课 程 设 计 报 告

课程名称 计算机程序设计基础 2

班　　级 无 24

学　　号 2022010597

姓　　名 陈彦旭

2023 年 5 月 16 日

一、设计内容与设计要求

1.课程设计目的

面向对象程序设计课程设计是集中实践性环节之一，是学习完《计算机程序设计基础 2》C++面向对象程序设计课程后进行的一次全面的综合练习。要求学生达到熟练掌握 C++语言的基本知识和技能；基本掌握面向对象程序设计的思想和方法；能够利用所学的基本知识和技能，解决简单的面向对象程序设计问题，从而提高动手编程解决实际问题的能力。**尤其重视创新思维培养。**

2.课题题目

1）学生成绩管理系统（或公司人事管理系统）

3. 文档设计要求

3.1 设计课题题目：每个同学都单独完成 1 道课题。后面有范题，仅供同学们参考，不列入本次课程设计的课题。

3.2 对于程设题目，按照范题的格式。自行虚构软件需求。并按照第 4 点要求，编写设计文档。基本要求系统中设计的类的数目不少于 4 个，每个类中要有各自的属性（多于 3 个）和方法（多于 3 个）；需要定义一个抽象类，采用继承方式派生这些类。并设计一个多重继承的派生类。在程序设计中，引入虚函数的多态性、运算符重载等机制。

4. 程序设计的基本要求：

（1）要求利用面向对象的方法以及 C++的编程思想来完成系统的设计；

（2）要求在设计的过程中，建立清晰的类层次；

（3）根据课题完成以下主要工作：

①完成系统需求分析：包括系统设计目的与意义；系统功能需求（系统流程图）；输入输出的要求。

②完成系统总体设计：包括系统功能分析；系统功能模块划分与设计（系统功能模块图）。

③完成系统详细设计：数据文件；类层次图；界面设计与各功能模块实现。

④系统调试：调试出现的主要问题，编译语法错误及修改，重点是运行逻辑问题修改和调整。

⑤使用说明书及编程体会：说明如何使用你编写的程序，详细列出每一步的操

作步骤。

⑥关键源程序（带注释）。

（4）自己设计测试数据，将测试数据存在文件中，通过文件进行数据读写来获得测试结果。

（5）按规定格式完成课程设计报告，并在网络学堂上按时提交。

（6）不得抄袭他人程序、课程设计报告，每个人应独立完成，在程序和设计报告中体现自己的个性设计。

5、进度安排

小学期 第 **2** 周			

注： 1、一定要保留自己那个课题的完整任务书在课程设计报告里面。

2、“评分表”放在“附录：源程序清单”的后面。

附录 1：评分表

第 1 题评分标准

项　　目	评　　价	
设计方案的合理性与创新性	**6**	
设计与调试结果	**8**	
设计说明书的质量	**2**	
程序基本要求涵盖情况	**8**	
程序代码编写素养情况	**4**	

课程设计周表现情况	**2**	
综合成绩	**30**	

要求和格式示范：学生考勤管理系统

说明：范例中红色字体的部分都是必须

目　录

1. 系统需求分析

成绩管理系统记录了学生的选课情况和各门课程的考试成绩、最终评级，包括该门课程的名称、授课教师、教师院系、学分、是否记 P/F、百分制成绩、绩点、等级（A+，A 等）。试学生成绩管理系统，使之能提供以下功能：

1、以教师为主体：

教师具有录入、修改自己教授课程的成绩，和查看学生成绩的权限：

（1）对于一门未记录成绩的课程，教师可根据选课名单逐一录入百分制成绩；

（2）对于一门已有成绩记录的课程，教师可以查看选课名单并选择某学生的成绩进行修改，修改成功后提示原成绩和现成绩。

（3）教师可以查看自己教授课程的学生成绩名单，按照百分制成绩由高到低排序；

（4）教师可以根据学生姓名或学号查看某位学生的所有已选课程成绩，不仅限于自己教授的课程；

（5）教师可以查看某班级的所有学生成绩情况和排名，按照由高到低排序，以及该班级的平均绩点；

（6）教师可以查看自己的个人信息，包括姓名、工号、所在院系、授课名称、课程学分、课程是否记 P/F；

2、以学生为主体：

（1）查看自己所选的某一门课程的成绩，包括百分制成绩、绩点、等级和课程信息。

（2）查看自己所选全部课程成绩，显示每门课程百分制成绩、绩点、等级、学分，以及自己修得总学分数、计入 GPA 总学分数、平均 GPA、总学分绩。

（3）选择一门课程记为 P/F（每位学生仅可至多选一门课程记为 P/F，不记自带 P/F 的课程），记录成功后该课程将不计入 GPA 计算，GPA、班级排名等其他数据将会随之更新。

（4）查看自己的平均绩点和班级排名，显示班级人数、最高均绩和最低均绩但不显示其他所有人的均绩。

（5）查看自己的个人信息，包括姓名、学号、班级。

3、系统以菜单方式工作：

系统启动后，用户进入主界面，需选择自己的身份并进行登录，登录入自己专属的操作界面，然后可以自由选择所要执行的功能，只需在键盘上输入功能前面对应的数字即可。

4、信息存档：

程序启动时对信息文件进行读取，用户在使用系统时的所有操作以及数据更改都会被记录，关闭时系统自动将数据重新导入文件实现数据的更新。

2. 总体设计

学生成绩管理系统（教师操作）包括六大功能：录入学生课程成绩、修改学生课程成绩、查询选修所授课程的所有学生成绩、查询某学生的所有课程成绩、查询某班级的所有学生成绩和排名、查看自己的个人信息。教师在主界面选择 1—教师身份后进行登录操作（教师姓名和工号均正确且匹配才能成功登录，输入不正确需要重新输入，直至登陆成功为止），登录成功后会进入教师主界面，并按照提示选择对应功能前面的序号，在键盘上直接输入数字即可进入对应的功能界面。1—6 是教师需要执行的具体功能，0 代表教师选择退出教师主界面。

1、在录入学生成绩时，教师可选择两种方式：

（1）根据系统搜索给出选课学生名单，然后教师逐一输入学生的百分制成绩，系统会自动判断成绩是否在 0—100 内，否则将提示教师重新输入，输入成功后系统会提示已成功录入成绩，并自动计算对应的绩点、等级。之后系统提示继续输入，直到录入完成所有选课学生的成绩为止。

（2）在该课程中加入一位新选课的学生，教师需要创建该学生的姓名、学号、班级（若该学号已存在，则会提示教师重新输入），并自动将其加入所在班级的学生名单中（改班级不存在时将会创建一个新的班级），然后按照（1）中的方式输入该学生的成绩。

2、在修改学生成绩时（未录入的学生成绩默认为 0），根据系统搜索给出学生和成绩名单，教师选择并输入学生姓名前的序号，对其成绩进行修改（若该序号不在列表范围内，系统将提示教师重新输入），输入百分制成绩后，系统会自动判断成绩是否在 0

—100 内，否则将提示教师重新输入，修改成功后系统会显示原成绩和现成绩，并提示是否继续修改，教师根据提示输入 1 可以继续选择下一位学生进行成绩修改，输入其它数字将会退出修改，返回教师主界面。

3、在查询课程成绩时，系统会自动判断教师所授课程以及选课学生名单，按照百分制成绩从高到低进行排名并依次输出，每行一个（同分学生排名相同），信息包括学生姓名、学号、百分制成绩、绩点、等级、选课内排名。此功能无需教师额外操作。

4、在查询学生课程成绩时，教师有两种方式查询某位学生成绩：按姓名或按学号。若输入 1 按姓名查找，输入 2 按学号查找，由于学生可能存在同名情况而学号是学生的唯一凭证，因此姓名查询的结果可能不唯一，系统将会显示所有该姓名的学生成绩，学号查询的唯一。教师输入相应数字选择查询方式后，系统继续提示输入要查询的学生姓名/学号，搜索并输出符合条件的学生（该姓名/学号不存在时将提示教师重新输入），包括个人信息（姓名、学号、班级）、所有已选课程成绩以及平均绩点、班级排名。系统输出完成后，教师可根据提示输入 1 继续查询其他学生的成绩（需要重新选择查询方式：姓名/学号），若输入其他数字将会退出查询，返回教师主界面。

5、在查询班级学生成绩时，教师可直接输入班级号码，系统将自动搜索该班级学生（若该班级不存在将会提示教师重新输入），并按照绩点由高到低进行排名并输出，每行一个（绩点相同的学生排名相同），并在末尾计算该班级整体的人数、平均绩点。系统输出完成后，教师可根据提示输入 1 继续查询其他班级的学生成绩，若输入其他数字将会退出查询，返回教师主界面。

6、查看自己的个人信息时，系统将自动输出教师姓名、工号、所在院系、所授课程名称、课程学分、课程是否自带记 P/F。此功能无需教师额外操作。

7、退出教师主界面，退出登录，返回系统主界面，再次使用时需要选择身份并进行登录。系统将会提示教师是否确定退出，教师若输入 1 则会退出登录，输入其他数字则会取消退出，继续留在当前的教师主界面。

学生成绩管理系统（学生操作）包括五大功能：查询自己所选某门课程成绩、查看所选所有课程成绩、选择一门课程考核方式记为 P/F（不计入总 GPA）、查看自己的均绩和班级排名、查看自己的个人信息。学生在主界面选择 2—学生身份后进行登录操作（学生姓名和学号均正确且匹配才能成功登录，输入不正确需要重新输入，直至登陆成功为止），登录成功后会进入学生主界面，并按照提示选择对应功能前面的序号，在键盘上直接输入数字即可进入对应的功能界面。1—5 是学生需要执行的具体功能，0 代

表学生选择退出学生主界面。

1、在查询自己所选某门课程的成绩时，系统会自动搜索该学生所选课程并每行逐个输出，学生可输入课程前方的序号对相应课程的成绩进行查询（若序号不在列表范围内系统将提示学生重新输入），查询结果会显示学生在该门课程取得的百分制成绩、绩点、等级，并附加课程信息（授课教师姓名、院系、学分、是否自带记 P/F）。输出完成后，学生可根据系统提示，输入 1 继续查询其它课程成绩，否则将会退出查询返回学生主界面。

2、在查看自己所选所有课程成绩时，系统将会自动搜索并输出已选课程的百分制成绩、绩点、等级，在末尾计算并输出该学生的 GPA、学分数、计入 GPA 学分数和总学分绩。此功能无需学生额外操作。

3、在选择一门课程记为 P/F，系统会先判断该学生之前是否已经将一门课程记为 P/F，如果有，系统提示学生选择 P/F 另一门课程或维持原 P/F 选择不变：

（1）若输入 0 则会选择一门课程考核方式记为 P/F 并推出该功能界面。

（2）若输入 1 则会取消原先 P/F 的课程，然后列出学生已选课程，学生选择可课程名前的序号对相应课程进行 P/F（自带 P/F 的课程也在选择范围之内，但若学生选择将其记为 P/F 将不产生任何影响，只算作使用过了 P/F 的机会），输入相应数字后系统会再次提示学生是否确定将其记为 P/F。记录成功后学生将退出并返回学生主界面。若学生在课程序号输入 0，则会视为放弃本次 P/F 选择。

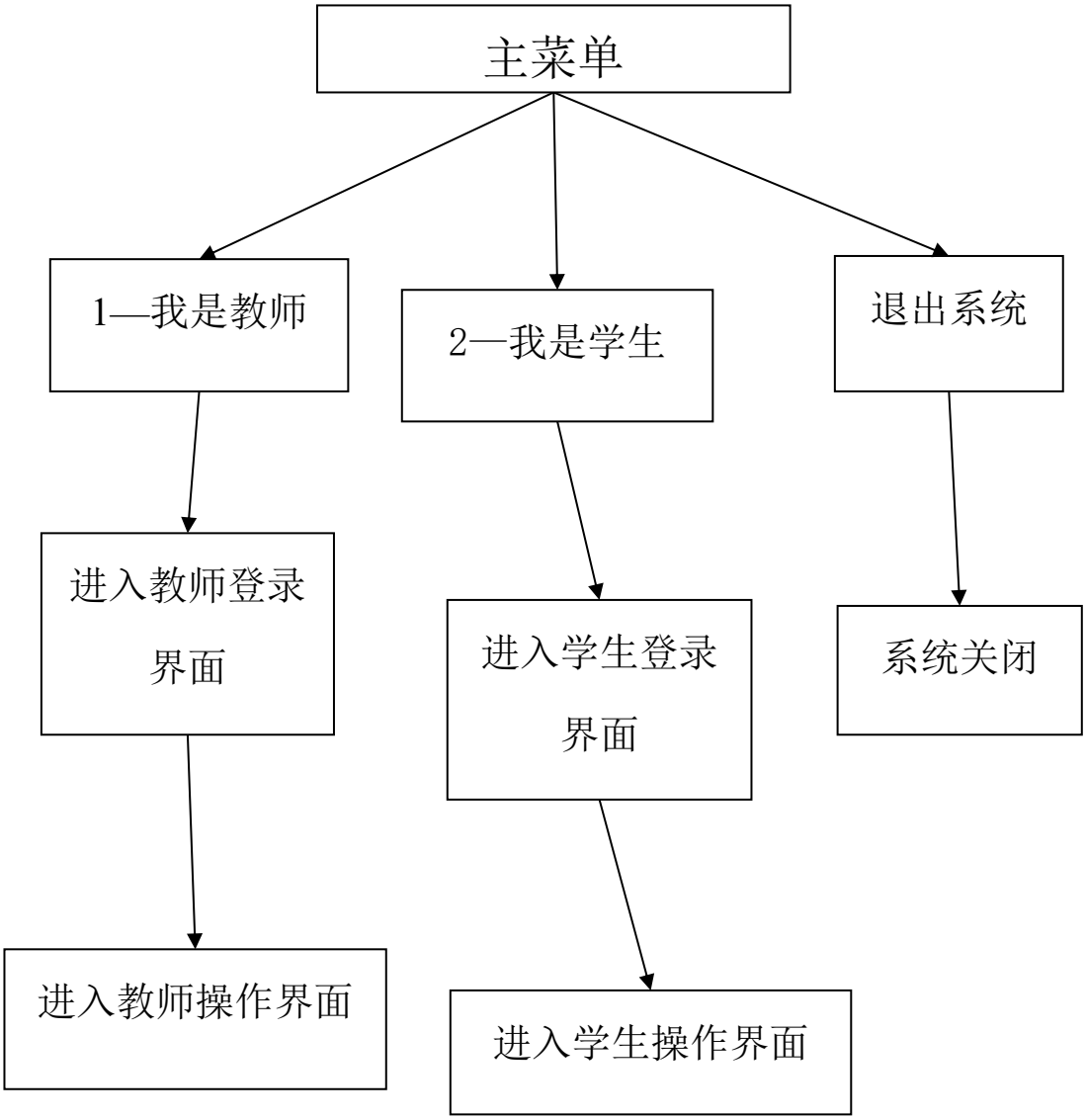
（3）若输入 1、0 之外的不合法数字，系统将会提示学生重新输入。

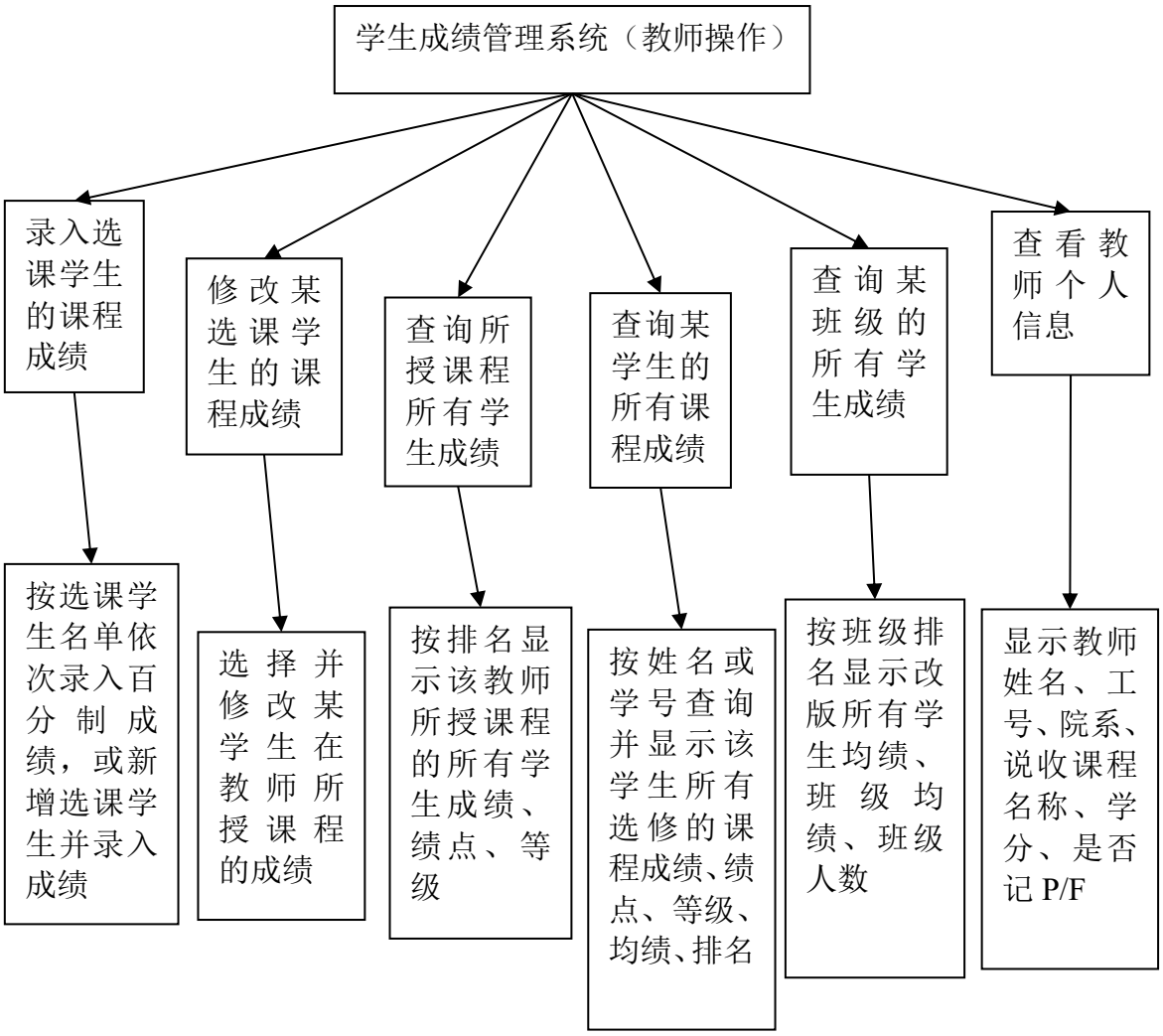
无论学生进行何种操作，退出该功能时，该学生的成绩、排名等信息将会自动更新。

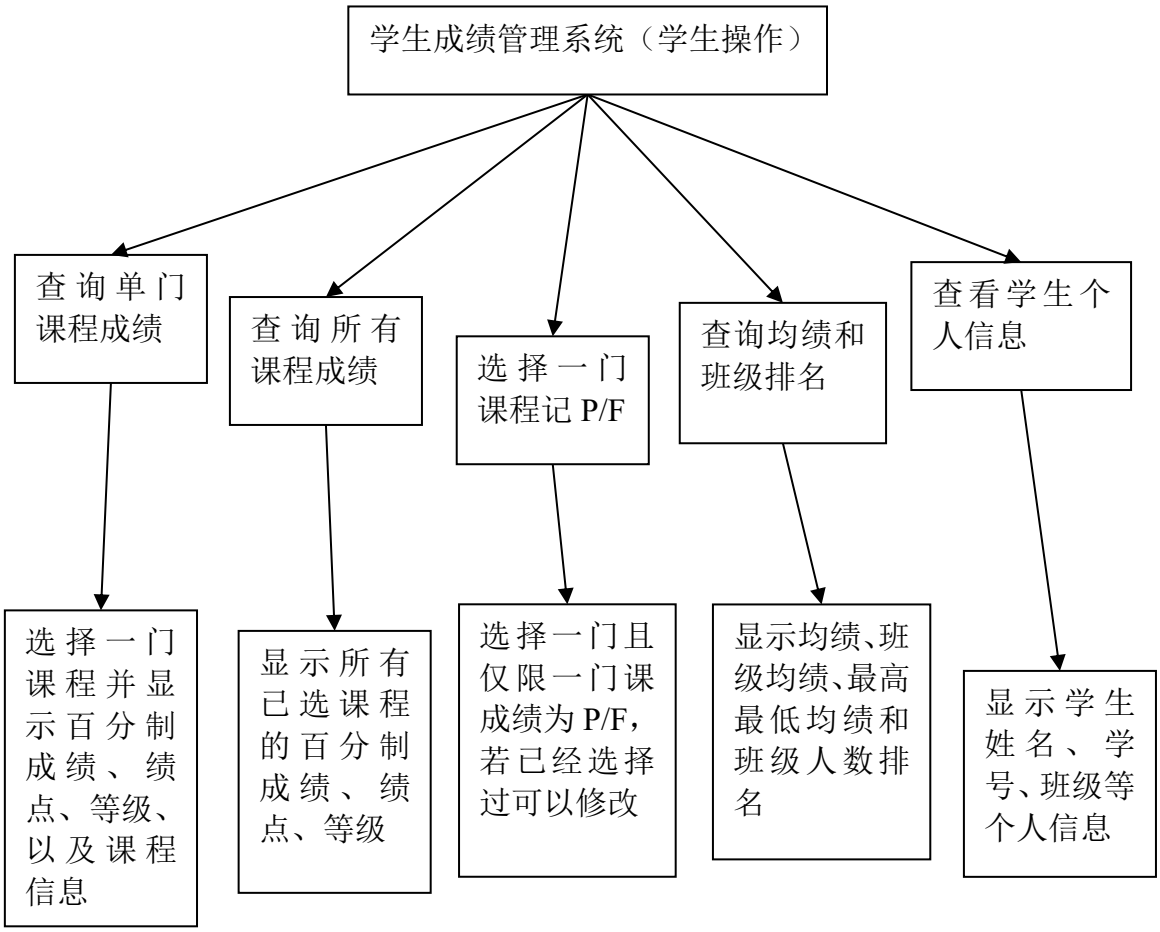
4、在查询均绩和班级排名时，系统将自动输出学生班级号码、该学生的平均绩点、班级排名、班级人数、班级均绩、最高均绩和最低均绩，但为保护同学之间隐私，不会显示最高、最低均绩对应的学生信息和其他所有学生信息。此功能无需学生额外操作。

5、退出学生主界面，退出登录，返回系统主界面，再次使用时需要选择身份并进行登录。系统将会提示学生是否确定退出，学生若输入 1 则会退出登录，输入其他数字则会取消退出，继续留在当前的学生主界面。

学生成绩管理系统中功能模块图：

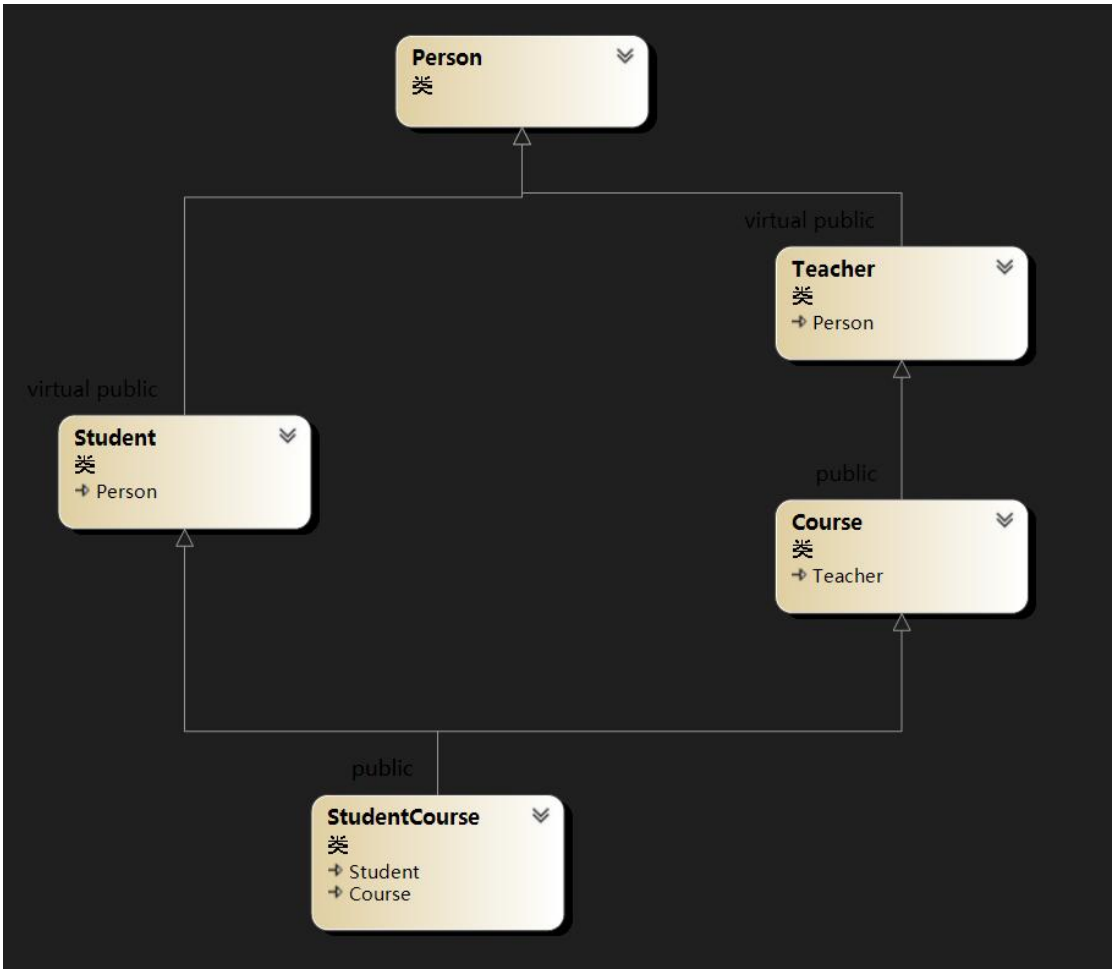

主菜单
1—我是教师
2—我是学生
退出系统
进入教师登录界面
进入学生登录界面
系统关闭
进入教师操作界面
进入学生操作界面


学生成绩管理系统（教师操作）
录入选课学生的课程成绩
修改某选课学生的课程成绩
查询所授课程所有学生成绩
查询某学生的所有课程成绩
查询某班级的所有学生成绩
查看教师个人信息
按选课学生名单依次录入百分制成绩，或新增选课学生并录入成绩
选择并修改某学生在教师所授课程的成绩
按排名显示该教师所授课程所有学生成绩、绩点、等级
按姓名或学号查询该学生并显示该有选修的课程成绩、绩点、等级、均绩、排名
按班级排名显示改版所有学生均绩、班级均绩、班级人数
显示教师姓名、工号、院系、说收课程名称、学分、是否记 P/F


学生成绩管理系统（学生操作）
查询单门课程成绩
查询所有课程成绩
选择一门课程记 P/F
查询均绩和班级排名
查看学生个人信息
选择一门课程并显示百分制成绩、绩点、等级、以及课程信息
显示所有已选课程的百分制成绩、绩点、等级
选择一门且仅限一门课成绩为 P/F，若已经选择过可以修改
显示均绩、班级均绩、最高最低均绩和班级人数排名
显示学生姓名、学号、班级等个人信息

3. 详细设计

学生成绩管理系统中五个类的类层次图为：



Person 类是一个抽象类，只有两个成员变量：name 和 id，成员函数只有两个：一个是纯虚函数 displayInfo() 用于输出该对象的信息，一个是虚析构函数~Person()。

