

C++程序设计课程设计

课设报告

**2022/2023(2)**



实验题目 学校开课查询系统

学生姓名 金宣成

学生学号 202203151310

学生班级 计算机科学技术03班

任课教师 毛国红

提交日期

**计算机科学与技术学院**

**学校开课查询系统 实验报告**

1. **大型实验的内容**

学校开课查询系统(CMS:Course Management System)旨在为学生和学校管理员提供一个便捷的方式来查询、更改学校课程。该系统提供管理课程、课程评价、选课退课等功能，并通过菜单实现功能选择。管理员可以使用该系统对课程信息进行输入、修改、删除操作，并可以按照给定的条件 (如编号、名称、任课教师、开课院系等) 进行查询和排序整理。学生可以通过该系统来选课退课，并且给予课程评价。

1. **运行环境**

学校开课查询系统在Visual Studio code1.77.3下开发。

操作系统：

Windows11。

硬件环境：

处理器：AMD Ryzen 7 6800HS with Radeon Graphics 3.20 GHz

内存：16.0 GB (15.2 GB 可用)

系统类型：64位操作系统

1. **实验课题分析**

**3.1学校开课查询系统的主要功能**

**学校开课查询系统(CSM)主要功能为：**学生登陆、管理员登陆，查询课程、对课程进行排序、增删，选课、退课，给予课程评价。详细的系统功能结构见图1。



图1 功能流程图

依据图1，可将系统分为三个模块：登陆模块、学生操作模块、管理员操作模块：

****

图2 功能模块图

**系统各模块的功能具体描述为：**

1. **登陆模块**

选择以学生身份、或以管理员身份登陆。选择成功后进行身份验证，即输入账号密码，若输入正确则进入下一个相应模块，输入错误则再次输入。

1. **学生操作模块**

学生操作模块可以进行选课退课操作，同时也可以给予课程评价，评价为百分制。

查询课程功能：进入查询课程后还可以选择按课程编号查询、按课程教师查询、按课程名称查询、按课程学院查询。

1. **管理员操作模块**

管理员操作模块可以对课程进行编辑操作。具体包括增加课程、删除课程、修改课程。除此之外也包括排序课程、查询课程等功能。

排序课程中具体包括按课程学分排序、按课程开设学院排序。

**3.2 系统分析及设计**

**系统涉及对象有三个基本类：**课程类、学生类、管理员类。其中各个类涉及的功能操作如图1所示，这里不再赘述。

用文本文件进行数据的保存，需要保存的数据主要包括课程数据（课程编号、课程名称、任课教师、学分、课程性质、开课院系、课程评价）、用户数据（包括学生、管理员的账号密码信息、学生选课数量和选课的编号）。设置数据操作类，实现所有的文本操作相关的功能。

**3.3系统的实现**

**（1）类的编写**

系统工程名为：CMS。包含了Student类（学生类）、Course类（课程类）、Manager类（管理员类）三个基本类，以及相对应的链表类Stulist类（学生链表类）及Courlist（课程链表类）。学生和管理员用户都有相应的学号\工号num和密码pw，同时学生用户还包括了学生的选课数量信息和已选课程的编号信息。

具体类结构声明如下：

* **Student类：**

#ifndef STUDENT\_HPP

#define STUDENT\_HPP

#include <string>

#include "Coulist.hpp"

using namespace std;

class Student{

public:

Student\* next;

Student(string numv = "",string pwv = "",int numofchosenv = 0,int chosenv[10] = {0}):num(numv),pw(pwv),numofchosen(numofchosenv){

//构造函数

for(int i = 0;i < numofchosen;i++){

chosen[i] = chosenv[i];

}

}

void addchosen(const Coulist& coulist);//在已有课程中选课，加入const以防止修改courlist

void delchosen(int bhv);//退出编号为bhv的课

void showchosen();//查看已选课程编号

void del\_nonecourse(const Coulist& coulist);//删除在coulist已经不存在，但是该学生已经选了的课程

string getnum(){return num;}//获取学号

string getpw(){return pw;}//获取密码

int getnumofchosen(){return numofchosen;}//获取已选课程数量

int getchosen(int i){return chosen[i];}//获取已选课程编号

private:

int chosen[10];//已选课程编号

int numofchosen;//已选课程数量

string num;//学号

string pw;//密码

};

**#endif**

* **Manager类：**

#ifndef MANAGER\_HPP

#define MANAGER\_HPP

#include <string>

using namespace std;

class Manager{

public:

Manager(string numv = "",string pwv = ""):num(numv),pw(pwv){};//构造函数

bool manageridentity();//管理员身份认证

string getnum(){return num;}//获取工号

string getpw(){return pw;}//获取密码

private:

string num;//工号

string pw;//密码

};

#endif

* **Course类：**

#ifndef COURSE\_HPP

#define COURSE\_HPP

#include <string>

using namespace std;

class Course{

public:

Course\* next;

Course(int bhv = -1,string namev = "",string teacherv = "",int creditv = -1,string catagoryv = "",string collegev = "",int assessmentv = -1):bh(bhv),name(namev),teacher(teacherv),catagory(catagoryv),credit(creditv),college(collegev),assessment(assessmentv){};//构造函数

void showcourse();//显示课程

int getbh()const{return bh;}//获取编号

int getassessment()const{return assessment;}//获取课程评价

int getcredit()const{return credit;}//获取学分

string getname()const{return name;}//获取课程名称

string getteacher()const{return teacher;}//获取认课教师

string getcollege()const{return college;}//获取开课学院

string getcatagory()const{return catagory;}//获取课程类别

void setbh(int bhv){bh = bhv;}//设置编号

void setassessment(int assessmentv){assessment = assessmentv;}//设置课程评价

void setcredit(int creditv){credit = creditv;}//设置学分

void setname(string namev){name = namev;}//设置课程名称

void setteacher(string teacherv){teacher = teacherv;}//设置认课教师

void setcollege(string collegev){college = collegev;}//设置开课学院

void setcatagory(string catagoryv){catagory = catagoryv;}//设置课程类别

private:

int bh;//编号

int assessment;//课程评价

int credit;//学分

string name;//课程名称

string teacher;//认课教师

string catagory;//课程性质

string college;//开课学院

};

#endif

**（2）链表的使用**

系统实现采用文件的输入输出流对文本数据进行读取与写入，但是由于学生信息、课程信息、都是一个数据的集合，于是对数据的存储组织使用了单向链表。

因为图书管理系统在登录、查找、修改、添加的时候都需要处理大量的数据，所以使用链表十分必要。以课程信息为例，在course的基础上定义一个对应的coulist类来管理课程数据，具体的结构声明如下：

* **Coulist类：**

#ifndef COULIST\_HPP

#define COULIST\_HPP

#include "Course.hpp"

class Coulist{

public:

Coulist();//构造函数

void addcourse(int bhv,string namev,string teacherv,int creditv,string catagoryv,string collegev,int assessmentv);//添加课程

bool delcourse(int bhv);//按照课程编号删除课程

bool modifycourse(int bhv);//修改课程

bool findcourse();//查找课程的大类，具体包括以下四点。如果没有找到就返回false，找到了就返回true

bool find\_and\_assesscourse();//查找需要评价的课程，并且给予评价

bool sortcourse();//总排序，具体分为以下两部分

void sort\_credit();//按照学分排序

void sort\_college();//按照学院排序

Course\* findbh(int bhv)const;//按照编号查找课程

Course\* findname(string namev)const;//按照名称查找课程

Course\* findteacher(string teacherv)const;//按照教师查找课程

Course\* findcollege(string collegev)const;//按照学院查找课程

void write();//写入文件

void showcourse();//显示课程

int getsize()const{return size;};//获得总课程数

void swapcourse(Course\*p,Course\*q);//交换课程

private:

int size = 0;

Course \*head;

};

#endif

在运用时，令当前图书的next结点指向新的图书结点，即结点的指针next保存新的图书结点的地址（如下图3所示），以此类推，所有图书信息就通过链表的形式串联起来了。具体工作模式如下图3所示：



图3 课程链表示意图

学生类（student）的链表设置与图书的类似，也是通过定义类的指针变量，通过指向下一个类的地址将信息串联起来。即在student的基础上定义stulist，各个类的结构声明如下：

* **Stulist类：**

#ifndef STULIST\_HPP

#define STULIST\_HPP

#include "Course.hpp"

#include "Student.hpp"

class Stulist{

public:

Stulist();

Student\* stuidentity();//学生身份认证

Student\* checkifstu(string numv,string pwv);//查看学生

void addstu(string& numv,string& pwv,int& numofchosenv,int (&chosen)[10]);//添加学生

void write();//写入文件

private:

int size;//学生数量

Student \*head;

};

#endif

CMS系统的信息的管理就具体表现为链表的操作。拿课程信息来说，课程信息的查找、修改、添加和删除与链表的查找、修改、添加、删除对应。

* **课程信息的查找：**

在用户查询课程信息时，会显示四种查询方式以供用户选择，具体如下：

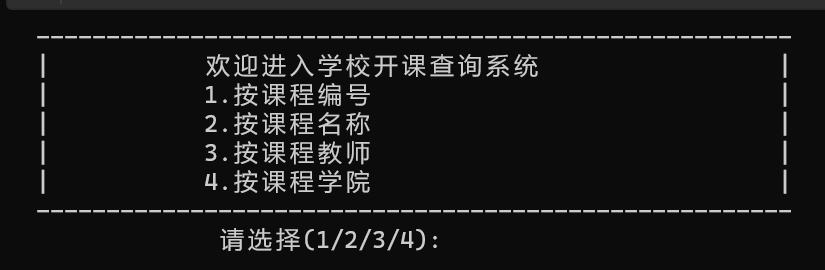


图4 查询课程界面

对于程序的编写，主要依靠四个函数进行实现此功能。具体可见下图5：



图5 查询课程代码流程图

具体解释这张代码图，也即是：

在main程序中，只需运行findcourse()函数即可，该函数的返回类型是bool型、让主程序判断是否查询成功。而在findcourse()函数当中包含四个子函数，分别对应四种查询的方式。

对于除课程编号查询以外的三种类型的查询类型，考虑到多门课程有相同的教师信息、开设学院信息、课程名称信息的问题，直接将返回类型设置为bool型，并且在判断时直接输出达标的课程信息，遍历整个链表。以下拿findname()函数举例。

bool Coulist::findname(string namev)const{

//按课程名称查找课程

Course\* p = head->next;

int count = 0;

while(p){

if(p->getname() == namev){

p->showcourse();

count++;

}

p = p->next;

}

if(count == 0){

cout << "查无此课" << endl;

return false;

}

else

cout << "共查找到" << count << "门课程" << endl;

return true;

}

对于课程编号无重复的特殊性，我们将在后续的选课、退课、课程评价时使用到该函数以判断用户要选退课程的合法性，所以不能将输出查询结果功能也封装在该函数中，所以将该函数的返回结果设置为Course\*型，若无法找到课程则返回NULL。具体的代码实现如下：

Course\* Coulist::findbh(int bhv)const{

// 按课程编号查找课程

Course\* p = head->next;

while(p){

if(p->getbh() == bhv)

return p;

p = p->next;

