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

Институт комплексной безопасности и специального приборостроения

Кафедра КБ-3 «Управление и моделирование систем»

КУРСОВОЙ ПРОЕКТ (РАБОТА)

по дисциплине «Языки программирования»

(наименование дисциплины)

Тема курсового проекта Разбить группу на 3 части: отличников, хорошистов и троечников, по сессии вводимой по желанию пользователя за всё время обучения

Студент группы БАСО-02-20 Андреев Игорь Константинович

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

Руководитель курсового проекта Русаков Алексей Михайлович

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

Рецензент (при наличии) Мерсов Андрей Александрович

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

Работа представлена к защите « » 2020 г.

Допущен к защите « » 2020 г.

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

Институт комплексной безопасности и специального приборостроения

Кафедра КБ-3 «Управление и моделирование систем»

Утверждаю

Заведующий кафедрой КБ-3

Пушкин П.Ю.

(Фамилия, Имя, Отчество) (подпись).

« » 2020 г.

ЗАДАНИЕ

на выполнение курсового проекта (работы)по дисциплине

«Языки программирования»

Студент Андреев Игорь Константинович Группа БАСО-02-20

Тема:

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

Перечень вопросов, подлежащих разработке и обязательного графического материала:

Срок представления к защите курсового проекта (работы) до «\_\_\_\_\_» 2020 г.

Задание на выполнение курсовой проект (работу) выдал Русаков А.М. ( )

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

Задание на курсовой проект (работу) получил Андреев И.К ( )

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

Содержание

[Введение 4](#_Toc73808491)

[1. Исходные данные 5](#_Toc73808492)

[2. Алгоритм решения задачи 6](#_Toc73808493)

[3. Описание классов и их методов 7](#_Toc73808494)

[4. Наследование, инкапсуляция, полиморфизм 10](#_Toc73808495)

[5. Блок-схема программы 13](#_Toc73808496)

[6. Анализ результатов 14](#_Toc73808497)

[7. Руководство пользователя 23](#_Toc73808498)

[Вывод 24](#_Toc73808499)

[Список используемых источников 25](#_Toc73808500)

[Листинг 26](#_Toc73808501)

# **Введение**

Задачей курсовой работы является разработка программы поиска всех данных о студентах, которые успевают с наибольшим и наименьшим успехом, с поиском среди лиц определенного пола.

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.

# **Исходные данные**

Информация о группе студентов из N человек, где запись о студенте содержит данные представленные на таблице 1.

|  |  |
| --- | --- |
| Основная структура данных | Тип Данных |
| Ф.И.О. студента | Строковый |
| Пол студента | Целочисленный |
| Число, месяц, год рождения | Целочисленный |
| Год поступления в институт | Целочисленный |
| Факультет | Строковый |
| Кафедра | Строковый |
| Группа | Строковый |
| Номер зачетной книжки | Строковый |
| Название предмета | Строковый |
| Оценка | Целочисленный |

Таблица 1 – исходные форматные данные

(Вариант 18) Разбить группу на 3 части:

1) отличников;

2) хорошистов;

3) троечников

по сессии, вводимой по желанию пользователя за все время обучения.

# **Алгоритм решения задачи**

Задача реализуется с помощью использования классов, таких как Studenet, FileManager, MenuPattern. В данных классах описаны методы решения поставленной задачи. В таблице 2 продемонстрирована роль используемых классов.

|  |  |
| --- | --- |
| Класс | Назначение |
| Student | Хранит в себе данные о студенте |
| FileManager | Выполняет необходимые действия с файлом |
| MenuPattern | Отвечает за выбор опций пользователем |

Таблица 2 – роль используемых классов

# **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 3 представлены поля и методы класса, используемые в проекте. Также описан смысл методов каждого класса, для каких целей он создан и результат их работы. Класс Поле класса

|  |  |  |
| --- | --- | --- |
| Класс | Поля класса | Методы класса |
| Student | FullNameTemplate fullName // имя, фамилия, отчество  BirthDataTemplate birthData // день, месяц, год  Int startYear // Год поступления  String faculty // Факультет  String department // кафедра  String group // группа  String recordBookNumber //номер зачётной книжки  Char sex // пол  SubjectTemplate subjects[SESSIONNUMBER][SUBJECTNUBER] // Двумерный массив, хранящий названия и оценки предметов по всем сессиям  Vector<string> booksData // Хранит номера зачётных книжек студентов  Public: static int studentCount // Счётчик студентов | addStudentData() // Добавление данных о студенте  nameValidation() // Проверка имени на валидность  birthDateValidation() // Проверка даты на валидность  recordBookValidation() // Проверка уникальности номера зачётной книжки  markValidation(int,int) // проверка оценки на валидность  addSubjects(int,int)// Добавление данных о предмете и проверка валидноси  virtual void printInfo() // Вывод информации о студенте на экран |
| FileManager | Ofstream fileWrite // Работа с чтением файла  Ifstream fileRead //Работа счтением файла  Int curLine // Счётчик строк файла  Vector<int> v3 // Массив Данных о троечниках  Vector<int> v4 // Массив Данных о хорошистах  Vector<int> v5 // Массив Данных об отличниках  Vector<string> oldData //Временный массив для записи информации из удаляемого файла | openToWrite() // Открыть файл для записи  openToRead(int) // Открыть файл для чтения  createNewTemporaryFile(int) // Создать временный файл для записи  writeInFile() // Записать данные в файл  printInfo(int, int) // Вывести данные из файла от a до b строки  string getInfo(int) // Получить строку из файла по номеру строки  printSessionInfo(int,int) // Вывести информацию об успеваемости  int searchData(string) // Поиск строки в файле, возвращает номер строки  countStudentPerfomance() // Вычисляет средний балл студентов  showStudentPerfomance(int) // Вывести на экран данные об успеваемости  int countAverageNumber // преобразовывает строку в число – оценку  int getStudentNumber(int) // Получить порядковый номер студента  int checkForStudentCount() // Проверить кол-во студентов в файле  changeStudentData(int) // Изменить данные студента  saveOldData(int) // Сохранить старые данные за искл. некоторых  copyFromOldData // Записать в файл старую информацию  deleteRecordBookNumber(string) // Удалить номер зачётной книжки из реестра  checkRBNumber(int) //Занести в реестр номера зачетных книжек из файла  killStudent(int) // Удалить студента из файла  clearStudentPerfomance(int) // Удалить данные об успеваемости студента |
| MenuPattern | String tRecordBookNumber // Переменная для временной записи номера зачетной книжки  FullNaeTemplate // Переменная для временной записи ФИО  Int menu // Переменная выбора опций  Int flag // Проверка выполнения условий | Int startMenu() // Выбор опций в меню |

# **Наследование, инкапсуляция, полиморфизм**

Наследование рассматривается как один из основополагающих принципов ООП. В соответствии с ним, классы могут использовать переменные и методы другого класса как свои собственные. В данном проекте присутствует наследование классов, показанное на рисунке 1.

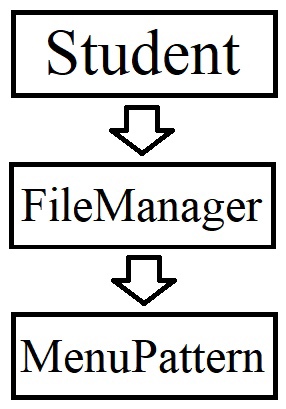
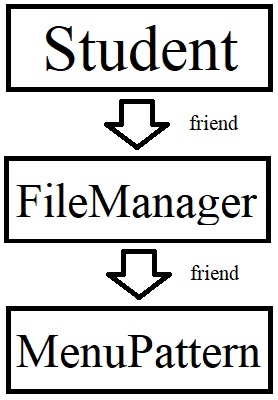


Рисунок 1 – наследование классов

Одно из основных преимуществ ООП - инкапсуляция данных в классе. 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 – друзья классов



На рисунке 2 показаны дружественные классы. Class MenuPattern является дружественным классу FileManager, а тот в свою очередь является, но дружественным классу Student.

