课 程 设 计 报 告

课程名称 C++面向对象编程

课题名称 学生成绩管理系统

班　　级 无 42
学　　号 2014011050
姓　　名 陈佳榕

2015 年　7 月　21 日

任务书：

自行虚构软件需求，并按照下面的基本要求，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，引入多函数的多态性、运算符重载等机制。

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，将将测试数据存在文件中，通过文件来进行数据读写来测试。

（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，每个人应体现自己的个性设计。

创新要求：

在基本要求达到后，可进行创新设计，如根据查找结果进行修改的功能。

学生成绩管理系统

目　录

1. 系统需求分析

学生的成绩信息包括以下 7 个人工输入的信息：学生姓名、学号、分数、课程名称（课程名称中不会出现空格）、相应课程学分、序号、学时以及以下 7 个不需要人工输入的信息：分数等级、未通过课程的数量、GPA（即总学分绩）、相应课程排名、总排名、未通过课程数、某个课程平均分及未通过人数。试设计学生成绩管理系统，使之能提供以下功能：

(1) 录入学生的成绩信息：从键盘输入数据（为避免重复从键盘输入数据，测试时可将数据存储在文件中，利用输入重定向功能读入），输入格式为：姓名 学号 课程名称 课程序号 课程学时 分数 相应课程学分

每行一条纪录。

例如：

冷锋 2014011088 语文 12345678 72 90 3

琪琪 2014011099 英语 15648254 36 88 2

(2) 修改某个学生的成绩信息：可以对成绩信息的人工输入部分进行修改，先显示修改前的成绩信息，修改之后再显示修改后的成绩信息。

(3) 查询某个学生的成绩信息：查询结果按照分数升序排序，相同成绩按照课程序号升序排序。

(4) 统计某个科目的成绩情况，查询结果先按分数降序排序，分数相同的学生按学号升序排序；

(5) 查询某个学生的 GPA，查询结果输出这个学生的姓名，学号以及 GPA；

(6) 统计所有学生 GPA 的排名，统计结果先按 GPA 降序排序，GPA 相同的按学号升序排序；

(7) 查看某个学生的未通过课程，结果输出未通过课程数量及未通过的科目及相应成绩，输出结果先按成绩降序排序，成绩相同的按课程序号升序排序；

(8) 查看某门课的平均分及未通过人数，结果输出平均分，未通过人数及具体情况，包括学生姓名、学号、分数，输出结果先按分数降序排序，分数相同的按学号升序排序；

(9) 系统以菜单方式工作。（所谓菜单指用户可以自由选择所要执行的功能。学生可以通过以上功能录入信息、修改信息、查询信息、整理统计出所要了解的信息，除了要实现上述的基本功能之外，本系统还应该在细节上下工夫，使用户使用方便，在使用的过程中保持一个愉快的心情。学生成绩管理系统有广大的用户群，这其中有老师、有教务处主任，也可以供给一般用户使用。总之该系统可以满足用户需求，实现对其管理人员的成绩管理工作。）

2. 系统总体设计

学生成绩管理系统包含八个大的功能，分别是：录入学生成绩信息、修改学生成绩信息、查询某个学生的成绩信息、统计某个科目的成绩情况，查询某个学生的 GPA、统计所有学生的 GPA 排名，查看某个学生的课程未通过情况，查看某门课程的平均分及未通过情况。学生的成绩信息主要包含学生姓名、学号、分数、分数等级、课程名称、课程序号、学时、相应课程学分、GPA（即总学分绩）、相应课程排名、总排名、未通过课程数 、某门课程平均分及未通过人数。

在录入学生成绩信息时根据系统提示逐一输入。每输入完一条信息，系统会提示是否继续输入，用户可以选择继续或返回主菜单。

在修改学生成绩信息时，用户首先输入要修改的学生姓名或学号，系统会检索，如果系统中有该学生的相关信息，则系统会输出该学生的信息，然后提示用户修改该学生的哪一部分信息，用户可自行选择，修改完成之后再输出修改后的学生信息。如果系统中没有该学生的相关信息，则系统会给相关提示，用户可以选择继续输入要修改的学生姓名或学号，抑或选择退出返回主菜单。

在查询学生成绩信息时，用户先输入要查询的学生姓名或学号，系统检索判断是否存在该学生的信息，如果系统中有该学生的相关信息，则查询结果按照分数升序排序，相同成绩按照课程序号升序排序，输出学生的姓名、学号、课程名称、课程序号、学时、分数及学分；如果系统中没有该学生的相关信息，则系统会给相关提示，用户可以选择继续输入要查询的学生姓名或选择退出返回主菜单。

在统计某个科目的成绩情况时，用户先输入课程名称或课程序号，系统检索判断是否存在该课程，如果有，那么查询结果可以有两种选择，一种是只显示分数及排名，按分数降序排序；另一种就是既显示排名也显示学生姓名、学号，先按分数降序排序，分数相同的学生按学号升序排序；如果没有，那么系统会给相关提示，用户可以选择继续输入要查询的可课程名称或选择退出返回主菜单。

在查询某个学生的 GPA 时（为了避免同姓名的学生造成统计出错，故以学号为输入），用户输入所要查询的学号，系统会检索判断是否存在该学号，如果存在，则输出该学号对应的学生的姓名、学号以及 GPA；如果不存在，那么系统会给出相应提示，用户可以选择继续输入要查询的学号或者选择退出返回主菜单。

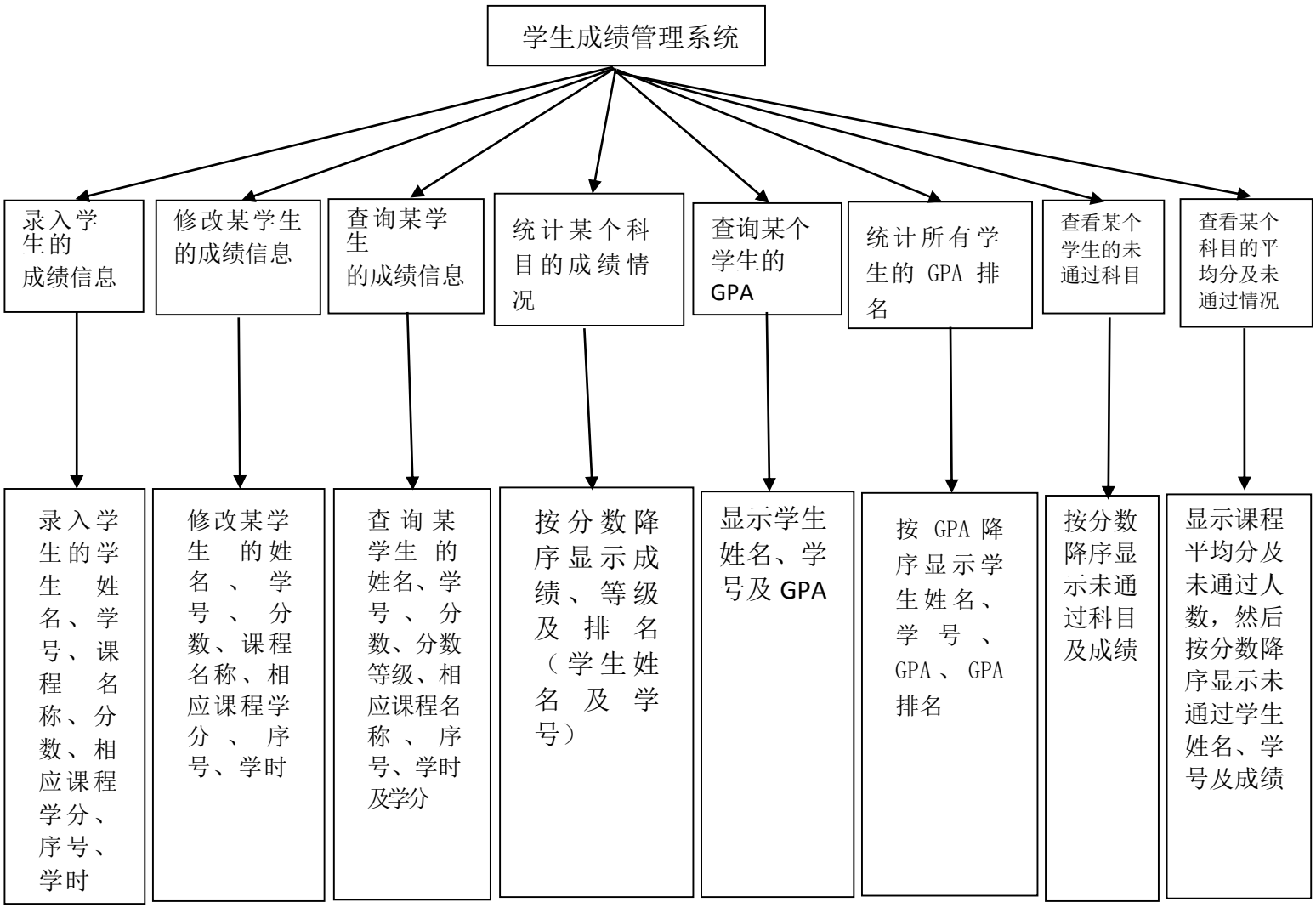
在统计所有学生 GPA 的排名时，统计结果先按 GPA 由高到低排序，GPA 相同的按学号升序排序。输出时输出学生姓名、学号、GPA 及 GPA 排名。

在查询某个学生的课程未通过情况时（为了避免同姓名的学生造成统计出错，故

以学号为输入），用户输入所要查询的学号，系统会检索是否存在该学号，如果存在，则输出相应的情况，有未通过的课程就输出相应的课程名称及分数；没有未通过的课程，系统也会给出提示。如果不存在该学号，那么系统会给出相应提示，用户可以选择继续输入要查询的学号或者选择退出返回主菜单。

在查询某门课程的平均分及未通过情况时，用户输入所要查询的课程名称或课程序号，系统会检索是否存在该课程，如果存在，则输出课程平均分、未通过人数及相应的未通过情况；如果不存在，那么系统会给出相应提示，用户可以选择继续输入要查询的课程名称或课程序号，抑或选择退出返回主菜单。

学生成绩管理系统中功能模块图：



3. 系统详细设计

学生成绩管理系统中四个类的类层次图为：

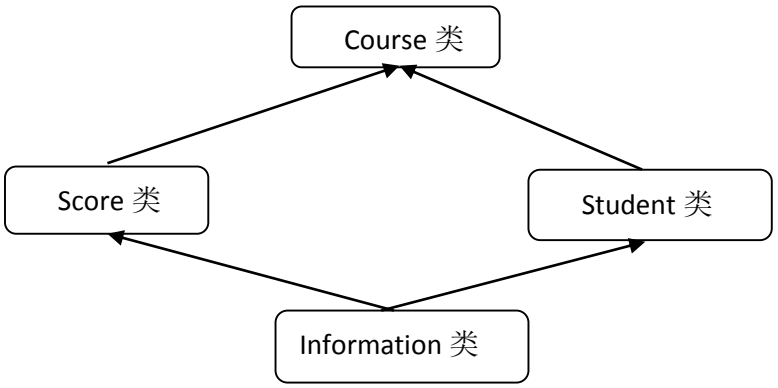
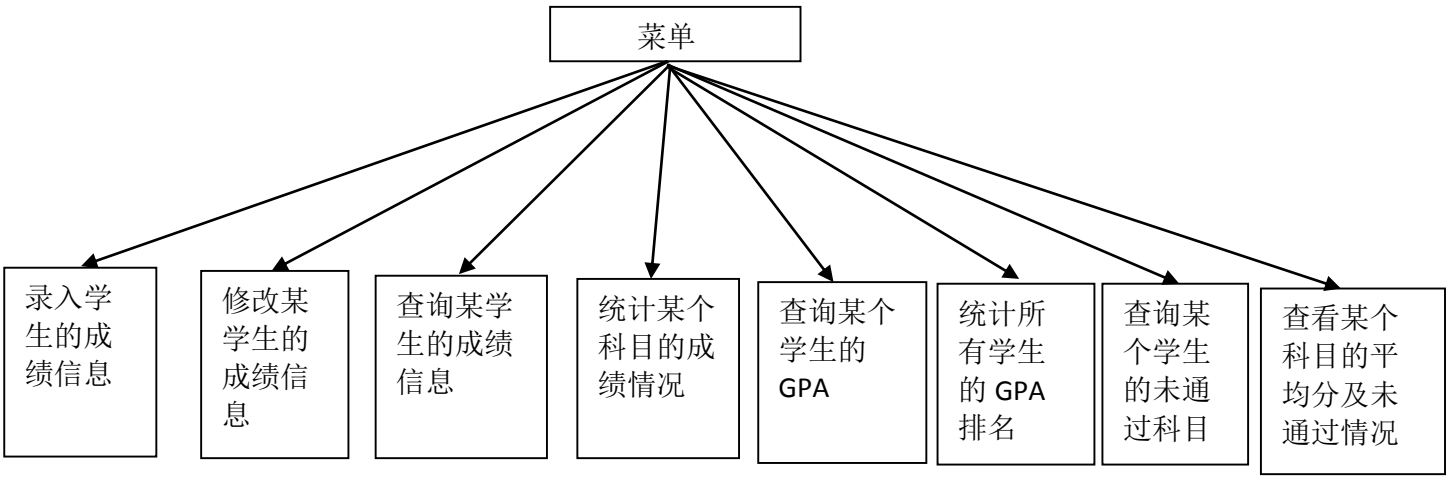


图 2　学生成绩管理系统中中四个类的类层次图

学生成绩管理系统中各功能模块的实现：

图 3　学生成绩管理系统中菜单函数的功能图



1、学生成绩信息录入功能模块:

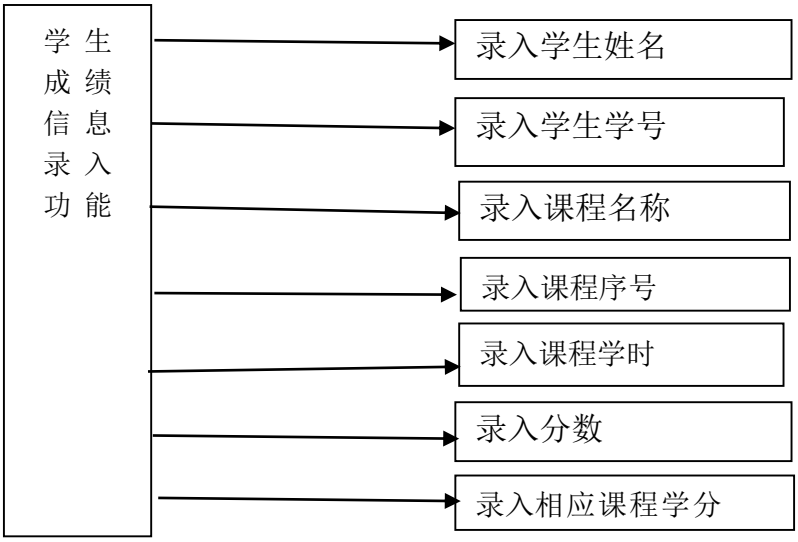


图 4　学生成绩信息录入的功能

2、修改某学生成绩信息功能的功能模块：

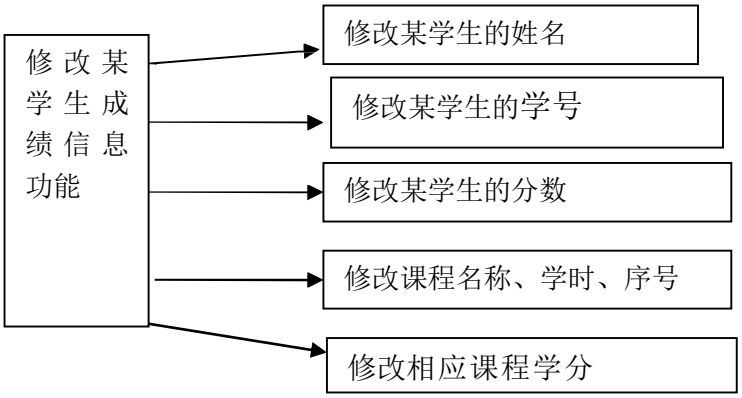


图 5　学生成绩管理系统修改学生成绩信息功能图

3 查询某学生成绩信息功能模块：

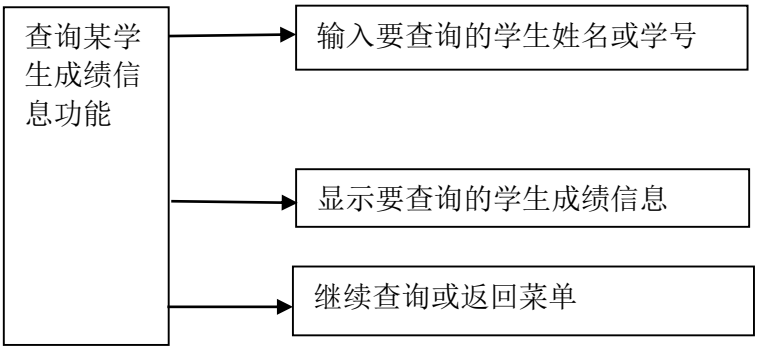


图 6 学生成绩管理系统查询学生成绩信息功能图

4、统计某个科目的成绩情况功能模块：

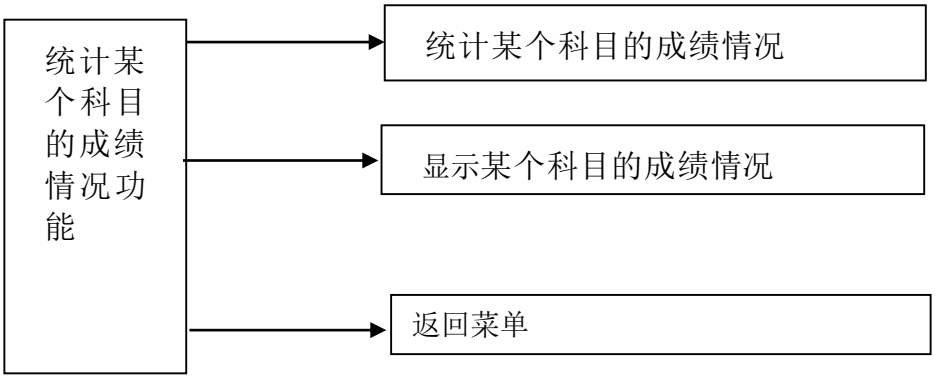


图 7 统计某个科目的成绩情况功能

5.查询某个学生的 GPA 功能模块

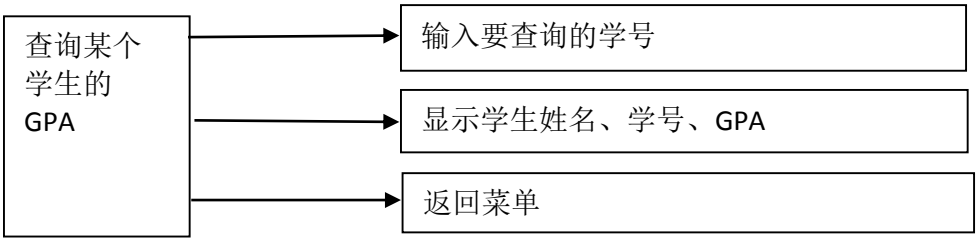


图 8 查询某个学生的 GPA 功能图

6、统计所有学生的 GPA 排名功能模块

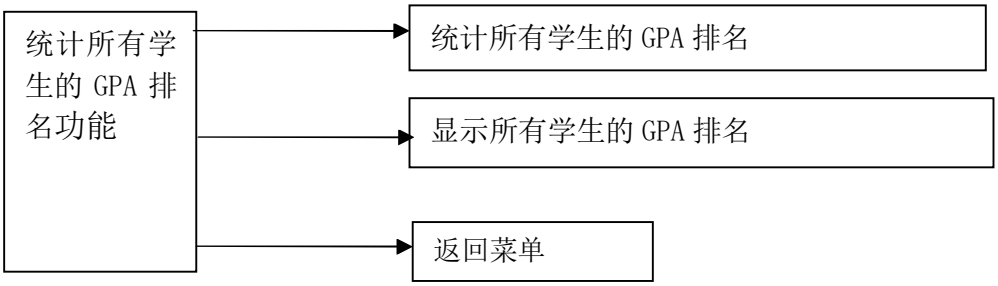


图 9 统计所有学生的 GPA 排名功能图

7、查询某个学生的未通过科目情况功能模块

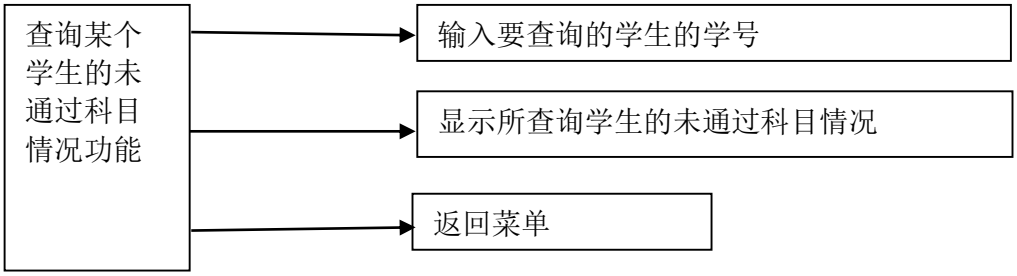


图 10 查询某个学生的未通过科目情况功能图

8、查询某个科目的平均分及未通过情况功能模块

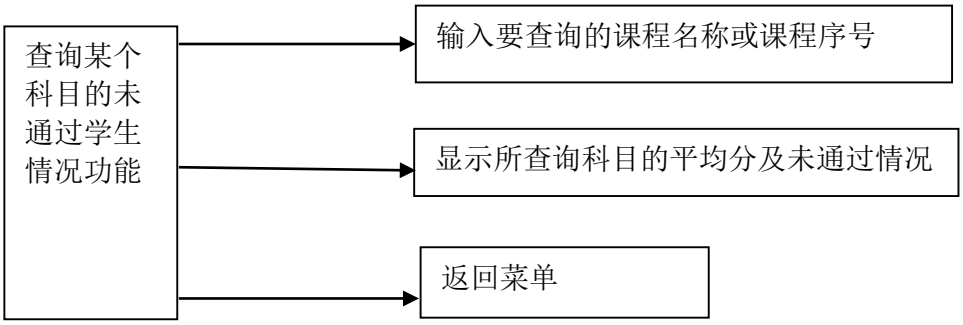


图 11 查询某个科目的平均分及未通过情况功能图

学生考勤管理系统中四个类的 UML 类图为：

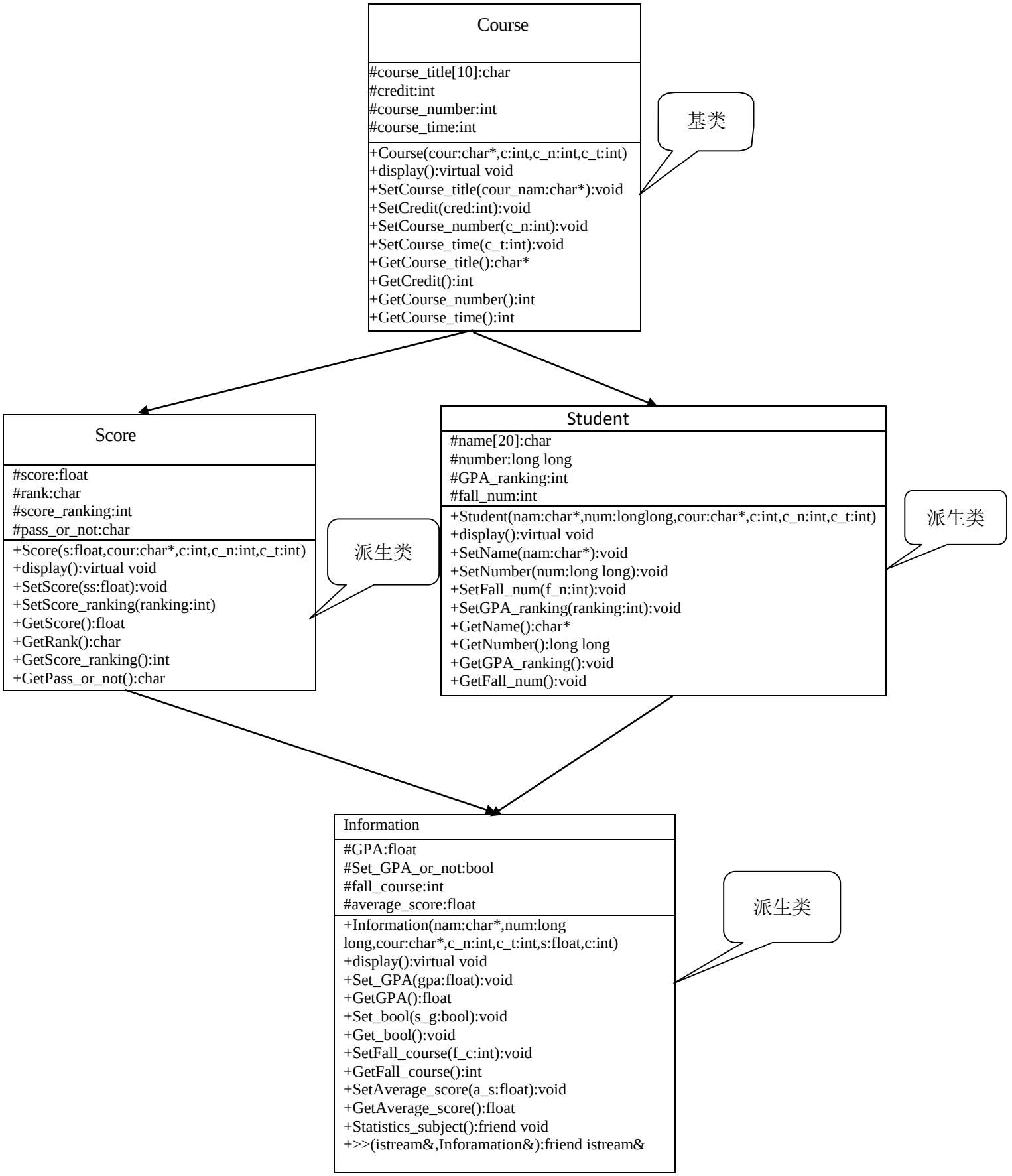


图 9　　学生考勤管理系统中四个类的 UML 类 图

4. 系统调试

程序调试过程中出现了以下问题：

①我在头文件“User_Interface.h”中定义了两个函数，然后又分别在两个 cpp 文件中包含了这个头文件，结果编译时就总是出错。尽管用了“#pragma once”，可依然报错，后来就我先把这个两个函数注释掉，然后试着又定义一个函数，不过这次我只是在头文件中声明，而在 cpp 文件中定义，实验的结果是成功了。虽然是成功了，但是我不太清楚其中的具体原因，如果可以的话希望老师解释一下。

②我在同一个函数之中先后打开了两个文件，且定义文件流对象时用的是同一个文件流对象名即 outfile，后来发现这样会出错，于是打开第二个文件时改了文件流对象名，改用 outfiles，然后就通过编译了。

③实现查询函数时，发现不能按照预想的按照分数升序排序，后来发现是循环嵌套出现了问题，把“}”的位置调动到正确的位置，功能便实现了。

④统计某个科目的成绩情况时，发现成绩相同的排名却不同，于是我加了一个判断语句，如果成绩相同就设置他们的排名相同，于是问题顺利解决。

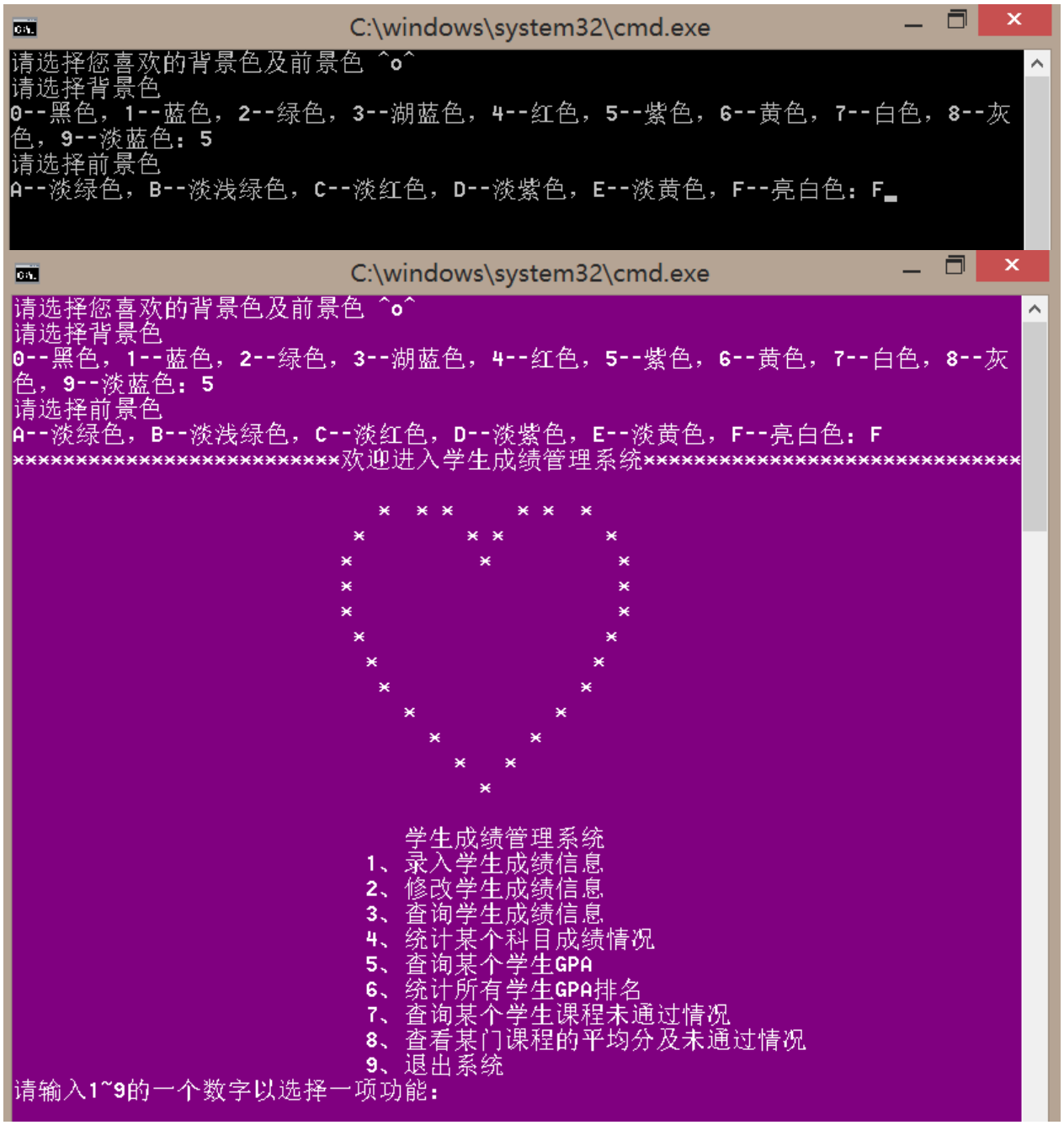
⑤设置背景色及前景色时用到的 strcat_s 函数，因为第二个参数设置的不合理，出现了像缓冲区不够这种问题，后来找到了解决办法，第二个参数只要设置为“源串大小 + 目标串大小 + 字符串结束符大小("\0")”就可以了。

⑥查询某个学生的 GPA 时，发现当输入的学号并不存在时，程序会直接崩溃，后来发现原来是查询时一个 if 语句少了一个控制条件，使得数组越界，改正之后功能正常。

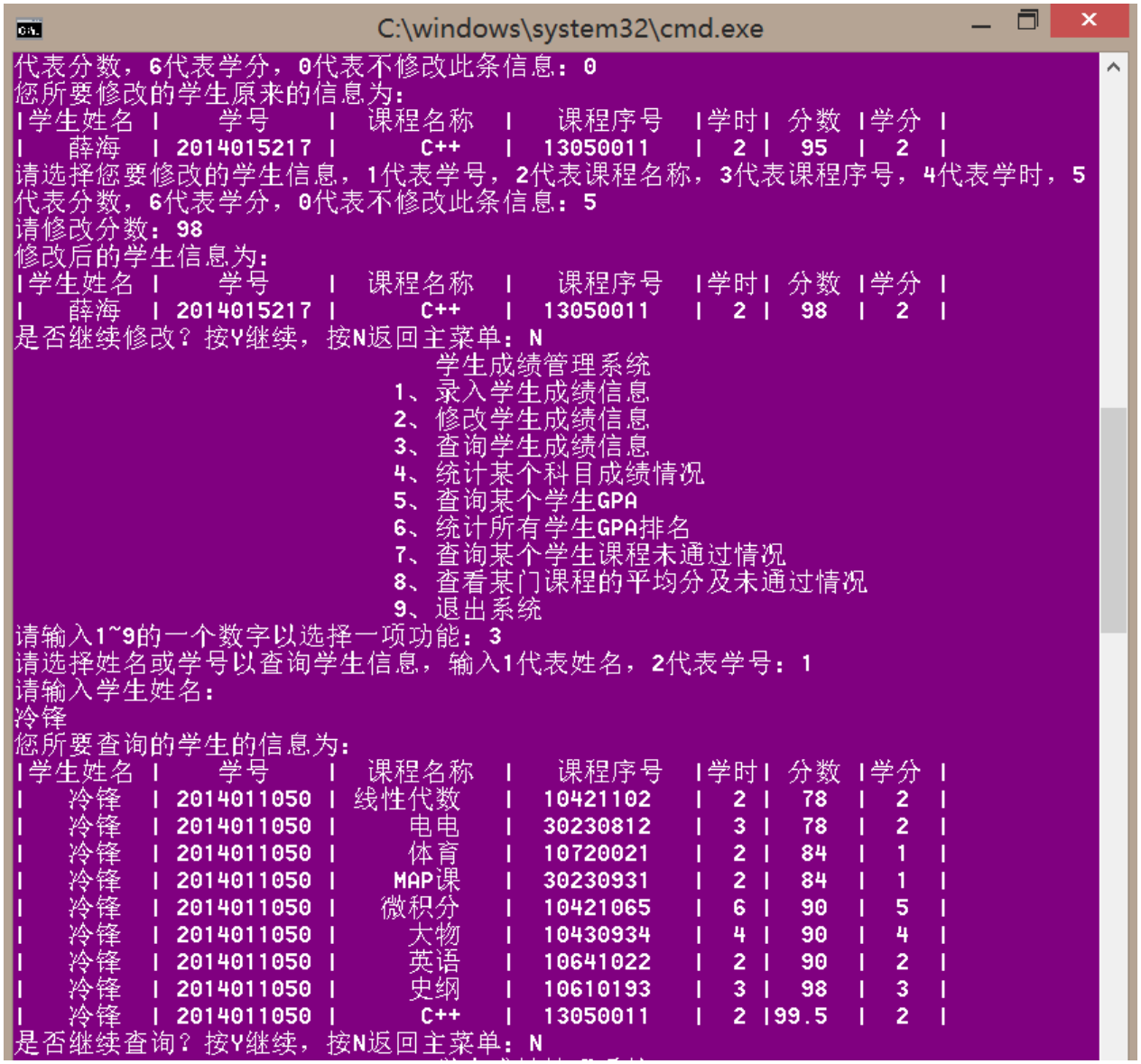
⑦判断是否继续时，虽然按了 N（表示否），但结果却还在继续，后来发现是形参的问题，因为形参和实参并不能公用一个空间导致了出错，于是我把形参改为一个引用类型，结果就正常了。读取学生人次时也是同样的情况。