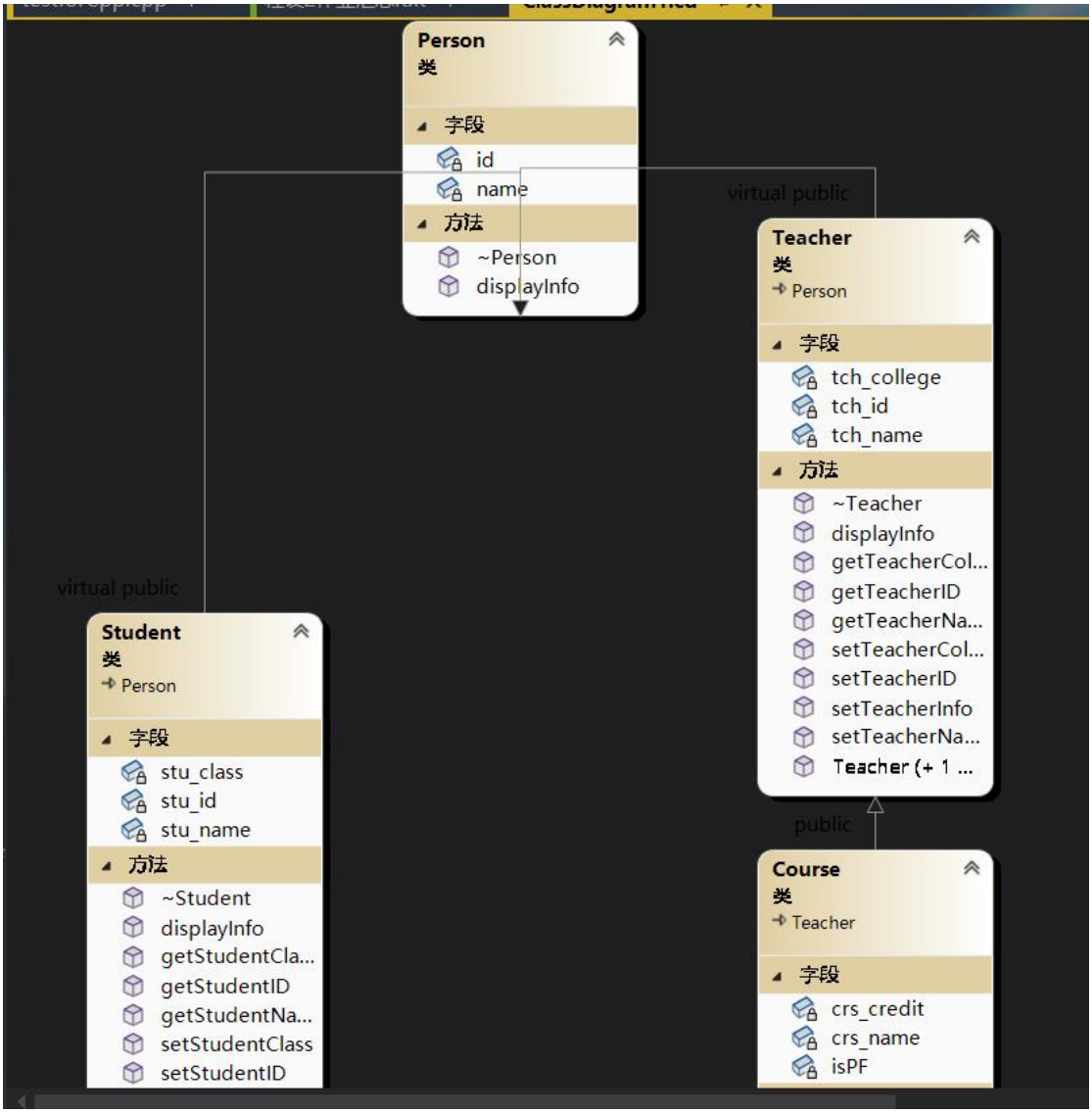
虚基类 Person 下公有派生了两个类：Student 和 Teacher 类，二者均是虚继承 Person 类。

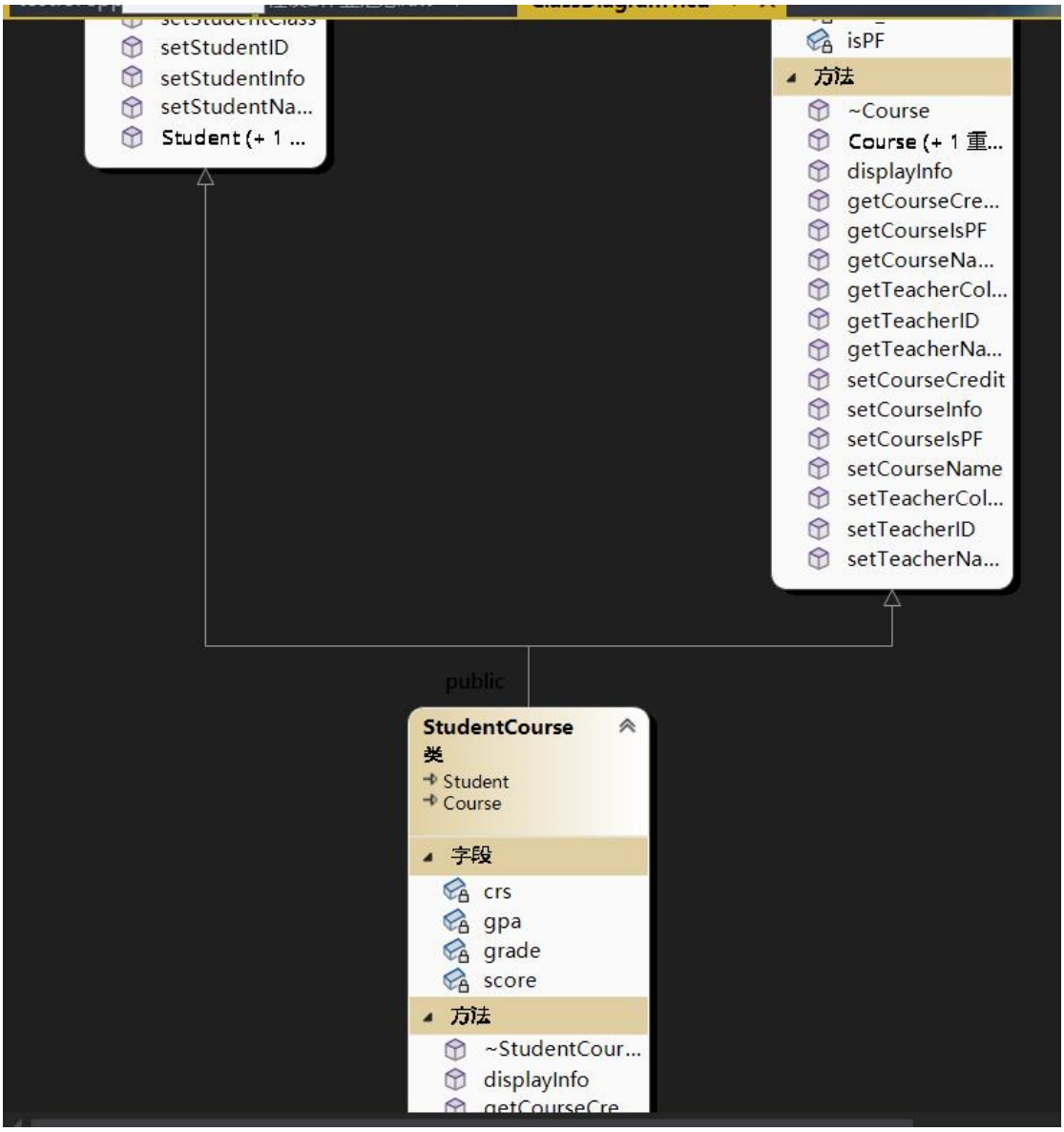
Student 类包含成员变量：学生姓名 stu_name、学号 stu_id、班级 stu_class，成员函数除了构造函数、复制构造函数、析构函数之外，还包括设置新值函数、取值函数以及重载虚函数 displayInfo()，共计 11 个。

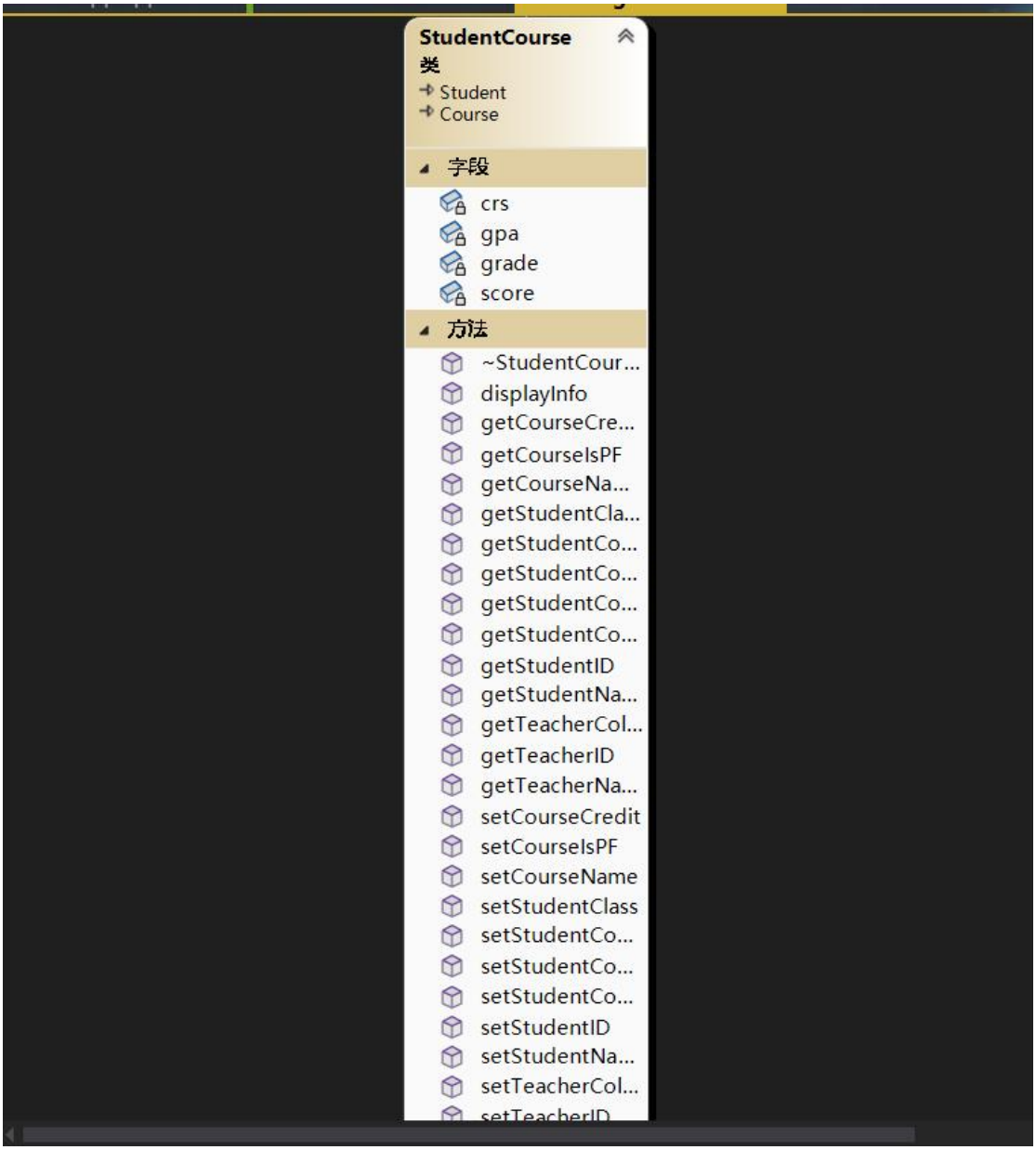
Teacher 类包含成员变量：教师姓名 tch_name、学号 tch_id、班级 tch_college，成员函数除了构造函数、复制构造函数、析构函数之外，还包括设置新值函数、取值函数以及重载虚函数 displayInfo()，共计 11 个。

Course 公有继承且虚继承自 Teacher，成员变量包括课程名称 crs_name、课程学分 crs_credit、是否记为 P/F 的 isPF，成员函数除了构造函数、复制构造函数、析构函数之外，还包括设置新值函数、取值函数、重载虚函数 displayInfo()、获取基类成员值的函数、设置基类成员值的函数，共计 17 个。

StudentCourse 多继承自 Student 类和 Course 类，成员变量包括：课程名称 crs、百分制成绩 score、绩点 gpa、等级 grade，成员函数除了构造函数、复制构造函数、析构函数之外，还包括设置新值函数、取值函数、重载虚函数 displayInfo()、获取基类成员值的函数、设置基类成员值的函数，共计 29 个。

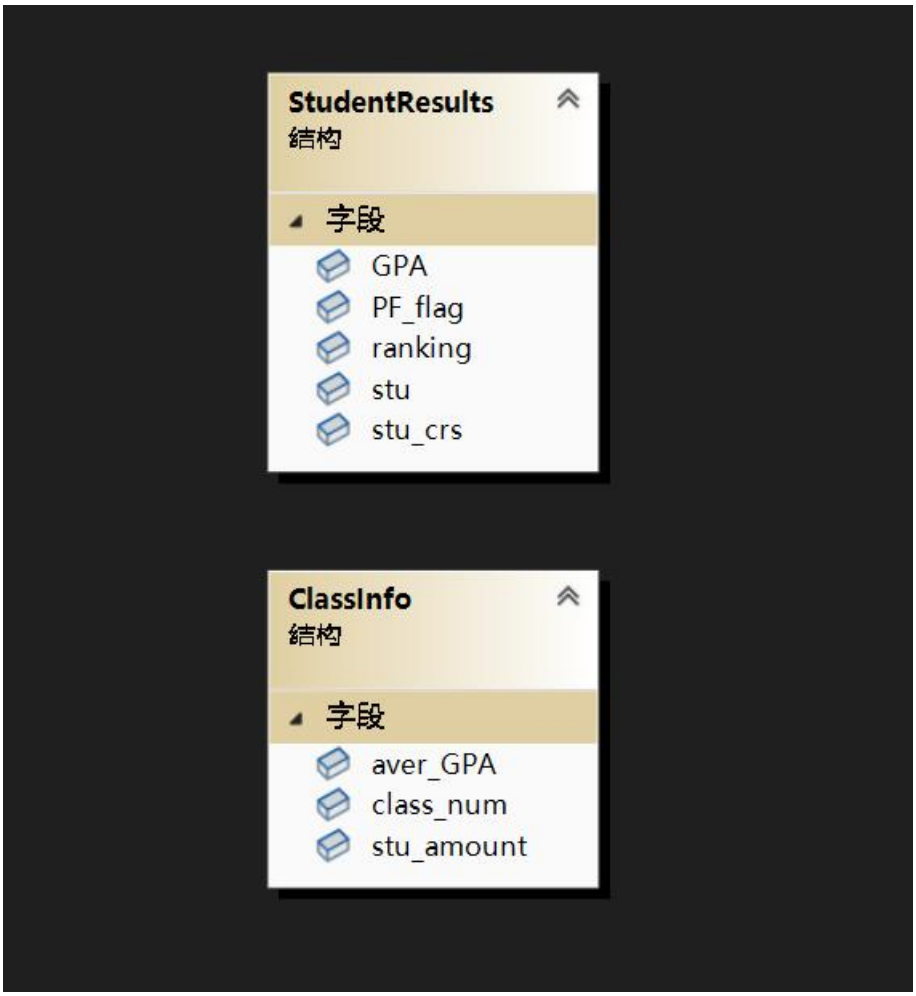

Person
类
字段
id
name
方法
~Person
displayInfo
virtual public
virtual public
Teacher
类
Person
字段
tch_college
tch_id
tch_name
方法
~Teacher
displayInfo
getTeacherCol...
getTeacherID
getTeacherNa...
setTeacherCol...
setTeacherID
setTeacherInfo
setTeacherNa...
Teacher (+ 1 ...
Student
类
Person
字段
stu_class
stu_id
stu_name
方法
~Student
displayInfo
getStudentCla...
getStudentID
getStudentNa...
setStudentClass
setStudentID
public
Course
类
Teacher
字段
crs_credit
crs_name
isPF


setStudentID
setStudentInfo
setStudentNa...
Student (+ 1 ...
isPF
方法
~Course
Course (+ 1 重...
displayInfo
getCourseCre...
getCourseIsPF
getCourseNa...
getTeacherCol...
getTeacherID
getTeacherNa...
setCourseCredit
setCourseInfo
setCourseIsPF
setCourseName
setTeacherCol...
setTeacherID
setTeacherNa...
public
StudentCourse
类
Student
Course
字段
crs
gpa
grade
score
方法
~StudentCour...
displayInfo



整体上，类的设计采用了“菱形继承”的多继承方式，适用抽象和类纯虚函数的使用增加了多态性，是的 displayInfo()函数在不同对象中执行不同特定的功能。虚基类和虚继承的使用避免了歧义。除了 Person 类，每个类都包含至少三个成员变量和三个成员函数。

StudentCourse 类可以记录一个学生选修的一门课程的所有信息（学生姓名、学号、班级、授课教师、工号、教师所在院系、课程名称、学分、是否记 P/F、百分制成绩、等级、绩点）。在此基础上，又设计了两个结构体。



一个是 StudentResults 结构体，能够记录一个学生的所有课程成绩和其他信息，它包括记录该学生个人基本信息的 Student 类的对象 stu、一个 StudentCourse 类的 vector 数组 stu_crs、该学生的均绩 GPA（double 型）、该学生的班级排名 ranking（int 型）、标记该学生是否已经将一门课程记为 P/F 的 PF_flag（bool 型）。

另一个结构体是 ClassInfo，它包含班号 class_num、班级学生人数 stu_amount、班级均绩 aver_gpa。

在该程序中，使用最多的就是 StudentResults 结构体，利用 StudentResults 的 vector 数组可以记录多名学生的全部成绩，再利用 Teacher 和 Course 的 vector 数组可以记录教师和课程的所有信息（其实一个 Course 的 vector 数组就够了，因为 Course 包含基类 Teacher 的所有成员信息）。

程序的执行思想是定义一个 StudentResults 类的全局变量 vector 数组、一个 Teacher 类的全局变量 vector 数组、一个 Course 类的全局变量 vector 数组，然后用四个整型全局变量 stu_obj、tch_obj、crs_obj、class_obj，存入用户在操作时需要使用的对应数组下

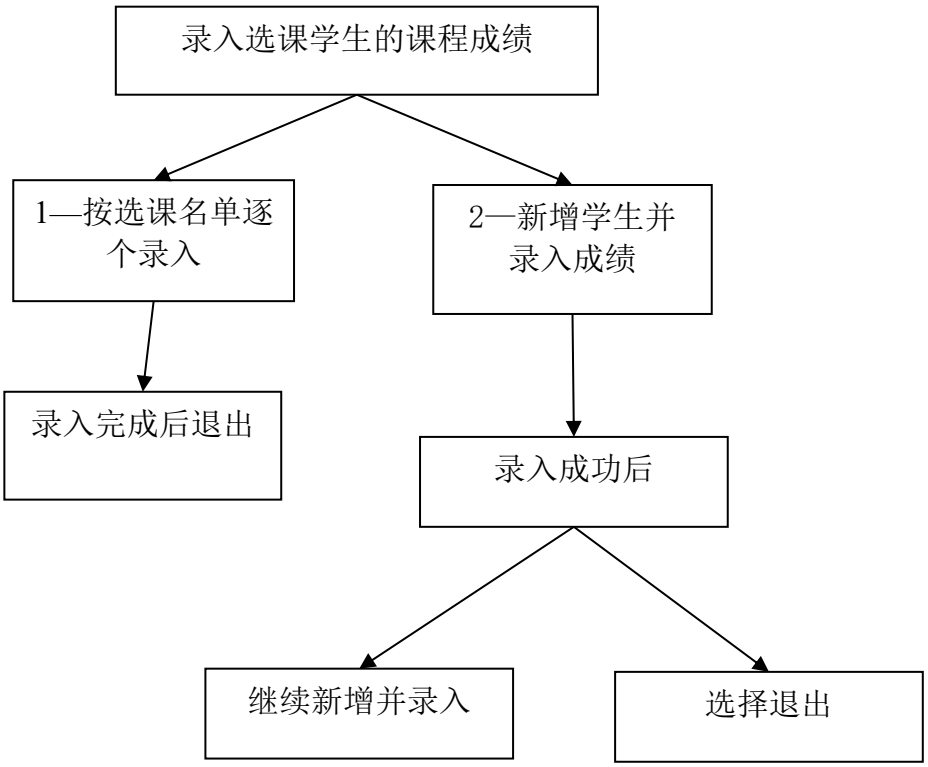
标，这样通过程序中改变这些变量的值可以使用户访问自己的信息并对自己所属的信息进行专属操作。

该程序的文件读写使用 CSV 文件，它是一种以逗号分隔的文本文件，读取和存写便于操作。程序启动时通过访问两个文件：一个是存储学生信息的文件，一个是存储教师信息的文件，将读取内容分别存入 vector<StudentResults>数组和 vector<Teacher>和 vector<Course>数组中，并开始用户操作，在程序结束时将数组中数据按格式重新存放入数组中。

