

# 程 序 设 计 报 告

课程名称 计算机程序设计基础 2

班 级 无 28  
学 号 2022010722  
姓 名 赵子恒

2023 年 7 月 20 日

# 目 录

<b>1 设计内容与设计要求</b>	<b>3</b>
1.1 课程设计目的	3
1.2 课题题目	3
1.3 文档设计要求	3
1.4 程序设计的基本要求	3
<b>2 系统需求分析</b>	<b>4</b>
<b>3 总体设计</b>	<b>5</b>
<b>4 详细设计</b>	<b>6</b>
4.1 数据结构设计	6
4.2 case1() 设计	7
4.3 case2() 设计	7
4.4 case3() 设计	8
4.5 case4() 设计	9
4.6 case5() 设计	9
<b>5 系统调试</b>	<b>10</b>
<b>6 测试结果与分析</b>	<b>11</b>
6.1 测试用数据	11
6.2 菜单界面	12
6.3 case1() 测试图	13
6.4 case2() 测试图	14
6.5 case3() 测试图	16
6.6 case4() 测试图	18
6.7 case5() 测试图	19
<b>7 总结</b>	<b>21</b>
<b>附录 1：源程序清单</b>	<b>22</b>
<b>附录 2：时间安排</b>	<b>55</b>
<b>评价表</b>	<b>55</b>

# 1 设计内容与设计要求

## 1.1 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。

## 1.2 课题题目

学生成绩管理系统（或公司人事管理系统）

## 1.3 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。

## 1.4 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。

## 2 系统需求分析

学生成绩管理系统中记录着学生的成绩情况，包括学生各门课程的等级，总体排名，绩点，各学期均绩等信息。同时，我们设计的学生成绩管理系统应该考虑到实际使用中，学生存在重名，因此还包含学号作为学生身份的唯一标识。该成绩管理系统也同时被老师和学生使用，因此并不能支持简单的查询，还要具有录入，修改和删除记录的功能。因此，系统的主要功能有以下六条：

1. 录入学生成绩：作为成绩管理系统，录入学生成绩是必不可少的功能。该系统应该可以通过键盘，用学生的姓名和学号确定唯一的学生并将新的课程记录录入到该学生的数据库中。同时在系统开启时，系统应有能力从存储文件中读取数据并将其和新输出的数据一起重新排序；在系统关闭时，录入的成绩和学生的姓名学号信息应该可以被储存回文件中。
2. 修改学生成绩：由于老师录入学生成绩时有可能出错，同时考虑到学生每学期都有 pf 课程的机会，该系统应该具有修改已录入数据的能力。但为了管理方便，记录中可以被修改的部分应该受到限制。同时，为了防止无法修改严重的课程录入错误，该系统应当具有删除录入数据的功能。
3. 查询学生成绩单：该系统应该可以使学生得知自己的成绩单。因此应当具有查询并输出指定学生所有课程及其绩点的能力。
4. 查询学生均绩：成绩管理系统应当使用学生录入的成绩信息计算出学生的绩点。同时应当同时提供查询单人的绩点和查询数据库中所有人绩点并给出排名的能力。
5. 查询课程的均绩：学生的成绩本身是给教师教学的反馈。因此该系统应具有根据曾上过这节课的所有人的分数计算出课程的历史平均绩点，从而为授课老师提供参考。同时，为了让学生更好的了解课程情况，也应该生成课程均绩的排名。
6. 本系统应该以菜单方式工作，这意味着使用者可以自由选择需要使用的功能。同时，本系统应当注意细节，提供良好的输入引导和恰当的输入错误警告。本系统同时应该具有良好的鲁棒性，这意味着该系统不应因为用户的错误输入导致系统崩溃或者输出错误的结果，而应该提醒用户的错误并准备重新输入。最后，本系统的设计应该符合正常的使用习惯，尽可能地为使用者提供便利。

### 3 总体设计

本系统一共有录入学生成绩，修改学生成绩，查询学生成绩单，查询学生均绩，查询课程的均绩五个主要功能。输入的学生成绩信息包括学生姓名，学生学号，课程绩点，如果输入绩点是 4.0 需要额外指定课程等级，课程名称，课程所在学期，课程是否是 pf 计分或被记为 pf 计分。

在本系统所有要求输入的地方，系统内部都会检查读入是否正常，如果出现了不符合输入要求的输入，将会触发 `typewrong()` 函数，要求用户重新输入，从而增加了系统的鲁棒性。当整个类型执行完毕后，用户可以选择继续进行该操作或者返回菜单页面，使得用户的操作更加连续，也减少了输出，使得整体上更为美观。

由于白色字体过于单调，本系统参考清华大学的绩点查询 App THU\_info，不同绩点的课程输出时的颜色不同，这使得用户可以更方便的了解课程的绩点。同时，当系统检测到输入错误或者向用户发出警告时输出的字是红色的，操作成功时输出的字是绿色的，这更加醒目，使得用户的使用更加便利。

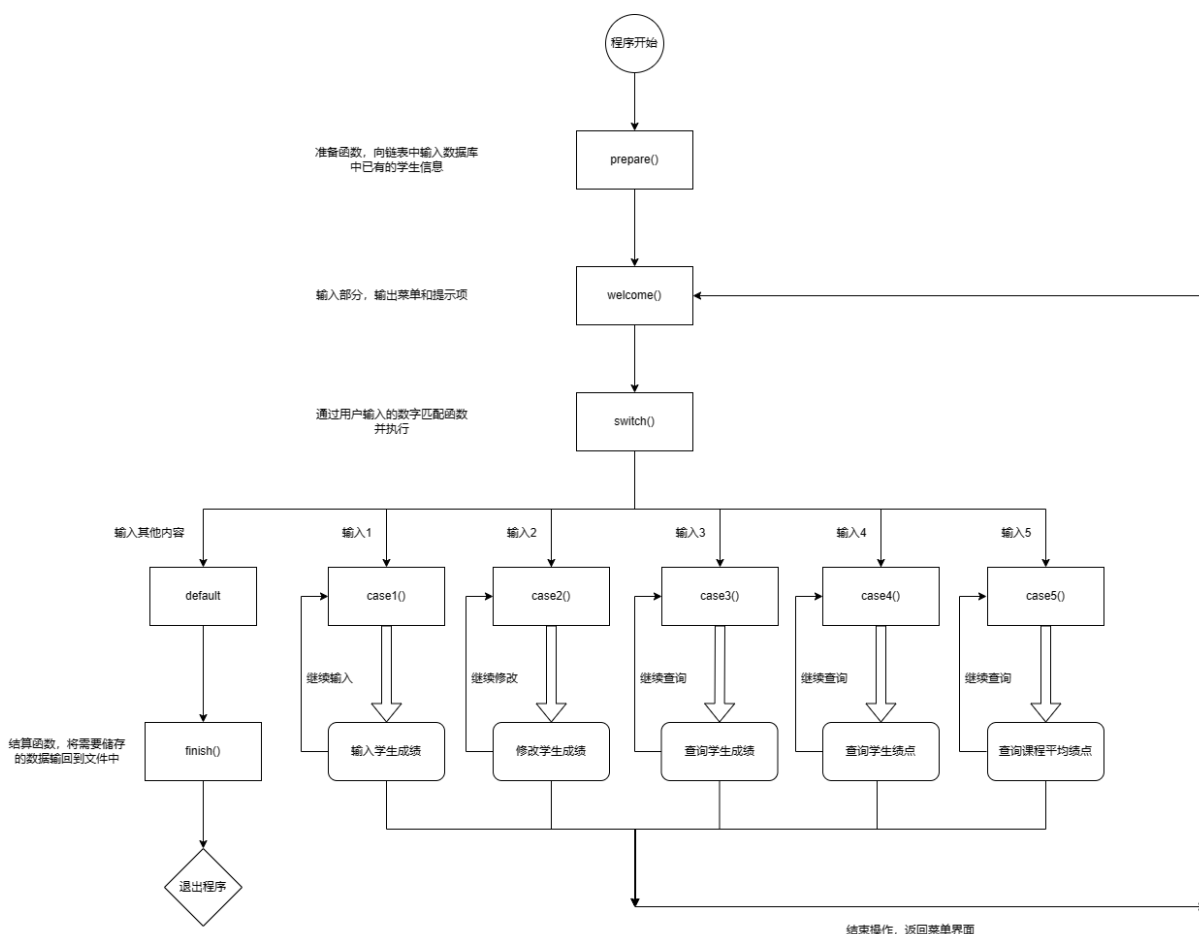


图 1: 学生管理系统的流程图

在使用流程方面，当程序开始运行时，首先启动 `prepare()` 函数，将文档中的数据加入到链表中，从而实现数据的连续性，防止重复输入相同数据或数据与先前数据冲突。数据导入完成后，`welcome()`

函数输出菜单，pipei() 函数读入用户的输入并匹配到下边具体的函数中去执行。如果想要退出，可以在菜单页面输入除了选择操作的数字外的任何字符。这将会启动 finish() 函数，将链表中的数据和其他储存在全局变量中的需保存数据储到文件中，等待下次启动时读入。最后，当 finish() 函数执行完毕，basic 类析构函数中的检查将会被激活，如果链表中的总数据与输入到文件中的数据条数不符，该检查将会输出一条警告，提醒用户有的数据可能丢失。

## 4 详细设计

### 4.1 数据结构设计

在数据结构方面，本系统主要由四个类和两个结构体组成。basic 类是做基类的抽象类；people 类储存着键盘输入和文件读入的数据信息，Node 类则继承 basic 类和 people 类，作为链表的节点。LinkedList 类则继承 basic 类，通过操作 Node 形成链表。每个类之间都是公有继承。student 和 course 则是结构体成员，用于统计链表中的数据，计算学生的绩点等等。在设计中通过让 Node 类同时共有继承 basic 类和 people 类体现了 C++ 的多继承，同时通过 basic 中的纯虚函数 wrong() 体现了 C++ 的多态性。每个类都至少有四个字段和四个方法，满足了大作业的要求。

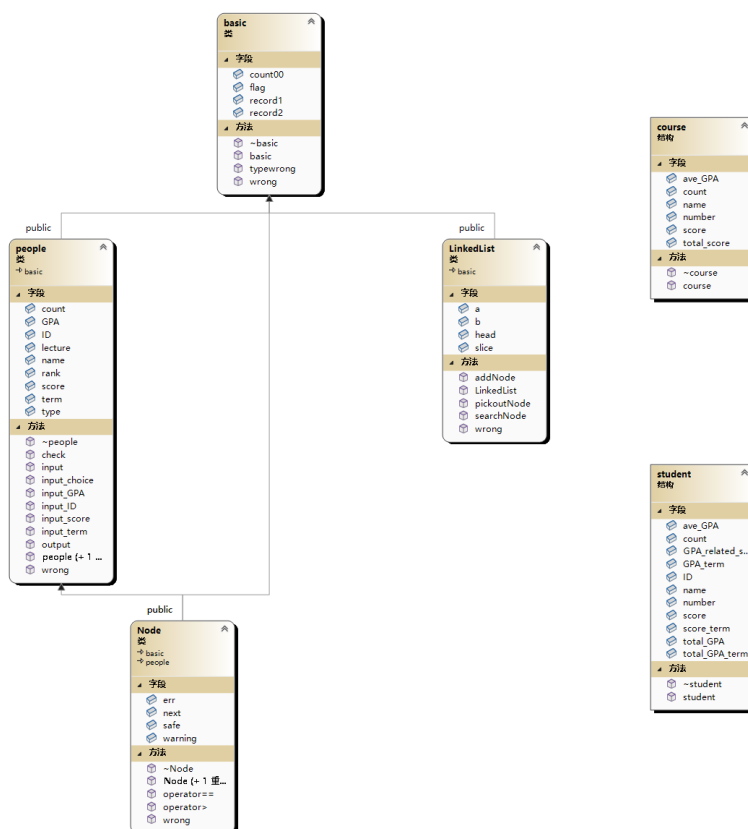


图 2: 学生管理系统的类视图

用于储存学生信息的结构体中主要有学生姓名，学生学号，学生总学分，学生总学分绩，学生绩点相关学分，学生学期内学分，学生学期内平均绩点，学生绩点和学生学期内总学分绩这 9 个数据以及辅助其他函数实现的学生人数计数和学生报的课程数两个数据。用于储存课程信息的结构体中则有课程名，课程学分数，课程报过的人数，报过该课程的总人数，课程的总学分绩和课程的平均绩点。这两个结构体并不含有特殊的方法，主要的作用就是在链表中储存数据。

## 4.2 case1() 设计

case1() 的作用是将用户从键盘输入的数据存储到链表中。在这一过程中，同时进行的还有链表中数据的排序，数据的统计与计算绩点。case1() 会调用全局函数 shuru()，创建一个 people 类临时变量 a，再调用全局变量 total 的成员函数 addNode 将其加入链表。