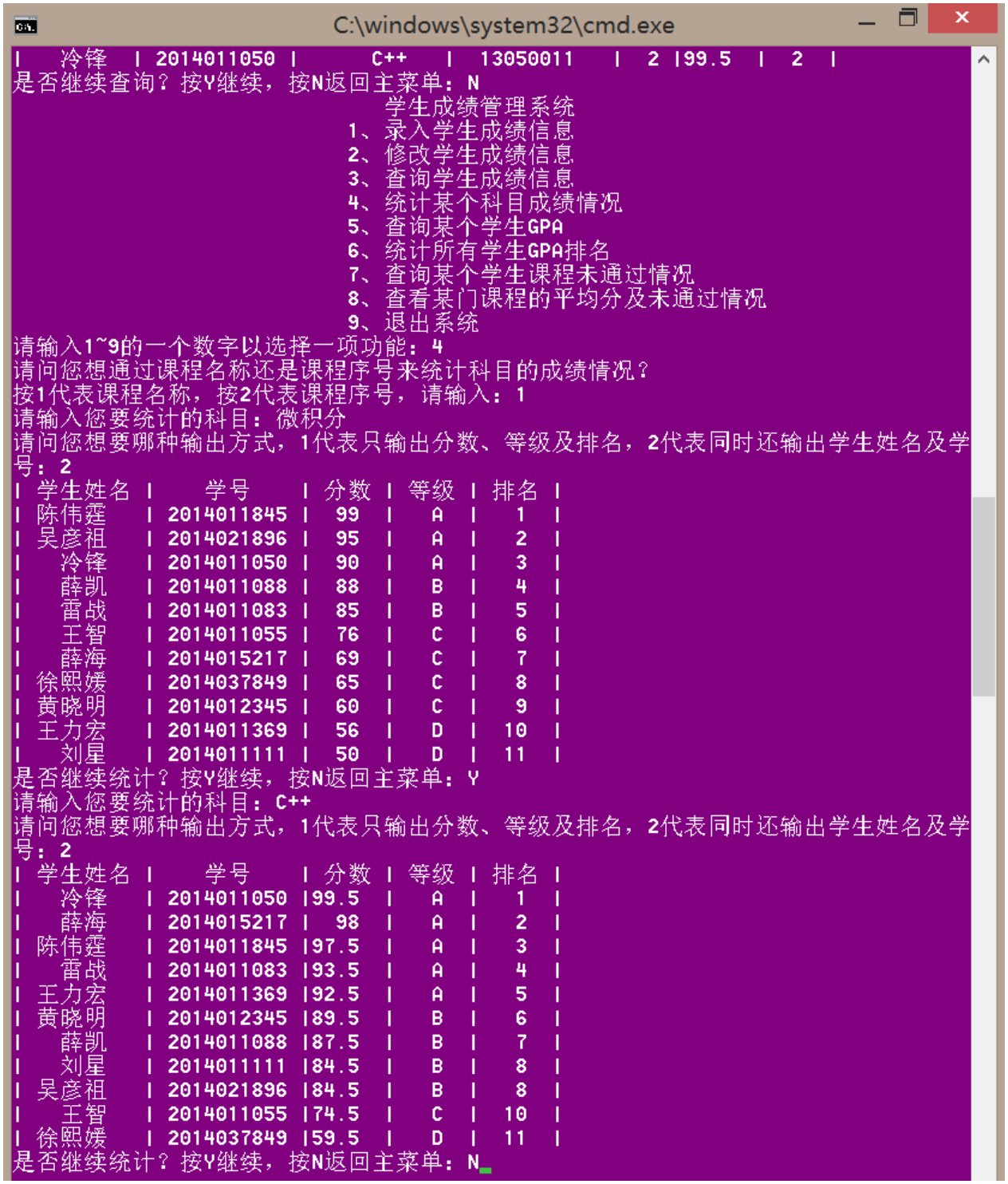
5.测试结果、使用说明书及编程体会

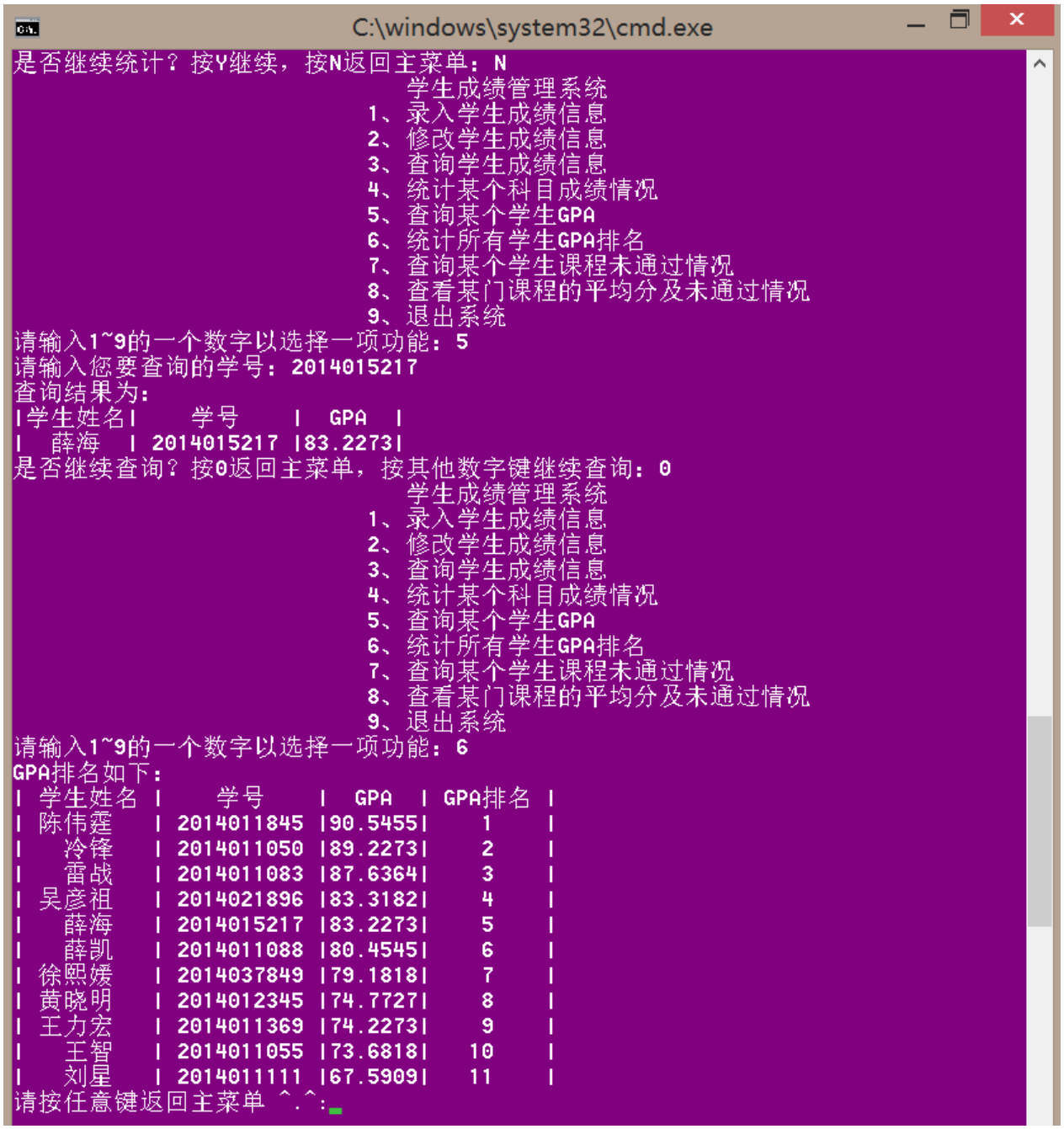
本程序的测试数据文件是“Stu_num.dat”及“Information.dat”， 测试结果截图如下图所示。

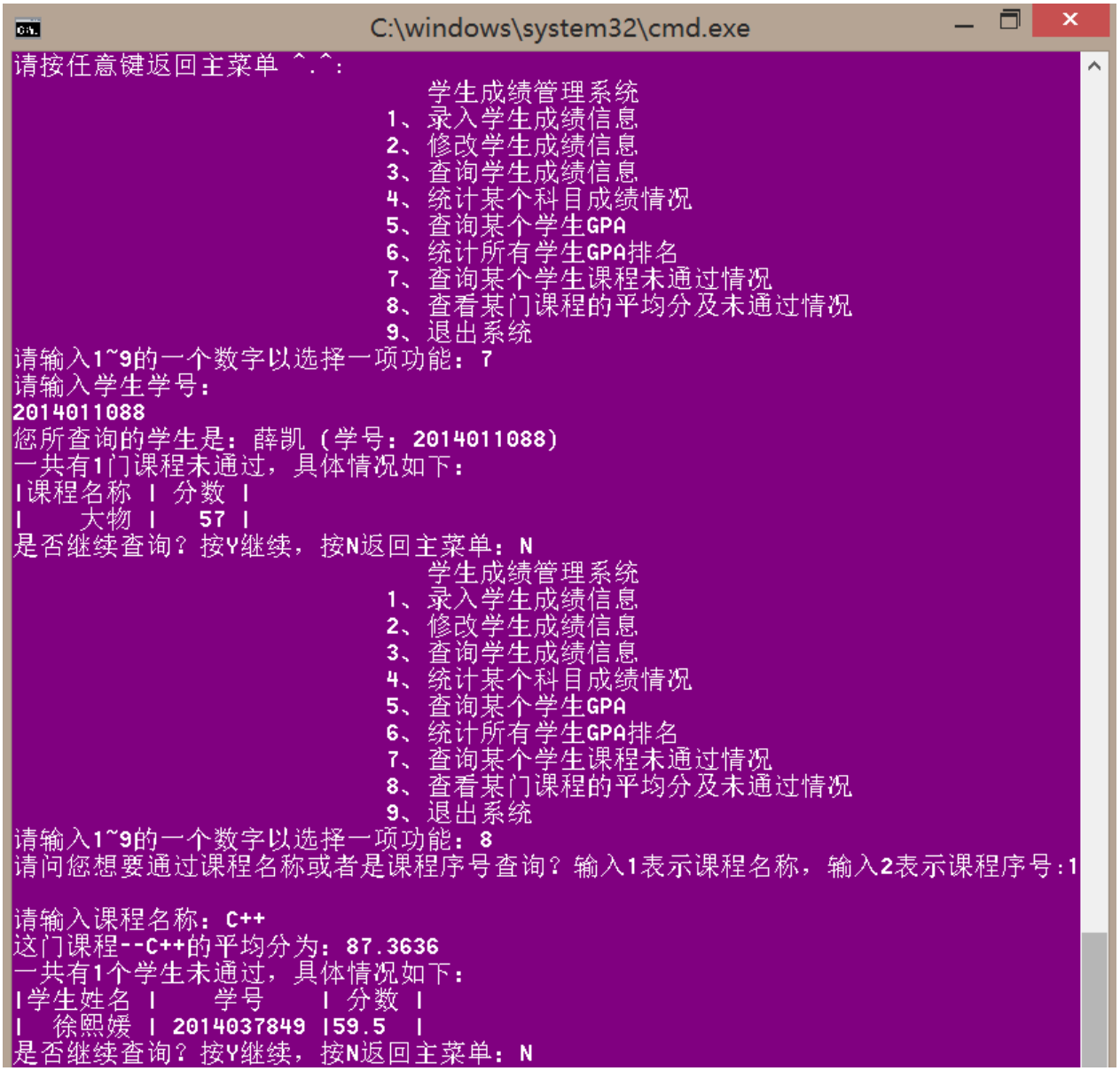



C:\windows\system32\cmd.exe
8、查看某门课程的平均分及未通过情况
9、退出系统
请输入1~9的一个数字以选择一项功能：2
请选择姓名或学号以修改学生信息，输入1代表姓名，2代表学号：1
请输入学生姓名：
薛海
您所要修改的学生原来的信息为：
|学生姓名 | 学号 | 课程名称 | 课程序号 |学时| 分数 |学分 |
| 薛海 | 2014015217 | 微积分 | 10421065 | 6 | 69 | 5 |
请选择您要修改的学生信息，1代表学号，2代表课程名称，3代表课程序号，4代表学时，5代表分数，6代表学分，0代表不修改此条信息：0
您所要修改的学生原来的信息为：
|学生姓名 | 学号 | 课程名称 | 课程序号 |学时| 分数 |学分 |
| 薛海 | 2014015217 | 线性代数 | 10421102 | 2 | 88 | 2 |
请选择您要修改的学生信息，1代表学号，2代表课程名称，3代表课程序号，4代表学时，5代表分数，6代表学分，0代表不修改此条信息：0
您所要修改的学生原来的信息为：
|学生姓名 | 学号 | 课程名称 | 课程序号 |学时| 分数 |学分 |
| 薛海 | 2014015217 | 大物 | 10430934 | 4 | 74 | 4 |
请选择您要修改的学生信息，1代表学号，2代表课程名称，3代表课程序号，4代表学时，5代表分数，6代表学分，0代表不修改此条信息：0
您所要修改的学生原来的信息为：
|学生姓名 | 学号 | 课程名称 | 课程序号 |学时| 分数 |学分 |
| 薛海 | 2014015217 | 史纲 | 10610193 | 3 | 96 | 3 |
请选择您要修改的学生信息，1代表学号，2代表课程名称，3代表课程序号，4代表学时，5代表分数，6代表学分，0代表不修改此条信息：0
您所要修改的学生原来的信息为：
|学生姓名 | 学号 | 课程名称 | 课程序号 |学时| 分数 |学分 |
| 薛海 | 2014015217 | 英语 | 10641022 | 2 | 88 | 2 |
请选择您要修改的学生信息，1代表学号，2代表课程名称，3代表课程序号，4代表学时，5代表分数，6代表学分，0代表不修改此条信息：0
您所要修改的学生原来的信息为：
|学生姓名 | 学号 | 课程名称 | 课程序号 |学时| 分数 |学分 |
| 薛海 | 2014015217 | 体育 | 10720021 | 2 | 91 | 1 |
请选择您要修改的学生信息，1代表学号，2代表课程名称，3代表课程序号，4代表学时，5代表分数，6代表学分，0代表不修改此条信息：0
您所要修改的学生原来的信息为：
|学生姓名 | 学号 | 课程名称 | 课程序号 |学时| 分数 |学分 |
| 薛海 | 2014015217 | 电电 | 30230812 | 3 | 88 | 2 |
请选择您要修改的学生信息，1代表学号，2代表课程名称，3代表课程序号，4代表学时，5代表分数，6代表学分，0代表不修改此条信息：0
您所要修改的学生原来的信息为：
|学生姓名 | 学号 | 课程名称 | 课程序号 |学时| 分数 |学分 |
| 薛海 | 2014015217 | MAP课 | 30230931 | 2 | 87 | 1 |
请选择您要修改的学生信息，1代表学号，2代表课程名称，3代表课程序号，4代表学时，5代表分数，6代表学分，0代表不修改此条信息：0


