = "openssl\\bin\\openssl.exe rsautl -encrypt -inkey rsa.public -pubin -in key.txt -out key.txt.enc";  system(command.c\_str()); |

Листинг 1.13.6

|  |  |
| --- | --- |
| 1  2  3 | if (remove("key.txt") != 0) {  cout << "[ERROR] - deleting file" << endl;  } |

Для реализации функции Decrypt поэтапно реализуем следующее:

1) Расшифровка зашифрованного файла с паролем с помощью приватного ключа и RSA;

2) Удаление зашифрованного файла с ключом;

3) Считывание ключа из файла;

4) Удаление файла с ключом;

5) Расшифровка зашифрованной базы данных с помощью ключа и алгоритма AES;

6) Удаление зашифрованной базы данных.

Листинг 1.13.7

|  |  |
| --- | --- |
| 1  -  2 | string command = "openssl\\bin\\openssl.exe rsautl -decrypt -inkey rsa.private -in key.txt.enc -out key.txt";  system(command.c\_str()); |

Листинг 1.13.8

|  |  |
| --- | --- |
| 1  2  3 | if (remove("key.txt.enc") != 0) {  cout << "[ERROR] - deleting file" << endl;  } |

Листинг 1.13.9

|  |  |
| --- | --- |
| 1  2  3  4  5 | char\* pass = new char[64];  ifstream file;  file.open("key.txt", ios::binary);  file.read(pass, 65);  file.close(); |

Листинг 1.13.10

|  |  |
| --- | --- |
| 1  2  3 | if (remove("key.txt") != 0) {  cout << "[ERROR] - deleting file" << endl;  } |

Листинг 1.13.11

|  |  |
| --- | --- |
| 1  -  2  3 | command = "openssl\\bin\\openssl.exe enc -aes-256-cbc -d -in file.txt.enc -out file.txt -pass pass:";  command += pass;  system(command.c\_str()); |

Листинг 1.13.12

|  |  |
| --- | --- |
| 1  2  3 | if (remove("file.txt.enc") != 0) {  cout << "[ERROR] - deleting file" << endl;  } |

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 | char[](массив символов) |
| Пол студента | char(символьный) |
| Число, месяц, год рождения | int(целочисленный) |
| Год поступления в институт | Int (целочисленный) |
|  | char[](массив символов) |
| Кафедра | char[](массив символов) |
| Группа | char[](массив символов) |
| Номер зачетной книжки | char[](массив символов) |
| Название предмета | char[](массив символов) |
| Оценка | char[](массив символов) |

По условиям курсовой работы, допустимо максимально 9 сессий и 10 предметов в каждом семестре, которые могут быть разными. Все данные являются форматными.

Для реализации соответствующей задачи были разработаны классы. В данных классах описаны методы решения поставленной задачи.

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

Таблица 2.1.2 – Классы

|  |  |
| --- | --- |
| Класс | Назначение |
| Core | Служебный класс, содержащий в себе базовые функции, которые используют классы-наследники |
| Task | Отвечает за выполнение поставленного задания (Вариант) |
| Stud\_marks | Класс, отвечающий за оценки и названия предметов (Все сопутствующие функции для работы с табелем успеваемости) |
| Menu | Отвечает за меню взаимодействия пользователя с программой |
| Student | Хранит персональные данные студента  (ФИО, дата рождения, пол и т.д.) |

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 |
| Core | int | val – целочисленное значение  result – переменная проверки корректного ввода |
| Task | int | count – счетчик  command-– команда введенная пользователем. |
| Stud\_marks | int | matan /phys /prac\_tise /inf\_tech /history /english – оценки по соответствующим предметам |
| Student | int | day/month/year – Дата рождения  year\_of\_receipt – Дата поступления |
| char\* | name – имя  surname – фамилия  patronymic – отчетство  faculty – факультет  department - кафедра  group – группа  gradebook\_number – номер студенческого билета |
| char | gender – пол студента |
| menu | int | command – команда введенная пользователем |

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 |
| Core | protection | Проверка корректного ввода команд пользователя | - | int |
| protect\_marks | Проверка корректного ввода для оценок | - | int |

Продолжение таблицы 2.1.4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Класс | Методы класса | | | |
| Название | Назначение | Аргументы  (их тип) | Тип возвращаемого значения |
| Core | protect\_change | Проверка корректного ввода для изменении информации о студенте | - | int |
| protect\_task | Проверка корректного ввода выполнения задачи | - | int |
| protect\_day  protect\_month  protect\_year | Проверка корректного ввода даты | - | int |
| protect\_ch | Проверка корректного ввода пола студента | - | char |
| cin\_cl | Очистка ввода (cin) | - | - |
| Task | task\_menu | Меню для работы с оценками студентов | - | - |
| find\_M  find\_F  find\_F\_or\_M | Проверка пола студента. | - | - |
| show\_stud | Дискриминация по оценкам |  |  |
| check\_marks | Проверка наличия оценок в файле | - | - |
| Student | Student | Конструктор класса | - | - |
| set | Ввод значений в поля класса | - | - |
| get | Вывод значений из поля класса | - | - |
| write | Запись студента в файл | - | - |
| ~Student | Деструктор класса | - | - |

Продолжение таблицы 2.1.4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Класс | Методы класса | | | |
| Название | Назначение | Аргументы  (их тип) | Тип возвращаемого значения |
| Stud\_marks | add\_mark | Ввод сессии (добавление студента в файл с оценками) | - | - |
| delete\_mark | Удаление из файла сессии студента по номеру билета | - | - |
| Change\_Marks | Изменение данных по сессии студента | - | - |
| Write\_Marks | Запись текущей сессии в файл | - | - |
| Show\_Marks | Вывести таблицу всей сессии на экран | - | - |
| check\_marks | Проверка на наличие оценок | - | - |
| Menu | welcome | Основное рабочее окно (меню) | - | - |
| Marks\_menu | Меню взаимодействия с сессией | - | - |
| Show\_Students | Вывести таблицу всех студентов из файла на экран | - | - |
| Delete\_Student | Удалить студента из файла | - | - |
| Change\_Student | Изменить информацию о студенте | - | - |
| Find\_Student | Найти студента по номеру студ. билета | - | - |

2.2 Алгоритм решения задачи

Наследование рассматривается как один из основополагающих принципов ООП. В соответствии с ним, классы могут использовать переменные и методы другого класса как свои собственные. Ниже показан рисунок, демонстрирующий наследование классов.

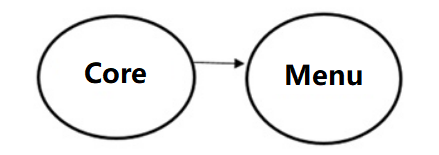


Рисунок 2.2.1 – Наследование классов

Одно из основных преимуществ ООП - инкапсуляция данных в клас-се. В C++ можно обойти правила инкапсуляции с помощью друзей. Друг класса – это функция или класс, не являющиеся членом этого класса, но имеющая доступ к его private и protected элементам. Своих друзей класс объявляет сам в любой зоне описания класса с помощью служебного слова friend.

