# 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 15 日

# 目　　录

# 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，输出的字体是红色的，这更加醒目，使得用户的使用更加便利。

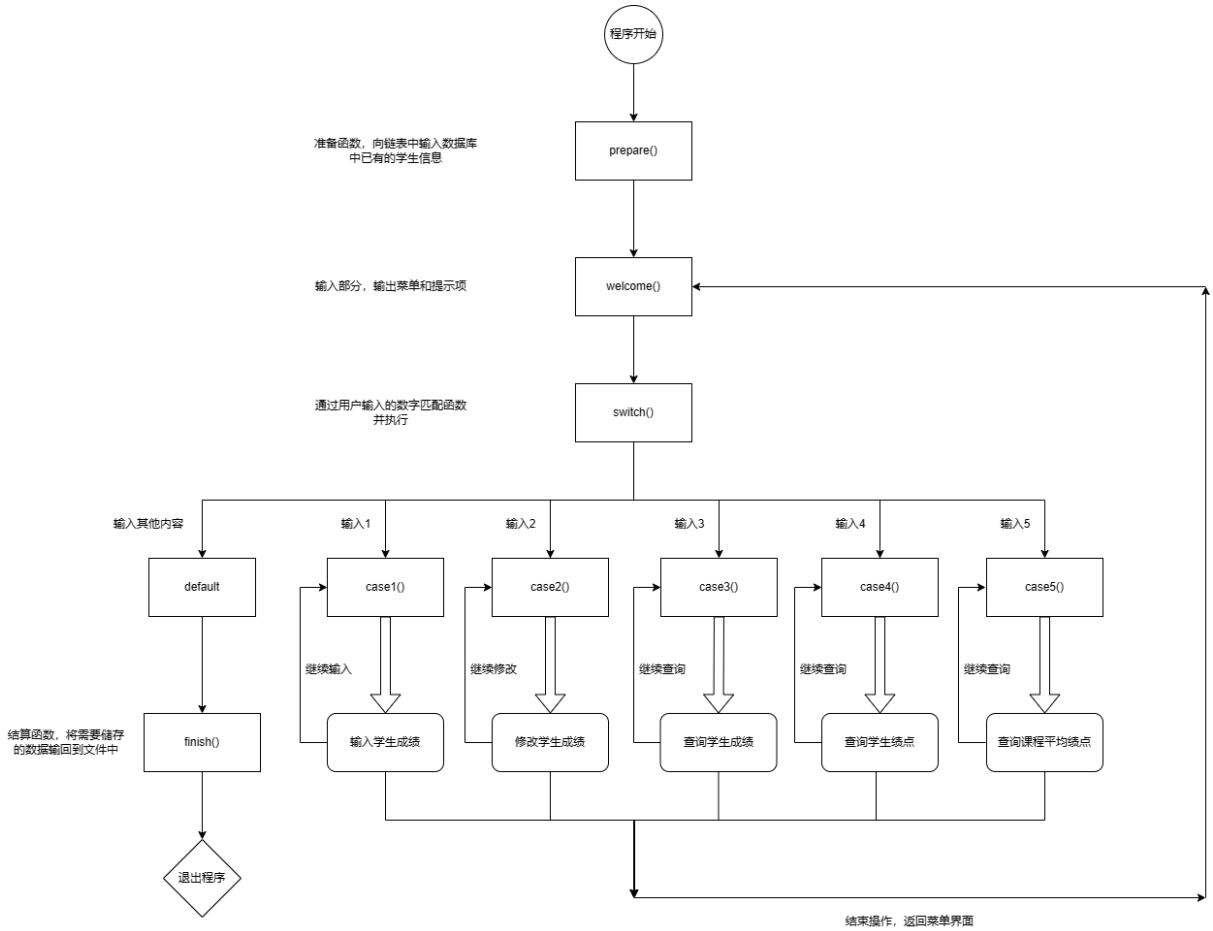


图 1: 学生管理系统的流程图

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存储到文件中，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