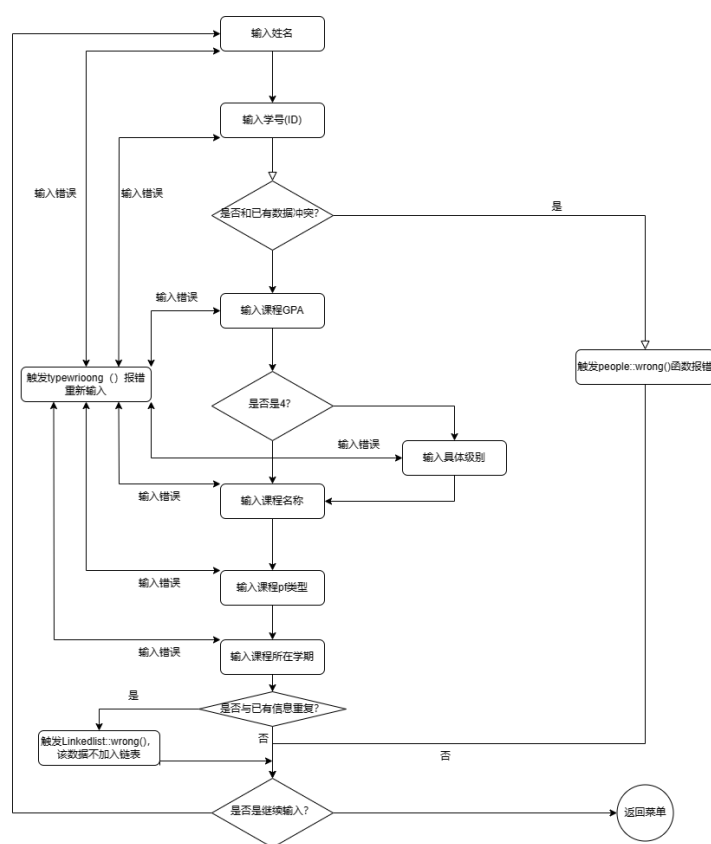


图 3: case1 的流程图

## 4.3 case2() 设计

case2() 的作用是根据输入的学生姓名和学号以及学期找到该学生在特定学期的所有课程记录，然后选择想要更改的记录，并从四种更改模式中选择一种更改数据，最后询问用户是否继续更改，如果不

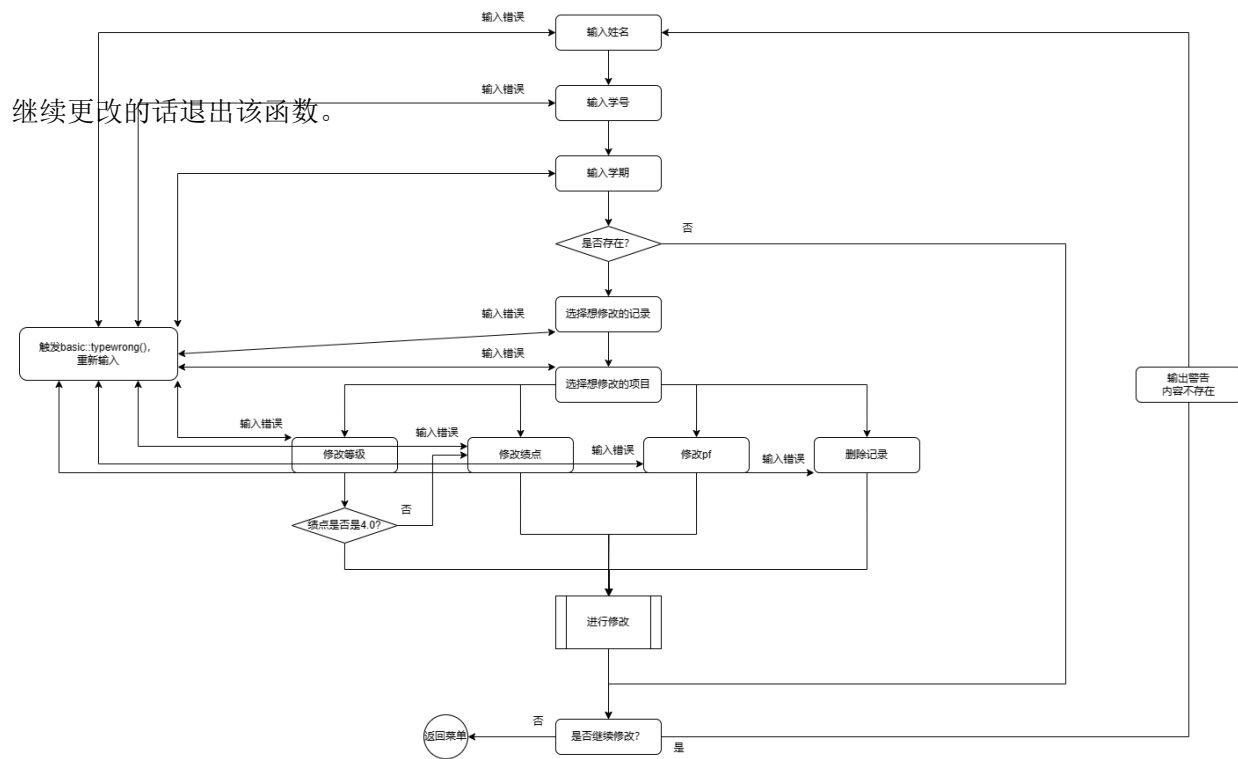


图 4: case2 的流程图

#### 4.4 case3() 设计

case3() 的作用是在输入学生的姓名和学号后输出链表中符合该信息的所有信息，相当于输出该学生上过的所有课程及其学分、绩点、等级、是否记为 pf 等信息，即打印出该学生的成绩单。

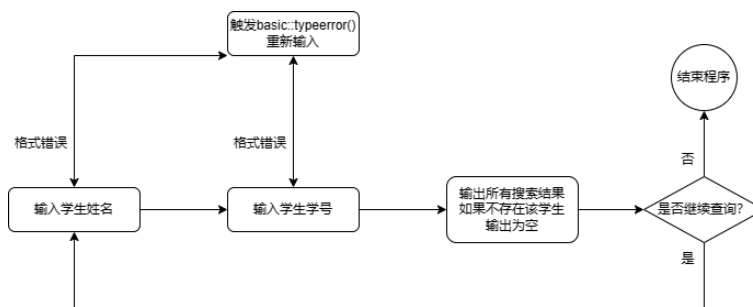


图 5: case3 的流程图



## 4.5 case4() 设计

case4() 的作用是查询学生的绩点。这一函数有两种选择，如果输入“查询全部学生”将会按绩点从高到低输出所有学生的总绩点，也可以只输入一个学生的姓名和学号，从而查询这个学生的总绩点和每学期的平均绩点。

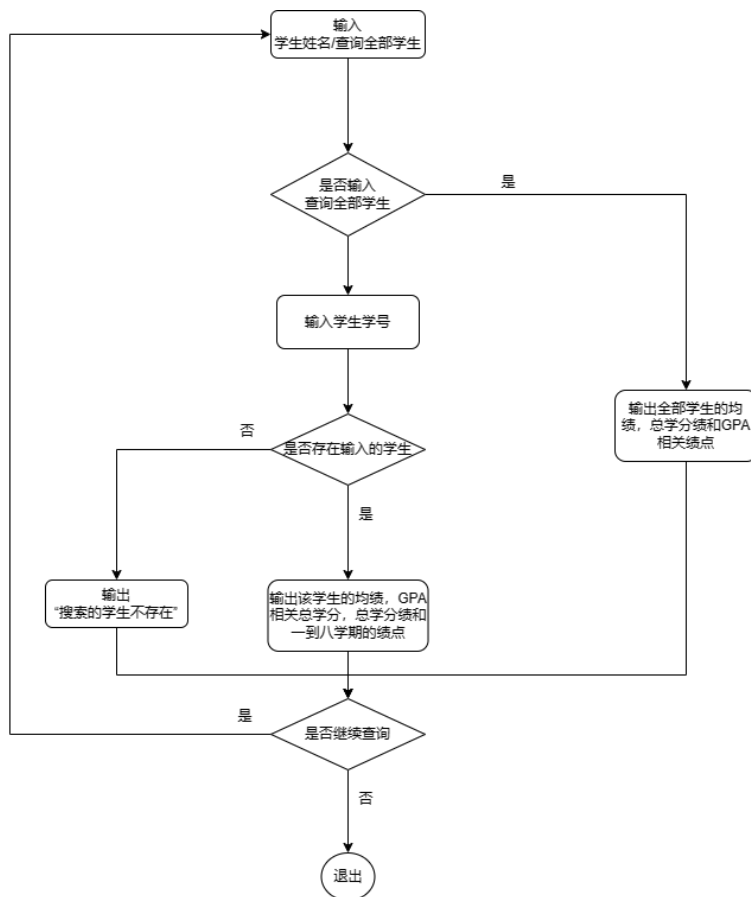


图 6: case4 的流程图

## 4.6 case5() 设计

case5() 的作用是查询课程的平均绩点，这一数值是上过该课的所有学生的绩点的平均值。这一值可以在一定程度上反应课程的给分和教学情况。用户可以输入课程名来查询特定的课程，也可以直接输入“查询所有课程”，看到课程的平均绩点排名。如果系统没有找到该课程，将会输出“未查到该课程”提醒用户。