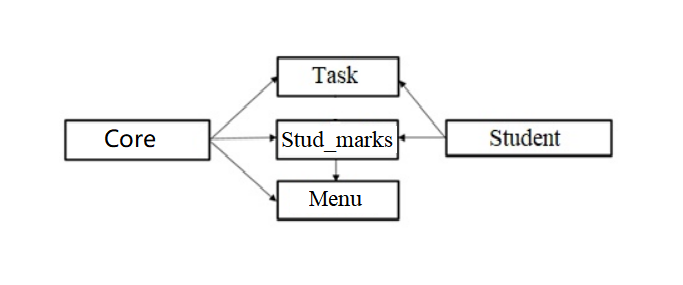


Рисунок 2.2.2 – Дружественные классы

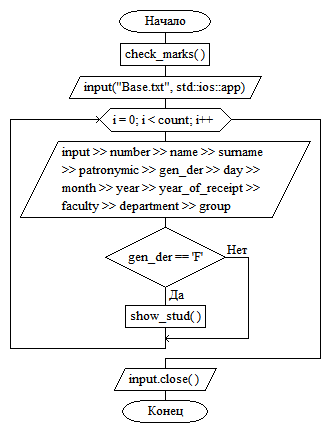


Рисунок 2.2.3 – Меню выбора пола

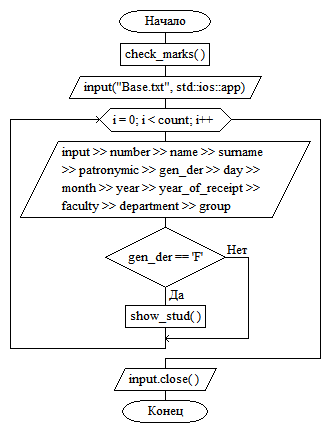


Рисунок 2.2.4 – Условие на проверку женского пола

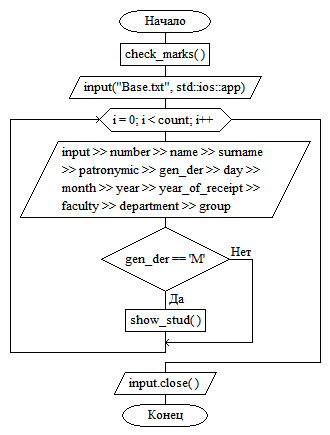


Рисунок 2.2.5 – Условие на проверку мужского пола

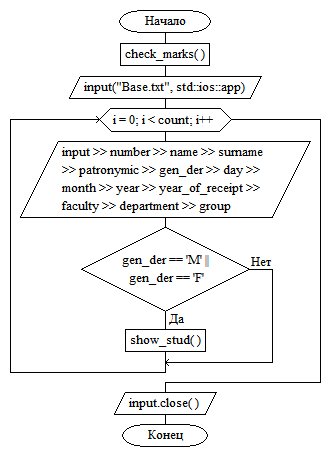


Рисунок 2.2.6 – Условие вывода мужского и женского пола

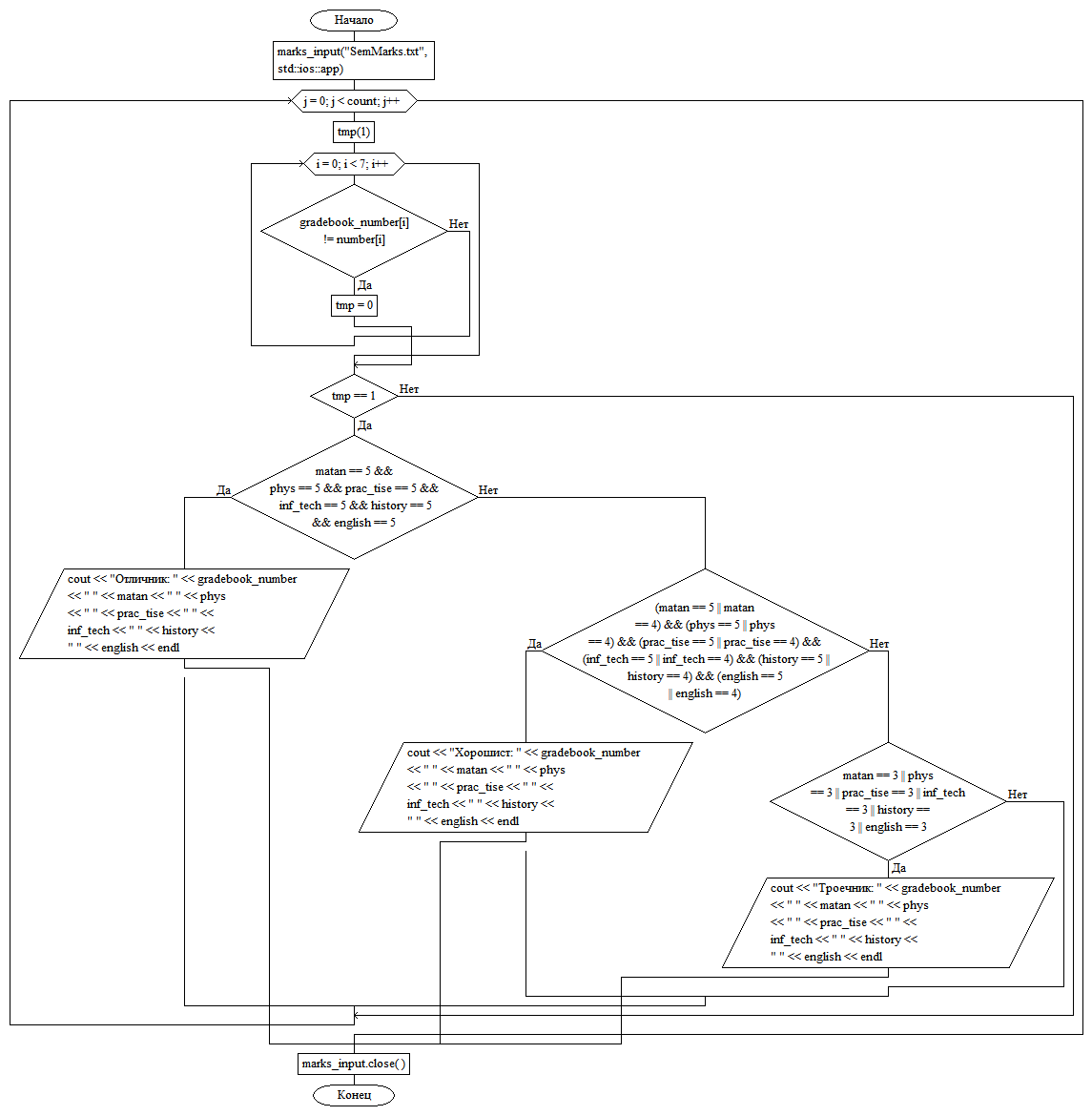


Рисунок 2.2.7 – Условие проверки учебного показателя (“Хорошист, Отличник, Троечник”)

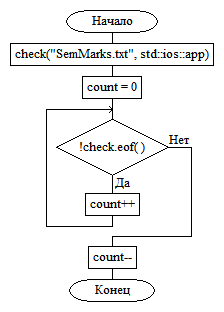


Рисунок 2.2.8 – Условие проверки количества оценок

2.3 Программная реализация задания

2.3.1 Чтение файла

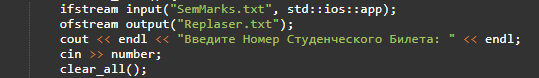


Рисунок 2.3.1.1 (Чтение файла)

2.3.2 Изменение данных

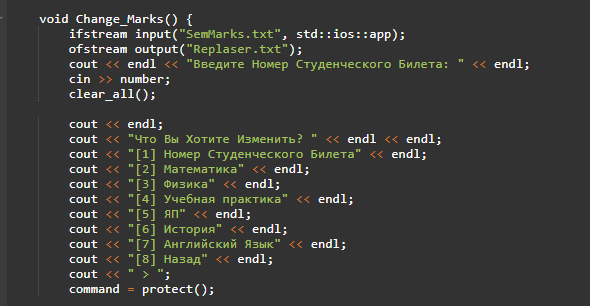


Рисунок 2.3.2.1 (Изменение данных 1)



Рисунок 2.3.2.2 (Изменение данных 2)

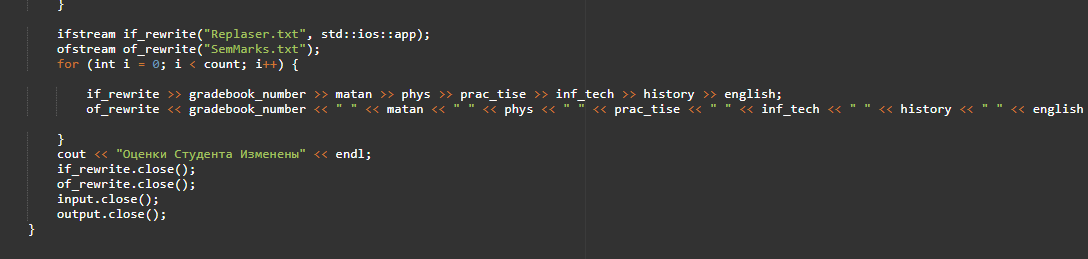


Рисунок 2.3.2.3 (Изменение данных 3)

2.3.3 Добавление информации

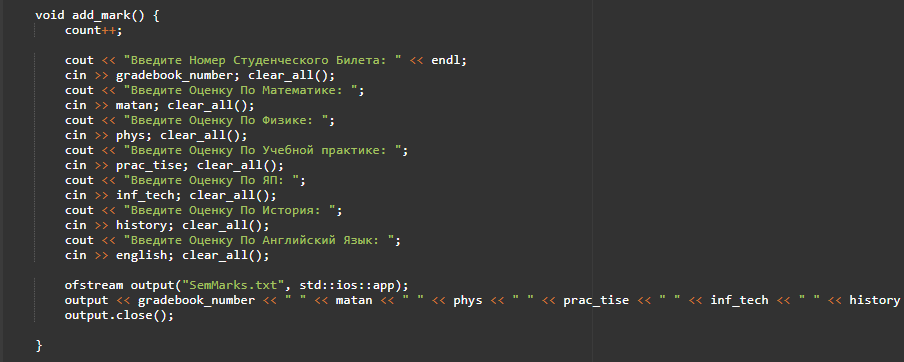


Рисунок 2.3.3.1 (Добавление данных)

2.3.4 Удаление данных

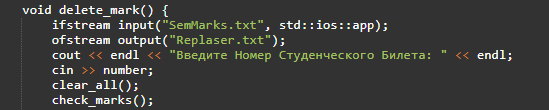


Рисунок 2.3.4.1 (Удаление данных 1)

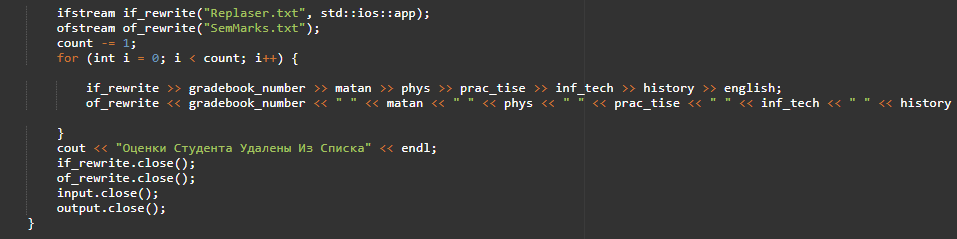


Рисунок 2.3.4.2 (Удаление данных 2)