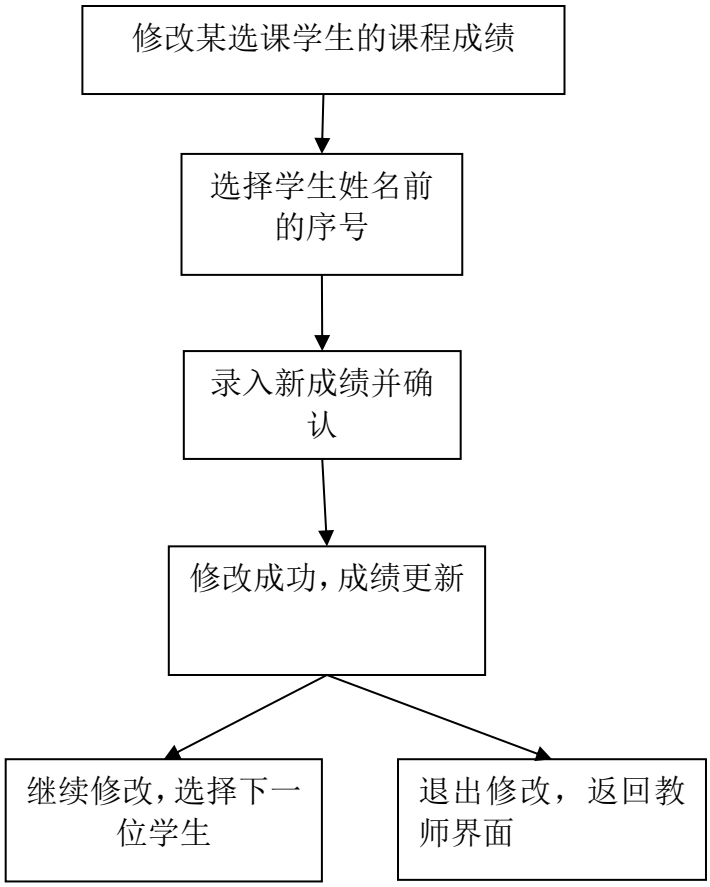
具体功能：

教师操作界面：

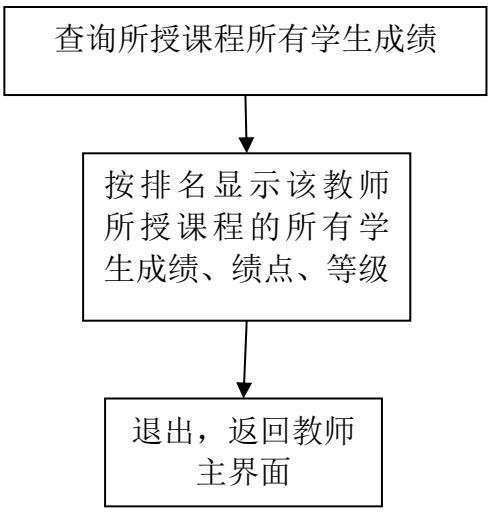
1、录入选课学生的课程成绩



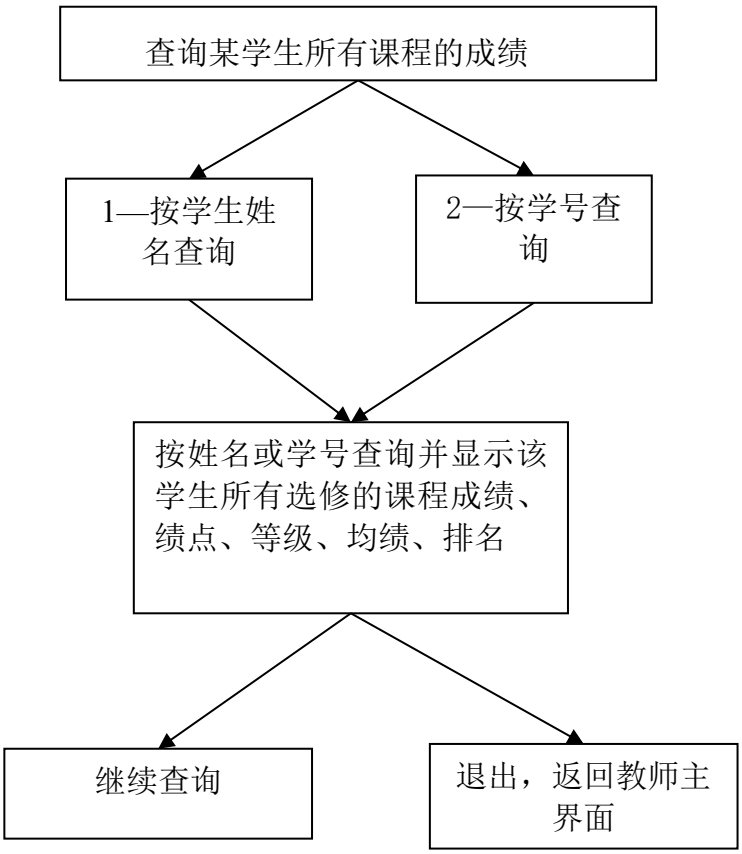
2、修改某学生的课程成绩



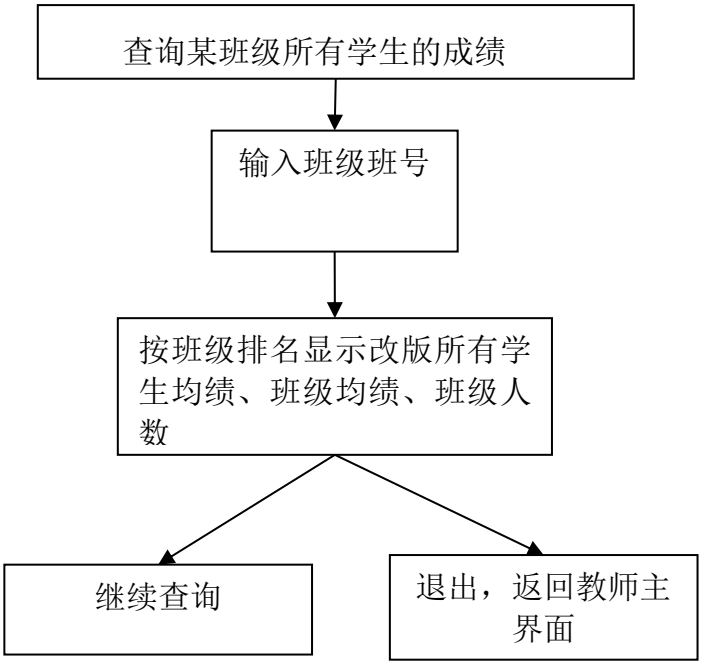
3、查询所授课程所有学生成绩



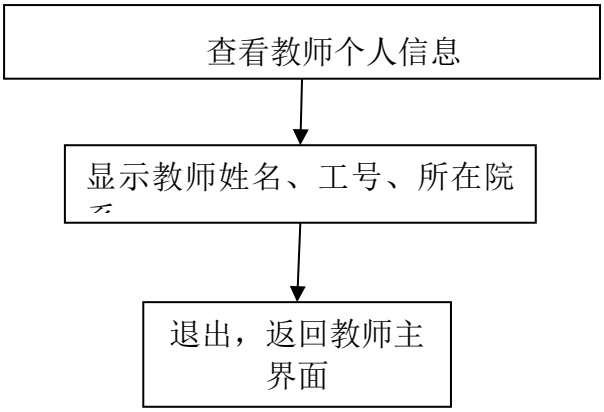
4、查询某学生所有课程的成绩



5、查询某班级的所有学生成绩：

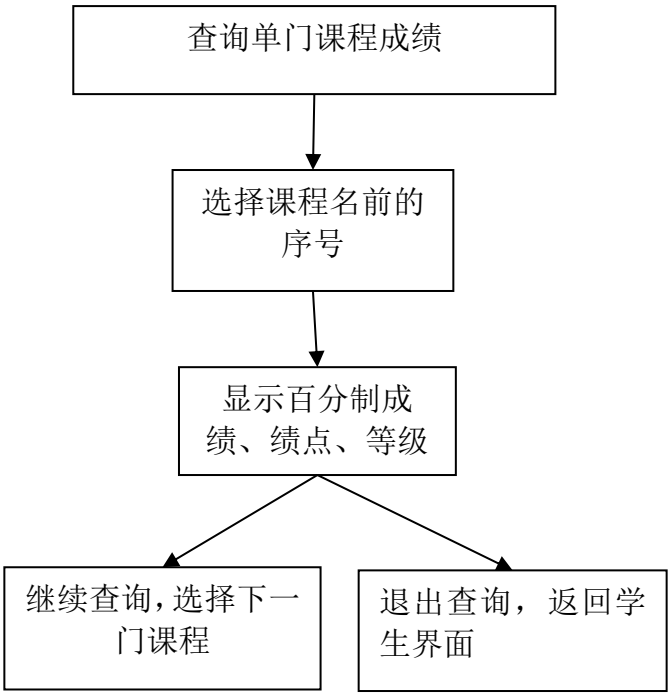


6、查询教师个人信息：

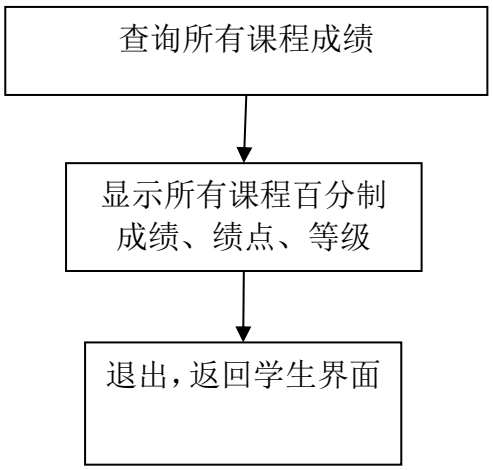


学生操作界面：

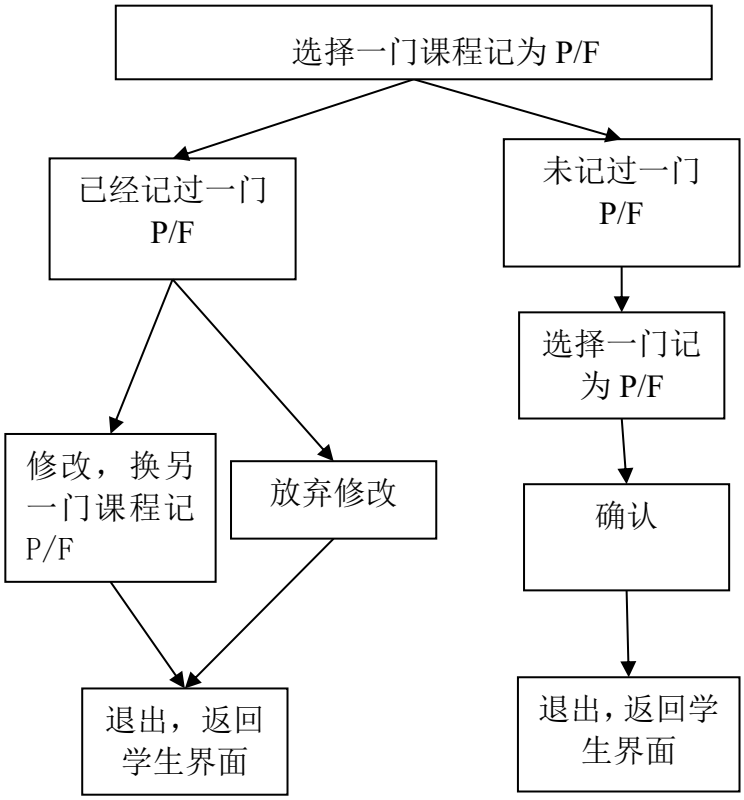
1、查询单门课程成绩；



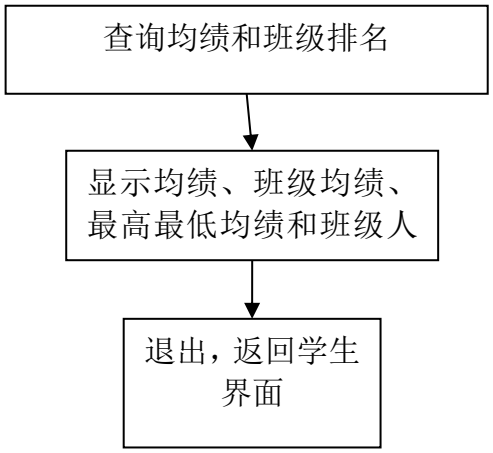
2、查询所有课程成绩：



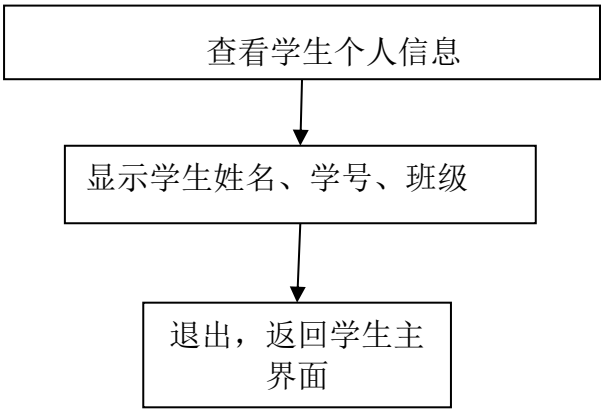
3、选择一门课程记为 P/F



4、查询均绩和班级排名：



5、查看学生个人信息



4. 系统调试

由于编写完程序代码后的调试中出现过这个几个问题：

1、文件读取错误。我在写完文件读写操作部分的代码之后，试图从测试文件中导入数据，测试文件的读写是否成功。但是出现了这样的问题：学生信息文件的读取较为完好，但是每次测试过后教师的信息文件或者变成空白，或者多列数据自动变成一列数据（因为 CSV 文件是用 Excel 打开的，表格类型），因此我尝试在主函数中直接定义几

个对象然后存入数组，并将文件读取的函数进行注释避免它的影响，然后输出这些对象的成员值，结果仍然有问题，后来我通过 StudentResource 类对象中的各种成员值读取函数，发现派生类中的成员变量成功读取而读取基类中的成员变量没有成功读取。我在网上寻找解决办法时发现由于我使用的是 vector 数组，vector 数组中的 push_back 函数会自动调用该对象的复制构造函数，而我在 Teacher 类、Course 类和 StudentCourse 类的复制构造函数中并没有将基类中的成员数据完整地复制到派生类中，导致我在使用 push_back 函数时该对象的副本中不存在基类中的成员。于是我将每一个类的复制构造函数修改完整后解决了这个问题。

2、数组越界问题。在测试教师和学生查询班级整体成绩和排名的函数功能时，程序直接发生错误，提示 vector 数组 out of range，于是我开始检查与班级相关的变量和函数。由于我设计的 ClassInfo 结构体本身包含了一个记录班号的变量 class_num，班号比它在 vector<ClassInfo>数组中的下标多 1，而我在调用有关班级的函数时用于记录对象在数组中位置编号的变量，有时使用的结构体中的成员 class_num，而有时使用的是 vector<ClassInfo>中的数组下标，因此造成了数组越界错误。

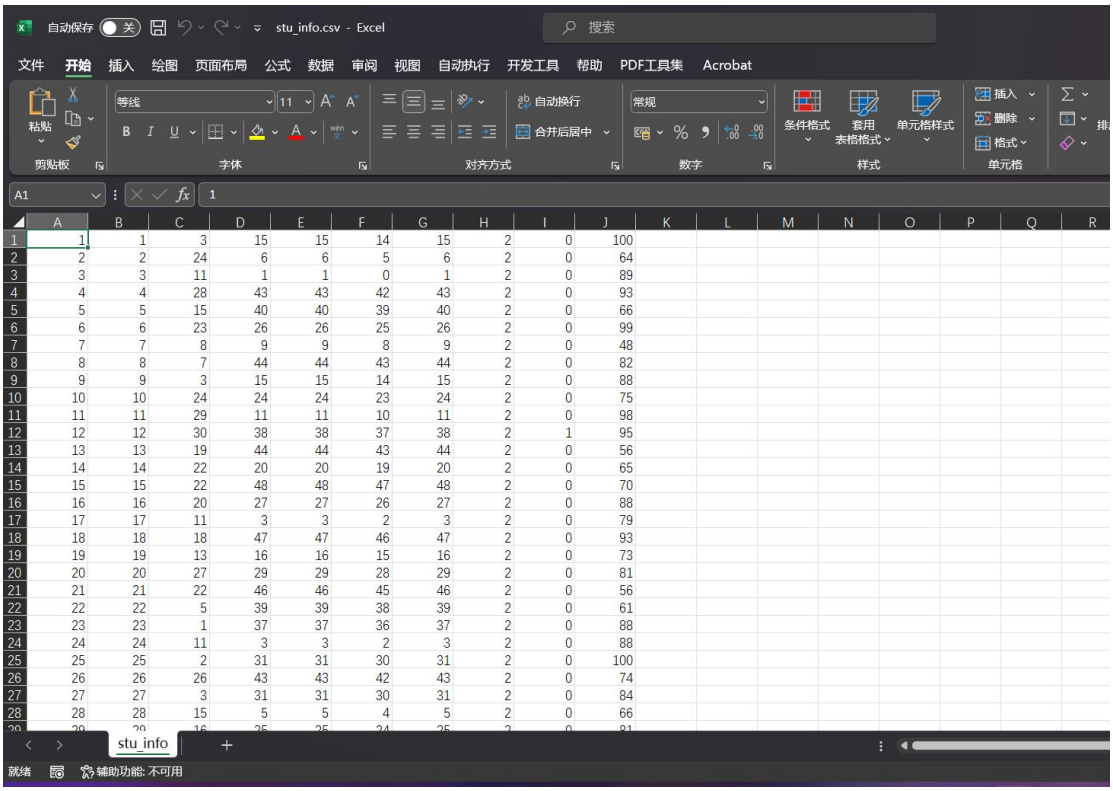
3、退出函数的错误：在学生和教师的操作界面中，我使用了一个用于退出当前界面返回主界面的函数，用户在执行该功能时需要输入 1 或 0 来确定自己是否退出。最开始这个退出函数的返回值是 void 类型，因此无论用户输入的数字是多少最终都必将会 return，意味着这个函数并没有发挥实际作用，用户仍然会停留在当前界面。于是我将返回值改为 int 类型，通过用户输入的数字返回 1 或 0，再将这个返回值在操作界面对应的函数中进行判断，如果返回值为 1 那么操作界面函数将会直接 return，实现真正返回了主界面。

5. 测试结果与分析

我的测试文件分别是 "D:\ASUS\Desktop\stu_info.csv" 与 "D:\ASUS\Desktop\tch_info.csv"，分别用于存储学生成绩和教师课程信息。为了便于测试，我设置 48 个教师，将教师姓名与工号均设置成数字 1—48，院系设置为 0—47，所教授课程设置为 1—48，学分均为 2，均不自带 P/F。设置 1000 名学生，学生姓名和学号相同且依次为 1—1000，设置 30 个班级，每个学生的班级均为 1—30 的随机数，每位

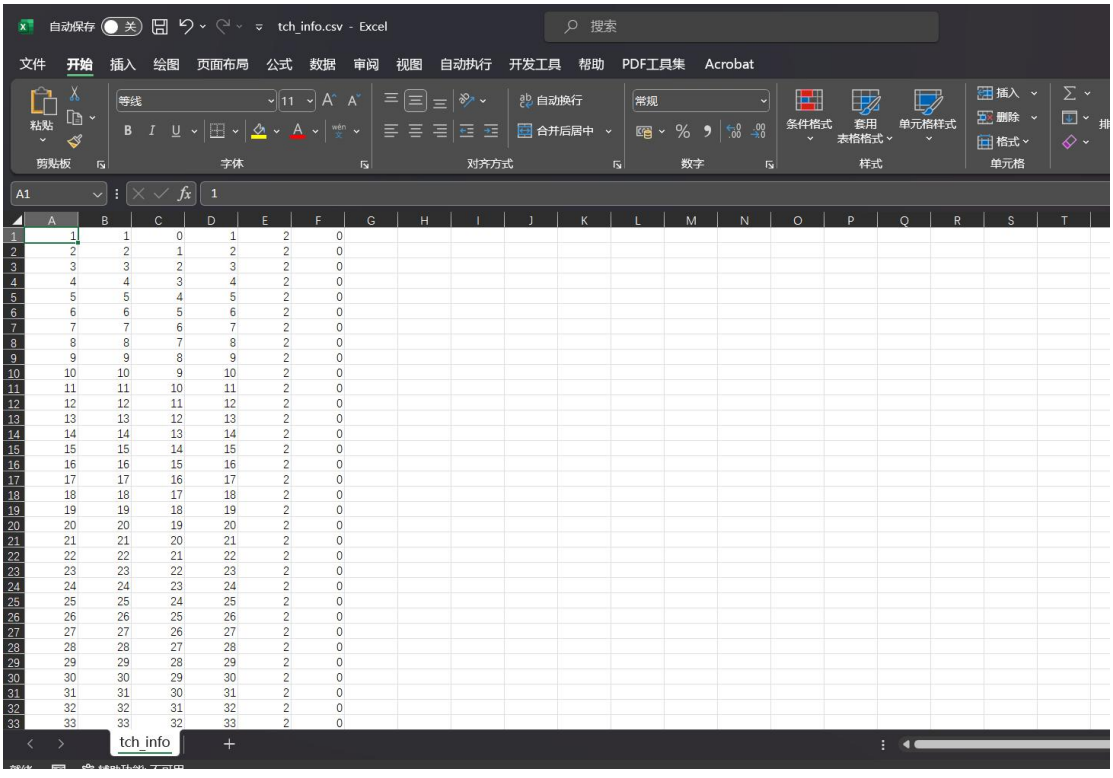
学生随机选择一门课程且课程成绩为 45—100 的随机数。

学生成绩信息文件：



	A	B	C	D	E	F	G	H	I	J
1	1	1	3	15	15	14	15	2	0	100
2	2	2	24	6	6	5	6	2	0	64
3	3	3	11	1	1	0	1	2	0	89
4	4	4	28	43	43	42	43	2	0	93
5	5	5	15	40	40	39	40	2	0	66
6	6	6	23	26	26	25	26	2	0	99
7	7	7	8	9	9	8	9	2	0	48
8	8	8	7	44	44	43	44	2	0	82
9	9	9	3	15	15	14	15	2	0	88
10	10	10	24	24	24	23	24	2	0	75
11	11	11	29	11	11	10	11	2	0	98
12	12	12	30	38	38	37	38	2	1	95
13	13	13	19	44	44	43	44	2	0	56
14	14	14	22	20	20	19	20	2	0	65
15	15	15	22	48	48	47	48	2	0	70
16	16	16	20	27	27	26	27	2	0	88
17	17	17	11	3	3	2	3	2	0	79
18	18	18	18	47	47	46	47	2	0	93
19	19	19	13	16	16	15	16	2	0	73
20	20	20	27	29	29	28	29	2	0	81
21	21	21	22	46	46	45	46	2	0	56
22	22	22	5	39	39	38	39	2	0	61
23	23	23	1	37	37	36	37	2	0	88
24	24	24	11	3	3	2	3	2	0	88
25	25	25	2	31	31	30	31	2	0	100
26	26	26	26	43	43	42	43	2	0	74
27	27	27	3	31	31	30	31	2	0	84
28	28	28	15	5	5	4	5	2	0	66

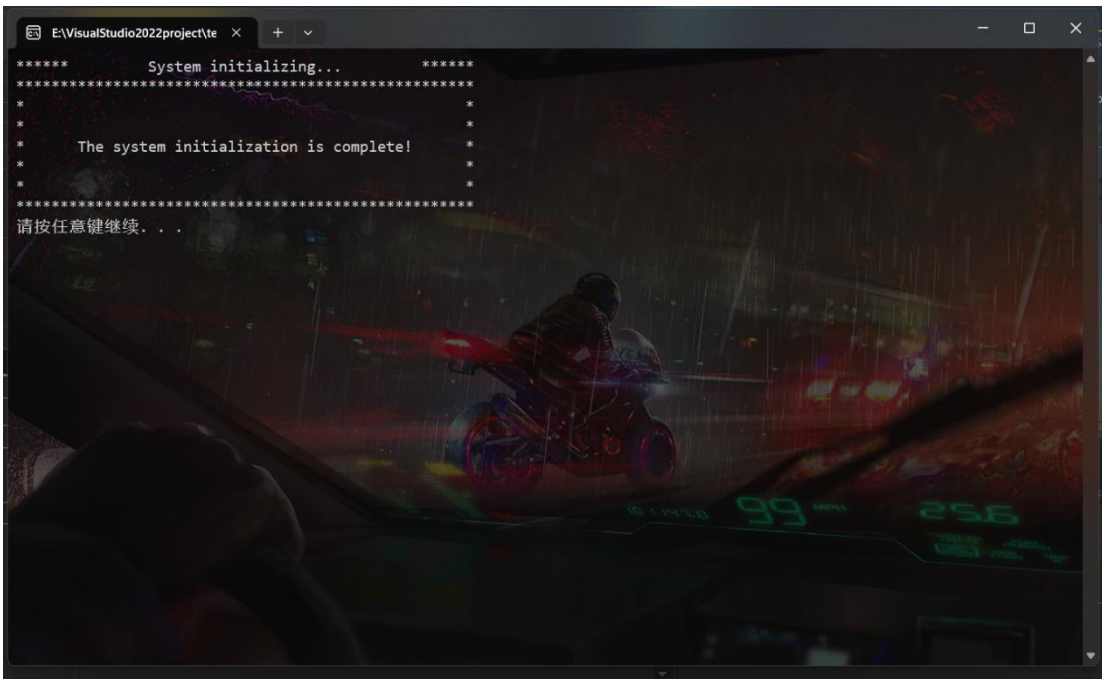
教师课程信息文件



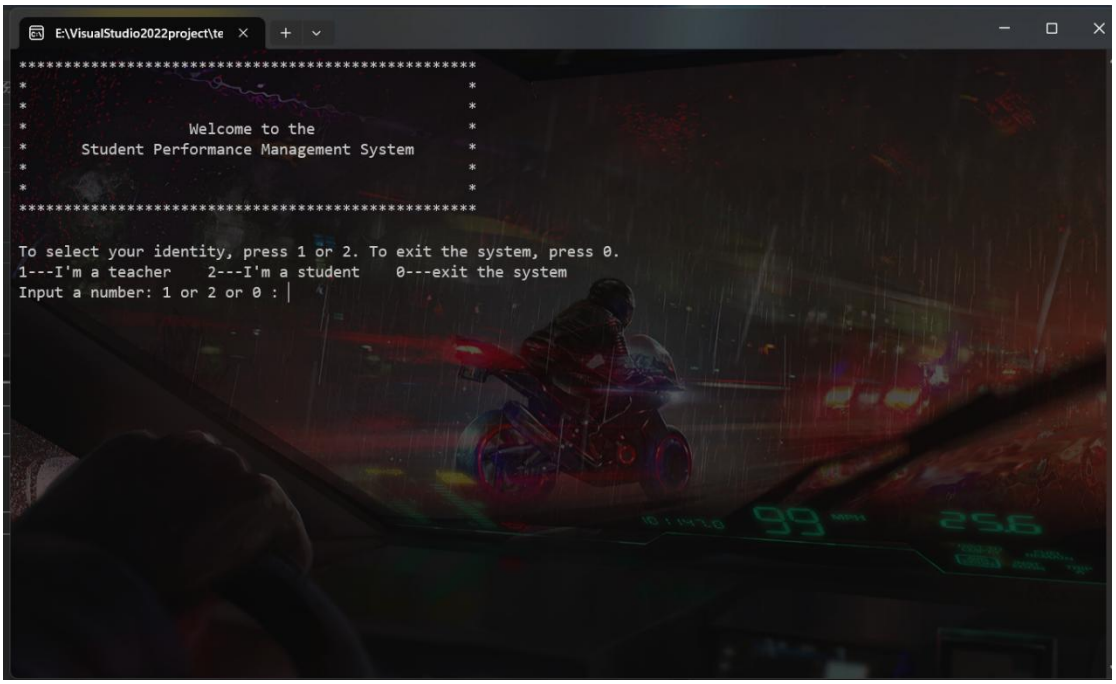
	A	B	C	D	E	F
1	1	1	0	1	2	0
2	2	2	1	2	2	0
3	3	3	2	3	2	0
4	4	4	3	4	2	0
5	5	5	4	5	2	0
6	6	6	5	6	2	0
7	7	7	6	7	2	0
8	8	8	7	8	2	0
9	9	9	8	9	2	0
10	10	10	9	10	2	0
11	11	11	10	11	2	0
12	12	12	11	12	2	0
13	13	13	12	13	2	0
14	14	14	13	14	2	0
15	15	15	14	15	2	0
16	16	16	15	16	2	0
17	17	17	16	17	2	0
18	18	18	17	18	2	0
19	19	19	18	19	2	0
20	20	20	19	20	2	0
21	21	21	20	21	2	0
22	22	22	21	22	2	0
23	23	23	22	23	2	0
24	24	24	23	24	2	0
25	25	25	24	25	2	0
26	26	26	25	26	2	0
27	27	27	26	27	2	0
28	28	28	27	28	2	0
29	29	29	28	29	2	0
30	30	30	29	30	2	0
31	31	31	30	31	2	0
32	32	32	31	32	2	0
33	33	33	32	33	2	0