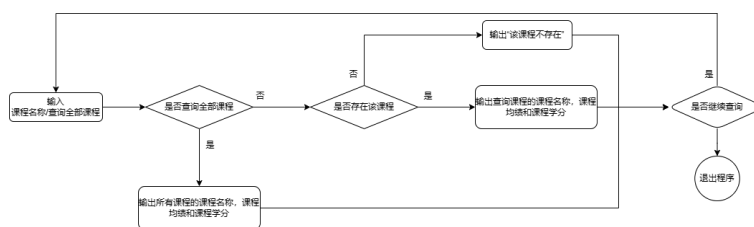


图 7: case5 的流程图

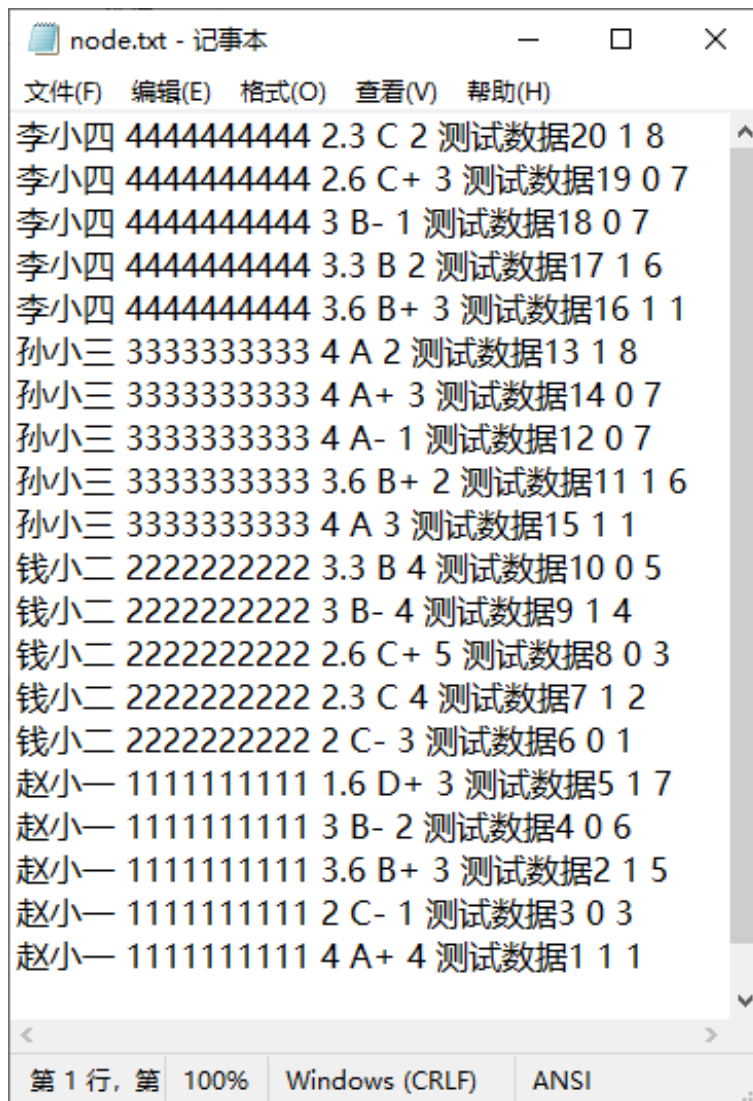
## 5 系统调试

我在写程序的过程中, 选择的是写完一个模块测试一个模块的做法。这种做法的好处是可以使得整体上更加清晰, 不会出现全部写完后测试一团乱麻, 不知道问题出在哪里。但缺点是在修改比较基础的东西时有可能影响已经测试好的模块, 导致其失效, 所以每过一段时间就需要确定之前能正常实现的功能是否还能正常实现。调试过程中, 首先出现的问题是输入意料之外的数据时, 程序可能会卡死。比如程序希望读入的是一个 `int` 型数据, 但输入了一个字符串。为了解决这一问题, 我在每一个输入后面都加入了 `cin.good()` 的检查, 一旦读入错误, 就会马上清除错误状态并忽略输入的错误数据。同时, 为了防止过量的数据堆在缓存区里, 导致下一次输入读入上一次输入的数据, 每次输入后都会无视缓存区中的剩余数据, 同时在数据容易堆积的函数中加入了清楚缓存区。在调试过程中, 我发现每次输入完成后都要返回菜单带来的使用感极差, 所以我加入了询问是否要继续输入的环节, 这提高了使用的流畅度, 改善了输入的手感。在运行中, 我发现我想要赋值的变量并没有被正确的赋值。在多次检查无果后, 我选择向学长求助, 结果发现是由于我的继承较为混乱, 程序实际上将值赋给了继承来的对象, 而非我希望赋值的对象。这也让我开始重新检查我的数据结构。在重新调整了数据结构后, 我解决了这一问题。我还在测试时发现, 有的输入明显与事实不符, 但仍然可以输入成功, 比如两条相同学号但姓名不同的数据, 为此我增加了一个数组来储存学生的姓名与学号的对应关系, 同时将这一数据输入到文件中, 这解决了可以录入学号相同但姓名不同数据的问题。随着文件存储功能的加入, 我发现了更多的 bug。我在文件存储中将文件中数据写入链表的函数调用的是从键盘输入的同一个函数, 这个函数在输入成功后会输出“输入成功”提醒用户, 这就使得在文件读入阶段, 菜单还未出现时屏幕上就输出了“输入成功”, 导致观感很差。我在考虑后增加了一个 `bool` 型变量来检测从文件读入过程是否完成, 如果未完成的话, 调用加入链表的函数不会输出“输入成功”的提示。我在设计把数据输出到文件的函数中设计了一个检查程序, 在输出函数完成后, 如果记录的链表中数据和实际上输出到文件的数据不相符, 将会报错, 提醒用户可能发生了数据丢失。这个检查过程和输出过程位于 `basic` 类的析构函数中, 同时有条件判断确保该输出只会在链表中的数据被全部输入文件后才会起到作用。调试过程很成功, 在正常流程下, 该程序确实提醒了用户可能的数据丢失。但如果突然终止调试的话, 虽然链表中的数据也没有被全部输入文件, 但程序并不会提醒用户。这就导致如果突然关闭对话框的话, 用户无法得知可能的数据丢失, 并没有完成我设计时的全部目标。同时, 我在基本完成作业后才发现 `string` 类数据非常便捷好用, 但我没有使用, 这也是我的一个遗憾。但我在整个系统调试的过程中还是学到了很多, 一些之前看不懂的报错现在也能看明白了, 最大的收获应该是我学会了熟练使用互联网或者向学长提问来获得答案, 也知道了如何正确的提出问题。

## 6 测试结果与分析

### 6.1 测试用数据

该程序使用的测试数据如下：



```
node.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
李小四 4444444444 2.3 C 2 测试数据20 1 8
李小四 4444444444 2.6 C+ 3 测试数据19 0 7
李小四 4444444444 3 B- 1 测试数据18 0 7
李小四 4444444444 3.3 B 2 测试数据17 1 6
李小四 4444444444 3.6 B+ 3 测试数据16 1 1
孙小三 3333333333 4 A 2 测试数据13 1 8
孙小三 3333333333 4 A+ 3 测试数据14 0 7
孙小三 3333333333 4 A- 1 测试数据12 0 7
孙小三 3333333333 3.6 B+ 2 测试数据11 1 6
孙小三 3333333333 4 A 3 测试数据15 1 1
钱小二 2222222222 3.3 B 4 测试数据10 0 5
钱小二 2222222222 3 B- 4 测试数据9 1 4
钱小二 2222222222 2.6 C+ 5 测试数据8 0 3
钱小二 2222222222 2.3 C 4 测试数据7 1 2
钱小二 2222222222 2 C- 3 测试数据6 0 1
赵小一 1111111111 1.6 D+ 3 测试数据5 1 7
赵小一 1111111111 3 B- 2 测试数据4 0 6
赵小一 1111111111 3.6 B+ 3 测试数据2 1 5
赵小一 1111111111 2 C- 1 测试数据3 0 3
赵小一 1111111111 4 A+ 4 测试数据1 1 1
```

图 8: 测试用学生信息

值得注意的是，测试数据是输入进文本文件的，但学生的学号信息是由该管理系统自动生成的。同时，学号信息中出现了几条数据并未在测试数据中出现。这是因为这些数据在测试完毕后被手动删除。并不是由于系统的故障产生的。

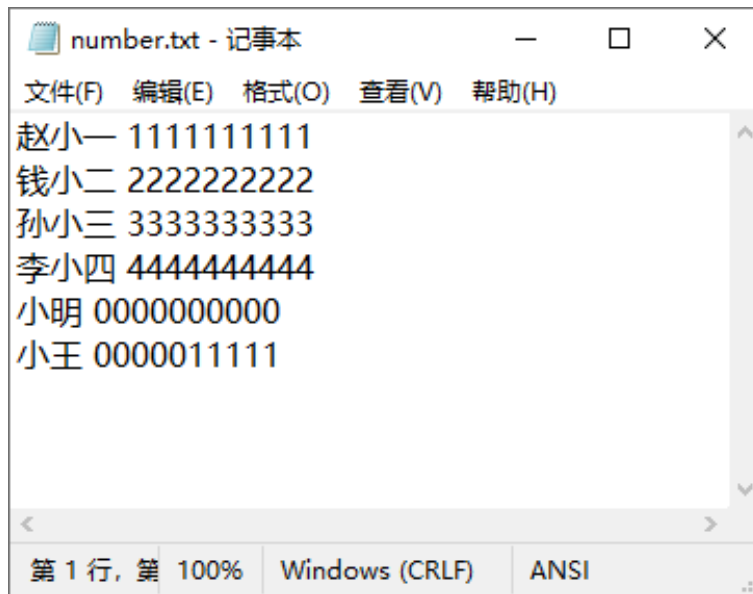


图 9: 学生学号信息

## 6.2 菜单界面

正常打开程序后，程序将会自动从文件中录入数据，然后显示菜单页面。

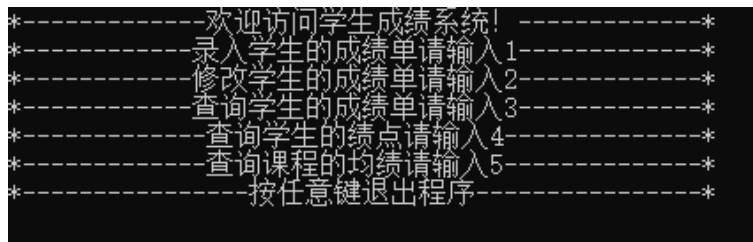


图 10: 菜单页面

在菜单界面输入指定的数字将会进入相应功能，输入其他字符将会退出程序。

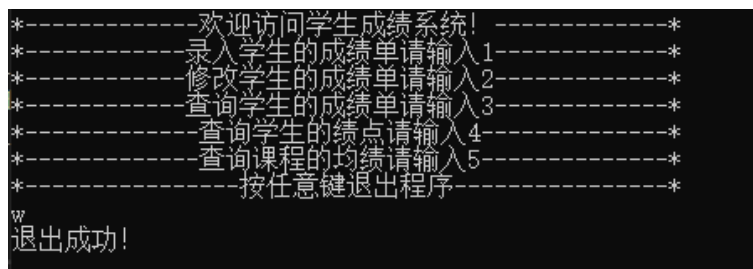


图 11: 退出

### 6.3 case1() 测试图

正常流程输入下，case1 的输入流程不会报错，在所有输入结束后，系统将会输出绿色的“输入成功”，提示用户信息已成功录入，参见图12。

```
请输入学生姓名
小明
请输入学生学号(十位)
0000000000
请输入该课程GPA, 退课GPA请输入-1, pf请输入0
4
请输入学生等级
A+
请输入课程学分(整数)
2
请输入课程名称
测试课程0
请输入课程类型, pf课程请输入0, 非pf课程请输入1
1
请输入课程学期名称, 大一上为1, 最高为8(大四下)
3
输入成功!
是否继续输入? (y/n)
```

图 12: case1 正常输入

如果学生的姓名和学号不匹配，将会触发通用类型错误，系统将会输出红色字体提醒用户，参见图13。

```
请输入学生姓名
小明
请输入学生学号(十位)
2222222222
error:通用类型错误! 您的输入可能与事实不符或者与已有数据冲突, 请检查后重新输入!
常见类型: 输入的GPA不存在, 学生等级与绩点不匹配, 输入学分数值过大或过小, 输入格式错误, 姓名与学号和已录入的数据不符...
是否继续输入? (y/n)
```

图 13: 学生姓名与学号不匹配时触发错误

如果输入的数据和系统希望得到的数据类型不相符，也将触发通用类型错误，系统将会输出红色字体提醒用户，参见图14。

```
请输入学生姓名
小王
请输入学生学号(十位)
输入
error:通用类型错误! 您的输入可能与事实不符或者与已有数据冲突, 请检查后重新输入!
常见类型: 输入的GPA不存在, 学生等级与绩点不匹配, 输入学分数值过大或过小, 输入格式错误, 姓名与学号和已录入的数据不符...
请输入学生学号(十位)
```

图 14: 输入不相符的数据时发出警告

6.4 case2() 测试图

正常流程输入下，case2 的输入流程不会报错。在输入学生姓名，学生学号和想要修改的学期后，系统将会输出所有符合条件的课程记录，将会进入 case2 的菜单页面。在菜单页面可以选择修改学生等级，学生绩点，课程 pf 或者直接删除该记录。修改的记录允许与上次的相同。在所有输入结束后，系统将会输出绿色的“输入成功”，提示用户已成功修改。

修改学生等级参见图15，修改学生绩点参见图16，修改课程 pf 参见图17，删除学生成绩参见图18。

```
*-----欢迎访问学生成绩系统!-----*
*-----录入学生的成绩单请输入1-----*
*-----修改学生的成绩单请输入2-----*
*-----查询学生的成绩单请输入3-----*
*-----查询学生的绩点请输入4-----*
*-----查询课程的均绩请输入5-----*
*-----按任意键退出程序-----*
2
请注意，只可修改学生等级，课程绩点与课程pf，不可修改其他值
请输入想要修改的学生姓名
王小明
请输入该学生的学号
1234567890
请输入想要修改的学期
1
选项 姓名      学号      课程名称      学分      绩点      等级      pf类型      学期
1 王小明      1234567890      测试用课程      2      4      A      非pf      1
请选择想要修改的记录(敲击记录前的数字)
1
王小明      1234567890      测试用课程      2      4      A      非pf      1
请选择想要修改的项目
1. 修改学生等级
2. 修改课程绩点
3. 修改课程pf
4. 删除该记录
1
请输入新的等级
A+
输入成功!
```

图 15: 修改学生等级

```
请注意，只可修改学生等级，课程绩点与课程pf，不可修改其他值
请输入想要修改的学生姓名
王小明
请输入该学生的学号
1234567890
请输入想要修改的学期
1
选项 姓名      学号      课程名称      学分      绩点      等级      pf类型      学期
1 王小明      1234567890      测试用课程      2      4      A+      非pf      1
请选择想要修改的记录(敲击记录前的数字)
1
王小明      1234567890      测试用课程      2      4      A+      非pf      1
请选择想要修改的项目
1. 修改学生等级
2. 修改课程绩点
3. 修改课程pf
4. 删除该记录
2
请输入该课程GPA, 退课GPA请输入-1, pf请输入0
3.6
输入成功!
```

图 16: 修改学生绩点

```

请注意，只可修改学生等级，课程绩点与课程pf，不可修改其他值
请输入想要修改的学生姓名
王小明
请输入该学生的学号
1234567890
请输入想要修改的学期
1
选项 姓名      学号      课程名称      学分      绩点      等级      pf类型      学期
1 王小明      1234567890      测试用课程      2      3.6      B+      非pf      1
请选择想要修改的记录(敲击记录前的数字)
1
王小明      1234567890      测试用课程      2      3.6      B+      非pf      1
请选择想要修改的项目
1. 修改学生等级
2. 修改课程绩点
3. 修改课程pf
4. 删除该记录
3
请输入课程类型，pf课程请输入0，非pf课程请输入1
0
输入成功!

```

图 17: 修改课程 pf

```

请注意，只可修改学生等级，课程绩点与课程pf，不可修改其他值
请输入想要修改的学生姓名
王小明
请输入该学生的学号
1234567890
请输入想要修改的学期
1
选项 姓名      学号      课程名称      学分      绩点      等级      pf类型      学期
1 王小明      1234567890      测试用课程      1      4      A-      非pf      1
请选择想要修改的记录(敲击记录前的数字)
5
cerror:TypeError!您输入的数据类型和要求的相符，请检查后重新输入!
请选择想要修改的记录(敲击记录前的数字)
1
王小明      1234567890      测试用课程      1      4      A-      非pf      1
请选择想要修改的项目
1. 修改学生等级
2. 修改课程绩点
3. 修改课程pf
4. 删除该记录
4
删除成功!