}

cout << "查无此课" << endl;

return NULL;

}

* **增加课程：**

对于增加代码实现，用了简单的链表添加方法，具体可见注释：

void Coulist::addcourse(int bhv,string namev,string teacherv,int creditv,string catagoryv,string collegev,int assessmentv){

//添加课程

Course\* p = new Course(bhv,namev,teacherv,creditv,catagoryv,collegev,assessmentv);

p->next = head->next;

head->next = p;

size++;

}

* **删除课程：**

对于删除课程代码的实现，使用的先遍历链表来得到指向对应编号的课程的指针。接

着使用链表删除方法删除该课程，具体如下：

bool Coulist::delcourse(int bhv){

//删除课程信息

Course \*p = head->next,\*pre = head;

while(p){

if(p->getbh() == bhv){

pre->next = p->next;

delete p;

size--;

return true;

}

pre = p;

p = p->next;

}

return false;

}

* **修改课程：**

对于删除课程代码的实现，使用上文提到的findbh()函数来得到指向对应编号的课程

的指针。接着使用Course对象提供的修改接口来逐个修改课程的信息。具体如下：

bool Coulist::modifycourse(int bhv){

//修改课程信息

int bhv2,creditv;

string namev,teacherv,catagoryv,collegev;

Course\* target = findbh(bhv);

if(target){

cout << "已查询到这节课，以下是课程信息：" << endl;

target->showcourse();

cout << "请按以下顺序输入课程信息：课程编号 课程名称 任课教师 学分 课程类别 开课学院(为公平,您无法修改课程的评价)" << endl;

cin >> bhv2 >> namev >> teacherv >> creditv >> catagoryv >> collegev;

target->setbh(bhv2);

target->setname(namev);

target->setteacher(teacherv);

target->setcredit(creditv);

target->setcatagory(catagoryv);

target->setcollege(collegev);

return true;

}

else

return false;

}

* **课程排序：**

该课程排序功能的代码实现具体涉及到链表当中的选择排序操作，分别初始化三个

Course\*的指针：p、q、mini。其中，p从head开始while循环，同时q初始化为p->next、mini初始化为p值，并遍历到链表尾部，并用mini来记录遍历过程中的最小值。最后判断mini和p值是否相等，若不相等则交换。具体代码实现如下：

void Coulist::sort\_credit(){

//按学分排序

Course \*p = head->next,\*q = NULL,\*mini = NULL;

while(p->next){

q = p->next;

mini = p;

while(q){

if(mini->getcredit() > q->getcredit()){

mini = q;

}

q = q->next;

}

if(mini != p){

swapcourse(mini,p);

}

p = p->next;

}

}

**（3）交互界面以及登录菜单的实现**

系统运行开始的界面如图6所示：

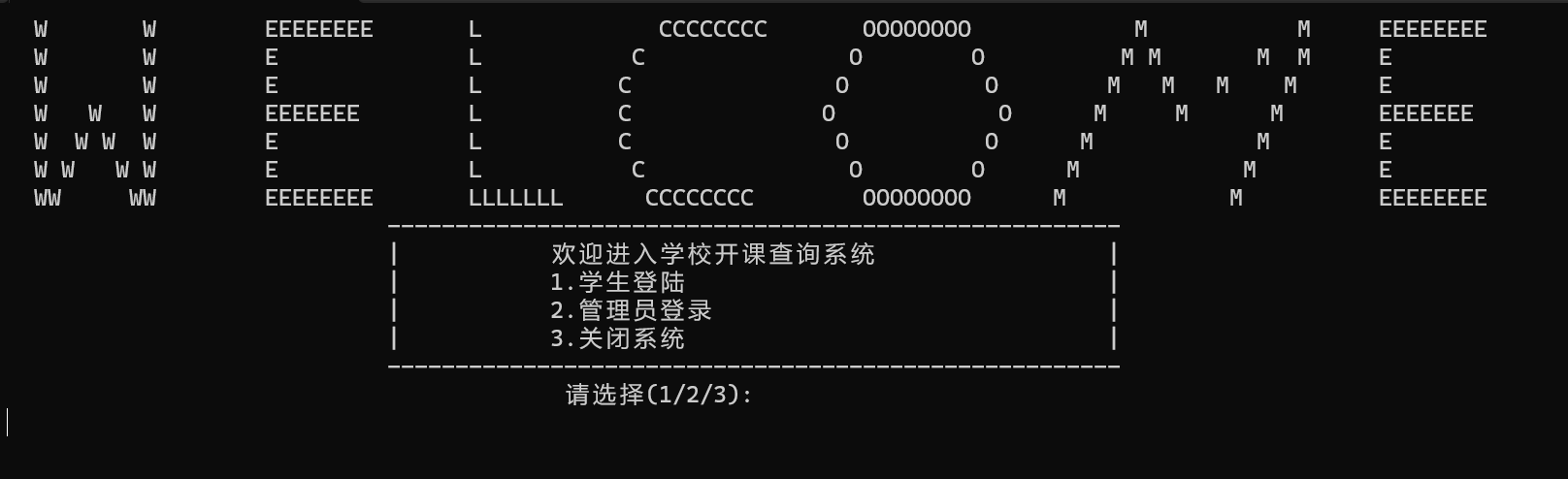


图6 交互主界面

主要通过选择结构和循环结构实现界面的前进和后退。例如，第一个登录界面出现3个选择：1.查询图书信息、2.登录、3.关闭系统，用switch case分别实现，选择之后转到下一个界面。

1. **实验调试、测试、运行记录及分析**

系统在调试测试过程中遇到若干问题，不过经过仔细反复的检查已经消除各种bug。

主要的测试经过如下：

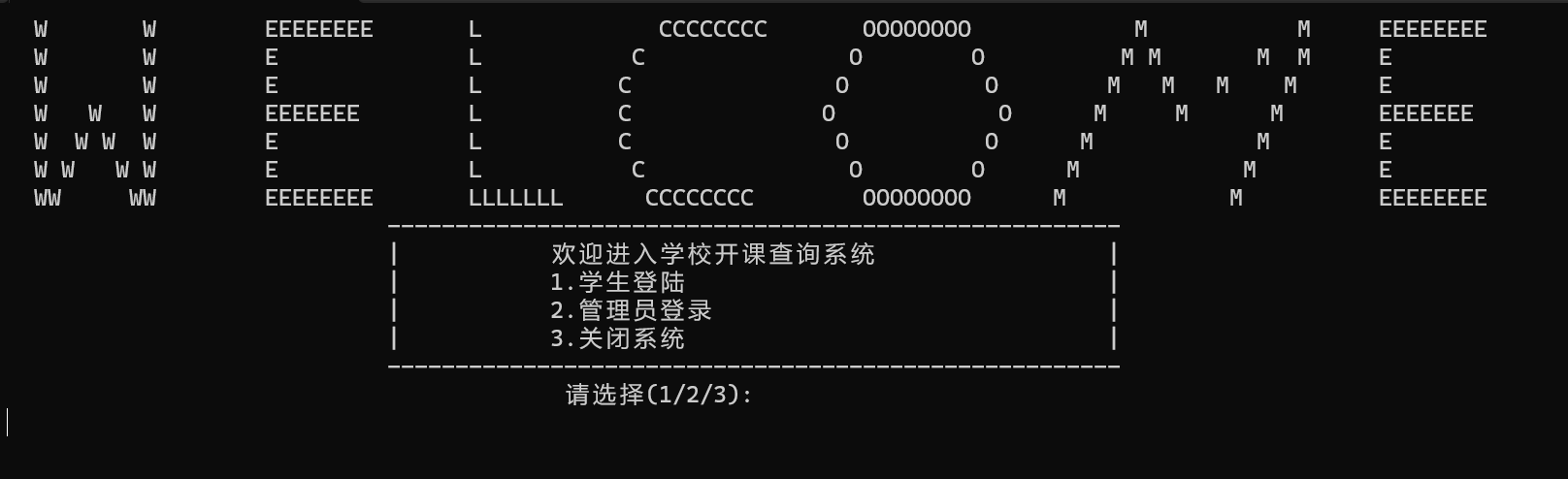
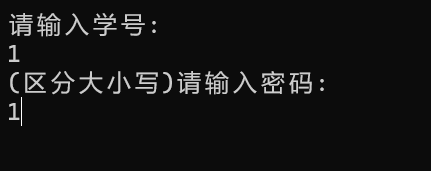