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。

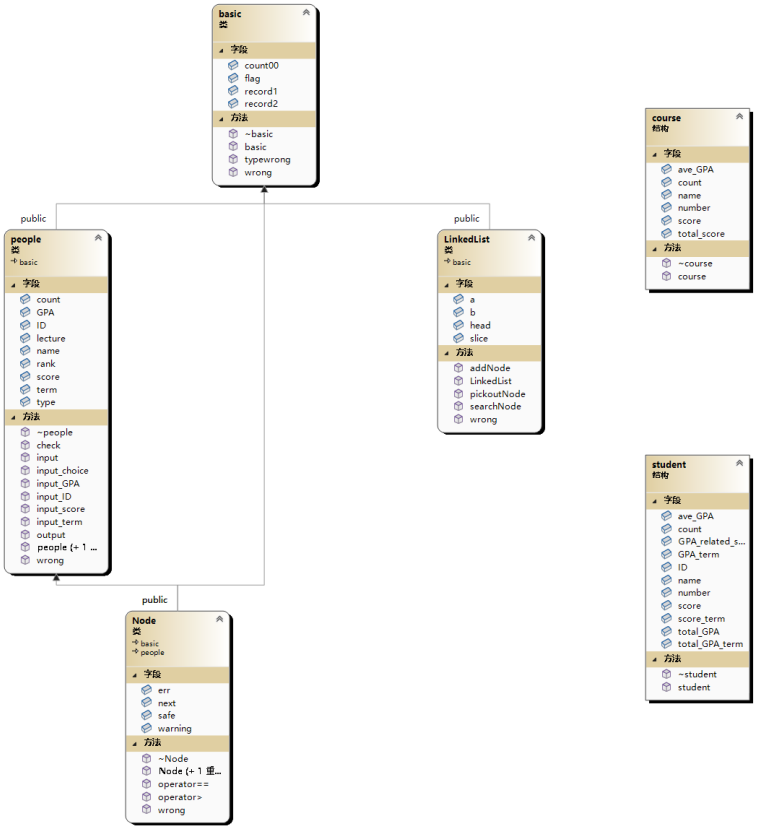


图 2: 学生管理系统的类视图

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。

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。

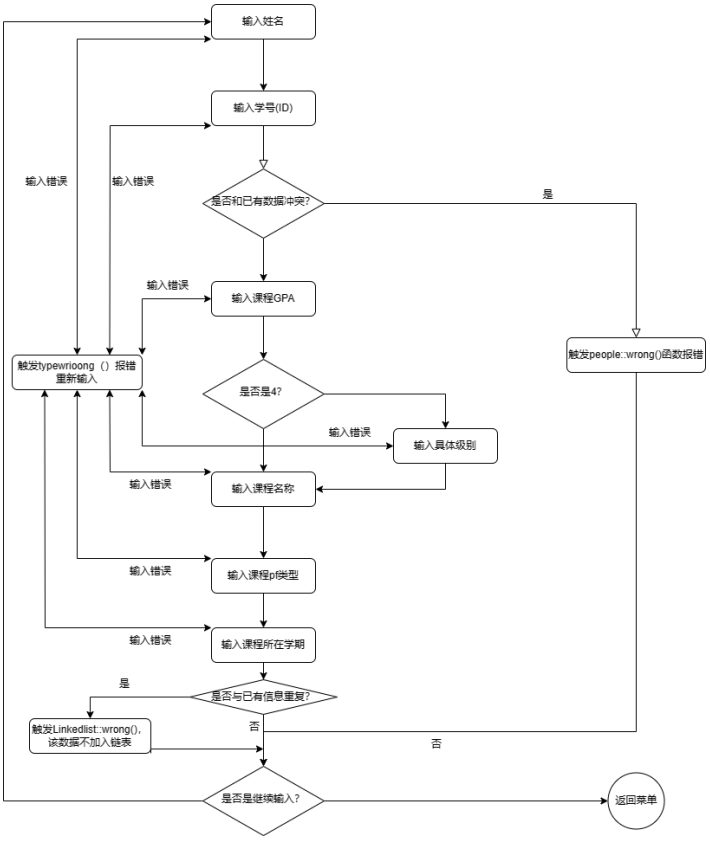


图 3: case1 的流程图

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，如果不继续更改的话退出该函数。

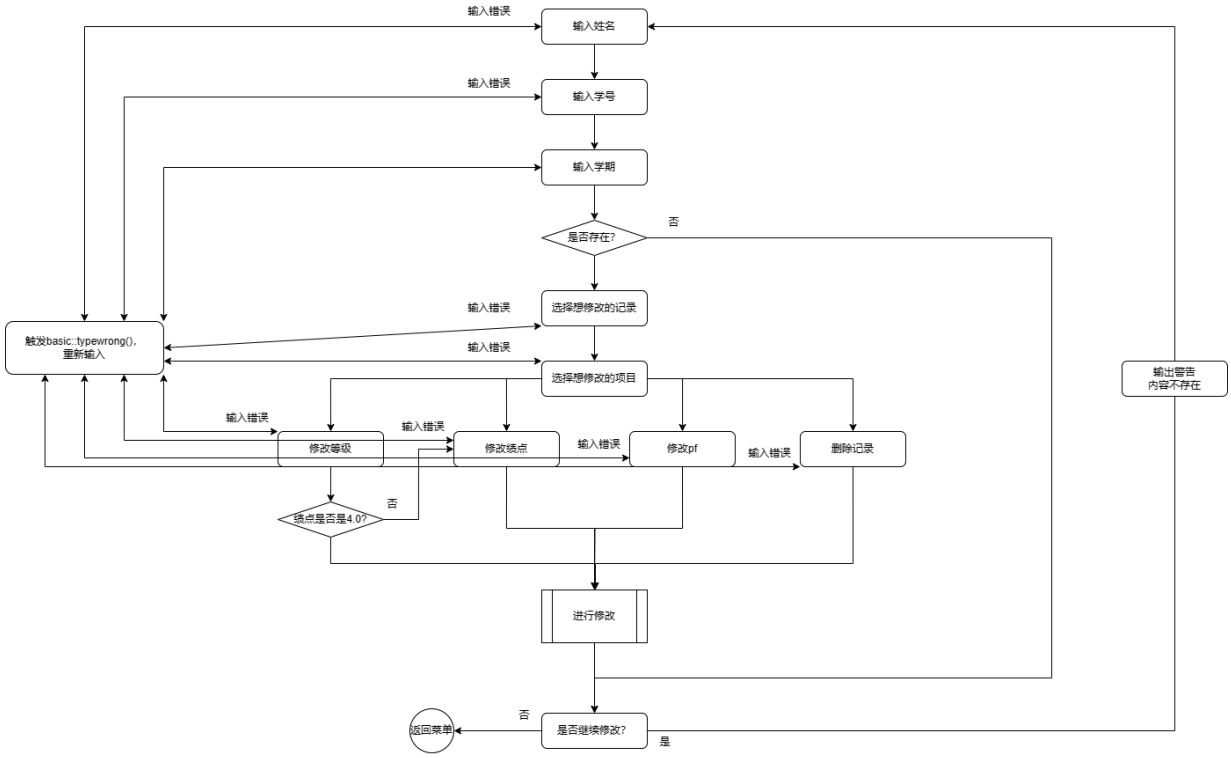


图 4: case2 的流程图

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。

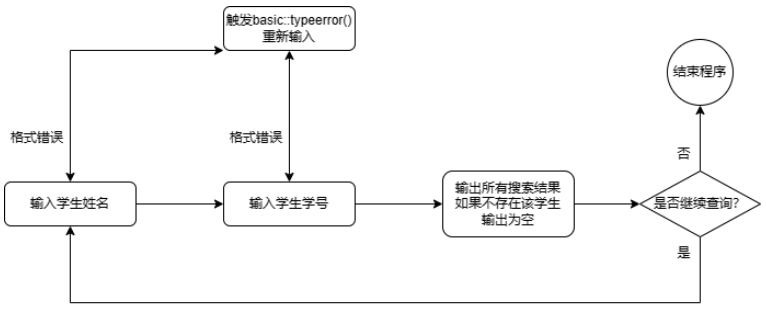


图 5: case3 的流程图

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。

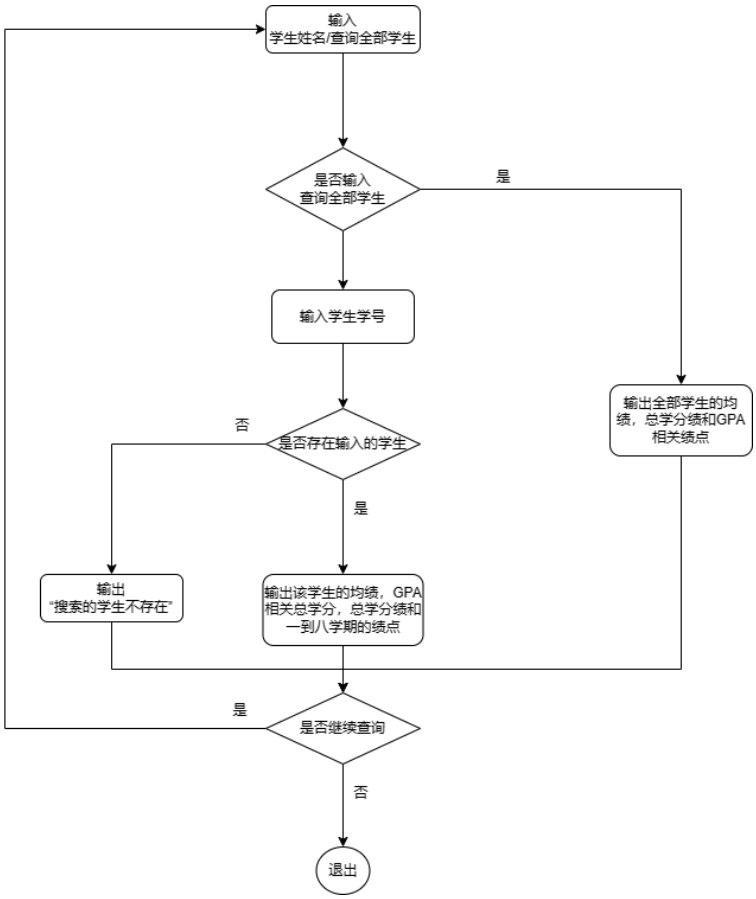


图 6: case4 的流程图

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。

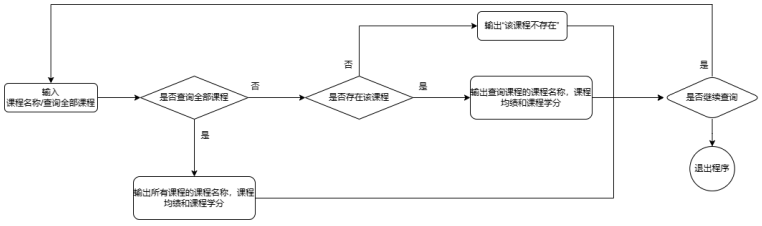


图 7: case5 的流程图

# 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东西，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 case1() 测试图

# 7 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东西，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。

# 8 测试结果与分析

## 8.1 case1() 测试图

```
#include<iostream>
#include<cstring>
#include<windows.h>
#include<fstream>

HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
#pragma warning(disable:4996)
using namespace std;