```

图 18: 删除学生成绩

应当注意的是，由于绩点和等级的对应性，只有特定的等级可以修改绩点。因此，如果学生的绩点不是 4.0，系统回发出警告并跳转到修改绩点处，参见图19。

```

请注意，只可修改学生等级，课程绩点与课程pf，不可修改其他值
请输入想要修改的学生姓名
王小明
请输入该学生的学号
1234567890
请输入想要修改的学期
1
选项 姓名      学号      课程名称      学分      绩点      等级      pf类型      学期
1 王小明      1234567890      测试用课程      2      3.6      B+      pf      1
请选择想要修改的记录(敲击记录前的数字)
1
王小明      1234567890      测试用课程      2      3.6      B+      pf      1
请选择想要修改的项目
1. 修改学生等级
2. 修改课程绩点
3. 修改课程pf
4. 删除该记录
1
无法修改等级! 请先修改绩点
请输入该课程GPA, 退课GPA请输入-1, pf请输入0

```

图 19: 无法修改学生等级

如果系统未找到想要修改学生的记录，将会输出红色字体警告用户不存在该学生的记录，参见图20。

```
请注意，只可修改学生等级，课程绩点与课程pf，不可修改其他值
请输入想要修改的学生姓名
不存在
请输入该学生的学号
1111111111
请输入想要修改的学期
1
选项 姓名      学号      课程名称      学分      绩点      等级      pf类型      学期
不存在该学生的记录!
```

图 20: 不存在该学生的记录

如果选择要修改的记录时，选择的记录与提供的所有选项都不相符，将会触发 `TypeError`，系统将会输出红色字体提醒用户，参见图21。

```
请注意，只可修改学生等级，课程绩点与课程pf，不可修改其他值
请输入想要修改的学生姓名
王小明
请输入该学生的学号
1234567890
请输入想要修改的学期
1
选项 姓名      学号      课程名称      学分      绩点      等级      pf类型      学期
1 王小明      1234567890      测试用课程      2      4      A      pf      1
请选择想要修改的记录(敲击记录前的数字)
5
cerror:TypeError!您输入的数据类型和要求的相符，请检查后重新输入!
请选择想要修改的记录(敲击记录前的数字)
```

图 21: 输入有误的选项

6.5 case3() 测试图

`case3()` 操作时需要输入想查询学生的姓名和学号，系统会将于此数据对应的所有数据排序输出。排序顺序为学期，课程，GPA, 等级和学分依次降序排列，参见图22。

```
请输入查询学生姓名
赵小一
请输入查询学生学号
1111111111
姓名      学号      课程名称      学分      绩点      等级      pf类型      学期
赵小一      1111111111      测试数据5      3      1.6      D+      pf      7
赵小一      1111111111      测试数据4      2      3      B-      pf      6
赵小一      1111111111      测试数据2      3      3.6      B+      非pf      5
赵小一      1111111111      测试数据3      1      2      C-      pf      3
赵小一      1111111111      测试数据1      4      4      A+      非pf      1
```

图 22: 赵小一成绩单

在完成一次输出后，系统将会询问是否继续输出，即允许用户连续查询成绩单，参见图23、图24。



```
3
请输入查询学生姓名
赵小一
请输入查询学生学号
1111111111
姓名      学号      课程名称      学分      绩点      等级      pf类型      学期
赵小一    1111111111    测试数据5      3      1.6      D+      非pf      7
赵小一    1111111111    测试数据4      2      3      B-      pf      6
赵小一    1111111111    测试数据2      3      3.6      B+      非pf      5
赵小一    1111111111    测试数据3      1      2      C-      pf      3
赵小一    1111111111    测试数据1      4      4      A+      非pf      1

输出完成!
是否继续查询? (y/n)
y
请输入查询学生姓名
钱小二
请输入查询学生学号
2222222222
姓名      学号      课程名称      学分      绩点      等级      pf类型      学期
钱小二    2222222222    测试数据10     4      3.3      B      pf      5
钱小二    2222222222    测试数据9      4      3      B-      非pf      4
钱小二    2222222222    测试数据8      5      2.6      C+      pf      3
钱小二    2222222222    测试数据7      4      2.3      C      非pf      2
钱小二    2222222222    测试数据6      3      2      C-      pf      1

输出完成!
是否继续查询? (y/n)
```

图 23: 赵小一 + 钱小二连续输出

```
C:\Users\zhaog\Desktop\bp\BigProject\64\Debug\BigProject.exe
是否继续查询? (y/n)
y
请输入查询学生姓名
钱小三
请输入查询学生学号
3333333333
姓名      学号      课程名称      学分      绩点      等级      pf类型      学期
钱小三    3333333333    测试数据13     2      4      A-      非pf      8
钱小三    3333333333    测试数据14     3      4      A+      pf      7
钱小三    3333333333    测试数据12     1      4      A-      pf      7
钱小三    3333333333    测试数据11     2      3.6      B+      非pf      6
钱小三    3333333333    测试数据15     3      4      A      非pf      1

输出完成!
是否继续查询? (y/n)
y
请输入查询学生姓名
李小四
请输入查询学生学号
4444444444
姓名      学号      课程名称      学分      绩点      等级      pf类型      学期
李小四    4444444444    测试数据20     2      2.3      C      非pf      8
李小四    4444444444    测试数据19     3      2.6      C+      pf      7
李小四    4444444444    测试数据18     1      3      B-      pf      7
李小四    4444444444    测试数据17     2      3.3      B      非pf      6
李小四    4444444444    测试数据16     3      3.6      B+      非pf      1

输出完成!
是否继续查询? (y/n)
```

图 24: 钱小三 + 李小四连续输出

如果用户输入不存在的学生或输入的学生姓名与学号不匹配，系统将不会输出任何课程记录，，参见图25、图26。



图 25: 输入不存在的学生

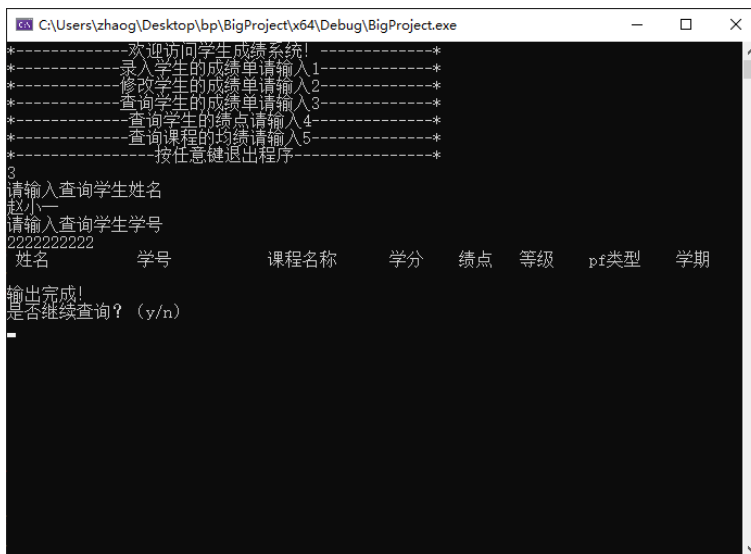


图 26: 输入错误的学生姓名和学号关系

## 6.6 case4() 测试图

case4() 有两个可选模式，用户可以选择查询所有学生的成绩绩点或查询单个学生的绩点。

如果输入“查询全部学生”，系统将会输出所有学生的总绩点，排序顺序为学期，课程，GPA, 等级和学分依次降序排列；如果输入想查询学生的姓名和学号，将会输出该学生的总绩点，总学分绩，GPA 相关总学分和每学期的平均绩点；在输出完成后，系统也会询问用户是否继续查询，参见图27。

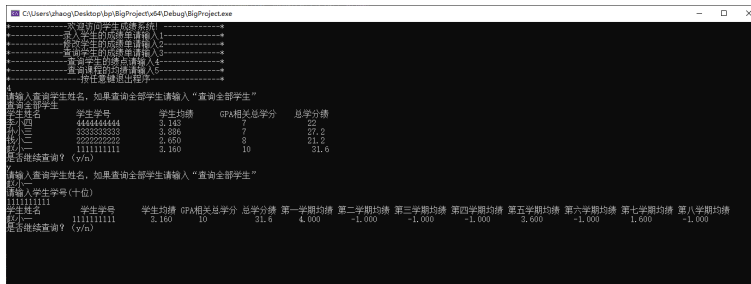


图 27: 查询全部学生 & 单独查询赵小一

同时，如果输入不存在的人，系统将会用红色字体警告查询的人不存在，参见图28。

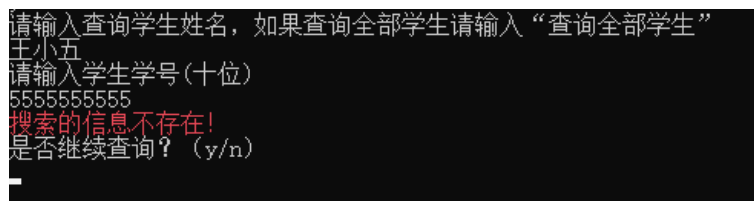


图 28: 输入不存在的人

## 6.7 case5() 测试图

case5() 有两个可选模式，用户可以选择查询所有课程的平均绩点或查询单个课程的平均绩点。两者的输出内容都是课程的名称，平均绩点和学分。参见图29、图30。

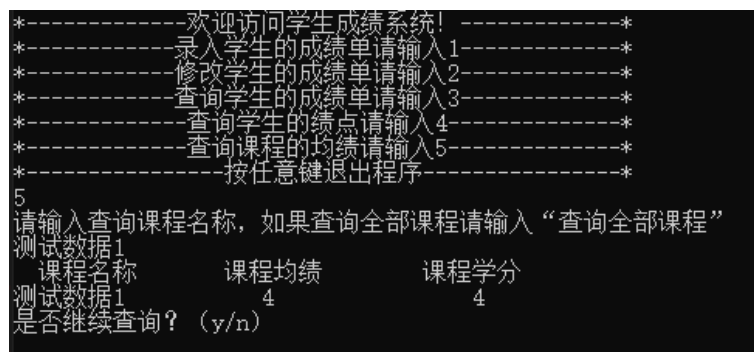


图 29: 查询单个课程

```
*-----欢迎访问学生成绩系统! -----*
*-----录入学生的成绩单请输入1-----*
*-----修改学生的成绩单请输入2-----*
*-----查询学生的成绩单请输入3-----*
*-----查询学生的绩点请输入4-----*
*-----查询课程的均绩请输入5-----*
*-----按任意键退出程序-----*
5
请输入查询课程名称，如果查询全部课程请输入“查询全部课程”
查询全部课程
课程名称      课程均绩      课程学分
测试数据20      2.3          2
课程名称      课程均绩      课程学分
测试数据19      2.6          3
课程名称      课程均绩      课程学分
测试数据18      3          1
课程名称      课程均绩      课程学分
测试数据17      3.3          2
课程名称      课程均绩      课程学分
测试数据16      3.6          3
课程名称      课程均绩      课程学分
测试数据13      4          2
课程名称      课程均绩      课程学分
测试数据14      4          3
课程名称      课程均绩      课程学分
测试数据12      4          1
课程名称      课程均绩      课程学分
测试数据11      3.6          2
课程名称      课程均绩      课程学分
测试数据15      4          3
课程名称      课程均绩      课程学分
测试数据10      3.3          4
课程名称      课程均绩      课程学分
测试数据9      3          4
课程名称      课程均绩      课程学分
测试数据8      2.6          5
课程名称      课程均绩      课程学分
测试数据7      2.3          4
课程名称      课程均绩      课程学分
测试数据6      2          3
课程名称      课程均绩      课程学分
测试数据5      1.6          3
课程名称      课程均绩      课程学分
测试数据4      3          2
课程名称      课程均绩      课程学分
测试数据2      3.6          3
课程名称      课程均绩      课程学分
测试数据3      2          1
课程名称      课程均绩      课程学分
测试数据1      4          4
是否继续查询？ (y/n)
```