2.3.5 Чтение данных

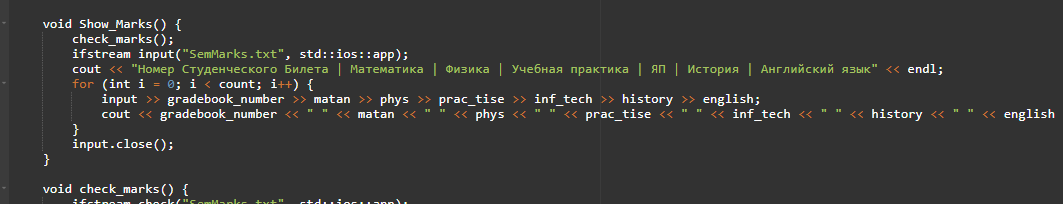


Рисунок 2.3.5.1 (Чтение данных)

2.3.6 Использование конструкторов и деструкторов

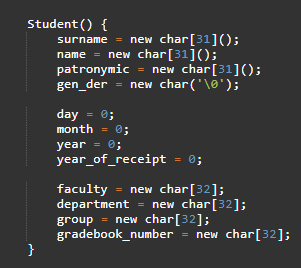


Рисунок 2.3.6.2 (Использование конструкторов)

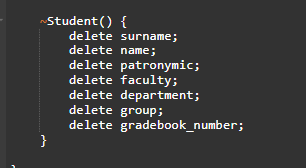


Рисунок 2.3.6.2 (Использование деструкторов)

2.3.7 Наследование и друзья классов

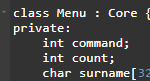


Рисунок 2.3.7.1(Наследование классов)

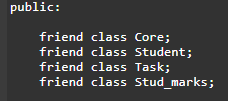


Рисунок 2.3.7.2(Друзья классов)

2.3.8 Перегрузка функций

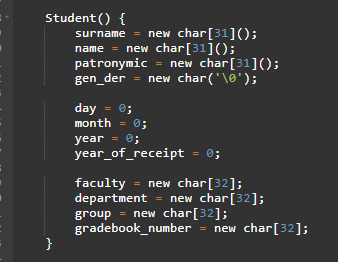


Рисунок 2.3.8.1 (Перегрузка класса Student)

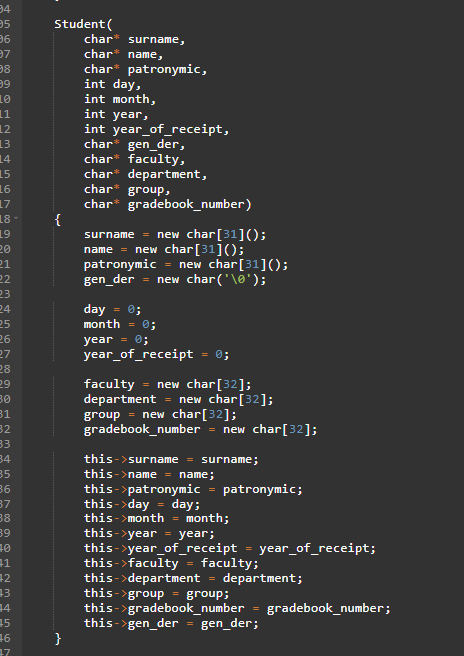


Рисунок 2.3.8.2(Перегрузка класса Student)

2.4 Контрольные данные

Таблица 2.4.1 – Данные всех студентов

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ФИО | пол | д/р | № з/к | г/п | группа | факультет | кафедра |
| 1 | Калинин Владимир Николаевич | М | 01.01.1999 | СК-7738 | 2019 | БББО-02-19 | ИБ | КБ-8 |
| 2 | Калинин Владимир Николаевич | М | 01.01.1999 | СК-7738 | 2019 | БББО-02-19 | ИБ | КБ-8 |
| 3 | Калинин Владимир Николаевич | М | 01.01.1999 | СК-7738 | 2019 | БББО-02-19 | ИБ | КБ-8 |
| 4 | Калинин Владимир Николаевич | М | 01.01.1999 | СК-7738 | 2019 | БББО-02-19 | ИБ | КБ-8 |
| 5 | Калинин Владимир Николаевич | М | 01.01.1999 | СК-7738 | 2019 | БББО-02-19 | ИБ | КБ-8 |
| 6 | Калинин Владимир Николаевич | М | 01.01.1999 | СК-7738 | 2019 | БББО-02-19 | ИБ | КБ-8 |
| 7 | Калинин Владимир Николаевич | М | 01.01.1999 | СК-7738 | 2019 | БББО-02-19 | ИБ | КБ-8 |

Таблица 2.4.2 – Зачетная книжка студента №1

|  |  |  |
| --- | --- | --- |
| Семестр | Предметы | Оценка |
| 1 | Физика | 2 |
| Алгебра | 3 |
| Яп | 4 |
| Аса | 3 |
| Английский | 4 |
| 2 | Английский | 3 |
| Яп | 4 |
| Алгебра | 3 |
| Матанализ | 5 |
| Философия | 2 |
| 3 | Физика | 3 |
| Яп | 4 |
| Алгебра | 1 |
| Английский | 3 |
| Аса | 4 |
| 4 | Аса | 3 |
| Яп | 5 |
| Английский | 4 |
| Физкультура | 3 |
| Впд | 4 |

Таблица 2.4.3 – Зачетная книжка студента №2

|  |  |  |
| --- | --- | --- |
| семестр | предметы | оценка |
| 1 | История | 5 |
| Матанализ | 5 |
| Алгебра | 5 |
| Физика | 4 |
| Яп | 5 |

2.5 Контрольный пример

При запуске программы в консоли выводится меню.

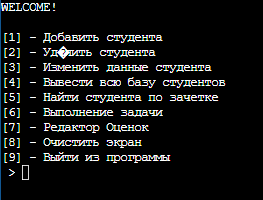


Рисунок 2.5.1 (Главное меню)

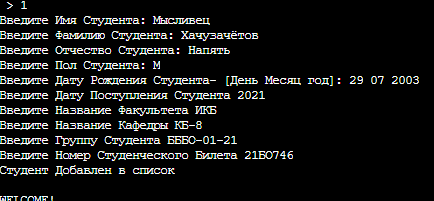


Рисунок 2.5.2 (Добавление студента)

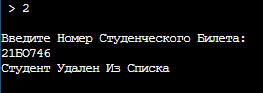


Рисунок 2.5.3 (Удаление Студента)

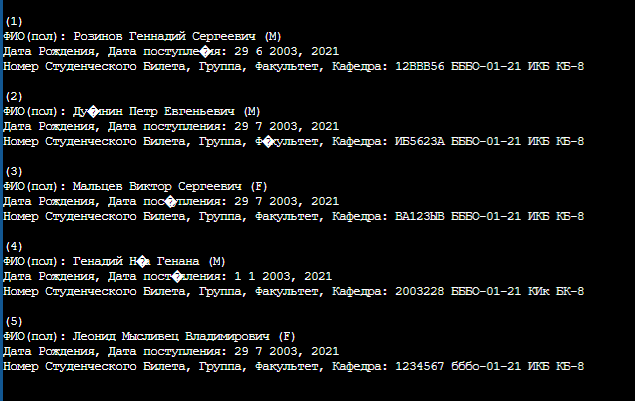


Рисунок 2.5.4 (Вывод списка студентов)



Рисунок 2.5.5 (Выборка в соответствии с вариантом)

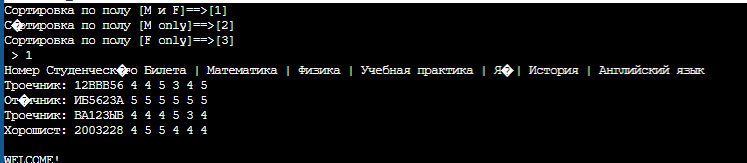


Рисунок 2.5.6 (Выбор 1 варианта)

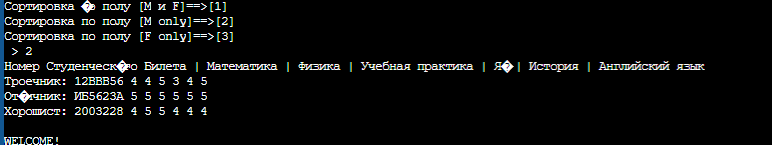


Рисунок 2.5.7 (Выбор 2 варианта)

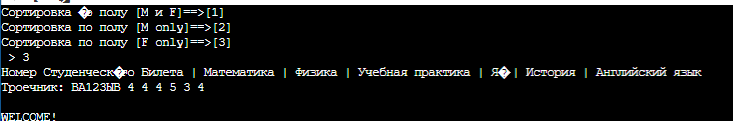


Рисунок 2.5.8 (Выбор 3 варианта)

Результат работы программы:

После того как пользователь выбрал условия, программа выводит на экран результат в соответствии с выбранным вариантом. Как можно заметить из Рисунков 2.5.6, 2.5.7, 2.5.8 при выборе варианта с полом программа выводит соответствующие результаты

2.6 Описание алгоритма варианта

Вариант 46

Открывается меню, в котором пользователь может выбрать несколько вариантов работы программы, а именно критерии сортировки по полу:

1) Мужской

2) Женский

3) Мужской и Женский

И во всех случаях пользователь получает выборку студентов, рассортированных по критериям успеваемости и пола.