图7 主交互界面

**（1）学生操作模块的测试**

在开始界面输入“1”即可跳转到学生登陆界面，并要求用户直接输入学号和密码。

按要求输入账号密码(在测试阶段，学生和管理员的账号密码均为1)：

  
图8 学生登陆界面

登陆成功后，显示学生操作界面：

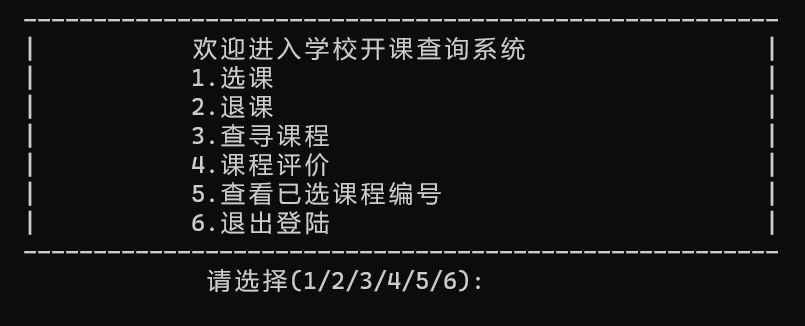


图9 学生操作界面

然后输入“1”，进入选课模式，系统会显示出已选的课程，并要求客户输入需要选的课程编号即可。

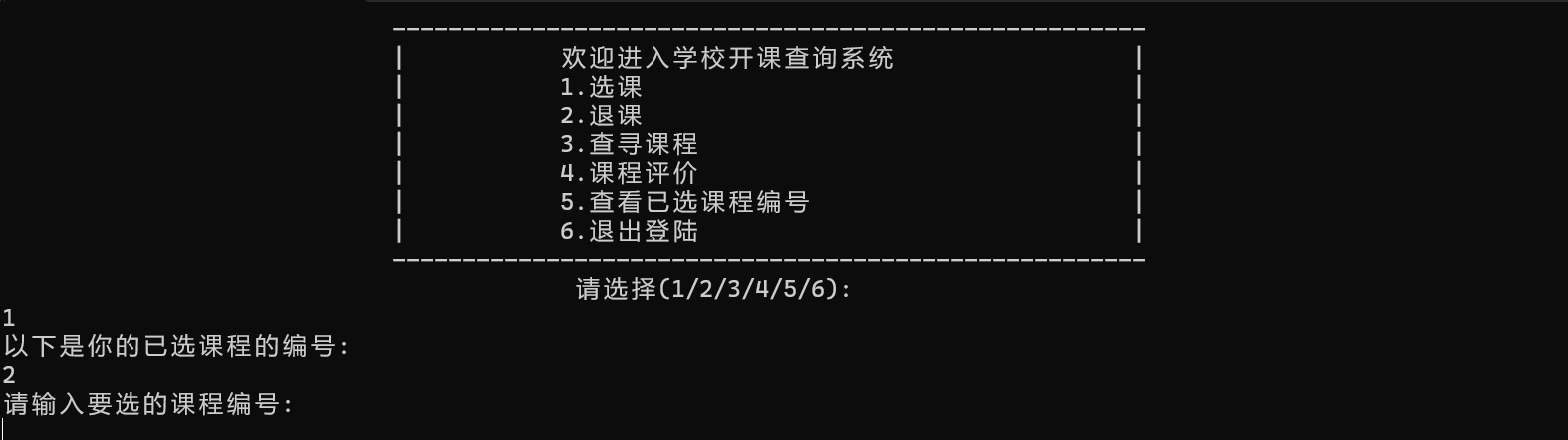


图10 选课界面

若所选课程编号不存在或者是已经选过，系统都会进行相应的提示：

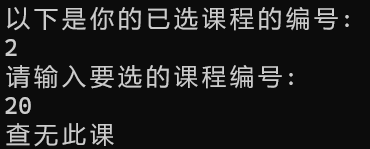


图11 选课报错界面1

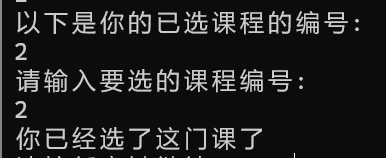


图12 选课报错界面2

然后输入“2”，进入退课模式，系统会显示出已选的课程，并要求客户输入需要退

的课程编号即可。

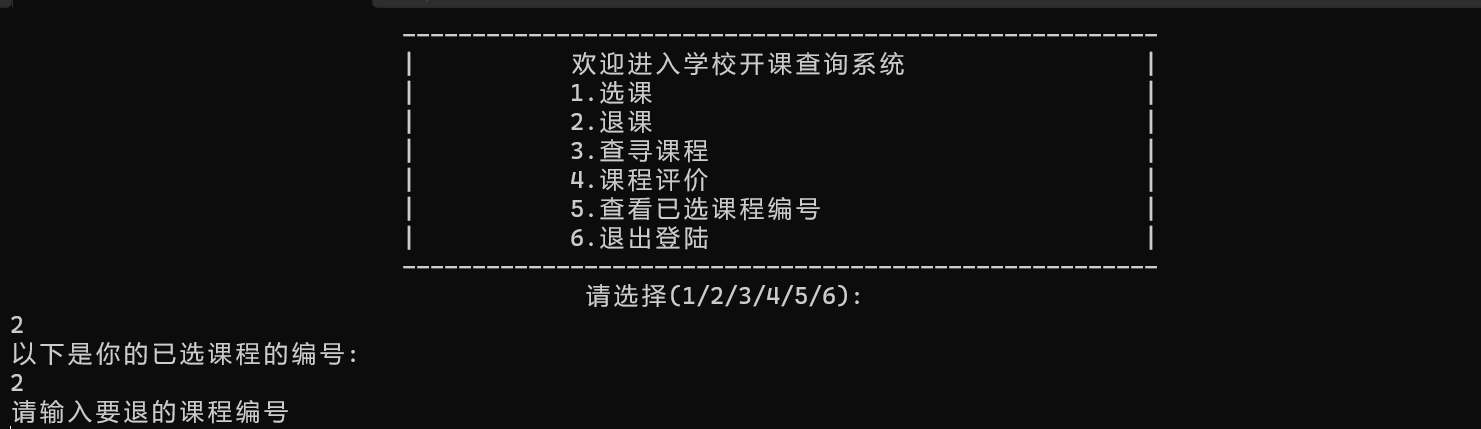


图13 退课界面

若所选课程编号未选或者是不存在或者用户一门课未选，系统都会进行相应的提示：

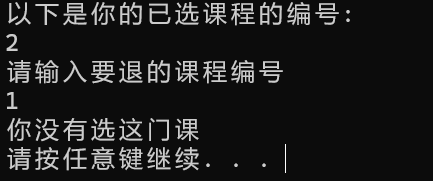


图14 退课报错界面1

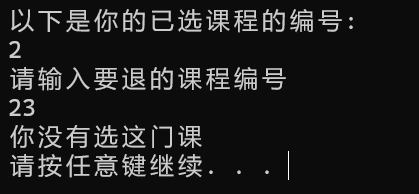


图15 退课报错界面2

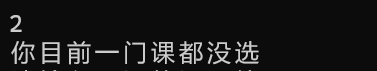


图16 退课报错界面3

然后输入“3”，进入查询课程模式，系统会显示出四种查询方式：

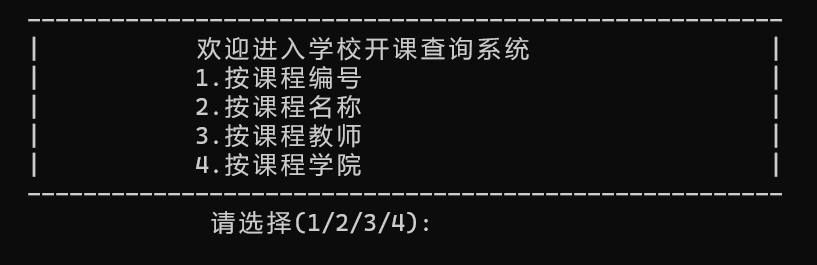


图17 查询课程界面

用户进入相应查询方式后，只需要分别输入所需要查询的课程编号、课程名称、课程教师、课程学员即可。以下拿课程学院举例：

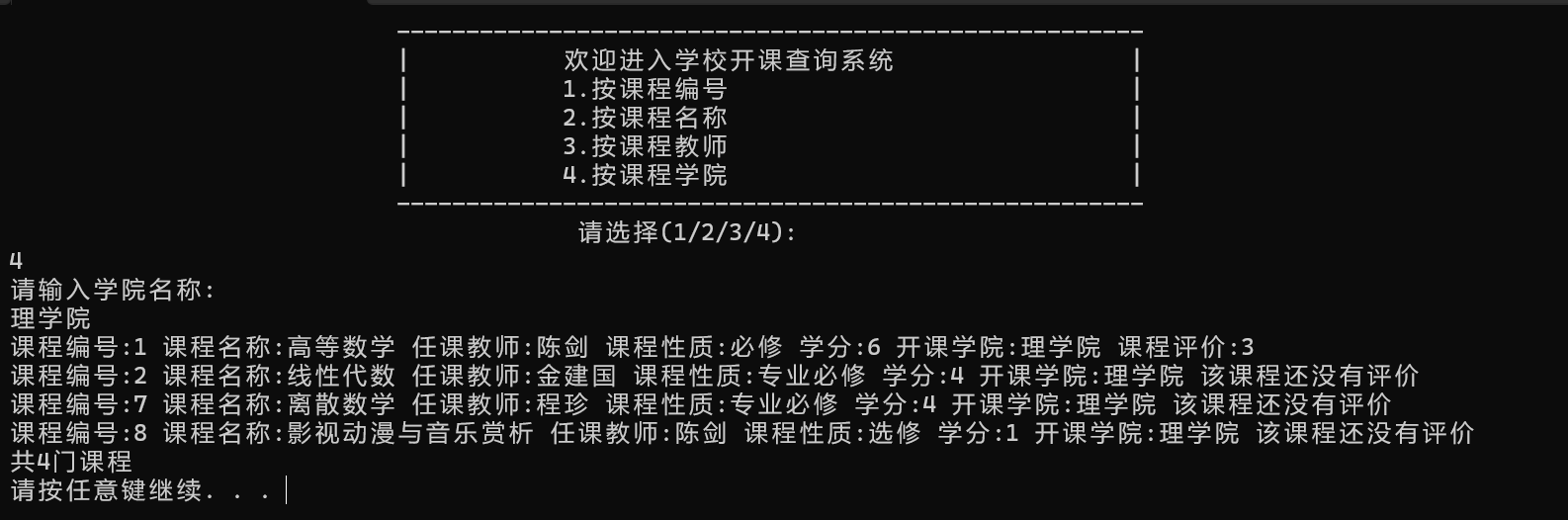


图18 查询课程学院界面

然后输入“4”，进入评价课程模式，用户只需要按照要求输入所需要评价的课程编号之后，系统先进行查询课程的合法性，若存在课程，则系统会先显示出课程的信息，在要求用户输入相应的评价分数。