调试过程中终端截图如下：

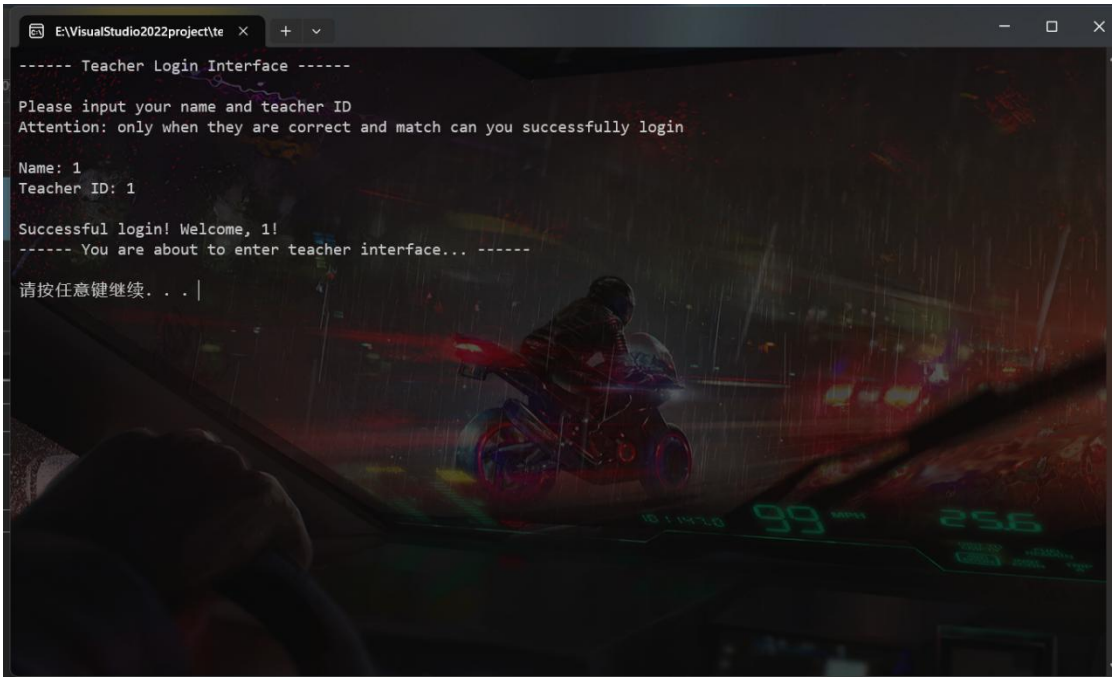
初始化界面：



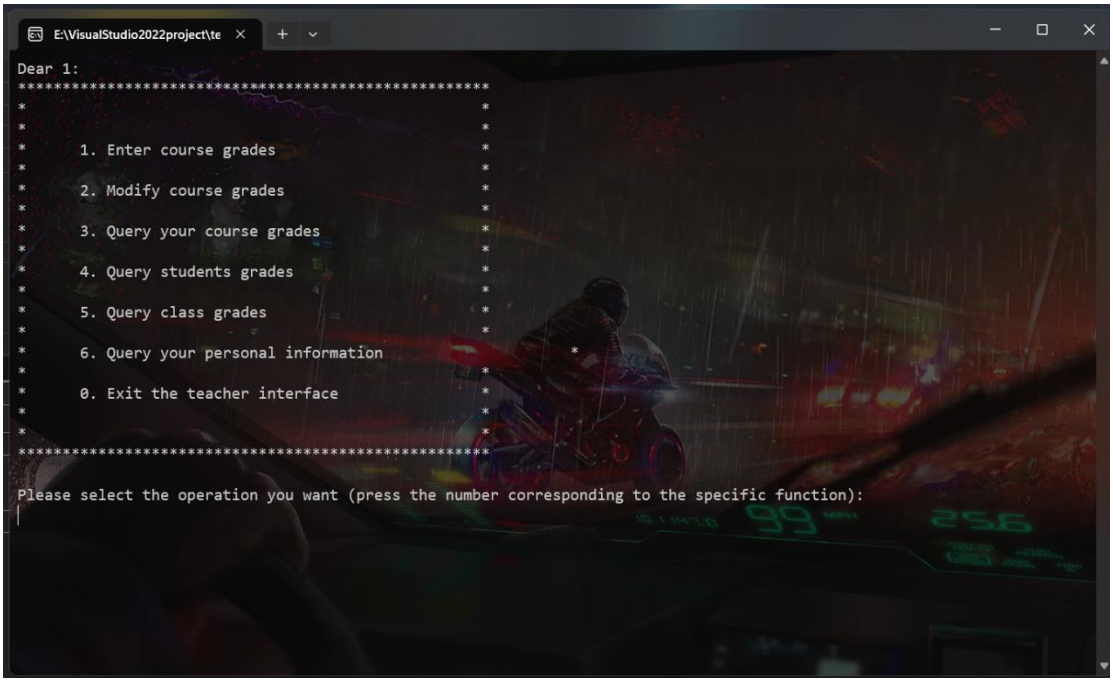
主界面：



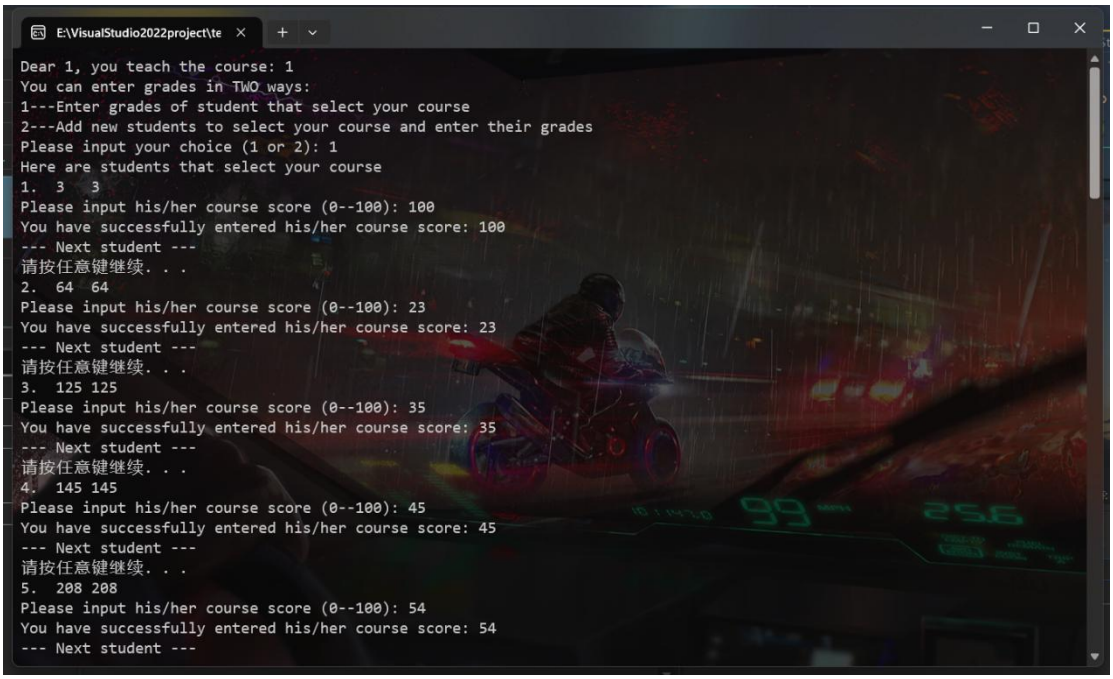
教师登录界面：



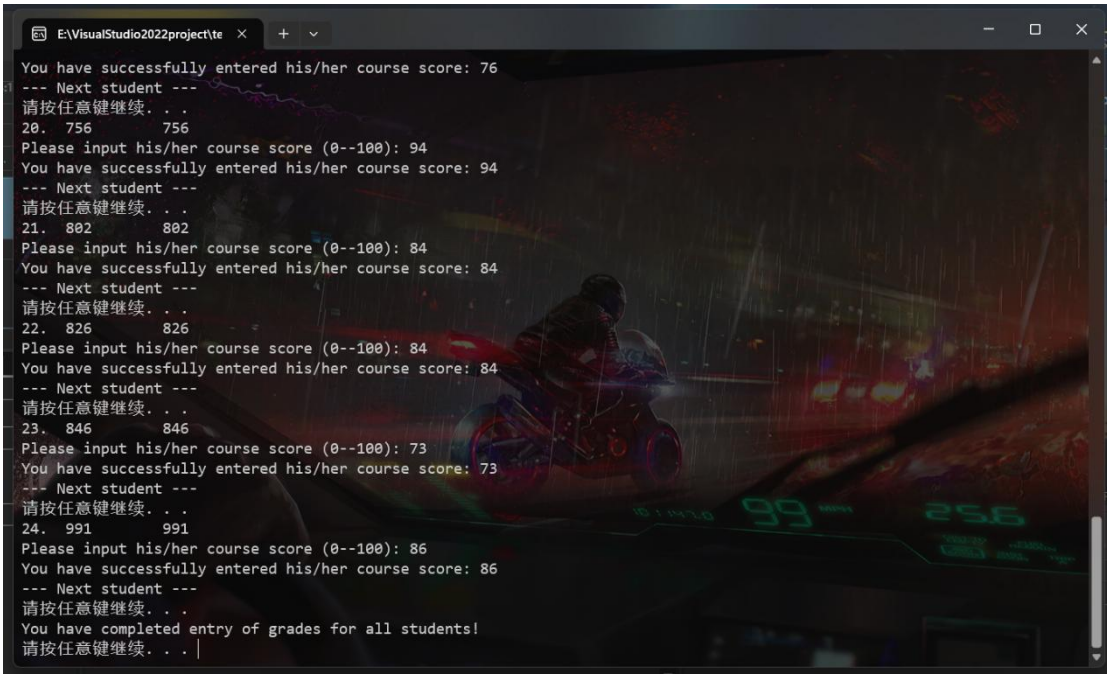
教师主界面：



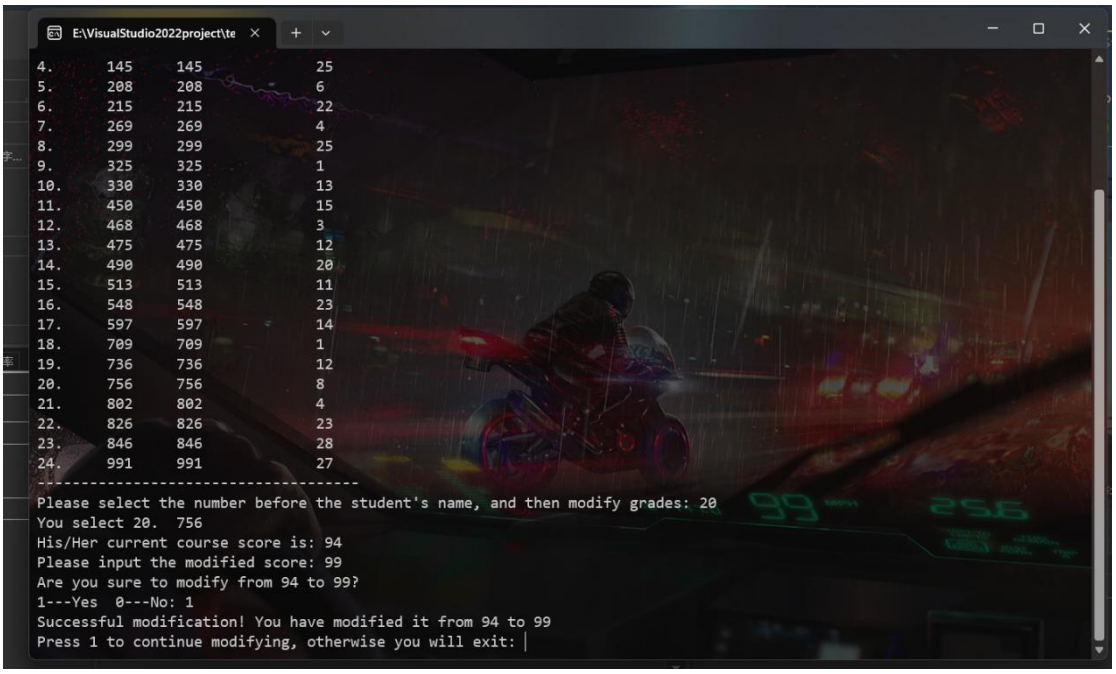
教师录入学生成绩：



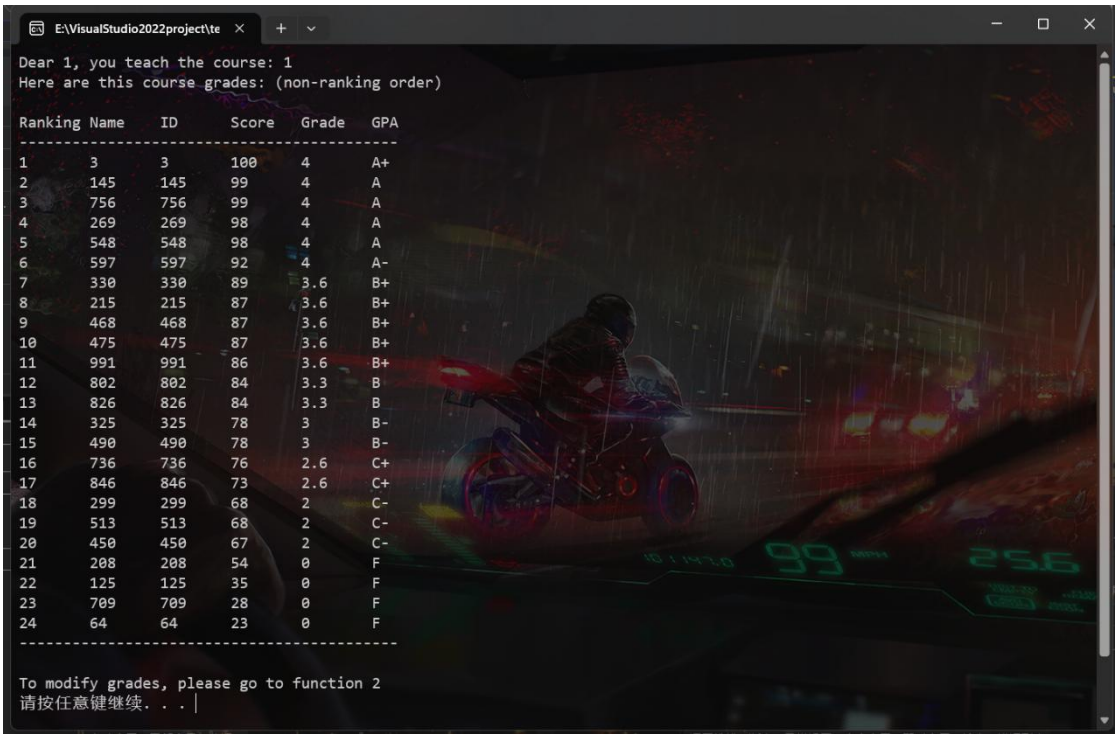
教师录入学生成绩完成：



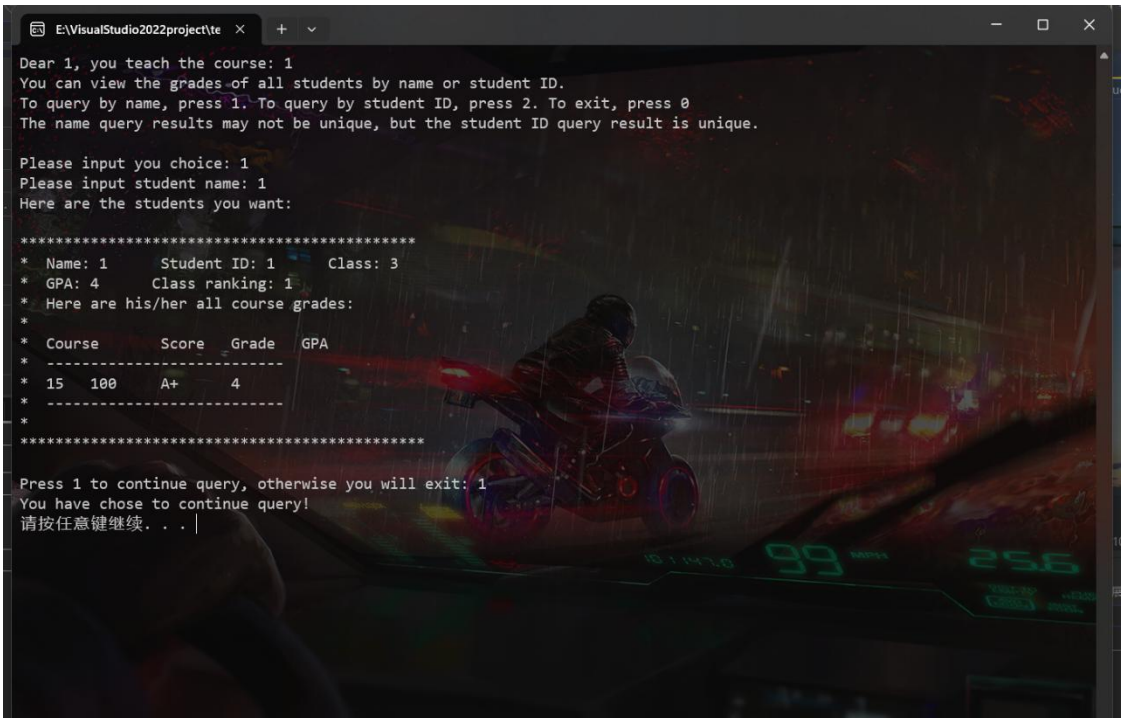
教师修改学生成绩：



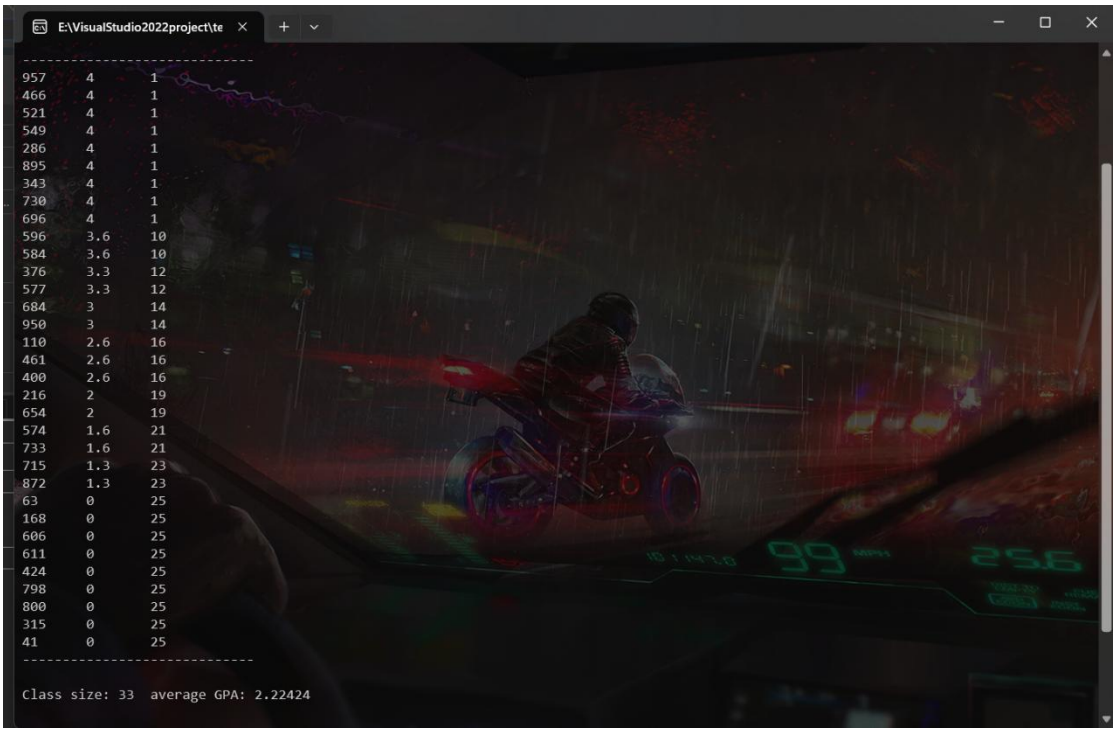
教师查询课程成绩：



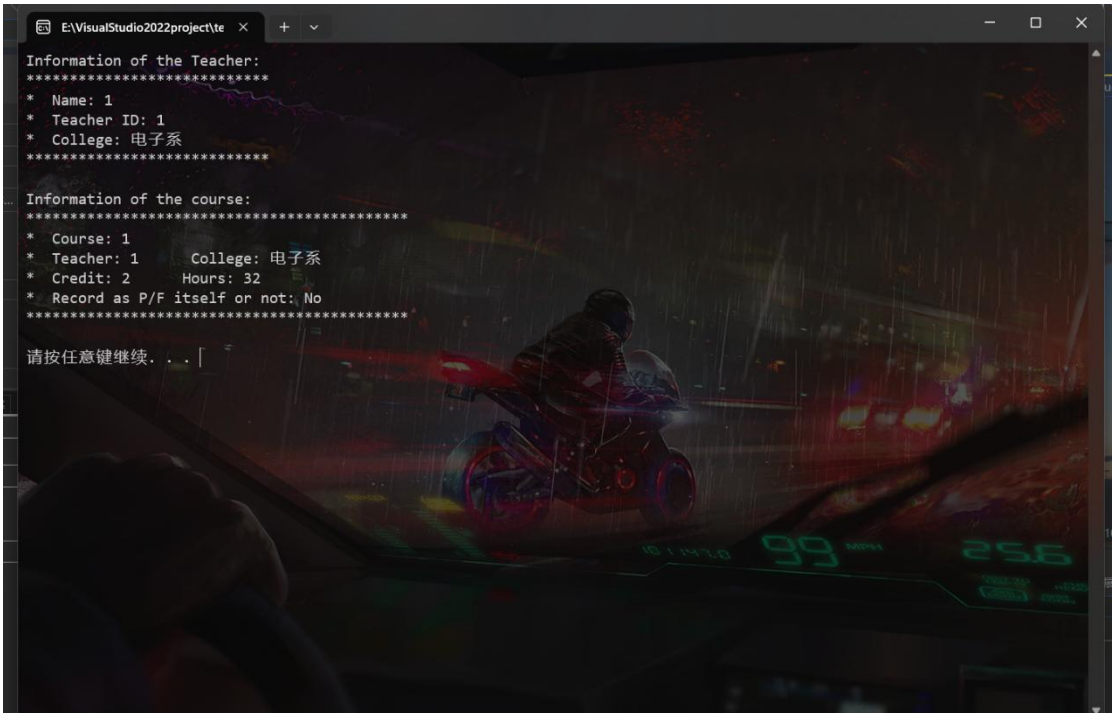
教师查询学生成绩：



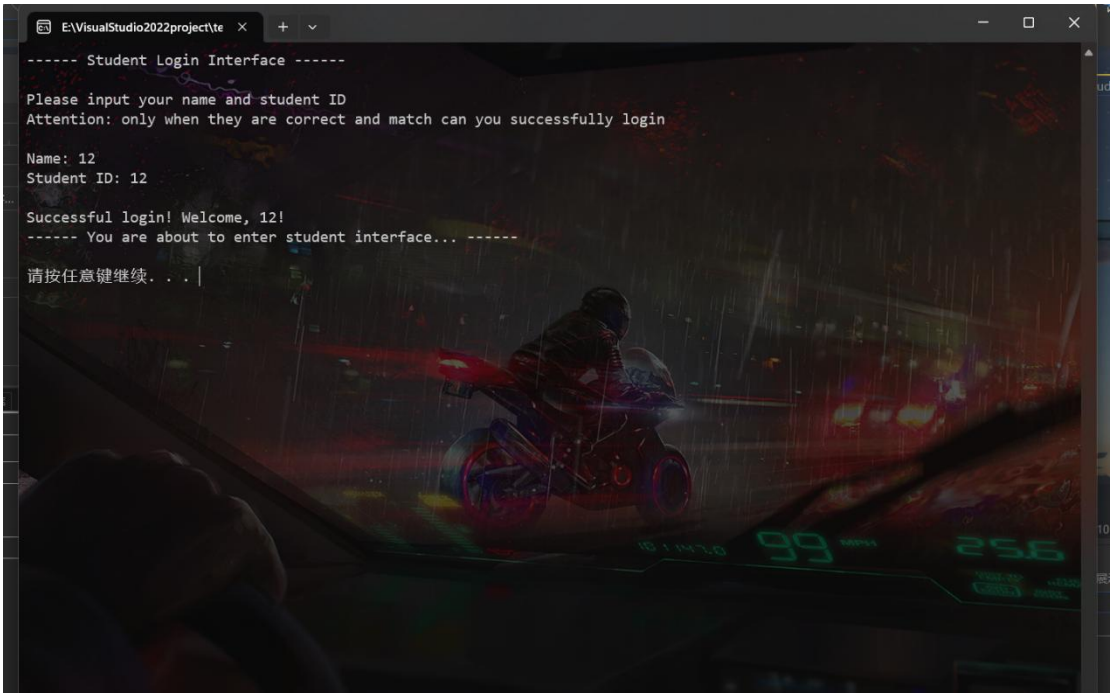
教师查询班级成绩：



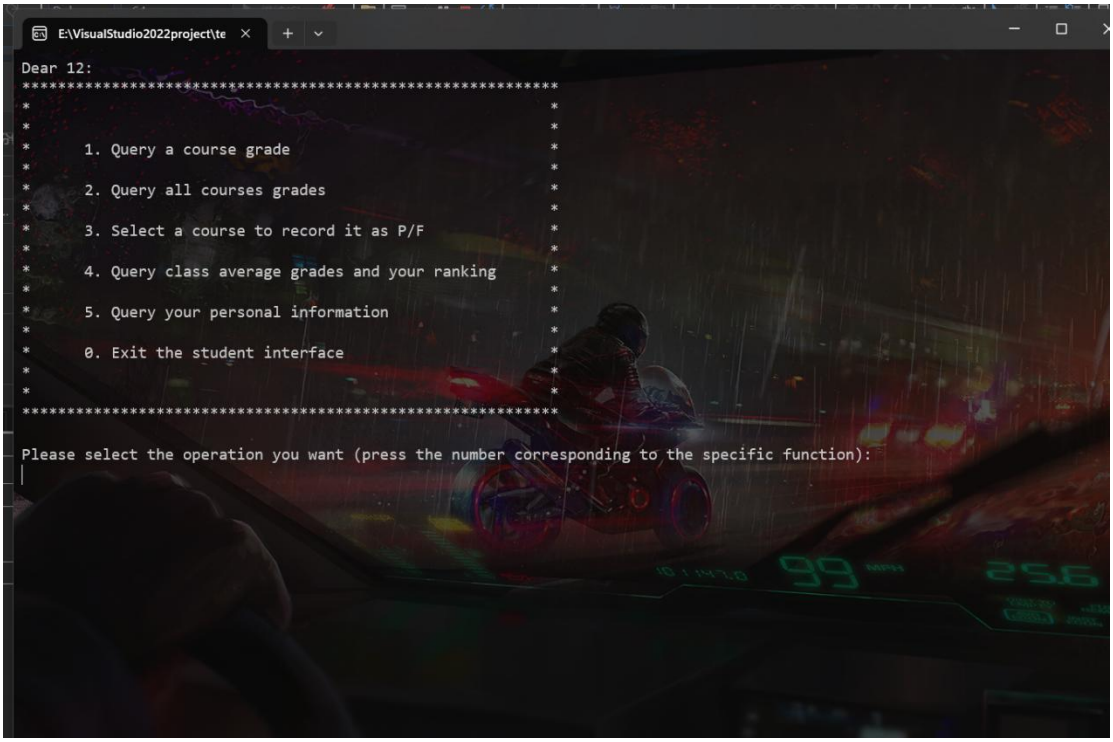
教师查看自己个人信息：



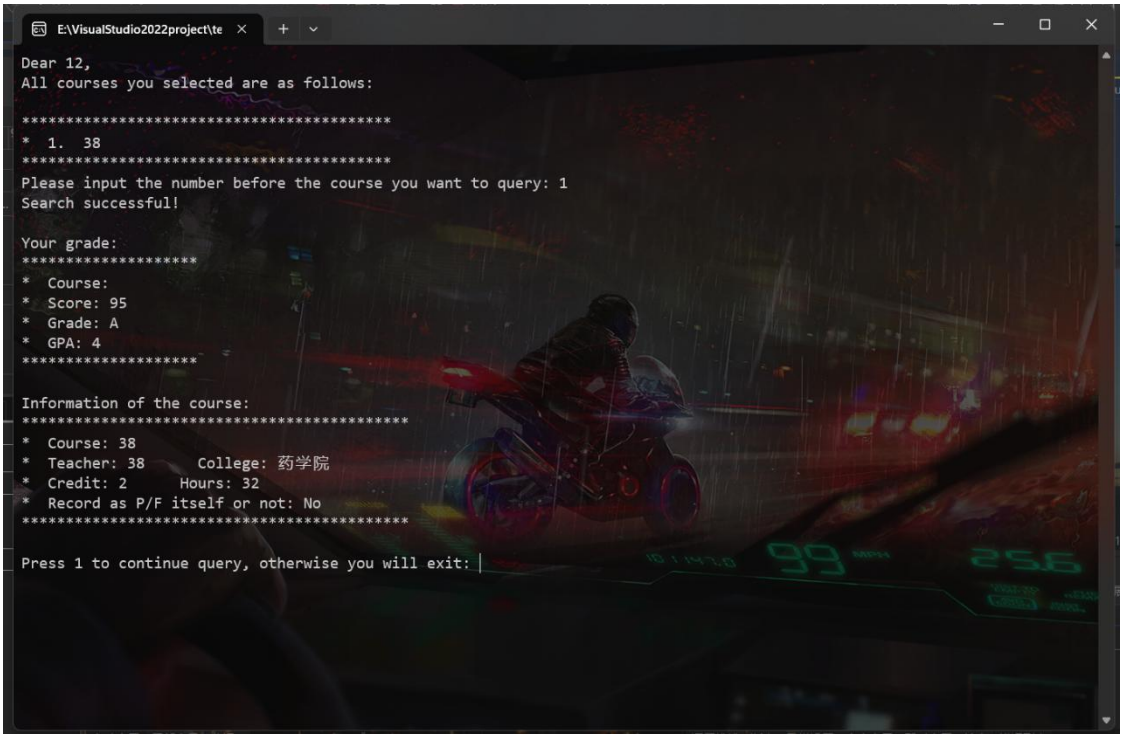
学生登录界面：



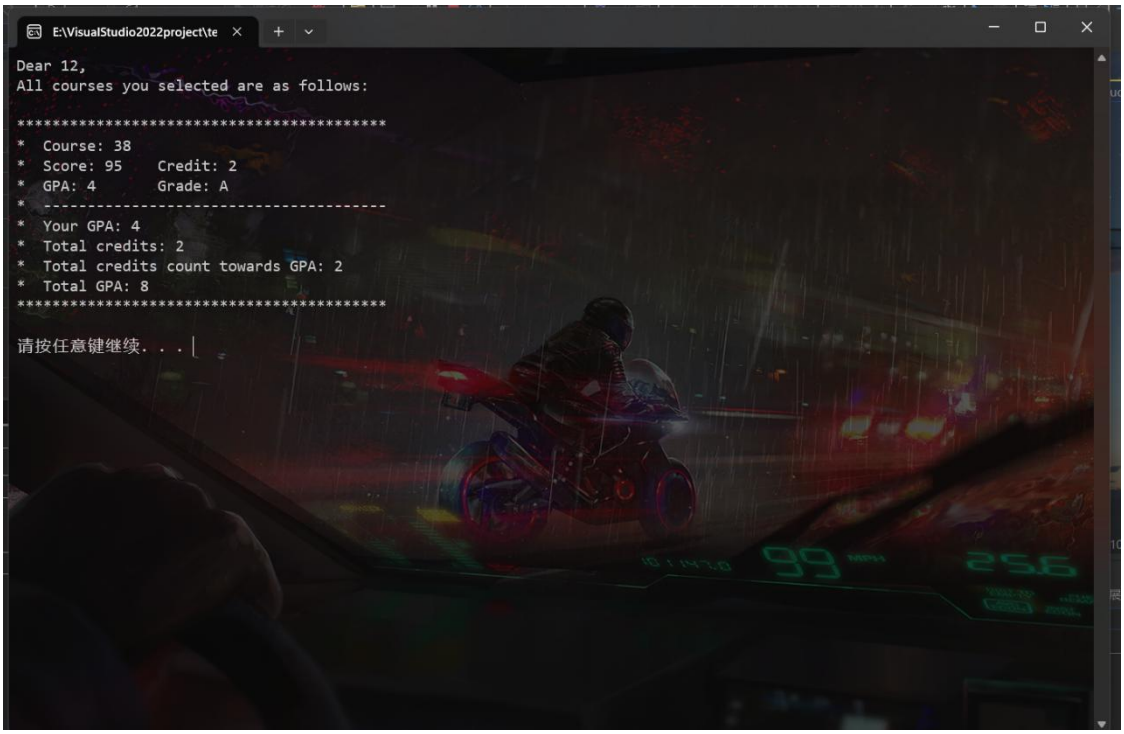
学生主界面：



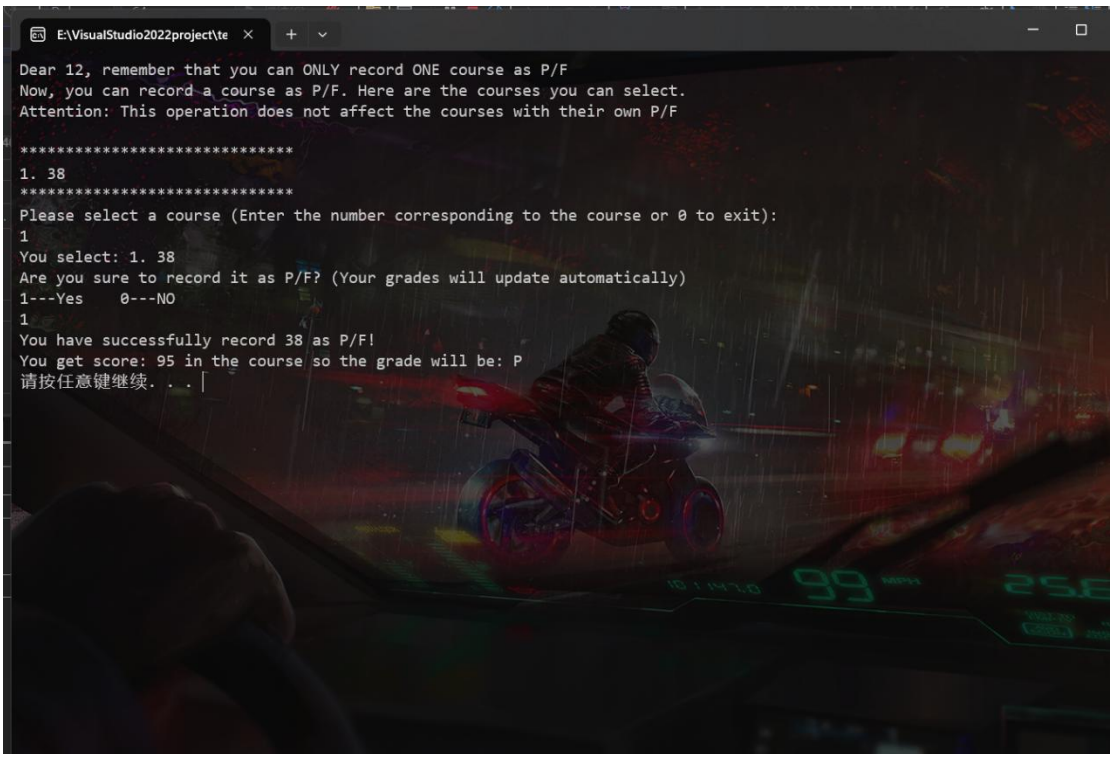
查询单门课程成绩：



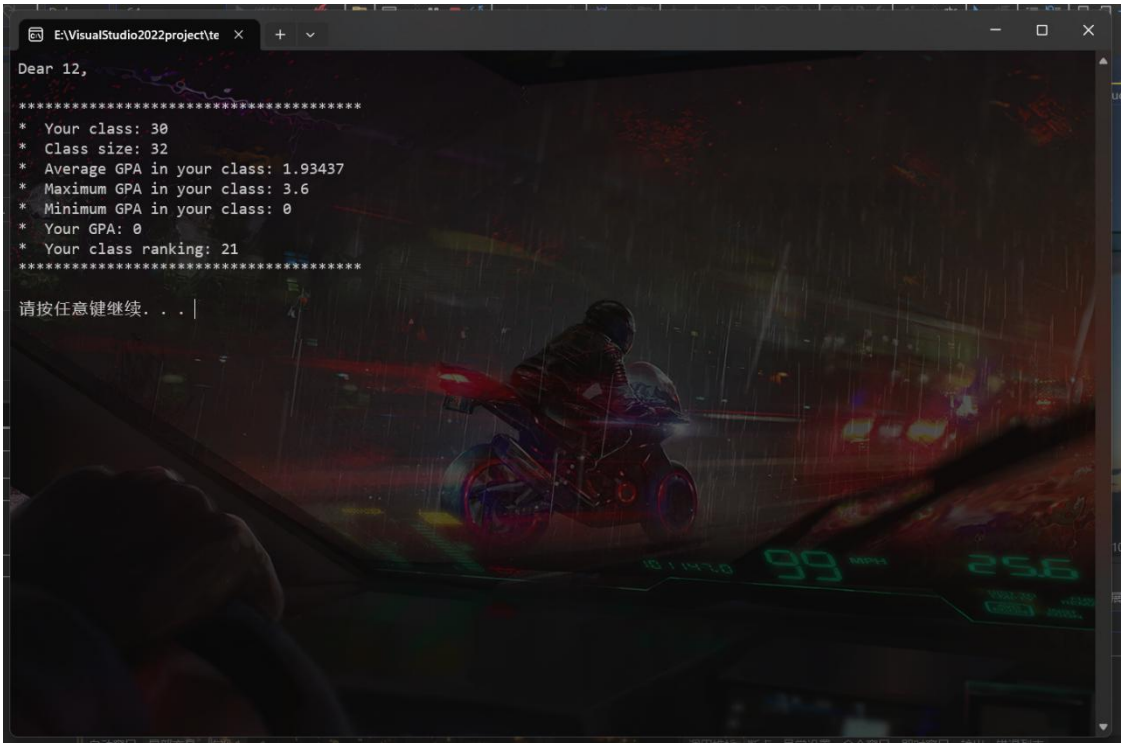
查询所有课程成绩：



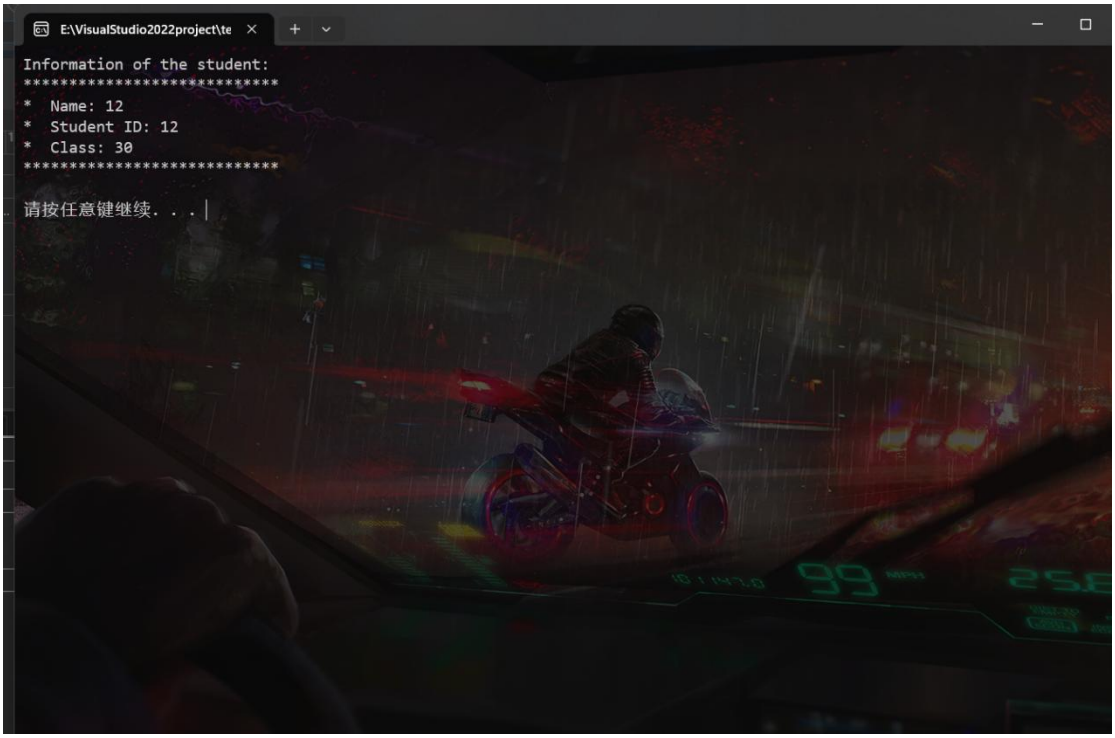
选择一门课程记为 P/F：



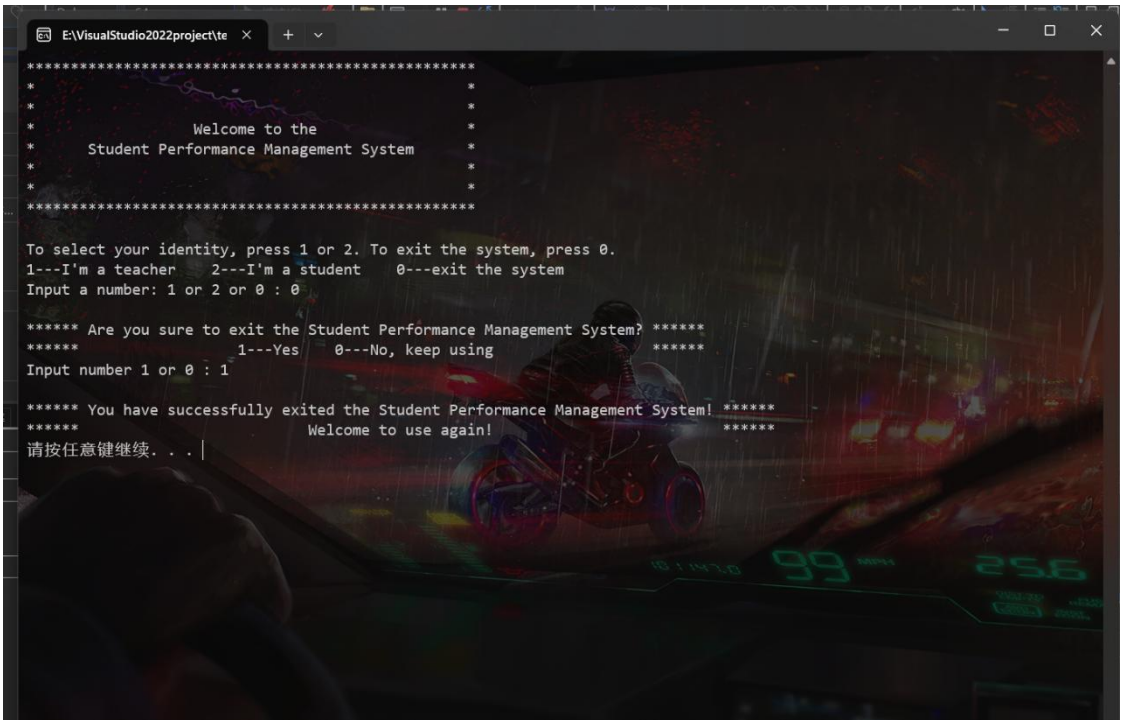
查询均绩和班级排名：



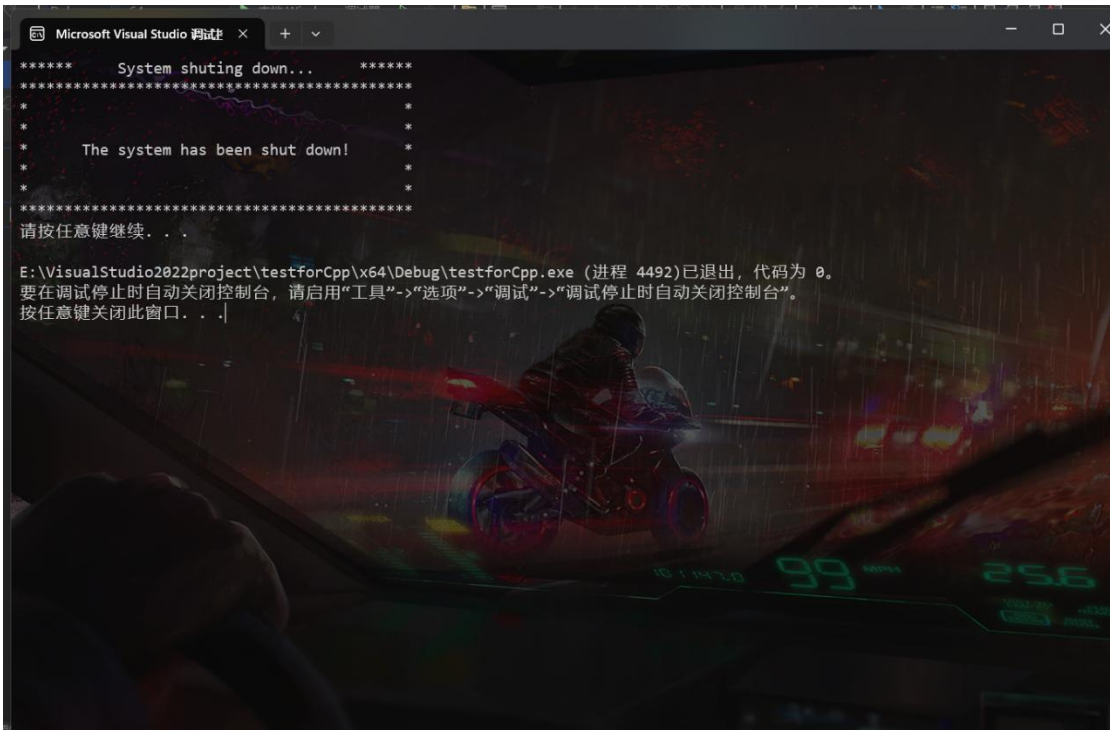
学生查看自己个人信息：



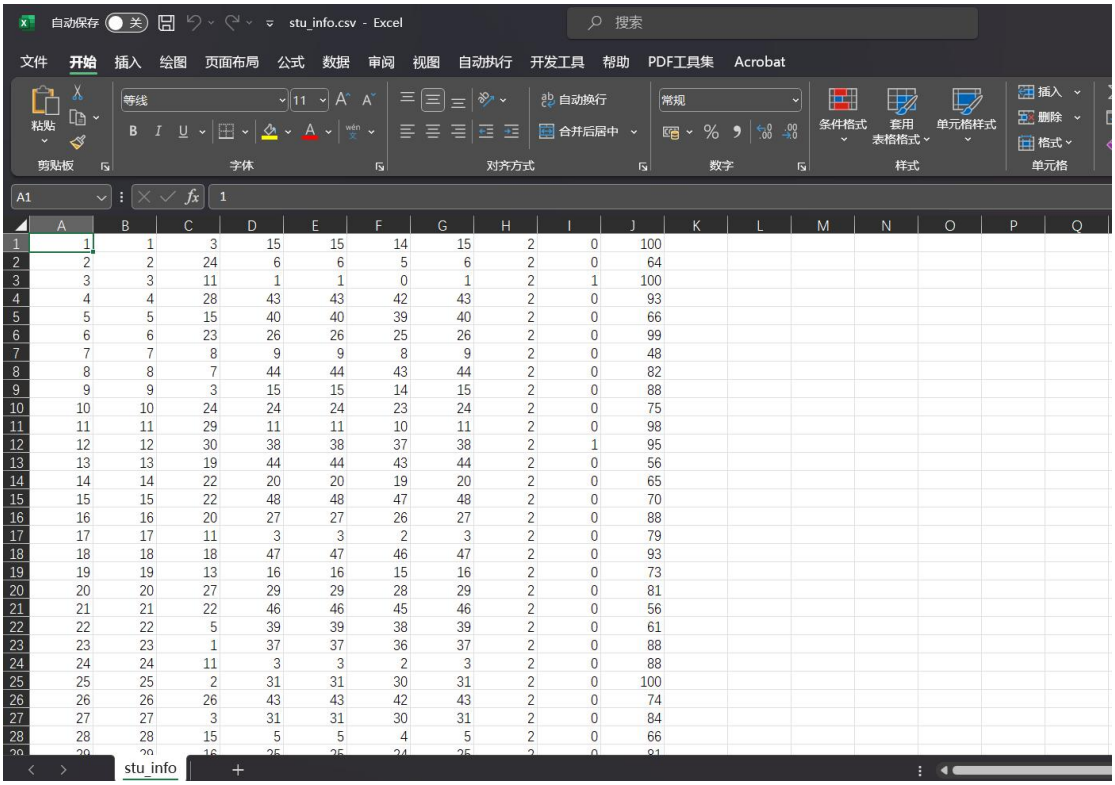
退出系统并确认：



系统关闭：

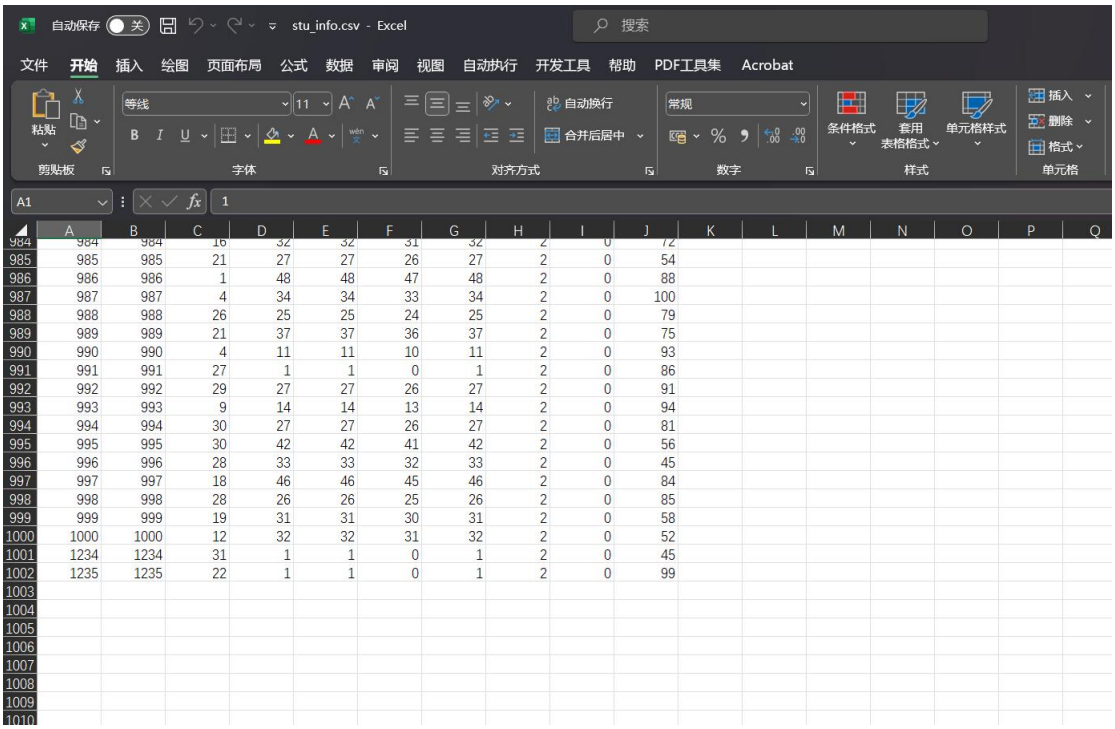


经过测试后，再次打开信息文件，发现数据确实发生了更改，证明调试和测试成功。



	A	B	C	D	E	F	G	H	I	J
1	1	1	3	15	15	14	15	2	0	100
2	2	2	24	6	6	5	6	2	0	64
3	3	3	11	1	1	0	1	2	1	100
4	4	4	28	43	43	42	43	2	0	93
5	5	5	15	40	40	39	40	2	0	66
6	6	6	23	26	26	25	26	2	0	99
7	7	7	8	9	9	8	9	2	0	48
8	8	8	7	44	44	43	44	2	0	82
9	9	9	3	15	15	14	15	2	0	88
10	10	10	24	24	24	23	24	2	0	75
11	11	11	29	11	11	10	11	2	0	98
12	12	12	30	38	38	37	38	2	1	95
13	13	13	19	44	44	43	44	2	0	56
14	14	14	22	20	20	19	20	2	0	65
15	15	15	22	48	48	47	48	2	0	70
16	16	16	20	27	27	26	27	2	0	88
17	17	17	11	3	3	2	3	2	0	79
18	18	18	18	47	47	46	47	2	0	93
19	19	19	13	16	16	15	16	2	0	73
20	20	20	27	29	29	28	29	2	0	81
21	21	21	22	46	46	45	46	2	0	56
22	22	22	5	39	39	38	39	2	0	61
23	23	23	1	37	37	36	37	2	0	88
24	24	24	11	3	3	2	3	2	0	88
25	25	25	2	31	31	30	31	2	0	100
26	26	26	26	43	43	42	43	2	0	74
27	27	27	3	31	31	30	31	2	0	84
28	28	28	15	5	5	4	5	2	0	66

（学生 3、12 进行了 P/F）



	A	B	C	D	E	F	G	H	I	J
985	985	985	21	27	27	26	27	2	0	54
986	986	986	1	48	48	47	48	2	0	88
987	987	987	4	34	34	33	34	2	0	100
988	988	988	26	25	25	24	25	2	0	79
989	989	989	21	37	37	36	37	2	0	75
990	990	990	4	11	11	10	11	2	0	93
991	991	991	27	1	1	0	1	2	0	86
992	992	992	29	27	27	26	27	2	0	91
993	993	993	9	14	14	13	14	2	0	94
994	994	994	30	27	27	26	27	2	0	81
995	995	995	30	42	42	41	42	2	0	56
996	996	996	28	33	33	32	33	2	0	45
997	997	997	18	46	46	45	46	2	0	84
998	998	998	28	26	26	25	26	2	0	85
999	999	999	19	31	31	30	31	2	0	58
1000	1000	1000	12	32	32	31	32	2	0	52
1001	1234	1234	31	1	1	0	1	2	0	45
1002	1235	1235	22	1	1	0	1	2	0	99

（末尾新增了 1234，1235 两个学生）

使用说明书：