```

```
10
11 int count_Node = 0;
12 int number_Node = 0;
13
14
15 class basic
16 {
17 public:
18     basic(){}
19     ~basic(){}
20     virtual void wrong() = 0;
21     static void typewrong()
22     {
23         SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY);
24         cout << "cerr:TypeError!您输入的数据类型和要求的不符，请检查后重新输入！" <<
               endl;
25         SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
               FOREGROUND_BLUE);
26     }
27     static int count00;
28     static char record1[10000][30];
29     static char record2[10000][11];
30     static bool flag;
31 };
32
33 int basic::count00 = 0;
34 char basic::record1[10000][30];
35 char basic::record2[10000][11];
36 bool basic::flag = false;
37
38
39 //学生类，里面继承了学生单门课程的信息
40 class people:public basic
41 {
42 public:
43     people() {};
44     ~people() {};
45     void wrong()
46     {
47         SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY);
48         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;
49         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;
50         SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
               FOREGROUND_BLUE);
```

```
51      }
52      people(const people &data) {
53          this->count = data.count;
54          this->score = data.score;
55          strcpy(this->name, data.name);
56          strcpy(this->lecture, data.lecture);
57          this->GPA = data.GPA;
58          strcpy(this->rank, data.rank);
59          this->type = data.type;
60          this->term = data.term;
61          strcpy(this->ID, data.ID);
62      }
63
64      //从键盘输入学生绩点并判断等级类型
65      void input_GPA()
66      {
67
68          cout << "请输入该课程GPA,退课GPA请输入-1, pf请输入0" << endl;
69          cin >> GPA;
70          while (!cin.good()) {
71          cin.clear();
72          cin.ignore(10000, '\n');
73          typewrong();
74          cout << "请输入该课程GPA,退课GPA请输入-1, pf请输入0" << endl;
75          cin >> GPA;
76          }
77          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) {
78              wrong();
79              input_GPA();
80          }
81          else {
82              if (GPA == -1) { strcat(rank, "W"); }
83              if (GPA == 0.0) { strcat(rank, "F"); }
84              if (GPA == 1.0) { strcat(rank, "D-"); }
85              if (GPA == 1.3) { strcat(rank, "D"); }
86              if (GPA == 1.6) { strcat(rank, "D+"); }
87              if (GPA == 2.0) { strcat(rank, "C-"); }
88              if (GPA == 2.3) { strcat(rank, "C"); }
89              if (GPA == 2.6) { strcat(rank, "C+"); }
90              if (GPA == 3.0) { strcat(rank, "B-"); }
91              if (GPA == 3.3) { strcat(rank, "B"); }
92              if (GPA == 3.6) { strcat(rank, "B+"); }
93              if (GPA == 4.0) {
94                  int flag = 0;
```

```
                while (!flag) {
                    flag = 1;
                    cout << "请输入学生等级" << endl;
                    cin >> rank;
                    while (!cin.good())
                        {
                            cin.clear();
                            cin.ignore(10000, '\n');
                            typewrong();
                            cout << "请输入该课程GPA,退课GPA请输入-1, pf请输入0" <<
                                endl;
                            cin >> rank;
                        }
                    if (strcmp(rank, "A") * strcmp(rank, "A-") * strcmp(rank, "A+")) {
                        wrong();
                        flag = 0;
                    }
                }
            }
        }
    }

    //从键盘输入课程学分
    void input_score()
    {
        cout << "请输入课程学分（整数）" << endl;
        cin >> score;
        while (!cin.good()) {
            cin.clear();
            cin.ignore(10000, '\n');
            typewrong();
            cin >> score;
        }
        if (score < 0 || score > 20 || score < 0) {
            wrong();
            input_score();
        }
    }

    //从键盘输入课程pf类型
    void input_choice() {
        cout << "请输入课程类型, pf课程请输入0, 非pf课程请输入1" << endl;
        cin >> type;
        while (!cin.good()) {
            typewrong();
            cin.clear();
```

```
140             cin.ignore(10000, '\n');
141             cin >> type;
142         }
143         while (type != 0 && type != 1) {
144             wrong();
145             cin >> type;
146         }
147     }
148
149     void input_term() {
150         cout << "请输入课程学期名称,大一上为1, 最高为8（大四下）" << endl;
151         cin >> term;
152         while (!cin.good()) {
153             typewrong();
154             cin.clear();
155             cin.ignore(10000, '\n');
156             cin >> term;
157         }
158         if (term != 1 && term != 2 && term != 3 && term != 4 && term != 5 && term != 6
                && term != 7 && term != 8) { wrong(); input_choice(); }
159     }
160
161     void input_ID() {
162         cout << "请输入学生学号(十位)" << endl;
163         cin >> ID;
164         while (!cin.good()) {
165             cin.clear();
166             cin.ignore(10000, '\n');
167             typewrong();
168             cin >> ID;
169         }
170         if (strlen(ID) - 10) {
171             wrong();
172             input_ID();
173         }
174     }
175
176     bool check()
177     {
178         for (int i = 0; i < count00; i++)
179         {
180             if (strcmp(record1[i], name) == 0 && strcmp(record2[i], ID) == 0)
181             {
182                 count--;
183                 return false;
184             }
```

```
185         }
186         strcat(record1[count00], name);
187         strcat(record2[count00], ID);
188         for (int i = 0; i < count00; i++)
189         {
190             if ((strcmp(record1[i], name) != 0 && strcmp(record2[i], ID) == 0))
191             {
192                 *record1[count00] = { 0 };
193                 *record2[count00] = { 0 };
194                 return true;
195             }
196         }
197         return false;
198     }
199     //输入函数，在用户选择1后输出指示，引导用户输入信息
200     int input()
201     {
202         cout << "请输入学生姓名" << endl;
203         cin >> name;
204         while (!cin.good()) {
205             cin.clear();
206             cin.ignore(10000, '\n');
207             typewrong();
208             cin >> name;
209         }
210         cin.clear();
211         cin.ignore(10000, '\n');
212         input_ID();
213         cin.clear();
214         cin.ignore(10000, '\n');
215         if (check()) {
216             *name = { 0 };
217             *ID = { 0 };
218             wrong();
219             return 0;
220         }
221         else
222         {
223             count00++;
224             input_GPA();
225             cin.clear();
226             cin.ignore(10000, '\n');
227             input_score();
228             cin.clear();
229             cin.ignore(10000, '\n');
230             cout << "请输入课程名称" << endl;
```

```
            cin >> lecture;
            while (!cin.good()) {
                cin.clear();
                cin.ignore(10000, '\n');
                typewrong();

                cin >> lecture;
            }
            cin.clear();
            cin.ignore(10000, '\n');
            input_choice();
            cin.clear();
            cin.ignore(10000, '\n');
            input_term();
            cin.clear();
            cin.ignore(10000, '\n');
            return 1;
        }
    }
    //输出函数，查询时输出储存好的信息。
    void output() {
        if (GPA == 4)
        {
            SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_BLUE |
                FOREGROUND_INTENSITY);
            cout << name;
            cout << "      " << ID << "        ";
            cout << lecture;
            cout << "      " << score << "      " << GPA << "      ";
            cout << rank;
            if (type) { cout << "      " << "非pf" << "      " << term << endl; }
            else { cout << "      " << "pf" << "      " << term << endl; }
            SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
                FOREGROUND_BLUE);
        }
        if (GPA < 4 && GPA >= 3)
        {
            SetConsoleTextAttribute(hConsole, FOREGROUND_BLUE | FOREGROUND_INTENSITY);
            cout << name;
            cout << "      " << ID << "        ";
            cout << lecture;
            cout << "      " << score << "      " << GPA << "      ";
            cout << rank;
            if (type) { cout << "      " << "非pf" << "      " << term << endl; }
            else { cout << "      " << "pf" << "      " << term << endl; }
            SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
```

```
                FOREGROUND_BLUE);
275         }
276         if (GPA < 3 && GPA >=2)
277         {
278             SetConsoleTextAttribute(hConsole, FOREGROUND_GREEN | FOREGROUND_BLUE |
                    FOREGROUND_INTENSITY);
279             cout << name;
280             cout << "       " << ID << "         ";
281             cout << lecture;
282             cout << "       " << score << "       " << GPA << "       ";
283             cout << rank;
284             if (type) { cout << "       " << "非pf" << "       " << term << endl; }
285             else { cout << "       " << "pf" << "       " << term << endl; }
286             SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
                    FOREGROUND_BLUE);
287         }
288         if (GPA < 2 && GPA > 0)
289         {
290             SetConsoleTextAttribute(hConsole, FOREGROUND_GREEN | FOREGROUND_INTENSITY)
                    ;
291             cout << name;
292             cout << "       " << ID << "         ";
293             cout << lecture;
294             cout << "       " << score << "       " << GPA << "       ";
295             cout << rank;
296             if (type) { cout << "       " << "非pf" << "       " << term << endl; }
297             else { cout << "       " << "pf" << "       " << term << endl; }
298             SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
                    FOREGROUND_BLUE);
299         }
300         if (GPA <= 0 )
301         {
302             cout << name;