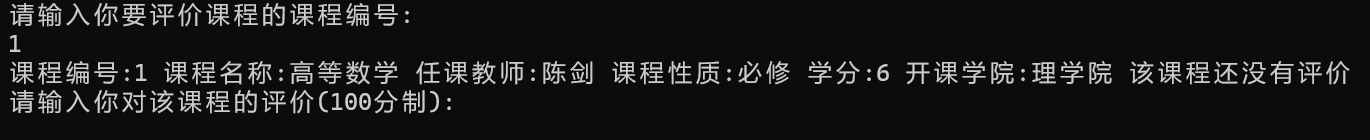


图19 课程评价界面

以下是正确评价后的输出：

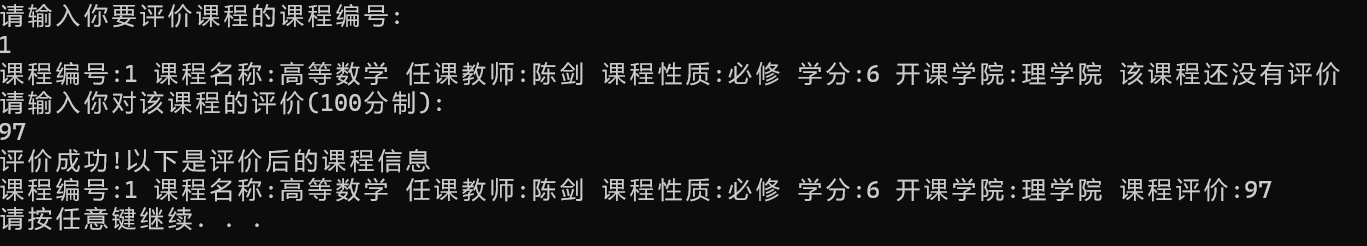


图20 课程评价成功界面

若输入的课程不合法，或未查找到课程编号对应的课程，系统都会进行提示：

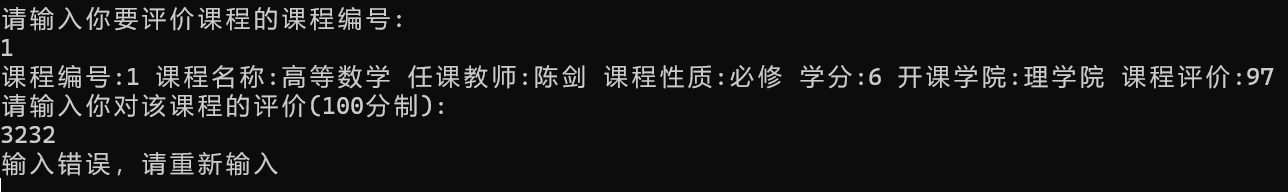


图21 课程评价报错界面1

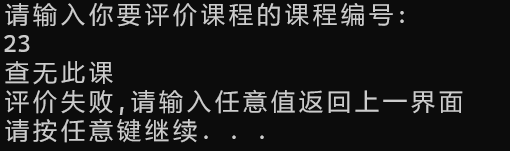


图22 课程评价报错界面2

然后输入“5”，进入查看已选课程模式，系统会直接展示出已选的课程编号：

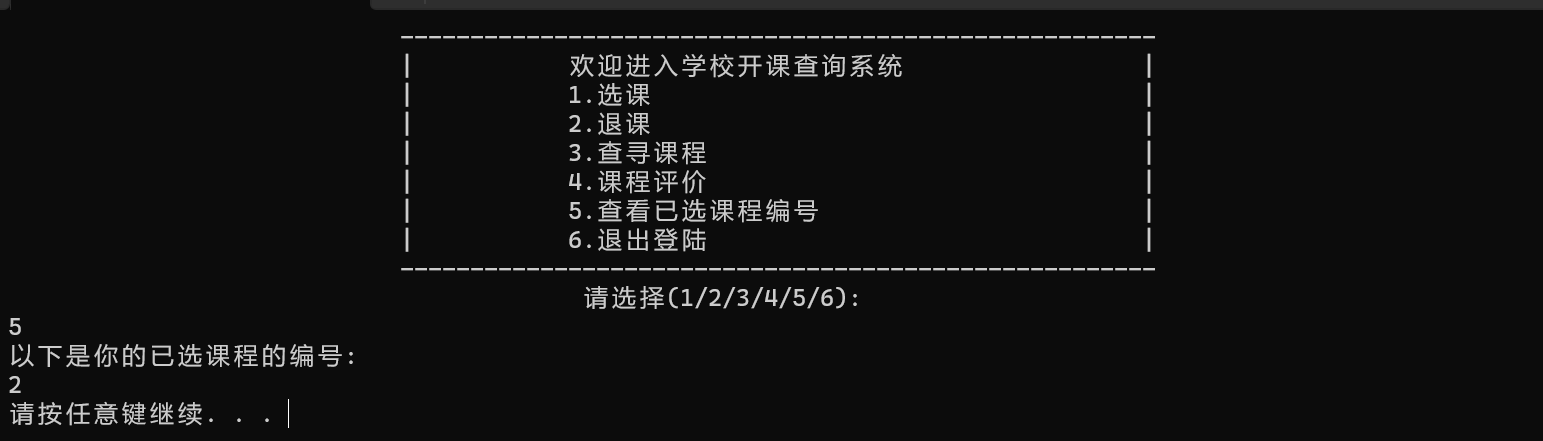


图23 查看已选课程界面

然后输入“6”，退出登陆，返回主界面：

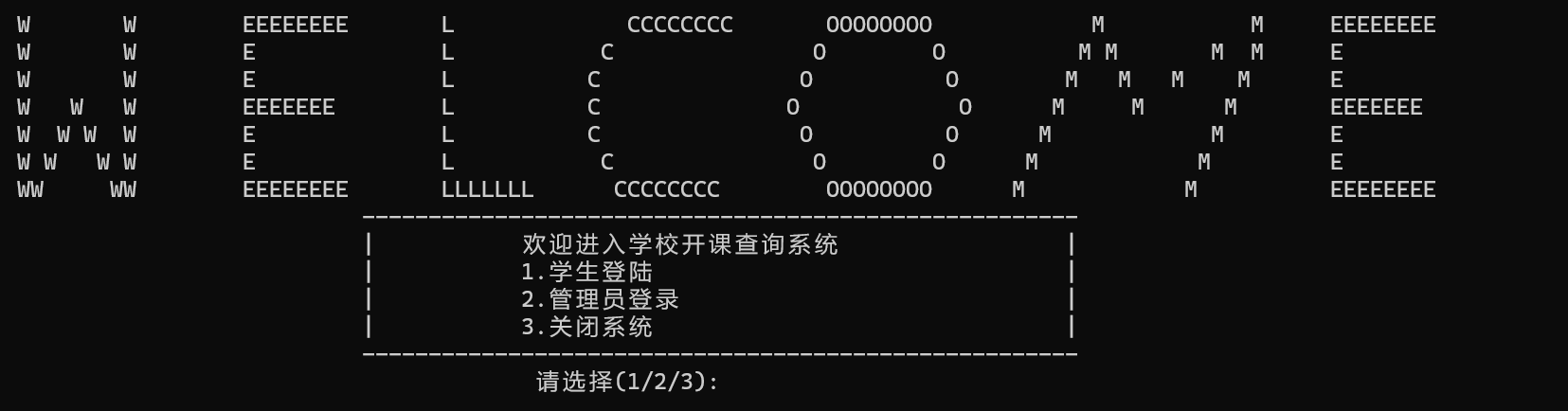


图24 主交互界面

**（2）管理员操作模块的测试**

在开始界面输入“2”即可跳转到管理员登陆界面，并要求用户直接输入工号和密码。

按要求输入工号密码(在测试阶段，学生和管理员的账号密码均为1)：

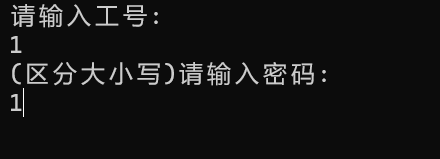


图25 管理员登陆界面

登陆成功后，显示管理员操作界面：

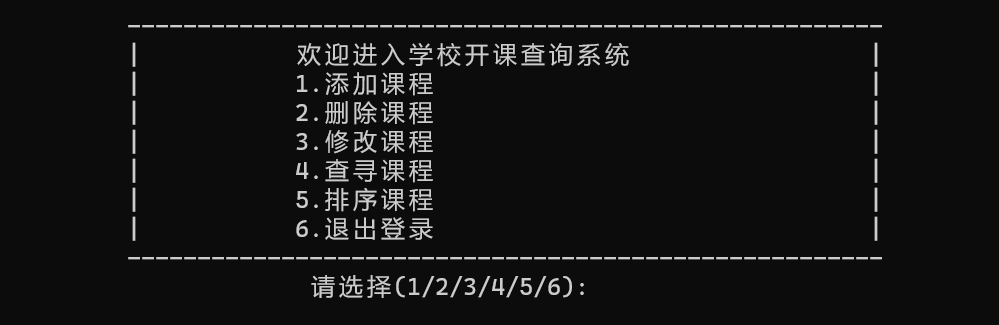


图26 管理员操作界面

在改界面输入“1”后，会进入添加课程模块：