图 30: 查询全部课程

如果用户输入的课程不存在，系统将会输出红色字体警告用户课程不存在，参见图31

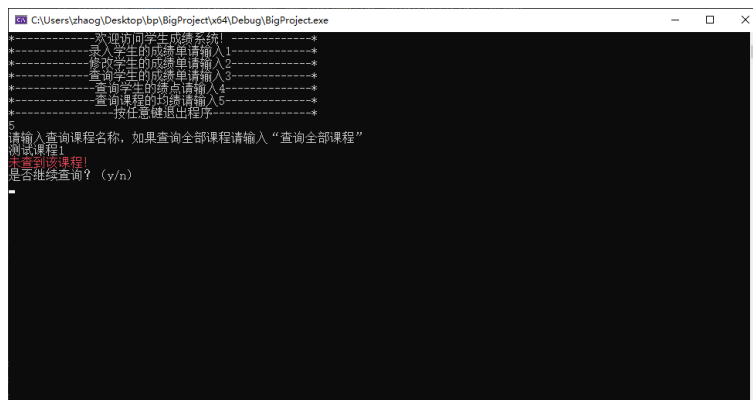


图 31: 输入不存在的课程

## 7 总结

从这次程设大作业中,我学到了很多东西。对我来说,这次大作业不仅仅给了我全面练习过去一年中所学的知识的机会,也让我学会了如何上手去写一个相对较大的项目。在这个过程中,我学会了如何清晰准确的表达出自己的疑问,学会了如何从互联网上寻找答案。在编程的亲身体验中,我也体会到了写注释和版本控制的重要性。这次程设大作业不仅仅是对我所学知识的一次应用,更是一次全新的学习的过程。

## 附录 1：源程序清单

```
1  #include<iostream>
2  #include<cstring>
3  #include<windows.h>
4  #include<fstream>
5  #include<iomanip>
6
7  HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
8  #pragma warning(disable:4996)
9  using namespace std;
10
11  //*****下面开始是类
12  //*****//
13
14  //基本类，里面装有基本的函数方法
15
16  class basic
17  {
18  public:
19      basic(){}
20      ~basic(){
21      }
22      virtual void wrong() = 0; //为其类中的报错函数提供接口
23      //较为通用的输入报错函数
24      static void typewrong()
25      {
26          SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY);
27          cout << "cerr:TypeError!您输入的数据类型和要求的相符，请检查后重新输入！"
28               << endl;
29          SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
30                                   FOREGROUND_BLUE);
31      }
32      static int count00; //记录现有的所有学生学号与姓名数据数
33      static char record1[10000][30]; //用于输入学生姓名
34      static char record2[10000][11]; //用于输入学生学号
35      static bool flag; //用于确认已经完成数据读入，防止数据读入时输出
36  };
37
38  int basic::count00 = 0;
39  char basic::record1[10000][30];
40  char basic::record2[10000][11];
41  bool basic::flag = false;
```

```

40
41
42 //学生类，里面继承了学生单门课程的信息
43 class people:virtual public basic
44 {
45 public:
46     people() {};
47     ~people() {};
48     void wrong()
49     {
50         SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY);
51         cout << "cerr:通用类型错误！您的输入可能与事实不符或者与已有数据冲突，请检
            查后重新输入！" << endl;
52         cout << "常见类型:输入的GPA不存在，学生等级与绩点不匹配，输入学分数值过大
            或过小，输入格式错误，姓名与学号和已录入的数据不符..." << endl;
53         SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
            FOREGROUND_BLUE);
54     }
55     people(const people &data) {
56         this->count = data.count;
57         this->score = data.score;
58         strcpy(this->name, data.name);
59         strcpy(this->lecture, data.lecture);
60         this->GPA = data.GPA;
61         strcpy(this->rank, data.rank);
62         this->type = data.type;
63         this->term = data.term;
64         strcpy(this->ID, data.ID);
65     }
66
67 //从键盘输入学生绩点并判断等级类型
68 void input_GPA()
69 {
70
71     cout << "请输入该课程GPA,退课GPA请输入-1, pf请输入0" << endl;
72     cin >> GPA;
73     while (!cin.good()) {
74         cin.clear();
75         cin.ignore(10000, '\n');
76         typewrong();
77         cout << "请输入该课程GPA,退课GPA请输入-1, pf请输入0" << endl;
78         cin >> GPA;
79     }
80     if (GPA != 4.0 && GPA != 3.6 && GPA != 3.3 && GPA != 3.0 && GPA != 2.6 &&
        GPA != 2.3 && GPA != 2.0 && GPA != 1.6 && GPA != 1.0 && GPA != 0.0 &&
        GPA != -1) {

```

```

81         wrong();
82         input_GPA();
83     }
84     else {
85         if (GPA == -1) { strcat(rank, "W"); }
86         if (GPA == 0.0) { strcat(rank, "F"); }
87         if (GPA == 1.0) { strcat(rank, "D-"); }
88         if (GPA == 1.3) { strcat(rank, "D"); }
89         if (GPA == 1.6) { strcat(rank, "D+"); }
90         if (GPA == 2.0) { strcat(rank, "C-"); }
91         if (GPA == 2.3) { strcat(rank, "C"); }
92         if (GPA == 2.6) { strcat(rank, "C+"); }
93         if (GPA == 3.0) { strcat(rank, "B-"); }
94         if (GPA == 3.3) { strcat(rank, "B"); }
95         if (GPA == 3.6) { strcat(rank, "B+"); }
96         if (GPA == 4.0) {
97             int flag = 0;
98             while (!flag) {
99                 flag = 1;
100                 cout << "请输入学生等级" << endl;
101                 cin >> rank;
102                 while (!cin.good())
103                     {
104                         cin.clear();
105                         cin.ignore(10000, '\n');
106                         typewrong();
107                         cout << "请输入该课程GPA,退课GPA请输入-1, pf请输入0"
108                             << endl;
109                         cin >> rank;
110                     }
111                 if (strcmp(rank, "A") * strcmp(rank, "A-") * strcmp(rank, "A+"
112                     )) {
113                     wrong();
114                     flag = 0;
115                 }
116             }
117         }
118     }
119     //从键盘输入课程学分
120     void input_score()
121     {
122         cout << "请输入课程学分（整数）" << endl;
123         cin >> score;
124         while (!cin.good()) {

```



```

125         cin.clear();
126         cin.ignore(10000, '\n');
127         typewrong();
128         cin >> score;
129     }
130     if (score < 0 || score > 20 || score < 0) {
131         wrong();
132         input_score();
133     }
134 }
135
136 //从键盘输入课程pf类型
137 void input_choice() {
138     cout << "请输入课程类型, pf课程请输入0, 非pf课程请输入1" << endl;
139     cin >> type;
140     while (!cin.good()) {
141         typewrong();
142         cin.clear();
143         cin.ignore(10000, '\n');
144         cin >> type;
145     }
146     while (type != 0 && type != 1) {
147         wrong();
148         cin >> type;
149     }
150 }
151
152 void input_term() {
153     cout << "请输入课程学期名称, 大一上为1, 最高为8 (大四下)" << endl;
154     cin >> term;
155     while (!cin.good()) {
156         typewrong();
157         cin.clear();
158         cin.ignore(10000, '\n');
159         cin >> term;
160     }
161     if (term != 1 && term != 2 && term != 3 && term != 4 && term != 5 && term
        != 6 && term != 7 && term != 8) { wrong(); input_choice(); }
162 }
163
164 void input_ID() {
165     cout << "请输入学生学号(十位)" << endl;
166     cin >> ID;
167     while (!cin.good()) {
168         cin.clear();
169         cin.ignore(10000, '\n');

```

```

170         typewrong();
171         cin >> ID;
172     }
173     if (strlen(ID) - 10) {
174         wrong();
175         input_ID();
176     }
177 }
178
179 bool check()
180 {
181     for (int i = 0; i < count00; i++)
182     {
183         if (strcmp(record1[i], name) == 0 && strcmp(record2[i], ID) == 0)
184         {
185             count--;
186             return false;
187         }
188     }
189     strcat(record1[count00], name);
190     strcat(record2[count00], ID);
191     for (int i = 0; i < count00; i++)
192     {
193         if ((strcmp(record1[i], name) != 0 && strcmp(record2[i], ID) == 0))
194         {
195             *record1[count00] = { 0 };
196             *record2[count00] = { 0 };
197             return true;
198         }
199     }
200     return false;
201 }
202 //输入函数，在用户选择1后输出指示，引导用户输入信息
203 int input()
204 {
205     cout << "请输入学生姓名" << endl;
206     cin >> name;
207     while (!cin.good()) {
208         cin.clear();
209         cin.ignore(10000, '\n');
210         typewrong();
211         cin >> name;
212     }
213     cin.clear();
214     cin.ignore(10000, '\n');
215     input_ID();

```

```

216     cin.clear();
217     cin.ignore(10000, '\n');
218     if (check()) {
219         *name = { 0 };
220         *ID = { 0 };
221         wrong();
222         return 0;
223     }
224     else
225     {
226         count00++;
227         input_GPA();
228         cin.clear();
229         cin.ignore(10000, '\n');
230         input_score();
231         cin.clear();
232         cin.ignore(10000, '\n');
233         cout << "请输入课程名称" << endl;
234         cin >> lecture;
235         while (!cin.good()) {
236             cin.clear();
237             cin.ignore(10000, '\n');
238             typewrong();
239
240             cin >> lecture;
241         }
242         cin.clear();
243         cin.ignore(10000, '\n');
244         input_choice();
245         cin.clear();
246         cin.ignore(10000, '\n');
247         input_term();
248         cin.clear();
249         cin.ignore(10000, '\n');
250         return 1;
251     }
252 }
253 //输出函数，查询时输出储存好的信息。
254 void output() {
255     if (GPA == 4)
256     {
257         SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_BLUE |
258             FOREGROUND_INTENSITY);
259         cout << name;
260         cout << " " << ID << " ";
261         cout << lecture;

```

```

261         cout << "          " << score << "          " << GPA << "          ";
262         cout << rank;
263         if (type) { cout << "          " << "非pf" << "          " << term << endl; }
264         else { cout << "          " << "pf" << "          " << term << endl; }
265         SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
                FOREGROUND_BLUE);
266     }
267     if (GPA < 4 && GPA >= 3)
268     {
269         SetConsoleTextAttribute(hConsole, FOREGROUND_BLUE |
                FOREGROUND_INTENSITY);
270         cout << name;
271         cout << "          " << ID << "          ";
272         cout << lecture;
273         cout << "          " << score << "          " << GPA << "          ";
274         cout << rank;
275         if (type) { cout << "          " << "非pf" << "          " << term << endl; }
276         else { cout << "          " << "pf" << "          " << term << endl; }
277         SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
                FOREGROUND_BLUE);
278     }
279     if (GPA < 3 && GPA >=2)
280     {
281         SetConsoleTextAttribute(hConsole, FOREGROUND_GREEN | FOREGROUND_BLUE |
                FOREGROUND_INTENSITY);
282         cout << name;
283         cout << "          " << ID << "          ";
284         cout << lecture;
285         cout << "          " << score << "          " << GPA << "          ";
286         cout << rank;
287         if (type) { cout << "          " << "非pf" << "          " << term << endl; }
288         else { cout << "          " << "pf" << "          " << term << endl; }
289         SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
                FOREGROUND_BLUE);
290     }
291     if (GPA < 2 && GPA > 0)
292     {
293         SetConsoleTextAttribute(hConsole, FOREGROUND_GREEN |
                FOREGROUND_INTENSITY);
294         cout << name;
295         cout << "          " << ID << "          ";
296         cout << lecture;
297         cout << "          " << score << "          " << GPA << "          ";
298         cout << rank;
299         if (type) { cout << "          " << "非pf" << "          " << term << endl; }
300         else { cout << "          " << "pf" << "          " << term << endl; }

```

```

301         SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
302             FOREGROUND_BLUE);
303     }
304     if (GPA <= 0 )
305     {
306         cout << name;
307         cout << " " << ID << " ";
308         cout << lecture;
309         cout << " " << score << " " << GPA << " ";
310         cout << rank;
311         if (type) { cout << " " << "非pf" << " " << term << endl; }
312         else { cout << " " << "pf" << " " << term << endl; }
313     }
314 }
315
316 static int count; //人数计数
317 int score = -1; //学分
318 char name[30] = {"ZanderZhao"}; //姓名
319 char lecture[30] = {"摸鱼课导论"}; //课程名
320 double GPA; //绩点
321 char rank[3] = {}; //等级
322 int type = 1; //是否pf
323 int term; //学期
324 char ID[11]; //学生学号
325 };
326
327 int people::count = 0;
328
329 //*****下面开始是存储数据的结构体
330 //*****//
331
332 //课程类，方便后面的课程排序
333 struct course
334 {
335     public:
336         course() {};
337         ~course() {};
338         static int count; //课程库里一共有几条课程
339         char name[30] = {"摸鱼课导论"}; // 课程名
340         double score = 0.0; // 学分数
341         int number = 0; // 报过的人数
342         double total_score = 0; //总学成绩
343         //int type; // 暂时去掉，因为pf机制的存在，课程本身的pf已经不重要了 重新更
344             改，这个现在是否手动置为pf
345         double ave_GPA; //平均分数
346 };
347
348 int course::count = 0;