Следующим принципом ООП является полиморфизм. Виртуальные функции в комбинации с производными типами позволяют языку С++ поддерживать полиморфизм времени исполнения. Виртуальная функция — это функция, объявленная с ключевым словом virtual в базовом классе и переопределенная в одном или в нескольких производных классах. В свою очередь, класс, содержащий одну или более виртуальных функций, называется полиморфным классом (polymorphic class). В данном проекте продемонстрирован полиморфный класс Student, в котором присутствует виртуальная функция «printInfo();», которая переопределена в производном классе FileManager. Этот полиморфизм важен для объектно-ориентированного программирования, поскольку он позволяет переопределять функции базового класса в классах-потомках с тем, чтобы иметь их версию применительно к данному конкретному классу. Таким образом, базовый класс определяет общий интерфейс, который имеют все производные от него классы, и вместе с тем полиморфизм позволяет производным классам иметь свои собственные реализации методов.

# **Блок-схема программы**

Структурная блок-схема программы представлена на рисунке 3

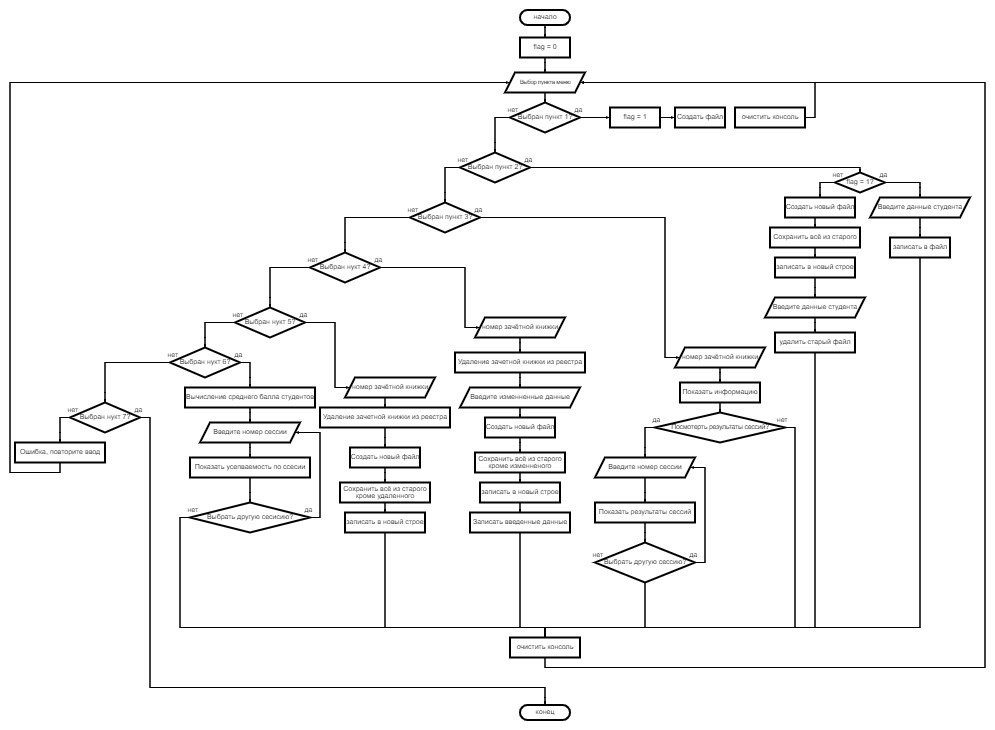


Рисунок 3 – структурная блок-схема программы

# **Анализ результатов**

Входные данные представлены в таблице 4

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ФИО | Пол | Д.Р. | № з.к. | Г.П. | Группа | Факультет | кафедра |
| 1 | Гущин Севастьян Антонович | М | 07.11.1996 | 19Б5647 | 2020 | БАСО-02 | КБСП | КБ-1 |
| 2 | Некрасов Исаак Борисович | М | 12.09.1999 | 19Б9230 | 2021 | БАСО-03 | КБСП | КБ-1 |
| 3 | Кулагин Мечислав Антонович | М | 12.11.1994 | 19Б9638 | 2020 | БАСО-04 | КБСП | КБ-2 |
| 4 | Соколова Элина Степановна | Ж | 05.10.2002 | 19Б7583 | 2019 | БАСО-02 | КБСП | КБ-1 |
| 5 | Богданова Жанна Еремеевна | Ж | 26.10.2001 | 19Б6439 | 2018 | БББО-03 | МФТИ | КБ-3 |

Таблица 4 – исходные данные

Зачетная книжка студента №1 представлена в таблице 5.1

|  |  |  |
| --- | --- | --- |
| Семестр | Предметы | Оценка |
| 1 | Мат. анализ | 5 |
| Линейная алгебра | 5 |
| Физика | 4 |
| ЯП | 4 |
| Английский | 5 |
| 2 | Мат. анализ | 5 |
| Английский | 5 |
| ЭВМ | 3 |
| СИБ | 4 |
| Электротехника | 5 |
| 3 | Мат. Анализ | 5 |
| СИБ | 4 |
| Электротехника | 5 |
| ЯП | 5 |
| Английский | 4 |

Таблица 5.1 – зачетная книжка студента номер 1

Зачетная книжка студента №2 представлена в таблице 5.2

|  |  |  |
| --- | --- | --- |
| Семестр | Предметы | Оценка |
| 1 | Мат. анализ | 3 |
| Линейная алгебра | 2 |
| Физика | 5 |
| ЯП | 5 |
| Английский | 3 |
| 2 | Мат. анализ | 3 |
| Английский | 2 |
| ЭВМ | 4 |
| СИБ | 4 |
| Электротехника | 2 |
| 3 | Мат. Анализ | 3 |
| СИБ | 4 |
| Электротехника | 3 |
| ЯП | 5 |
| Английский | 3 |

Таблица 5.2 – зачетная книжка студента номер 2

Зачетная книжка студента №3 представлена в таблице 5.3

|  |  |  |
| --- | --- | --- |
| Семестр | Предметы | Оценка |
| 1 | Мат. анализ | 4 |
| Линейная алгебра | 4 |
| Физика | 4 |
| ЯП | 4 |
| Английский | 4 |
| 2 | Мат. анализ | 4 |
| Английский | 4 |
| ЭВМ | 4 |
| СИБ | 4 |
| Электротехника | 4 |
| 3 | Мат. Анализ | 4 |
| СИБ | 4 |
| Электротехника | 4 |
| ЯП | 4 |
| Английский | 4 |

Таблица 5.3 – зачетная книжка студента номер 3

Зачетная книжка студента №4 представлена в таблице 5.4

|  |  |  |
| --- | --- | --- |
| Семестр | Предметы | Оценка |
| 1 | Мат. анализ | 5 |
| Линейная алгебра | 5 |
| Физика | 5 |
| ЯП | 5 |
| Английский | 5 |
| 2 | Мат. анализ | 5 |
| Английский | 5 |
| ЭВМ | 5 |
| СИБ | 5 |
| Электротехника | 5 |
| 3 | Мат. Анализ | 5 |
| СИБ | 5 |
| Электротехника | 5 |
| ЯП | 5 |
| Английский | 5 |

Таблица 5.4 – зачетная книжка студента номер 4

Зачетная книжка студента №5 представлена в таблице 5.5

|  |  |  |
| --- | --- | --- |
| Семестр | Предметы | Оценка |
| 1 | Мат. анализ | 5 |
| Линейная алгебра | 4 |
| Физика | 3 |
| Экономика | 3 |
| Английский | 5 |
| 2 | Мат. анализ | 4 |
| Английский | 5 |
| Философия | 5 |
| Правоведение | 4 |
| История | 5 |
| 3 | Мат. Анализ | 5 |
| Экономика | 4 |
| Правоведение | 5 |
| История | 5 |
| Английский | 4 |

Таблица 5.5 – зачетная книжка студента номер 5

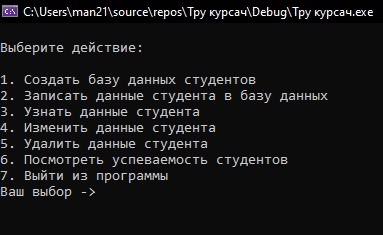
При запуске программы в консоли выводится меню, что и представлено на рисунке 4.1.

Рисунок 4.1. – Меню программы

На рисунке 4.2. продемонстрирован вывод данных после добавления студентов.