用户选择增加课程后，会先显示已有课程信息，并且要求用户输入新增课程的所有

信息

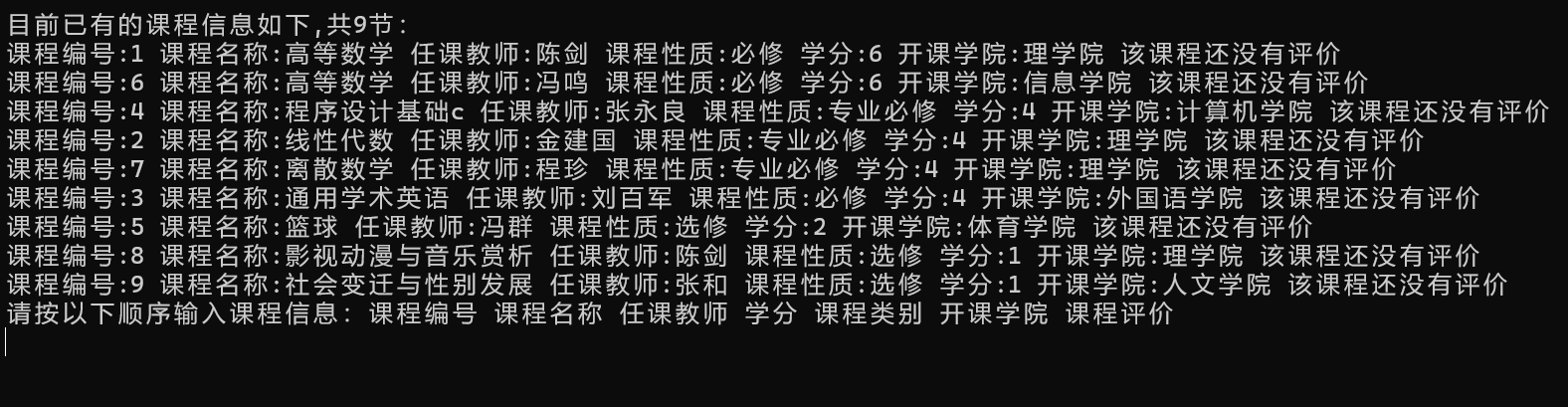


图27 增加课程界面

只要按照要求的顺序输入新课程的所有信息，即可添加完成(课程评价-1表示暂未评

价)。

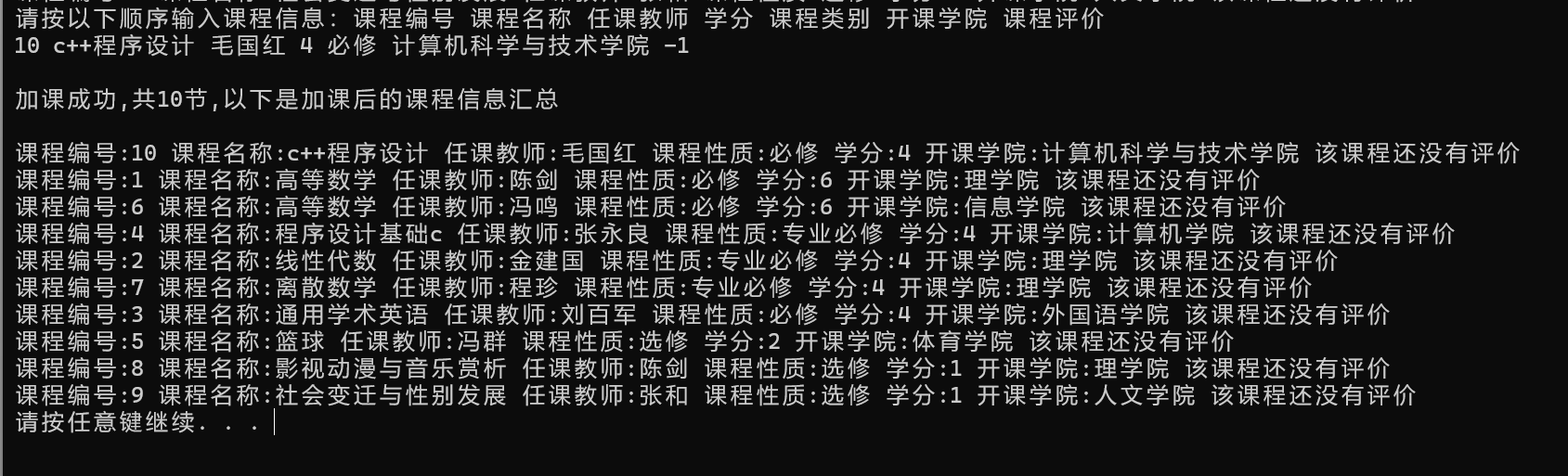


图28 增加课程成功界面

然后输入“2”进入删除课程界面：

用户进入删除课程界面后，也会先展示出已有的课程，并要求用户输入要删除的编号

即可(考虑到课程编号的不可重复性)。具体界面如下：

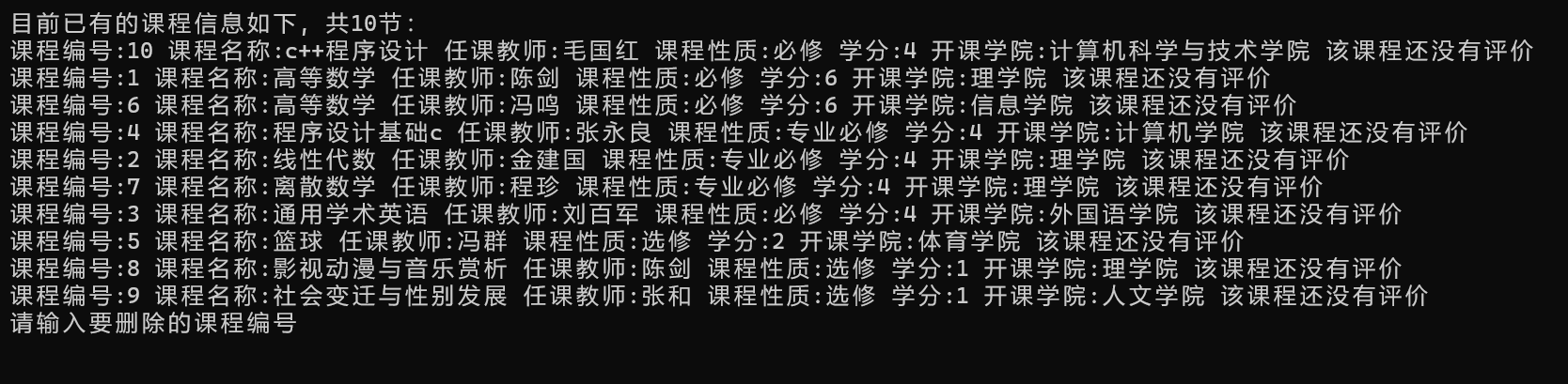


图29 删除课程界面

只需要按照要求输入对应的课程编号即可实现：

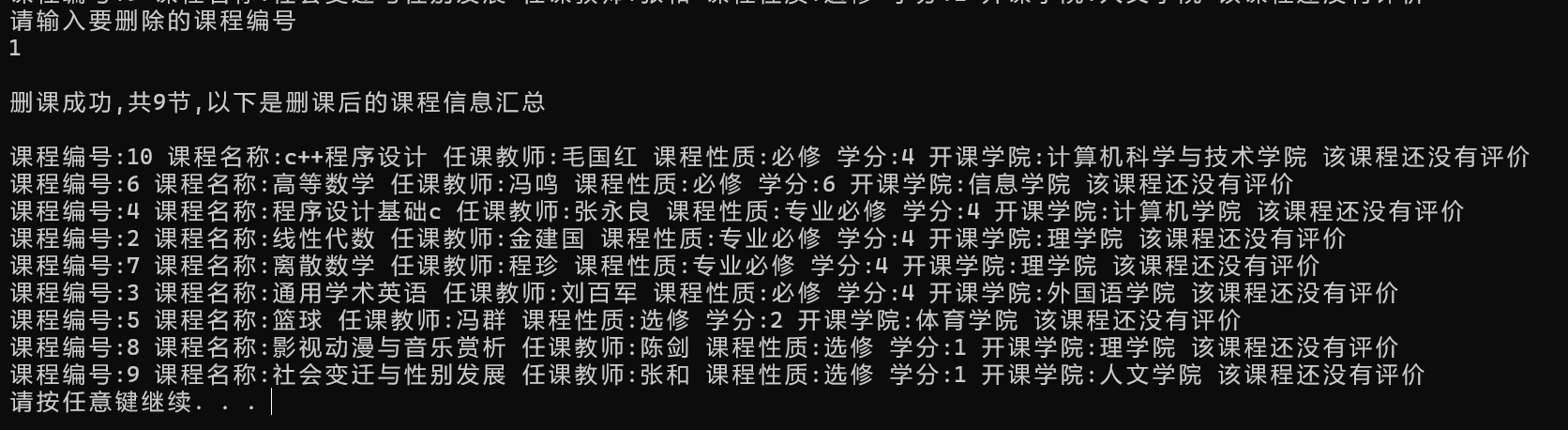


图30 删除课程成功界面

然后输入“3”进入修改课程界面：

用户进入删除课程界面后，也会先展示出已有的课程，并要求用户输入要修改的编号

即可(考虑到课程编号的不可重复性)，接着在输入新的课程编号的信息。具体界面如下：

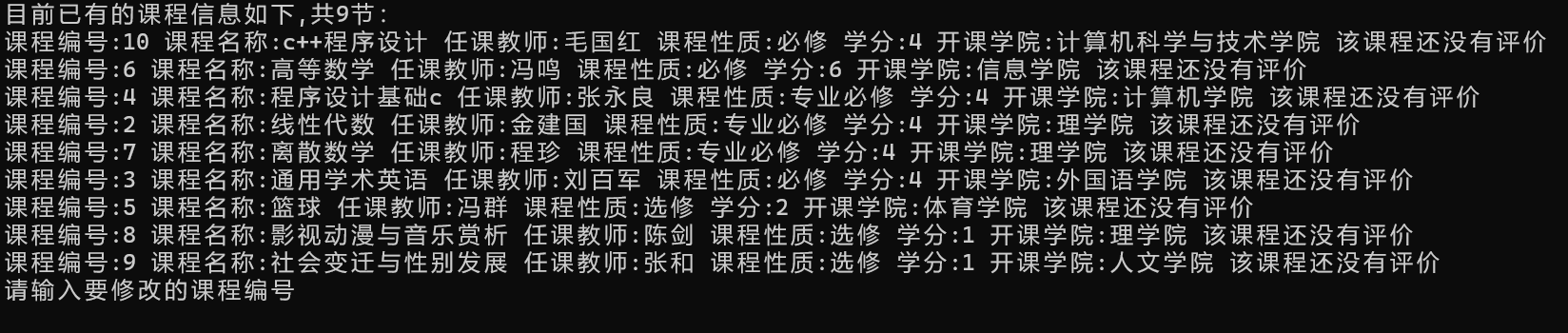


图31 修改课程界面

按要求输入所需要修改课程的编号，系统会先调用findbh()函数来得到指向目标课程的

指针，同时判断是否有该课程(即是否为NULL值)。接着按照要求输入：课程编号 课程名称 任课教师 学分 课程类别 开课学院(为公平,管理员无法修改课程的评价)。具体操作如下