C:\windows\system32\cmd.exe
代表分数，6代表学分，0代表不修改此条信息：0
您所要修改的学生原来的信息为：
|学生姓名 | 学号 | 课程名称 | 课程序号 |学时| 分数 |学分 |
| 薛海 | 2014015217 | C++ | 13050011 | 2 | 95 | 2 |
请选择您要修改的学生信息，1代表学号，2代表课程名称，3代表课程序号，4代表学时，5代表分数，6代表学分，0代表不修改此条信息：5
请修改分数：98
修改后的学生信息为：
|学生姓名 | 学号 | 课程名称 | 课程序号 |学时| 分数 |学分 |
| 薛海 | 2014015217 | C++ | 13050011 | 2 | 98 | 2 |
是否继续修改？按Y继续，按N返回主菜单：N
学生成绩管理系统
1、录入学生成绩信息
2、修改学生成绩信息
3、查询学生成绩信息
4、统计某个科目成绩情况
5、查询某个学生GPA
6、统计所有学生GPA排名
7、查询某个学生课程未通过情况
8、查看某门课程的平均分及未通过情况
9、退出系统
请输入1~9的一个数字以选择一项功能：3
请选择姓名或学号以查询学生信息，输入1代表姓名，2代表学号：1
请输入学生姓名：
冷锋
您所要查询的学生的信息为：
学生姓名	学号	课程名称	课程序号	学时	分数	学分
冷锋	2014011050	线性代数	10421102	2	78	2
冷锋	2014011050	电电	30230812	3	78	2
冷锋	2014011050	体育	10720021	2	84	1
冷锋	2014011050	MAP课	30230931	2	84	1
冷锋	2014011050	微积分	10421065	6	90	5
冷锋	2014011050	大物	10430934	4	90	4
冷锋	2014011050	英语	10641022	2	90	2
冷锋	2014011050	史纲	10610193	3	98	3
冷锋	2014011050	C++	13050011	2	99.5	2
是否继续查询？按Y继续，按N返回主菜单：N


C:\windows\system32\cmd.exe
| 冷锋 | 2014011050 | C++ | 13050011 | 2 |99.5 | 2 |
是否继续查询？按Y继续，按N返回主菜单：N
学生成绩管理系统
1、录入学生成绩信息
2、修改学生成绩信息
3、查询学生成绩信息
4、统计某个科目成绩情况
5、查询某个学生GPA
6、统计所有学生GPA排名
7、查询某个学生课程未通过情况
8、查看某门课程的平均分及未通过情况
9、退出系统
请输入1~9的一个数字以选择一项功能：4
请问您想通过课程名称还是课程序号来统计科目的成绩情况？
按1代表课程名称，按2代表课程序号，请输入：1
请输入您要统计的科目：微积分
请问您想要哪种输出方式，1代表只输出分数、等级及排名，2代表同时还输出学生姓名及学号：2
学生姓名	学号	分数	等级	排名
陈伟霆	2014011845	99	A	1
吴彦祖	2014021896	95	A	2
冷锋	2014011050	90	A	3
薛凯	2014011088	88	B	4
雷战	2014011083	85	B	5
王智	2014011055	76	C	6
薛海	2014015217	69	C	7
徐熙媛	2014037849	65	C	8
黄晓明	2014012345	60	C	9
王力宏	2014011369	56	D	10
刘星	2014011111	50	D	11
是否继续统计？按Y继续，按N返回主菜单：Y
请输入您要统计的科目：C++
请问您想要哪种输出方式，1代表只输出分数、等级及排名，2代表同时还输出学生姓名及学号：2
学生姓名	学号	分数	等级	排名
冷锋	2014011050	99.5	A	1
薛海	2014015217	98	A	2
陈伟霆	2014011845	97.5	A	3
雷战	2014011083	93.5	A	4
王力宏	2014011369	92.5	A	5
黄晓明	2014012345	89.5	B	6
薛凯	2014011088	87.5	B	7
刘星	2014011111	84.5	B	8
吴彦祖	2014021896	84.5	B	8
王智	2014011055	74.5	C	10
徐熙媛	2014037849	59.5	D	11
是否继续统计？按Y继续，按N返回主菜单：N


C:\windows\system32\cmd.exe
是否继续统计？按Y继续，按N返回主菜单：N
学生成绩管理系统
1、录入学生成绩信息
2、修改学生成绩信息
3、查询学生成绩信息
4、统计某个科目成绩情况
5、查询某个学生GPA
6、统计所有学生GPA排名
7、查询某个学生课程未通过情况
8、查看某门课程的平均分及未通过情况
9、退出系统
请输入1~9的一个数字以选择一项功能：5
请输入您要查询的学号：2014015217
查询结果为：
|学生姓名| 学号 | GPA |
| 薛海 | 2014015217 |83.2273|
是否继续查询？按0返回主菜单，按其他数字键继续查询：0
学生成绩管理系统
1、录入学生成绩信息
2、修改学生成绩信息
3、查询学生成绩信息
4、统计某个科目成绩情况
5、查询某个学生GPA
6、统计所有学生GPA排名
7、查询某个学生课程未通过情况
8、查看某门课程的平均分及未通过情况
9、退出系统
请输入1~9的一个数字以选择一项功能：6
GPA排名如下：
学生姓名	学号	GPA	GPA排名
陈伟霆	2014011845	90.5455	1
冷锋	2014011050	89.2273	2
雷战	2014011083	87.6364	3
吴彦祖	2014021896	83.3182	4
薛海	2014015217	83.2273	5
薛凯	2014011088	80.4545	6
徐熙媛	2014037849	79.1818	7
黄晓明	2014012345	74.7727	8
王力宏	2014011369	74.2273	9
王智	2014011055	73.6818	10
刘星	2014011111	67.5909	11
请按任意键返回主菜单 ^.^:


C:\windows\system32\cmd.exe
请按任意键返回主菜单 ^.^:
学生成绩管理系统
1、录入学生成绩信息
2、修改学生成绩信息
3、查询学生成绩信息
4、统计某个科目成绩情况
5、查询某个学生GPA
6、统计所有学生GPA排名
7、查询某个学生课程未通过情况
8、查看某门课程的平均分及未通过情况
9、退出系统
请输入1~9的一个数字以选择一项功能: 7
请输入学生学号:
2014011088
您所查询的学生是: 薛凯 (学号: 2014011088)
一共有1门课程未通过, 具体情况如下:
|课程名称 | 分数 |
| 大物 | 57 |
是否继续查询? 按Y继续, 按N返回主菜单: N
学生成绩管理系统
1、录入学生成绩信息
2、修改学生成绩信息
3、查询学生成绩信息
4、统计某个科目成绩情况
5、查询某个学生GPA
6、统计所有学生GPA排名
7、查询某个学生课程未通过情况
8、查看某门课程的平均分及未通过情况
9、退出系统
请输入1~9的一个数字以选择一项功能: 8
请问您想要通过课程名称或者是课程序号查询? 输入1表示课程名称, 输入2表示课程序号:1
请输入课程名称: C++
这门课程--C++的平均分为: 87.3636
一共有1个学生未通过, 具体情况如下:
|学生姓名 | 学号 | 分数 |
| 徐熙媛 | 2014037849 |59.5 |
是否继续查询? 按Y继续, 按N返回主菜单: N


C:\windows\system32\cmd.exe
请输入课程名称：C++
这门课程--C++的平均分为：87.3636
一共有1个学生未通过，具体情况如下：
|学生姓名 | 学号 | 分数 |
| 徐熙媛 | 2014037849 |59.5 |
是否继续查询？按Y继续，按N返回主菜单：N
学生成绩管理系统
1、录入学生成绩信息
2、修改学生成绩信息
3、查询学生成绩信息
4、统计某个科目成绩情况
5、查询某个学生GPA
6、统计所有学生GPA排名
7、查询某个学生课程未通过情况
8、查看某门课程的平均分及未通过情况
9、退出系统
请输入1~9的一个数字以选择一项功能：8
请问您想要通过课程名称或者是课程序号查询？输入1表示课程名称，输入2表示课程序号:1
请输入课程名称：电电
这门课程--电电的平均分为：73.3636
一共有2个学生未通过，具体情况如下：
学生姓名	学号	分数
吴彦祖	2014021896	53
黄晓明	2014012345	45
是否继续查询？按Y继续，按N返回主菜单：N
学生成绩管理系统
1、录入学生成绩信息
2、修改学生成绩信息
3、查询学生成绩信息
4、统计某个科目成绩情况
5、查询某个学生GPA
6、统计所有学生GPA排名
7、查询某个学生课程未通过情况
8、查看某门课程的平均分及未通过情况
9、退出系统
请输入1~9的一个数字以选择一项功能：9
谢谢使用！欢迎您再次使用。
请按任意键继续. . .

使用说明书：

首先，可执行文件需与“Stu_num.dat”及“Information.dat”这两个数据文件以及“天空之城.wav”这个音乐文件放置于同一个目录下。若是想从头开始输入学生成绩信息而不用原来的测试数据，那么需要删除“Information.dat”或清空其中的信息，还需要将“Stu_num.dat”中的数字修改为0。否则的话则不需要做任何删除及修改。

打开可执行文件之后，系统提示设置背景色，用户根据提示输入数字，然后系统继续提示设置前景色，用户根据提示输入字母，即可完成背景色及前景色的设置。

然后进入主菜单，用户根据系统提示输入数字选择相应功能：

⑴选择1进入录入功能，用户根据系统提示依次输入学生姓名（默认中文名）、学生学号（默认10位）、课程名称（没有空格且不超过10位）、课程序号、课程学时、分数及课程学分，输完一个学生之后系统会提示继续输入还是返回主菜单，用户根据系统提示输入相应数字即可。

⑵选择2进入修改功能，用户先根据系统提示输入相应数字选择要通过姓名还是学号来修改，然后输入相应的姓名或学号，若不存在此姓名或学号，系统提示要返回主菜单还是继续输入姓名或学号，用户只要根据提示输入相应数字即可。若存在，系统会逐一显示该学生的信息，用户根据提示输入相应数字以修改相应的信息或者跳过不修改。修改完成之后，系统提示继续修改与否，用户根据提示输入相应字母选择继续修改抑或返回主菜单。

⑶选择3进入查询学生成绩信息功能，用户先根据系统提示输入相应数字选择要通过姓名还是学号来查询，然后输入相应的姓名或学号，若不存在此姓名或学号，系统提示要返回主菜单还是继续输入姓名或学号，用户只要根据提示输入相应数字即可。若存在，系统会显示该学生相应的成绩信息，然后提示用户要继续查询与否，用户根据提示输入相应字母选择继续查询抑或返回主菜单。

⑷选择4进入统计某个科目成绩情况功能，用户先根据系统提示输入相应数字选择要通过课程名称还是课程序号来查询，然后输入相应的课程名称或课程序号，若不存在此课程名称或课程序号，系统提示要返回主菜单还是继续输入课程名称或课程序号，用户只要根据提示输入相应数字即可。若存在，系统会提示用户选择一种输出方式，用户根据提示输入相应的数字即可，然后系统便会根据用户的选择输出相应的统计结果，并提示用户要继续统计与否，用户根据提示输入相应字母选择继续统计抑或返回主菜单。

⑸选择 5 进入统计某个学生 GPA 功能，用户只要输入相应的学号即可。若不存在此学号，系统提示要返回主菜单还是继续输入学号，用户只要根据提示输入相应数字即可。若存在，则系统会输出相应的信息，并提示用户要继续统计与否，用户根据提示输入相

应字母选择继续查询抑或返回主菜单。

⑹选择6进入统计所有学生GPA的功能，若系统中无学生信息，系统会作出相应提示并提示按任意键返回主菜单，用户可以按任意键返回主菜单；若系统中有学生信息，系统会显示所有学生的GPA情况并提示按任意键返回主菜单，用户可以按任意键返回主菜单。

⑺选择7进入查询某个学生课程未通过情况功能，用户只需输入相应学号即可。若不存在此学号，系统提示要返回主菜单还是继续输入学号，用户只要根据提示输入相应数字即可。若存在，则系统会输出相应的课程未通过信息，并提示用户要继续查询与否，用户根据提示输入相应字母选择继续查询抑或返回主菜单。

⑻选择8进入查看某门课程的平均分及未通过情况功能，用户先根据系统提示输入相应数字选择要通过课程名称还是课程序号来查询，然后输入相应的课程名称或课程序号，若不存在此课程名称或课程序号，系统提示要返回主菜单还是继续输入课程名称或课程序号，用户只要根据提示输入相应数字即可。若存在，系统会输出相应的信息，并提示用户要继续查询与否，用户根据提示输入相应字母选择继续统计抑或返回主菜单。

⑼选择9退出程序。

编程体会：

这次程设大作业，是我第一次写出上千行的代码，这放在之前肯定会以为能写出这么多很厉害，不过现在自己也写出来了，也就那么回事，原来一点点写，逐层完善，代码行数就会一不小心突破一千。

我做得比较好的我想应该是以下几点：

①界面显示，因为个人对于界面的观赏性比较看重，所以会做得尽量完美。

②对于输入的处理，这方面考虑得比较多，对于用户可能出现的输入错误作了比较完善的考虑，尽量避免因为输入错误造成系统的崩溃。

③还有做得比较好的是注释，作了很多注释，相信一般都可以看懂我的源代码，而且这些注释基本上都是对齐的（可是复制到这里来之后发现格式没了只好重新排了一次），看起来也很方便。

④此外，我用了比较多的文件操作以及动态申请空间，而且都用得挺好的，进一步让我熟悉了这些操作，消除了很多畏惧感。

至于不足之处，我想有以下几点：

①在各个函数之中出现了几处相似度较高的代码段，我于是试着将相同的地方单独作为一个函数，于是成功写出了几个，但还是有一些因为与前后面的代码关联较大，各个函数里面包含了比较特殊的部分，很难单独作为一个函数分开来。不过也许是可以成功的，只是受我能力所限制，需要我在以后的学习生涯中继续学习，这果然需要我活到老学到老啊！

②四个类之间的关系我觉得不是很恰当，可是受迫于一定要有四个类的要求也只能这样，通过交流我发现其他同学也有这样的感觉，但我不知道应该怎么改进比较好。

③虽然用到了运算符重载和多态性，但是用的比较少，因为我没有发现哪些地方是一定要用的，也许有一些地方用了会比不用更方便灵活一点，可是我没有发现这些地方。

④变量名的命名，虽然尽量使得变量名的设置更为易懂，但还是有的命名不是很好，毕竟出现了太多的变量，有点疲于应付。

6. 总结

小学期的程设给了我一个综合运用所学知识的机会，也给我一个查漏补缺的机会。个人感觉相比上学期，程序设计的能力是有进步的。

学生成绩管理系统要求至少要有四个类，一开始思考后我确定了四个类，并确定了彼此之间的关系。不过在写代码时发现这种关系不是很好，所以后来我又稍作修改，调整了这四个类彼此之间的关系。原本我是用一个成绩类作为基类，不过后来想到因为成绩是附属于课程的，所以我改了一下将课程类作为基类。然后才派生出成绩类。想到课程是由学生来上的，我又派生出学生类。而由学生和成绩则构成了学生信息，所以自然派生出信息类。这么多次的程设作业，我清楚地意识到平时要多编程，只有多编程才能增强自己的编程能力。而且平时每写完一次作业都应该要作总结，总结自己的收获和教训，将收获化为自己的知识，将教训作为警戒，避免再犯。而且在写作业之前应该将基础知识融会贯通，用起来方能游刃有余，不然写起作业来很容易丢三落四，写的不好还效率低下。

这次大作业，因为我是逐层写的，写完一部分就会编译一遍，确保无误才会进行下一部分的编写，所以可以将问题及时解决，避免堆积。每次遇到问题，我会先看看是不是低级错误，如果是的话很容易解决，不是的话我会先自己思考问题所在，尽量自己解决，如果实在解决不了我会将错误原因通过百度查找，这确实很有帮助，一般都可以发现问题所在，而且很多其实是我们所不了解的，这是一箭双雕，不仅解决了自己的问题还学习到了新知识。此外，有时候想到一个新的功能，比如我加入的播放音乐功能，就需要用到一些我们之前压根未曾听说过的函数，这时候网络的巨大作用就显示出来了，只要正确搜索都可以找到自己想要的函数，然后根据网上对函数的具体介绍，我们便可以稍作理解然后为己所用。这不失为一种快速获取知识的好方法，在前人的经验上获取知识，汲取前人的知识来充实自己，这不就是人类几千年来得以超越一般动物的根本原因么！同时，这次大作业也提醒我，我到目前为止所学到的程设知识只不过是基础知识，仅仅是这个知识海洋的冰山一角， 还有很多的知识等待我在以后的学习生涯去探索，去学习，可谓活到老学到老。

总之，我的收获就是对于程设，我们不太可能掌握它的所有知识，信息时代的信息爆炸，每天都有很多新知识加入，特别是在编程方面，那我们只好不断学习，该用到哪部分知识就去学习然后化为己用，只有不断学习才能支持我们的编程能力跟上时代的发展。用我反复说到的一句话，就是“活到老学到老”！