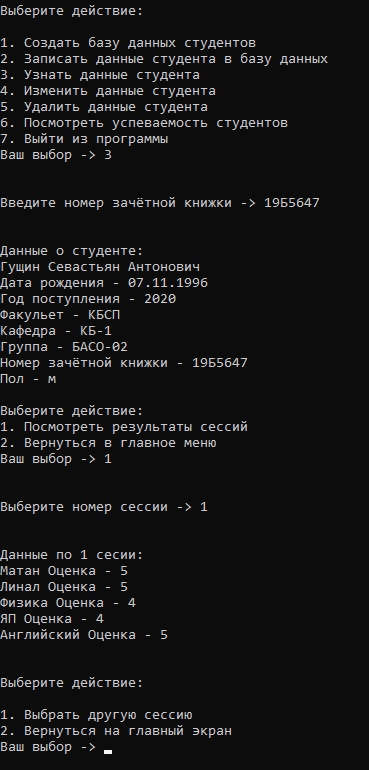


Рисунок 4.2. – Отображение информации о студенте

Далее делаем выборку в соответствии с вариантом, что и показано на рисунке 4.3.

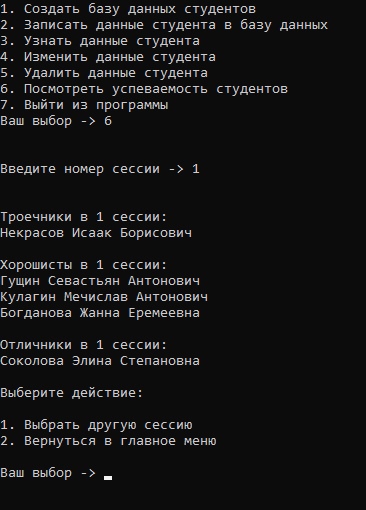


Рисунок 4.3. – Выборка варианта №18

Убедимся, что программа работает исправно и подсчитаем средний балл у каждого студента в 1 сессии:

1. (5 + 5 + 4 + 4 + 5) / 5 = 4,6 – хорошист
2. (3 + 2 + 5 + 5 + 3) / 5 = 3,6 – троечник
3. (4 + 4 + 4 + 4 + 4) / 5 = 4 – хорошист
4. (5 + 5 + 5 + 5 + 5) / 5 = 5 – отличница
5. (5 + 4 + 3 + 3 + 5) / 5 = 4 – хорошистка

Выходные данные:

Троечники в 1 сессии:

* Некрасов Исаак Борисович (студент номер 2)

Хорошисты в 1 сессии:

* Гущин Севастьян Антонович (студент номер 1)
* Кулагин Мечислав Антонович (студент номер 3)
* Богданова Жанна Еремеевна (студент номер 5)

Отличники в 1 сессии:

* Соколова Элина Степановна (студент номер 4)

# **Руководство пользователя**

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

При вводе «1» создаётся файл, открытый для чтения и записи, после чего в консоль выводится запись об успешном создании.

При вводе «2» в консоль предлагается ввести данные о студенте, после чего будет даваться выбор – добавить предмет или прейти к следующей сессии. По окончанию, на консоль выведется запись об успешном добавлении данных.

При вводе «3» в консоль предлагается ввести номер зачётной книжки студента. Если такой не нашёлся, пользователя вернет в главное меню. В обратном случае в консоль будет выведена информация о найденном студенте, после чего будет предложен выбор – выйти в главное меню, или отобразить информацию о сессии по выбору пользователя.

При вводе «4» в консоль предлагается ввести номер зачётной книжки студента. Если такой не нашёлся, то аналогично пункту «3», пользователя вернет в главное меню. В обратном случае будет предложено заново вести данные студенте, после чего новые данные сохранятся в файл, и в консоль будет выведена запись об успешном выполнении.

При вводе «5» в консоль предлагается ввести номер зачётной книжки студента. Если такой не нашёлся, то аналогично пунктам «3» и «4», пользователя вернет в главное меню. В обратном случае, из файла удалятся данные о студенте, и в консоль будет выведена запись об успешном выполнении.

При вводе «6» в консоль предлагается ввести номер сессии, по которой будет производиться поиск, после чего будет посчитан средний балл студентов и в консоль будет выведена информация о них.

При вводе «7» будет осуществлен выход из программы.

# **Вывод**

В процессе выполнения курсовой работы была написана программа поиска всех данных о студентах, которые успевают с наибольшим и наименьшим успехом, с поиском среди лиц определенного пола.

Программа была реализована c использованием технологии ООП.

# **Список используемых источников**

1. Прата С. Язык программирования С. Лекции и упражнения. – Litres,

2021

1. Лафоре Р. Объектно-ориентированное программирование в C++:[пер. с

англ.]. – Издательский дом" Питер", 2013

1. Страуструп Б. Программирование. Принципы и практика

использования C++. – Litres, 2021

1. Ravesli | Программирование для начинающих // URL: https://ravesli.com/

# **Листинг**

#include <iostream>

#include <fstream>

#include <string>

#include <windows.h>

#include <vector>

#include <conio.h>

**using** **namespace** std**;**

const int SESSIONNUMBER **=** 3**;** // Кол-во сессий

const int SUBJECTNUMBER **=** 5**;** // Макс. Кол-во предметов

const int DECREMENT **=** SESSIONNUMBER **\*** **(**SUBJECTNUMBER **+** 1**)** **+** 8**;** // Обозначает кол-во строк отводимых для 1 студента

const int THIS\_YEAR **=** 2021**;**

int mode **=** 0**;**

struct BirthDataTemplate

**{**

int day**;**

int month**;**

int year**;**

**};**

struct SubjectTemplate

**{**

int session**;**

string name**;**

int mark**;**

**};**

struct FullNameTemplate

**{**

string firstName**;**

string secondName**;**

string thirdName**;**

**};**

/// <summary>

/// Хранит в себе все необходимые данные о студенте

/// </summary>

class Student

**{**

friend class FileManager**;**

FullNameTemplate fullName**;**

BirthDataTemplate birthData**;**

int startYear**;**

string faculty**;**

string department**;**

string group**;**

string recordBookNumber**;**

char sex**;**

SubjectTemplate subjects**[**SESSIONNUMBER**][**SUBJECTNUMBER**];**

string subjectNames**[**SUBJECTNUMBER**];**

int subjectsCount **=** 0**;**

vector**<**string**>** booksData**;**

public**:**

static int studentCount**;**

/// <summary>

/// Записывает данные о студенте, считываемые с клавиатуры и заносит данные об их успеваемости по сесииям в массив векторов

/// </summary>

void addStudentData**()**

**{**

studentCount**++;**

check1**:**

cout **<<** "\n\nВведите ФИО студента (через пробел) -> "**;**

cin **>>** fullName**.**firstName**;**

cin**.**ignore**();**

cin **>>** fullName**.**secondName**;**

cin**.**ignore**();**

cin **>>** fullName**.**thirdName**;**

cin**.**ignore**();**

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

**if** **(**nameValidation**()** **==** **-**1**)**

**{**

**goto** check1**;**

**}**

check2**:**

cout **<<** "Введите дату рождения (хх.хх.хххх) ->"**;**

cin **>>** birthData**.**day**;**

cin**.**ignore**();**

cin **>>** birthData**.**month**;**

cin**.**ignore**();**

cin **>>** birthData**.**year**;**

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

**if** **(**birthDateValidation**()** **==** **-**1**)**

**{**

**goto** check2**;**

**}**

check3**:**

cout **<<** "Введите год поступления -> "**;**

cin **>>** startYear**;**

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

**if** **(**startYear **<** birthData**.**year**)**

**{**

cout **<<** "\nВозможно вы ошиблись при вводе пожалуйста повторите попытку!!!\n"**;**