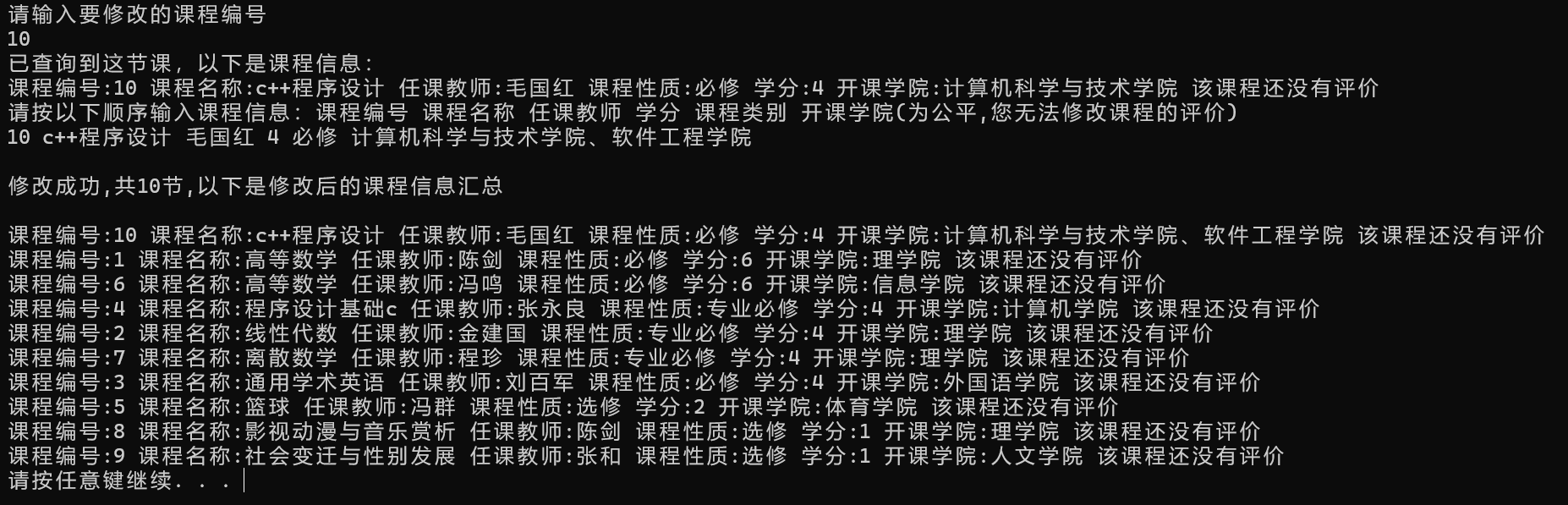


图32 修改课程成功界面

然后输入“4”进入查询课程界面，由于管理员操作的查询课程方法与实现与学生操作的查询课程完全一致，这里不再赘述。

然后输入“5”进入课程排序界面：

在用户课程排序时，会显示两种排序方式以供用户选择，具体如下：

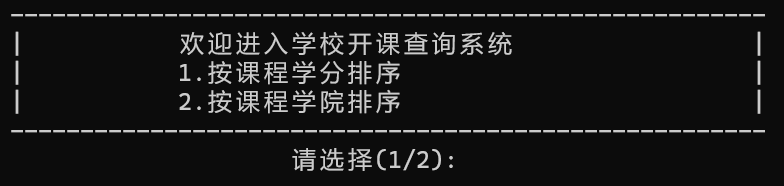


图33 课程排序界面

两种排序方式都会分别输出排序前后的课程信息，以下拿按课程学分升序排序举例：

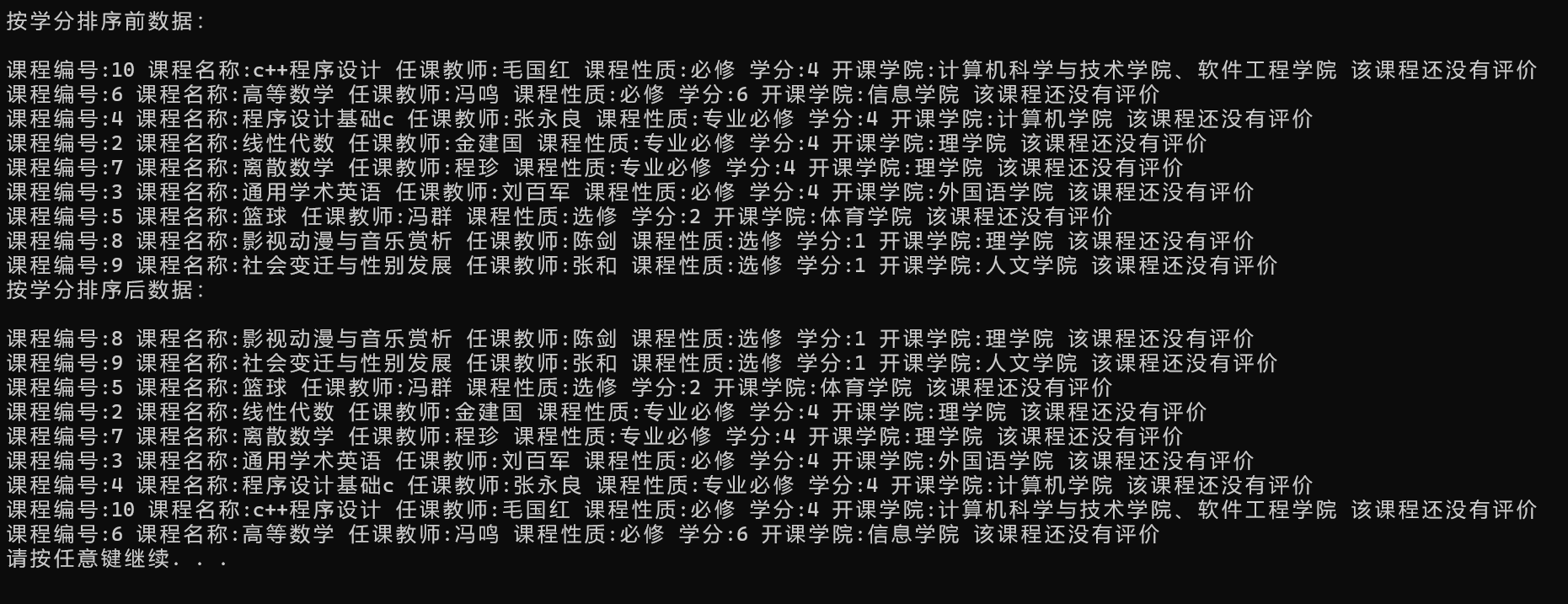


图34 课程排序成功界面

然后输入“6”，退出登陆，返回主界面：

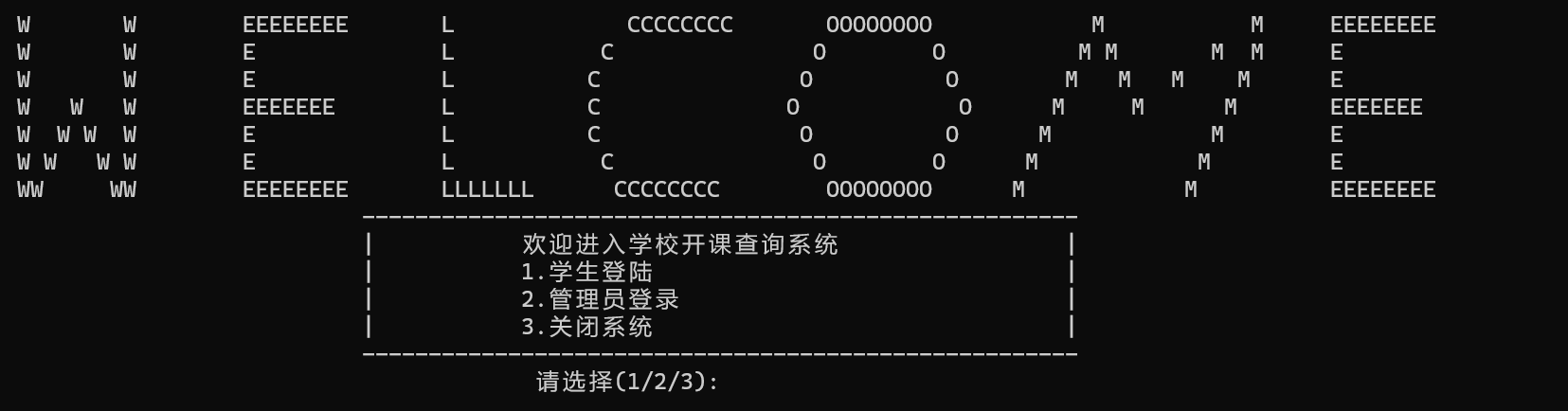


图35 主交互界面

**（3）关闭系统操作模块的测试**

输入“3”后会显示如下界面，用户输入信息后则会结束程序。

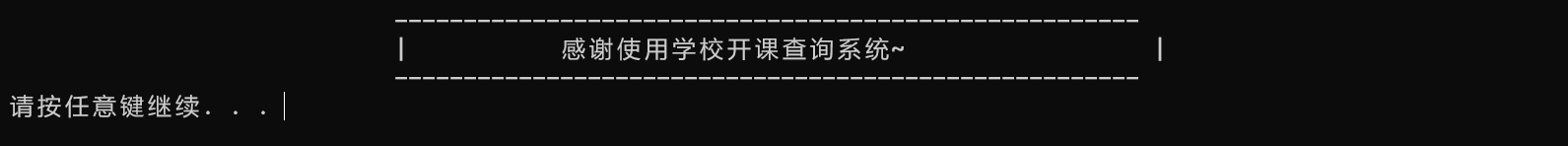


图36 关闭系统界面

至此，CMS系统测试完毕。

* + **问题1：**

**问题描述：**起初，我将所有coulist当中的findbh、findname等等函数的返回类型全

部都设置成了bool型。但是发现后续对课程进行评价等等操作时，只知道课程是否存在还不够，还需要知道确切的课程时哪一个。

**解决方法：**考虑到课程编号的不可测重复性，我将findbh()(查找课程编号相对于的课程函数)的返回类型设置为了Course\*，这样在学生选课，退课，课程评价的时候直接要求学生输入相应的课程编号，可以直接确定哪一个课程是目标课程，不用去考虑课程教师、学院重名的情况导致的不确定性。具体函数声明可见下图：

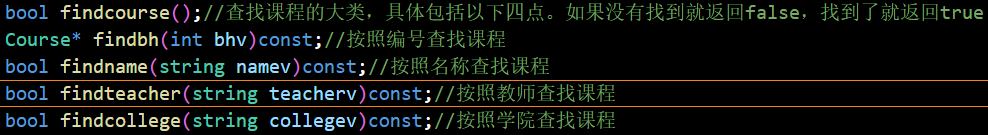


图37 查找函数申明图

* + **问题2：**

**问题描述：**在main程序中，主要通过switch case、goto方法来进行界面的跳转工作。但是如果在其中的一个case中定义了一个变量，则就会出现变量的作用域不确定的情况，使编译出错。

**解决方法：**只需要在每一个case后加一个中括号即可解决，确定的定义变量的作用域：

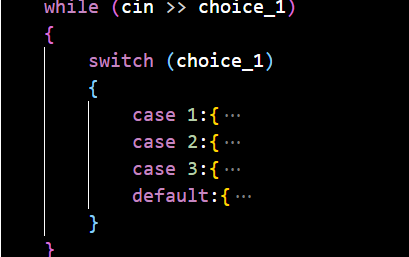


图38 case问题解决图

* + **问题3：**

**问题描述：**在传入学生的选课信息时，由于该信息是以数组的形式存在的，所以在增加

学生函数时要传入改数组，将传入的参数数组写成了int& chosenv[10]导致编译错误。

**解决方法：**这里是数组的引用和引用的数组的问题，我们应该将int& chosenv[10]改写成int (&chosenv) [10]。因为&一般来说是之后结合的。

* + **问题4：**

**问题描述：**要在Studet类里面用到Coulist时(用于学生的选课、退课以及课程评价)，发

现即便引用了Coulist的头文件还是无法识别这个类。

**解决方法：**发现是因为Coulist里面也引用了Student这个头文件，当时想的时头文件加了也无所谓，但是现在还是发现不应该这样。要用到什么头文件，就引用什么不能重复引用。同时，也最好在每个hpp文件当中加入#ifndef、#define、#endif来避免头文件重复引用的问题：

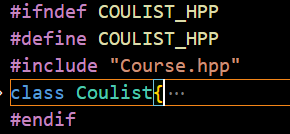


图39 避免文件重复引用图

1. **实验总结**

我设计的CMS系统，功能十分完善。在原本的查询课程、增删课程、排序课程之外

还加入了许多我认为十分有必要，能增加实用性的功能，比如学生、管理员的登陆以及相应不同的操作权限、学生的选课、退课操作以及课程评价操作。也考虑到平性，管理员在修改课程的时候不能修改课程的评价以保证课程评价的公平。

存在的缺点：存在的问题主要是学生选课退课的时候，只能通过课程的编号来选择要选的、退的课程。因为如果涉及到其他信息的选课退课，就要考虑到这些信息的重复性。例如，如果是按照课程的开课学院来选课的话，如果输入理学院就有四个符合条件的课程，还要设计一个让用户在这几个符合条件的课程中再进行下一步的选择，进行下一步的选择又要考虑到课程教师等等的重复性，十分的复杂。所以本系统只能实现到用不重复的课程编号来实现这些选课、退课操作。

通过这次C++的大型实验，我深刻的明白到：课本知识与实践能力相结合的重要性。如果说学c++只是会读程序，写题目是远远不够的，更要能够上手将一个项目完整的写下来才算真正的学到了编程。同时在这些大项目编写的过程中，遇到问题解决问题的方式也十分有益，可以教会你很多计算机在运行时的原理，而不只是停留在会写这个层面。总而言之，实践有益！

1. **附录：源代码**
2. **Coulist.hpp**

#ifndef COULIST\_HPP

#define COULIST\_HPP

#include "Course.hpp"

class Coulist{

public:

Coulist();//构造函数

void addcourse(int bhv,string namev,string teacherv,int creditv,string catagoryv,string collegev,int assessmentv);//添加课程

bool delcourse(int bhv);//按照课程编号删除课程

bool modifycourse(int bhv);//修改课程

bool find\_and\_assesscourse();//查找需要评价的课程，并且给予评价

bool sortcourse();//总排序，具体分为以下两部分

void sort\_credit();//按照学分排序

void sort\_college();//按照学院排序

bool findcourse();//查找课程的大类，具体包括以下四点。如果没有找到就返回false，找到了就返回true

Course\* findbh(int bhv)const;//按照编号查找课程

bool findname(string namev)const;//按照名称查找课程

bool findteacher(string teacherv)const;//按照教师查找课程

bool findcollege(string collegev)const;//按照学院查找课程

void write();//写入文件

void showcourse();//显示课程

int getsize()const{return size;};//获得总课程数

void swapcourse(Course\*p,Course\*q);//交换课程

private:

int size = 0;

Course \*head;

};

#endif

1. **Course.hpp**

#ifndef COURSE\_HPP

#define COURSE\_HPP

#include <string>

using namespace std;

class Course{

public:

Course\* next;

Course(int bhv = -1,string namev = "",string teacherv = "",int creditv = -1,string catagoryv = "",string collegev = "",int assessmentv = -1):bh(bhv),name(namev),teacher(teacherv),catagory(catagoryv),credit(creditv),college(collegev),assessment(assessmentv){};//构造函数

void showcourse();//显示课程

int getbh()const{return bh;}//获取编号

int getassessment()const{return assessment;}//获取课程评价

int getcredit()const{return credit;}//获取学分

string getname()const{return name;}//获取课程名称

string getteacher()const{return teacher;}//获取认课教师

string getcollege()const{return college;}//获取开课学院

string getcatagory()const{return catagory;}//获取课程类别

void setbh(int bhv){bh = bhv;}//设置编号

void setassessment(int assessmentv){assessment = assessmentv;}//设置课程评价

void setcredit(int creditv){credit = creditv;}//设置学分

void setname(string namev){name = namev;}//设置课程名称

void setteacher(string teacherv){teacher = teacherv;}//设置认课教师

void setcollege(string collegev){college = collegev;}//设置开课学院

void setcatagory(string catagoryv){catagory = catagoryv;}//设置课程类别

private:

int bh;//编号

int assessment;//课程评价

int credit;//学分

string name;//课程名称

string teacher;//认课教师

string catagory;//课程性质

string college;//开课学院

};

#endif

1. **Manager.hpp**

#ifndef MANAGER\_HPP

#define MANAGER\_HPP

#include <string>

using namespace std;

class Manager{

public:

Manager(string numv = "",string pwv = ""):num(numv),pw(pwv){};//构造函数

bool manageridentity();//管理员身份认证

string getnum(){return num;}//获取工号

string getpw(){return pw;}//获取密码

private:

string num;//工号

string pw;//密码

};

#endif

1. **Student.hpp**

#ifndef STUDENT\_HPP

#define STUDENT\_HPP

#include <string>

#include "Coulist.hpp"

using namespace std;

class Student{

public:

Student\* next;

Student(string numv = "",string pwv = "",int numofchosenv = 0,int chosenv[10] = {0}):num(numv),pw(pwv),numofchosen(numofchosenv){

//构造函数

for(int i = 0;i < numofchosen;i++){

chosen[i] = chosenv[i];

}

}

void addchosen(const Coulist& coulist);//在已有课程中选课，加入const以防止修改courlist

void delchosen(int bhv);//退出编号为bhv的课

void showchosen();//查看已选课程编号

void del\_nonecourse(const Coulist& coulist);//删除在coulist已经不存在，但是该学生已经选了的课程

string getnum(){return num;}//获取学号

string getpw(){return pw;}//获取密码

int getnumofchosen(){return numofchosen;}//获取已选课程数量

int getchosen(int i){return chosen[i];}//获取已选课程编号

private:

int chosen[10];//已选课程编号

int numofchosen;//已选课程数量

string num;//学号

string pw;//密码

};

#endif

1. **Stulist.hpp**

#ifndef STULIST\_HPP

#define STULIST\_HPP

#include "Course.hpp"

#include "Student.hpp"

class Stulist{

public:

Stulist();

Student\* stuidentity();//学生身份认证

Student\* checkifstu(string numv,string pwv);//查看学生

void addstu(string& numv,string& pwv,int& numofchosenv,int (&chosen)[10]);//添加学生

void write();//写入文件

private:

int size;//学生数量

Student \*head;

};

#endif

1. **Coulist.cpp**

#include "../hpp/Coulist.hpp"

#include <iostream>

#include <fstream>

using namespace std;

Coulist::Coulist(){

//创建一个带有哨兵的链表

head = new Course;

head->next = NULL;

size = 0;

}

void Coulist::write(){

//写入文件

ofstream out("../txt/course.txt");

Course\* p = head;

while(p->next != NULL){

out << p->getbh() << ' ';

out << p->getname() << ' ';

out << p->getteacher() << ' ';

out << p->getcredit() << ' ';

out << p->getcatagory() << ' ';

out << p->getcollege() << ' ';

out << p->getassessment() << endl;

p = p->next;

}

out.close();

}

void Coulist::addcourse(int bhv,string namev,string teacherv,int creditv,string catagoryv,string collegev,int assessmentv){

//添加课程

Course\* p = new Course(bhv,namev,teacherv,creditv,catagoryv,collegev,assessmentv);

p->next = head->next;

head->next = p;

size++;

}

bool Coulist::sortcourse(){

//总排序，具体分为以下两部分

//按照学分排序

//按照学院排序

int choice\_1;

cin >> choice\_1;

switch(choice\_1){

case 1:{

//按学分排序

system("cls");

cout << "按学分排序前数据：" << endl << endl;

this->showcourse();

this->sort\_credit();

cout << "按学分排序后数据：" << endl << endl;

this->showcourse();

break;

}

case 2:{

//按学院排序

system("cls");

cout << "按学院排序前数据：" << endl << endl;

this->showcourse();

this->sort\_college();

cout << "按学院排序后数据：" << endl << endl;

this->showcourse();

break;

}

default:{

cout << "输入错误，请重新输入" << endl;

return false;

}

}

return true;

}

void Coulist::swapcourse(Course\*p,Course\*q){

//交换课程

int temp;

string temp1;

temp = p->getbh();

p->setbh(q->getbh());

q->setbh(temp);

temp = p->getassessment();

p->setassessment(q->getassessment());

q->setassessment(temp);

temp = p->getcredit();

p->setcredit(q->getcredit());

q->setcredit(temp);

temp1 = p->getname();

p->setname(q->getname());

q->setname(temp1);

temp1 = p->getteacher();

p->setteacher(q->getteacher());

q->setteacher(temp1);

temp1 = p->getcollege();

p->setcollege(q->getcollege());

q->setcollege(temp1);

temp1 = p->getcatagory();

p->setcatagory(q->getcatagory());

q->setcatagory(temp1);

}

void Coulist::sort\_credit(){

//按学分排序

Course \*p = head->next,\*q = NULL,\*mini = NULL;

while(p->next){

q = p->next;

mini = p;

while(q){

if(mini->getcredit() > q->getcredit()){

mini = q;

}

q = q->next;

}

if(mini != p){

swapcourse(mini,p);

}

p = p->next;

}

}

void Coulist::sort\_college(){

//按学院排序

Course \*p = head->next,\*q = NULL,\*mini = NULL;

while(p->next){

q = p->next;

mini = p;

while(q){

if(mini->getcollege() > q->getcollege()){

mini = q;

}

q = q->next;

}

if(mini != p){

swapcourse(mini,p);

}

p = p->next;

}

}

bool Coulist::delcourse(int bhv){

//删除课程信息

Course \*p = head->next,\*pre = head;

while(p){

if(p->getbh() == bhv){

pre->next = p->next;

delete p;

size--;

return true;

}

pre = p;

p = p->next;

}

return false;

}

void Coulist::showcourse(){

//显示所有课程信息

Course\* p = head->next;

while(p){

p->showcourse();

p = p->next;

}

}

bool Coulist::findcourse(){

//查找课程信息

int choice\_1,count = 0;

string teacherv,collegev,namev;

cin >> choice\_1;//用户输入要如何查找？

switch(choice\_1){

case 1:{

//按课程编号查找

cout << "请输入课程编号:" << endl;

int bhv;

cin >> bhv;

if(findbh(bhv))

findbh(bhv)->showcourse();

else

return false;

break;

}

case 2:{

//按课程名称查找

cout << "请输入课程名称:" << endl;

cin >> namev;

if(!findname(namev))

return false;

break;

}

case 3:{

//按教师查找

cout << "请输入教师姓名:" << endl;

cin >> teacherv;

if(!findteacher(teacherv))

return false;

break;

}

case 4:{

//按学院查找

cout << "请输入学院名称:" << endl;

cin >> collegev;

if(!findcollege(collegev))

return false;

break;

}

default:{

//输入错误

cout << "输入错误" << endl;

return false;

break;

}

}

return true;

}

bool Coulist::find\_and\_assesscourse(){

//查找课程信息并评价

int choice\_1,assessmentv;

Course\* course;//临时变量储存找到的课程，以免下面函数二次调用

cout << "请输入你要评价课程的课程编号:" << endl;

int bhv;

cin >> bhv;

course = findbh(bhv);

if(course){

course->showcourse();

cout << "请输入你对该课程的评价(100分制):" << endl;

while(cin >> assessmentv){

if(assessmentv >= 0 && assessmentv <= 100)

break;

else

cout << "输入错误，请重新输入" << endl;

}//保证输入的评价在0-100之间

course->setassessment(assessmentv);

}

else{

return false;

}

cout << "评价成功!以下是评价后的课程信息" << endl;

course->showcourse();

system("pause");

return true;

}

bool Coulist::modifycourse(int bhv){

//修改课程信息

int bhv2,creditv;

string namev,teacherv,catagoryv,collegev;

Course\* target = findbh(bhv);

if(target){

cout << "已查询到这节课，以下是课程信息：" << endl;

target->showcourse();

cout << "请按以下顺序输入课程信息：课程编号 课程名称 任课教师 学分 课程类别 开课学院(为公平,您无法修改课程的评价)" << endl;

cin >> bhv2 >> namev >> teacherv >> creditv >> catagoryv >> collegev;

target->setbh(bhv2);

target->setname(namev);

target->setteacher(teacherv);

target->setcredit(creditv);

target->setcatagory(catagoryv);

target->setcollege(collegev);

return true;

}

else

return false;

}

Course\* Coulist::findbh(int bhv)const{

// 按课程编号查找课程

Course\* p = head->next;

while(p){

if(p->getbh() == bhv)

return p;

p = p->next;

}

cout << "查无此课" << endl;

return NULL;

}

bool Coulist::findname(string namev)const{

//按课程名称查找课程

Course\* p = head->next;

int count = 0;

while(p){

if(p->getname() == namev){

p->showcourse();

count++;

}

p = p->next;

}

if(count == 0){

cout << "查无此课" << endl;

return false;

}

else

cout << "共查找到" << count << "门课程" << endl;

return true;

}

bool Coulist::findteacher(string teacherv)const{

//按教师查找课程

Course\* p = head->next;

int count = 0;

while(p){

if(p->getteacher() == teacherv){

p->showcourse();

count++;

}

p = p->next;

}

if(count == 0){

cout << "查无此课" << endl;

return false;

}

else

cout << "共查找到" << count << "门课程" << endl;

return true;

}

bool Coulist::findcollege(string collegev)const{

//按学院查找课程

Course\* p = head->next;

int count = 0;

while(p){

if(p->getcollege() == collegev){

p->showcourse();

count++;

}

p = p->next;

}

if(count == 0){

cout << "查无此课" << endl;

return false;

}

else

cout << "共查找到" << count << "门课程" << endl;

return true;

}

1. **Main.cpp**

#include "../hpp/Coulist.hpp"

#include "../hpp/Stulist.hpp"

#include "../hpp/Manager.hpp"

#include "../hpp/Student.hpp"