附录：源程序清单及评分表

附录 1：源程序清单

```
//头文件 Score_Managment.h 存放各类的声明
#pragma once
#include <iostream>
using namespace std;
//抽象基类 Course 类
class Course
{
public:
  Course(char *cour="NULL",int c=0,int c_n=0,int c_t=0);
//参数初始化表且使用默认参数
  virtual void display() const=0;                    //纯虚函数
  void SetCourse_title(char *cour_nam);              //重新设置课程名称
  void SetCredit(int cred);                          //重新设置课程学分
  void SetCourse_number(int c_n);                    //重新设置课程序号
  void SetCourse_time(int c_t);                      //重新设置课程学时
  char* GetCourse_title();                           //返回课程名称
  int GetCredit();                                   //返回课程学分
  int GetCourse_number();                            //返回课程序号
  int GetCourse_time();                              //返回学时
protected:
  char course_title[10];                             //课程名称
  int credit;                                        //课程学分
  int course_number;                                 //课程序号
  int course_time;                                   //学时
```

```
};

//派生类 Score 类
class Score :virtual public Course
{
public:
  Score(float s = 0, char *cour = "NULL", int c = 0, int c_n = 0, int c_t = 0);                    //参数初始化表且使用默认参数
  virtual void display() const;                    //重定义虚函数
  void SetScore(float ss);                         //重新设置分数
  void SetScore_ranking(int ranking);              //设置分数排名
  float GetScore();                                //返回分数
  char GetRank();                                  //返回分数等级
  int GetScore_ranking();                          //返回分数排名
  char GetPass_or_not();
protected:
  float score;                                     //分数
  char rank;                                       //分数等级
  int score_ranking;                               //分数排名
  char pass_or_not;                                //是否挂科
};

//派生类 Student 类
class Student :virtual public Course      //以公有继承方式继承 Score 类
{
public:
```

```
    Student(char *nam = "NULL", long long num = 0, char *cour = "NULL",
int c = 0, int c_n = 0, int c_t = 0);    //参数初始化表且使用默认参数
    virtual void display() const;        //重定义虚函数
    void SetName(char *nam);             //重新设置姓名
    void SetNumber(long long num);       //重新设置学号
    void SetFall_num(int f_n);           //设置挂科数
    void SetGPA_ranking(int ranking);    //设置 GPA 排名
    char* GetName();                     //返回学生姓名
    long long GetNumber();               //返回学号
    int GetGPA_ranking();                //返回 GPA 排名
    int GetFall_num();                   //返回挂科数
  protected:
    char name[20];                       //学生姓名
    long long number;                    //学号
    int GPA_ranking;                     //GPA 排名
    int fall_num;                        //挂科数
  };
  //多重继承的派生类 Information 类
  class Information :public Score, public Student
  //Course 和 Student 为直接基类
  {
  public:
    Information(char *nam = "NULL", long long num = 0, char *cour ="NULL",
int c_n = 0, int c_t = 0, float s = 0, int c = 0);
  //参数初始化表且使用默认参数
    virtual void display() const;              //重定义虚函数
    void Set_GPA(float gpa);                   //设置 GPA
```

```
    float GetGPA();                          //返回 GPA
    void Set_bool(bool s_g);                 //设置布尔变量 Set_GPA
    bool Get_bool();                         //返回布尔变量 Set_GPA
    void SetFall_course(int f_c);            //设置课程挂科人数
    int GetFall_course();                    //返回课程挂科人数
    void SetAverage_score(float a_s);        //设置课程平均分
    float GetAverage_score();                //返回课程平均分
    friend void Statistics_subject();        //友元函数，统计某科目的成绩情况
//声明重载运算符“>>“函数
     friend istream& operator >>(istream&, Information&);
protected:
    float GPA;                               //学分绩
    bool Set_GPA_or_not;                     //判断此学号是否已经计算过 GPA
    int fall_course;                         //课程挂科人数
    float average_score;                     //课程平均分
};

//头文件 User_Interface.h 存放各种界面和键盘操作函数
#pragma once
#include <iostream>
#include <iomanip>
using namespace std;
void Display_Interface()                     //打印开始界面
{
  cout.fill('*');
  cout <<setw(50)<< "欢迎进入学生成绩管理系统" <<setw(30)<<'*'<< endl;
```

```
    cout.fill(' ');
    cout << setw(30) << "*" << setw(3) << "*" << setw(2) << "*" << setw(6)
<< "*" << setw(2) << "*" << setw(3) << "*" << endl;
    cout << setw(28) << "*" << setw(9) << "*" << setw(2) << "*" << setw(9)
<< "*" << endl;
    cout << setw(27) << "*" << setw(11) << "*" << setw(11) << "*" << endl;
    cout << setw(27) << "*" << setw(22) << "*" << endl;
    cout << setw(27) << "*" << setw(22) << "*" << endl;
    cout << setw(28) << "*" << setw(20) << "*" << endl;
    cout << setw(29) << "*" << setw(18) << "*" << endl;
    cout << setw(30) << "*" << setw(16) << "*" << endl;
    cout << setw(32) << "*" << setw(12) << "*" << endl;
    cout << setw(34) << "*" << setw(8) << "*" << endl;
    cout << setw(36) << "*" << setw(4) << "*" << endl;
    cout << setw(38) << "*" << endl << endl;
}
void SetColor()
//设置背景色及前景色
{
    char choose_background_color[2];              //选择背景色
    char choose_foreground_color[2];              //选择前景色
    bool Judge_background_color = true;           //判断输入正确与否
    bool Judge_foreground_color = true;           //判断输入正确与否
    cout << "请选择您喜欢的背景色及前景色 ^o^ "<<endl;
    cout << "请选择背景色" << endl;
    cout<<"0--黑色，1--蓝色，2--绿色，3--湖蓝色，4--红色，5--紫色，6--黄
色，7--白色，8--灰色，9--淡蓝色：";
```

```
    for (int i = 0; Judge_background_color; i++)
    {
        cin >> choose_background_color;
        if (choose_background_color[0] < '0'||choose_background_color[0]>'9')  //输入有误，重复输入
        {
            cout << "对不起 >o< 输入有误，请输入 0~9 的整数：";
        }
        else                                //输入正确，跳出循环
            Judge_background_color = false;
    }
    cout << "请选择前景色" << endl;
    cout<<"A--淡绿色，B--淡浅绿色，C--淡红色，D--淡紫色，E--淡黄色，F--亮白色：";
    for (int i = 0; Judge_foreground_color; i++)
    {
        cin >> choose_foreground_color;
        if (choose_foreground_color[0] < 'A' || choose_foreground_color[0]>'F')  //输入有误，重复输入
        {
            cout << "对不起 >o< 输入有误，请输入 A~F 的字母：";
        }
        else                                //输入正确，跳出循环
            Judge_foreground_color = false;
    }
    char *color_num_word = new char[10];   //动态申请空间
    memset(color_num_word, 0, sizeof(color_num_word));  //初始化
```

```
    char *num_word = new char[2];            //动态申请空间
    memset(num_word, 0, sizeof(num_word));  //初始化
    num_word[0] = choose_background_color[0];
    num_word[1] = choose_foreground_color[0];
    char *color="color ";
    strcat_s(color_num_word, strlen(color) + 1, color);      //连接字符串
    strcat_s(color_num_word, strlen(color_num_word)+strlen(num_word) + 1,
num_word);
    system(color_num_word);                  //实现背景色及前景色的改变
  }
  //主函数用到的函数的声明
  void Choose_Function();


  //Score_Managment.cpp 存放各类方法的实现及各种函数的定义
  #include <iostream>
  #include <iomanip>
  #include<fstream>
  #include "Score_Managment.h"
  using namespace std;
  //抽象基类 Course 类成员函数定义
  Course::Course(char      *cour,      int      c,int      c_n,      int
c_t ) :credit(c),course_number(c_n),course_time(c_t)
  //参数初始化表初始化数据成员
  {
    strcpy_s(course_title, strlen(cour)+1,cour);
  }
```

```
void Course::SetCourse_title(char *cour_nam)
//重新设置课程名称，cour_nam 为新课程名称
{
  strcpy_s(course_title, strlen(cour_nam)+1,cour_nam);
}
void Course::SetCredit(int cred)  //重新设置课程学分，cred 为新学分
{
  credit = cred;
}
void Course::SetCourse_number(int c_n) //重新设置课程序号，c_n 为新序号
{
  course_number=c_n;
}
void Course::SetCourse_time(int c_t)   //重新设置课程学时，c_t 为新学时
{
  course_time=c_t;
}
char* Course::GetCourse_title()        //返回课程名称
{
  return course_title;
}
int Course::GetCredit()                //返回课程学分
{
  return credit;
}
int Course::GetCourse_number()         //返回课程序号
```

```
{
  return course_number;
}
int Course::GetCourse_time()          //返回学时
{
  return course_time;
}

//派生类 Score 类成员函数定义
Score::Score(float s,char *cour, int c, int c_n, int c_t) : score(s), Course(cour, c, c_n, c_t)                //用参数的初始化表对数据成员初始化
{
  if (s >= 90)
        rank = 'A';
  if (s >= 80 && s< 90)
        rank = 'B';
  if (s >= 60 && s < 80)
        rank = 'C';
  if (s < 60)
        rank = 'D';
}
void Score::display() const
{
  cout << "|" << setw(4) << score << setw(3)<<"|" <<setw(4)<<rank <<setw(3)<<"|" <<setw(4)<<score_ranking<<setw(3)<<"|"<< endl;
}
void Score::SetScore(float ss)            //重新设置分数，ss 为新分数
```

```
{
  score = ss;
  if (ss >= 90)
  {
        rank = 'A';
        pass_or_not = 'Y';
  }
  else if (ss >= 80 && ss < 90)
  {
        rank = 'B';
        pass_or_not = 'Y';
  }
  else if (ss >= 60 && ss < 80)
  {
        rank = 'C';
        pass_or_not = 'Y';
  }
  else if (ss < 60)
  {
        rank = 'D';
        pass_or_not = 'N';
  }
}
float Score::GetScore()                    //返回分数
{
  return score;
```

```
}
char Score::GetRank()                          //返回分数等级
{
  return rank;
}
char Score::GetPass_or_not()
{
  return pass_or_not;
}
void Score::SetScore_ranking(int ranking)  //设置分数排名
{
  score_ranking = ranking;
}
int Score::GetScore_ranking()              //返回分数排名
{
  return score_ranking;
}


//派生类 Student 类成员函数定义
Student::Student(char *nam, long long num, char *cour, int c , int c_n, int c_t) :number(num), Course(cour, c, c_n, c_t)
//参数初始化表初始化数据成员
{
  strcpy_s(name, strlen(nam) + 1, nam);
}
void Student::display() const          //对虚函数进行再定义，为输出函数
```

```
{
  cout<<"|" <<setw(7)<< name <<setw(4)<<"|"<<setw(11)<< number<<" ";
}
void Student::SetName(char *nam)          //重新设置姓名，nam 为新姓名
{
  strcpy_s(name, strlen(nam) + 1, nam);
}
void Student::SetNumber(long long num)  //重新设置学号，num 为新学号
{
  number = num;
}
void Student::SetGPA_ranking(int ranking) //设置 GPA 排名
{
  GPA_ranking = ranking;
}
void Student::SetFall_num(int f_n)
{
  fall_num = f_n;
}
char* Student::GetName()                    //返回学生姓名
{
  return name;
}
long long Student::GetNumber()              //返回学号
{
  return number;
```

```
}
int Student::GetGPA_ranking()              //返回 GPA 排名
{
  return GPA_ranking;
}
int Student::GetFall_num()                 //返回挂科数
{
  return fall_num;
}

//多重继承的派生类 Information 类成员函数定义
Information::Information(char *nam, long long num, char *cour, int c_n,
int c_t, float s, int c) :
Student(nam, num, cour, c, c_n, c_t), Score(s, cour, c, c_n, c_t),
Course(cour, c, c_n, c_t)                  //参数初始化表初始化数据成员
{
  fall_num = 0;
}
void Information::display() const          //对虚函数进行再定义，为输出函数
{
  cout << "|" << setw(7) << name << setw(3)<<"|" << setw(11) << number
<< setw(2)<<"|" << setw(9) << course_title << setw(4)<<"|"<< setw(10) <<
course_number << setw(4)<<"|" << setw(3) << course_time << setw(2)<<"|" <<
setw(4) << score <<setw(3)<< "|" << setw(3) << credit <<setw(3)<<"|"<< endl;
}
void Information::Set_GPA(float gpa)       //设置 GPA
{
  GPA = gpa;
```

```
}
float Information::GetGPA()            //返回 GPA
{
  return GPA;
}
void Information::Set_bool(bool s_g)  //设置布尔变量
{
  Set_GPA_or_not = s_g;
}
bool Information::Get_bool()           //返回布尔变量
{
  return Set_GPA_or_not;
}
void Information::SetFall_course(int f_c)      //设置课程挂科人数
{
  fall_course = f_c;
}
int Information::GetFall_course()              //返回课程挂科人数
{
  return fall_course;
}
void Information::SetAverage_score(float a_s)  //设置课程平均分
{
  average_score = a_s;
}
float Information::GetAverage_score()          //返回课程平均分
```

```
{
  return average_score;
}
//定义重载运算符”>>"函数
istream& operator >>(istream& input, Information& Info)
{
  char nam[20];                //局部变量用以临时输入学生信息并作为实参，
  long long num;               // 将学生信息传给类中成员
  char cour[10];
  int cour_num;
  int cour_tim;
  float sco;
  int cred;
  cout << "请输入学生姓名：";    //输入部分
  input >> nam;
  Info.SetName(nam);
  cout << "请输入学生学号：";
  input >> num;
  Info.SetNumber(num);
  cout << "请输入课程名称：";
  input >> cour;
  Info.SetCourse_title(cour);
  cout << "请输入课程序号：";
  input >> cour_num;
  Info.SetCourse_number(cour_num);
  cout << "请输入课程学时：";
```

```
    input >> cour_tim;
    Info.SetCourse_time(cour_tim);
    cout << "请输入学生分数：";
    input >> sco;
    Info.SetScore(sco);
    cout << "请输入课程学分：";
    input >> cred;
    Info.SetCredit(cred);
    return input;
}

//主函数体中各个函数的定义
//打印输出表头
void Display_title()
{
    cout << "|" << setw(6) << "学生姓名" << setw(2) << "|" << setw(8) << "学号" << setw(5) << "|" << setw(10) << "课程名称" << setw(3) << "|" << setw(11)<< "课程序号" << setw(3) << "|" << setw(3) << "学时" << setw(1) << "|" << setw(5) << "分数" << setw(2) << "|" << setw(3) << "学分" << setw(2) << "|" << endl;
}
//打印功能函数
void Display_Function()
{
    cout.width(47);
    cout << "学生成绩管理系统" << endl;
    cout.width(47);
    cout << "1、录入学生成绩信息" << endl;
```

```
  cout.width(47);
  cout << "2、修改学生成绩信息" << endl;
  cout.width(47);
  cout << "3、查询学生成绩信息" << endl;
  cout.width(51);
  cout << "4、统计某个科目成绩情况" << endl;
  cout.width(46);
  cout << "5、查询某个学生 GPA" << endl;
  cout.width(50);
  cout << "6、统计所有学生 GPA 排名" << endl;
  cout.width(57);
  cout << "7、查询某个学生课程未通过情况" << endl;
  cout.width(63);
  cout << "8、查看某门课程的平均分及未通过情况" << endl;
  cout.width(39);
  cout << "9、退出系统" << endl;
}
//提前引用声明
void Input();
void Modify();
void Search();
void Statistics_subject();
void Search_GPA();
void Statistics_GPA();
void Fall_of_stu();
void Search_aver_fall();
```

```
//选择功能函数
void Choose_Function()
{
  Display_Function();
  int choose_fun;
  cout << "请输入1~9的一个数字以选择一项功能：";
  cin >> choose_fun;
  switch (choose_fun)          //根据用户输入数字调用相应的函数
  {
  case 1:Input(); break;
  case 2:Modify(); break;
  case 3:Search(); break;
  case 4:Statistics_subject(); break;
  case 5:Search_GPA(); break;
  case 6:Statistics_GPA(); break;
  case 7:Fall_of_stu(); break;
  case 8:Search_aver_fall(); break;
  case 9: cout << "谢谢使用！欢迎您再次使用。" << endl; break;
  default:cout << "输入有误！请输入1~9的整数!" << endl;
Choose_Function();                //用户输入有误，重新输入
  }
}
//录入函数
void Input()
{
  int stu_num=0;                  //存放输入学生的人次
  Information *p = new Information; //动态申请空间存放学生成绩信息
```

```
    ofstream outfile("Information.dat", ios::out | ios::app);
  //将学生信息保存在文件里
    if (!outfile)                        //如果打开失败
    {
        cerr << "输入时文件打开出错！" << endl;
        exit(0);
    }
    bool continue_or_not = true;     //布尔变量判断是否继续输入
    for (int i = 0; continue_or_not; i++)            //输入部分
    {
        stu_num++;                   //每执行一次输入则学生人次加 1
        cin >> *p;
        outfile << p->GetName() << " " << p->GetNumber() << " " <<
p->GetCourse_title()  <<  "  "  <<  p->GetCourse_number()  <<  "  "<<
p->GetCourse_time() << " " << p->GetScore() << " " << p->GetRank() << " "
<< p->GetCredit() << " ";
        outfile.clear();
        int choice_input;          //用户输入数字以选择继续或者返回
        cout << "是否继续输入？输入 0 返回主菜单，其他数字键继续输入
~^=^~" << endl;
        cin >> choice_input;
        if (choice_input != 0)
            continue_or_not = true;     //继续输入
        else
            continue_or_not = false;    //返回主菜单
    }
    outfile.close();
    delete p;                    //删除动态申请空间
```

```
    int former_num;              //前一次输入时的学生人次，要求第一次使用时
                                 //已经存在这个文件且其中保存的数字为0
    ifstream infiles("Stu_num.dat", ios::in);
    if (!infiles)                //如果打开失败
    {
        cerr << "输入时打开Stu_num.dat文件以输出原来学生人次出错！" <<
endl;
        exit(0);
    }
    infiles >> former_num;     //将前一次输入时的学生人次赋给变量
    infiles.close();
    ofstream outfiles("Stu_num.dat", ios::out);//将总的学生人次保存到文件
    if (!outfiles)               //如果打开失败
    {
        cerr << "输入时文件打开出错！" << endl;
        exit(0);
    }
    outfiles << (stu_num + former_num);
  //清空文件中原有的内容，将新的总人次保存在文件里
    outfiles.close();
    Choose_Function();         //返回主菜单
  }
  //读取学生人次函数
  void Read_Stu_num(char *process, int &stu_num)
  {
    ifstream infiles("Stu_num.dat", ios::in);   //从文件中读取学生人次
```

```
  if (!infiles)                                    //如果打开失败
  {
        cerr << process<<"时打开文件读入学生人次时出错！" << endl;
        exit(0);
  }
  infiles >> stu_num;                              //读入学生人次
  infiles.close();
}
//保存信息至临时申请空间函数
void Save_to_dynamic(char *process,int stu_num,Information *p)
{
  ifstream infile("Information.dat", ios::in); //从文件中读取学生信息
  if (!infile)                                       //如果打开失败
  {
        cerr << process<<"时文件打开出错！" << endl;
        exit(0);
  }