system**(**"pause"**);**

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

**goto** check3**;**

**}**

cout **<<** "Введите факультет -> "**;**

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

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

cout **<<** "Введите кафедру -> "**;**

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

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

cout **<<** "Введите группу -> "**;**

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

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

check4**:**

cout **<<** "Введите номер зачётной книжки -> "**;**

cin **>>** recordBookNumber**;**

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

**if** **(**recordBookValidation**()** **==** **-**1**)**

**{**

**goto** check4**;**

**}**

check5**:**

cout **<<** "Введите пол (одна буква, м/ж, без нажатия enter) -> "**;**

sex **=** \_getch**();**

**if** **((**sex **!=** 'м'**)** **&&** **(**sex **!=** 'ж'**))**

**{**

cout **<<** "\nНерпавильный ввод, пожалуйста повторите попытку\n"**;**

system**(**"pause"**);**

**goto** check5**;**

**}**

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

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

**for** **(**int i **=** 0**;** i **<** SESSIONNUMBER**;** i**++)**

**{**

cout **<<** "Для " **<<** i **+** 1 **<<** " сессии:\n"**;**

subjectsCount **=** 0**;**

**for** **(**int j **=** 0**;** j **<** SUBJECTNUMBER**;** j**++)**

**{**

**if** **(**addSudjects**(**i**,** j**)** **==** 0**)**

**{**

**for** **(**int k **=** j**;** k **<** SUBJECTNUMBER**;** k**++)**

**{**

subjects**[**i**][**j**].**name **=** " "**;**

subjects**[**i**][**j**].**mark **=** 0**;**

**}**

**break;**

**}**

subjectsCount**++;**

**}**

**for** **(**int k **=** 0**;** k **<** SUBJECTNUMBER**;** k**++)**

**{**

subjectNames**[**k**]** **=** " "**;**

**}**

**}**

**}**

/// <summary>

/// Проверка правильности вода ФИО

/// </summary>

/// <returns>Возвращает -1 в случае ошибки, 0 - в случае корректного ввода</returns>

int nameValidation**()**

**{**

string letter**;**

**if** **((**fullName**.**firstName**.**size**()** **<** 2**)** **||** **(**fullName**.**firstName**.**size**()** **>** 12**))**

**{**

cout **<<** "\nВозможно вы ошиблись при вводе имени пожалуйста повторите попытку!!!\n"**;**

system**(**"pause"**);**

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

**return** **-**1**;**

**}**

**if** **((**fullName**.**secondName**.**size**()** **<** 2**)** **||** **(**fullName**.**firstName**.**size**()** **>** 12**))**

**{**

cout **<<** "\nВозможно вы ошиблись при вводе имени пожалуйста повторите попытку!!!\n"**;**

system**(**"pause"**);**

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

**return** **-**1**;**

**}**

**if** **((**fullName**.**thirdName**.**size**()** **<** 2**)** **||** **(**fullName**.**firstName**.**size**()** **>** 12**))**

**{**

cout **<<** "\nВозможно вы ошиблись при вводе имени пожалуйста повторите попытку!!!\n"**;**

system**(**"pause"**);**

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

**return** **-**1**;**

**}**

**for** **(**int i **=** 0**;** i **<** 10**;** i**++)**

**{**

letter **=** to\_string**(**i**);**

**if** **(**fullName**.**firstName**.**find**(**letter**)** **!=** string**::**npos**)**

**{**

cout **<<** "\nВозможно вы ошиблись при вводе имени пожалуйста повторите попытку!!!\n"**;**

system**(**"pause"**);**

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

**return** **-**1**;**

**}**

**if** **(**fullName**.**secondName**.**find**(**letter**)** **!=** string**::**npos**)**

**{**

cout **<<** "\nВозможно вы ошиблись при вводе имени пожалуйста повторите попытку!!!\n"**;**

system**(**"pause"**);**

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

**return** **-**1**;**

**}**

**if** **(**fullName**.**thirdName**.**find**(**letter**)** **!=** string**::**npos**)**

**{**

cout **<<** "\nВозможно вы ошиблись при вводе имени пожалуйста повторите попытку!!!\n"**;**

system**(**"pause"**);**

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

**return** **-**1**;**

**}**

**}**

**return** 1**;**

**}**

/// <summary>

/// Проверяет правильность ввода даты рождения

/// </summary>

/// <returns>Возвращает -1 в случае ошибки, 0 - в случае корректного ввода</returns>

int birthDateValidation**()**

**{**

**if** **((**birthData**.**day **>** 31**)** **||** **(**birthData**.**day **<** 0**))**

**{**

cout **<<** "\nВозможно вы ошиблись при вводе даты пожалуйста повторите попытку!!!\n"**;**

system**(**"pause"**);**

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

**return** **-**1**;**

**}**

**if** **((**birthData**.**month **>** 12**)** **||** **(**birthData**.**month **<** 0**))**

**{**

cout **<<** "\nВозможно вы ошиблись при вводе даты пожалуйста повторите попытку!!!\n"**;**

system**(**"pause"**);**

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

**return** **-**1**;**

**}**

**if** **((**birthData**.**year **<** 1900**)** **||** **(**birthData**.**year **>** THIS\_YEAR**))**

**{**

cout **<<** "\nВозможно вы ошиблись при вводе даты пожалуйста повторите попытку!!!\n"**;**

system**(**"pause"**);**

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

**return** **-**1**;**

**}**

**return** 0**;**

**}**

/// <summary>

/// Проверяет правильность ввода номера зачётной книжки (уникальность)

/// </summary>

/// <returns>Возвращает 0 в случае уникальности номера, -1 - если повторяется</returns>

int recordBookValidation**()**

**{**

**if** **(**studentCount **>** 1**)**

**{**

**for** **(**int i **=** 0**;** i **<** booksData**.**size**();** i**++)**

**{**

**if** **(**booksData**[**i**]** **==** recordBookNumber**)**

**{**

cout **<<** "\nСтудент с таким номером уже существует, проверьте ввод и повторите попытку\n"**;**

system**(**"pause"**);**

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

**return** **-**1**;**

**}**

**}**

booksData**.**push\_back**(**recordBookNumber**);**

**return** 0**;**

**}**

**else**

**{**

booksData**.**push\_back**(**recordBookNumber**);**

**return** 0**;**

**}**

**}**

/// <summary>

/// Проверяет правильность ввода оценки

/// </summary>

/// <param name="i"></param>

/// <param name="j"></param>

/// <returns> Возвращает 0 в случае успешной проверки </returns>

int markValidation**(**int i**,** int j**)**

**{**

**if** **((**subjects**[**i**][**j**].**mark **<** 2**)** **||** **(**subjects**[**i**][**j**].**mark **>** 5**))**

**{**

cout **<<** "\nОшибка ввода, повторите попытку!\n"**;**

system**(**"pause"**);**

**return** **-**1**;**

**}**

**if** **(!(**isdigit**(**subjects**[**i**][**j**].**mark**))** **==** 0**)**

**{**

cout **<<** "\nОшибка ввода, повторите попытку!\n"**;**

system**(**"pause"**);**

**return** **-**1**;**

**}**

**return** 0**;**

**}**

/// <summary>

/// Добавляет название и оценку по одного предмета

/// </summary>

/// <param name="i"></param>

/// <param name="j"></param>

/// <returns> Возвращает 1 в случае успешного добавления </returns>

int addSudjects**(**int i**,** int j**)**

**{**

int indicator**;**

start**:**

cout **<<** "\n\nВыберите действие:\n"**;**

cout **<<** "1. Добавить предмет\n2. Перейти к след. сессии\n"**;**

cout **<<** "Ваш выбор -> "**;**

cin **>>** indicator**;**

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

**if** **(**indicator **==** 1**)**

**{**

checkSName**:**

cout **<<** "\nНазвание предмета -> "**;**

cin**.**ignore**();**

getline**(**cin**,** subjects**[**i**][**j**].**name**);**

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

**for** **(**int k **=** 0**;** k **<** SUBJECTNUMBER**;** k**++)**

**{**

**if** **(**subjects**[**i**][**j**].**name **==** subjectNames**[**k**])**

**{**

cout **<<** "\nПредмет с таким именем уже существует! Повторите попытку\n"**;**

**goto** checkSName**;**

**}**

**}**

subjectNames**[**subjectsCount**]** **=** subjects**[**i**][**j**].**name**;**

checkMark**:**

cout **<<** "Оценку -> "**;**

cin **>>** subjects**[**i**][**j**].**mark**;**

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

**if** **(**markValidation**(**i**,** j**)** **==** **-**1**)**

**{**

cout **<<** "\nНерпавильный ввод, пожалуйста повторите попытку\n"**;**

system**(**"pause"**);**

**goto** checkMark**;**

**}**

**return** 1**;**

**}**

**else** **if** **(**indicator **==** 2**)**

**{**

**return** 0**;**

**}**

**else**

**{**

cout **<<** "\nОшибка ввода повторите попытку!\n"**;**

**goto** start**;**

**}**

**}**

/// <summary>

/// Выводит данные о последнем запомненном студенте

/// </summary>

virtual void printInfo**()**

**{**

cout **<<** "\nДанные о последнем студенте:\n"**;**

cout **<<** fullName**.**firstName **<<** ' ' **<<** fullName**.**secondName **<<** ' ' **<<** fullName**.**thirdName**;**