```

```

344 //学生类，用来记录学生均绩，从而方便后面排序
345 struct student
346 {
347     public:
348         student() {};
349         ~student() {};
350         static int count; //学生人数计数
351         char name[30] = { "nullptr" }; //学生姓名
352         char ID[11] = { "0000000000" }; //学生学号
353         int score = 0; //学生总学分
354         double total_GPA = 0; //学生总学绩
355         double GPA_related_score = 0; //学生绩点相关学分
356         double ave_GPA = 0; //学生总绩点
357         double score_term[8] = { 0,0,0,0,0,0,0,0 }; //学生学期内总学分
358         double GPA_term[8] = { -1,-1,-1,-1,-1,-1,-1,-1 }; //学生学期内平均绩点
359         double total_GPA_term[8] = { 0,0,0,0,0,0,0,0 }; //学生学期内总绩点
360         int number = 0; //学生报的课程数
361 };
362 int student::count = 0;
363
364 //*****下面开始是链表
365 //*****//
366
367 // 节点类
368 class Node :virtual public basic, public people
369 {
370     public:
371         Node() {};
372         ~Node() {};
373         Node* next = NULL;
374         static int safe;
375         static int warning;
376         char err[40] = { "cerr:警告! 调用了错误的函数实例!" };
377         bool operator>(const Node& temp) {
378             if (strcmp(ID,temp.ID) > 0) { return 1; }
379             else if (strcmp(ID, temp.ID) == 0 && term > temp.term) { return 1; }
380             else if (strcmp(ID, temp.ID) == 0 && term == temp.term && strcmp(name, temp.
381                 name) > 0) { return 1; }
382             else if (strcmp(ID, temp.ID) == 0 && term == temp.term && strcmp(name, temp.
383                 name) == 0 && strcmp(lecture, temp.lecture) > 0) { return 1; }
384             else if (strcmp(ID, temp.ID) == 0 && term == temp.term && strcmp(name, temp.
385                 name) == 0 && strcmp(lecture, temp.lecture) == 0 && GPA > temp.GPA) {
386                 return 1; }
387             else if (strcmp(ID, temp.ID) == 0 && term == temp.term && strcmp(name, temp.
388                 name) == 0 && strcmp(lecture, temp.lecture) == 0 && GPA == temp.GPA &&

```

```

384     strcmp(rank, temp.rank) > 0) { return 1; }
else if (strcmp(ID, temp.ID) == 0 && term == temp.term && strcmp(name, temp.
    name) == 0 && strcmp(lecture, temp.lecture) == 0 && GPA == temp.GPA &&
    strcmp(rank, temp.rank) == 0 && score > temp.score) { return 1; }
385 else if (strcmp(ID, temp.ID) == 0 && term == temp.term && strcmp(name, temp.
    name) == 0 && strcmp(lecture, temp.lecture) == 0 && GPA == temp.GPA &&
    strcmp(rank, temp.rank) == 0 && score == temp.score && type > temp.type) {
    return 1; }
386 else return 0;
387 }
388 void wrong()
389 {
390     SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY);
391     cout << err << endl;
392     SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
        FOREGROUND_BLUE);
393 }
394 bool operator==(const Node& temp) {
395     if (!strcmp(ID, temp.ID) && strcmp(name, temp.name) == 0 && strcmp(lecture,
        temp.lecture) == 0 && GPA == temp.GPA && strcmp(rank, temp.rank) == 0 &&
        score == temp.score && type == temp.type && temp.term == term) { return 1;
        }
396     else return 0;
397 }
398
399 Node(Node& data) {
400     this->count = data.count;
401     this->score = data.score;
402     strcpy(this->name, data.name);
403     strcpy(this->lecture, data.lecture);
404     this->GPA = data.GPA;
405     strcpy(this->rank, data.rank);
406     this->type = data.type;
407     this->term = data.term;
408     strcpy(this->ID, data.ID);
409     this->next = nullptr;
410 }
411 };
412 int Node::safe = 0;
413 int Node::warning = 0;
414
415
416
417
418
419 // 链表类

```

```

420 class LinkedList :public basic {
421 public:
422     Node* head;
423     course a[20000];
424     student b[100000];
425     Node* slice[100] = { NULL };
426     LinkedList() {
427         head = nullptr;
428     }
429     void wrong()
430     {
431         SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY);
432         cout << "该条数据已经存在！" << endl;
433         SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
434             FOREGROUND_BLUE);
435         count00--;
436     }
437     // 添加节点函数
438     void addNode(people data) {
439         Node* newNode = new Node(data);
440         Node::safe++;
441         int i = 1;
442         int flag = 1;
443         char mid[30];
444         double middle;
445         char mid_id[11];
446         if (head == nullptr) {
447             head = newNode;
448             if (basic::flag)
449             {
450                 SetConsoleTextAttribute(hConsole, FOREGROUND_GREEN);
451                 cout << "输入成功！" << endl;
452                 SetConsoleTextAttribute(hConsole, FOREGROUND_RED |
453                     FOREGROUND_GREEN | FOREGROUND_BLUE);
454             }
455         }
456         else {
457             Node* current = head;
458
459             //链表插入顺序控制
460
461             if (*newNode == *current) {
462                 if (basic::flag) wrong();
463                 i = 0;
464                 flag = 0;
465                 Node::safe--;

```



```

464     }
465     else if (*newNode > (*current)) {
466         newNode->next = current;
467         head = newNode;
468         i = 0;
469         if (basic::flag) {
470             SetConsoleTextAttribute(hConsole, FOREGROUND_GREEN);
471             cout << "输入成功!" << endl;
472             SetConsoleTextAttribute(hConsole, FOREGROUND_RED |
473                                     FOREGROUND_GREEN | FOREGROUND_BLUE);
474         }
475     }
476     else if (current->next == nullptr) {
477         current->next = newNode;
478         i = 0;
479         if (basic::flag) {
480             SetConsoleTextAttribute(hConsole, FOREGROUND_GREEN);
481             cout << "输入成功!" << endl;
482             SetConsoleTextAttribute(hConsole, FOREGROUND_RED |
483                                     FOREGROUND_GREEN | FOREGROUND_BLUE);
484         }
485     }
486     else
487     {
488         while (i && current->next != nullptr)
489         {
490             if (*newNode == *current->next)
491             {
492                 if (basic::flag) wrong();
493                 i = 0;
494                 flag = 0;
495                 Node::safe--;
496                 break;
497             }
498             else if (*newNode > *current->next)
499             {
500                 newNode->next = current->next;
501                 if (basic::flag) {
502                     current->next = newNode;
503                     i = 0;
504                     SetConsoleTextAttribute(hConsole, FOREGROUND_GREEN);
505                     cout << "输入成功!" << endl;
506                     SetConsoleTextAttribute(hConsole, FOREGROUND_RED |
507                                             FOREGROUND_GREEN | FOREGROUND_BLUE);
508                     current = current->next;
509                 }
510             }
511         }
512     }

```

```

507         }
508         else
509         {
510             current = current->next;
511         }
512     }
513 }
514 if (i) {
515     current->next = newNode;
516     if (basic::flag) {
517         SetConsoleTextAttribute(hConsole, FOREGROUND_GREEN);
518         cout << "输入成功!" << endl;
519         SetConsoleTextAttribute(hConsole, FOREGROUND_RED |
520             FOREGROUND_GREEN | FOREGROUND_BLUE);
521     }
522 }
523 if (flag) {
524     int flag = 0;
525     //登记课程名
526     for (i = 0; i < a[0].count; i++)
527     {
528         if (!strcmp(a[i].name, data.lecture)) {
529             a[i].number++;
530             flag = 1;
531             a[i].total_score += data.GPA;
532             a[i].ave_GPA = a[i].total_score / a[i].number;
533         }
534     }
535 }
536 if (flag == 0) {
537     strcpy(a[i].name, data.lecture);
538     flag = 1;
539     a[i].count++;
540     a[i].number++;
541     a[i].total_score += data.GPA;
542     a[i].score = data.score;
543     a[i].ave_GPA = a[i].total_score / a[i].number;
544 }
545
546 flag = 0;
547
548 //登记学生姓名与id
549 for (i = 0; i < b[0].count; i++)
550 {
551     if (!strcmp(b[i].name, data.name) && !strcmp(b[i].ID, data.ID)) {

```

```

552         b[i].number++;
553         b[i].score += data.score;
554         b[i].GPA_related_score += data.type * data.score;
555         b[i].total_GPA += data.GPA * data.type * data.score;
556         b[i].ave_GPA = b[i].total_GPA / b[i].GPA_related_score;
557         b[i].total_GPA_term[data.term - 1] += data.GPA * data.type *
            data.score;
558         b[i].score_term[data.term - 1] += data.type * data.score;
559         if (b[i].score_term[data.term - 1] != 0) {
560             b[i].GPA_term[data.term - 1] = b[i].total_GPA_term[data.
                term - 1] / b[i].score_term[data.term - 1];
561         }
562         flag = 1;
563     }
564 }
565 if (flag == 0) {
566     strcpy(b[i].name, data.name);
567     strcpy(b[i].ID, data.ID);
568     flag = 1;
569     b[i].number++;
570     b[i].count++;
571     b[i].total_GPA += data.GPA * data.type * data.score;
572     b[i].GPA_related_score += data.type * data.score;
573     b[i].ave_GPA = b[i].total_GPA / b[i].GPA_related_score;
574     b[i].score += data.score;
575     b[i].total_GPA_term[data.term - 1] += data.GPA * data.type * data.
        score;
576     b[i].score_term[data.term - 1] += data.type * data.score;
577     if (b[i].score_term[data.term - 1] != 0) {
578         b[i].GPA_term[data.term - 1] = b[i].total_GPA_term[data.term -
            1] / b[i].score_term[data.term - 1];
579     }
580 }
581
582 flag = 0;
583
584 //重新排序课程
585 while (!flag)
586 {
587     flag = 1;
588     for (i = 0; i < a[0].count - 1; i++)
589     {
590         if (a[i].number < a[i + 1].number) {
591             strcpy(mid, a[i].name);
592             strcpy(a[i].name, a[i + 1].name);
593             strcpy(a[i + 1].name, mid);

```

```

594         middle = a[i].number;
595         a[i].number = a[i + 1].number;
596         a[i + 1].number = middle;
597         flag = 0;
598     }
599     if (a[i].number == a[i].number && strcmp(a[i].name, a[i].name)
600         < 0) {
601         strcpy(mid, a[i].name);
602         strcpy(a[i].name, a[i + 1].name);
603         strcpy(a[i + 1].name, mid);
604         middle = a[i].number;
605         a[i].number = a[i + 1].number;
606         a[i + 1].number = middle;
607         flag = 0;
608     }
609 }
610 flag = 0;
611 //重新排序学生姓名, GPA, ID和姓名降序排序。
612 while (!flag)
613 {
614     flag = 1;
615     for (i = 0; i < b[0].count - 1; i++)
616     {
617         if (b[i].ave_GPA < b[i + 1].ave_GPA) {
618             strcpy(mid, b[i].name);
619             strcpy(b[i].name, b[i + 1].name);
620             strcpy(b[i + 1].name, mid);
621             strcpy(mid_id, b[i].ID);
622             strcpy(b[i].ID, b[i + 1].ID);
623             strcpy(b[i + 1].ID, mid_id);
624             middle = b[i].score;
625             b[i].score = b[i + 1].score;
626             b[i + 1].score = middle;
627             middle = b[i].total_GPA;
628             b[i].total_GPA = b[i + 1].total_GPA;
629             b[i + 1].total_GPA = middle;
630             middle = b[i].GPA_related_score;
631             b[i].GPA_related_score = b[i + 1].GPA_related_score;
632             b[i + 1].GPA_related_score = middle;
633             middle = b[i].ave_GPA;
634             b[i].ave_GPA = b[i + 1].ave_GPA;
635             b[i + 1].ave_GPA = middle;
636             flag = 0;
637         }
638         else if (b[i].ave_GPA == b[i + 1].ave_GPA && strcmp(b[i].ID, b[i + 1].