  char nam[20];                                    //局部变量用以临时输入学生信息
  long long num;
  char cour[10];
  int cour_num;
  int cour_tim;
  float sco;
  char rank;
  int cred;
  for (int i = 0; i < stu_num; i++)
```

```
 //将文件中保存的学生信息先保存到临时申请的空间中
  {
        infile >> nam;
        p[i].SetName(nam);
        infile >> num;
        p[i].SetNumber(num);
        infile >> cour;
        p[i].SetCourse_title(cour);
        infile >> cour_num;
        p[i].SetCourse_number(cour_num);
        infile >> cour_tim;
        p[i].SetCourse_time(cour_tim);
        infile >> sco;
        p[i].SetScore(sco);
        infile >> rank;
        infile >> cred;
        p[i].SetCredit(cred);
  }
  infile.close();
}
//判断继续与否函数
void Continue_or_not(char *process, int &input)
{
  char Yes_or_Not;                //用户选择正确与否时输入的字符
  bool Judge_Yes_or_Not = true;  //布尔变量判断用户输入正确与否
```

```
  for (int i = 0; Judge_Yes_or_Not; i++)
  {
        cout << "是否继续" << process << "? 按 Y 继续，按 N 返回主菜单：";
        cin >> Yes_or_Not;
        if (Yes_or_Not == 'Y' || Yes_or_Not == 'y')        //继续
        {
              Judge_Yes_or_Not = false;
        }
        else if (Yes_or_Not == 'N' || Yes_or_Not == 'n')  //返回主菜单
        {
              input = 0;
              Judge_Yes_or_Not = false;
        }
        else                                                //重新输入
        {
              Judge_Yes_or_Not = true;
        }
  }
}
//修改函数
void Modify()
{
  int stu_num;
  Read_Stu_num("修改", stu_num);
  Information *p = new Information[stu_num];//动态申请空间以存放学生信息
  Save_to_dynamic("修改", stu_num, p);    //读取学生信息到动态申请空间中
```

```
    int choose_nam_or_num;                //存放用户输入数字选项
    bool Input_judge = true;              //布尔变量，判断输入正确与否
    cout << "请选择姓名或学号以修改学生信息，输入 1 代表姓名，2 代表学号：
";
    for (int i = 0; Input_judge; i++) //用户输入数字选择并判断输入合法与否
    {
        cin >> choose_nam_or_num;
        if (!(choose_nam_or_num == 1 || choose_nam_or_num == 2))
            cout << "输入有误，请输入数字 1 或 2 哦=^-^=" << endl;
        else
            Input_judge = false;
    }
    if (choose_nam_or_num == 1)       //按学生姓名为不变量修改
    {
        char nam[20];                 //临时输入学生信息
        long long num;
        char cour[10];
        int cour_num;
        int cour_tim;
        float sco;
        int cred;
        int choose_input_name = 1;
  //系统中无该姓名的学生时用户选择的数字，非 0 代表继续输入学生姓名
        for (int i = 0; choose_input_name; i++)
        {
            cout << "请输入学生姓名：" << endl;
```

```
                cin >> nam;
                int Judge_existence = 0;  //判断该学生是否存在
                for (int j = 0; j < stu_num; j++)
                {
                      if (strcmp(p[j].GetName(), nam) == 0)
    //查找到该学生然后修改信息
                      {
                            Judge_existence++;  //存在该学生则+1
                            cout << "您所要修改的学生原来的信息为：" <<
endl;
                            Display_title();
                            p[j].display();   //显示所要修改学生原来的信息
                            int choose_of_modify;
    //用户输入数字以选择所要修改的信息
                            bool Judge_choose_modify = true;
    //布尔变量判断用户输入正确与否
                            bool Judge_output = true;
    //布尔变量判断是否输出修改后的信息
                            for (int k = 0; Judge_choose_modify; k++)
                            {
                                  cout << "请选择您要修改的学生信息，1 代
表学号，2 代表课程名称，3 代表课程序号，4 代表学时，5 代表分数，6 代表学分，0
代表不修改此条信息：";
                                  cin >> choose_of_modify;
                                  if (choose_of_modify == 0)
    //不修改此条信息，布尔变量为 false，不用重复输入过程
                                  {
```

```
                    Judge_choose_modify = false;
                    Judge_output = false;
//布尔变量为假，不输出修改后的信息，因为并没有修改
                }
                else if (choose_of_modify == 1)
//修改学号，输入正确，故布尔变量为 false，不用重复输入过程
                {
                    Judge_choose_modify = false;
                    cout << "请修改学号：";
                    cin >> num;
                    p[j].SetNumber(num);
                }
                else if (choose_of_modify == 2)
//修改课程名称，输入正确，故布尔变量为 false，不用重复输入过程
                {
                    Judge_choose_modify = false;
                    cout << "请修改课程名称：";
                    cin >> cour;
                    p[j].SetCourse_title(cour);
                }
                else if (choose_of_modify == 3)
//修改课程序号，输入正确，故布尔变量为 false，不用重复输入过程
                {
                    Judge_choose_modify = false;
                    cout << "请修改课程序号：";
                    cin >> cour_num;
```

```
                p[j].SetCourse_number(cour_num);
            }
            else if (choose_of_modify == 4)
//修改课程学时，输入正确，故布尔变量为 false，不用重复输入过程
            {
                Judge_choose_modify = false;
                cout << "请修改课程学时：";
                cin >> cour_tim;
                p[j].SetCourse_time(cour_tim);
            }
            else if (choose_of_modify == 5)
//修改分数，输入正确，故布尔变量为 false，不用重复输入过程
            {
                Judge_choose_modify = false;
                cout << "请修改分数：";
                cin >> sco;
                p[j].SetScore(sco);
            }
            else if (choose_of_modify == 6)
//修改课程学分，输入正确，故布尔变量为 false，不用重复输入过程
            {
                Judge_choose_modify = false;
                cout << "请修改相应课程学分：";
                cin >> cred;
                p[j].SetCredit(cred);
            }
```

```
                                    else
//输入错误，故布尔变量为 true，重复输入过程
                                    {
                                        Judge_choose_modify = true;
                                        cout << "对不起输入有误>o<请重新输入 1~6 的整数："<<endl;
                                    }
                                }
                                if (Judge_output)   //判断是否输出修改后的信息
                                {
                                    cout << "修改后的学生信息为：" << endl;
                                    Display_title();
                                    p[j].display();
                                }
                            }
                        }
                        if (Judge_existence == 0)       //不存在该学生
                        {
                            cout << "对不起>~<不存在该学生，请按 0 返回主菜单或者其他数字键继续输入学生姓名：";
                            cin >> choose_input_name;
                        }
                        else
                        {
                            ofstream outfile("Information.dat", ios::out);
  //将文件中原来保存的信息情清空后再将修改后的学生信息保存在文件里
                            if (!outfile)             //如果打开失败
```

```
                        {
                              cerr << "修改完成后保存时文件打开出错！" <<
endl;
                              exit(0);
                        }
                        for (int i = 0; i < stu_num; i++)  //逐条保存
                        {
                              outfile  <<  p[i].GetName()  <<  "  "  <<
p[i].GetNumber()  <<  "  "  <<  p[i].GetCourse_title()  <<  "  "  <<
p[i].GetCourse_number()  <<  "  "<<  p[i].GetCourse_time()  <<  "  "  <<
p[i].GetScore() << " " << p[i].GetRank() << " " << p[i].GetCredit() << " ";
                              outfile.clear();
                        }
                        outfile.close();           //关闭文件
                        Continue_or_not("修改", choose_input_name);
    //判断是否继续修改
                  }
            }
            delete p;                        //返回主菜单之前删除动态申请空间
            if (choose_input_name == 0)     //返回主菜单
                  Choose_Function();
      }
      else if (choose_nam_or_num == 2)      //按学生学号为不变量修改
      {
            char nam[20];                  //临时输入学生信息
            long long num;
            char cour[10];
            int cour_num;
```

```
            int cour_tim;
            float sco;
            int cred;
            int choose_input_number = 1;
    //系统中无该学号的学生时用户选择的数字，非 0 代表继续输入学生学号
            for (int i = 0; choose_input_number; i++)
            {
                  cout << "请输入学生学号：" << endl;
                  cin >> num;
                  int Judge_existence = 0;
    //变量 Judge_existence 判断该学生是否存在
                  for (int j = 0; j < stu_num; j++)
                  {
                        if (p[j].GetNumber()== num)  //查找到该学生
                        {
                              Judge_existence++;   //存在该学生则+1
                              cout << "您所要修改的学生原来的信息为：" <<
endl;
                              Display_title();
                              p[j].display();  //显示所要修改学生原来的信息
                              int choose_of_modify;
    //用户输入数字以选择所要修改的信息
                              bool Judge_choose_modify = true;
    //布尔变量判断用户输入正确与否
                              bool Judge_output=true;
    //布尔变量判断是否输出修改后的信息
```

```
                    for (int k = 0; Judge_choose_modify; k++)
                    {
                        cout << "请选择您要修改的学生信息，1 代表姓名，2 代表课程名称，3 代表课程序号，4 代表学时，5 代表分数，6 代表学分，0 代表不修改：";
                        cin >> choose_of_modify;
                        if (choose_of_modify == 0)
    //不修改此条信息，布尔变量为 false，不用重复输入过程
                        {
                            Judge_choose_modify = false;
                            Judge_output = false;
    //布尔变量为假，不输出修改后的信息，因为并没有修改
                        }
                        else if (choose_of_modify == 1)
    //修改姓名，输入正确，故布尔变量为 false，不用重复输入过程
                        {
                            Judge_choose_modify = false;
                            cout << "请修改姓名：";
                            cin >> nam;
                            p[j].SetName(nam);
                        }
                        else if (choose_of_modify == 2)
    //修改课程名称，输入正确，故布尔变量为 false，不用重复输入过程
                        {
                            Judge_choose_modify = false;
                            cout << "请修改课程名称：";
                            cin >> cour;
```

```
                p[j].SetCourse_title(cour);
            }
            else if (choose_of_modify == 3)
//修改课程序号，输入正确，故布尔变量为 false，不用重复输入过程
            {
                Judge_choose_modify = false;
                cout << "请修改课程序号：";
                cin >> cour_num;
                p[j].SetCourse_number(cour_num);
            }
            else if (choose_of_modify == 4)
//修改课程学时，输入正确，故布尔变量为 false，不用重复输入过程
            {
                Judge_choose_modify = false;
                cout << "请修改课程学时：";
                cin >> cour_tim;
                p[j].SetCourse_time(cour_tim);
            }
            else if (choose_of_modify == 5)
//修改分数，输入正确，故布尔变量为 false，不用重复输入过程
            {
                Judge_choose_modify = false;
                cout << "请修改分数：";
                cin >> sco;
                p[j].SetScore(sco);
            }
```

```
                                else if (choose_of_modify == 6)
    //修改课程学分，输入正确，故布尔变量为 false，不用重复输入过程
                                {
                                      Judge_choose_modify = false;
                                      cout << "请修改相应课程学分：";
                                      cin >> cred;
                                      p[j].SetCredit(cred);
                                }
                                else
    //输入错误，故布尔变量为 true，重复输入过程
                                {
                                      Judge_choose_modify = true;
                                      cout << "对不起，输入有误 >o< 请重新输入 1~6 的整数：" << endl;
                                }
                          }
                          if (Judge_output)  //判断是否输出修改后的信息
                          {
                                cout << "修改后的学生信息为：" << endl;
                                Display_title();
                                p[j].display();
                          }
                    }
              }
              if (Judge_existence == 0)        //不存在该学生
              {
```

```
                    cout << "对不起 >~< 不存在该学生，请按 0 返回主菜单或者其他数字键继续输入学生学号：";
                    cin >> choose_input_number;
                }
                else
                {
                    ofstream outfile("Information.dat", ios::out); //将文件中原来保存的信息清空后再将修改后的学生信息保存在文件里
                    if (!outfile)           //如果打开失败
                    {
                        cerr << "修改完成后保存时文件打开出错！" << endl;
                        exit(0);
                    }
                    for (int i = 0; i < stu_num; i++)   //逐条保存
                    {
                        outfile << p[i].GetName() << " " << p[i].GetNumber() << " " << p[i].GetCourse_title() << " " << p[i].GetCourse_number() << " "<< p[i].GetCourse_time() << " " << p[i].GetScore() << " " << p[i].GetRank() << " " << p[i].GetCredit() << " ";
                        outfile.clear();
                    }
                    outfile.close();        //关闭文件
                    Continue_or_not("修改", choose_input_number);
                   //判断是否继续修改
                }
            }
            delete p;  //返回主菜单之前删除动态申请空间
            if (choose_input_number == 0)  //返回主菜单
```

```
            Choose_Function();
        }
    }
    //查询函数
    void Search()
    {
        int stu_num;
        Read_Stu_num("查询", stu_num);              //读取学生人次
        Information *p = new Information[stu_num];//动态申请空间以存放学生信息
        Save_to_dynamic("查询", stu_num, p);      //读取学生信息到动态申请空间
        int choose_of_search;                       //存放用户输入数字选项
        bool Input_judge = true;                    //布尔变量，判断输入正确与否
        cout << "请选择姓名或学号以查询学生信息，输入 1 代表姓名，2 代表学号：
";
        for (int i = 0; Input_judge; i++) //用户输入数字选择并判断输入合法与否
        {
            cin >> choose_of_search;
            if (!(choose_of_search == 1 || choose_of_search == 2))
           //输入错误，重复输入
                cout << "输入有误，请输入数字 1 或 2 哦 =^-^= " << endl;
            else                          //输入正确跳出循环
                Input_judge = false;
        }
        if (choose_of_search == 1)        //按学生姓名查询
        {
            char nam[20];                 //局部变量用以临时输入学生姓名
```

```
        int *pp = new int[stu_num];
    //动态申请空间以存放查找到的学生在对象数组中的位置
        int choose_input_name = 1;
    //系统中无该姓名的学生时用户选择的数字，非 0 代表继续输入学生姓名
        for (int i = 0; choose_input_name; i++)
        {
            cout << "请输入学生姓名：" << endl;
            cin >> nam;
            int Judge_existence = 0;
    //变量 Judge_existence 判断该学生是否存在
            for (int j = 0; j < stu_num; j++)
            {
                if (strcmp(p[j].GetName(), nam) == 0)
    //查找到该学生然后输出信息
                {
                    Judge_existence++;    //存在该学生则+1
                    pp[Judge_existence - 1] = j;
    //存放查找到的学生在对象数组中的位置
                }
            }
            if (Judge_existence == 0)       //不存在该学生
            {
                cout << "对不起 >~< 不存在该学生，请按 0 返回主菜单或者其他数字键继续输入学生姓名：";
                cin >> choose_input_name;
            }
```

```
                else
                {
                        int i, j, k, t;
                        for (i = 0; i < Judge_existence - 1; i++)
                       //用选择法排序
                        {
                                k = i;
                                for (j = i + 1; j < Judge_existence; j++)
                                {
                                        if        (p[pp[j]].GetScore()        <
p[pp[k]].GetScore())                  //按分数从高到低排序
                                                k = j;
                                        else    if    (p[pp[j]].GetScore()    ==
p[pp[k]].GetScore())                  //分数相同时按课程序号从小到大排序
                                                if (p[pp[j]].GetCourse_number() <
p[pp[k]].GetCourse_number())
                                                        k = j;
                                }
                                t = pp[k];
                                pp[k] = pp[i];
                                pp[i] = t;
                        }
                        cout << "您所要查询的学生的信息为：" << endl;
                        Display_title();
                        for (int i = 0; i < Judge_existence; i++)
                                p[pp[i]].display();    //输出学生信息
                        Continue_or_not("查询", choose_input_name);
```

```
//判断是否继续查询
            }
        }
        delete pp;              //删除动态申请空间
        delete p;               //返回主菜单之前删除动态申请空间