При вводе: «1» программа выводит список из студентов мужского и женского пола, а также фильтрует их по успеваемости (“Хорошист, отличник, троечник”)

При вводе: «2» программа выводит список из студентов мужского пола, а также фильтрует их по успеваемости (“Хорошист, отличник, троечник”)

При вводе: «3» программа выводит список из студентов женского пола, а также фильтрует их по успеваемости (“Хорошист, отличник, троечник”)

2.7 Руководство пользователя

При запуске программы на экране выводится консоль с меню, где пользователь может выбрать один из нескольких пунктов, таких как: ввести/добавить студентов, удалить студента, показать всех студентов в университете, изменить информацию о студенте или показать выборку. В зависимости от выбранного пользователем варианта, выполняется определенная часть программы.

При вводе «9» осуществляется выход из программы.

При вводе «1» программа предлагает ввести студента с клавиатуры.

При вводе «2» программа запрашивает номер зачетной книжки, далее удаляет соответствующего этому номеру студента.

При вводе «3» запрашивается номер зачетной книжки для изменения определенной информации о студенте.

При вводе «4» программа выводит общий список студентов в университете, определенную информацию о них (ФИО, пол, дата дня рождения, год поступления, группа, номер зачетки, кафедра, факультет).

При вводе «5» программа предлагает найти студента по зачётке.

При вводе «6» программа выполняет задание варианта.

При вводе «7» программа предлагает отредактировать оценки.

При вводе «8» программа очищает экран.

2.8 Системные требования

Язык программирования: С++.

Операционная система: Linux Mint 20.1, Linux Ubuntu 21.10, Linux Ubuntu 20.04, Windows XP, Windows NT, Windows 7, Windows 10.

ОЗУ: 1 Гб и более.

Свободное место на диске: 200 Мб и более.

ЗАКЛЮЧЕНИЕ

В процессе выполнения курсовой работы была написана программа, Вариант 46**.** Разбить группу на 3 части, с поиском среди лиц определенного пола: отличников, хорошистов, троечников за все время обучения.

Кроме того, в процессе реализации программы мною были закреплены базовые навыки программирования, полученные при изучении дисциплины Языки программирования и языка программирования C++.

Были подробно рассмотрены теоретические выкладки, использованные в процессе написания программы.

Программа была реализована c использованием технологии ООП. Более того, была внедрена работа с динамической памятью, работа с файлами. Более того, детально были разобраны и применены на практике методы симметричного и ассиметричного шифрования и дешифрования файлов.

Дополнительно были созданы необходимые и достаточные условия для корректного выполнения индивидуального варианта задания.

По результату выполнения тестирования программного продукта, можно сделать вывод о том, что программа работает корректно и справляется с поставленной задачей.

Перспективы разработки данного продукта характеризуются высокой потребностью в универсальном и безопасном программном продукте, предназначенном для внедрения в средства автоматизации и информационные системы.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Мерсов А. А. Основы объектно-ориентированного программирования на языке С++ [Электронный ресурс]: практикум / А. А. Мерсов, А. М. Русаков, В. В. Филатов. — М.: РТУ МИРЭА, 2021. — Электрон. опт. диск (ISO)

2. Павловская Т. А. C/C++. Программирование на языке высокого уровня: учебник для вузов. – " Издательский дом"" Питер""", 2021.

3. Страуструп Б. Программирование. Принципы и практика использования C++. – Litres, 2021.

4. ООП в картинках [Электронный ресурс].- Интернет-сайт, 2019.- Режим доступа: https://habr.com/ru/post/463125, свободный (дата обращения 07.05.2022).

5. Лафоре Р. Объектно-ориентированное программирование в C++. 4-е изд., перераб. и доп //Санкт-Петербург.: Питер. – 2018.

6. Быстрая сортировка [Электронный ресурс] Интернет-сайт, 2020. - Режим доступа: https://habr.com/ru/company/otus/blog/524948, сво-бодный (дата обращения 08.05.2022).

7. Стенли Липпман Язык программирования С++: полное руководство / Липпман Стенли, ЛажойеЖози. – Саратов: Профобразование, 2017. 1104 c.

8. Страуструп Б. Дизайн и эволюция языка С++. – Litres, 2022.

9. Объектно-ориентированное мышление / Вайсфельд М. / СПб.: Питер, 2014. / ISBN 978-5-496-00793-1.

10. Уроки по С++ для начинающих – Ravesli [Электронный ресурс].- Интернет-сайт, 2015.- Режим доступа: https://ravesli.com/uroki-cpp, свободный (дата обращения 01.05.2022).

ПРИЛОЖЕНИЕ А

#include <stdio.h>

#include <iostream>

#include <fstream>

#include <iomanip>

using namespace std**;**

**class** **Core** **{**

public**:**

**int** protect\_marks**()** **{**

**int** val**;**

**while** **(**true**)** **{**

cin **>>** val**;**

**if** **(**val **>** 0 **&&** val **<=** 5**)** **{**

clear\_all**();**

**return** val**;**

**}**

cout **<<** "[CODE\_ERROR] - Введены неверные данные" **<<** endl**;**

clear\_all**();**

**}**

**}**

char protect**()** **{**

char val**;**

**int** result**;**

**while** **(**true**)** **{**

cin **>>** val**;**

result **=** val **-** '0'**;**

**if** **(!**isdigit**(**val**))** **{**

cout **<<** "[CODE\_ERROR] - Введены неверные данные" **<<** endl**;**

**return** 8**;**

**}**

**else** **if** **(**result **>** 0 **&&** result **<=** 9**)** **{**

clear\_all**();**

**return** result**;**

**}**

cout **<<** "[CODE\_ERROR] - Введены неверные данные" **<<** endl**;**

clear\_all**();**

**}**

**}**

**int** protect\_change**()** **{**

**int** val**;**

**while** **(**true**)** **{**

cin **>>** val**;**

**if** **(**val **>** 0 **&&** val **<=** 4**)** **{**

clear\_all**();**

**return** val**;**

**}**

cout **<<** "[CODE\_ERROR] - Введены неверные данные" **<<** endl**;**

clear\_all**();**

**}**

**}**

**int** protect\_task**()** **{**

**int** val**;**

**while** **(**true**)** **{**

cin **>>** val**;**

**if** **(**val **>** 0 **&&** val **<=** 5**)** **{**

clear\_all**();**

**return** val**;**

**}**

cout **<<** "[CODE\_ERROR] - Введены неверные данные" **<<** endl**;**

clear\_all**();**

**}**

**}**

**int** protect\_day**()** **{**

**int** val**;**

**while** **(**true**)** **{**

cin **>>** val**;**

**if** **(**val **>** 0 **&&** val **<=** 31**)** **{**

clear\_all**();**

**return(**val**);**

**}**

cout **<<** "[CODE\_ERROR] - Введены неверные данные" **<<** endl**;**

cout **<<** "Неверная Дата" **<<** endl**;**

clear\_all**();**

**}**

**}**

**int** protect\_month**()** **{**

**int** val**;**

**while** **(**true**)** **{**

cin **>>** val**;**

**if** **(**val **>** 0 **&&** val **<=** 12**)** **{**

clear\_all**();**

**return(**val**);**

**}**

cout **<<** "[CODE\_ERROR] - Введены неверные данные" **<<** endl**;**

cout **<<** "Неверная Дата" **<<** endl**;**

clear\_all**();**

**}**

**}**

**int** protect\_year**()** **{**

**int** val**;**

**while** **(**true**)** **{**

cin **>>** val**;**

**if** **(**val **>** 1922 **&&** val **<=** 2022**)** **{**

clear\_all**();**

**return(**val**);**

**}**

cout **<<** "[CODE\_ERROR] - Введены неверные данные" **<<** endl**;**

cout **<<** "Неверная Дата" **<<** endl**;**

clear\_all**();**

**}**

**}**

char protect\_ch**()** **{**

char val**;**

**while** **(**true**)** **{**

cin **>>** val**;**

**if** **(**val **==** 'M' **||** val **==** 'F'**)** **{**

clear\_all**();**

**return** val**;**

**}**

cout **<<** "[CODE\_ERROR] - Введены неверные данные" **<<** endl**;**

clear\_all**();**

**}**

**}**

void clear\_all**()** **{**

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

cin**.**clear**();**

**}**

**};**

**class** **Task** **:** Core **{**

private**:**

**int** command**;**

**int** a **=** 0**;**

**int** b **=** 0**;**

**int** c **=** 0**;**

char gradebook\_number**[**32**];**

char number**[**32**];**

**int** matan**;**

**int** phys**;**

**int** prac\_tise**;**

**int** inf\_tech**;**

**int** history**;**

**int** english**;**

**int** count **=** 0**;**

char surname**[**32**];**

char name**[**32**];**

char patronymic**[**32**];**

**int** day**;**

**int** month**;**

**int** year**;**

**int** year\_of\_receipt**;**

char gen\_der**;**

char faculty**[**32**];**

char department**[**32**];**

char group**[**32**];**

public**:**

void task\_menu**()** **{**

check\_marks**();**

cout **<<** "Сортировка по полу [M и F]==>[1]" **<<** endl**;**

cout **<<** "Сортировка по полу [M only]==>[2]" **<<** endl**;**

cout **<<** "Сортировка по полу [F only]==>[3]" **<<** endl**;**

cout **<<** " > "**;**

cin **>>** command**;** clear\_all**();**

cout **<<** "Номер Студенческого Билета | Математика | Физика | Учебная практика | ЯП | История | Английский язык" **<<** endl**;**

switch **(**command**)** **{**

case 1**:**

**{**

find\_F\_or\_M**();**

**break;**

**}**

case 2**:**

**{**

find\_M**();**

**break;**

**}**

case 3**:**

**{**

find\_F**();**

**break;**

**}**

**}**

**}**

void show\_stud**()** **{**

ifstream marks\_input**(**"SemMarks.txt"**,** std**::**ios**::**app**);**

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

marks\_input **>>** gradebook\_number **>>** matan **>>** phys **>>** prac\_tise **>>** inf\_tech **>>** history **>>** english**;**

