浙江工业大学

C++程序设计课程设计
课设报告

2023/2024(2)

实验题目 学校开课查询系统

学生姓名 李飞飞

学生学号 202105710309

学生班级 软工 2206

任课教师 毛国红

提交日期 2024.5.1

计算机科学与技术学院

学校开课查询系统 实验报告

一、 大型实验的内容

学校开课查询系统用于管理学校开课与学生选课，包括学生模块，管理员模块，课程模块三个部分

1、设计菜单实现功能选择；

2、能够对课程信息进行输入、修改、删除操作；

3、按给定的条件（编号、名称、任课教师、开课院系等）精确或者模糊查询课程信息；

4、能对开课信息按学分、开课学院排序整理；

5、以文件形式保存相关信息，可以读取默认文件中的信息进行查询等操作。

添加了学生退课，课程容量上限，模糊查询课程的功能，所有接收输入 chu4 均做了非法输入的处理，具有良好的鲁棒性

二、 运行环境

图书管理系统（LMS）在 Visual Studio 6.0 平台下开发，操作系统：Windows 7。

硬件环境：(备注：可以查看“计算机”属性)

处理器：AMD Ryzen 7 5800H with Radeon Graphics 3.20 GHz

内存：16.0 GB

系统类型：64 位操作系统

三、 实验课题分析（主要的模块功能、流程图）

3.1 学校开课查询系统的主要功能

学校开课查询主要功能为：学生管理、课程管理、管理人员管理，可以完成查询开课信息，对开课信息进行排序，学生，管理人员登陆，修改学生，课程信息，学生退，选课

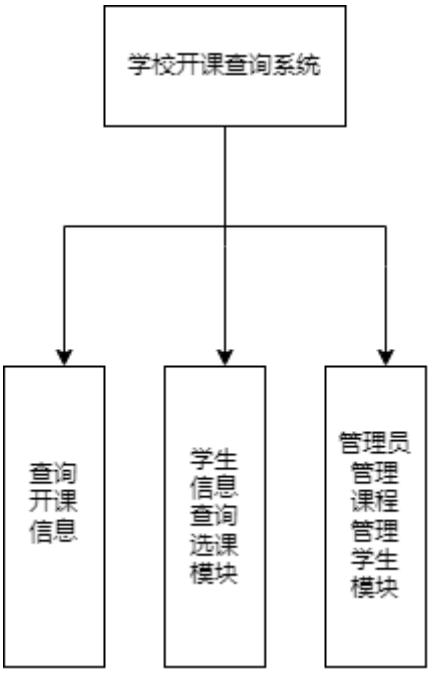


图 1

系统各模块的功能具体描述为：

1、 **登录模块**

此模块可以查询所有课程并按学分，院系排序后展示或是查找指定课程

2、 读者模块

读者必须先进行登录，登录成功之后可以查询课程信息，并按编号进行选课，如果已经选了，可以进行退选，并且可以展示个人信息，修改密码，查询已选课，查看课程列表。课程列表可按编号，课程名，教师姓名，开课院系，其中教师姓名和开课院系还可模糊搜索

3、 管理员模块

登录成功后可添加删除修改学生与课程信息，修改管理员信息

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。

人员类涉及的功能操作归纳为如下图所示：

对象	涉及的对象操作	
学生	查询读者信息(个人信息和学生选课情况)	
	查询课程信息	
	退选与选课操作	
管理人员	维护课程信息	添加课程信息
		删除课程信息
		修改课程信息
	维护学生信息	添加学生信息
		修改学生信息
		删除学生信息