```

```

639         ID) < 0) {
640             strcpy(mid, b[i].name);
641             strcpy(b[i].name, b[i + 1].name);
642             strcpy(b[i + 1].name, mid);
643             strcpy(mid_id, b[i].ID);
644             strcpy(b[i].ID, b[i + 1].ID);
645             strcpy(b[i + 1].ID, mid_id);
646             middle = b[i].score;
647             b[i].score = b[i + 1].score;
648             b[i + 1].score = middle;
649             middle = b[i].total_GPA;
650             b[i].total_GPA = b[i + 1].total_GPA;
651             b[i + 1].total_GPA = middle;
652             middle = b[i].GPA_related_score;
653             b[i].GPA_related_score = b[i + 1].GPA_related_score;
654             b[i + 1].GPA_related_score = middle;
655             middle = b[i].ave_GPA;
656             b[i].ave_GPA = b[i + 1].ave_GPA;
657             b[i + 1].ave_GPA = middle;
658             flag = 0;
659         }
660     else if (b[i].ave_GPA == b[i + 1].ave_GPA && strcmp(b[i].ID, b[i + 1].
661         ID) == 0 && strcmp(b[i].ID, b[i + 1].ID) < 0) {
662         strcpy(mid, b[i].name);
663         strcpy(b[i].name, b[i + 1].name);
664         strcpy(b[i + 1].name, mid);
665         strcpy(mid_id, b[i].ID);
666         strcpy(b[i].ID, b[i + 1].ID);
667         strcpy(b[i + 1].ID, mid_id);
668         middle = b[i].score;
669         b[i].score = b[i + 1].score;
670         b[i + 1].score = middle;
671         middle = b[i].total_GPA;
672         b[i].total_GPA = b[i + 1].total_GPA;
673         b[i + 1].total_GPA = middle;
674         middle = b[i].GPA_related_score;
675         b[i].GPA_related_score = b[i + 1].GPA_related_score;
676         b[i + 1].GPA_related_score = middle;
677         middle = b[i].ave_GPA;
678         b[i].ave_GPA = b[i + 1].ave_GPA;
679         b[i + 1].ave_GPA = middle;
680         flag = 0;
681     }
682 }

```

```

683     }
684     //搜索节点
685     int searchNode(const char name[30], const char ID[11], int term) {
686         Node* current = head;
687         int j = -1;
688         int i = 0;
689         while (current != NULL)
690         {
691             if (strcmp(current->name, name) == 0 && strcmp(current->ID, ID) == 0 &&
692                 term == current->term)
693             {
694                 cout << i + 1 << "      ";
695                 current->output();
696                 slice[i] = current;
697                 i++;
698             }
699             j = i - 1;
700             current = current->next;
701         }
702         return j;
703     }
704     //将节点从链表中摘出
705     people pickoutNode(Node t) {
706         Node* current = head;
707         if (*head == t)
708         {
709             current = head->next;
710             head->next = NULL;
711             head = current;
712         }
713         else
714         {
715             while (current->next != NULL && !(*current->next == t)){
716                 current = current->next;
717             }
718             current->next = t.next;
719             t.next = NULL;
720         }
721         int flag = 1;
722         if (flag) {
723             int i;
724             //将该课程名字登记在course类中方便记录
725             for (i = 0; i < a[0].count; i++)
726             {
727                 if (!strcmp(a[i].name, t.lecture)) {

```

```

728         if (a[i].number == 0)
729         {
730             a[i].count--;
731             strcpy(a[i].name, "摸鱼课导论");
732         }
733         a[i].total_score -= t.GPA;
734         if (a[i].number != 0)
735         {
736             a[i].ave_GPA = a[i].total_score / a[i].number;
737             a[i].count--;
738         }
739         else a[i].ave_GPA = 0;
740     }
741 }
742
743 //将该学生名字登记在student类中方便记录
744 for (i = 0; i < b[0].count; i++)
745 {
746     if (!strcmp(b[i].name, t.name) && !strcmp(b[i].ID, t.ID)) {
747         b[i].number--;
748
749         b[i].score -= t.score;
750         b[i].GPA_related_score -= t.type * t.score;
751         b[i].total_GPA -= t.GPA * t.type * t.score;
752
753         b[i].total_GPA_term[t.term - 1] -= t.GPA * t.type * t.score;
754         b[i].score_term[t.term - 1] -= t.type * t.score;
755         if (!b[i].number)
756         {
757             strcpy(b[i].name, "nullptr");
758             strcpy(b[i].ID, "1111111111");
759             b[i].ave_GPA = 0;
760             b[i].GPA_term[t.term - 1] = 0;
761             b[i].count--;
762         }
763         else
764         {
765             b[i].ave_GPA = b[i].total_GPA / b[i].GPA_related_score;
766             b[i].GPA_term[t.term - 1] = b[i].total_GPA_term[t.term - 1] /
                b[i].score_term[t.term - 1];
767         }
768     }
769 }
770
771 flag = 0;
772 char mid[30];

```

```

773     char mid_id[11];
774     double middle;
775     while (!flag)
776     {
777         flag = 1;
778         for (i = 0; i < a[0].count - 1; i++)
779         {
780             if (a[i].number < a[i+1].number) {
781                 strcpy(mid, a[i].name);
782                 strcpy(a[i].name, a[i + 1].name);
783                 strcpy(a[i + 1].name, mid);
784                 middle = a[i].number;
785                 a[i].number = a[i + 1].number;
786                 a[i + 1].number = middle;
787                 flag = 0;
788             }
789             if (a[i].number == a[i+1].number && strcmp(a[i].name, a[i+1].name)
790                 < 0) {
791                 strcpy(mid, a[i].name);
792                 strcpy(a[i].name, a[i + 1].name);
793                 strcpy(a[i + 1].name, mid);
794                 middle = a[i].number;
795                 a[i].number = a[i + 1].number;
796                 a[i + 1].number = middle;
797                 flag = 0;
798             }
799         }
800
801         flag = 0;
802
803         //重新排序学生姓名 base on GPA
804
805         while (!flag)
806         {
807             flag = 1;
808             for (i = 0; i < b[0].count - 1; i++)
809             {
810                 if (b[i].ave_GPA < b[i+1].ave_GPA) {
811                     strcpy(mid, b[i].name);
812                     strcpy(b[i].name, b[i + 1].name);
813                     strcpy(b[i + 1].name, mid);
814                     strcpy(mid_id, b[i].ID);
815                     strcpy(b[i].ID, b[i + 1].ID);
816                     strcpy(b[i + 1].ID, mid_id);
817                     middle = b[i].score;

```



```

818         b[i].score = b[i + 1].score;
819         b[i + 1].score = middle;
820         middle = b[i].total_GPA;
821         b[i].total_GPA = b[i + 1].total_GPA;
822         b[i + 1].total_GPA = middle;
823         middle = b[i].GPA_related_score;
824         b[i].GPA_related_score = b[i + 1].GPA_related_score;
825         b[i + 1].GPA_related_score = middle;
826         middle = b[i].ave_GPA;
827         b[i].ave_GPA = b[i + 1].ave_GPA;
828         b[i + 1].ave_GPA = middle;
829         flag = 0;
830     }
831     else if (b[i].ave_GPA == b[i].ave_GPA && strcmp(b[i].ID, b[i].
832         ID) < 0) {
833         strcpy(mid, b[i].name);
834         strcpy(b[i].name, b[i + 1].name);
835         strcpy(b[i + 1].name, mid);
836         strcpy(mid_id, b[i].ID);
837         strcpy(b[i].ID, b[i + 1].ID);
838         strcpy(b[i + 1].ID, mid_id);
839         middle = b[i].score;
840         b[i].score = b[i + 1].score;
841         b[i + 1].score = middle;
842         middle = b[i].total_GPA;
843         b[i].total_GPA = b[i + 1].total_GPA;
844         b[i + 1].total_GPA = middle;
845         middle = b[i].GPA_related_score;
846         b[i].GPA_related_score = b[i + 1].GPA_related_score;
847         b[i + 1].GPA_related_score = middle;
848         middle = b[i].ave_GPA;
849         b[i].ave_GPA = b[i + 1].ave_GPA;
850         b[i + 1].ave_GPA = middle;
851         flag = 0;
852     }
853     else if (b[i].ave_GPA == b[i].ave_GPA && strcmp(b[i].ID, b[i].
854         ID) == 0 && strcmp(b[i].ID, b[i].ID) < 0) {
855         strcpy(mid, b[i].name);
856         strcpy(b[i].name, b[i + 1].name);
857         strcpy(b[i + 1].name, mid);
858         strcpy(mid_id, b[i].ID);
859         strcpy(b[i].ID, b[i + 1].ID);
860         strcpy(b[i + 1].ID, mid_id);
861         middle = b[i].score;
862         b[i].score = b[i + 1].score;
863         b[i + 1].score = middle;

```

```

862         middle = b[i].total_GPA;
863         b[i].total_GPA = b[i + 1].total_GPA;
864         b[i + 1].total_GPA = middle;
865         middle = b[i].GPA_related_score;
866         b[i].GPA_related_score = b[i + 1].GPA_related_score;
867         b[i + 1].GPA_related_score = middle;
868         middle = b[i].ave_GPA;
869         b[i].ave_GPA = b[i + 1].ave_GPA;
870         b[i + 1].ave_GPA = middle;
871         flag = 0;
872     }
873 }
874 }
875 }
876     return people(t);
877 }
878 };
879 LinkedList total;
880
881
882 //*****下面开始是函数
883 *****
884
885 // 给定需要输出的学生姓名，遍历链表并打印节点的值
886 void output(char* a, char* id)
887 {
888     Node* current = total.head;
889     while (current != nullptr) {
890         if (strcmp(current->name, a) == 0 && strcmp(current->ID, id) == 0) current
            ->output();
891         current = current->next;
892     }
893     cout << endl;
894 }
895
896 // 输出欢迎栏
897 void welcome()
898 {
899     cout << "*-----欢迎访问学生成绩系统! -----*" << endl;
900     cout << "*-----录入学生的成绩单请输入1-----*" << endl;
901     cout << "*-----修改学生的成绩单请输入2-----*" << endl;
902     cout << "*-----查询学生的成绩单请输入3-----*" << endl;
903     cout << "*-----查询学生的绩点请输入4-----*" << endl;
904     cout << "*-----查询课程的均绩请输入5-----*" << endl;
905     cout << "*-----按任意键退出程序-----*" << endl;

```

```

906     }
907
908     //提前说明函数
909     void case1();
910     void case2();
911     void case3();
912     void case4();
913     void case5();
914     void finish();
915
916     //匹配函数
917     void pipei()
918     {
919         int a;
920         cin >> a;
921         switch (a)
922         {
923             case 1:
924                 case1();
925             case 2:
926                 case2();
927             case 3:
928                 case3();
929             case 4:
930                 case4();
931             case 5:
932                 case5();
933             default:
934                 finish();
935                 cout << "退出成功！" << endl;
936                 if (Node::safe != Node::warning)
937                 {
938                     SetConsoleTextAttribute(hConsole, FOREGROUND_RED |
939                                             FOREGROUND_INTENSITY);
940                     cout << "警告：发生未知错误，您存储的数据很可能并未被正确存储进文件！"
941                         << endl;
942                     SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
943                                             FOREGROUND_BLUE);
944                 }
945                 exit(0);
946             }
947         }
948
949     //公用输入函数，将输入输入到链表里面
950     void shuru()
951     {

```

```

949     int flag;
950     people a;
951     flag = a.input();
952     if (flag) total.addNode(a);
953 }
954
955 //录入成绩单函数
956 void case1()
957 {
958     shuru();
959     cout << "是否继续输入? (y/n) " << endl;
960     char a[2];
961     cin >> a;
962     while (!cin.good())
963     {
964         cin.clear();
965         cin.ignore(10000, '\n');
966         basic::typewrong();
967         cin >> a;
968     }
969     cin.clear();
970     cin.ignore(10000, '\n');
971     if (!strcmp(a, "y")) case1();
972     else {
973         welcome();
974         pipei();
975     }
976 }
977
978 //修改成绩单函数
979 void case2()
980 {
981     int term;
982     char name[30];
983     char ID[11];
984     int j;
985     int choose;
986     int match;
987     cout << "请注意, 只可修改学生等级, 课程绩点与课程pf, 不可修改其他值" << endl;
988     cout << "请输入想要修改的学生姓名" << endl;
989     cin >> name;
990     while (!cin.good())
991     {
992         cin.clear();
993         cin.ignore(10000, '\n');
994         basic::typewrong();

```

```

995         cin >> name;
996     }
997     cout << "请输入该学生的学号" << endl;
998     cin >> ID;
999     while (!cin.good())
1000     {
1001         cin.clear();
1002         cin.ignore(10000, '\n');
1003         basic::typewrong();
1004         cin >> ID;
1005     }
1006     cout << "请输入想要修改的学期" << endl;
1007     cin >> term;
1008     while (!cin.good())
1009     {
1010         cin.clear();
1011         cin.ignore(10000, '\n');
1012         basic::typewrong();
1013         cin >> term;
1014     }
1015     cout << "选项 姓名          学号          课程名称          学分          绩点          等级
          pf类型          学期" << endl;
1016     j = total.searchNode(name, ID, term);
1017     if (j != -1) {
1018         cout << "请选择想要修改的记录(敲击记录前的数字)" << endl;
1019         cin >> choose;
1020         choose = choose - 1;
1021         while (choose > j || choose < 0)
1022         {
1023             basic::typewrong();
1024             cout << "请选择想要修改的记录(敲击记录前的数字)" << endl;
1025             cin >> choose;
1026             choose = choose - 1;
1027         }
1028         people a;
1029         a = total.pickoutNode(*total.slice[choose]);
1030         a.output();
1031         cout << "请选择想要修改的项目" << endl;
1032         cout << "1.修改学生等级" << endl;
1033         cout << "2.修改课程绩点" << endl;
1034         cout << "3.修改课程pf" << endl;
1035         cout << "4.删除该记录" << endl;
1036         cin >> match;
1037         cin.clear();
1038         cin.ignore(10000, '\n');
1039         while (match != 1 && match != 2 && match != 3 && match != 4)