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

cout **<<** "Дата рождения - " **<<** birthData**.**day **<<** '.' **<<** birthData**.**month **<<** '.' **<<** birthData**.**year**;**

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

cout **<<** "Год поступления - " **<<** startYear**;**

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

cout **<<** "Факульет - " **<<** faculty**;**

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

cout **<<** "Кафедра - " **<<** department**;**

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

cout **<<** "Группа - " **<<** group**;**

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

cout **<<** "Номер зачётной книжки - " **<<** recordBookNumber**;**

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

cout **<<** "Пол - " **<<** sex**;**

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

**for** **(**int i **=** 0**;** i **<** SESSIONNUMBER**;** i**++)**

**{**

cout **<<** "Сессия номер " **<<** i **+** 1 **<<** " Студента номер " **<<** studentCount**;**

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

**for** **(**int j **=** 0**;** j **<** SUBJECTNUMBER**;** j**++)**

**{**

**if** **(**subjects**[**i**][**j**].**mark **==** 0**)**

**{**

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

**}**

**else**

**{**

cout **<<** subjects**[**i**][**j**].**name **<<** " Оценка - " **<<** subjects**[**i**][**j**].**mark**;**

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

**}**

**}**

**}**

**}**

**};**

/// <summary>

/// Создаёт файл, выполняет необходимые действия с ним и закрывает при разрушении

/// </summary>

class FileManager **:** public Student

**{**

ofstream fileWrite**;**

ifstream fileRead**;**

int curLine**;**

string tLine**;**

vector**<**int**>** v3**;**

vector**<**int**>** v4**;** // Вектора, хранящие код хорошистов - младший разряд числа отвечает за номер сессии от 1 до SESSIONNUMBER, а остальные цифры - порядковый номер студента

vector**<**int**>** v5**;**

vector**<**string**>** oldData**;**

friend class MenuPattern**;**

public**:**

/// <summary>

/// Создает БД студентов открытую на запись, и проверяет успешность открытия

/// </summary>

void openToWrite**()**

**{**

studentCount **=** 0**;**

booksData**.**clear**();**

v5**.**clear**();**

v4**.**clear**();**

v3**.**clear**();**

fileWrite**.**open**(**"StudentsData.txt"**);**

**if** **(!(**fileWrite**.**is\_open**()))**

**{**

cout **<<** "\nФайл не был создан!!!\n"**;**

**}**

**else**

**{**

cout **<<** "\nФайл успешно создан!\n\n"**;**

**}**

**}**

/// <summary>

/// Открывает файл для чтения и проверяет успешность открытия

/// </summary>

void openToRead**(**int mode**)**

**{**

**switch** **(**mode**)**

**{**

**case** 0**:**

**{**

fileRead**.**open**(**"StudentsData.txt"**);**

**if** **(!(**fileRead**.**is\_open**()))**

**{**

cout **<<** "\nФайл не был открыт для чтения !!!\n"**;**

system**(**"pause"**);**

**}**

**break;**

**}**

**case** 1**:**

**{**

fileRead**.**open**(**"TemporaryStudentsData.txt"**);**

**if** **(!(**fileRead**.**is\_open**()))**

**{**

cout **<<** "\nФайл не был открыт для чтения !!!\n"**;**

system**(**"pause"**);**

**}**

**break;**

**}**

**}**

**}**

/// <summary>

/// Создаёт временный файл для записи

/// </summary>

void createNewTemporaryFile**(**int mode**)**

**{**

**switch** **(**mode**)**

**{**

**case** 0**:**

**{**

fileWrite**.**open**(**"TemporaryStudentsData.txt"**);**

**if** **(!(**fileWrite**.**is\_open**()))**

**{**

cout **<<** "\nФайл не был создан!!!\n"**;**

**}**

**break;**

**}**

**case** 1**:**

**{**

fileWrite**.**open**(**"StudentsData.txt"**);**

**if** **(!(**fileWrite**.**is\_open**()))**

**{**

cout **<<** "\nФайл не был создан!!!\n"**;**

**}**

**break;**

**}**

**}**

**}**

/// <summary>

/// Записывает в файл данные о студенте, взятые из класса Student

/// </summary>

void writeInFile**()**

**{**

addStudentData**();**

fileWrite **<<** fullName**.**firstName **<<** ' ' **<<** fullName**.**secondName **<<** ' ' **<<** fullName**.**thirdName**;**

fileWrite **<<** endl**;**

fileWrite **<<** "Дата рождения - " **<<** birthData**.**day **<<** '.' **<<** birthData**.**month **<<** '.' **<<** birthData**.**year**;**

fileWrite **<<** endl**;**

fileWrite **<<** "Год поступления - " **<<** startYear**;**

fileWrite **<<** endl**;**

fileWrite **<<** "Факульет - " **<<** faculty**;**

fileWrite **<<** endl**;**

fileWrite **<<** "Кафедра - " **<<** department**;**

fileWrite **<<** endl**;**

fileWrite **<<** "Группа - " **<<** group**;**

fileWrite **<<** endl**;**

fileWrite **<<** "Номер зачётной книжки - " **<<** recordBookNumber**;**

fileWrite **<<** endl**;**

fileWrite **<<** "Пол - " **<<** sex**;**

fileWrite **<<** endl**;**

**for** **(**int i **=** 0**;** i **<** SESSIONNUMBER**;** i**++)**

**{**

fileWrite **<<** "Сессия номер " **<<** i **+** 1 **<<** " Студента номер " **<<** studentCount**;**

fileWrite **<<** endl**;**

**for** **(**int j **=** 0**;** j **<** SUBJECTNUMBER**;** j**++)**

**{**

**if** **(**subjects**[**i**][**j**].**mark **==** 0**)**

**{**

fileWrite **<<** endl**;**

**}**

**else**

**{**

fileWrite **<<** subjects**[**i**][**j**].**name **<<** " Оценка - " **<<** subjects**[**i**][**j**].**mark**;**

fileWrite **<<** endl**;**

**}**

**}**

**}**

**}**

/// <summary>

/// Выводит в консоль данные из файла от startPos до endPos строки

/// </summary>

/// <param name="startPos"> начало вывода данных </param>

/// <param name="endPos"> конец вывода данных </param>

void printInfo**(**int startPos**,** int endPos**)**

**{**

openToRead**(**mode**);**

curLine **=** 0**;**

**if** **(**fileRead**.**is\_open**())**

**{**

fileRead**.**seekg**(**0**,** ios**::**beg**);**

**while** **(**getline**(**fileRead**,** tLine**))**

**{**

**if** **((**curLine **>=** startPos **-** 1**)** **&&** **(**curLine **<** endPos**))**

**{**

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

**}**

curLine**++;**

**}**

fileRead**.**close**();**

**}**

**else**

**{**

cout **<<** "\n Файл закрыт для чтения!!!\n"**;**

**}**

**}**

/// <summary>

/// Выводит строку с веденным номером

/// </summary>

/// <param name="startFilePos"></param>

/// <returns> Возвращает строку </returns>

string getInfo**(**int linePos**)**

**{**

openToRead**(**mode**);**

curLine **=** 0**;**

**if** **(**fileRead**.**is\_open**())**

**{**

fileRead**.**seekg**(**0**,** ios**::**beg**);**

**while** **(**getline**(**fileRead**,** tLine**))**

**{**

**if** **(**curLine **==** linePos**)**

**{**

fileRead**.**close**();**

**return** tLine**;**

**}**

curLine**++;**

**}**

**}**

**else**

**{**

cout **<<** "\n Файл закрыт для чтения!!!\n"**;**

**return** "Error!!!"**;**

**}**

**}**

/// <summary>

/// Выводит на экран информацию о результатах выбранной сессии

/// </summary>

/// <param name="sessionNumber"> Номер сессии </param>

void printSessionInfo**(**int sessionNumber**,** int studentNumber**)**

**{**

string fullLine1 **=** "Сессия номер "**;**

string fullLine2 **=** " Студента номер "**;**

int tNumberLine**;**

fullLine1 **+=** to\_string**(**sessionNumber**);**

fullLine1 **+=** fullLine2**;**

fullLine1 **+=** to\_string**(**studentNumber**);**

tNumberLine **=** searchData**(**fullLine1**);**

cout **<<** "\nДанные по " **<<** sessionNumber **<<** " сесии:\n"**;**

printInfo**(**tNumberLine **+** 1**,** **(**tNumberLine **+** SUBJECTNUMBER**));**