图 2

可以采用面向对象的方式实现图书管理系统，根据不同的使用权限，使用对象分为学生、管理员与一般访客。系统的主要的类结构如下图所示。

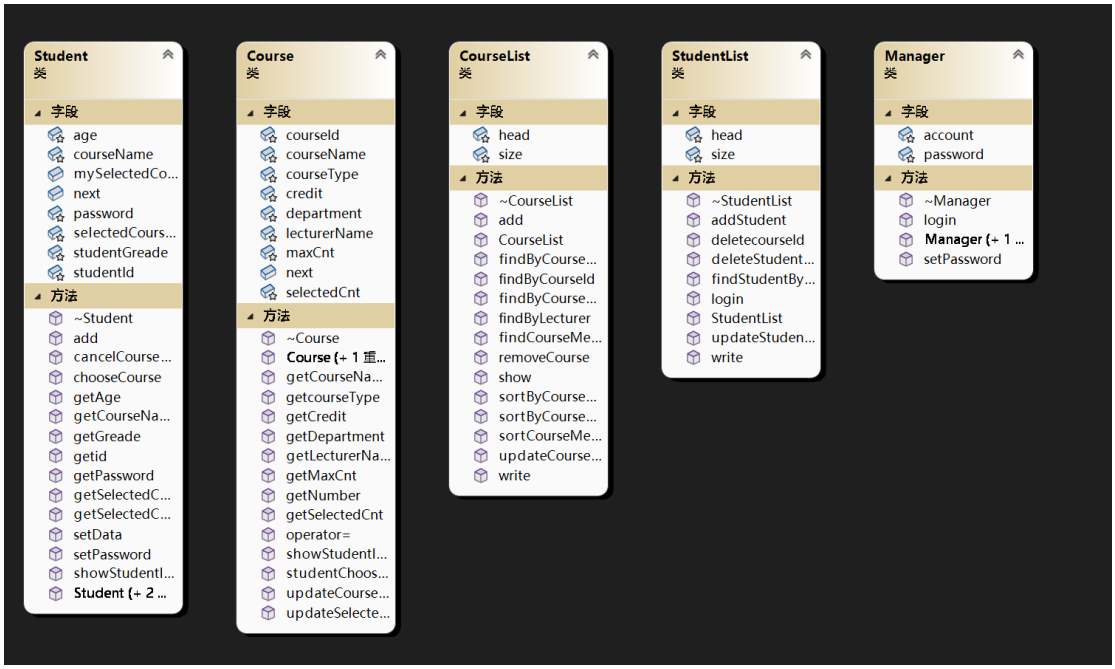


图 3

分别设计学生类、管理员类，学生链表类，课程链表类，管理员类

用文本文件进行数据的保存，需要保存的数据主要包括图书数据、用户数据（包括读者、课程、管理员），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：学生开课管理系统。包含了 Student 类，Course 类，Manager 类三个基本类以及相对应的链表类 StudentList 类（学生链表类）及 CourseList（课程链表类）。具体类结构声明如下

- Student

```
#pragma once
#include<string>
using namespace std;
class Student {
public:
    Student(string, string, int, int, string, int, int*);
    Student(string, string, int, int, string);
    virtual ~Student();
    void showStudentInfo();//展示个人信息
    void add(int temp);//将课程编号编入 mySelectedCourses 末尾
    Student(const Student& student);
    Student* next;
    //选课或取消已选的课
    int chooseCourse(int id, bool choiceable);
    //根据课程号获取自己选次课的序号，没有返回-1
    int getSelectedCoursePlaceById(int id);
    //取消选课
    void cancelCourseByPlaceId(int id);
    int* mySelectedCourses;//存放学生所选课程的学生编号

    //getter setter
    string getid();//得到学号
    string getCourseName();
    int getAge();
    int getGreade();
    string getPassword();
    int getSelectedCourseCnt();
    void setPassword();//修改密码
    void setData(string id, string na, int a, int ey, string pa);
protected:
    string studentId;//学号
    string courseName;
    string password;//密码
    int age;
    int studentGreade;//入学年份
    int selectedCourseCnt;//已选课科目数量
};
```

- Course

```
#pragma once
#include<string>
using namespace std;
class Course {
public:
    Course* next;
    Course() {};
    Course(int id, string name, string teacherName, double credit, string
type, string department, int cnt, int s);
    virtual ~Course();
    void showStudentInfo();
    void updateCourseInfo(string, string, double, string, string);
    Course& operator =(Course& c);
    bool studentChooseable();
    void updateSelectedCnt(int x);

    //getter setter
    int getNumber();
    string getCourseName();
    string getLecturerName();
    double getCredit();
    string getcourseType();
    string getDepartment();
    int getSelectedCnt();
    int getMaxCnt();

protected:
    int courseId;//课程编号
    string courseName;
    string lecturerName;//任课教师
    double credit;//学分
    string courseType;//课程性质
    string department;//开课院系
    int selectedCnt;//已选该课的人数
    int maxCnt;//该科可选的最大人数
};
```