303             cout << "       " << ID << "         ";
304             cout << lecture;
305             cout << "       " << score << "       " << GPA << "       ";
306             cout << rank;
307             if (type) { cout << "       " << "非pf" << "       " << term << endl; }
308             else { cout << "       " << "pf" << "       " << term << endl; }
309         }
310     }
311
312     static int count;  //人数计数
313     int score = -1; //学分
314     char name[30] = {"ZanderZhao"};//姓名
315     char lecture[30] = {"摸鱼课导论"};//课程名
```

```
316     double GPA;//绩点
317     char rank[3] = {};//等级
318     int type = 1;//是否pf
319     int term; //学期
320     char ID[11]; //学生学号
321 };
322 int people::count = 0;
323 
324 //****************************************************下面开始是存储数据的结构体
        ****************************************************
325 
326 //课程类，方便后面的课程排序
327 struct course
328 {
329 public:
330     course() {};
331     ~course() {};
332     static int count; //课程库里一共有几条课程
333     char name[30] = {"摸鱼课导论"}; // 课程名
334     double score = 0.0; // 学分数
335     int number = 0; // 报过的人数
336     double total_score = 0; //总学分绩
337     //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     重新更改，
            这个现在是是否手动置为pf
338     double ave_GPA; //平均分数
339 };
340 int course::count = 0;
341 
342 //学生类，用来记录学生均绩，从而方便后面排序
343 struct student
344 {
345 public:
346     student() {};
347     ~student() {};
348     static int count; //学生人数计数
349     char name[30] = { "nullptr" }; //学生姓名
350     char ID[11] = { "0000000000" };//学生学号
351     int score = 0; //学生总学分
352     double total_GPA = 0; //学生总学分绩
353     double GPA_related_score = 0; //学生绩点相关学分
354     double ave_GPA = 0;  //学生总绩点
355     double score_term[8] = { 0,0,0,0,0,0,0,0 }; //学生学期内总学分
356     double GPA_term[8] = { -1,-1,-1,-1,-1,-1,-1,-1 }; //学生学期内平均绩点
357     double total_GPA_term[8] = { 0,0,0,0,0,0,0,0 }; //学生学期内总绩点
358     int number = 0; //我妥协了，这是学生报的课程数
359 };
```

```
360 int student::count = 0;
361
362 //**************************************************下面开始是链表
        **************************************************
363
364 // 节点类
365 class Node :public basic, public people
366 {
367 public:
368     Node() {};
369     ~Node() {};
370     Node* next = NULL;
371     static int safe;
372     static int warning;
373     char err[40] = { "cerr:警告! 调用了错误的函数实例!" };
374     bool operator>(const Node& temp) {
375     if (strcmp(ID,temp.ID) > 0) { return 1; }
376     else if (strcmp(ID, temp.ID) == 0 && term > temp.term) { return 1; }
377     else if (strcmp(ID, temp.ID) == 0 && term == temp.term && strcmp(name, temp.name)
            > 0) { return 1; }
378     else if (strcmp(ID, temp.ID) == 0 && term == temp.term && strcmp(name, temp.name)
            == 0 && strcmp(lecture, temp.lecture) > 0) { return 1; }
379     else if (strcmp(ID, temp.ID) == 0 && term == temp.term && strcmp(name, temp.name)
            == 0 && strcmp(lecture, temp.lecture) == 0 && GPA > temp.GPA) { return 1; }
380     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) { return 1; }
381     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; }
382     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; }
383     else return 0;
384     }
385     void wrong()
386     {
387         SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY);
388         cout << err << endl;
389         SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
                FOREGROUND_BLUE);
390     }
391     bool operator==(const Node& temp) {
392     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; }
```

```
393     else return 0;
394 }
395 
396     Node(people& data) {
397         this->count = data.count;
398         this->score = data.score;
399         strcpy(this->name,data.name);
400         strcpy(this->lecture, data.lecture);
401         this->GPA = data.GPA;
402         strcpy(this->rank, data.rank);
403         this->type = data.type;
404         this->term = data.term;
405         strcpy(this->ID, data.ID);
406         this->next = nullptr;
407     }
408 };
409 int Node::safe = 0;
410 int Node::warning = 0;
411 
412 
413 
414 
415 
416 // 链表类
417 class LinkedList :public basic {
418 public:
419     Node* head;
420     course a[20000];
421     student b[100000];
422     Node* slice[100] = { NULL };
423     LinkedList() {
424         head = nullptr;
425     }
426     void wrong()
427     {
428         SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY);
429         cout << "该条数据已经存在！" << endl;
430         SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
                FOREGROUND_BLUE);
431         count00--;
432     }
433     // 添加节点(已完成，现链表内数据顺序正确，可以直接输出，学生排序可以从b中找到(基于
        GPA)，课程排序(基于报课人数)可以从a中找到)
434     void addNode(people data) {
435         Node* newNode = new Node(data);
436         Node::safe++;
```

```
437         int i = 1;
438         int flag = 1;
439         char mid[30];
440         double middle;
441         char mid_id[11];
442         if (head == nullptr) {
443             head = newNode;
444             if (basic::flag) cout << "输入成功！" << endl;
445         }
446         else {
447             Node* current = head;
448 
449             //链表插入顺序控制
450 
451             if (*newNode == *current) {
452                 if (basic::flag) wrong();
453                 i = 0;
454                 flag = 0;
455                 Node::safe--;
456             }
457             else if (*newNode > (*current)) {
458                 newNode->next = current;
459                 head = newNode;
460                 i = 0;
461                 if (basic::flag) cout << "输入成功！" << endl;
462             }
463             else if (current->next == nullptr) {
464                 current->next = newNode;
465                 i = 0;
466                 if (basic::flag) cout << "输入成功！" << endl;
467             }
468             else
469             {
470                 while (i && current->next != nullptr)
471                 {
472                     if (*newNode == *current->next)
473                     {
474                         if (basic::flag) wrong();
475                         i = 0;
476                         flag = 0;
477                         Node::safe--;
478                         break;
479                     }
480                     else if (*newNode > *current->next)
481                     {
482                         newNode->next = current->next;
```

```
483                         if (basic::flag) current->next = newNode; i = 0;     cout << "
                                输入成功！" << endl;
484                         current = current->next;
485                     }
486                     else
487                     {
488                         current = current->next;
489                     }
490                 }
491             }
492             if (i) { current->next = newNode;   if (basic::flag) cout << "输入成功！"
                    << endl; }
493         }
494         if (flag) {
495             int flag = 0;
496             //将该课程名字登记在course类中方便记录
497             for (i = 0; i < a[0].count; i++)
498             {
499                 if (!strcmp(a[i].name, data.lecture)) {
500                     a[i].number++;
501                     flag = 1;
502                     a[i].total_score += data.GPA;
503                     a[i].ave_GPA = a[i].total_score / a[i].number;
504                 }
505
506             }
507             if (flag == 0) {
508                 strcpy(a[i].name, data.lecture);
509                 flag = 1;
510                 a[i].count++;
511                 a[i].number++;
512                 a[i].total_score += data.GPA;
513                 a[i].score = data.score;
514                 a[i].ave_GPA = a[i].total_score / a[i].number;
515             }
516
517             flag = 0;
518
519             //将该学生名字登记在student类中方便记录
520             for (i = 0; i < b[0].count; i++)
521             {
522                 if (!strcmp(b[i].name, data.name) && !strcmp(b[i].ID, data.ID)) {
523                     b[i].number++;
524                     b[i].score += data.score;
525                     b[i].GPA_related_score += data.type * data.score;
526                     b[i].total_GPA += data.GPA * data.type * data.score;
```

```
527                     b[i].ave_GPA = b[i].total_GPA / b[i].GPA_related_score;
528                     b[i].total_GPA_term[data.term - 1] += data.GPA * data.type * data.
                            score;
529                     b[i].score_term[data.term - 1] += data.type * data.score;
530                     b[i].GPA_term[data.term - 1] = b[i].total_GPA_term[data.term - 1]
                            / b[i].score_term[data.term - 1];
531                     flag = 1;
532                 }
533             }
534             if (flag == 0) {
535                 strcpy(b[i].name, data.name);
536                 strcpy(b[i].ID, data.ID);
537                 flag = 1;
538                 b[i].number++;
539                 b[i].count++;
540                 b[i].total_GPA += data.GPA * data.type * data.score;
541                 b[i].GPA_related_score += data.type * data.score;
542                 b[i].ave_GPA = b[i].total_GPA / b[i].GPA_related_score;
543                 b[i].score += data.score;
544                 b[i].total_GPA_term[data.term - 1] += data.GPA * data.type * data.
                        score;
545                 b[i].score_term[data.term - 1] += data.type * data.score;
546                 b[i].GPA_term[data.term - 1] = b[i].total_GPA_term[data.term - 1] / b[
                        i].score_term[data.term - 1];
547             }
548
549             flag = 0;
550
551             //重新排序course类课程 base on people number
552             while (!flag)
553             {
554                 flag = 1;
555                 for (i = 0; i < a[0].count - 1; i++)
556                 {
557                     if (a[i].number < a[i].number) {
558                         strcpy(mid, a[i].name);
559                         strcpy(a[i].name, a[i + 1].name);