**}**

/// <summary>

/// Ищет в файле строку по введенной строке

/// </summary>

/// <param name="bookNumber"> Строка, которую нужно найти </param>

/// <returns> Возвращает целое число - номер искомой строки, если найдено, в противном случае -1 </returns>

int searchData**(**string fullLine**)**

**{**

openToRead**(**mode**);**

curLine **=** 0**;**

**if** **(**fileRead**.**is\_open**())**

**{**

fileRead**.**seekg**(**0**,** ios**::**beg**);**

**while** **(**getline**(**fileRead**,** tLine**))**

**{**

curLine**++;**

**if** **(**tLine**.**find**(**fullLine**)** **!=** string**::**npos**)**

**{**

fileRead**.**close**();**

**return** curLine**;**

**}**

**}**

**return** **-**1**;**

**}**

**else**

**{**

cout **<<** "\nФайл закрыт для чтения!!!\n"**;**

**return** **-**1**;**

**}**

**}**

/// <summary>

/// Считывает строки с информацией о результатах сессий из файла и заносит данные в вектора в виде кода

/// </summary>

void countStudentsPerfomance**()**

**{**

string fullLine**;**

string fullLine1**;**

int tNumberLine**;**

int averageNumber**;**

**for** **(**int j **=** 0**;** j **<** studentCount**;** j**++)**

**{**

**for** **(**int i **=** 0**;** i **<** SESSIONNUMBER**;** i**++)**

**{**

tNumberLine **=** 0**;**

averageNumber **=** 0**;**

fullLine **=** "Сессия номер "**;**

fullLine1 **=** " Студента номер "**;**

fullLine **+=** to\_string**(**i **+** 1**);**

fullLine **+=** fullLine1**;**

fullLine **+=** to\_string**(**j **+** 1**);**

string temporary **=** getInfo**(**j **\*** DECREMENT**);**

**if** **(**temporary **==** "----------"**)**

**{**

**break;**

**}**

tNumberLine **=** searchData**(**fullLine**);**

**for** **(**int k **=** 0**;** k **<** SUBJECTNUMBER**;** k**++)**

**{**

tLine **=** getInfo**(**tNumberLine **+** k**);**

averageNumber **+=** countAverageNumber**(**tLine**);**

**}**

averageNumber **=** averageNumber **/** SUBJECTNUMBER**;**

**if** **(**averageNumber **==** 5**)**

**{**

v5**.**push\_back**(**i **+** 1 **+** **(**j **+** 1**)** **\*** 10**);**

**}**

**if** **(**averageNumber **==** 4**)**

**{**

v4**.**push\_back**(**i **+** 1 **+** **(**j **+** 1**)** **\*** 10**);**

**}**

**if** **(**averageNumber **==** 3**)**

**{**

v3**.**push\_back**(**i **+** 1 **+** **(**j **+** 1**)** **\*** 10**);**

**}**

**}**

**}**

**}**

/// <summary>

/// Выводит на экран данные о результатах сессии, вводимой с клавиатуры

/// </summary>

void showStudentPerfomance**(**int sessionNumber**)**

**{**

int nameMarker**;**

int Count **=** 0**;**

int code **=** 0**;**

cout **<<** "\nТроечники в " **<<** sessionNumber **<<** " сессии:\n"**;**

**if** **(**v3**.**size**()** **<** 1**)**

**{**

cout **<<** "Отсутствуют\n"**;**

**}**

**else**

**{**

**for** **(**int i **=** 0**;** i **<** v3**.**size**();** i**++)**

**{**

code **=** v3**[**i**]** **%** 10**;**

**if** **(**code **==** sessionNumber**)**

**{**

Count**++;**

code **=** **(**v3**[**i**]** **-** code**)** **/** 10**;**

nameMarker **=** 1 **+** **(**code **-** 1**)** **\*** **(**SESSIONNUMBER **\*** **(**SUBJECTNUMBER **+** 1**)** **+** 8**);**

printInfo**(**nameMarker**,** nameMarker**);**

**}**

**}**

**if** **(**Count **==** 0**)**

**{**

cout **<<** "Отсутствуют\n"**;**

**}**

**}**

Count **=** 0**;**

code **=** 0**;**

cout **<<** "\nХорошисты в " **<<** sessionNumber **<<** " сессии:\n"**;**

**if** **(**v4**.**size**()** **<** 1**)**

**{**

cout **<<** "Отсутствуют\n"**;**

**}**

**else**

**{**

**for** **(**int i **=** 0**;** i **<** v4**.**size**();** i**++)**

**{**

code **=** v4**[**i**]** **%** 10**;**

**if** **(**code **==** sessionNumber**)**

**{**

Count**++;**

code **=** **(**v4**[**i**]** **-** code**)** **/** 10**;**

nameMarker **=** 1 **+** **(**code **-** 1**)** **\*** **(**SESSIONNUMBER **\*** **(**SUBJECTNUMBER **+** 1**)** **+** 8**);**

printInfo**(**nameMarker**,** nameMarker**);**

**}**

**}**

**if** **(**Count **==** 0**)**

**{**

cout **<<** "Отсутствуют\n"**;**

**}**

**}**

Count **=** 0**;**

code **=** 0**;**

cout **<<** "\nОтличники в " **<<** sessionNumber **<<** " сессии:\n"**;**

**if** **(**v5**.**size**()** **<** 1**)**

**{**

cout **<<** "Отсутствуют\n"**;**

**}**

**else**

**{**

**for** **(**int i **=** 0**;** i **<** v5**.**size**();** i**++)**

**{**

code **=** v5**[**i**]** **%** 10**;**

**if** **(**code **==** sessionNumber**)**

**{**

Count**++;**

code **=** **(**v5**[**i**]** **-** code**)** **/** 10**;**

nameMarker **=** 1 **+** **(**code **-** 1**)** **\*** **(**SESSIONNUMBER **\*** **(**SUBJECTNUMBER **+** 1**)** **+** 8**);**

printInfo**(**nameMarker**,** nameMarker**);**

**}**

**}**

**if** **(**Count **==** 0**)**

**{**

cout **<<** "Отсутствуют\n"**;**

**}**

**}**

**}**

/// <summary>

/// Преобразует строку string типа: Предмет - оценка, в одно целое число int - оценку по предмету

/// </summary>

/// <returns> Возвращает целое число - оценку по предмету </returns>

int countAverageNumber**(**string line**)**

**{**

**if** **(**line**.**find**(**'5'**)** **!=** string**::**npos**)**

**{**

**return** 5**;**

**}**

**else** **if** **(**line**.**find**(**'4'**)** **!=** string**::**npos**)**

**{**

**return** 4**;**

**}**

**else** **if** **(**line**.**find**(**'3'**)** **!=** string**::**npos**)**

**{**

**return** 3**;**

**}**

**else** **if** **(**line**.**find**(**'2'**)** **!=** string**::**npos**)**

**{**

**return** 2**;**

**}**

**else**

**{**

cout **<<** "\nОшибка!!! Введена неверная строка в countAveraheNumber()\n"**;**

system**(**"pause"**);**

**return** 0**;**

**}**

**}**

/// <summary>

/// Вычисляет порядковый номер студента по строке с его ФИО

/// </summary>

/// <returns> Возвращает порядковый номер студента </returns>

int getStudentNumber**(**int startFilePos**)**

**{**

int count **=** 1**;**

**if** **(**startFilePos **==** 1**)**

**{**

**return** 1**;**

**}**

**else** **if** **(**startFilePos **>** 1**)**

**{**

**while** **(**startFilePos **!=** 1**)**

**{**

startFilePos **-=** DECREMENT**;**

count**++;**

**}**

**return** count**;**

**}**

**else**

**{**

cout **<<** "\nОшибка! Задана неверная строка при получении номера студента!\n"**;**