系统启动后，先进行初始化过程，时间约 1.3 秒。初始化完成之后按任意键继续，进入系统主界面，在主界面可选择用户身份：选择 1 为老师，选择 2 为学生，选择 0 为退出系统，若选择其他数字则会被系统提示重新输入。

若用户选择 1 则进入教师登录界面，需要根据系统提示依次输入教师的姓名和工号，只有二者正确且完全匹配才能成功登录。如果输入有误则会被系统提示重新输入。进入教师界面之后教师可执行六项功能分别是（1）录入学生成绩、（2）修改学生成绩、（3）查看所教授课程的所有学生成绩、（4）查看某学生的所有课程成绩、（5）查看某班级所有学生的成绩、（6）查看自己的个人信息、（0）退出学生主界面。教师需根据所要执行功能输入功能前的数字，如果输入范围有误则会被系统提示重新输入。以下是 6 种具体教师功能的说明。

1、在录入学生成绩时，教师可选择两种方式：

（1）根据系统搜索给出选课学生名单，然后教师逐一输入学生的百分制成绩，系统会自动判断成绩是否在 0—100 内，否则将提示教师重新输入，输入成功后系统会提示已成功录入成绩，并自动计算对应的绩点、等级。之后系统提示继续输入，直到录入完成所有选课学生的成绩为止。

（2）在该课程中加入一位新选课的学生，教师需要创建该学生的姓名、学号、班级（若该学号已存在，则会提示教师重新输入），并自动将其加入所在班级的学生名单中（改班级不存在时将会创建一个新的班级），然后按照（1）中的方式输入该学生的成绩。

2、在修改学生成绩时（未录入的学生成绩默认为 0），根据系统搜索给出学生和成绩名单，教师选择并输入学生姓名前的序号，对其成绩进行修改（若该序号不在列表范围内，系统将提示教师重新输入），输入百分制成绩后，系统会自动判断成绩是否在 0—100 内，否则将提示教师重新输入，修改成功后系统会显示原成绩和现成绩，并提示是否继续修改，教师根据提示输入 1 可以继续选择下一位学生进行成绩修改，输入其它数字将会退出修改，返回教师主界面。

3、在查询课程成绩时，系统会自动判断教师所授课程以及选课学生名单，按照百分制成绩从高到低进行排名并依次输出，每行一个（同分学生排名相同），信息包括学生姓名、学号、百分制成绩、绩点、等级、选课内排名。此功能无需教师额外操作。

4、在查询学生课程成绩时，教师有两种方式查询某位学生成绩：按姓名或按学号。若输入 1 按姓名查找，输入 2 按学号查找，由于学生可能存在同名情况而学号是学生的

唯一凭证，因此姓名查询的结果可能不唯一，系统将会显示所有该姓名的学生成绩，学号查询的唯一。教师输入相应数字选择查询方式后，系统继续提示输入要查询的学生姓名/学号，搜索并输出符合条件的学生（该姓名/学号不存在时将提示教师重新输入），包括个人信息（姓名、学号、班级）、所有已选课程成绩以及平均绩点、班级排名。系统输出完成后，教师可根据提示输入 1 继续查询其他学生的成绩（需要重新选择查询方式：姓名/学号），若输入其他数字将会退出查询，返回教师主界面。

5、在查询班级学生成绩时，教师可直接输入班级号码，系统将自动搜索该班级学生（若该班级不存在将会提示教师重新输入），并按照绩点由高到低进行排名并输出，每行一个（绩点相同的学生排名相同），并在末尾计算该班级整体的人数、平均绩点。系统输出完成后，教师可根据提示输入 1 继续查询其他班级的学生成绩，若输入其他数字将会退出查询，返回教师主界面。

6、查看自己的个人信息时，系统将自动输出教师姓名、工号、所在院系、所授课程名称、课程学分、课程是否自带记 P/F。此功能无需教师额外操作。

7、退出教师主界面，退出登录，返回系统主界面，再次使用时需要选择身份并进行登录。系统将会提示教师是否确定退出，教师若输入 1 则会退出登录，输入其他数字则会取消退出，继续留在当前的教师主界面。

若用户选择 2 则进入学生登录界面，需要根据系统提示依次输入学生的姓名和学号，只有二者正确且完全匹配才能成功登录。如果输入有误则会被系统提示重新输入。进入学生界面之后学生可执行六项功能分别是（1）查询某门课程成绩、（2）查看所有课程成绩、（3）选择一门课程记为 P/F、（4）查看自己的均绩班级排名、（5）查看自己的个人信息、（0）退出学生主界面。学生需根据所要执行功能输入功能前的数字，如果输入范围有误则会被系统提示重新输入。以下是 5 种具体学生功能的说明。

1、在查询自己所选某门课程的成绩时，系统会自动搜索该学生所选课程并每行逐个输出，学生可输入课程前方的序号对相应课程的成绩进行查询（若序号不在列表范围内系统将提示学生重新输入），查询结果会显示学生在该门课程取得的百分制成绩、绩点、等级，并附加课程信息（授课教师姓名、院系、学分、是否自带记 P/F）。输出完成后，学生可根据系统提示，输入 1 继续查询其它课程成绩，否则将会退出查询返回学生主界面。

2、在查看自己所选所有课程成绩时，系统将会自动搜索并输出已选课程的百分制成绩、绩点、等级，在末尾计算并输出该学生的 GPA、学分数、计入 GPA 学分数和总学分绩。此功能无需学生额外操作。

3、在选择一门课程记为 P/F，系统会先判断该学生之前是否已经将一门课程记为 P/F，如果有，系统提示学生选择 P/F 另一门课程或维持原 P/F 选择不变：

（1）若输入 0 则会选择一门课程考核方式记为 P/F 并推出该功能界面。

（2）若输入 1 则会取消原先 P/F 的课程，然后列出学生已选课程，学生选择可课程名前的序号对相应课程进行 P/F（自带 P/F 的课程也在选择范围之内，但若学生选择将其记为 P/F 将不产生任何影响，只算作使用过了 P/F 的机会），输入相应数字后系统会再次提示学生是否确定将其记为 P/F。记录成功后学生将退出并返回学生主界面。若学生在课程序号输入 0，则会视为放弃本次 P/F 选择。

（3）若输入 1、0 之外的不合法数字，系统将会提示学生重新输入。

无论学生进行何种操作，退出该功能时，该学生的成绩、排名等信息将会自动更新。

4、在查询均绩和班级排名时，系统将自动输出学生班级号码、该学生的平均绩点、班级排名、班级人数、班级均绩、最高均绩和最低均绩，但为保护同学之间隐私，不会显示最高、最低均绩对应的学生信息和其他所有学生信息。此功能无需学生额外操作。

5、退出学生主界面，退出登录，返回系统主界面，再次使用时需要选择身份并进行登录。系统将会提示学生是否确定退出，学生若输入 1 则会退出登录，输入其他数字则会取消退出，继续留在当前的学生主界面。