560                         strcpy(a[i + 1].name, mid);
561                         middle = a[i].number;
562                         a[i].number = a[i + 1].number;
563                         a[i + 1].number = middle;
564                         flag = 0;
565                     }
566                     if (a[i].number == a[i].number && strcmp(a[i].name, a[i].name) <
                            0) {
567                         strcpy(mid, a[i].name);
```

```
568                     strcpy(a[i].name, a[i + 1].name);
569                     strcpy(a[i + 1].name, mid);
570                     middle = a[i].number;
571                     a[i].number = a[i + 1].number;
572                     a[i + 1].number = middle;
573                     flag = 0;
574                 }
575             }
576         }
577 
578         flag = 0;
579 
580         //重新排序学生姓名 base on GPA
581 
582         while (!flag)
583         {
584             flag = 1;
585             for (i = 0; i < b[0].count - 1; i++)
586             {
587                 if (b[i].ave_GPA < b[i].ave_GPA) {
588                     strcpy(mid, b[i].name);
589                     strcpy(b[i].name, b[i + 1].name);
590                     strcpy(b[i + 1].name, mid);
591                     strcpy(mid_id, b[i].ID);
592                     strcpy(b[i].ID, b[i + 1].ID);
593                     strcpy(b[i + 1].ID, mid_id);
594                     middle = b[i].score;
595                     b[i].score = b[i + 1].score;
596                     b[i + 1].score = middle;
597                     middle = b[i].total_GPA;
598                     b[i].total_GPA = b[i + 1].total_GPA;
599                     b[i + 1].total_GPA = middle;
600                     middle = b[i].GPA_related_score;
601                     b[i].GPA_related_score = b[i + 1].GPA_related_score;
602                     b[i + 1].GPA_related_score = middle;
603                     middle = b[i].ave_GPA;
604                     b[i].ave_GPA = b[i + 1].ave_GPA;
605                     b[i + 1].ave_GPA = middle;
606                     flag = 0;
607                 }
608                 else if (b[i].ave_GPA == b[i].ave_GPA && strcmp(b[i].ID, b[i].ID)
                        < 0) {
609                     strcpy(mid, b[i].name);
610                     strcpy(b[i].name, b[i + 1].name);
611                     strcpy(b[i + 1].name, mid);
612                     strcpy(mid_id, b[i].ID);
```

```
613                         strcpy(b[i].ID, b[i + 1].ID);
614                         strcpy(b[i + 1].ID, mid_id);
615                         middle = b[i].score;
616                         b[i].score = b[i + 1].score;
617                         b[i + 1].score = middle;
618                         middle = b[i].total_GPA;
619                         b[i].total_GPA = b[i + 1].total_GPA;
620                         b[i + 1].total_GPA = middle;
621                         middle = b[i].GPA_related_score;
622                         b[i].GPA_related_score = b[i + 1].GPA_related_score;
623                         b[i + 1].GPA_related_score = middle;
624                         middle = b[i].ave_GPA;
625                         b[i].ave_GPA = b[i + 1].ave_GPA;
626                         b[i + 1].ave_GPA = middle;
627                         flag = 0;
628                     }
629                     else if (b[i].ave_GPA == b[i].ave_GPA && strcmp(b[i].ID, b[i].ID)
                            == 0 && strcmp(b[i].ID, b[i].ID) < 0) {
630                         strcpy(mid, b[i].name);
631                         strcpy(b[i].name, b[i + 1].name);
632                         strcpy(b[i + 1].name, mid);
633                         strcpy(mid_id, b[i].ID);
634                         strcpy(b[i].ID, b[i + 1].ID);
635                         strcpy(b[i + 1].ID, mid_id);
636                         middle = b[i].score;
637                         b[i].score = b[i + 1].score;
638                         b[i + 1].score = middle;
639                         middle = b[i].total_GPA;
640                         b[i].total_GPA = b[i + 1].total_GPA;
641                         b[i + 1].total_GPA = middle;
642                         middle = b[i].GPA_related_score;
643                         b[i].GPA_related_score = b[i + 1].GPA_related_score;
644                         b[i + 1].GPA_related_score = middle;
645                         middle = b[i].ave_GPA;
646                         b[i].ave_GPA = b[i + 1].ave_GPA;
647                         b[i + 1].ave_GPA = middle;
648                         flag = 0;
649                     }
650                 }
651             }
652         }
653         count_Node = Node::safe;
654     }
655     int searchNode(const char name[30], const char ID[11], int term) {
656         Node* current = head;
657         int j = -1;
```

```
658         int i = 0;
659         while (current != NULL)
660         {
661         if (strcmp(current->name, name) == 0 && strcmp(current->ID, ID) == 0 && term
                == current->term)
662         {
663             cout << i + 1 << "    ";
664             current->output();
665             slice[i] = current;
666             i++;
667         }
668         j = i - 1;
669         current = current->next;
670         }
671         return j;
672     }
673     people pickoutNode(Node t) {
674         Node* current = head;
675         if (*head == t)
676         {
677             current = head->next;
678             head->next = NULL;
679             head = current;
680         }
681         else
682         {
683             while (current->next != NULL && !(*current->next == t)){
684                 current = current->next;
685             }
686             current->next = t.next;
687             t.next = NULL;
688         }
689         int flag = 1;
690         if (flag) {
691             int i;
692             //将该课程名字登记在course类中方便记录
693             for (i = 0; i < a[0].count; i++)
694             {
695                 if (!strcmp(a[i].name, t.lecture)) {
696                     a[i].number--;
697                     if (a[i].number == 0)
698                     {
699                         a[i].count--;
700                         strcpy(a[i].name, "摸鱼课导论");
701                     }
702                     a[i].total_score -= t.GPA;
```

```
703                 if (a[i].number != 0)
704                 {
705                     a[i].ave_GPA = a[i].total_score / a[i].number;
706                     a[i].count--;
707                 }
708                 else a[i].ave_GPA = 0;
709             }
710         }
711 
712         //将该学生名字登记在student类中方便记录
713         for (i = 0; i < b[0].count; i++)
714         {
715             if (!strcmp(b[i].name, t.name) && !strcmp(b[i].ID, t.ID)) {
716                 b[i].number--;
717 
718                 b[i].score -= t.score;
719                 b[i].GPA_related_score -= t.type * t.score;
720                 b[i].total_GPA -= t.GPA * t.type * t.score;
721 
722                 b[i].total_GPA_term[t.term - 1] -= t.GPA * t.type * t.score;
723                 b[i].score_term[t.term - 1] -= t.type * t.score;
724                 if (!b[i].number)
725                 {
726                     strcpy(b[i].name, "nullptr");
727                     strcpy(b[i].ID, "1111111111");
728                     b[i].ave_GPA = 0;
729                     b[i].GPA_term[t.term - 1] = 0;
730                     b[i].count--;
731                 }
732                 else
733                 {
734                 b[i].ave_GPA = b[i].total_GPA / b[i].GPA_related_score;
735                 b[i].GPA_term[t.term - 1] = b[i].total_GPA_term[t.term - 1] / b[i
                        ].score_term[t.term - 1];
736                 }
737             }
738         }
739 
740         flag = 0;
741         //b[i].ave_GPA = b[i].total_GPA / b[i].GPA_related_score;
742         //b[i].GPA_term[t.data.term - 1] = b[i].total_GPA_term[t.data.term - 1] /
                b[i].score_term[t.data.term - 1];
743         //重新排序course类课程 base on people number
744         char mid[30];
745         char mid_id[11];
746         double middle;
```

```
747             while (!flag)
748             {
749                 flag = 1;
750                 for (i = 0; i < a[0].count - 1; i++)
751                 {
752                     if (a[i].number < a[i].number) {
753                         strcpy(mid, a[i].name);
754                         strcpy(a[i].name, a[i + 1].name);
755                         strcpy(a[i + 1].name, mid);
756                         middle = a[i].number;
757                         a[i].number = a[i + 1].number;
758                         a[i + 1].number = middle;
759                         flag = 0;
760                     }
761                     if (a[i].number == a[i].number && strcmp(a[i].name, a[i].name) <
                            0) {
762                         strcpy(mid, a[i].name);
763                         strcpy(a[i].name, a[i + 1].name);
764                         strcpy(a[i + 1].name, mid);
765                         middle = a[i].number;
766                         a[i].number = a[i + 1].number;
767                         a[i + 1].number = middle;
768                         flag = 0;
769                     }
770                 }
771             }
772
773             flag = 0;
774
775             //重新排序学生姓名 base on GPA
776
777             while (!flag)
778             {
779                 flag = 1;
780                 for (i = 0; i < b[0].count - 1; i++)
781                 {
782                     if (b[i].ave_GPA < b[i].ave_GPA) {
783                         strcpy(mid, b[i].name);
784                         strcpy(b[i].name, b[i + 1].name);
785                         strcpy(b[i + 1].name, mid);
786                         strcpy(mid_id, b[i].ID);
787                         strcpy(b[i].ID, b[i + 1].ID);
788                         strcpy(b[i + 1].ID, mid_id);
789                         middle = b[i].score;