        if (choose_input_name == 0)
            Choose_Function();      //返回主菜单
    }
    else if (choose_of_search == 2)     //按学生学号查询
    {
        long long num;          //局部变量用以临时输入学生学号
    int *pp = new int[stu_num]; //动态申请空间以存放查找到的学生在对
                                //象数组中的位置
    int choose_input_number = 1;  //系统中无该学号的学生时用户选择的
                                  //数字，非0代表继续输入学生学号
        for (int i = 0; choose_input_number; i++)
        {
            cout << "请输入学生学号：" << endl;
            cin >> num;
            int Judge_existence = 0;      //判断该学生是否存在
            for (int j = 0; j < stu_num; j++)
            {
                if (p[j].GetNumber() == num) //查找到该学生
                {
                    Judge_existence++;    //存在该学生则+1
                    pp[Judge_existence - 1] = j;
```

```
//存放查找到的学生在对象数组中的位置
                    }
                }
                if (Judge_existence == 0)       //不存在该学生
                {
                    cout << "对不起 >~< 不存在该学生，请按 0 返回主菜单或者其他数字键继续输入学生学号：";
                    cin >> choose_input_number;
                }
                else
                {
                    int i, j, k, t;
                    for (i = 0; i < Judge_existence - 1; i++)
                     //用选择法排序
                    {
                        k = i;
                        for (j = i + 1; j < Judge_existence; j++)
                        {
                            if (p[pp[j]].GetScore() < p[pp[k]].GetScore())              //按分数从低到高排序
                                k = j;
                            else if (p[pp[j]].GetScore() == p[pp[k]].GetScore())            //分数相同的按课程号从小到大排序
                                if (p[pp[j]].GetCourse_number() < p[pp[k]].GetCourse_number())
                                    k = j;
                        }
                        t = pp[k];
```

```
                        pp[k] = pp[i];
                        pp[i] = t;
                    }
                    cout << "您所要查询的学生的信息为：" << endl;
                    Display_title();
                    for (int i = 0; i < Judge_existence; i++)
                        p[pp[i]].display();   //输出学生信息
                    Continue_or_not("查询", choose_input_number);
//判断是否继续查询
                }
            }
            delete pp;              //删除动态申请空间
            delete p;               //返回主菜单之前删除动态申请空间
            if (choose_input_number == 0)    //返回主菜单
                Choose_Function();
        }
}
//统计科目成绩情况函数
void Statistics_subject()
{
    int stu_num;
    Read_Stu_num("统计科目成绩情况", stu_num);              //读取学生人次
    Information *p = new Information[stu_num];//动态申请空间以存放学生信息
    Save_to_dynamic("统计科目成绩情况", stu_num, p);
                                            //保存学生信息到动态申请空间
    int choose_title_number;                //存放用户选择的数字
```

```
    bool Judge_title_number = true;          //判断用户输入的正确性
    cout << "请问您想通过课程名称还是课程序号来统计科目的成绩情况？" << endl;
    for (int i = 0; Judge_title_number; i++)
    {
        cout << "按 1 代表课程名称，按 2 代表课程序号，请输入：";
        cin >> choose_title_number;
        if (choose_title_number == 1 || choose_title_number == 2)
  //输入正确，跳出循环
            Judge_title_number = false;
        else            //输入有误，继续输入
        {
            cout << "输入有误 >o< 请重新输入哦" << endl;
        }
    }
    if (choose_title_number == 1)
    {
        int *pp = new int[stu_num];
  //动态申请空间以存放查找到的课程名称对应的在对象数组中的位置
        int choose_input_title = 1;
  //系统中无该课程名称时用户选择的数字，非 0 代表继续输入课程名称
        for (int i = 0; choose_input_title; i++)
        {
            char cour[10];
            cout << "请输入您要统计的科目：";
            cin >> cour;
```

```
int Judge_existence = 0;          //判断该课程名称是否存在
for (int j = 0; j < stu_num; j++)
{
      if (strcmp(p[j].GetCourse_title(), cour) == 0)
     //查找到该课程名称
      {
            Judge_existence++;  //存在该课程名称则+1
            pp[Judge_existence - 1] = j;
//存放查找到的课程名称对应的在对象数组中的位置
      }
}
if (Judge_existence == 0)       //不存在该课程名称
{
      cout << "对不起 >~< 不存在该课程名称，请按 0 返回主菜单或者其他数字键继续输入课程名称：";
      cin >> choose_input_title;
}
else
{
      int i, j, k, t;
      for (i = 0; i < Judge_existence - 1; i++)
       //用选择法排序
      {
            k = i;
            for (j = i + 1; j < Judge_existence; j++)
            {
```

```
                    if (p[pp[j]].GetScore() > p[pp[k]].GetScore())          //按分数从高到低排序
                        k = j;
                    else if (p[pp[j]].GetScore() == p[pp[k]].GetScore())          //分数相同则按学号从小到大排序
                        if (p[pp[j]].GetNumber() < p[pp[k]].GetNumber())
                            k = j;
                }
                t = pp[k];
                pp[k] = pp[i];
                pp[i] = t;
            }
            for (i = 0; i < Judge_existence; i++)//设置分数排名
            {
                p[pp[0]].SetScore_ranking(1);
//第一个分数最高，排名设置为 1
                if (i >= 1)
                {
                    if (p[pp[i]].GetScore() == p[pp[i - 1]].GetScore())          //分数相同则排名相同
                        p[pp[i]].SetScore_ranking(p[pp[i - 1]].GetScore_ranking());
                    else
                        p[pp[i]].SetScore_ranking(i + 1);
                }
            }
            int choose_output;   //存放用户选择的输出方式
            bool Judge_choose_output = true;
```

```
    //布尔变量判断用户输入正确与否
                        for (i = 0; Judge_choose_output; i++)
                        {
                              cout << "请问您想要哪种输出方式，1 代表只输出分数、等级及排名，2 代表同时还输出学生姓名及学号：";
                              cin >> choose_output;
                              if (choose_output == 1 || choose_output == 2)
//输入正确，跳出循环
                                    Judge_choose_output = false;
                              else
                                    cout << "输入有误 >o< 请重新输入数字 1 或 2 哦" << endl;
                        }
                        if (choose_output == 1)
                        {
                              Score *output_score = new Score[Judge_existence];
                                    //动态申请空间，返回 Score 类指针
                              cout << "| " << setw(3) << "分数" << " | " << setw(2) << "等级" << " | " << setw(3) << "排名 |" << endl;
                              for (i = 0; i < Judge_existence; i++)
                              {
                                    output_score[i] = p[pp[i]];
                                    output_score[i].display();
//调用 Score 类的 display()函数
                              }
                              delete output_score;     //删除申请的空间
                        }
                        else if (choose_output == 2)
                        {
```

```
                        Score       *output_score       =       new
Score[Judge_existence];                 //动态申请空间，返回 Score 类指针

                        Student      *output_student      =      new
Student[Judge_existence];               //动态申请空间，返回 Student 类指针

                        cout << "| 学生姓名 |" << "    学号    | " <<
setw(3) << "分数" << " | " << setw(2) << "等级" << " | " << setw(3) << "排
名 |" << endl;

                        for (i = 0; i < Judge_existence; i++)

                        {

                              output_score[i] = p[pp[i]];

                              output_student[i] = p[pp[i]];

                              output_student[i].display();
//调用 Score 类的 display()函数

                              output_score[i].display();
//调用 Student 类的 display()函数

                        }

                  }

                  Continue_or_not(" 统  计 ",   choose_input_title);
//判断是否继续统计

            }

      }

      delete pp;                    //删除动态申请空间

      delete p;                     //返回主菜单之前删除动态申请空间

      if (choose_input_title == 0)

            Choose_Function();      //返回主菜单

}

if (choose_title_number == 2)

{

      int *pp = new int[stu_num];
```

```
//动态申请空间以存放查找到的课程序号对应的在对象数组中的位置
        int choose_input_number = 1;
//系统中无该课程序号时用户选择的数字，非 0 代表继续输入课程序号
        for (int i = 0; choose_input_number; i++)
        {
              int cour_num;
              cout << "请输入您要统计的科目的课程序号：";
              cin >> cour_num;
              int Judge_existence = 0;
//变量 Judge_existence 判断该课程序号是否存在
              for (int j = 0; j < stu_num; j++)
              {
                    if (p[j].GetCourse_number() == cour_num)
                   //查找到该课程序号
                    {
                          Judge_existence++;        //存在该课程名称则+1
                          pp[Judge_existence - 1] = j;
//存放查找到的课程序号对应的在对象数组中的位置
                    }
              }
              if (Judge_existence == 0)             //不存在该课程序号
              {
                    cout << "对不起 >~< 不存在该课程序号，请按 0 返回主菜单或者其他数字键继续输入课程序号：";
                    cin >> choose_input_number;
              }
```

```
                else
                {
                    int i, j, k, t;
                    for (i = 0; i < Judge_existence - 1; i++)
                    //用选择法排序
                    {
                        k = i;
                        for (j = i + 1; j < Judge_existence; j++)
                        {
                            if (p[pp[j]].GetScore() > p[pp[k]].GetScore())  //按分数从高到低排序
                                k = j;
                            else if (p[pp[j]].GetScore() == p[pp[k]].GetScore())  //分数相同的按学号从小到大排序
                                if (p[pp[j]].GetNumber() < p[pp[k]].GetNumber())
                                    k = j;
                        }
                        t = pp[k];
                        pp[k] = pp[i];
                        pp[i] = t;
                    }
                    for (i = 0; i < Judge_existence; i++) //设置分数排名
                    {
                        p[pp[0]].SetScore_ranking(1);
    //第一个分数最高，排名设置为 1
                        if (i >= 1)
```

```
                {
                    if (p[pp[i]].GetScore() == p[pp[i - 1]].GetScore())                    //分数相同则排名相同
                        p[pp[i]].SetScore_ranking(p[pp[i - 1]].GetScore_ranking());
                    else
                        p[pp[i]].SetScore_ranking(i + 1);
                }
            }
            int choose_output;   //存放用户选择的输出方式
            bool Judge_choose_output = true;
    //布尔变量判断用户输入正确与否
            for (i = 0; Judge_choose_output; i++)
            {
                cout << "请问您想要哪种输出方式，1 代表只输出分数、等级及排名，2 代表同时还输出学生姓名及学号：";
                cin >> choose_output;
                if (choose_output == 1 || choose_output == 2)
//输入正确，跳出循环
                    Judge_choose_output = false;
                else
                    cout <<"输入有误 >o< 请重新输入数字 1 或 2 哦" << endl;
            }
            if (choose_output == 1)
            {
                Score *output_score = new Score[Judge_existence];          //动态申请空间，返回 Score 类指针
                cout << "| " << setw(3) << "分数" << " | " <<
```

```
setw(2) << "等级" << " | " << setw(3) << "排名 |" << endl;
                        for (i = 0; i < Judge_existence; i++)
                        {
                              output_score[i] = p[pp[i]];
                              output_score[i].display();
//调用 Score 类的 display()函数
                        }
                        delete output_score;  //删除动态申请空间
                  }
                  else if (choose_output == 2)
                  {
                        Score        *output_score        =        new
Score[Judge_existence];                //动态申请空间，返回 Score 类指针
                        Student       *output_student       =       new
Student[Judge_existence];              //动态申请空间，返回 Student 类指针
                        cout << "| 学生姓名 |" << "    学号    | " <<
setw(3) << "分数" << " | " << setw(2) << "等级" << " | " << setw(3) << "排
名 |" << endl;
                        for (i = 0; i < Judge_existence; i++)
                        {
                              output_score[i] = p[pp[i]];
                              output_student[i] = p[pp[i]];
                              output_student[i].display();
//调用 Score 类的 display()函数
                              output_score[i].display();
//调用 Student 类的 display()函数
                        }
                  }
                  Continue_or_not(" 统 计 ",   choose_input_number);
```

```
//判断继续统计与否
                }
            }
            delete pp;                          //删除动态申请空间
            delete p;                           //返回主菜单之前删除动态申请空间
            if (choose_input_number == 0)
                Choose_Function();     //返回主菜单
      }
    }
    //计算 GPA 函数
    int Calculate_GPA(int stu_num, Information *p)
    {
      int *n = new int[stu_num];          //动态申请空间
      int time = 0;                       //此学号出现次数
      int num_of_stu = 0;                 //学生人数
      for (int i = 0; i < stu_num; i++)
      {
            time = 0;                     //每次都要初始化为 0
            float total_score = 0;        //总成绩初始化为 0
            int total_credit = 0;         //总学分初始为 0
            if (p[i].Get_bool())          //如果此学号之前没有计算过 GPA
            {
                  num_of_stu++;           //学生人数+1
                  for (int j = i; j < stu_num; j++)
                  {
                        if (p[j].GetNumber() == p[i].GetNumber())
```

```
    //找到此学号
                        {
                              total_score                                    +=
p[j].GetScore()*p[j].GetCredit();
                              total_credit += p[j].GetCredit();//计算总学分
                              n[time] = j; //存放该学号对应的在数组中的位置
                              time++;       //出现次数+1
                              p[j].Set_bool(false);
    //布尔变量设置为假，下次不会重复计算这个学号
                        }
                  }
                  p[n[0]].Set_bool(true);   //第一个的布尔变量设置为真
                  for (int k = 0; k < time; k++)    //设置 GPA
                  {
                        p[n[k]].Set_GPA(total_score / total_credit);
                  }
            }
            else                              //如果此学号之前已经计算过 GPA
            {
                  for (int j = i; j >= 0; j--)
                  {
                        if (p[j].GetNumber() == p[i].GetNumber())
    //找到此学号
                        {
                              p[i].Set_GPA(p[j].GetGPA());
//学号相同的 GPA 设置为相同
                        }
```

```
            }
        }
    }
    delete n;                       //删除动态申请空间
    return num_of_stu;              //返回学生人数，不计重复
}
//查询 GPA 函数
void Search_GPA()
{
    int stu_num;
    Read_Stu_num("查询 GPA", stu_num);        //读取学生人次
    Information *p = new Information[stu_num];//动态申请空间以存放学生信息
    Save_to_dynamic("查询 GPA", stu_num, p); //保存学生信息至动态申请空间
    bool Judge_title_number = true;          //判断用户输入的正确性
    Calculate_GPA(stu_num, p);               //计算 GPA
    long long numb;                          //用户输入学号
    bool times = true;                       //判断是否输出了一次
    bool input_num = true;                   //判断是否存在此学号
    int Judge_num = 1;                       //用户选择继续还是返回主菜单
    for (int i = 0; Judge_num; i++)
    {
        cout << "请输入您要查询的学号：";
        cin >> numb;
        int j;
        for (j = 0,times=true; j<stu_num&&times; j++)
        {
```

```
                if (p[j].GetNumber() == numb) //找到该学号，只输出一次
                {
                    cout << "查询结果为：" << endl;
                    cout << "|学生姓名|" << "     学号     |" << "  GPA
|"  << endl;
                    cout << "|  " << setw(4) << p[j].GetName() << "  | "
<< setw(10) << p[j].GetNumber() << " |" << setw(7) << p[j].GetGPA() << "|"
<< endl;
                    times = false;
                }
            }
            if (j==stu_num&&times==true)       //不存在该学号
            {
                cout << "对不起 >o< 系统中没有这个学号，请按 0 返回主菜单或
其他数字键继续输入：";
                cin >> Judge_num;
            }
            else                                //存在该学号
            {
                cout << "是否继续查询？按 0 返回主菜单，按其他数字键继续查
询：";
                cin >>Judge_num;
            }
        }
        delete p;                               //删除动态申请空间
        if (Judge_num == 0)
            Choose_Function();                  //返回主菜单
    }
```

```
//统计 GPA 排名函数
void Statistics_GPA()
{
  int stu_num;
  Read_Stu_num("统计 GPA 排名", stu_num);     //读取学生人次
  Information *p = new Information[stu_num];//动态申请空间以存放学生信息
  Save_to_dynamic("统计 GPA 排名", stu_num, p);