**int** tmp**(**1**);**

**for** **(int** i **=** 0**;** i **<** 7**;** i**++)** **{**

**if** **(**gradebook\_number**[**i**]** **!=** number**[**i**])** **{**

tmp **=** 0**;**

**break;**

**}**

**}**

**if** **(**tmp **==** 1**)** **{**

**if** **(**matan **==** 5 **&&** phys **==** 5 **&&** prac\_tise **==** 5 **&&** inf\_tech **==** 5 **&&** history **==** 5 **&&** english **==** 5**)** **{**

cout **<<** "Отличник: " **<<** gradebook\_number **<<** " " **<<** matan **<<** " " **<<** phys **<<** " " **<<** prac\_tise **<<** " " **<<** inf\_tech **<<** " " **<<** history **<<** " " **<<** english **<<** endl**;**

**break;**

**}**

**else** **if** **((**matan **==** 5 **||** matan **==** 4**)** **&&** **(**phys **==** 5 **||** phys **==** 4**)** **&&** **(**prac\_tise **==** 5 **||** prac\_tise **==** 4**)** **&&** **(**inf\_tech **==** 5 **||** inf\_tech **==** 4**)** **&&** **(**history **==** 5 **||** history **==** 4**)** **&&** **(**english **==** 5 **||** english **==** 4**))** **{**

cout **<<** "Хорошист: " **<<** gradebook\_number **<<** " " **<<** matan **<<** " " **<<** phys **<<** " " **<<** prac\_tise **<<** " " **<<** inf\_tech **<<** " " **<<** history **<<** " " **<<** english **<<** endl**;**

**break;**

**}**

**else** **if** **(**matan **==** 3 **||** phys **==** 3 **||** prac\_tise **==** 3 **||** inf\_tech **==** 3 **||** history **==** 3 **||** english **==** 3**)** **{**

cout **<<** "Троечник: " **<<** gradebook\_number **<<** " " **<<** matan **<<** " " **<<** phys **<<** " " **<<** prac\_tise **<<** " " **<<** inf\_tech **<<** " " **<<** history **<<** " " **<<** english **<<** endl**;**

**break;**

**}**

**}**

**}**

marks\_input**.**close**();**

**}**

void find\_M**()** **{**

check\_marks**();**

ifstream **input(**"Base.txt"**,** std**::**ios**::**app**);**

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

**input** **>>** number **>>** name **>>** surname **>>** patronymic **>>** gen\_der **>>** day **>>** month **>>** year **>>** year\_of\_receipt **>>** faculty **>>** department **>>** group**;**

**if** **(**gen\_der **==** 'M'**)** **{**

show\_stud**();**

**}**

**}**

**input.**close**();**

**}**

void find\_F**()** **{**

check\_marks**();**

ifstream **input(**"Base.txt"**,** std**::**ios**::**app**);**

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

**input** **>>** number **>>** name **>>** surname **>>** patronymic **>>** gen\_der **>>** day **>>** month **>>** year **>>** year\_of\_receipt **>>** faculty **>>** department **>>** group**;**

**if** **(**gen\_der **==** 'F'**)** **{**

show\_stud**();**

**}**

**}**

**input.**close**();**

**}**

void find\_F\_or\_M**()** **{**

check\_marks**();**

ifstream **input(**"Base.txt"**,** std**::**ios**::**app**);**

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

**input** **>>** number **>>** name **>>** surname **>>** patronymic **>>** gen\_der **>>** day **>>** month **>>** year **>>** year\_of\_receipt **>>** faculty **>>** department **>>** group**;**

**if** **(**gen\_der **==** 'M' **||** gen\_der **==** 'F'**)** **{**

show\_stud**();**

**}**

**}**

**input.**close**();**

**}**

void check\_marks**()** **{**

ifstream check**(**"SemMarks.txt"**,** std**::**ios**::**app**);**

count **=** 0**;**

**while** **(!**check**.**eof**())** **{**

count**++;**

check **>>** gradebook\_number **>>** matan **>>** phys **>>** prac\_tise **>>** inf\_tech **>>** history **>>** english**;**

**}**

count**--;**

**}**

friend **class** **Student;**

friend **class** **Menu;**

friend **class** **Stud\_marks;**

void clear\_all**()** **{**

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

cin**.**clear**();**

**}**

**};**

**class** **Student** **:** Core **{**

private**:**

char**\*** surname **=** nullptr**;**

char**\*** name **=** nullptr**;**

char**\*** patronymic **=** nullptr**;**

**int** day**;**

**int** month**;**

**int** year**;**

char**\*** gen\_der **=** nullptr**;**

**int** year\_of\_receipt**;**

char**\*** faculty **=** nullptr**;**

char**\*** department **=** nullptr**;**

char**\*** group **=** nullptr**;**

char**\*** gradebook\_number **=** nullptr**;**

public**:**

Student**()** **{**

surname **=** new char**[**31**]();**

name **=** new char**[**31**]();**

patronymic **=** new char**[**31**]();**

gen\_der **=** new char**(**'\0'**);**

day **=** 0**;**

month **=** 0**;**

year **=** 0**;**

year\_of\_receipt **=** 0**;**

faculty **=** new char**[**32**];**

department **=** new char**[**32**];**

group **=** new char**[**32**];**

gradebook\_number **=** new char**[**32**];**

**}**

Student**(**

char**\*** surname**,**

char**\*** name**,**

char**\*** patronymic**,**

**int** day**,**

**int** month**,**

**int** year**,**

**int** year\_of\_receipt**,**

char**\*** gen\_der**,**

char**\*** faculty**,**

char**\*** department**,**

char**\*** group**,**

char**\*** gradebook\_number**)**

**{**

surname **=** new char**[**31**]();**

name **=** new char**[**31**]();**

patronymic **=** new char**[**31**]();**

gen\_der **=** new char**(**'\0'**);**

day **=** 0**;**

month **=** 0**;**

year **=** 0**;**

year\_of\_receipt **=** 0**;**

faculty **=** new char**[**32**];**

department **=** new char**[**32**];**

group **=** new char**[**32**];**

gradebook\_number **=** new char**[**32**];**

this**->**surname **=** surname**;**

this**->**name **=** name**;**

this**->**patronymic **=** patronymic**;**

this**->**day **=** day**;**

this**->**month **=** month**;**

this**->**year **=** year**;**

this**->**year\_of\_receipt **=** year\_of\_receipt**;**

this**->**faculty **=** faculty**;**

this**->**department **=** department**;**

this**->**group **=** group**;**

this**->**gradebook\_number **=** gradebook\_number**;**

this**->**gen\_der **=** gen\_der**;**

**}**

void **set()** **{**

set\_name**();**

set\_surname**();**

set\_patronymic**();**

set\_gen\_der**();**

set\_birthdate**();**

set\_year\_of\_receipt**();**

set\_faculty**();**

set\_department**();**

set\_group**();**

set\_gradebook\_number**();**

**}**

void write**()** **{**

ofstream output**(**"Base.txt"**,** std**::**ios**::**app**);**

output **<<** gradebook\_number **<<** " " **<<** name **<<** " " **<<** surname **<<** " " **<<** patronymic **<<** " " **<<** gen\_der **<<** " " **<<** day **<<** " " **<<** month **<<** " " **<<** year **<<** " " **<<** year\_of\_receipt **<<** " " **<<** faculty **<<** " " **<<** department **<<** " " **<<** group **<<** endl**;**