- Manager

```
#pragma once
#include<string>
using namespace std;
class Manager {
```

```
public:
    Manager(string, string);
    Manager() {};
    virtual ~Manager();
    bool login();
    void setPassword();
protected:
    string account;
    string password;
};
```

（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。

学生	课程	学生链表	课程链表	管理员
学生数据	课程数据	链表表头	链表表头	数据
学生类指针（*next）	课程类指针（*next）	链表大小	链表大小	

图 4

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 CourseList 类来管理课程数据，具体的结构声明如下：

- CourseList

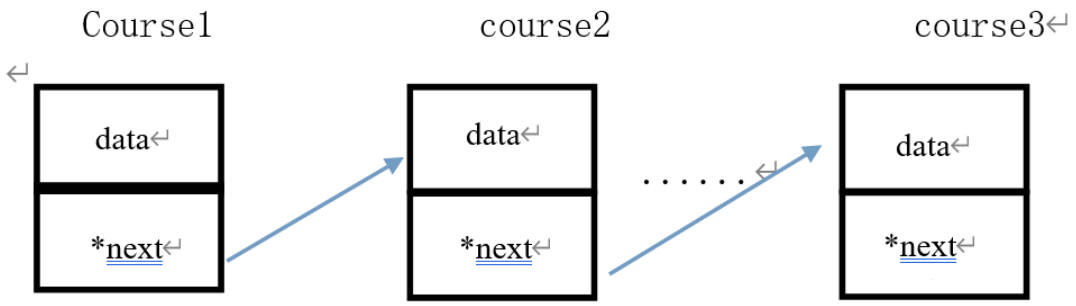
```
#pragma once
#include"Course.hpp"
using namespace std;
class CourseList{
    public:
        CourseList();
        virtual ~CourseList();
        //尾插法
        void add(Course &cou);
        //按编号查找
        Course *findByCourseId(int id);
        //按名字查找
        void findByCourseName(string n);
        //按院系查找
        void findByCourseDepartment(string f);
```

```
        //按教师名字查找
        void findByLecturer(string t);
        //查找
        void findCourseMenu();
        //展示所有课程信息
        void show();
        //写入文件
        void write();
        //按学分排序
        void sortByCourseCredit();
        //按开课学院排序
        void sortByCourseDepartment();

        void sortCourseMenu();//排序
        //修改信息
        void updateCourseInfo(Course *target);
        //删除课程
        void removeCourse(int id);
    protected:
        Course *head;
        int size;

};
```

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。



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 StudentList，各个类的结构声明如下：

- StudentList

```
#pragma once
#include"Student.hpp"
class StudentList {
public:
```

```
    StudentList();
    virtual ~StudentList();
    //头插法添加学生
    void addStudent(Student& stu2);
    //学生登录
    Student* login();
    //信息录入文件
    void write();
    //根据学号查找学生
    Student* findStudentById(string id);
    //取消 id 为 id 的课的选课记录
    void deletecourseId(int id);
    //根据学号更新学生信息
    void updateStudentInfo(string id);
    //根据学生姓名删除学生
    void deleteStudentByName(string id);//删除特定学生
protected:
    Student* head;
    int size;
};
```