#include "../hpp/Course.hpp"

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

void welcome();

void stuidentity();

void manidentity();

void stuwelcome();

void manwelcome();

void howsort();

void howfind();

void bye();

int main()

{

Stulist stulist;

Coulist coulist;

ifstream inStudent("../txt/Student.txt");

ifstream inCourse("../txt/Course.txt");

ifstream inManager("../txt/Manager.txt");

string numv, pwv, collegev, namev, teacherv, catagoryv;

int bhv, assessmentv, creditv,numofchosenv,chosenv[10];

while (inStudent >> numv >> pwv >> numofchosenv)

{

for(int i = 0;i < numofchosenv;i++) inStudent >> chosenv[i];

stulist.addstu(numv, pwv,numofchosenv,chosenv);

}

inStudent.close(); //学生和课程信息读取完毕

while (inCourse >> bhv >> namev >> teacherv >> creditv >> catagoryv >> collegev >> assessmentv)

{

coulist.addcourse(bhv, namev, teacherv, creditv, catagoryv, collegev, assessmentv);

}

inCourse.close(); //学生和课程信息读取完毕

inManager >> numv >> pwv; //输入管理员账号密码

Manager manager(numv, pwv); //创建管理员对象,此项目只有唯一管理员

inManager.close(); //管理员账号密码输入完毕

int choice\_1, choice\_2, wrongtime = 0; //设置用户选项、错误次数

out1:

welcome();

while (cin >> choice\_1)

{

switch (choice\_1)

{

case 1:{

Student \*target = stulist.stuidentity(); //认证并储存身份

while (target == NULL)

{

cout << "学号或密码错误" << endl

<< endl;

system("pause");

target = stulist.stuidentity();

wrongtime++;

if (wrongtime == 3)

{

cout << "错误次数过多,请稍后再试" << endl;

system("pause");

goto out1;

}

} //学生身份认证

wrongtime = 0; //错误次数清零

out2:

target->del\_nonecourse(coulist);

stuwelcome();

cin >> choice\_2;

switch (choice\_2)

{

case 1:

//选课

target->addchosen(coulist);

system("pause");

goto out2;

break;

case 2:

//退课

target->showchosen();

if(target->getnumofchosen() == 0)

{

cout << "您还没有选课,请输入任意值返回上一界面" << endl;

system("pause");

goto out2;

}

cout << "请输入要退的课程编号" << endl;

cin >> bhv;

target->delchosen(bhv);

system("pause");

goto out2;

break;

case 3:

//查询课程

howfind();

if (!coulist.findcourse())

{

//查找，如果没找到就执行if，返回上一个界面

cout << "输入任意值返回上一界面" << endl;

system("pause");

goto out2;

}

system("pause");

goto out2;

break;

case 4:

//课程评价

if (!coulist.find\_and\_assesscourse())

{

cout << "评价失败,请输入任意值返回上一界面" << endl;

system("pause");

}

goto out2;

break;

case 5:

//查看已选课程

target->showchosen();

system("pause");

goto out2;

case 6:

//重新登陆

goto out1;

default:

cout << "输入错误,请重新输入" << endl;

break;

}

break;

}

case 2:{

while(!manager.manageridentity())

{

cout << "账号或密码错误" << endl;

system("pause");

wrongtime++;

if (wrongtime == 3)

{

cout << "错误次数过多,请稍后再试" << endl;

system("pause");

goto out1;

}

}

wrongtime = 0; //错误次数清零

out3:

manwelcome();

cin >> choice\_2;

switch (choice\_2)

{

case 1:

//添加课程

system("cls");

cout << "目前已有的课程信息如下,共" << coulist.getsize() << "节：" << endl;

coulist.showcourse();

cout << "请按以下顺序输入课程信息：课程编号 课程名称 任课教师 学分 课程类别 开课学院 课程评价" << endl;

cin >> bhv >> namev >> teacherv >> creditv >> catagoryv >> collegev >> assessmentv;

coulist.addcourse(bhv, namev, teacherv, creditv, catagoryv, collegev, assessmentv);

cout << endl << "加课成功,共" << coulist.getsize() << "节,以下是加课后的课程信息汇总" << endl << endl;

coulist.showcourse();

system("pause");

goto out3;

break;

case 2:

//删除课程

system("cls");

cout << "目前已有的课程信息如下, 共" << coulist.getsize() << "节：" << endl;

coulist.showcourse();

cout << "请输入要删除的课程编号" << endl;

cin >> bhv;

if (!coulist.delcourse(bhv))

{

cout << "删除失败!因为您提供的课程不存在" << endl;

system("pause");

goto out3;

}

cout << endl << "删课成功,共" << coulist.getsize() << "节,以下是删课后的课程信息汇总" << endl << endl;

coulist.showcourse();

system("pause");

goto out3;

break;

case 3:

//修改课程

system("cls");

cout << "目前已有的课程信息如下,共" << coulist.getsize() << "节：" << endl;

coulist.showcourse();

cout << "请输入要修改的课程编号" << endl;

cin >> bhv;

if (!coulist.modifycourse(bhv))

{

cout << "修改失败!因为您提供的课程不存在" << endl;

system("pause");

goto out3;

}

cout << endl << "修改成功,共" << coulist.getsize() << "节,以下是修改后的课程信息汇总" << endl << endl;

coulist.showcourse();

system("pause");

goto out3;

break;

case 4:{

//查询课程

howfind();

if (!coulist.findcourse())

{

//查找，如果没找到就执行if，返回上一个界面

cout << "输入任意值返回上一界面" << endl;

system("pause");

goto out3;

}

system("pause");

goto out3;

break;

}

case 5:{

//排序课程

howsort();

if (!coulist.sortcourse())

{

cout << "排序失败,请输入任意值返回上一界面" << endl;

system("pause");

}

system("pause");

goto out3;

}

case 6:

//重新登陆

goto out1;

default:

cout << "输入错误,请重新输入" << endl;

goto out3;

break;

}

break;

}

case 3:{

bye();

system("pause");

stulist.write();

coulist.write();

return 0;

}

default:{

cout << "输入错误,请重新输入" << endl;

break;

}

}

}

return 0;

}

void welcome()

{

system("cls");

cout << " W W EEEEEEEE L CCCCCCCC OOOOOOOO M M EEEEEEEE" << endl;

cout << " W W E L C O O M M M M E " << endl;

cout << " W W E L C O O M M M M E " << endl;

cout << " W W W EEEEEEE L C O O M M M EEEEEEE " << endl;

cout << " W W W W E L C O O M M E " << endl;

cout << " W W W W E L C O O M M E " << endl;

cout << " WW WW EEEEEEEE LLLLLLL CCCCCCCC OOOOOOOO M M EEEEEEEE " << endl;

cout << " ------------------------------------------------------" << endl;

cout << " | 欢迎进入学校开课查询系统 |" << endl;

cout << " | 1.学生登陆 |" << endl;

cout << " | 2.管理员登录 |" << endl;

cout << " | 3.关闭系统 |" << endl;

cout << " ------------------------------------------------------" << endl;

cout << " 请选择(1/2/3):" << endl;

}

void stuwelcome()

{

system("cls");

cout << " ------------------------------------------------------" << endl;

cout << " | 欢迎进入学校开课查询系统 |" << endl;

cout << " | 1.选课 |" << endl;

cout << " | 2.退课 |" << endl;

cout << " | 3.查寻课程 |" << endl;

cout << " | 4.课程评价 |" << endl;

cout << " | 5.查看已选课程编号 |" << endl;

cout << " | 6.退出登陆 |" << endl;

cout << " ------------------------------------------------------" << endl;

cout << " 请选择(1/2/3/4/5/6):" << endl;

}

void manwelcome()

{

system("cls");

cout << " ------------------------------------------------------" << endl;

cout << " | 欢迎进入学校开课查询系统 |" << endl;

cout << " | 1.添加课程 |" << endl;

cout << " | 2.删除课程 |" << endl;

cout << " | 3.修改课程 |" << endl;

cout << " | 4.查寻课程 |" << endl;

cout << " | 5.排序课程 |" << endl;

cout << " | 6.退出登录 |" << endl;

cout << " ------------------------------------------------------" << endl;

cout << " 请选择(1/2/3/4/5/6):" << endl;

}

void howfind()

{

system("cls");

cout << " ------------------------------------------------------" << endl;

cout << " | 欢迎进入学校开课查询系统 |" << endl;

cout << " | 1.按课程编号 |" << endl;

cout << " | 2.按课程名称 |" << endl;

cout << " | 3.按课程教师 |" << endl;

cout << " | 4.按课程学院 |" << endl;

cout << " ------------------------------------------------------" << endl;

cout << " 请选择(1/2/3/4):" << endl;

}

void howsort()

{

system("cls");

cout << " ------------------------------------------------------" << endl;

cout << " | 欢迎进入学校开课查询系统 |" << endl;

cout << " | 1.按课程学分排序 |" << endl;

cout << " | 2.按课程学院排序 |" << endl;

cout << " ------------------------------------------------------" << endl;

cout << " 请选择(1/2):" << endl;

}

void bye(){

system("cls");

cout << " ------------------------------------------------------" << endl;

cout << " | 感谢使用学校开课查询系统~ |" << endl;

cout << " ------------------------------------------------------" << endl;

}

1. **Manager.cpp**

#include "../hpp/Manager.hpp"

#include <iostream>

#include <string>

using namespace std;

bool Manager::manageridentity(){

//管理员身份认证

system("cls");

string num,pw;

cout << "请输入工号:" << endl;

cin >> num;

cout << "(区分大小写)请输入密码:" << endl;

cin >> pw;

if(num == this->num && pw == this->pw)

return true;

else

return false;

}

1. **Student.cpp**

#include "../hpp/Stulist.hpp"

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

Stulist::Stulist(){

//初始化学生链表,带有哨兵的链表

head = new Student;

head->next = NULL;

size = 0;

}

void Stulist::write(){

//写入文件

ofstream out("../txt/student.txt");

Student\* p = head->next;

while(p){

out << p->getnum() << ' ';

out << p->getpw() << ' ';//写入学号和密码

out << p->getnumofchosen() << ' ';//写入已选课程数量

for(int i = 0;i < p->getnumofchosen();i++)

out << p->getchosen(i) << ' ';//写入已选课程编号

out << endl;

p = p->next;

}

out.close();

}

Student\* Stulist::checkifstu(string numv,string pwv){

//查看是否有这个学生账号

Student \*p = head->next;

while(p){

if(p->getnum() == numv && p->getpw() == pwv)

return p;

p = p->next;

}

return NULL;

}

Student\* Stulist::stuidentity(){

//学生身份验证

system("cls");

string num,pw;

cout << "请输入学号:" << endl;

cin >> num;

cout << "(区分大小写)请输入密码:" << endl;

cin >> pw;

return checkifstu(num,pw);

}

void Stulist::addstu(string& numv,string& pwv,int& numofchosenv,int (&chosenv)[10]){

//添加学生

Student \*p = new Student(numv,pwv,numofchosenv,chosenv);

p->next = head->next;

head->next = p;

size++;

}