output**.**close**();**

**}**

void set\_name**()** **{**

cout **<<** "Введите Имя Студента: "**;**

cin **>>** name**;**

clear\_all**();**

**}**

void set\_surname**()** **{**

cout **<<** "Введите Фамилию Студента: "**;**

cin **>>** surname**;**

clear\_all**();**

**}**

void set\_patronymic**()** **{**

cout **<<** "Введите Отчество Студента: "**;**

cin **>>** patronymic**;**

clear\_all**();**

**}**

void set\_gen\_der**()** **{**

cout **<<** "Введите Пол Студента: "**;**

**\***gen\_der **=** protect\_ch**();**

**}**

void set\_birthdate**()** **{**

cout **<<** "Введите Дату Рождения Студента- [День Месяц год]: "**;**

day **=** protect\_day**();**

month **=** protect\_month**();**

year **=** protect\_year**();**

**}**

void set\_year\_of\_receipt**()** **{**

cout **<<** "Введите Дату Поступления Студента "**;**

year\_of\_receipt **=** protect\_year**();**

**}**

void set\_faculty**()** **{**

cout **<<** "Введите Название Факультета "**;**

cin **>>** faculty**;**

clear\_all**();**

**}**

void set\_department**()** **{**

cout **<<** "Введите Название Кафедры "**;**

cin **>>** department**;**

clear\_all**();**

**}**

void set\_group**()** **{**

cout **<<** "Введите Группу Студента "**;**

cin **>>** group**;**

clear\_all**();**

**}**

void set\_gradebook\_number**()** **{**

cout **<<** "Введите Номер Студенческого Билета "**;**

cin **>>** gradebook\_number**;**

clear\_all**();**

**}**

void get**()** **{**

get\_name**();**

get\_surname**();**

get\_patronymic**();**

get\_gen\_der**();**

get\_birthdate**();**

get\_year\_of\_receipt**();**

get\_faculty**();**

get\_department**();**

get\_group**();**

get\_gradebook\_number**();**

**}**

void get\_name**()** **{**

cout **<<** endl **<<** "Имя Студента: "**;**

cout **<<** name**;**

**}**

void get\_surname**()** **{**

cout **<<** endl **<<** "Фамилия Студента: "**;**

cout **<<** surname**;**

**}**

void get\_patronymic**()** **{**

cout **<<** endl **<<** "Отчество Студента: "**;**

cout **<<** patronymic**;**

**}**

void get\_gen\_der**()** **{**

cout **<<** endl **<<** "Пол Студента: "**;**

cout **<<** gen\_der**;**

**}**

void get\_birthdate**()** **{**

cout **<<** endl **<<** "Дата Рождения Студента -[Д М Г]: "**;**

cout **<<** day **<<** " " **<<** month **<<** " " **<<** year**;**

**}**

void get\_year\_of\_receipt**()** **{**

cout **<<** endl **<<** "Дата Поступления Студента: "**;**

cout **<<** year\_of\_receipt**;**

**}**

void get\_faculty**()** **{**

cout **<<** endl **<<** "Факультет Студента: "**;**

cout **<<** faculty**;**

**}**

void get\_department**()** **{**

cout **<<** endl **<<** "Кафедра Студента: "**;**

cout **<<** department**;**

**}**

void get\_group**()** **{**

cout **<<** endl **<<** "Группа Студента: "**;**

cout **<<** group**;**

**}**

void get\_gradebook\_number**()** **{**

cout **<<** endl **<<** "Номер Студенческого Билета: "**;**

cout **<<** gradebook\_number**;**

**}**

char ret\_gradebook\_number**()** **{**

**return** **\***gradebook\_number**;**

**}**

char ret\_name**()** const **{**

**return** **\***name**;**

**}**

char ret\_surname**()** const **{**

**return** **\***surname**;**

**}**

char ret\_patronymic**()** const **{**

**return** **\***patronymic**;**

**}**

char ret\_gen\_der**()** const **{**

**return** **\***gen\_der**;**

**}**

**int** ret\_day**()** const **{**

**return** day**;**

**}**

**int** ret\_month**()** const **{**

**return** month**;**

**}**

**int** ret\_year**()** const **{**

**return** year**;**

**}**

**int** ret\_year\_of\_receipt**()** const **{**

**return** year\_of\_receipt**;**

**}**

char ret\_faculty**()** const **{**

**return** **\***faculty**;**

**}**

char ret\_department**()** const **{**

**return** **\***department**;**

**}**

char ret\_group**()** const **{**

**return** **\***group**;**

**}**

char ret\_gradebook\_number**()** const **{**

**return** **\***gradebook\_number**;**

**}**

void clear\_all**()** **{**

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

cin**.**clear**();**

**}**

**~**Student**()** **{**

delete surname**;**

delete name**;**

delete patronymic**;**

delete faculty**;**

delete department**;**

delete group**;**

delete gradebook\_number**;**

**}**

**};**

**class** **Stud\_marks** **:** Core **{**

private**:**

**int** count**;**

char gradebook\_number**[**32**];**

char number**[**32**];**

**int** command**;**

**int** matan**;**

**int** phys**;**

**int** prac\_tise**;**

**int** inf\_tech**;**

**int** history**;**

**int** english**;**

public**:**

friend **class** **Student;**

friend **class** **Core;**

friend **class** **Menu;**

Stud\_marks**()** **{**

count **=** 0**;**

check\_marks**();**

**}**

void add\_mark**()** **{**

count**++;**

cout **<<** "Введите Номер Студенческого Билета: " **<<** endl**;**

cin **>>** gradebook\_number**;** clear\_all**();**

cout **<<** "Введите Оценку По Математике: "**;**

cin **>>** matan**;** clear\_all**();**

cout **<<** "Введите Оценку По Физике: "**;**

cin **>>** phys**;** clear\_all**();**

cout **<<** "Введите Оценку По Учебной практике: "**;**

cin **>>** prac\_tise**;** clear\_all**();**

cout **<<** "Введите Оценку По ЯП: "**;**

cin **>>** inf\_tech**;** clear\_all**();**

cout **<<** "Введите Оценку По История: "**;**

cin **>>** history**;** clear\_all**();**

cout **<<** "Введите Оценку По Английский Язык: "**;**

cin **>>** english**;** clear\_all**();**

ofstream output**(**"SemMarks.txt"**,** std**::**ios**::**app**);**

output **<<** gradebook\_number **<<** " " **<<** matan **<<** " " **<<** phys **<<** " " **<<** prac\_tise **<<** " " **<<** inf\_tech **<<** " " **<<** history **<<** " " **<<** english **<<** endl**;**

output**.**close**();**

**}**

void delete\_mark**()** **{**

ifstream **input(**"SemMarks.txt"**,** std**::**ios**::**app**);**

ofstream output**(**"Replaser.txt"**);**

cout **<<** endl **<<** "Введите Номер Студенческого Билета: " **<<** endl**;**

cin **>>** number**;**

clear\_all**();**

check\_marks**();**

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

**input** **>>** gradebook\_number **>>** matan **>>** phys **>>** prac\_tise **>>** inf\_tech **>>** history **>>** english**;**

**int** tmp**(**1**);**

**for** **(int** i **=** 0**;** i **<** 7**;** i**++)** **{**

**if** **(**gradebook\_number**[**i**]** **!=** number**[**i**])** **{**

tmp **=** 0**;**

**break;**

**}**

**}**

**if** **(**tmp **==** 0**)** **{**

output **<<** gradebook\_number **<<** " " **<<** matan **<<** " " **<<** phys **<<** " " **<<** prac\_tise **<<** " " **<<** inf\_tech **<<** " " **<<** history **<<** " " **<<** english **<<** endl**;**

**}**

**}**

ifstream if\_rewrite**(**"Replaser.txt"**,** std**::**ios**::**app**);**

ofstream of\_rewrite**(**"SemMarks.txt"**);**

count **-=** 1**;**

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

if\_rewrite **>>** gradebook\_number **>>** matan **>>** phys **>>** prac\_tise **>>** inf\_tech **>>** history **>>** english**;**

of\_rewrite **<<** gradebook\_number **<<** " " **<<** matan **<<** " " **<<** phys **<<** " " **<<** prac\_tise **<<** " " **<<** inf\_tech **<<** " " **<<** history **<<** " " **<<** english **<<** endl**;**

**}**

cout **<<** "Оценки Студента Удалены Из Списка" **<<** endl**;**

if\_rewrite**.**close**();**

of\_rewrite**.**close**();**

**input.**close**();**

output**.**close**();**

**}**

void Change\_Marks**()** **{**

ifstream **input(**"SemMarks.txt"**,** std**::**ios**::**app**);**

ofstream output**(**"Replaser.txt"**);**

cout **<<** endl **<<** "Введите Номер Студенческого Билета: " **<<** endl**;**

cin **>>** number**;**

clear\_all**();**

cout **<<** endl**;**

cout **<<** "Что Вы Хотите Изменить? " **<<** endl **<<** endl**;**

cout **<<** "[1] Номер Студенческого Билета" **<<** endl**;**

cout **<<** "[2] Математика" **<<** endl**;**

cout **<<** "[3] Физика" **<<** endl**;**

cout **<<** "[4] Учебная практика" **<<** endl**;**

cout **<<** "[5] ЯП" **<<** endl**;**

cout **<<** "[6] История" **<<** endl**;**

cout **<<** "[7] Английский Язык" **<<** endl**;**

cout **<<** "[8] Назад" **<<** endl**;**

cout **<<** " > "**;**

command **=** protect**();**

check\_marks**();**

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

**input** **>>** gradebook\_number **>>** matan **>>** phys **>>** prac\_tise **>>** inf\_tech **>>** history **>>** english**;**