790                         b[i].score = b[i + 1].score;
791                         b[i + 1].score = middle;
```

```
792                     middle = b[i].total_GPA;
793                     b[i].total_GPA = b[i + 1].total_GPA;
794                     b[i + 1].total_GPA = middle;
795                     middle = b[i].GPA_related_score;
796                     b[i].GPA_related_score = b[i + 1].GPA_related_score;
797                     b[i + 1].GPA_related_score = middle;
798                     middle = b[i].ave_GPA;
799                     b[i].ave_GPA = b[i + 1].ave_GPA;
800                     b[i + 1].ave_GPA = middle;
801                     flag = 0;
802                 }
803                 else if (b[i].ave_GPA == b[i].ave_GPA && strcmp(b[i].ID, b[i].ID)
                        < 0) {
804                     strcpy(mid, b[i].name);
805                     strcpy(b[i].name, b[i + 1].name);
806                     strcpy(b[i + 1].name, mid);
807                     strcpy(mid_id, b[i].ID);
808                     strcpy(b[i].ID, b[i + 1].ID);
809                     strcpy(b[i + 1].ID, mid_id);
810                     middle = b[i].score;
811                     b[i].score = b[i + 1].score;
812                     b[i + 1].score = middle;
813                     middle = b[i].total_GPA;
814                     b[i].total_GPA = b[i + 1].total_GPA;
815                     b[i + 1].total_GPA = middle;
816                     middle = b[i].GPA_related_score;
817                     b[i].GPA_related_score = b[i + 1].GPA_related_score;
818                     b[i + 1].GPA_related_score = middle;
819                     middle = b[i].ave_GPA;
820                     b[i].ave_GPA = b[i + 1].ave_GPA;
821                     b[i + 1].ave_GPA = middle;
822                     flag = 0;
823                 }
824                 else if (b[i].ave_GPA == b[i].ave_GPA && strcmp(b[i].ID, b[i].ID)
                        == 0 && strcmp(b[i].ID, b[i].ID) < 0) {
825                     strcpy(mid, b[i].name);
826                     strcpy(b[i].name, b[i + 1].name);
827                     strcpy(b[i + 1].name, mid);
828                     strcpy(mid_id, b[i].ID);
829                     strcpy(b[i].ID, b[i + 1].ID);
830                     strcpy(b[i + 1].ID, mid_id);
831                     middle = b[i].score;
832                     b[i].score = b[i + 1].score;
833                     b[i + 1].score = middle;
834                     middle = b[i].total_GPA;
835                     b[i].total_GPA = b[i + 1].total_GPA;
```

```
                        b[i + 1].total_GPA = middle;
                        middle = b[i].GPA_related_score;
                        b[i].GPA_related_score = b[i + 1].GPA_related_score;
                        b[i + 1].GPA_related_score = middle;
                        middle = b[i].ave_GPA;
                        b[i].ave_GPA = b[i + 1].ave_GPA;
                        b[i + 1].ave_GPA = middle;
                        flag = 0;
                    }
                }
            }
        }
        return people(t);
}
};
LinkedList total;


//*****************************************************下面开始是函数
    ***************************************************

// 给定需要输出的学生姓名，遍历链表并打印节点的值

void output(char* a, char* id)
{
    Node* current = total.head;
    while (current != nullptr) {

        if (strcmp(current->name, a) == 0 && strcmp(current->ID, id) == 0) current->
            output();
        current = current->next;
    }
    cout << endl;
}

// 输出欢迎栏
void welcome()
{
    cout << "*-------------欢迎访问学生成绩系统！-------------*" << endl;
    cout << "*------------录入学生的成绩单请输入1-------------*" << endl;
    cout << "*------------修改学生的成绩单请输入2-------------*" << endl;
    cout << "*------------查询学生的成绩单请输入3-------------*" << endl;
    cout << "*-------------查询学生的绩点请输入4--------------*" << endl;
    cout << "*-------------查询课程的均绩请输入5--------------*" << endl;
    cout << "*----------------按任意键退出程序----------------*" << endl;
}
```

```
880
881 //提前说明函数
882 void case1();
883 void case2();
884 void case3();
885 void case4();
886 void case5();
887 void finish();
888 //匹配函数
889 void pipei()
890 {
891     int a;
892     cin >> a;
893     switch (a)
894     {
895     case 1:
896         case1();
897     case 2:
898         case2();
899     case 3:
900         case3();
901     case 4:
902         case4();
903     case 5:
904         case5();
905     default:
906         finish();
907         cout << "退出成功！" << endl;
908         if (Node::safe != Node::warning)
909         {
910             SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY);
911             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！" <<
                    endl;
912             SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
                    FOREGROUND_BLUE);
913         }
914         exit(0);
915     }
916 }
917
918 //公用输入函数，将输入输入到链表里面（未完成，未增加排序）
919 void shuru()
920 {
921     int flag;
922     people a;
923     flag = a.input();
```

```
924     if (flag) total.addNode(a);
925 }
926 
927 
928 //录入成绩单函数 //已完成
929 void case1()
930 {
931     shuru();
932     cout << "是否继续输入？（y/n）" << endl;
933     char a[2];
934     cin >> a;
935     while (!cin.good())
936     {
937         cin.clear();
938         cin.ignore(10000, '\n');
939         basic::typewrong();
940         cin >> a;
941     }
942     cin.clear();
943     cin.ignore(10000, '\n');
944     if (!strcmp(a, "y")) case1();
945     else {
946         welcome();
947         pipei();
948     }
949 }
950 
951 //修改成绩单函数(未实现)
952 void case2()
953 {
954     int term;
955     char name[30];
956     char ID[11];
957     int j;
958     int choose;
959     int match;
960     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;
961     cout << "请输入想要修改的学生姓名" << endl;
962     cin >> name;
963     while (!cin.good())
964     {
965         cin.clear();
966         cin.ignore(10000, '\n');
967         basic::typewrong();
968         cin >> name;
969     }
```

```
970     cout << "请输入该学生的学号" << endl;
971     cin >> ID;
972     while (!cin.good())
973     {
974         cin.clear();
975         cin.ignore(10000, '\n');
976         basic::typewrong();
977         cin >> ID;
978     }
979     cout << "请输入想要修改的学期" << endl;
980     cin >> term;
981     while (!cin.good())
982     {
983         cin.clear();
984         cin.ignore(10000, '\n');
985         basic::typewrong();
986         cin >> term;
987     }
988     cout << "选项 姓名          学号            课程名称      学分    绩点    等级    pf
        类型    学期" << endl;
989     j = total.searchNode(name, ID,term);
990     if (j != -1) {
991         cout << "请选择想要修改的记录(敲击记录前的数字)" << endl;
992         cin >> choose;
993         choose = choose - 1;
994         while (choose > j || choose < 0)
995         {
996             basic::typewrong();
997             cout << "请选择想要修改的记录(敲击记录前的数字)" << endl;
998             cin >> choose;
999         }
1000        people a;
1001        a = total.pickoutNode(*total.slice[choose]);
1002        a.output();
1003        //total.slice[choose]->data.output();
1004        cout << "请选择想要修改的项目" << endl;
1005        cout << "1.修改学生等级" << endl;
1006        cout << "2.修改课程绩点" << endl;
1007        cout << "3.修改课程pf" << endl;
1008        cout << "4.删除该记录" << endl;
1009        cin >> match;
1010        cin.clear();
1011        cin.ignore(10000, '\n');
1012        while (match != 1 && match != 2 && match != 3 && match != 4)
1013        {
1014            basic::typewrong();
```

```
1015            cout << "请选择想要修改的项目" << endl;
1016            cout << "1.修改学生等级" << endl;
1017            cout << "2.修改课程绩点" << endl;
1018            cout << "3.修改课程pf" << endl;
1019            cin >> match;
1020            cin.clear();
1021            cin.ignore(10000, '\n');
1022        }
1023        if (match == 1)
1024        {
1025            if (total.slice[choose]->GPA != 4.0) {
1026                SetConsoleTextAttribute(hConsole, FOREGROUND_RED |
                        FOREGROUND_INTENSITY);
1027                cout << "无法修改等级！请先修改绩点" << endl;
1028                SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
                        FOREGROUND_BLUE);
1029                match = 2;
1030            }
1031            else {
1032                cout.flush();
1033                cout << "请输入新的等级" << endl;
1034                char temp[3];
1035                cin >> temp;
1036                while (strcmp(temp, "A-") * strcmp(temp, "A+") * strcmp(temp, "A") !=
                        0)
1037                {
1038                    basic::typewrong();
1039                    cin >> temp;
1040                }
1041                a.rank[0] = NULL;
1042                a.rank[1] = NULL;
1043                a.rank[2] = NULL;
1044                strcpy( a.rank,temp);
1045                total.addNode(a);
1046            }
1047        }
1048        if (match == 2)
1049        {
1050            a.rank[0] = NULL;
1051            a.rank[1] = NULL;
1052            a.rank[2] = NULL;
1053            a.input_GPA();
1054            total.addNode(a);
1055        }
1056        if (match == 3)
1057        {
```

```
1058                a.input_choice();
1059                total.addNode(a);
1060            }
1061            if (match == 4)
1062            {
1063                a.~people();
1064                Node::safe--;
1065                count_Node = Node::safe;
1066            }
1067            welcome();
1068            pipei();
1069        }
1070        else {
1071            SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY);
1072            cout << "不存在该学生的记录！" << endl;
1073            SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
                    FOREGROUND_BLUE);
1074            welcome();
1075            pipei();
1076        }
1077        cout << "是否继续查询？（y/n）" << endl;
1078        char a[2];
1079        cin >> a;
1080        while (!cin.good())
1081        {
1082            cin.clear();
1083            cin.ignore(10000, '\n');
1084            basic::typewrong();
1085            cin >> a;