**return** **-**1**;**

**}**

**}**

/// <summary>

/// Проверяет есть ли в базе данных студенты, если да, то возвращает их кол-во

/// </summary>

/// <returns> Вовзвращает кол-во студентов в БД </returns>

int checkForStudentCount**()**

**{**

openToRead**(**mode**);**

curLine **=** 0**;**

**if** **(**fileRead**.**is\_open**())**

**{**

fileRead**.**seekg**(**0**,** ios**::**beg**);**

**while** **(**getline**(**fileRead**,** tLine**))**

**{**

curLine**++;**

**}**

curLine **=** curLine **/** DECREMENT**;**

fileRead**.**close**();**

**return** curLine**;**

**}**

**else**

**{**

cout **<<** "\nФайл закрыт для чтения!!!\n"**;**

**return** **-**1**;**

**}**

**}**

/// <summary>

/// Создаёт временный файл, помещает туда изменненные данные, затем удаляет старый файл

/// </summary>

void changeStudentData**(**int line**)**

**{**

int studentNumber**;**

int tStudentCount**;**

studentNumber **=** getStudentNumber**(**line**);**

fileWrite**.**close**();**

createNewTemporaryFile**(**mode**);**

oldData**.**clear**();**

saveOldData**(**line**);**

tStudentCount **=** studentCount**;**

studentCount **=** studentNumber **-** 1**;**

**if** **(**studentNumber **==** 1**)**

**{**

writeInFile**();**

copyFromOldData**(**0**,** oldData**.**size**());**

**}**

**else** **if** **(**studentNumber **==** tStudentCount**)**

**{**

copyFromOldData**(**0**,** oldData**.**size**());**

writeInFile**();**

**}**

**else**

**{**

copyFromOldData**(**0**,** line **-** 1**);**

writeInFile**();**

copyFromOldData**(**line **-** 1**,** oldData**.**size**());**

**}**

**if** **(**mode **==** 0**)**

**{**

mode **=** 1**;**

remove**(**"StudentsData.txt"**);**

**}**

**else**

**{**

mode **=** 0**;**

remove**(**"TemporaryStudentsData.txt"**);**

**}**

studentCount **=** tStudentCount**;**

**}**

/// <summary>

/// Сохраняет неизмененные данные из файла

/// </summary>

/// <param name="startPos"> номер строки, начниная с которой данные изменены, если передать -1, то все данные будут скопированы </param>

void saveOldData**(**int startPos**)**

**{**

curLine **=** 0**;**

openToRead**(**mode**);**

**if** **(**fileRead**.**is\_open**())**

**{**

fileRead**.**seekg**(**0**,** ios**::**beg**);**

**if** **(**startPos **!=** **-**1**)**

**{**

**while** **(**getline**(**fileRead**,** tLine**))**

**{**

**if** **((**curLine **<** startPos **-** 1**)** **||** **(**curLine **>=** startPos **+** DECREMENT **-** 1**))**

**{**

oldData**.**push\_back**(**tLine**);**

**}**

curLine**++;**

**}**

**}**

**else**

**{**

**while** **(**getline**(**fileRead**,** tLine**))**

**{**

oldData**.**push\_back**(**tLine**);**

**}**

**}**

fileRead**.**close**();**

**}**

**else**

**{**

cout **<<** "\nФайл закрыт для чтения!!!\n"**;**

**}**

**}**

/// <summary>

/// Помещает в файл неизменненные данные из старого файла от start до end строки

/// </summary>

void copyFromOldData**(**int start**,** int end**)**

**{**

**for** **(**int i **=** start**;** i **<** end**;** i**++)**

**{**

fileWrite **<<** oldData**[**i**];**

fileWrite **<<** endl**;**

**}**

**}**

/// <summary>

/// Удаляет номер зачётной книжки из реестра

/// </summary>

void deleteRecordBookNumber**(**string rbNumber**)**

**{**

int elem**;**

**for** **(**int i **=** 0**;** i **<** booksData**.**size**();** i**++)**

**{**

**if** **(**booksData**[**i**]** **==** rbNumber**)**

**{**

elem **=** i**;**

**}**

**}**

booksData**.**erase**(**booksData**.**begin**()** **+** elem**);**

**}**

/// <summary>

/// Проверяет файл на наличие уже записанных номеров зачётных книжек

/// </summary>

void checkRBNumber**(**int studentsNumber**)**

**{**

int coef **=** searchData**(**"Номер зачётной книжки - "**)** **-** 1**;**

**for** **(**int i **=** 0**;** i **<** studentsNumber**;** i**++)**

**{**

tLine **=** getInfo**(**coef **+** DECREMENT **\*** i**);**

tLine**.**erase**(**0**,** 24**);**

booksData**.**push\_back**(**tLine**);**

**}**

**}**

/// <summary>

/// Удаляет данные о студенте из файла, оставляя на их месте пробел

/// </summary>

void killStudent**(**int line**)**

**{**

int studentNumber**;**

studentNumber **=** getStudentNumber**(**line**);**

clearStudentPerfomance**(**studentNumber**);**

fileWrite**.**close**();**

createNewTemporaryFile**(**mode**);**

saveOldData**(**line**);**

**if** **(**studentNumber **==** 1**)**

**{**

**for** **(**int i **=** 0**;** i **<** DECREMENT**;** i**++)**

**{**

fileWrite **<<** "----------\n"**;**

**}**

copyFromOldData**(**0**,** oldData**.**size**());**

**}**

**else** **if** **(**studentNumber **==** studentCount**)**

**{**

copyFromOldData**(**0**,** oldData**.**size**());**

**for** **(**int i **=** 0**;** i **<** DECREMENT**;** i**++)**

**{**

fileWrite **<<** "----------\n"**;**

**}**

**}**

**else**

**{**

copyFromOldData**(**0**,** line **-** 1**);**

**for** **(**int i **=** 0**;** i **<** DECREMENT**;** i**++)**

**{**

fileWrite **<<** "----------\n"**;**

**}**

copyFromOldData**(**line **-** 1**,** oldData**.**size**());**

**}**

**if** **(**mode **==** 0**)**

**{**

mode **=** 1**;**

oldData**.**clear**();**

remove**(**"StudentsData.txt"**);**

**}**

**else**

**{**

mode **=** 0**;**

oldData**.**clear**();**

remove**(**"TemporaryStudentsData.txt"**);**

**}**

**}**

void clearStudentPerfomance**(**int StudentNumber**)**

**{**

int temp**;**

**for** **(**int i **=** 0**;** i **<** v5**.**size**();** i**++)**

**{**

temp **=** v5**[**i**]** **/** 10**;**

**if** **(**temp **==** StudentNumber**)**

**{**

v5**.**erase**(**v5**.**begin**()** **+** i**);**

**}**

**}**

**for** **(**int i **=** 0**;** i **<** v4**.**size**();** i**++)**

**{**

temp **=** v4**[**i**]** **/** 10**;**

**if** **(**temp **==** StudentNumber**)**

**{**

v4**.**erase**(**v4**.**begin**()** **+** i**);**

**}**

**}**

**for** **(**int i **=** 0**;** i **<** v3**.**size**();** i**++)**

**{**

temp **=** v3**[**i**]** **/** 10**;**

**if** **(**temp **==** StudentNumber**)**

**{**

v3**.**erase**(**v3**.**begin**()** **+** i**);**

**}**

**}**

**}**

FileManager**()**

**{**

fileRead**.**open**(**"StudentsData.txt"**);**

**if** **(**fileRead**.**is\_open**())**

**{**

fileRead**.**close**();**

studentCount **=** checkForStudentCount**();**

**if** **(**studentCount **!=** 0**)**

**{**

checkRBNumber**(**studentCount**);**

**}**

**}**

**}**

**~**FileManager**()**

**{**

fileRead**.**close**();**

fileWrite**.**close**();**

rename**(**"TemporaryStudentsData.txt"**,** "StudentsData.txt"**);**

**}**

**};**

/// <summary>

/// Отвечает за управление БД

/// </summary>

class MenuPattern **:** public FileManager

**{**

string tRecordBookNumber**;**

FullNameTemplate tFullName**;**

int menu**;**

int flag**;**

public**:**

/// <summary>

/// Создаёт консольный интерфейс меню

/// </summary>

/// <returns>Возвращает 1 до тех пор пока пользователь не решит выйти</returns>

int startMenu**()**

**{**

menu **=** 0**;**

cout **<<** "\nВыберите действие:\n"**;**

cout **<<** "\n1. Создать базу данных студентов\n2. Записать данные студента в базу данных\n3. Узнать данные студента\n4. Изменить данные студента\n5. Удалить данные студента\n6. Посмотреть успеваемость студентов\n7. Выйти из программы\n"**;**