**int** tmp**(**1**);**

**for** **(int** i **=** 0**;** i **<** 7**;** i**++)** **{**

**if** **(**gradebook\_number**[**i**]** **!=** number**[**i**])** **{**

tmp **=** 0**;**

**break;**

**}**

**}**

**if** **(**tmp **==** 1**)** **{**

switch **(**command**)** **{**

case 1**:**

**{**

cout **<<** "Введите Новый Номер Студенческого Билета: "**;**

cin **>>** gradebook\_number**;**

clear\_all**();**

**break;**

**}**

case 2**:**

**{**

cout **<<** "Введите Оценку По Математике: "**;**

cin **>>** matan**;** clear\_all**();**

**break;**

**}**

case 3**:**

**{**

cout **<<** "Введите Оценку По Физике: "**;**

cin **>>** phys**;** clear\_all**();**

**break;**

**}**

case 4**:**

**{**

cout **<<** "Введите Оценку По Учебной практике: "**;**

cin **>>** prac\_tise**;** clear\_all**();**

**break;**

**}**

case 5**:**

**{**

cout **<<** "Введите Оценку По ЯП: "**;**

cin **>>** inf\_tech**;** clear\_all**();**

**break;**

**}**

case 6**:**

**{**

cout **<<** "Введите Оценку По История: "**;**

cin **>>** history**;** clear\_all**();**

**}**

case 7**:**

**{**

cout **<<** "Введите Оценку По Английский Язык: "**;**

cin **>>** english**;** clear\_all**();**

**}**

case 8**:**

**{**

**break;**

**}**

**}**

output **<<** gradebook\_number **<<** " " **<<** matan **<<** " " **<<** phys **<<** " " **<<** prac\_tise **<<** " " **<<** inf\_tech **<<** " " **<<** history **<<** " " **<<** english **<<** endl**;**

**}**

**else** output **<<** gradebook\_number **<<** " " **<<** matan **<<** " " **<<** phys **<<** " " **<<** prac\_tise **<<** " " **<<** inf\_tech **<<** " " **<<** history **<<** " " **<<** english **<<** endl**;**

**}**

ifstream if\_rewrite**(**"Replaser.txt"**,** std**::**ios**::**app**);**

ofstream of\_rewrite**(**"SemMarks.txt"**);**

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

if\_rewrite **>>** gradebook\_number **>>** matan **>>** phys **>>** prac\_tise **>>** inf\_tech **>>** history **>>** english**;**

of\_rewrite **<<** gradebook\_number **<<** " " **<<** matan **<<** " " **<<** phys **<<** " " **<<** prac\_tise **<<** " " **<<** inf\_tech **<<** " " **<<** history **<<** " " **<<** english **<<** endl**;**

**}**

cout **<<** "Оценки Студента Изменены" **<<** endl**;**

if\_rewrite**.**close**();**

of\_rewrite**.**close**();**

**input.**close**();**

output**.**close**();**

**}**

void Show\_Marks**()** **{**

check\_marks**();**

ifstream **input(**"SemMarks.txt"**,** std**::**ios**::**app**);**

cout **<<** "Номер Студенческого Билета | Математика | Физика | Учебная практика | ЯП | История | Английский язык" **<<** endl**;**

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

**input** **>>** gradebook\_number **>>** matan **>>** phys **>>** prac\_tise **>>** inf\_tech **>>** history **>>** english**;**

cout **<<** gradebook\_number **<<** " " **<<** matan **<<** " " **<<** phys **<<** " " **<<** prac\_tise **<<** " " **<<** inf\_tech **<<** " " **<<** history **<<** " " **<<** english **<<** endl**;**

**}**

**input.**close**();**

**}**

void check\_marks**()** **{**

ifstream check**(**"SemMarks.txt"**,** std**::**ios**::**app**);**

count **=** 0**;**

**while** **(!**check**.**eof**())** **{**

count**++;**

check **>>** gradebook\_number **>>** matan **>>** phys **>>** prac\_tise **>>** inf\_tech **>>** history **>>** english**;**

**}**

count**--;**

**}**

**~**Stud\_marks**()** **{**

**}**

void clear\_all**()** **{**

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

cin**.**clear**();**

**}**

**};**

**class** **Menu** **:** Core **{**

private**:**

**int** command**;**

**int** count**;**

char surname**[**32**];**

char name**[**32**];**

char patronymic**[**32**];**

**int** day**;**

**int** month**;**

**int** year**;**

**int** year\_of\_receipt**;**

char gen\_der**;**

char faculty**[**32**];**

char department**[**32**];**

char group**[**32**];**

char gradebook\_number**[**32**];**

char number**[**32**];**

public**:**

friend **class** **Core;**

friend **class** **Student;**

friend **class** **Task;**

friend **class** **Stud\_marks;**

Menu**()** **{**

count **=** 0**;**

**}**

**int** welcome**()** **{**

ifstream check**(**"Base.txt"**);**

count **=** 0**;**

**while** **(!**check**.**eof**())** **{**

count**++;**

check **>>** gradebook\_number **>>** name **>>** surname **>>** patronymic **>>** gen\_der **>>** day **>>** month **>>** year **>>** year\_of\_receipt **>>** faculty **>>** department **>>** group**;**

**}**

count**--;**

cout **<<** "WELCOME!" **<<** endl **<<** endl**;**

cout **<<** "[1] - Добавить студента" **<<** endl**;**

cout **<<** "[2] - Удалить студента" **<<** endl**;**

cout **<<** "[3] - Изменить данные студента" **<<** endl**;**

cout **<<** "[4] - Вывести всю базу студентов" **<<** endl**;**

cout **<<** "[5] - Найти студента по зачетке" **<<** endl**;**

cout **<<** "[6] - Выполнение задачи" **<<** endl**;**

cout **<<** "[7] - Редактор Оценок" **<<** endl**;**

cout **<<** "[8] - Очистить экран" **<<** endl**;**

cout **<<** "[9] - Выйти из программы" **<<** endl**;**

cout **<<** " > "**;**

**//**char check**[**32**];** cin **>>** check**;**

command **=** protect**();**

**if** **(**command **==** 1**)** **{**

Student student**;**

student**.set();**

student**.**write**();**

count**++;**

cout **<<** "Студент Добавлен в список" **<<** endl **<<** endl**;**

welcome**();**

**}**

**else** **if** **(**command **==** 2**)** **{**

Delete\_Student**();**

cout **<<** endl**;**

welcome**();**

**}**

**else** **if** **(**command **==** 3**)** **{**

Change\_Student**();**

cout **<<** endl**;**

welcome**();**

**}**

**else** **if** **(**command **==** 4**)** **{**

Show\_Students**();**

cout **<<** endl**;**

welcome**();**

**}**

**else** **if** **(**command **==** 5**)** **{**

Find\_Student**();**

cout **<<** endl**;**

welcome**();**

**}**

**else** **if** **(**command **==** 6**)** **{**

cout **<<** "\033[2J\033[1;1H"**;**

Task ttask**;**

ttask**.**task\_menu**();**

cout **<<** endl**;**

welcome**();**

**}**

**else** **if** **(**command **==** 7**)** **{**

cout **<<** "\033[2J\033[1;1H"**;**

Marks\_menu**();**

**}**

**else** **if** **(**command **==** 8**)** **{**

cout **<<** "\033[2J\033[1;1H"**;**

welcome**();**

**}**

**else** **if** **(**command **==** 9**)** **{**

**return** 0**;**

**}**

**return** 0**;**

**}**

void Marks\_menu**()** **{**

cout **<<** "[1] - Добавить Оценки Студента (По Студенческому Билету)" **<<** endl**;**

cout **<<** "[2] - Удалить Оценки Студента (По Студенческому Билету)" **<<** endl**;**

cout **<<** "[3] - Изменить Оценки Студента (По Студенческому Билету)" **<<** endl**;**

cout **<<** "[4] - Вывести весь список оценок" **<<** endl**;**

cout **<<** "[5] - Назад" **<<** endl**;**

cout **<<** " > "**;**

**int** command **=** protect\_marks**();**

switch **(**command**)** **{**

case 1**:** **{**

Stud\_marks Marks**;**

Marks**.**add\_mark**();**

cout **<<** endl**;**

Marks\_menu**();**

**break;**

**}**

case 2**:** **{**

Stud\_marks Marks**;**

Marks**.**delete\_mark**();**

cout **<<** endl**;**

Marks\_menu**();**

**break;**

**}**

case 3**:** **{**

cout **<<** endl**;**

Stud\_marks Marks**;**

Marks**.**Change\_Marks**();**

Marks\_menu**();**

**break;**

**}**

case 4**:** **{**

Stud\_marks Marks**;**

Marks**.**Show\_Marks**();**

cout **<<** endl**;**

Marks\_menu**();**

**break;**

**}**

case 5**:** **{**

cout **<<** "\033[2J\033[1;1H"**;**

welcome**();**

**break;**

**}**

**}**

**}**

void Show\_Students**()** **{**

ifstream **input(**"Base.txt"**);**

cout **<<** endl**;**

**int** counter **=** 0**;**

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

**input** **>>** gradebook\_number **>>** name **>>** surname **>>** patronymic **>>** gen\_der **>>** day **>>** month **>>** year **>>** year\_of\_receipt **>>** faculty **>>** department **>>** group**;**

counter**++;**

cout **<<** "(" **<<** counter **<<** ")" **<<** endl**;**

cout **<<** "ФИО(пол): " **<<** surname **<<** " " **<<** name **<<** " " **<<** patronymic **<<** " (" **<<** gen\_der **<<** ") " **<<** endl**;**

cout **<<** "Дата Рождения, Дата поступления: " **<<** day **<<** " " **<<** month **<<** " " **<<** year **<<** ", " **<<** year\_of\_receipt **<<** endl**;**

cout **<<** "Номер Студенческого Билета, Группа, Факультет, Кафедра: " **<<** gradebook\_number **<<** " " **<<** group **<<** " " **<<** faculty **<<** " " **<<** department **<<** endl **<<** endl**;**