若用户选择 0，系统提示用户是否确定退出系统，若用户输入 1 则表示确定，退出系统，输入 0 表示取消，继续使用，输入其他不合法数字将会被提示重新输入。

最终用户退出系统时，系统将会执行关机过程，时间约 1.3 秒，最后用户按任意键继续，完成此次使用。

编程体会：

这次的程序设计我结合了课程所学知识，并加以课外自主搜索最终实现了预期完整功能。考虑到代码的简洁性、编写代码的难易程度、程序的可执行性，我对许多细节地方进行了优化。

字符串类型并没有使用 char*或 char[]，它们的初始化以及释放都需要考虑多种情况如字符串是否为空、指针是否指向同一地址等，因此我使用了 C++中的 string 字符串类型，它可以在字符串之间直接进行赋值、比较、拼接和运算符操作，不需要进行运算符重载，使用起来更为方便。

数组方面我并没有使用传统的数组类型和 new 关键字创建动态数组，而是使用容器 vector 动态数组，它本身具有许多便于使用的操作函数，比如我在课程排名和班级排名中使用 sort 排序函数，方便易懂，避免了冒泡排序等排序方式的复杂与传值时出现的问题。在遍历数组时没有使用普通的 for 循环，而是使用 Range-Based for 循环，除了不能直接使用数组下标之外，使得程序更容易编写，可读性较强。

在函数的形参中尽量使用 const 关键字以保护读取的数据不被无意修改。多使用引用&，传递参数时直接传递地址而非复制副本，提高程序的运行效率。在用户需要执行特定功能时，我都在功能函数中内置 while 循环，用于判断用户输入的数据数字是否合法，若不合法将会重新输入直至合法为止，这样避免了用户无意中的错误输入导致程序运行错误的问题。

在利用变量存储学生成绩时，我最开始是想在 StudentCourse 类中定义直接定义一个 vector<Course>成员，对应一个学生类 Student，但是这样设计不符合继承与派生的理念：StudentCourse 继承了 Course，作为派生类本身就包含基类 Course 类的成员，但是这样做却又在 StudentCourse 中定义了基类 Course 的数组，所以之后改为：每个 Student 对象对应一个 StudentCourse 数组，用于记录他的所有课程成绩，再将他们结合均绩、排名等信息，一起存放入 StudentResults 这样一个结构体，再定义一个 StudentResults 的数组，即可记录所有学生的所有成绩。

然而，我的程序设计也有一些不足之处和需要改进的地方：

对于操作界面的语言，我最开始使用的是中文，而在后来程序的编写和调试中出现过几次莫名其妙的警告和错误，比如常量中不能含有转义序列等未曾见过的问题，编译器提示问题出现在似不可能存在错误的一个自定义的常量字符串，几次更改都没有解决。我上网搜索得知可能是编码和字符的问题，因此我将界面语言全部更换为英文，可能会增加用户的适应成本和使用难度，降低使用体验。

对于登录权限的问题，我的设计过于简单。教师或学生只需正确输入自己的姓名和工号或学号即可登录，并没有设置额外的密码来限定用户的使用权限，对于个人信息的保护措施不够，在实际应用中这样可能导致用户登录他人账号、学生登录教师账号违规操作等情况。因此我认为还需要在增加管理员权限，严格控制不同账号之间的界限。

在 Course 和 StudentCourse 类中均具有“课程名称”这一成员函数，出现了重复。我在类的设计上使用了抽象类、虚继承等方式，可以使用基类指针直接使用和访问基类和派生类的成员，但是在后续的编程中我并没有使用太多指针来进行操作，可能是因为指针的使用难度较大，因此在许多地方我都使用了引用&来代替。我认为指针是 C 语言

和 C++的精髓，指针使用的欠缺减少了我理解和熟悉指针的机会，也使得这个程序缺少了某些灵魂。另外，我虽然设置了各种类之间的运算符重载，但是在后面具体函数的实现中并没有过多使用，这也是未来我应该改进的地方。

我的程序设计中默认的是一个老师只教授一门课程，因此 Teacher 类对象和 Course 类对象其实是一一对应的，但显然这样并不符合实际，一位老师可以教授多门课程而我的程序并没有实现这样的功能。另外虽然老师可以增加学生来选他的课程，也就是说创建了新的学生个体，但是教师和课程都是从数据文件中导入的，在程序中并没有管理员权限可以增加教师和课程。

虽然我在用户输入的数字进行了范围是否合法的判断但是如果用户无意中输入了字母或者符号仍然无法处理因此可以使用异常处理：抛出异常并进行判断，然后进行执行相应解决措施。

6. 总结

大作业要求创建一个能够有效管理学生成绩的系统，起初，我对于类的设计有过许多想法，如何定义类能够有效且高效地存储和访问数据，并且符合各种程序设计的规范，通过仔细阅读项目需求和反复思考，我最终确定现在这种菱形继承的多继承方式，设计出 Student、Course 等 5 个类，并且我明白我不是为了使用多继承而使用多继承，是因为系统的核心是学生和成绩信息，因此决定了可以用一个类将学生信息和成绩信息结合起来，这样即满足了多继承的方式也满足系统的核心设计需求。“学生类”包括学生的个人信息，而“课程类”则包含课程信息以及它所继承的教师信息，二者共同派生出“学生成绩类”，这些类之间的关系决定了存储数据的方式与操作，通过这样的精心设计能够实现数据的便捷和快速读取。

在代码编写中我不断地在查找互联网资料解决出现的问题，并且学习和使用一些新的函数和算法，在运用课上所学知识的同时也进行了一些额外的拓展。由于我在整体设计上分出、封装多个函数模块，每编写完一个就进行调试和改进。这种模块化的设计让我更容易定位和解决问题，同时也提高了代码的可维护性。在最终的测试中也并没有遇到太多困难。当然，这次的程序设计也不是完美无缺的，有一些预期功能并没有完全实现，并且程序也有一些地方存在考虑不完善的地方，值得优化和改进，使得系统更急用

户友好、高效运行。

总之，这次的学生成绩管理系统大作业让我获得了宝贵的经验和教训。我明白了编程不仅仅是写代码，还包括设计、测试和维护系统，并且只有通过充分的测试才能够确保程序的稳定性和正确性。另外，还需要更加注重细节的处理，在细节上优化到极致，才能在系统运行时极大程度减少错误或意外。以上这些都对我的代码能力有一定的提升，之前的作业都是完成简单要求、检验课堂学习内容的小型代码，而这次的大作业则是面向用户需求、需要全方位考虑和设计的综合项目，是我对上一年程序设计学习的一个实践和总结，在之后的学习中我会继续运用所学，不断拓展汲取新知识。

附录：源程序清单

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <fstream>
#include <sstream>
#include <cstdlib>
#include <cmath>
#include <cstring>
#include <iomanip>
#include <vector>
#include <iterator>
#include <algorithm>
#include <functional>
#include <windows.h> //调用休眠函数
const int MAX_STU_NUM = 10000; //能够录入的最大学生数目
const int MAX_STRLEN = 1024; //该系统中出现的字符串最大长度
const std::string College[48] = { //院系编号
    "电子系", // 0
    "建筑学院", // 1
    "土木系", // 2
    "水利系", // 3
    "环境学院", // 4
    "机械系", // 5
    "精仪系", // 6
    "能动系", // 7
    "车辆学院", // 8
    "工业工程系", // 9
    "电机系", // 10
    "计算机系", // 11
```

```
    "自动化系", // 12
    "软件学院", // 13
    "集成电路学院", // 14
    "航院", // 15
    "工程物理系", // 16
    "化工系", // 17
    "材料学院", // 18
    "数学系", // 19
    "物理系", // 20
    "化学系", // 21
    "生命学院", // 22
    "地学系", // 23
    "交叉信息院", // 24
    "经管学院", // 25
    "公管学院", // 26
    "金融学院", // 27
    "外文系", // 28
    "法学院", // 29
    "新闻学院", // 30
    "马克思主义学院", // 31
    "人文学院", // 32
    "社科学院", // 33
    "美术学院", // 34
    "卫健学院", // 35
    "医学院", // 36
    "药学院", // 37
    "天文系", // 38
    "网络研究院", // 39
    "语言中心", // 40
    "新雅书院", // 41
    "未央书院", // 42
    "致理书院", // 43
    "日新书院", // 44
    "行健书院", // 45
    "为先书院", // 46
    "秀钟书院", // 47
};
const std::string Grade[13] = { "A+","A","A-","B+","B","B-","C+","C","C-","D+","D","P","F" };
//程序开始运行时从文件中读取信息存入容器中，结束时将新的信息写入文件中保存
const std::string STUDENT_FILE_PATH = "D:\\ASUS\\Desktop\\stu_info.csv";
const std::string TEACHER_FILE_PATH = "D:\\ASUS\\Desktop\\tch_info.csv";
```

```
//抽象类：人员
class Person
{
public:
    virtual void displayInfo()const = 0;
    virtual ~Person() {};
private:
    std::string name;
    std::string id;
};


//记录学生基本信息的 Student 类
class Student :virtual public Person
{
private:
    std::string stu_name;//姓名
    std::string stu_id;//学号
    int stu_class;//班级
public:
    Student(std::string stu_name = "N/A", std::string stu_id = "N/A", int stu_class = 0)
    {
        this->stu_name = stu_name;
        this->stu_id = stu_id;
        this->stu_class = stu_class;
    }
    Student(const Student& other)
    {
        this->stu_name = other.stu_name;
        this->stu_id = other.stu_id;
        this->stu_class = other.stu_class;
    }
    virtual ~Student() {};
    void displayInfo()const override//打印学生信息
    {
        std::cout << "Information of the student: " << std::endl;
        std::cout << "****************************" << std::endl;
        std::cout << "*    Name: " << stu_name << std::endl;
        std::cout << "*    Student ID: " << stu_id << std::endl;
        std::cout << "*    Class: " << stu_class << std::endl;
        std::cout << "****************************\n" << std::endl;
        return;
    }
    void setStudentInfo(const std::string stu_name = "N/A", const std::string stu_id = "N/A",
```

```
const int stu_class = 0)
    {
        this->stu_name = stu_name;
        this->stu_id = stu_id;
        this->stu_class = stu_class;
        return;
    }//修改全部信息
    void setStudentName(const std::string new_name = "N/A") {
        this->stu_name = new_name;
        return;
    } //修改学生姓名
    void setStudentID(const std::string new_id = "N/A") {
        this->stu_id = new_id;
        return;
    } //修改学生学号
    void setStudentClass(const int new_class = 0) {
        this->stu_class = new_class;
        return;
    } //修改学生班级
    std::string getStudentName()const { return this->stu_name; }
    std::string getStudentID()const { return this->stu_id; }
    int getStudentClass()const { return this->stu_class; }
};


//记录老师基本信息的 Teacher 类
class Teacher :virtual public Person
{
private:
    std::string tch_name;//老师姓名
    std::string tch_id;//老师工号
    int tch_college;//老师院系编号
public:
    Teacher(std::string tch_name = "N/A", std::string tch_id = "N/A", int tch_college = 10)
    {
        this->tch_name = tch_name;
        this->tch_id = tch_id;
        this->tch_college = tch_college;
    }
    Teacher(const Teacher& other)
    {
        this->tch_name = other.tch_name;
        this->tch_id = other.tch_id;
        this->tch_college = other.tch_college;
```

```
    }
    virtual ~Teacher() {};
    void displayInfo()const override//打印教师信息
    {
        std::cout << "Information of the Teacher: " << std::endl;
        std::cout << "****************************" << std::endl;
        std::cout << "*    Name: " << tch_name << std::endl;
        std::cout << "*    Teacher ID: " << tch_id << std::endl;
        std::cout << "*    College: " << College[tch_college] << std::endl;
        std::cout << "****************************\n" << std::endl;
    }
    void setTeacherInfo(const std::string tch_name = "N/A", const std::string tch_id = "N/A",
const int tch_college = 0)
    {
        this->tch_name = tch_name;
        this->tch_id = tch_id;
        this->tch_college = tch_college;
    } //修改教师信息
    void setTeacherName(const std::string new_name = "N/A") {
        this->tch_name = new_name;
        return;
    } //修改教师姓名
    void setTeacherID(const std::string new_id = "N/A") {
        this->tch_id = new_id;
        return;
    } //修改教师工号
    void setTeacherCollege(const int new_college = 0) {
        this->tch_college = new_college;
        return;
    } //修改教师院系
    std::string getTeacherName()const { return this->tch_name; }
    std::string getTeacherID()const { return this->tch_id; }
    int getTeacherCollege()const { return this->tch_college; }
};


//记录课程基本信息的 Course 类
class Course :virtual public Teacher
{
private:
    std::string crs_name;//课程名
    int crs_credit;//课程学分
    bool isPF;//是否记 PF
public:
```

```
    Course(std::string crs_name = "N/A", int crs_credit = 0, bool isPF = false)
    {
        this->crs_name = crs_name;
        this->crs_credit = crs_credit;
        this->isPF = isPF;
    }
    Course(const Course& other) :Teacher(other), crs_name(other.crs_name), crs_credit(other.crs_credit), isPF(other.isPF) {};
    virtual ~Course() {};
    void displayInfo()const override
    {
        std::cout << "Information of the course: " << std::endl;
        std::cout << "*****************************************" << std::endl;
        std::cout << "*    Course: " << crs_name << std::endl;
        std::cout << "*    Teacher: " << Teacher::getTeacherName() << "          College: " << College[Teacher::getTeacherCollege()] << std::endl;
        std::cout << "*    Credit: " << crs_credit << "          Hours: " << crs_credit * 16 << std::endl;
        std::cout << "*    Record as P/F itself or not: " << (isPF ? "Yes" : "No") << std::endl;
        std::cout << "*****************************************\n" << std::endl;
    }
    void setCourseInfo(const std::string tch_name = "N/A", const std::string id = "N/A", const int college = 0, const std::string crs_name = "N/A", const int crs_credit = 0, const bool isPF = false)
    {
        Teacher::setTeacherInfo(tch_name, id, college);
        this->crs_name = crs_name;
        this->crs_credit = crs_credit;
        this->isPF = isPF;
    } //修改课程信息
    void setCourseName(const std::string new_name = "N/A") {
        this->crs_name = new_name;
        return;
    } //修改课程名称
    void setCourseCredit(const int new_credit = 0) {
        this->crs_credit = new_credit;
        return;
    } //修改课程学分
    void setCourseIsPF(const bool new_isPF = false) {
        this->isPF = new_isPF;
        return;
    } //修改课程是否记 PF
    //修改基类中的成员
    void setTeacherName(const std::string new_name = "N/A")
```

```
{ Teacher::setTeacherName(new_name); }
    void setTeacherID(const std::string new_id = "N/A") { Teacher::setTeacherID(new_id); }
    void setTeacherCollege(const int new_college = 0)
{ Teacher::setTeacherCollege(new_college); }
    //获取成员信息
    std::string getCourseName()const { return this->crs_name; }
    int getCourseCredit()const { return this->crs_credit; }
    bool getCourseIsPF()const { return this->isPF; }
    //在派生类中获取基类的信息
    std::string getTeacherName()const { return Teacher::getTeacherName(); }
    std::string getTeacherID()const { return Teacher::getTeacherID(); }
    int getTeacherCollege()const { return Teacher::getTeacherCollege(); }
};


//记录学生+课程的成绩类
class StudentCourse :public Student, public Course
{
private:
    std::string crs;//课程名称
    int score;//百分制成绩
    double gpa;//课程绩点
    int grade;//课程等级
public:
    StudentCourse(std::string crs = "N/A", int score = 0, double gpa = 0.0, int grade = 0)
    {
        this->crs = crs;
        this->score = score;
        if (Course::getCourseIsPF() == true) {
            if (60 <= score && score <= 100) {
                this->gpa = 4.0;//4.0 代替 P
                this->grade = 11;//"P"
            }
            else {
                this->gpa = 0.0;//0 代替 F
                this->grade = 12;//"F"
            }
        }
        else {
            if (score == 100) {
                this->gpa = 4.0;
                this->grade = 0;
            }
            else if (95 <= score && score <= 99) {
```

```
            this->gpa = 4.0;
            this->grade = 1;
        }
        else if (90 <= score && score <= 94) {
            this->gpa = 4.0;
            this->grade = 2;
        }
        else if (85 <= score && score <= 89) {
            this->gpa = 3.6;
            this->grade = 3;
        }
        else if (80 <= score && score <= 84) {
            this->gpa = 3.3;
            this->grade = 4;
        }
        else if (77 <= score && score <= 79) {
            this->gpa = 3.0;
            this->grade = 5;
        }
        else if (73 <= score && score <= 76) {
            this->gpa = 2.6;
            this->grade = 6;
        }
        else if (70 <= score && score <= 72) {
            this->gpa = 2.3;
            this->grade = 7;
        }
        else if (67 <= score && score <= 69) {
            this->gpa = 2.0;
            this->grade = 8;
        }
        else if (63 <= score && score <= 66) {
            this->gpa = 1.6;
            this->grade = 9;
        }
        else if (60 <= score && score <= 62) {
            this->gpa = 1.3;
            this->grade = 10;
        }
        else {
            this->gpa = 0.0;
            this->grade = 12;
        }
    }
```

```
    }
    StudentCourse(const StudentCourse& other) :Student(other), Teacher(other), Course(other)
    {
        this->crs = other.crs;
        this->score = other.score;
        this->gpa = other.gpa;
        this->grade = other.grade;
    }
    virtual ~StudentCourse() {};
    void displayInfo()const override
    {
        std::cout << "*    Course: " << crs << std::endl;
        std::cout << "*    Score: " << score << "\tCredit: " << Course::getCourseCredit() << std::endl;
        std::cout << "*    GPA: " << gpa << "\tGrade: " << Grade[grade] << std::endl;
        return;
    }
    void setStudentCourseInfo(const std::string name1 = "N/A", const std::string id1 = "N/A", const int class1 = 0, const std::string name2 = "N/A", const std::string id2 = "N/A", const int college = 0, const std::string name3 = "N/A", const int credit = 0, const bool isPF = false, const int score = 0)
    {
        Student::setStudentInfo(name1, id1, class1);
        Teacher::setTeacherInfo(name2, id2, college);
        Course::setCourseInfo(name2, id2, college, name3, credit, isPF);
        this->crs = name3;
        this->score = score;
        if (Course::getCourseIsPF() == true) {
            if (60 <= score && score <= 100) {
                this->gpa = 4.0;//4.0 代替 P
                this->grade = 11;//"P"
            }
            else {
                this->gpa = 0.0;//0 代替 F
                this->grade = 12;//"F"
            }
        }
        else {
            if (score == 100) {
                this->gpa = 4.0;
                this->grade = 0;
            }
            else if (95 <= score && score <= 99) {
```

```
            this->gpa = 4.0;
            this->grade = 1;
        }
        else if (90 <= score && score <= 94) {
            this->gpa = 4.0;
            this->grade = 2;
        }
        else if (85 <= score && score <= 89) {
            this->gpa = 3.6;
            this->grade = 3;
        }
        else if (80 <= score && score <= 84) {
            this->gpa = 3.3;
            this->grade = 4;
        }
        else if (77 <= score && score <= 79) {
            this->gpa = 3.0;
            this->grade = 5;
        }
        else if (73 <= score && score <= 76) {
            this->gpa = 2.6;
            this->grade = 6;
        }
        else if (70 <= score && score <= 72) {
            this->gpa = 2.3;
            this->grade = 7;
        }
        else if (67 <= score && score <= 69) {
            this->gpa = 2.0;
            this->grade = 8;
        }
        else if (63 <= score && score <= 66) {
            this->gpa = 1.6;
            this->grade = 9;
        }
        else if (60 <= score && score <= 62) {
            this->gpa = 1.3;
            this->grade = 10;
        }
        else {
            this->gpa = 0.0;
            this->grade = 12;
        }
    }
```

```
}
void setStudentCourseName(const std::string new_name = "N/A")
{
    this->crs = new_name;
}
void setStudentCourseScore(const int new_score = 0)
{
    this->score = new_score;
    if (Course::getCourseIsPF() == true) {
        if (60 <= score && score <= 100) {
            this->gpa = 4.0;//4.0 代替 P
            this->grade = 11;//"P"
        }
        else {
            this->gpa = 0.0;//0 代替 F
            this->grade = 12;//"F"
        }
    }
    else {
        if (score == 100) {
            this->gpa = 4.0;
            this->grade = 0;
        }
        else if (95 <= score && score <= 99) {
            this->gpa = 4.0;
            this->grade = 1;
        }
        else if (90 <= score && score <= 94) {
            this->gpa = 4.0;
            this->grade = 2;
        }
        else if (85 <= score && score <= 89) {
            this->gpa = 3.6;
            this->grade = 3;
        }
        else if (80 <= score && score <= 84) {
            this->gpa = 3.3;
            this->grade = 4;
        }
        else if (77 <= score && score <= 79) {
            this->gpa = 3.0;
            this->grade = 5;
        }
        else if (73 <= score && score <= 76) {
```

```
                this->gpa = 2.6;
                this->grade = 6;
            }
            else if (70 <= score && score <= 72) {
                this->gpa = 2.3;
                this->grade = 7;
            }
            else if (67 <= score && score <= 69) {
                this->gpa = 2.0;
                this->grade = 8;
            }
            else if (63 <= score && score <= 66) {
                this->gpa = 1.6;
                this->grade = 9;
            }
            else if (60 <= score && score <= 62) {
                this->gpa = 1.3;
                this->grade = 10;
            }
            else {
                this->gpa = 0.0;
                this->grade = 12;
            }
        }
    }
    std::string getStudentCourseName()const { return this->crs; }
    int getStudentCourseScore()const { return this->score; }
    double getStudentCourseGPA()const { return this->gpa; }
    int getStudentCourseGrade()const { return this->grade; }
    //在派生类中修改基类的信息
    void setStudentName(const std::string new_name = "N/A")
{ Student::setStudentName(new_name); }
    void setStudentID(const std::string new_id = "N/A") { Student::setStudentID(new_id); }
    void setStudentClass(const int new_class = 0) { Student::setStudentClass(new_class); }
    void setTeacherName(const std::string new_name = "N/A")
{ Teacher::setTeacherName(new_name); }
    void setTeacherID(const std::string new_id = "N/A") { Teacher::setTeacherID(new_id); }
    void setTeacherCollege(const int new_college = 0)
{ Teacher::setTeacherCollege(new_college); }
    void setCourseName(const std::string new_name = "N/A")
{ Course::setCourseName(new_name); }
    void setCourseCredit(const int new_credit = 0) { Course::setCourseCredit(new_credit); }
    void setCourseIsPF(const bool new_isPF = false) { Course::setCourseIsPF(new_isPF); }
    //在派生类中获取基类的信息
```

```
    std::string getStudentName()const { return Student::getStudentName(); }
    std::string getStudentID()const { return Student::getStudentID(); }
    int getStudentClass()const { return Student::getStudentClass(); }
    std::string getTeacherName()const { return Teacher::getTeacherName(); }
    std::string getTeacherID()const { return Teacher::getTeacherID(); }
    int getTeacherCollege()const { return Teacher::getTeacherCollege(); }
    std::string getCourseName()const { return Course::getCourseName(); }
    int getCourseCredit()const { return Course::getCourseCredit(); }
    bool getCourseIsPF()const { return Course::getCourseIsPF(); }
};


//结构体，记录单个学生的信息、多门课程、均绩、排名
struct StudentResults
{
    Student stu;
    std::vector<StudentCourse> stu_crs;
    double GPA = 0.0;
    int ranking = 1;
    //标记学生是否 PF 了一门课，若没有则自动初始为-1，若有则赋值为该课程在 stu_crs
中的下标
    int PF_flag = -1;
};

//结构体，记录班级人数和均绩
struct ClassInfo
{
    int class_num;//班号
    int stu_amount;//学生数量
    double aver_GPA;//班级均绩
};


//全局变量
std::vector<StudentResults> res_vec;
std::vector<Teacher> tch_vec;
std::vector<Course> crs_vec;
std::vector<ClassInfo> class_vec;
int stu_obj = 0;//定位对应学生
int tch_obj = 0;//定位对应教师
int crs_obj = 0;//定位对应课程
int class_obj = 1;//定位对应班级
int class_max = 1;//最大班级序号
```

```
//函数声明
void readFromStudentInfoFile(const std::string& file_name, std::vector<StudentResults>& res_vec);
void readFromTeacherInfoFile(const std::string& file_name, std::vector<Teacher>& tch_vec, std::vector<Course>& crs_vec);
void writeIntoStudentInfoFile(const std::string& file_name, const std::vector<StudentResults>& res_vec);
void wrtieIntoTeacherInfoFile(const std::string& file_name, const std::vector<Course>& crs_vec);
void initialiseSystem(std::vector<StudentResults>& res_vec, std::vector<Teacher>& tch_vec, std::vector<Course>& crs_vec, std::vector<ClassInfo>& class_vec);
void shutDownSystem(const std::vector<StudentResults>& res_vec, const std::vector<Teacher>& tch_vec, const std::vector<Course>& crs_vec);
void showMainMenu();//显示主页面菜单
int returnToMainMenu();//返回主菜单

int showStudentLoginMenu(const std::vector<StudentResults>& res_vec, int& stu_obj);//显示学生登陆界面
void showStudentMenu(const std::vector<StudentResults>& res_vec, const int& stu_obj);//显示学生菜单界面
void studentMenuOperate(std::vector<StudentResults>& res_vec, const int& stu_obj);//学生进行操作
void studentInquireSingleResult(const std::vector<StudentResults>& res_vec, const int& stu_obj);//查询单科成绩
void studentInquireAllResults(std::vector<StudentResults>& res_vec, const int& stu_obj);//查询所有成绩
void studentSelectOneToPF(std::vector<StudentResults>& res_vec, const int& stu_obj);//选择一门 PF
void studentInquireClassRanking(const std::vector<StudentResults>& res_vec, const int& stu_obj);//查询班级排名
void studentCheckInfo(const std::vector<StudentResults>& res_vec, const int& stu_obj);//查看个人信息
void updateStudentResults(std::vector<StudentResults>& res_vec, const int& stu_obj);//更新学生成绩
void updateClassInfo(std::vector<StudentResults>& res_vec, std::vector<ClassInfo>& class_vec, const int& class_num);
void updateAllClassInfo(std::vector<StudentResults>& res_vec, std::vector<ClassInfo>& class_vec);

int showTeacherLoginMenu(const std::vector<Teacher>& tch_vec, int& tch_obj);//显示教师登陆界面
void showTeacherMenu(const std::vector<Teacher>& tch_vec, const int& tch_obj);//显示教
```

师菜单界面
void teacherMenuOperate(std::vector<StudentResults>& res_vec, std::vector<Teacher>& tch_vec, std::vector<Course>& crs_vec, std::vector<ClassInfo>& class_vec, const int& tch_obj);//教师进行操作
void teacherEnterResults(std::vector<StudentResults>& res_vec, std::vector<Teacher>& tch_vec, std::vector<Course>& crs_vec, std::vector<ClassInfo>& class_vec, const int& tch_obj);
void teacherModifyResults(std::vector<StudentResults>& res_vec, std::vector<Teacher>& tch_vec, std::vector<Course>& crs_vec, std::vector<ClassInfo>& class_vec, const int& tch_obj);
void teacherInquireCourseResults(const std::vector<StudentResults>& res_vec, const std::vector<Teacher>& tch_vec, const std::vector<Course>& crs_vec, const int& tch_obj);
void teacherInquireStudentResults(const std::vector<StudentResults>& res_vec, const std::vector<Teacher>& tch_vec, const std::vector<Course>& crs_vec, const int& tch_obj);
void teacherInquireClassResults(const std::vector<StudentResults>& res_vec, const std::vector<Teacher>& tch_vec, const std::vector<Course>& crs_vec, const std::vector<ClassInfo>& class_vec, const int& tch_obj);
void teacherCheckInfo(const std::vector<Teacher>& tch_vec, const std::vector<Course>& crs_vec, const int& tch_obj);

int isStudentExist(const std::vector<StudentResults>& res_vec, const std::string& name, const std::string& id);
bool isStudentNameExist(const std::vector<StudentResults>& res_vec, const std::string& name);
int isStudentIDExist(const std::vector<StudentResults>& res_vec, const std::string& id);
int isTeacherExist(const std::vector<Teacher>& tch_vec, const std::string& name, const std::string& id);
int isTeacherNameExist(const std::vector<Teacher>& tch_vec, const std::string& name);
int isTeacherIDExist(const std::vector<Teacher>& tch_vec, const std::string& id);
int isCourseExist(const std::vector<Course>& crs_vec, const std::string& name);
int isStudentSelectCourse(const std::vector<StudentResults>& res_vec, const int& stu_obj, const std::string& name);

//运算符重载
bool operator>(const StudentResults& lhs, const StudentResults& rhs);
bool operator>=(const StudentResults& lhs, const StudentResults& rhs);
bool operator<(const StudentResults& lhs, const StudentResults& rhs);
bool operator<=(const StudentResults& lhs, const StudentResults& rhs);
bool operator==(const StudentResults& lhs, const StudentResults& rhs);
bool operator!=(const StudentResults& lhs, const StudentResults& rhs);

bool operator>(const StudentCourse& lhs, const StudentCourse& rhs);
bool operator>=(const StudentCourse& lhs, const StudentCourse& rhs);
bool operator<(const StudentCourse& lhs, const StudentCourse& rhs);