1086        }
1087        cin.clear();
1088        cin.ignore(10000, '\n');
1089        if (!strcmp(a, "y")) case2();
1090        else {
1091            welcome();
1092            pipei();
1093        }
1094    }
1095
1096    //查询成绩单函数 //已完成
1097    void case3()
1098    {
1099        char a[30];
1100        cout << "请输入查询学生姓名" << endl;
1101        cin >> a;
1102        while (!cin.good())
```

```
1103        {
1104            cin.clear();
1105            cin.ignore(10000, '\n');
1106            basic::typewrong();
1107            cin >> a;
1108        }
1109        char id[11];
1110        cout << "请输入查询学生学号" << endl;
1111        cin >> id;
1112        while (!cin.good())
1113        {
1114            cin.clear();
1115            cin.ignore(10000, '\n');
1116            basic::typewrong();
1117            cin >> id;
1118        }
1119        cout << " 姓名          学号          课程名称      学分    绩点    等级    pf类型
                学期" << endl;
1120        output(a,id);
1121
1122        cout << "输出完成！" << endl;
1123        cout << "是否继续查询？（y/n）" << endl;
1124        char b[2];
1125        cin >> b;
1126        while (!cin.good())
1127        {
1128            cin.clear();
1129            cin.ignore(10000, '\n');
1130            basic::typewrong();
1131            cin >> b;
1132        }
1133        cin.clear();
1134        cin.ignore(10000, '\n');
1135        if (!strcmp(b, "y")) case3();
1136        else {
1137            welcome();
1138            pipei();
1139        }
1140    }
1141
1142    //查询均绩函数
1143    void case4()
1144    {
1145        int i;
1146        char name[20];
1147        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;
```

```
1148        cin >> name;
1149        if (!strcmp(name, "查询全部学生"))
1150        {
1151            cout << "学生姓名" << "         " << "学生学号" << "       " << "学生均绩" << "
                    " << "GPA相关总学分" << "     " << "总学分绩" << "      " << endl;
1152            for (i = 0; i < total.b[0].count; i++)
1153            {
1154                cout << total.b[i].name << "           " << total.b[i].ID << "          " <<
                         total.b[i].ave_GPA << "               " << total.b[i].
                        GPA_related_score << "               " << total.b[i].total_GPA << endl;
1155            }
1156        }
1157        else
1158        {
1159            char ID[11];
1160            cout << "请输入学生学号(十位)" << endl;
1161            cin >> ID;
1162            while (!cin.good()) {
1163                cin.clear();
1164                cin.ignore(10000, '\n');
1165                basic::typewrong();
1166                cin >> ID;
1167            }
1168            for (i = 0; i < total.b[0].count; i++)
1169            {
1170                if (!strcmp(name, total.b[i].name) && !strcmp(ID, total.b[i].ID)) break;
1171            }
1172            if (i < total.b[0].count)
1173            {
1174                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;
1175                cout << total.b[i].name << "          " << total.b[i].ID << "         " <<
                        total.b[i].ave_GPA << "       " << total.b[i].GPA_related_score << "
                             " << total.b[i].total_GPA << "       ";
1176                for (int j = 0; j < 8; j++)
1177                {
1178                    cout << total.b[i].GPA_term[j] << "      ";
1179                }
1180                cout << endl;
1181            }
1182            else
1183            {
1184                SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY);
```

```
1185                cout << "搜索的信息不存在！" << endl;
1186                SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
                        FOREGROUND_BLUE);
1187            }
1188        }
1189        cout << "是否继续查询？（y/n）" << endl;
1190        char a[2];
1191        cin >> a;
1192        while (!cin.good())
1193        {
1194            cin.clear();
1195            cin.ignore(10000, '\n');
1196            basic::typewrong();
1197            cin >> a;
1198        }
1199        cin.clear();
1200        cin.ignore(10000, '\n');
1201        if (!strcmp(a, "y")) case4();
1202        else {
1203            welcome();
1204            pipei();
1205        }
1206    }
1207
1208    //查询课程函数
1209    void case5()
1210    {
1211        int i;
1212        char name[20];
1213        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;
1214        cin >> name;
1215        if (!strcmp(name, "查询全部课程"))
1216        {
1217            for (i = 0; i < total.a[0].count; i++)
1218            {
1219                cout << " " << "课程名称" << "        " << "课程均绩" << "         " << "课
                        程学分" << " " << endl;
1220                cout << total.a[i].name << "      " << total.a[i].ave_GPA << "          " <<
                         total.a[i].score /* << " " << total.a[i].type*/ << endl;
1221            }
1222        }
1223        else
1224        {
1225            for (i = 0; i < total.a[0].count; i++)
1226            {
1227                if (!strcmp(name, total.a[i].name)) break;
```

```
1228            }
1229            if (i == total.a[0].count) {
1230                SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY);
1231                cout << "未查到该课程！" << endl;
1232                SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
                        FOREGROUND_BLUE);
1233            }
1234            else {
1235                cout << "  课程名称" << "        " << "课程均绩" << "         " << "课程学分
                        " << " " << endl;
1236                cout << total.a[i].name << "      " << total.a[i].ave_GPA << "          " <<
                         total.a[i].score  << endl;
1237            }
1238        }
1239        cout << "是否继续查询？（y/n）" << endl;
1240        char a[2];
1241        cin >> a;
1242        while (!cin.good())
1243        {
1244            cin.clear();
1245            cin.ignore(10000, '\n');
1246            basic::typewrong();
1247            cin >> a;
1248        }
1249        cin.clear();
1250        cin.ignore(10000, '\n');
1251        if (!strcmp(a, "y")) case5();
1252        else {
1253            welcome();
1254            pipei();
1255        }
1256    }
1257
1258    static void prepare()
1259    {
1260        fstream number("number.txt", ios::in); //全局变量学号存储文件
1261        fstream node("node.txt", ios::in); //链表节点存储文件
1262        if (!node.is_open()) {
1263            SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY);
1264            cout << "链表节点存储文件打开或创建失败！" << endl;
1265            SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
                    FOREGROUND_BLUE);
1266        }
1267
1268        if (!number.is_open()) {
1269            SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY);
```

```
1270            cout << "学号存储文件打开或创建失败！" << endl;
1271            SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
                    FOREGROUND_BLUE);
1272        }
1273        number.clear();
1274        node.clear();
1275        while (!node.eof())
1276        {
1277            people temp;
1278            node >> temp.name;
1279            if (node.eof()) break;
1280            node.ignore(1);
1281            node >> temp.ID;
1282            node.ignore(1);
1283            node >> temp.GPA;
1284            node.ignore(1);
1285            node >> temp.rank;
1286            node.ignore(1);
1287            node >> temp.score;
1288            node.ignore(1);
1289            node >> temp.lecture;
1290            node.ignore(1);
1291            node >> temp.type;
1292            node.ignore(1);
1293            node >> temp.term;
1294            node.ignore(1);
1295            total.addNode(temp);
1296        }
1297        int count_temp = 0;
1298        while (!number.eof())
1299        {
1300            number >> basic::record1[count_temp];
1301            if (number.eof()) break;
1302            node.ignore(1);
1303            number >> basic::record2[count_temp];
1304            node.ignore(1);
1305            count_temp++;
1306            basic::count00++;
1307        }
1308        node.close();
1309        number.close();
1310        basic::flag = true;
1311    }
1312
1313    //问题查明未修改：重复输入一人的信息时重复粘贴
1314    static void finish()
```

```
{
    fstream number("number.txt", ios::trunc | ios::out); //全局变量学号存储文件
    fstream node("node.txt", ios::trunc | ios::out); //链表节点存储文件
    number.clear();
    node.clear();
    if (!node.is_open()) {
        SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY);
        cout << "链表节点存储文件打开或创建失败！" << endl;
        SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
            FOREGROUND_BLUE);
    }
    if (!number.is_open()) {
        SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_INTENSITY);
        cout << "学号存储文件打开或创建失败！" << endl;
        SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
            FOREGROUND_BLUE);
    }
    Node* current = total.head;
    while (current != nullptr)
    {
        node << current->name;
        node << " ";
        node << current->ID;
        node << " ";
        node << current->GPA;
        node << " ";
        node << current->rank;
        node << " ";
        node << current->score;
        node << " ";
        node << current->lecture;
        node << " ";
        node << current->type;
        node << " ";
        node << current->term;
        node << endl;
        current = current->next;
        Node::warning++;
        number_Node = Node::warning;
    }
    int count = 0;
    while (count < basic::count00)
    {
        number << basic::record1[count];
        number << " ";
        number << basic::record2[count];
```

```
1359            number << endl;
1360            count++;
1361        }
1362        node.close();
1363        number.close();
1364    }
1365    //主函数
1366    int main()
1367    {
1368        SetConsoleTextAttribute(hConsole, FOREGROUND_RED | FOREGROUND_GREEN |
                FOREGROUND_BLUE);
1369        prepare();
1370        welcome();
1371        pipei();
1372    }
```