```

```

1040 {
1041     basic::typewrong();
1042     cout << "请选择想要修改的项目" << endl;
1043     cout << "1.修改学生等级" << endl;
1044     cout << "2.修改课程绩点" << endl;
1045     cout << "3.修改课程pf" << endl;
1046     cin >> match;
1047     cin.clear();
1048     cin.ignore(10000, '\n');
1049 }
1050 if (match == 1)
1051 {
1052     if (total.slice[choose]->GPA != 4.0) {
1053         SetConsoleTextAttribute(hConsole, FOREGROUND_RED |
1054             FOREGROUND_INTENSITY);
1055         cout << "无法修改等级! 请先修改绩点" << endl;
1056         SetConsoleTextAttribute(hConsole, FOREGROUND_RED |
1057             FOREGROUND_GREEN | FOREGROUND_BLUE);
1058         match = 2;
1059     }
1060     else {
1061         cout.flush();
1062         cout << "请输入新的等级" << endl;
1063         char temp[3];
1064         cin >> temp;
1065         while (strcmp(temp, "A-") * strcmp(temp, "A+") * strcmp(temp, "A")
1066             != 0)
1067         {
1068             basic::typewrong();
1069             cin >> temp;
1070         }
1071         a.rank[0] = NULL;
1072         a.rank[1] = NULL;
1073         a.rank[2] = NULL;
1074         strcpy( a.rank,temp);
1075         total.addNode(a);
1076     }
1077 }
1078 if (match == 2)
1079 {
1080     a.rank[0] = NULL;
1081     a.rank[1] = NULL;
1082     a.rank[2] = NULL;
1083     a.input_GPA();
1084     total.addNode(a);
1085 }

```

```

1083         if (match == 3)
1084         {
1085             a.input_choice();
1086             total.addNode(a);
1087         }
1088         if (match == 4)
1089         {
1090             a.~people();
1091             Node::safe--;
1092             SetConsoleTextAttribute(hConsole, FOREGROUND_GREEN);
1093             cout << "删除成功!" << endl;
1094             SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
                                     FOREGROUND_BLUE);
1095         }
1096         welcome();
1097         pipei();
1098     }
1099     else {
1100         SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY);
1101         cout << "不存在该学生的记录!" << endl;
1102         SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
                                 FOREGROUND_BLUE);
1103         welcome();
1104         pipei();
1105     }
1106     cout << "是否继续查询? (y/n)" << endl;
1107     char a[2];
1108     cin >> a;
1109     while (!cin.good())
1110     {
1111         cin.clear();
1112         cin.ignore(10000, '\n');
1113         basic::typewrong();
1114         cin >> a;
1115     }
1116     cin.clear();
1117     cin.ignore(10000, '\n');
1118     if (!strcmp(a, "y")) case2();
1119     else {
1120         welcome();
1121         pipei();
1122     }
1123 }
1124
1125 //查询成绩单函数
1126 void case3()

```

```

1127 {
1128     char a[30];
1129     cout << "请输入查询学生姓名" << endl;
1130     cin >> a;
1131     while (!cin.good())
1132     {
1133         cin.clear();
1134         cin.ignore(10000, '\n');
1135         basic::typewrong();
1136         cin >> a;
1137     }
1138     char id[11];
1139     cout << "请输入查询学生学号" << endl;
1140     cin >> id;
1141     while (!cin.good())
1142     {
1143         cin.clear();
1144         cin.ignore(10000, '\n');
1145         basic::typewrong();
1146         cin >> id;
1147     }
1148     cout << " 姓名          学号          课程名称          学分          绩点          等级          pf
          类型          学期" << endl;
1149     output(a,id);
1150
1151     cout << "输出完成!" << endl;
1152     cout << "是否继续查询? (y/n) " << endl;
1153     char b[2];
1154     cin >> b;
1155     while (!cin.good())
1156     {
1157         cin.clear();
1158         cin.ignore(10000, '\n');
1159         basic::typewrong();
1160         cin >> b;
1161     }
1162     cin.clear();
1163     cin.ignore(10000, '\n');
1164     if (!strcmp(b, "y")) case3();
1165     else {
1166         welcome();
1167         pipei();
1168     }
1169 }
1170
1171 //查询均绩函数

```



```

1172 void case4()
1173 {
1174     int i;
1175     char name[20];
1176     cout << "请输入查询学生姓名，如果查询全部学生请输入“查询全部学生”" << endl;
1177     cin >> name;
1178     if (!strcmp(name, "查询全部学生"))
1179     {
1180         cout << "学生姓名" << " " << "学生学号" << " " << "学生均
            绩" << " " << "GPA相关总学分" << " " << "总学分绩" << " "
            << endl;
1181         for (i = 0; i < total.b[0].count; i++)
1182         {
1183             cout << total.b[i].name << " " << total.b[i].ID << "
                " << std::fixed << std::setprecision(3) << total.b[i].ave_GPA;
1184             cout.unsetf(ios::fixed);
1185             cout << " " << total.b[i].GPA_related_score << "
                " << total.b[i].total_GPA << endl;
1186         }
1187     }
1188     else
1189     {
1190         char ID[11];
1191         cout << "请输入学生学号(十位)" << endl;
1192         cin >> ID;
1193         while (!cin.good()) {
1194             cin.clear();
1195             cin.ignore(10000, '\n');
1196             basic::typewrong();
1197             cin >> ID;
1198         }
1199         for (i = 0; i < total.b[0].count; i++)
1200         {
1201             if (!strcmp(name, total.b[i].name) && !strcmp(ID, total.b[i].ID))
                break;
1202         }
1203         if (i < total.b[0].count)
1204         {
1205             cout << "学生姓名" << " " << "学生学号" << " " << "学生均
                绩" << " " << "GPA相关总学分" << " " << "总学分绩" << " " << "第一
                学期均绩" << " " << "第二学期均绩" << " " << "第三学期均绩" << " "
                << "第四学期均绩" << " " << "第五学期均绩" << " " << "第六学期均
                绩" << " " << "第七学期均绩" << " " << "第八学期均绩" << endl;
1206             cout << total.b[i].name << " " << total.b[i].ID << " "
                << std::fixed << std::setprecision(3) << total.b[i].ave_GPA;
1207             cout.unsetf( ios::fixed );

```

```

1208         cout << "          " << total.b[i].GPA_related_score << "          " <<
            total.b[i].total_GPA << "          ";
1209     for (int j = 0; j < 8; j++)
1210     {
1211         cout << std::fixed << std::setprecision(3) << total.b[i].GPA_term[
            j] << "          ";
1212     }
1213     cout.unsetf(ios::fixed);
1214     cout << endl;
1215 }
1216 else
1217 {
1218     SetConsoleTextAttribute(hConsole, FOREGROUND_RED |
        FOREGROUND_INTENSITY);
1219     cout << "搜索的信息不存在!" << endl;
1220     SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
        FOREGROUND_BLUE);
1221 }
1222 }
1223 cout << "是否继续查询? (y/n) " << endl;
1224 char a[2];
1225 cin >> a;
1226 while (!cin.good())
1227 {
1228     cin.clear();
1229     cin.ignore(10000, '\n');
1230     basic::typewrong();
1231     cin >> a;
1232 }
1233 cin.clear();
1234 cin.ignore(10000, '\n');
1235 if (!strcmp(a, "y")) case4();
1236 else {
1237     welcome();
1238     pipei();
1239 }
1240 }
1241
1242 //查询课程函数
1243 void case5()
1244 {
1245     int i;
1246     char name[20];
1247     cout << "请输入查询课程名称, 如果查询全部课程请输入“查询全部课程” " << endl;
1248     cin >> name;
1249     if (!strcmp(name, "查询全部课程"))

```

```

1250 {
1251     for (i = 0; i < total.a[0].count; i++)
1252     {
1253         cout << " " << "课程名称" << " " << "课程均绩" << " " <<
            "课程学分" << " " << endl;
1254         cout << total.a[i].name << " " << total.a[i].ave_GPA << "
            " << total.a[i].score << endl;
1255     }
1256 }
1257 else
1258 {
1259     for (i = 0; i < total.a[0].count; i++)
1260     {
1261         if (!strcmp(name, total.a[i].name)) break;
1262     }
1263     if (i == total.a[0].count) {
1264         SetConsoleTextAttribute(hConsole, FOREGROUND_RED |
            FOREGROUND_INTENSITY);
1265         cout << "未查到该课程!" << endl;
1266         SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
            FOREGROUND_BLUE);
1267     }
1268     else {
1269         cout << " 课程名称" << " " << "课程均绩" << " " << "课程
            学分" << " " << endl;
1270         cout << total.a[i].name << " " << total.a[i].ave_GPA << "
            " << total.a[i].score << endl;
1271     }
1272 }
1273 cout << "是否继续查询? (y/n) " << endl;
1274 char a[2];
1275 cin >> a;
1276 while (!cin.good())
1277 {
1278     cin.clear();
1279     cin.ignore(10000, '\n');
1280     basic::typewrong();
1281     cin >> a;
1282 }
1283 cin.clear();
1284 cin.ignore(10000, '\n');
1285 if (!strcmp(a, "y")) case5();
1286 else {
1287     welcome();
1288     pipei();
1289 }

```

```

1290 }
1291
1292 //将数据从文件中读入到系统中
1293 static void prepare()
1294 {
1295     fstream number("number.txt", ios::in); //全局变量学号存储文件
1296     fstream node("node.txt", ios::in); //链表节点存储文件
1297     if (!node.is_open()) {
1298         SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY);
1299         cout << "链表节点存储文件打开或创建失败！" << endl;
1300         SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
1301             FOREGROUND_BLUE);
1302     }
1303
1304     if (!number.is_open()) {
1305         SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY);
1306         cout << "学号存储文件打开或创建失败！" << endl;
1307         SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
1308             FOREGROUND_BLUE);
1309     }
1310     number.clear();
1311     node.clear();
1312     while (!node.eof())
1313     {
1314         people temp;
1315         node >> temp.name;
1316         if (node.eof()) break;
1317         node.ignore(1);
1318         node >> temp.ID;
1319         node.ignore(1);
1320         node >> temp.GPA;
1321         node.ignore(1);
1322         node >> temp.rank;
1323         node.ignore(1);
1324         node >> temp.score;
1325         node.ignore(1);
1326         node >> temp.lecture;
1327         node.ignore(1);
1328         node >> temp.type;
1329         node.ignore(1);
1330         node >> temp.term;
1331         node.ignore(1);
1332         total.addNode(temp);
1333         cout << "1" << endl;
1334     }
1335     int count_temp = 0;

```

```

1334     while (!number.eof())
1335     {
1336         number >> basic::record1[count_temp];
1337         if (number.eof()) break;
1338         node.ignore(1);
1339         number >> basic::record2[count_temp];
1340         node.ignore(1);
1341         count_temp++;
1342         basic::count00++;
1343     }
1344     node.close();
1345     number.close();
1346     basic::flag = true;
1347 }
1348
1349 //将数据输入到文件中，会清空文件中已有的数据
1350 static void finish()
1351 {
1352     fstream node("node.txt", ios::trunc | ios::out); //链表节点存储文件
1353     number.clear();
1354     node.clear();
1355     if (!node.is_open()) {
1356         SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY);
1357         cout << "链表节点存储文件打开或创建失败！" << endl;
1358         SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
1359             FOREGROUND_BLUE);
1360     }
1361     if (!number.is_open()) {
1362         SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY);
1363         cout << "学号存储文件打开或创建失败！" << endl;
1364         SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
1365             FOREGROUND_BLUE);
1366     }
1367     Node* current = total.head;
1368     while (current != nullptr)
1369     {
1370         node << current->name;
1371         node << " ";
1372         node << current->ID;
1373         node << " ";
1374         node << current->GPA;
1375         node << " ";
1376         node << current->rank;
1377         node << " ";
1378         node << current->score;
1379         node << " ";

```

```

1378         node << current->lecture;
1379         node << " ";
1380         node << current->type;
1381         node << " ";
1382         node << current->term;
1383         node << endl;
1384         current = current->next;
1385         Node::warning++;
1386     }
1387     int count = 0;
1388     while (count < basic::count00)
1389     {
1390         number << basic::record1[count];
1391         number << " ";
1392         number << basic::record2[count];
1393         number << endl;
1394         count++;
1395     }
1396     node.close();
1397     number.close();
1398 }
1399
1400 //主函数
1401 int main()
1402 {
1403     SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
1404                             FOREGROUND_BLUE);
1405     prepare();
1406     welcome();
1407     pipei();
1408 }

```

## 附录 2：时间安排

小学期 第一周	第一天	搭建大致框架
	第二天	完成类的定义
	第三天	编写并调试case1()
	第四天	编写并调试case2()
	第五天	编写并调试case3()
	第六天	编写case4()并修改bug
	第七天	继续编写case4()
小学期 第二周	第一天	编写case5()
	第二天	为已编写的程序添加功能
	第三天	整体调试
	第四天	整体调试
	第五天	细节微调
	第六天	编写实验报告
	第七天	编写实验报告

## 评价表

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	