**}**

**input.**close**();**

**}**

void Delete\_Student**()** **{**

ifstream **input(**"Base.txt"**,** std**::**ios**::**app**);**

ofstream output**(**"Replaser.txt"**);**

cout **<<** endl **<<** "Введите Номер Студенческого Билета: " **<<** endl**;**

cin **>>** number**;**

clear\_all**();**

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

**input** **>>** gradebook\_number **>>** name **>>** surname **>>** patronymic **>>** gen\_der **>>** day **>>** month **>>** year **>>** year\_of\_receipt **>>** faculty **>>** department **>>** group**;**

**int** tmp**(**1**);**

**for** **(int** i **=** 0**;** i **<** 7**;** i**++)** **{**

**if** **(**gradebook\_number**[**i**]** **!=** number**[**i**])** **{**

tmp **=** 0**;**

**break;**

**}**

**}**

**if** **(**tmp **==** 0**)** **{**

output **<<** gradebook\_number **<<** " " **<<** name **<<** " " **<<** surname **<<** " " **<<** patronymic **<<** " " **<<** gen\_der **<<** " " **<<** day **<<** " " **<<** month **<<** " " **<<** year **<<** " " **<<** year\_of\_receipt **<<** " " **<<** faculty **<<** " " **<<** department **<<** " " **<<** group **<<** endl**;**

**}**

**}**

ifstream if\_rewrite**(**"Replaser.txt"**,** std**::**ios**::**app**);**

ofstream of\_rewrite**(**"Base.txt"**);**

count **-=** 1**;**

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

if\_rewrite **>>** gradebook\_number **>>** name **>>** surname **>>** patronymic **>>** gen\_der **>>** day **>>** month **>>** year **>>** year\_of\_receipt **>>** faculty **>>** department **>>** group**;**

of\_rewrite **<<** gradebook\_number **<<** " " **<<** name **<<** " " **<<** surname **<<** " " **<<** patronymic **<<** " " **<<** gen\_der **<<** " " **<<** day **<<** " " **<<** month **<<** " " **<<** year **<<** " " **<<** year\_of\_receipt **<<** " " **<<** faculty **<<** " " **<<** department **<<** " " **<<** group **<<** endl**;**

**}**

cout **<<** "Студент Удален Из Списка" **<<** endl**;**

if\_rewrite**.**close**();**

of\_rewrite**.**close**();**

**input.**close**();**

output**.**close**();**

**}**

void Change\_Student**()** **{**

ifstream **input(**"Base.txt"**,** std**::**ios**::**app**);**

ofstream output**(**"Replaser.txt"**);**

cout **<<** endl **<<** "Введите Номер Студенческого Билета: " **<<** endl**;**

cin **>>** number**;**

clear\_all**();**

cout **<<** endl**;**

cout **<<** "Что Вы Хотите Изменить? " **<<** endl **<<** endl**;**

cout **<<** "[1] ФИО, Пол" **<<** endl**;**

cout **<<** "[2] {День, Месяц, Год} - Рождения и Год Поступления" **<<** endl**;**

cout **<<** "[3] Факультет, Кафедра, Группа" **<<** endl**;**

cout **<<** "[4] Номер Студенческого Билета" **<<** endl**;**

cout **<<** " > "**;**

command **=** protect\_change**();**

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

**input** **>>** gradebook\_number **>>** name **>>** surname **>>** patronymic **>>** gen\_der **>>** day **>>** month **>>** year **>>** year\_of\_receipt **>>** faculty **>>** department **>>** group**;**

**int** tmp**(**1**);**

**for** **(int** i **=** 0**;** i **<** 7**;** i**++)** **{**

**if** **(**gradebook\_number**[**i**]** **!=** number**[**i**])** **{**

tmp **=** 0**;**

**break;**

**}**

**}**

**if** **(**tmp **==** 1**)** **{**

switch **(**command**)** **{**

case 1**:**

**{**

cout **<<** "Введите ФИО и Пол{M/F} Студента: " **<<** endl**;**

cin **>>** surname **>>** name **>>** patronymic **>>** gen\_der**;**

clear\_all**();**

**break;**

**}**

case 2**:**

**{**

cout **<<** "Введите {День, Месяц, Год} - Рождения и Год По-ступления: " **<<** endl**;**

day **=** protect\_day**();**

month **=** protect\_month**();**

year **=** protect\_year**();**

year\_of\_receipt **=** protect\_year**();**

**break;**

**}**

case 3**:**

**{**

cout **<<** "Введите Название Факультета, Кафедры, Группы" **<<** endl**;**

cin **>>** faculty **>>** department **>>** group**;**

clear\_all**();**

**break;**

**}**

case 4**:**

**{**

cout **<<** "Введите Новый Номер Студенческого Билета" **<<** endl**;**

cin **>>** gradebook\_number**;**

clear\_all**();**

**break;**

**}**

**}**

output **<<** gradebook\_number **<<** " " **<<** name **<<** " " **<<** surname **<<** " " **<<** patronymic **<<** " " **<<** gen\_der **<<** " " **<<** day **<<** " " **<<** month **<<** " " **<<** year **<<** " " **<<** year\_of\_receipt **<<** " " **<<** faculty **<<** " " **<<** department **<<** " " **<<** group **<<** endl**;**

**}**

**else** output **<<** gradebook\_number **<<** " " **<<** name **<<** " " **<<** surname **<<** " " **<<** patronymic **<<** " " **<<** gen\_der **<<** " " **<<** day **<<** " " **<<** month **<<** " " **<<** year **<<** " " **<<** year\_of\_receipt **<<** " " **<<** faculty **<<** " " **<<** department **<<** " " **<<** group **<<** endl**;**

**}**

ifstream if\_rewrite**(**"Replaser.txt"**,** std**::**ios**::**app**);**

ofstream of\_rewrite**(**"Base.txt"**);**

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

if\_rewrite **>>** gradebook\_number **>>** name **>>** surname **>>** patronymic **>>** gen\_der **>>** day **>>** month **>>** year **>>** year\_of\_receipt **>>** faculty **>>** department **>>** group**;**

of\_rewrite **<<** gradebook\_number **<<** " " **<<** name **<<** " " **<<** surname **<<** " " **<<** patronymic **<<** " " **<<** gen\_der **<<** " " **<<** day **<<** " " **<<** month **<<** " " **<<** year **<<** " " **<<** year\_of\_receipt **<<** " " **<<** faculty **<<** " " **<<** department **<<** " " **<<** group **<<** endl**;**

**}**

cout **<<** "Студент Изменен" **<<** endl**;**

if\_rewrite**.**close**();**

of\_rewrite**.**close**();**

**input.**close**();**

output**.**close**();**

**}**

void Find\_Student**()** **{**

ifstream **input(**"Base.txt"**,** std**::**ios**::**app**);**

**int** C\_tmp**(**0**);**

cout **<<** endl **<<** "Введите Номер Студенческого Билета: " **<<** endl**;**

cin **>>** number**;**

clear\_all**();**

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

**input** **>>** gradebook\_number **>>** name **>>** surname **>>** patronymic **>>** gen\_der **>>** day **>>** month **>>** year **>>** year\_of\_receipt **>>** faculty **>>** department **>>** group**;**

**int** tmp**(**1**);**

**for** **(int** i **=** 0**;** i **<** 7**;** i**++)** **{**

**if** **(**gradebook\_number**[**i**]** **!=** number**[**i**])** **{**

tmp **=** 0**;**

**break;**

**}**

**}**

**if** **(**tmp **==** 1**)** **{**

cout **<<** endl **<<** "Имя Студента: "**;**

cout **<<** name**;**

cout **<<** endl **<<** "Фамилия Студента: "**;**

cout **<<** surname**;**

cout **<<** endl **<<** "Отчество Студента: "**;**

cout **<<** patronymic**;**

cout **<<** endl **<<** "Пол Студента: "**;**

cout **<<** gen\_der**;**

cout **<<** endl **<<** "Дата Рождения Студента {Д,М,Г}: "**;**

cout **<<** day **<<** " " **<<** month **<<** " " **<<** year**;**

cout **<<** endl **<<** "Дата Поступления Студента: "**;**

cout **<<** year\_of\_receipt**;**

cout **<<** endl **<<** "Факультет Студента: "**;**

cout **<<** faculty**;**

cout **<<** endl **<<** "Кафедра Студента: "**;**

cout **<<** department**;**

cout **<<** endl **<<** "Группа Студента: "**;**

cout **<<** group**;**

cout **<<** endl **<<** "Номер Студенческого Билета: "**;**

cout **<<** gradebook\_number**;**

cout **<<** endl **<<** endl**;**

C\_tmp **=** 1**;**

**break;**

**}**

**}**

**input.**close**();**

**if** **(**C\_tmp **==** 0**)** cout **<<** "Студент Не Найден" **<<** endl **<<** endl**;**

**}**

void clear\_all**()** **{**

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

cin**.**clear**();**

**}**

**~**Menu**()** **{**

**}**

**};**

**int** main**()** **{**

Menu menu**;**

menu**.**welcome**();**

**return** 0**;**

**}**