图书管理系统的信息的管理就具体表现为链表的操作。拿图书信息来说，图书信息的查找、修改、添加和删除与链表的查找、修改、添加、删除对应

- **课程的查找：**

在用户在查询课程时，有四种不同的模式，见图 4：

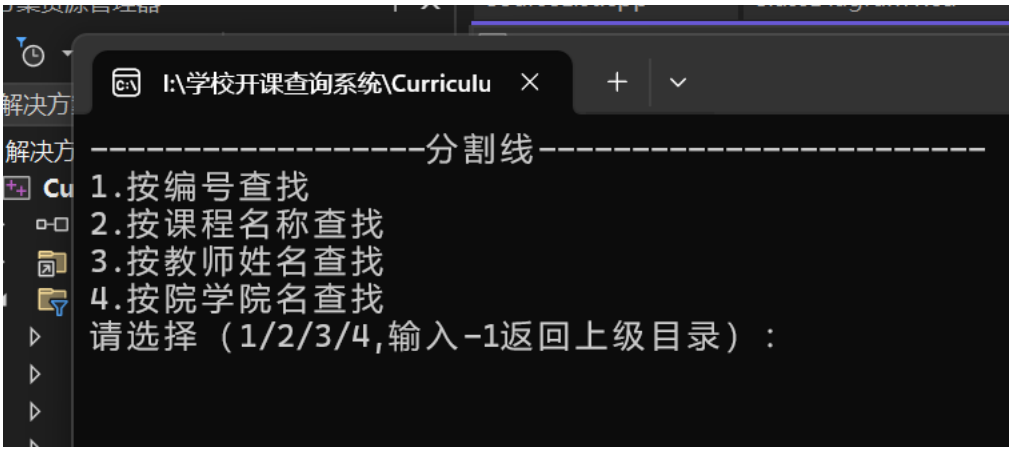


图 5

这三种情况都需要对链表中的所有数据进行顺序的搜索。 例如按院系的查找，定义一个 Book 类指针变量 p 并对其初始化，使其为 head。先精确查询，如果查到了就直

接返回。否则模糊查询所有元素的相似度（相同字符数量/被查询元素长度），输出所有相似的最大的元素信息

课程查找的流程图如下

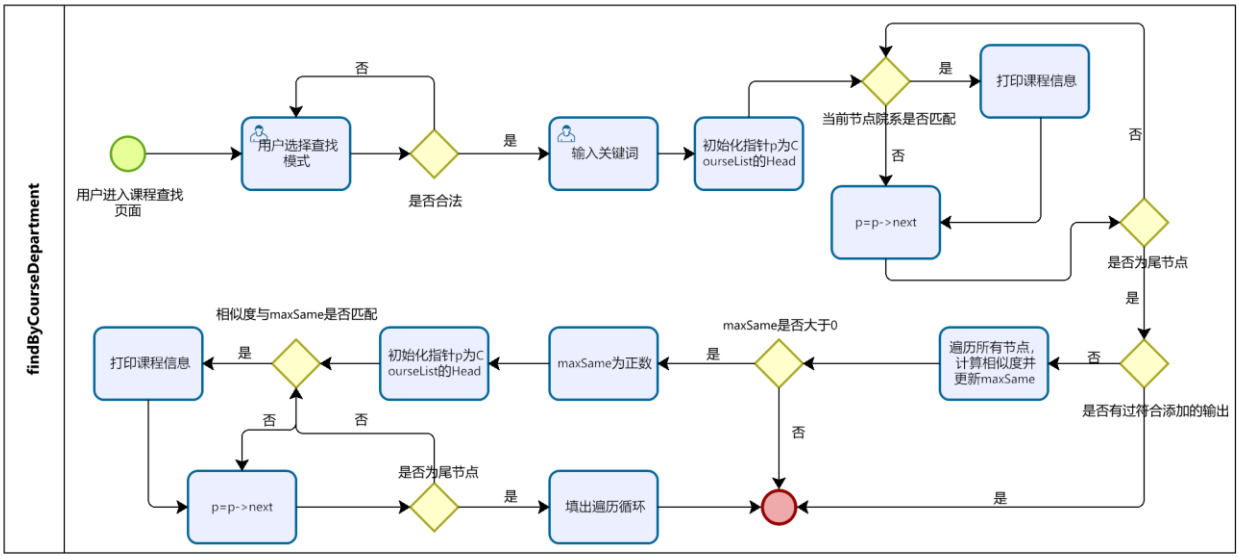


图 6 程序流程图

- **课程的修改：**
  管理员输入选中课程——输入新的课程属性——程序正常退出——完成修改
- **课程的添加：**
  管理员输入选中课程——输入新课程属性——新课程加入 CourseList——程序正常退出——完成添加
- **课程的删除：**
  管理员输入选中课程——程序正常退出——完成删除
- **学生信息的修改删除添加**
  类似上述过程，不再赘述

（3）交互界面以及登录菜单的实现

- **主页面**

  Main 函数中有一个 switchCase 用于控制一级目录跳转，并利用 continueRunning 变量控制程序结束。不同输入调用不同子菜单的函数进而实现跳转，同时做了非法输入的处理。