//将学生信息保存至动态申请空间
  bool Judge_title_number = true;          //判断用户输入的正确性
  int num_of_stu=Calculate_GPA(stu_num, p); //计算 GPA 并返回学生人数
  int *pp = new int[num_of_stu];             //动态申请空间
  for (int i = 0,j=0; i < stu_num; i++)
  {
        if (p[i].Get_bool())
        {
              pp[j] = i;         //存放用来排序的学号对应的在数组中的位置
              j++;
        }
  }
  int i, j, k,t;
  for (i = 0; i < num_of_stu - 1; i++)
  {
        k = i;
        for (j = i + 1; j < num_of_stu; j++)
        {
              if (p[pp[j]].GetGPA() > p[pp[k]].GetGPA())
```

```
    //按 GPA 从高到低排序
                        k = j;
                else    if    (p[pp[j]].GetGPA()    ==    p[pp[k]].GetGPA())
//GPA 相同的按学号从小到大排序
                        if (p[pp[j]].GetNumber() < p[pp[k]].GetNumber())
                            k = j;
            }
            t = pp[k];
            pp[k] = pp[i];
            pp[i] = t;
        }
        for (i = 0; i < num_of_stu; i++)        //设置 GPA 排名
        {
            p[pp[0]].SetGPA_ranking(1);        //第一个 GPA 最高，排名设置为 1
            if (i >= 1)
            {
                if (p[pp[i]].GetGPA() == p[pp[i - 1]].GetGPA())
    //GPA 相同则排名相同
                        p[pp[i]].SetGPA_ranking(p[pp[i                        -
1]].GetGPA_ranking());
                else
                        p[pp[i]].SetGPA_ranking(i + 1);
            }
        }
        if (num_of_stu > 0)          //系统中有学生信息
        {
            cout << "GPA 排名如下：" << endl;
```

```
            cout <<  "|"<<setw(6)<<" 学生姓名"<<setw(2)<<"|" <<setw(8)<< "学号"<<setw(5)<<"|" << setw(5) << "GPA" << setw(3)<<"|" << setw(8) << "GPA 排名"<<setw(2)<<"|" << endl;

            for (i = 0; i < num_of_stu; i++)

            {

                  cout <<  "|" << setw(7) << p[pp[i]].GetName() <<setw(4)<< "|" << setw(11) << p[pp[i]].GetNumber() <<setw(2)<< "|" << setw(7) << p[pp[i]].GetGPA()    <<    "|"    <<setw(5)<<    p[pp[i]].GetGPA_ranking() <<setw(5)<<"|"<< endl;

            }

      }

      else                              //系统中无学生信息

            cout << "对不起 >o< 系统中暂无学生信息" << endl;

      cout << "请按任意键返回主菜单 ^.^:";

      delete pp;                        //删除动态申请空间

      delete p;                         //删除动态申请空间

      if (getchar() != -1)

      {

            getchar();

            Choose_Function();   //返回主菜单

      }

   }

   //查询某个学生未通过课程情况函数

   void Fall_of_stu()

   {

      int stu_num;

      Read_Stu_num("统计 GPA 排名", stu_num);     //读取学生人次

      Information *p = new Information[stu_num];//动态申请空间以存放学生信息
```

```
    Save_to_dynamic("统计 GPA 排名", stu_num, p);     //保存至动态申请空间
    long long num;                  //局部变量用以临时输入学生学号
    int *pp = new int[stu_num];  //存放查找到的学生在对象数组中的位置
  int choose_input_number = 1; //系统中无该学号的学生时用户选择的数字，
                                 //非 0 代表继续输入学生名称
    for (int i = 0; choose_input_number; i++)
    {
          cout << "请输入学生学号：" << endl;
          cin >> num;
          int Judge_existence = 0;//变量 Judge_existence 判断该学生是否存在
          for (int j = 0; j < stu_num; j++)
          {
                if (p[j].GetNumber() == num)  //查找到该学生
                {
                      Judge_existence++;      //存在该学生则+1
                      pp[Judge_existence - 1] = j;
  //存放查找到的学生在对象数组中的位置
                }
          }
          if (Judge_existence == 0)           //不存在该学生
          {
                cout << "对不起 >~< 不存在该学生，请按 0 返回主菜单或者其他
数字键继续输入学生学号：";
                cin >> choose_input_number;
          }
          else
```

```
        {
            cout << "您所查询的学生是：" << p[pp[0]].GetName() << "(学号：" << p[pp[0]].GetNumber() << ")" << endl;
            int i, j, k, t, no_pass = 0;
            for (int i = 0; i < Judge_existence; i++)//计算未通过课程数
            {
                if (p[pp[i]].GetPass_or_not() == 'N')
                {
                    pp[no_pass] = pp[i];
//记录该名学生未通过课程对应的在数组中的位置
                    no_pass++;
                }
            }
            if (no_pass > 0)                        //有未通过课程
            {
                for (i = 0; i < no_pass; i++)
                {
                    p[pp[i]].SetFail_num(no_pass);
                }
                for (i = 0; i < no_pass - 1; i++)  //用选择法排序
                {
                    k = i;
                    for (j = i + 1; j < no_pass; j++)
                    {
                        if (p[pp[j]].GetScore() > p[pp[k]].GetScore())  //按分数从高到低排序
```

```
                                        k = j;
                                    else    if    (p[pp[j]].GetScore()    ==
p[pp[k]].GetScore())                          //分数相同的按课程号从小到大排序
                                        if (p[pp[j]].GetCourse_number() <
p[pp[k]].GetCourse_number())
                                            k = j;
                                }
                                t = pp[k];
                                pp[k] = pp[i];
                                pp[i] = t;
                            }
                            cout << "一共有" << p[pp[0]].GetFall_num() << "门课
程未通过，具体情况如下：" << endl;
                            cout << "|" << setw(7) << "课程名称" << setw(2) <<
"|" << setw(5) << "分数" << setw(2) << "|" << endl;
                            for (int i = 0; i < no_pass; i++)
                            {
                                cout     <<     "|"     <<     setw(8)     <<
p[pp[i]].GetCourse_title()   <<   setw(2)   <<   "|"   <<   setw(5)   <<
p[pp[i]].GetScore() << setw(2) << "|" << endl;
                            }
                            Continue_or_not(" 查 询 ",   choose_input_number);
//判断是否继续查询
                        }
                        else                //没有未通过课程
                        {
                            cout << "该名学生没有未通过课程记录 ^_^ " << endl;
                            cout << "请按 0 返回主菜单或其他数字键继续输入学号：
";
```

```
                cin >> choose_input_number;
            }
        }
    }
    delete pp;                          //删除动态申请空间
    delete p;                           //返回主菜单之前删除动态申请空间
    if (choose_input_number == 0)  //返回主菜单
        Choose_Function();
}
//通过课程名称计算平均分函数
float Average_score(char *title, int stu_num, Information *p)
{
    float total_score = 0;              //总分
    int total_stu = 0;                  //总人数
    for (int i = 0; i < stu_num; i++)
    {
        if (strcmp(p[i].GetCourse_title(), title) == 0)
        {
            total_score += p[i].GetScore();//将其中分数相加得到总分
            total_stu++;            //总人数加 1
        }
    }
    if (total_stu != 0)                 //存在该课程名
        return (total_score / total_stu);           //返回平均分
    else                                            //不存在该课程名
        return -1;                                  //返回-1
```

```
}
//通过课程序号计算平均分函数
float Average_score(int cour_num, int stu_num, Information *p)
{
  float total_score = 0;                          //总分
  int total_stu = 0;                              //总人数
  for (int i = 0; i < stu_num; i++)
  {
        if (p[i].GetCourse_number() == cour_num)
        {
              total_score += p[i].GetScore();//将其中分数相加得到总分
              total_stu++;                        //总人数加 1
        }
  }
  if (total_stu != 0)                             //存在该课程名
        return (total_score / total_stu);         //返回平均分
  else                                            //不存在该课程名
        return -1;                                //返回-1
}
//查询课程平均分及未通过情况函数
void Search_aver_fall()
{
  int stu_num;
  Read_Stu_num("统计 GPA 排名", stu_num);    //读取学生人次
  Information *p = new Information[stu_num];//动态申请空间以存放学生信息
  Save_to_dynamic("统计 GPA 排名", stu_num, p);
```

```
    //将学生信息保存至动态申请空间
      int *pp = new int[stu_num];
    //动态申请空间以存放查找到的课程在对象数组中的位置
      int choose_title_number;
      bool Judge_title_number = true;
      for (int i = 0; Judge_title_number; i++)
      {
            cout << "请问您想要通过课程名称或者是课程序号查询？输入 1 表示课程名称，输入 2 表示课程序号:";
            cin >> choose_title_number;
            if (choose_title_number == 1 || choose_title_number == 2)
                  Judge_title_number = false;
      }
      if (choose_title_number == 1)
      {
            char title[10];                    //局部变量用以临时输入课程名称
            int choose_input_title = 1;
    //系统中无该课程名称时用户选择的数字，非 0 代表继续输入课程名称
            for (int i = 0; choose_input_title; i++)
            {
                  cout << "请输入课程名称：" ;
                  cin >> title;
                  int Judge_existence = 0;   //判断该课程是否存在
                  for (int j = 0; j < stu_num; j++)
                  {
                        if  (strcmp(p[j].GetCourse_title(),  title)  ==  0)
//查找到该课程
```

```
                {
                    Judge_existence++;      //存在该课程则+1
                    pp[Judge_existence - 1] = j;
//存放查找到的课程在对象数组中的位置
                }
            }
            if (Judge_existence == 0)           //不存在该课程
            {
                cout << "对不起 >~< 不存在该课程名称，请按 0 返回主菜单或者其他数字键继续输入课程名称：";
                cin >> choose_input_title;
            }
            else
            {
                cout << "这门课程--" << p[pp[0]].GetCourse_title() << "的平均分为：" << Average_score(title, stu_num, p) << endl;
                int i, j, k, t, no_pass = 0;
                for (int i = 0; i < Judge_existence; i++)
//计算未通过课程数
                {
                    if (p[pp[i]].GetPass_or_not() == 'N')
                    {
                        pp[no_pass] = pp[i];
//记录该课程未通过的学生对应的在数组中的位置
                        no_pass++;
                    }
                }
```

```
                    if (no_pass > 0)                    //有未通过课程
                    {
                        for (i = 0; i < no_pass; i++)
                        {
                            p[pp[i]].SetFall_course(no_pass);
                        }
                        for (i = 0; i < no_pass - 1; i++)
                        //用选择法排序
                        {
                            k = i;
                            for (j = i + 1; j < no_pass; j++)
                            {
                                if (p[pp[j]].GetScore() > p[pp[k]].GetScore())        //按分数从高到低排序
                                    k = j;
                                else if (p[pp[j]].GetScore() == p[pp[k]].GetScore())        //分数相同的按学号从小到大排序
                                    if (p[pp[j]].GetNumber() < p[pp[k]].GetNumber())
                                        k = j;
                            }
                            t = pp[k];
                            pp[k] = pp[i];
                            pp[i] = t;
                        }
                        cout << "一共有" << p[pp[0]].GetFall_course() << "个学生未通过，具体情况如下：" << endl;
                        cout << "|" << setw(8) << "学生姓名" << setw(2)
```

```
<< "|" << setw(8) << "学号" << setw(5) << "|" << setw(5) << "分数" << setw(2) << "|" << endl;
                        for (int i = 0; i < no_pass; i++)
                        {
                            cout << "|" << setw(8) << p[pp[i]].GetName() << setw(2) << "|" << setw(11) << p[pp[i]].GetNumber() << setw(2) << "|" << setw(4) << p[pp[i]].GetScore() << setw(3) << "|" << endl;
                        }
                        Continue_or_not("查询", choose_input_title); //判断是否继续查询
                    }
                    else                        //没有未通过课程
                    {
                        cout << "该门课程没有未通过学生 ^_^ " << endl;
                        cout << "请按 0 返回主菜单或其他数字键继续输入课程名称：";
                        cin >> choose_input_title;
                    }
                }
            }
            delete pp;                  //删除动态申请空间
            delete p;                   //返回主菜单之前删除动态申请空间
            if (choose_input_title == 0)    //返回主菜单
                Choose_Function();
        }
        else if (choose_title_number == 2)
        {
            int cour_num;               //局部变量用以临时输入课程序号
```

```
        int choose_input_number = 1;
    //系统中无该课程序号时用户选择的数字，非0代表继续输入课程序号
        for (int i = 0; choose_input_number; i++)
        {
            cout << "请输入课程序号：";
            cin >> cour_num;
            int Judge_existence = 0;
    //变量Judge_existence判断该课程是否存在
            for (int j = 0; j < stu_num; j++)
            {
                if (p[j].GetCourse_number() == cour_num)
    //查找到该课程
                {
                    Judge_existence++;        //存在该课程则量+1
                    pp[Judge_existence - 1] = j;
    //存放查找到的课程在对象数组中的位置
                }
            }
            if (Judge_existence == 0)           //不存在该课程
            {
                cout << "对不起 >~< 不存在该课程序号，请按0返回主菜单或者其他数字键继续输入课程序号：";
                cin >> choose_input_number;
            }
            else
            {
```

```
                        cout << "这门课程--" << p[pp[0]].GetCourse_title()
<< "的平均分为：" << Average_score(cour_num, stu_num, p) << endl;
                        int i, j, k, t, no_pass = 0;
                        for (int i = 0; i < Judge_existence; i++)
    //计算未通过课程数
                        {
                              if (p[pp[i]].GetPass_or_not() == 'N')
                              {
                                    pp[no_pass] = pp[i];
    //记录该课程未通过的学生对应的在数组中的位置
                                    no_pass++;
                              }
                        }
                        if (no_pass > 0)            //有未通过课程
                        {
                              for (i = 0; i < no_pass; i++)
                              {
                                    p[pp[i]].SetFail_course(no_pass);
                              }
                              for (i = 0; i < no_pass - 1; i++)
                             //用选择法排序
                              {
                                    k = i;
                                    for (j = i + 1; j < no_pass; j++)
                                    {
                                          if     (p[pp[j]].GetScore()     >
p[pp[k]].GetScore())       //按分数从高到低排序
```

```
                                        k = j;
                                    else if (p[pp[j]].GetScore() ==
p[pp[k]].GetScore())      //分数相同的按学号从小到大排序
                                        if (p[pp[j]].GetNumber() <
p[pp[k]].GetNumber())
                                            k = j;
                                }
                                t = pp[k];
                                pp[k] = pp[i];
                                pp[i] = t;
                            }
                            cout << "一共有" << p[pp[0]].GetFall_course()
<< "个学生未通过，具体情况如下：" << endl;
                            cout << "|" << setw(8) << "学生姓名" << setw(2)
<< "|" << setw(8) << "学号" << setw(5) << "|" << setw(5) << "分数" << setw(2)
<< "|" << endl;
                            for (int i = 0; i < no_pass; i++)
                            {
                                cout << "|" << setw(8) <<
p[pp[i]].GetName() << setw(2) << "|" << setw(11) << p[pp[i]].GetNumber() <<
setw(2) << "|" << setw(4) << p[pp[i]].GetScore() << setw(3) << "|" << endl;
                            }
                            Continue_or_not("查询", choose_input_number);
//判断是否继续查询
                        }
                        else                    //没有未通过课程
                        {
                            cout << "该门课程没有未通过学生 ^_^ " << endl;
                            cout << "请按 0 返回主菜单或其他数字键继续输入
课程序号：";
```

```
                cin >> choose_input_number;
            }
        }
    }
    delete pp;              //删除动态申请空间
    delete p;               //返回主菜单之前删除动态申请空间
    if (choose_input_number == 0)    //返回主菜单
        Choose_Function();
  }
}


//主程序文件 main.cpp
#include <iostream>
#include <Windows.h>
#pragma comment(lib,"winmm.lib")
#include "Score_Managment.h"
#include "User_Interface.h"
using namespace std;
//主函数
int main()
{
  PlaySound(TEXT(" 天 空 之 城 .wav"),  NULL,  SND_FILENAME  |
SND_ASYNC|SND_LOOP);          //为使得使用者有好的使用心情，循环播放音乐
  SetColor();               //设置背景色及前景色
  Display_Interface();      //打印封面
  Choose_Function();        //选择功能
```

```
    cout << endl;
    return 0;
}
```

附录 2：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	**3**	
设计与调试结果	**4**	
设计说明书的质量	**1**	
程序基本要求涵盖情况	**4**	
程序代码编写素养情况	**2**	
课程设计周表现情况	**1**	
综合成绩	**15**	

教师签名：_

日　　期：_