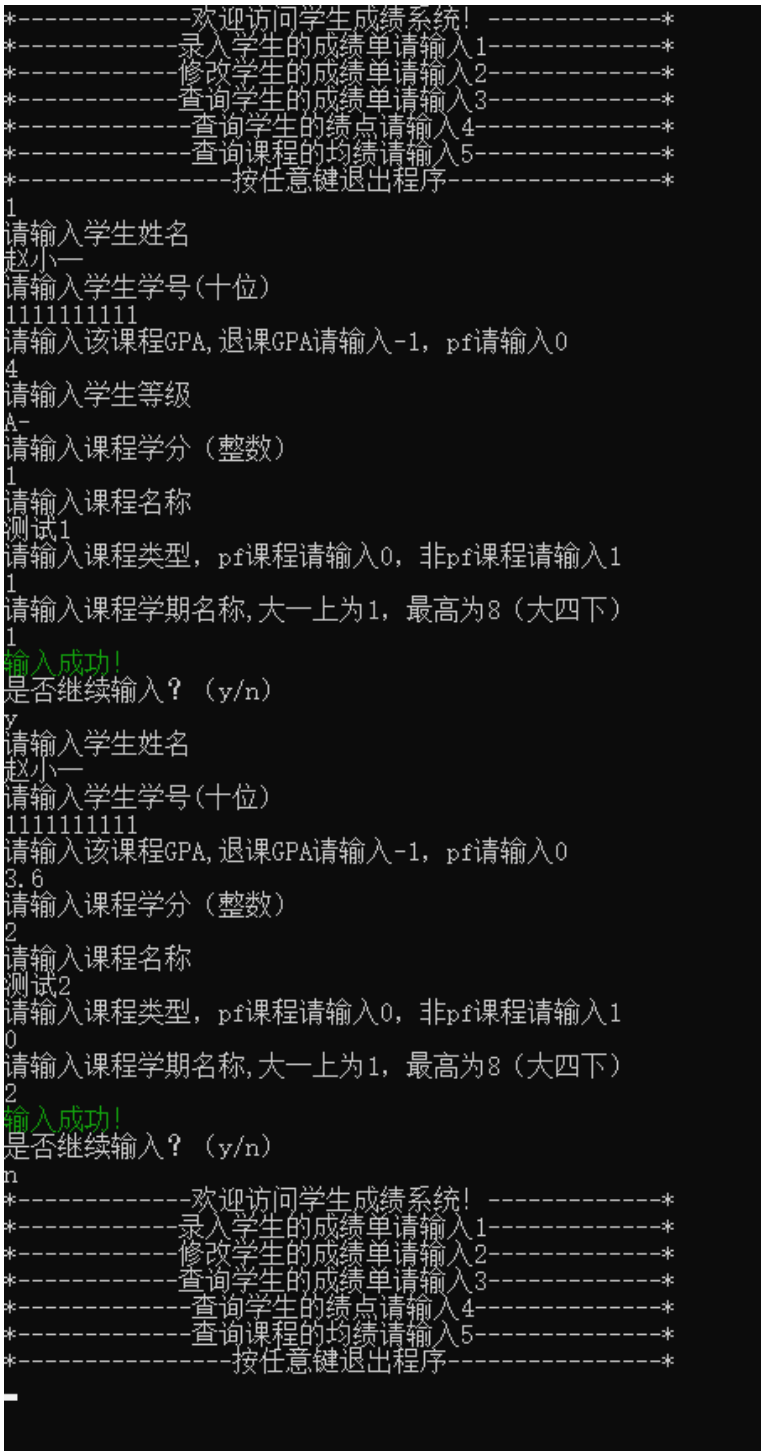


图 8: case5 的流程图

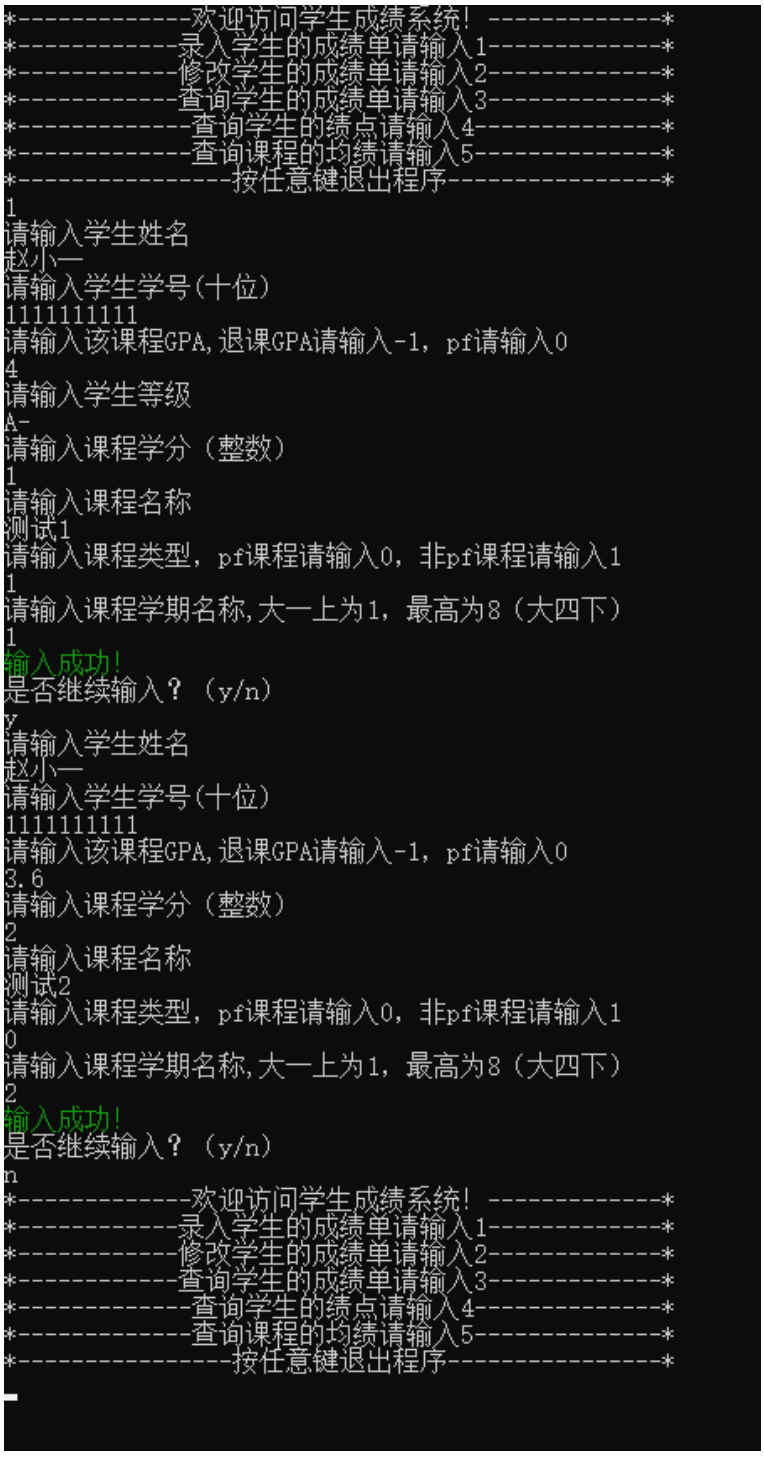


图 9: case5 的流程图