cout **<<** "Ваш выбор -> "**;**

cin **>>** menu**;**

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

**if** **((**menu **>** 1**)** **&&** **(**menu **<** 7**))**

**{**

openToRead**(**mode**);**

**if** **(!(**fileRead**.**is\_open**()))**

**{**

system**(**"cls"**);**

**return** 1**;**

**}**

fileRead**.**close**();**

**}**

**switch** **(**menu**)**

**{**

**case** 1**:**

**{**

flag **=** 1**;**

openToWrite**();**

system**(**"pause"**);**

system**(**"cls"**);**

**return** 1**;**

**}**

**case** 2**:**

**{**

**if** **(**flag **==** 0**)**

**{**

createNewTemporaryFile**(**mode**);**

saveOldData**(-**1**);**

copyFromOldData**(**0**,** oldData**.**size**());**

writeInFile**();**

**if** **(**mode **==** 0**)**

**{**

mode **=** 1**;**

remove**(**"StudentsData.txt"**);**

**}**

**else**

**{**

mode **=** 0**;**

remove**(**"TemporaryStudentsData.txt"**);**

**}**

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

system**(**"pause"**);**

**}**

**else**

**{**

writeInFile**();**

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

system**(**"pause"**);**

**}**

system**(**"cls"**);**

**return** 1**;**

**}**

**case** 3**:**

**{**

int searchedLine**;**

int studentNumber**;**

int sessionNumber**;**

cout **<<** "\nВведите номер зачётной книжки -> "**;**

cin **>>** tRecordBookNumber**;**

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

searchedLine **=** searchData**(**tRecordBookNumber**);**

**if** **(**searchedLine **==** **-**1**)**

**{**

cout **<<** "\nТакой номер не был найден. Проверьте правильность ввода и повторите попытку\n"**;**

system**(**"pause"**);**

system**(**"cls"**);**

**return** 1**;**

**}**

searchedLine **-=** 6**;**

cout **<<** "\nДанные о студенте:\n"**;**

printInfo**(**searchedLine**,** searchedLine **+** 7**);**

B**:**

cout **<<** "\nВыберите действие:\n"**;**

cout **<<** "1. Посмотреть результаты сессий\n2. Вернуться в главное меню\n"**;**

cout **<<** "Ваш выбор -> "**;**

cin **>>** menu**;**

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

**if** **(**menu **==** 1**)**

**{**

studentNumber **=** getStudentNumber**(**searchedLine**);**

**if** **(**studentNumber **==** **-**1**)**

**{**

system**(**"pause"**);**

**return** 1**;**

**}**

C**:**

cout **<<** "\nВыберите номер сессии -> "**;**

cin **>>** sessionNumber**;**

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

**if** **((**sessionNumber **<** 1**)** **||** **(**sessionNumber **>** SESSIONNUMBER**))**

**{**

cout **<<** "\nОшибка ввода, повторите попытку\n"**;**

system**(**"pause"**);**

system**(**"cls"**);**

**goto** C**;**

**}**

printSessionInfo**(**sessionNumber**,** studentNumber**);**

E**:**

cout **<<** "\n\nВыберите действие:\n"**;**

cout **<<** "\n1. Выбрать другую сессию\n2. Вернуться на главный экран\n"**;**

cout **<<** "Ваш выбор -> "**;**

cin **>>** menu**;**

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

**if** **(**menu **==** 1**)**

**{**

**goto** C**;**

**}**

**else** **if** **(**menu **==** 2**)**

**{**

system**(**"cls"**);**

**return** 1**;**

**}**

**else**

**{**

cout **<<** "\nОшибка ввода повторите попытку!!!\n"**;**

system**(**"pause"**);**

**goto** E**;**

**}**

**}**

**else** **if** **(**menu **==** 2**)**

**{**

system**(**"cls"**);**

**return** 1**;**

**}**

**else**

**{**

cout **<<** "\nОшибка ввода повторите попытку!!!\n"**;**

system**(**"pause"**);**

**goto** B**;**

**}**

**}**

**case** 4**:**

**{**

int tLine**;**

cout **<<** "\nВведите номер зачётной книжки студента, данные которого нужно изменить -> "**;**

cin **>>** tRecordBookNumber**;**

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

tLine **=** searchData**(**tRecordBookNumber**);**

**if** **(**tLine **==** **-**1**)**

**{**

cout **<<** "\nТакой номер не был найден. Проверьте правильность ввода и повторите попытку\n"**;**

system**(**"pause"**);**

system**(**"cls"**);**

**return** 1**;**

**}**

tLine **-=** 6**;**

cout **<<** "\nВведите новые данные студента:\n"**;**

deleteRecordBookNumber**(**tRecordBookNumber**);**

changeStudentData**(**tLine**);**

cout **<<** "\nДанные успешно изменены!\n"**;**

system**(**"pause"**);**

system**(**"cls"**);**

oldData**.**clear**();**

**return** 1**;**

**}**

**case** 5**:**

**{**

int tLine**;**

cout **<<** "\nВведите номер зачётной книжки студента, данные которого нужно удалить -> "**;**

cin **>>** tRecordBookNumber**;**

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

tLine **=** searchData**(**tRecordBookNumber**);**

**if** **(**tLine **==** **-**1**)**

**{**

cout **<<** "\nТакой номер не был найден. Проверьте правильность ввода и повторите попытку\n"**;**

system**(**"pause"**);**

system**(**"cls"**);**

**return** 1**;**

**}**

tLine **-=** 6**;**

deleteRecordBookNumber**(**tRecordBookNumber**);**

killStudent**(**tLine**);**

cout **<<** "\nДанные успешно удалены!\n"**;**

system**(**"pause"**);**

system**(**"cls"**);**

**return** 1**;**

**}**

**case** 6**:**

**{**

**if** **(**studentCount **<** 1**)**

**{**

studentCount **=** checkForStudentCount**();**

**}**

countStudentsPerfomance**();**

A1**:**

cout **<<** "\nВведите номер сессии -> "**;**

cin **>>** menu**;**

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

**if** **((**menu **<** 1**)** **||** **(**menu **>** SESSIONNUMBER**))**

**{**

cout **<<** "\nОшибка ввода, повторите попытку\n"**;**

system**(**"pause"**);**

**goto** A1**;**

**}**

showStudentPerfomance**(**menu**);**

A2**:**

cout **<<** "\nВыберите действие:\n"**;**

cout **<<** "\n1. Выбрать другую сессию\n2. Вернуться в главное меню\n"**;**

cout **<<** "\nВаш выбор -> "**;**

cin **>>** menu**;**

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

**if** **(**menu **==** 1**)**

**{**

**goto** A1**;**

**}**

**else** **if** **(**menu **==** 2**)**

**{**

system**(**"cls"**);**

**return** 1**;**

**}**

**else**

**{**

cout **<<** "\nОшибка ввода повторите попытку!!!\n"**;**

system**(**"pause"**);**

**goto** A2**;**

**}**

**}**

**case** 7**:**

**{**

system**(**"cls"**);**

**return** 0**;**

**}**

**default:**

**{**

cout **<<** "\nОшибка ввода!!!!\nПовторите попытку\n"**;**

system**(**"pause"**);**

system**(**"cls"**);**

**return** 1**;**

**}**

**}**

**}**

MenuPattern**()**

**{**

flag **=** 0**;**

**}**

MenuPattern**(**int flag**)**

**{**

**while** **(**flag**)**

**{**

flag **=** startMenu**();**

**}**

**}**

**};**

int Student**::**studentCount **=** 0**;**

/// <summary>

/// Тело программы. Зацикливает выбор опций в объекте меню

/// </summary>

/// <returns>Возвращает 0 при успешном завершении</returns>

int main**()**

**{**

int indicator **=** 1**;**

SetConsoleCP**(**1251**);**

SetConsoleOutputCP**(**1251**);**

MenuPattern**\*** Menu **=** **new** MenuPattern**;**

**while** **(**indicator **!=** 0**)**

**{**

indicator **=** Menu**->**startMenu**();**

**}**

**delete** Menu**;**

**return** 0**;**

**}**