```
bool operator<=(const StudentCourse& lhs, const StudentCourse& rhs);
bool operator==(const StudentCourse& lhs, const StudentCourse& rhs);
bool operator!=(const StudentCourse& lhs, const StudentCourse& rhs);

bool compareStudentCourseByScore(const StudentCourse& lhs, const StudentCourse& rhs);
bool compareStudentByGPA(const StudentResults& lhs, const StudentResults& rhs);
//函数声明结束



//函数定义
//
//
//从文件中读取学生信息
void readFromStudentInfoFile(const std::string& file_name, std::vector<StudentResults>& res_vec)
{
    std::ifstream stu_file(file_name, std::ios_base::in);
    std::vector<StudentCourse> sc_vec;
    std::vector<Student> stu_vec;
    std::string line;
    res_vec.clear();
    class_vec.clear();

    if (!stu_file.is_open()) {
        std::cerr << "The initialising process ran into a problem and the system needs to restart" << std::endl;
        std::cerr << "Details: Failed to open the file " << file_name << std::endl;
        exit(1);
    }
    while (std::getline(stu_file, line)) {
        std::stringstream ss(line);
        std::string stu_name, stu_id, stu_class, tch_name, tch_id, tch_college;
        std::string crs_name, crs_credit, isPF, score;
        if (std::getline(ss, stu_name, ',') &&
            std::getline(ss, stu_id, ',') &&
            std::getline(ss, stu_class, ',') &&
            std::getline(ss, tch_name, ',') &&
            std::getline(ss, tch_id, ',') &&
            std::getline(ss, tch_college, ',') &&
            std::getline(ss, crs_name, ',') &&
            std::getline(ss, crs_credit, ',') &&
            std::getline(ss, isPF, ',') &&
            std::getline(ss, score, ','))
```

```
        {
            //字符串类型需要转换为整数、浮点数和布尔值
            int stu_class0 = std::stoi(stu_class);
            int tch_college0 = std::stoi(tch_college);
            int crs_credit0 = std::stoi(crs_credit);
            bool isPF0 = (isPF == "1");
            int score0 = std::stoi(score);

            StudentCourse sc;
            sc.setStudentCourseInfo(stu_name, stu_id, stu_class0, tch_name, tch_id, tch_college0, crs_name, crs_credit0, isPF0, score0);

            int flag = 0;
            for (const auto& res : res_vec) {
                flag++;
                if (res.stu.getStudentName() == stu_name && res.stu.getStudentID() == stu_id) {
                    break;
                }
            }
            flag--;//flag 定位出该学生在 res_vec 中的下标，如果没有新建
            if (flag < (res_vec.size() - 1)) {
                res_vec[flag].stu_crs.push_back(sc);
            }
            else {
                StudentResults new_res;
                new_res.stu.setStudentInfo(stu_name, stu_id, stu_class0);
                new_res.ranking = 1;
                new_res.PF_flag = -1;
                new_res.stu_crs.push_back(sc);
                res_vec.push_back(new_res);
            }
        }
    }
    for (int i = 0; i < res_vec.size(); i++) {
        updateStudentResults(res_vec, i);
    }
    updateAllClassInfo(res_vec, class_vec);
    //创建班级
    for (const auto& res : res_vec) {
        if (res.stu.getStudentClass() > class_max) {
            class_max = res.stu.getStudentClass();
        }
    }
```

```
    for (int i = 0; i < class_max; i++) {
        ClassInfo ci;
        ci.class_num = i + 1;
        int amount = 0;
        double sum_gpa = 0;
        for (const auto& res : res_vec) {
            if (res.stu.getStudentClass() == i + 1) {
                amount++;
                sum_gpa += res.GPA;
            }
        }
        ci.stu_amount = amount;
        ci.aver_GPA = sum_gpa / amount;
        class_vec.push_back(ci);
    }
    for (int i = 0; i < res_vec.size(); i++) {
        updateStudentResults(res_vec, i);
    }
    updateAllClassInfo(res_vec, class_vec);
    stu_file.close();
    return;
}


void readFromTeacherInfoFile(const std::string& file_name, std::vector<Teacher>& tch_vec,
std::vector<Course>& crs_vec)
{
    std::ifstream tch_file(file_name, std::ios_base::in);
    std::string line;
    tch_vec.clear();
    crs_vec.clear();
    if (!tch_file.is_open()) {
        std::cerr << "The initialising process ran into a problem and the system needs to
restart" << std::endl;
        std::cerr << "Details: Failed to open the file: " << file_name << std::endl;
        exit(1);
    }
    while (std::getline(tch_file, line))
    {
        std::stringstream ss(line);
        std::string tch_name, tch_id, tch_college, crs_name, crs_credit, isPF;
        if (std::getline(ss, tch_name, ',') &&
            std::getline(ss, tch_id, ',') &&
            std::getline(ss, tch_college, ',') &&
```

```
            std::getline(ss, crs_name, ',') &&
            std::getline(ss, crs_credit, ',') &&
            std::getline(ss, isPF))
        {
            //字符串类型需要转换为整数、浮点数和布尔值
            int tch_college0 = std::stoi(tch_college);
            int crs_credit0 = std::stoi(crs_credit);
            bool isPF0 = (isPF == "1");

            tch_vec.emplace_back(tch_name, tch_id, tch_college0);
            Course crs;
            crs.setCourseInfo(tch_name, tch_id, tch_college0, crs_name, crs_credit0, isPF0);
            crs_vec.push_back(crs);
        }
    }
    tch_file.close();
    return;
}


void writeIntoStudentInfoFile(const std::string& file_name, const
std::vector<StudentResults>& res_vec)
{
    std::ofstream stu_file(file_name, std::ios_base::out | std::ios_base::trunc);
    if (!stu_file.is_open()) {
        std::cerr << "The shutdown process ran into a problem" << std::endl;
        std::cerr << "Failed to open the file: " << file_name << std::endl;
        exit(1);
    }
    for (const auto& res : res_vec) {
        for (const auto& sc : res.stu_crs) {
            std::ostringstream line;
            line << sc.getStudentName() << ',' << sc.getStudentID() << ',' <<
std::to_string(sc.getStudentClass()) << ',';
            line << sc.getTeacherName() << ',' << sc.getTeacherID() << ',' <<
std::to_string(sc.getTeacherCollege()) << ',';
            line << sc.getCourseName() << ',' << std::to_string(sc.getCourseCredit()) << ','
<< (sc.getCourseIsPF() ? "1" : "0") << ',';
            line << std::to_string(sc.getStudentCourseScore()) << std::endl;
            stu_file << line.str();
        }
    }
    stu_file.close();
    return;
```

```
}


void wrtieIntoTeacherInfoFile(const std::string& file_name, const std::vector<Course>&
crs_vec)
{
    std::ofstream tch_file(file_name, std::ios_base::out | std::ios_base::trunc);
    if (!tch_file.is_open()) {
        std::cerr << "The shutdown process ran into a problem" << std::endl;
        std::cerr << "Failed to open the file: " << file_name << std::endl;
        exit(1);
    }
    for (const auto& crs : crs_vec) {
        std::ostringstream line;
        line << crs.getTeacherName() << ',' << crs.getTeacherID() << ',' <<
std::to_string(crs.getTeacherCollege()) << ',';
        line << crs.getCourseName() << ',' << std::to_string(crs.getCourseCredit()) << ',' <<
(crs.getCourseIsPF() ? "1" : "0") << std::endl;
        tch_file << line.str();
    }
    tch_file.close();
    return;
}


//初始化函数
void initialiseSystem(std::vector<StudentResults>& res_vec, std::vector<Teacher>& tch_vec,
std::vector<Course>& crs_vec, std::vector<ClassInfo>& class_vec)
{
    readFromStudentInfoFile(STUDENT_FILE_PATH, res_vec);
    readFromTeacherInfoFile(TEACHER_FILE_PATH, tch_vec, crs_vec);
    for (int i = 0; i < res_vec.size(); i++) {
        updateStudentResults(res_vec, i);
    }
    updateAllClassInfo(res_vec, class_vec);
    std::cout << "******              System initializing...              ******" << std::endl;
    Sleep(1314);//休眠 1.314 秒
    std::cout    <<    "**************************************************"    <<
std::endl;
    std::cout << "*                                                        *" <<
std::endl;
    std::cout << "*                                                        *" <<
std::endl;
    std::cout << "*          The system initialization is complete!        *" << std::endl;
```

```
    std::cout << "*                                                    *" <<
std::endl;
    std::cout << "*                                                    *" <<
std::endl;
    std::cout    <<    "******************************************************"    <<
std::endl;
    return;
}

//关机函数
void shutDownSystem(const std::vector<StudentResults>& res_vec, const
std::vector<Teacher>& tch_vec, const std::vector<Course>& crs_vec)
{
    writeIntoStudentInfoFile(STUDENT_FILE_PATH, res_vec);
    wrtieIntoTeacherInfoFile(TEACHER_FILE_PATH, crs_vec);
    std::cout << "******        System shuting down...        ******" << std::endl;
    Sleep(1314);//休眠 1.314 秒
    std::cout << "******************************************" << std::endl;
    std::cout << "*                                        *" << std::endl;
    std::cout << "*                                        *" << std::endl;
    std::cout << "*        The system has been shut down!        *" << std::endl;
    std::cout << "*                                        *" << std::endl;
    std::cout << "*                                        *" << std::endl;
    std::cout << "******************************************" << std::endl;
    return;
}

//主函数
int main(int argc, const char* argv[])
{
    int select_num;//用户根据菜单界面相应功能输入的数字
    initialiseSystem(res_vec, tch_vec, crs_vec, class_vec);
    system("pause");
    system("cls");
    while (true)
    {
        showMainMenu();
        std::cout << "Input a number: 1 or 2 or 0 : ";
        std::cin >> select_num;
        if (select_num == 1)
        {
            std::cout << "\nYou choose : 1" << std::endl;
            std::cout << "------ You are about to enter the teacher interface ------" <<
std::endl;
```

```
            system("pause");
            int isLogIn = showTeacherLoginMenu(tch_vec, tch_obj);
            if (isLogIn != -1) {
                teacherMenuOperate(res_vec, tch_vec, crs_vec, class_vec, isLogIn);
            }
        }
        else if (select_num == 2)
        {
            std::cout << "\nYou choose : 2" << std::endl;
            std::cout << "------ You are about to enter the student interface ------" <<
std::endl;
            system("pause");
            int isLogIn = showStudentLoginMenu(res_vec, stu_obj);
            if (isLogIn != -1) {
                studentMenuOperate(res_vec, isLogIn);
            }
        }
        else if (select_num == 0) {
            std::cout << std::endl;
            std::cout << "****** Are you sure to exit the Student Performance Management
System? ******" << std::endl;
            std::cout << "******                                1---Yes        0---No, keep using
******" << std::endl;
            int isExit;
            std::cout << "Input number 1 or 0 : ";
            while (true)
            {
                std::cin >> isExit;
                if (isExit == 1)
                {
                    std::cout << std::endl;
                    std::cout << "****** You have successfully exited the Student
Performance Management System! ******" << std::endl;
                    std::cout << "******                                Welcome to use
again!                                ******" << std::endl;
                    system("pause");
                    system("cls");
                    shutDownSystem(res_vec, tch_vec, crs_vec);
                    stu_obj = tch_obj = crs_obj = class_obj = 0;
                    system("pause");
                    return 0;
                    break;
                }
                else if (isExit == 0)
```

```
                {
                    std::cout << "****** You canceled to exit the Student Performance Management System! ******" << std::endl;
                    std::cout << "******                                    Please continue to use... ******" << std::endl;
                    system("pause");
                    system("cls");
                    break;
                }
                else std::cout << "\nInvalid input! Please input number 1 or 0 : ";
            }
        }
        else {
            std::cout << "\nInvalid input! Please input number 1 or 2 or 0 : " << std::endl;
            system("pause");
            system("cls");
        }
    }
    system("pause");
    return 0;
}


//该学生是否已存在，不存在返回-1，存在返回在 res_vec 中的下标
int isStudentExist(const std::vector<StudentResults>& res_vec, const std::string& name, const std::string& id)
{
    for (int i = 0; i < res_vec.size(); i++) {
        if (res_vec[i].stu.getStudentName() == name && res_vec[i].stu.getStudentID() == id)
{
            return i;
        }
    }
    return -1;
}

//该学生姓名是否存在(学生可能存在同名)
bool isStudentNameExist(const std::vector<StudentResults>& res_vec, const std::string& name)
{
    for (int i = 0; i < res_vec.size(); i++) {
        if (res_vec[i].stu.getStudentName() == name)
            return true;
    }
```

```
    return false;
}

//该学号是否存在，不存在返回-1，存在返回在 res_vec 中的下标
int isStudentIDExist(const std::vector<StudentResults>& res_vec, const std::string& id)
{
    for (int i = 0; i < res_vec.size(); i++) {
        if (res_vec[i].stu.getStudentID() == id)
            return i;
    }
    return -1;
}

//该老师是否存在，不存在返回-1，存在返回在 tch_vec 中的下标
int isTeacherExist(const std::vector<Teacher>& tch_vec, const std::string& name, const std::string& id)
{
    for (int i = 0; i < tch_vec.size(); i++) {
        if (tch_vec[i].getTeacherName() == name && tch_vec[i].getTeacherID() == id)
            return i;
    }
    return -1;
}

//该老师姓名是否存在，不存在返回-1，存在返回在 tch_vec 中的下标，默认老师不存在同名
int isTeacherNameExist(const std::vector<Teacher>& tch_vec, const std::string& name)
{
    for (int i = 0; i < tch_vec.size(); i++) {
        if (tch_vec[i].getTeacherName() == name)
            return i;
    }
    return -1;
}

//该老师工号是否存在，不存在返回-1，存在返回在 tch_vec 中的下标
int isTeacherIDExist(const std::vector<Teacher>& tch_vec, const std::string& id)
{
    for (int i = 0; i < tch_vec.size(); i++) {
        if (tch_vec[i].getTeacherID() == id)
            return i;
    }
    return -1;
}
```

```
//该课程是否存在，未选返回-1，选了返回在 crs_vec 中的下标
int isCourseExist(const std::vector<Course>& crs_vec, const std::string& name)
{
    for (int i = 0; i < crs_vec.size(); i++) {
        if (crs_vec[i].getCourseName() == name)
            return i;
    }
    return -1;
}

//学生是否选了这门课,未选返回-1，选了返回在 stu_crs 中的下标
int isStudentSelectCourse(const std::vector<StudentResults>& res_vec, const int& stu_obj,
const std::string& name)
{
    for (int i = 0; i < res_vec[stu_obj].stu_crs.size(); i++) {
        if (res_vec[stu_obj].stu_crs[i].getCourseName() == name)
            return i;
    }
    return -1;
}


//关于 StudentResults 的运算符重载
bool operator<(const StudentResults& lhs, const StudentResults& rhs) {
    return lhs.GPA < rhs.GPA;
}

bool operator>(const StudentResults& lhs, const StudentResults& rhs) {
    return lhs.GPA > rhs.GPA;
}

bool operator<=(const StudentResults& lhs, const StudentResults& rhs) {
    return lhs.GPA <= rhs.GPA;
}

bool operator>=(const StudentResults& lhs, const StudentResults& rhs) {
    return lhs.GPA >= rhs.GPA;
}

bool operator==(const StudentResults& lhs, const StudentResults& rhs) {
    return lhs.GPA == rhs.GPA;
}
```

```
bool operator!=(const StudentResults& lhs, const StudentResults& rhs) {
    return lhs.GPA != rhs.GPA;
}


//关于 StudentCourse 的运算符重载
bool operator<(const StudentCourse& lhs, const StudentCourse& rhs) {
    return lhs.getStudentCourseScore() < rhs.getStudentCourseScore();
}

bool operator>(const StudentCourse& lhs, const StudentCourse& rhs) {
    return lhs.getStudentCourseScore() > rhs.getStudentCourseScore();
}

bool operator<=(const StudentCourse& lhs, const StudentCourse& rhs) {
    return lhs.getStudentCourseScore() <= rhs.getStudentCourseScore();
}

bool operator>=(const StudentCourse& lhs, const StudentCourse& rhs) {
    return lhs.getStudentCourseScore() >= rhs.getStudentCourseScore();
}

bool operator==(const StudentCourse& lhs, const StudentCourse& rhs) {
    return lhs.getStudentCourseScore() == rhs.getStudentCourseScore();
}

bool operator!=(const StudentCourse& lhs, const StudentCourse& rhs) {
    return lhs.getStudentCourseScore() != rhs.getStudentCourseScore();
}

//退出当前页面,返回主菜单
int returnToMainMenu()
{
    std::cout << "\n****** Are you sure to exit the current page and return to the main menu?
******" << std::endl;
    std::cout << "******                                1---Yes, return          Else---No, keep using
******" << std::endl;
    int exit;
    std::cout << "Please input: ";
    std::cin >> exit;
    if (exit == 1) {
        std::cout << "------ You are about to return to the main interface... ------" << std::endl;
        return 1;
    }
```

```
    else {
        std::cout << "****** You canceled to return to the main interface! ******" <<
std::endl;
        std::cout << "******                Please continue to use...                ******" <<
std::endl;
        return 0;
    }
}

//显示主菜单
void showMainMenu()
{
    std::cout    <<    "**************************************************"    <<
std::endl;
    std::cout << "*                                                        *" <<
std::endl;
    std::cout << "*                                                        *" <<
std::endl;
    std::cout << "*                    Welcome to the                      *" << std::endl;
    std::cout << "*          Student Performance Management System         *" << std::endl;
    std::cout << "*                                                        *" <<
std::endl;
    std::cout << "*                                                        *" <<
std::endl;
    std::cout    <<    "**************************************************"    <<
std::endl;
    std::cout << std::endl;
    std::cout << "To select your identity, press 1 or 2. To exit the system, press 0." <<
std::endl;
    std::cout << "1---I'm a teacher       2---I'm a student       0---exit the system" << std::endl;
    return;
}


//
//
//学生部分
//
//
//显示学生登录介面
int showStudentLoginMenu(const std::vector<StudentResults>& res_vec, int& stu_obj)
{
    system("cls");
    std::string name, id;
```

```
    int flag = 0;
    std::cout << "------ Student Login Interface ------\n" << std::endl;
    std::cout << "Please input your name and student ID" << std::endl;
    std::cout << "Attention: only when they are correct and match can you successfully login\n" << std::endl;
    while (true)
    {
        std::cout << "Name: ";
        std::cin >> name;
        std::cout << "Student ID: ";
        std::cin >> id;
        if (isStudentNameExist(res_vec, name) == true && isStudentIDExist(res_vec, id) != -1 && isStudentIDExist(res_vec, id) == isStudentExist(res_vec, name, id))
        {
            stu_obj = isStudentExist(res_vec, name, id);
            std::cout << "\nSuccessful login! Welcome, " << res_vec[stu_obj].stu.getStudentName() << "!" << std::endl;
            std::cout << "------ You are about to enter student interface... ------\n" << std::endl;
            system("pause");
            return stu_obj;
            break;
        }
        else std::cout << "Login failed! Please check your name or student ID, or whether the two match\n" << std::endl;
    }
    system("pause");
    return -1;
}

//显示特定学生菜单界面
void showStudentMenu(const std::vector<StudentResults>& res_vec, const int& stu_obj)
{
    system("cls");
    std::cout << "Dear " << res_vec[stu_obj].stu.getStudentName() << ":" << std::endl;
    std::cout << "************************************************************" << std::endl;
    std::cout << "*                                                          *" << std::endl;
    std::cout << "*                                                          *" << std::endl;
    std::cout << "*          1. Query a course grade                         *" << std::endl;
    std::cout << "*                                                          *"
```

```
<< std::endl;
    std::cout << "*          2. Query all courses grades                    *" <<
std::endl;
    std::cout << "*                                                         *"
<< std::endl;
    std::cout << "*          3. Select a course to record it as P/F         *" <<
std::endl;
    std::cout << "*                                                         *"
<< std::endl;
    std::cout << "*          4. Query class average grades and your ranking *" <<
std::endl;
    std::cout << "*                                                         *"
<< std::endl;
    std::cout << "*          5. Query your personal information             *" <<
std::endl;
    std::cout << "*                                                         *"
<< std::endl;
    std::cout << "*          0. Exit the student interface                  *" <<
std::endl;
    std::cout << "*                                                         *"
<< std::endl;
    std::cout << "*                                                         *"
<< std::endl;
    std::cout << "***********************************************************"
<< std::endl;
    std::cout << "\nPlease select the operation you want (press the number corresponding to
the specific function):" << std::endl;
    return;
}

//显示学生操作界面
void studentMenuOperate(std::vector<StudentResults>& res_vec, const int& stu_obj)//在学
生界面进行的操作
{
    int select;
    int ret;
    while (true)
    {
        showStudentMenu(res_vec, stu_obj);
        std::cin >> select;
        switch (select)
        {
        case 1:
            studentInquireSingleResult(res_vec, stu_obj);
```

```
            system("pause");
            break;
        case 2:
            studentInquireAllResults(res_vec, stu_obj);
            system("pause");
            break;
        case 3:
            studentSelectOneToPF(res_vec, stu_obj);
            system("pause");
            break;
        case 4:
            studentInquireClassRanking(res_vec, stu_obj);
            system("pause");
            break;
        case 5:
            studentCheckInfo(res_vec, stu_obj);
            system("pause");
            break;
        case 0:
            ret = returnToMainMenu();
            system("pause");
            system("cls");
            if (ret == 1) {
                return;
                break;
            }
            else break;
        default:
            std::cout << "Invalid input! Please input number again (0--5)" << std::endl;
            system("pause");
            break;
        }
    }
    return;
}

//查询单科成绩
void studentInquireSingleResult(const std::vector<StudentResults>& res_vec, const int& stu_obj)
{
    system("cls");
    std::string crs_name;//课程名称
    int stu_crs_obj = 0;//定位课程在学生所选的位置
    std::cout << "Dear " << res_vec[stu_obj].stu.getStudentName() << "," << std::endl;
```

```
    std::cout << "All courses you selected are as follows:\n" << std::endl;
    std::cout << "****************************************" << std::endl;
    int num = 0;
    for (const auto& s_c : res_vec[stu_obj].stu_crs) {
        std::cout << "*    " << num + 1 << ".    " << s_c.getCourseName() << std::endl;
        num++;
    }
    std::cout << "****************************************" << std::endl;
    while (true)
    {
        std::cout << "Please input the number before the course you want to query: ";
        std::cin >> num;
        if (1 <= num && num <= res_vec[stu_obj].stu_crs.size()) {
            const auto& s_c = res_vec[stu_obj].stu_crs[num - 1];
            std::cout << "Search successful!\n" << std::endl;
            std::cout << "Your grade:" << std::endl;
            std::cout << "********************" << std::endl;
            std::cout << "*    Course: " << crs_name << std::endl;
            std::cout << "*    Score: " << s_c.getStudentCourseScore() << std::endl;
            std::cout << "*    Grade: " << Grade[s_c.getStudentCourseGrade()] << std::endl;
            std::cout << "*    GPA: " << s_c.getStudentCourseGPA() << std::endl;
            std::cout << "********************\n" << std::endl;
            s_c.Course::displayInfo();

            int exit;
            std::cout << "Press 1 to continue query, otherwise you will exit: ";
            std::cin >> exit;
            if (exit == 1) {
                std::cout << "You have chose to continue query!" << std::endl;
                system("pause");
                std::cout << std::endl;
            }
            else {
                std::cout << "You have chose: " << exit << std::endl;
                std::cout << "------ You are about to exit ------" << std::endl;
                return;
            }
        }
        else {
            std::cout << "Invalid input! Not between 1--" << res_vec[stu_obj].stu_crs.size();
        }
    }
    return;
}
```

```
//查询所有成绩
void studentInquireAllResults(std::vector<StudentResults>& res_vec, const int& stu_obj)
{
    system("cls");
    double sum_gpa = 0;//总学分绩
    double sum_credit = 0;//总学分数
    double sum_credit_in_gpa = 0;//计入 GPA 学分数
    std::cout << "Dear " << res_vec[stu_obj].stu.getStudentName() << "," << std::endl;
    std::cout << "All courses you selected are as follows:\n" << std::endl;
    std::cout << "****************************************" << std::endl;
    for (const auto& s_c : res_vec[stu_obj].stu_crs) {
        s_c.displayInfo();
        std::cout << "*    ---------------------------------" << std::endl;
        sum_credit += (double)s_c.getCourseCredit();
        if (s_c.getCourseIsPF() == false) {
            sum_gpa += s_c.getCourseCredit() * s_c.getStudentCourseGPA();
            sum_credit_in_gpa += (double)s_c.getCourseCredit();
        }
    }
    res_vec[stu_obj].GPA = sum_gpa / sum_credit;
    std::cout << "*    Your GPA: " << sum_gpa / sum_credit << std::endl;
    std::cout << "*    Total credits: " << sum_credit << std::endl;
    std::cout << "*    Total credits count towards GPA: " << sum_credit_in_gpa << std::endl;
    std::cout << "*    Total GPA: " << sum_gpa << std::endl;
    std::cout << "****************************************\n" << std::endl;
    return;
}

//选择一门课程记 PF
void studentSelectOneToPF(std::vector<StudentResults>& res_vec, const int& stu_obj)
{
    system("cls");
    std::cout << "Dear " << res_vec[stu_obj].stu.getStudentName() << ", remember that you
can ONLY record ONE course as P/F" << std::endl;
    if (res_vec[stu_obj].PF_flag != -1) {
        std::cout << "You have already recorded a course as P/F! You choose:" << std::endl;
        std::cout << "1---Modify and record another course as P/F" << std::endl;
        std::cout << "0---Keep the original P/F course selection unchanged" << std::endl;
        int select;
        std::cout << "Please input number 1 or 0 : ";
        while (true)
        {
            std::cin >> select;
```

```
                if (select == 1) {
                    std::cout << "You have canceled P/F to the course: " << res_vec[stu_obj].stu_crs[res_vec[stu_obj].PF_flag].getCourseName() << std::endl;
                    res_vec[stu_obj].stu_crs[res_vec[stu_obj].PF_flag].setCourseIsPF(false);
                    res_vec[stu_obj].PF_flag = -1;
                    updateStudentResults(res_vec, stu_obj);
                    updateClassInfo(res_vec, class_vec, res_vec[stu_obj].stu.getStudentClass());
                    updateAllClassInfo(res_vec, class_vec);
                    system("pause");
                    break;
                }
                else if (select == 0) {
                    std::cout << "You have abandoned the modification! The original P/F choice keeps unchanged.\n" << std::endl;
                    std::cout << "------ You are about to exit and return to student interface ------" << std::endl;
                    return;
                    break;
                }
                else std::cout << "Invalid input! Please input number 1 or 0 : ";
            }
        }
        std::cout << "Now, you can record a course as P/F. Here are the courses you can select." << std::endl;
        std::cout << "Attention: This operation does not affect the courses with their own P/F\n" << std::endl;
        std::cout << "******************************" << std::endl;
        for (int i = 0; i < res_vec[stu_obj].stu_crs.size(); i++) {
            std::cout << i + 1 << ". " << res_vec[stu_obj].stu_crs[i].getCourseName() << std::endl;
        }
        std::cout << "******************************" << std::endl;
        std::cout << "Please select a course (Enter the number corresponding to the course or 0 to exit):" << std::endl;
        int choice;
        while (true)
        {
            std::cin >> choice;
            if (1 <= choice && choice <= res_vec[stu_obj].stu_crs.size()) {
                std::cout << "You select: " << choice << ". " << res_vec[stu_obj].stu_crs[choice - 1].getCourseName() << std::endl;
                int sure_num;
                std::cout << "Are you sure to record it as P/F? (Your grades will update automatically)" << std::endl;
```

```
            std::cout << "1---Yes       0---NO" << std::endl;
            while (true) {
                std::cin >> sure_num;
                if (sure_num == 1) {
                    res_vec[stu_obj].stu_crs[sure_num - 1].setCourseIsPF(true);
                    res_vec[stu_obj].PF_flag = sure_num - 1;
                    updateStudentResults(res_vec, stu_obj);//更新
                    updateClassInfo(res_vec,                                    class_vec,
res_vec[stu_obj].stu.getStudentClass());
                    updateAllClassInfo(res_vec, class_vec);
                    std::cout    <<    "You    have    successfully    record    "    <<
res_vec[stu_obj].stu_crs[choice - 1].getCourseName() << " as P/F!" << std::endl;
                    std::cout << "You get score: " << res_vec[stu_obj].stu_crs[choice -
1].getStudentCourseScore() << " in the course";
                    if (res_vec[stu_obj].stu_crs[choice - 1].getStudentCourseScore() < 60)
                        std::cout << " so the grade will be: F" << std::endl;
                    else std::cout << " so the grade will be: P" << std::endl;
                    return;
                }
                else if (sure_num == 0) {
                    std::cout << "You have abandoned the P/F selection.\n" << std::endl;
                    std::cout << "------ You are about to exit and return to student interface
------" << std::endl;
                    return;
                }
                else std::cout << "Invalid input! Please input number 1 or 0" << std::endl;
            }
        }
        else if (choice == 0) {
            std::cout << "You have abandoned the P/F selection.\n" << std::endl;
            std::cout << "------ You are about to exit and return to student interface ------" <<
std::endl;
            return;
        }
        else {
            std::cout    <<    "Invalid    input!    Please    input    number    1--"    <<
res_vec[stu_obj].stu_crs.size() << std::endl;
        }
    }
    return;
}

//查询班级排名
void studentInquireClassRanking(const std::vector<StudentResults>& res_vec, const int&
```

```
stu_obj)
{
    system("cls");
    double min_gpa = res_vec[stu_obj].GPA, max_gpa = res_vec[stu_obj].GPA;
    for (const auto& res : res_vec) {
        if (res > res_vec[stu_obj])
            max_gpa = res.GPA;
        if (res.GPA < res_vec[stu_obj].GPA)
            min_gpa = res.GPA;
    }
    std::cout << "Dear " << res_vec[stu_obj].stu.getStudentName() << ",\n" << std::endl;
    std::cout << "****************************************" << std::endl;
    std::cout << "*    Your class: " << res_vec[stu_obj].stu.getStudentClass() << std::endl;
    std::cout << "*    Class size: " << class_vec[res_vec[stu_obj].stu.getStudentClass() -
1].stu_amount << std::endl;
    std::cout << "*    Average GPA in your class: " <<
class_vec[res_vec[stu_obj].stu.getStudentClass() - 1].aver_GPA << std::endl;
    std::cout << "*    Maximum GPA in your class: " << max_gpa << std::endl;
    std::cout << "*    Minimum GPA in your class: " << min_gpa << std::endl;
    std::cout << "*    Your GPA: " << res_vec[stu_obj].GPA << std::endl;
    std::cout << "*    Your class ranking: " << res_vec[stu_obj].ranking << std::endl;
    std::cout << "****************************************\n" << std::endl;
    return;
}


//查看个人信息
void studentCheckInfo(const std::vector<StudentResults>& res_vec, const int& stu_obj)
{
    system("cls");
    res_vec[stu_obj].stu.displayInfo();
    return;
}


//更新该学生各科成绩、均绩和班级排名
void updateStudentResults(std::vector<StudentResults>& res_vec, const int& stu_obj)
{
    int class_num = res_vec[stu_obj].stu.getStudentClass();
    double sum_gpa = 0;//总学分绩
    double sum_credit = 0;//总学分
    for (auto& s_c : res_vec[stu_obj].stu_crs) {
        s_c.setStudentCourseScore(s_c.getStudentCourseScore());
        sum_credit += (double)s_c.getCourseCredit();
```

```
            if (s_c.getCourseIsPF() == false) {
                sum_gpa += s_c.getCourseCredit() * s_c.getStudentCourseGPA();
            }
        }
        res_vec[stu_obj].GPA = sum_gpa / sum_credit;//更新均绩

        int ranking = 1;
        for (const auto& res : res_vec) {
            if (class_num == res.stu.getStudentClass() && res_vec[stu_obj] < res)
                ranking++;
        }
        res_vec[stu_obj].ranking = ranking;//更新班级排名
        return;
}


//更新该班级所有学生排名和班级均绩，class_obj 是班级数，不是班级在 class_vec 中的下标
void updateClassInfo(std::vector<StudentResults>& res_vec, std::vector<ClassInfo>& class_vec, const int& class_obj)
{
        double sum_gpa = 0;
        int amount = 0;//班级人数
        int flag = 0;
        for (auto& res : res_vec) {
            if (res.stu.getStudentClass() == class_obj) {
                updateStudentResults(res_vec, flag);
                sum_gpa += res.GPA;
                amount++;
            }
            flag++;
        }
        class_vec[class_obj - 1].aver_GPA = sum_gpa / class_vec[class_obj - 1].stu_amount;
        class_vec[class_obj - 1].stu_amount = amount;
        return;
}


//更新所有班级信息
void updateAllClassInfo(std::vector<StudentResults>& res_vec, std::vector<ClassInfo>& class_vec)
{
        for (int i = 0; i < class_vec.size(); i++) {
            updateClassInfo(res_vec, class_vec, i + 1);
```

```
    }
    return;
}


//比较选同一门课的两个学生的成绩，降序
bool compareStudentCourseByScore(const StudentCourse& lhs, const StudentCourse& rhs)
{
    return lhs.getStudentCourseScore() > rhs.getStudentCourseScore();
}

//比较两个同班的学生的 GPA，用于班内排名，升序
bool compareStudentByGPA(const StudentResults& lhs, const StudentResults& rhs)
{
    return lhs.GPA > rhs.GPA;
}


//
//
//教师部分
//
//
//显示教师登陆界面
int showTeacherLoginMenu(const std::vector<Teacher>& tch_vec, int& tch_obj)
{
    system("cls");
    std::string name, id;
    int flag = 0;
    std::cout << "------ Teacher Login Interface ------\n" << std::endl;
    std::cout << "Please input your name and teacher ID" << std::endl;
    std::cout << "Attention: only when they are correct and match can you successfully login\n" << std::endl;
    while (true)
    {
        std::cout << "Name: ";
        std::cin >> name;
        std::cout << "Teacher ID: ";
        std::cin >> id;
        if (isTeacherIDExist(tch_vec, id) != -1 && isTeacherIDExist(tch_vec, id) == isTeacherNameExist(tch_vec, name) && isTeacherIDExist(tch_vec, id) == isTeacherExist(tch_vec, name, id))
        {
            tch_obj = isTeacherExist(tch_vec, name, id);
```

```
            std::cout << "\nSuccessful login! Welcome, " << tch_vec[tch_obj].getTeacherName() << "!" << std::endl;
            std::cout << "------ You are about to enter teacher interface... ------\n" << std::endl;
            system("pause");
            return tch_obj;
            break;
        }
        else std::cout << "Login failed! Please check your name or teacher ID, or whether the two match\n" << std::endl;
    }
    system("pause");
    return -1;
}

//显示教师界面菜单
void showTeacherMenu(const std::vector<Teacher>& tch_vec, const int& tch_obj)
{
    system("cls");
    std::cout << "Dear " << tch_vec[tch_obj].getTeacherName() << ":" << std::endl;
    std::cout << "**************************************************" << std::endl;
    std::cout << "*                                                *" << std::endl;
    std::cout << "*                                                *" << std::endl;
    std::cout << "*        1. Enter course grades                  *" << std::endl;
    std::cout << "*                                                *" << std::endl;
    std::cout << "*        2. Modify course grades                 *" << std::endl;
    std::cout << "*                                                *" << std::endl;
    std::cout << "*        3. Query your course grades             *" << std::endl;
    std::cout << "*                                                *" << std::endl;
    std::cout << "*        4. Query students grades                *" << std::endl;
    std::cout << "*                                                *" << std::endl;
    std::cout << "*        5. Query class grades                   *" << std::endl;
    std::cout << "*                                                *" << std::endl;
    std::cout << "*        6. Query your personal information      *" << std::endl;
    std::cout << "*                                                *" << std::endl;
```

```
std::endl;
    std::cout << "*          0. Exit the teacher interface                    *" << std::endl;
    std::cout << "*                                                           *" <<
std::endl;
    std::cout << "*                                                           *" <<
std::endl;
    std::cout   <<   "***********************************************"   <<
std::endl;
    std::cout << "\nPlease select the operation you want (press the number corresponding to
the specific function):" << std::endl;
    return;
}

//教师操作界面
void teacherMenuOperate(std::vector<StudentResults>& res_vec, std::vector<Teacher>&
tch_vec, std::vector<Course>& crs_vec, std::vector<ClassInfo>& class_vec, const int&
tch_obj)//教师界面进行的操作
{
    int select;
    int ret;
    while (true)
    {
        showTeacherMenu(tch_vec, tch_obj);
        std::cin >> select;
        switch (select)
        {
        case 1:
            teacherEnterResults(res_vec, tch_vec, crs_vec, class_vec, tch_obj);
            system("pause");
            break;
        case 2:
            teacherModifyResults(res_vec, tch_vec, crs_vec, class_vec, tch_obj);
            system("pause");
            break;
        case 3:
            teacherInquireCourseResults(res_vec, tch_vec, crs_vec, tch_obj);
            system("pause");
            break;
        case 4:
            teacherInquireStudentResults(res_vec, tch_vec, crs_vec, tch_obj);
            system("pause");
            break;
        case 5:
            teacherInquireClassResults(res_vec, tch_vec, crs_vec, class_vec, tch_obj);
```

```
            system("pause");
            break;
        case 6:
            teacherCheckInfo(tch_vec, crs_vec, tch_obj);
            system("pause");
            break;
        case 0:
            ret = returnToMainMenu();
            system("pause");
            system("cls");
            if (ret == 1) {
                return;
                break;
            }
            else break;
        default:
            std::cout << "Invalid input! Please input number again (0--6)" << std::endl;
            system("pause");
            break;
        }
    }
    return;
}

//录入课程成绩
void teacherEnterResults(std::vector<StudentResults>& res_vec, std::vector<Teacher>& tch_vec, std::vector<Course>& crs_vec, std::vector<ClassInfo>& class_vec, const int& tch_obj)
{
    system("cls");
    std::cout << "Dear " << tch_vec[tch_obj].getTeacherName() << ", you teach the course: " << crs_vec[tch_obj].getCourseName() << std::endl;
    int choice;
    std::cout << "You can enter grades in TWO ways:" << std::endl;
    std::cout << "1---Enter grades of student that select your course" << std::endl;
    std::cout << "2---Add new students to select your course and enter their grades" << std::endl;
    while (true)
    {
        std::cout << "Please input your choice (1 or 2): ";
        std::cin >> choice;
        if (choice == 1) {
            std::cout << "Here are students that select your course" << std::endl;
            int num = 0;
```

```
            for (auto& res : res_vec) {
                for (auto& s_c : res.stu_crs) {
                    if (s_c.getCourseName() == crs_vec[tch_obj].getCourseName()) {
                        int score;
                        num++;
                        std::cout << num << ".    " << res.stu.getStudentName() << "\t" <<
res.stu.getStudentID() << std::endl;
                        while (true)
                        {
                            std::cout << "Please input his/her course score (0--100): ";
                            std::cin >> score;
                            if (0 <= score && score <= 100) {
                                s_c.setStudentCourseScore(score);
                                std::cout << "You have successfully entered his/her course
score: " << score << std::endl;
                                std::cout << "--- Next student ---" << std::endl;
                                system("pause");
                                break;
                            }
                            else std::cout << "Invalid input! Please input again" <<
std::endl;
                        }
                        break;
                    }
                }
            }
            std::cout << "You have completed entry of grades for all students!" << std::endl;
            return;
        }

        else if (choice == 2) {
            std::cout << "Next, you will add a new student to select your course..." <<
std::endl;
            std::string name, id;
            int class_num;
            while (true)
            {
                StudentResults temp_res;
                StudentCourse temp_sc;
                std::cout << "Please input student name: ";
                std::cin >> name;
                std::cout << "Please input student ID: ";
                std::cin >> id;
                if (isStudentIDExist(res_vec, id) != -1) {
```

```
                    std::cout << "This student ID have already existed! Please input again" << std::endl;
                    continue;
                }
                std::cout << "Please input student class (A new class will be created if not exist): ";
                std::cin >> class_num;
                if (1 <= class_num && class_num <= class_vec.size()) {
                    class_vec[class_num - 1].stu_amount++;
                }
                else {
                    std::cout << "There are " << class_vec.size() << " classes, now a new class " << class_vec.size() + 1 << " will be created" << std::endl;
                    class_num = class_vec.size() + 1;
                    ClassInfo temp_class = { class_num,1,0 };
                    class_vec.push_back(temp_class);
                }
                auto& tch = tch_vec[tch_obj];
                auto& crs = crs_vec[tch_obj];
                temp_sc.setStudentCourseInfo(name, id, class_num, tch.getTeacherName(), tch.getTeacherID(), tch.getTeacherCollege(), crs.getCourseName(), crs.getCourseCredit(), crs.getCourseIsPF(), 100);
                temp_res.stu.setStudentInfo(name, id, class_num);
                temp_res.PF_flag = -1;
                temp_res.GPA = 0;
                temp_res.ranking = 1;
                temp_res.stu_crs.push_back(temp_sc);
                int score;
                while (true)
                {
                    std::cout << "Please input his/her course score (0--100): ";
                    std::cin >> score;
                    if (0 <= score && score <= 100) {
                        temp_res.stu_crs[0].setStudentCourseScore(score);
                        res_vec.push_back(temp_res);
                        updateStudentResults(res_vec, res_vec.size() - 1);//此时学生是最后一个
                        updateClassInfo(res_vec, class_vec, class_num - 1);
                        updateAllClassInfo(res_vec, class_vec);
                        std::cout << "You have successfully entered his/her course score: " << score << std::endl;
                        int next;
                        std::cout << "Press 1 to continue entering, otherwise you will exit: ";
```

```
                            std::cin >> next;
                            if (next == 1) {
                                std::cout << "You have chose to continue entering!\n" << std::endl;
                                system("pause");
                                break;
                            }
                            else {
                                std::cout << "------ You are about to exit ------" << std::endl;
                                return;
                            }
                        }
                        else std::cout << "Invalid score! Not between 0 and 100" << std::endl;
                    }
                }
            }

            else {
                int next;
                std::cout << "Invalid input! Press 1 to continue entering, otherwise you will exit: ";
                std::cin >> next;
                if (next == 1) {
                    std::cout << "You have chose to continue entering!" << std::endl;
                    system("pause");
                    std::cout << std::endl;
                }
                else {
                    std::cout << "------ You are about to exit ------" << std::endl;
                    return;
                }
            }
        }
        return;
}

//修改课程成绩
void teacherModifyResults(std::vector<StudentResults>& res_vec, std::vector<Teacher>& tch_vec, std::vector<Course>& crs_vec, std::vector<ClassInfo>& class_vec, const int& tch_obj)
{
    system("cls");
    std::cout << "Dear " << tch_vec[tch_obj].getTeacherName() << ", you teach the course: " << crs_vec[tch_obj].getCourseName() << std::endl;
```

```
    std::cout << "Here are student that select your course and their grades:\n" << std::endl;
    std::cout << "-------------------------------------" << std::endl;
    std::cout << "number\tName\tStudent ID\tClass" << std::endl;
    int num = 0;
    std::vector<StudentCourse> temp_vec;
    temp_vec.clear();
    for (const auto& res : res_vec) {
        for (const auto& s_c : res.stu_crs) {
            if (s_c.getCourseName() == crs_vec[tch_obj].getCourseName()) {
                temp_vec.push_back(s_c);
                num++;
                std::cout << num << ".\t" << res.stu.getStudentName() << "\t" << res.stu.getStudentID() << "\t\t" << res.stu.getStudentClass() << std::endl;
                break;
            }
        }
    }
    std::cout << "-------------------------------------" << std::endl;
    int choice;
    while (true)
    {
        std::cout << "Please select the number before the student's name, and then modify grades: ";
        std::cin >> choice;
        if (1 <= choice && choice <= num) {
            std::cout << "You select " << choice << ".    " << temp_vec[choice - 1].getStudentName() << std::endl;
            std::cout << "His/Her current course score is: " << temp_vec[choice - 1].getStudentCourseScore() << std::endl;
            int score;
            int stuobj = isStudentExist(res_vec, temp_vec[choice - 1].getStudentName(), temp_vec[choice - 1].getStudentID());
            int crsobj = isStudentSelectCourse(res_vec, stuobj, temp_vec[choice - 1].getStudentCourseName());
            while (true)
            {
                std::cout << "Please input the modified score: ";
                std::cin >> score;
                if (0 <= score && score <= 100) {
                    int sure;
                    std::cout << "Are you sure to modify from " << temp_vec[choice - 1].getStudentCourseScore() << " to " << score << "?" << std::endl;
                    std::cout << "1---Yes  0---No: ";
                    std::cin >> sure;
```

```
                    while (true) {
                        if (sure == 1) {
                            std::cout << "Successful modification! You have modified it from " << temp_vec[choice - 1].getStudentCourseScore() << " to " << score << std::endl;
                            res_vec[stuobj].stu_crs[crsobj].setStudentCourseScore(score);
                            updateStudentResults(res_vec, stuobj);
                            updateClassInfo(res_vec, class_vec, res_vec[stuobj].stu.getStudentClass());
                            updateAllClassInfo(res_vec, class_vec);
                            break;
                        }
                        else if (sure == 0) {
                            std::cout << "You have canceled the modification!" << std::endl;
                            break;
                        }
                        else std::cout << "Invalid input! Please input again" << std::endl;
                    }
                    int next;
                    std::cout << "Press 1 to continue modifying, otherwise you will exit: ";
                    std::cin >> next;
                    if (next == 1) {
                        std::cout << "You have chose to continue modifying!" << std::endl;
                        system("pause");
                        std::cout << std::endl;
                        break;
                    }
                    else {
                        std::cout << "------ You are about to exit ------" << std::endl;
                        return;
                    }
                }
                else std::cout << "Invalid score! Not between 0 and 100" << std::endl;
            }
        }
        else std::cout << "Invalid input! Please input again" << std::endl;
    }
    return;
}

//查看该课程所有学生成绩,老师只能看见自己教的课的成绩
void teacherInquireCourseResults(const std::vector<StudentResults>& res_vec, const std::vector<Teacher>& tch_vec, const std::vector<Course>& crs_vec, const int& tch_obj)
{
```

```
    system("cls");
    std::cout << "Dear " << tch_vec[tch_obj].getTeacherName() << ", you teach the course: "
<< crs_vec[tch_obj].getCourseName() << std::endl;
    std::cout << "Here are this course grades: (non-ranking order)\n" << std::endl;
    std::cout << "Ranking\tName\tID\tScore\tGrade\tGPA" << std::endl;
    std::cout << "----------------------------------------" << std::endl;
    std::vector<StudentCourse> temp_vec;
    temp_vec.clear();
    for (const auto& res : res_vec) {
        for (const auto& sc : res.stu_crs) {
            if (sc.getCourseName() == crs_vec[tch_obj].getCourseName()) {
                temp_vec.push_back(sc);
            }
        }
    }
    std::sort(temp_vec.begin(), temp_vec.end(), compareStudentCourseByScore);
    int num = 1;
    for (const auto& s_c : temp_vec) {
        std::cout << num << "\t" << s_c.getStudentName() << "\t" << s_c.getStudentID() <<
"\t" << s_c.getStudentCourseScore() << "\t";
        std::cout << s_c.getStudentCourseGPA() << "\t" <<
Grade[s_c.getStudentCourseGrade()] << std::endl;
        num++;
    }
    std::cout << "----------------------------------------\n" << std::endl;
    std::cout << "To modify grades, please go to function 2" << std::endl;
    return;
}

//查看某个学生各门成绩，老师能看见所有学生的成绩
void teacherInquireStudentResults(const std::vector<StudentResults>& res_vec, const
std::vector<Teacher>& tch_vec, const std::vector<Course>& crs_vec, const int& tch_obj)
{
    system("cls");
    std::cout << "Dear " << tch_vec[tch_obj].getTeacherName() << ", you teach the course: "
<< crs_vec[tch_obj].getCourseName() << std::endl;
    std::cout << "You can view the grades of all students by name or student ID." <<
std::endl;
    std::cout << "To query by name, press 1. To query by student ID, press 2. To exit, press
0" << std::endl;
    std::cout << "The name query results may not be unique, but the student ID query result
is unique." << std::endl;
    int choice, exit;
    while (true)
```

```
    {
        std::cout << "\nPlease input you choice: ";
        std::cin >> choice;
        if (choice == 1) {
            std::string name;
            std::cout << "Please input student name: ";
            std::cin >> name;
            std::cout << "Here are the students you want:\n" << std::endl;
            std::cout    <<    "*****************************************"    <<
std::endl;
            for (const auto& res : res_vec) {
                if (res.stu.getStudentName() == name) {
                    std::cout << "*     Name: " << name << "              Student ID: " <<
res.stu.getStudentID() << "          Class: " << res.stu.getStudentClass() << std::endl;
                    std::cout << "*    GPA: " << res.GPA << "            Class ranking: " <<
res.ranking << std::endl;
                    std::cout << "*    Here are his/her all course grades:" << std::endl;
                    std::cout << "*    " << std::endl;
                    std::cout << "*    Course\tScore\tGrade\tGPA" << std::endl;
                    std::cout << "*    -------------------------" << std::endl;
                    for (const auto& s_c : res.stu_crs) {
                        std::cout << "*     " << s_c.getStudentCourseName() << "\t" <<
s_c.getStudentCourseScore() << "\t";
                        std::cout   <<   Grade[s_c.getStudentCourseGrade()]   <<   "\t"   <<
s_c.getStudentCourseGPA() << "\t" << std::endl;
                    }
                    std::cout << "*    -------------------------" << std::endl;
                }
            }
            std::cout << "*    " << std::endl;
            std::cout   <<   "*****************************************\n"   <<
std::endl;
            std::cout << "Press 1 to continue query, otherwise you will exit: ";
            std::cin >> exit;
            if (exit == 1) {
                std::cout << "You have chose to continue query!" << std::endl;
                system("pause");
                std::cout << std::endl;
            }
            else {
                std::cout << "You have chose: " << exit << std::endl;
                std::cout << "------ You are about to exit ------" << std::endl;
                return;
            }
```

```
        }

        else if (choice == 2) {
            std::string id;
            std::cout << "Please input student ID: ";
            std::cin >> id;
            std::cout << "Here are the students you want:\n" << std::endl;
            std::cout << "****************************************" << std::endl;
            for (const auto& res : res_vec) {
                if (res.stu.getStudentName() == id) {
                    std::cout << "Name: " << res.stu.getStudentName() << "          Student ID: " << id << "          Class: " << res.stu.getStudentClass() << std::endl;
                    std::cout << "GPA: " << res.GPA << "          Class ranking: " << res.ranking << std::endl;
                    std::cout << "Here are his/her all course grades: " << std::endl;
                    std::cout << "Course\tScore\tGrade\tGPA" << std::endl;
                    std::cout << "--------------------------" << std::endl;
                    for (const auto& s_c : res.stu_crs) {
                        std::cout << s_c.getStudentCourseName() << "\t" << s_c.getStudentCourseScore() << "\t";
                        std::cout << Grade[s_c.getStudentCourseGrade()] << "\t" << s_c.getStudentCourseGPA() << "\t" << std::endl;
                    }
                    std::cout << "--------------------------\n" << std::endl;
                }
            }
            std::cout << "****************************************\n" << std::endl;
            std::cout << "Press 1 to continue query, otherwise you will exit: ";
            std::cin >> exit;
            if (exit == 1) {
                std::cout << "You have chose to continue query!" << std::endl;
                system("pause");
                std::cout << std::endl;
            }
            else {
                std::cout << "You have chose: " << exit << std::endl;
                std::cout << "------ You are about to exit ------" << std::endl;
                return;
            }
        }

        else if (choice == 0) {
            std::cout << "You have chose: 0" << std::endl;
            std::cout << "------ You are about to exit ------" << std::endl;
```

```
            break;
            return;
        }
        else {
            std::cout << "Invalid input! Please input again" << std::endl;
        }
    }
    return;
}

//查看班级成绩，老师能查看所有班级的成绩
void teacherInquireClassResults(const std::vector<StudentResults>& res_vec, const std::vector<Teacher>& tch_vec, const std::vector<Course>& crs_vec, const std::vector<ClassInfo>& class_vec, const int& tch_obj)
{
    system("cls");
    std::cout << "Dear " << tch_vec[tch_obj].getTeacherName() << ", you teach the course: " << crs_vec[tch_obj].getCourseName() << std::endl;
    std::cout << "You can view the grades of all classes." << std::endl;
    while (true)
    {
        int class_num;
        std::cout << "Please select class: 1--" << class_vec.size() << " : ";
        std::cin >> class_num;
        if (1 <= class_num && class_num <= class_vec.size()) {
            std::cout << "\nYou select class: " << class_num << std::endl;
            std::cout << "Here are the students grades of class " << class_num << "\n\n";
            std::cout << "Name\tGPA\tClass ranking" << std::endl;
            std::cout << "-----------------------------" << std::endl;
            std::vector<StudentResults> temp_vec;
            temp_vec.clear();
            for (const auto& res : res_vec) {
                if (res.stu.getStudentClass() == class_num)
                    temp_vec.push_back(res);
            }
            std::sort(temp_vec.begin(), temp_vec.end(), compareStudentByGPA);
            for (const auto& temp : temp_vec) {
                std::cout << temp.stu.getStudentName() << "\t" << temp.GPA << "\t" << temp.ranking << std::endl;
            }
            std::cout << "-----------------------------\n" << std::endl;
            std::cout << "Class size: " << class_vec[class_num - 1].stu_amount << " average GPA: " << class_vec[class_num - 1].aver_GPA << "\n\n";
            int exit;
```

```
            std::cout << "Press 1 to continue query, otherwise you will exit: ";
            std::cin >> exit;
            if (exit == 1) {
                std::cout << "You have chose to continue query!" << std::endl;
                system("pause");
                std::cout << std::endl;
            }
            else {
                std::cout << "You have chose: " << exit << std::endl;
                std::cout << "------ You are about to exit ------" << std::endl;
                return;
            }
        }
        else
            std::cout << "Invalid input! Not between 1--" << class_vec.size() << std::endl;
    }
    return;
}


void teacherCheckInfo(const std::vector<Teacher>& tch_vec, const std::vector<Course>& crs_vec, const int& tch_obj)
{
    system("cls");
    tch_vec[tch_obj].displayInfo();
    crs_vec[tch_obj].displayInfo();
    return;
}
```