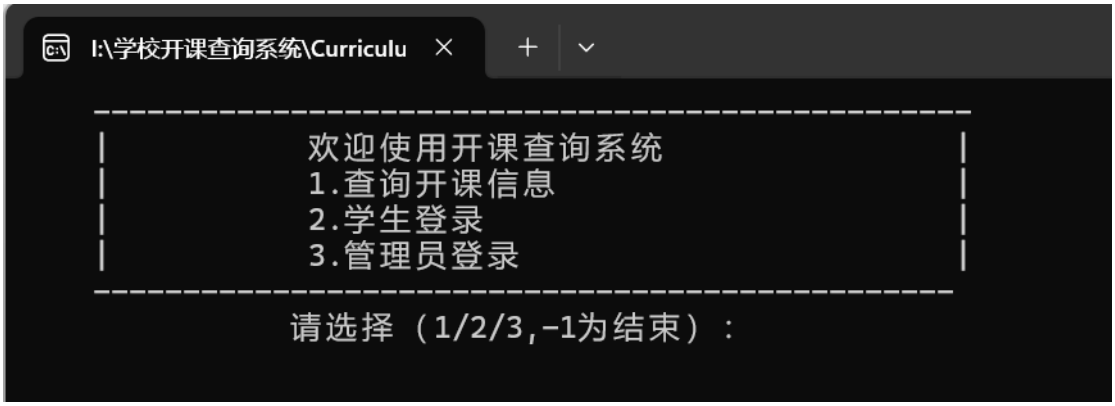


图 7

```
int main() {
    init();
    bool continueRunning = true;
    while (continueRunning) {
        bool subMenuRunning = true;
        showMainMenu();
        int mainMenuId; cin >> mainMenuId;
        system("cls");
        switch (mainMenuId) {
        case 1:
            subMenu1(subMenuRunning);
            break;
        case 2: {
            subMenu2(subMenuRunning);
            break;
        }
        case 3:
            subMenu3(subMenuRunning);
            break;
        case -1:
            continueRunning = false; break;
        default:
            cout << "输入错误" << endl;
            break;
        }
    }
```

图 8

- **游客查询页面**

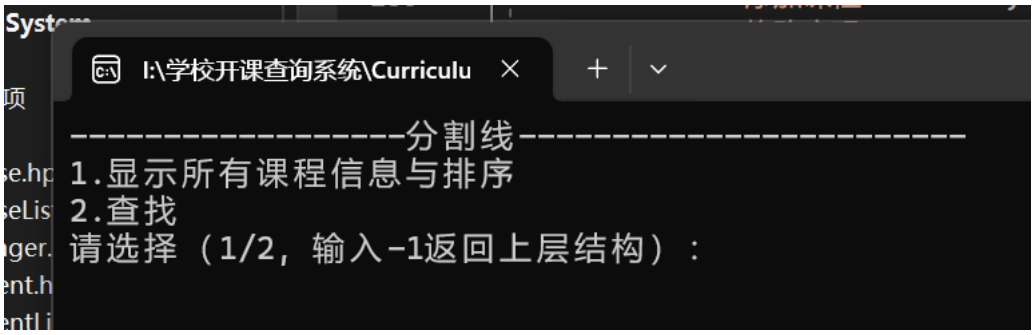


图 9

- **学生登录后页面**

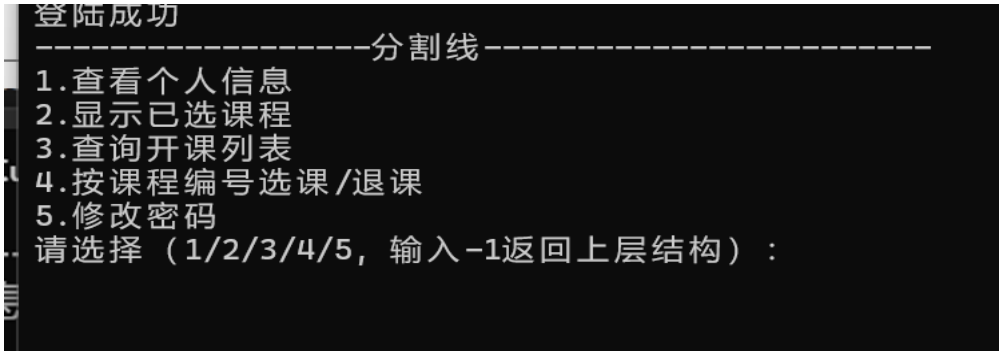


图 10

- **管理员登录后页面**

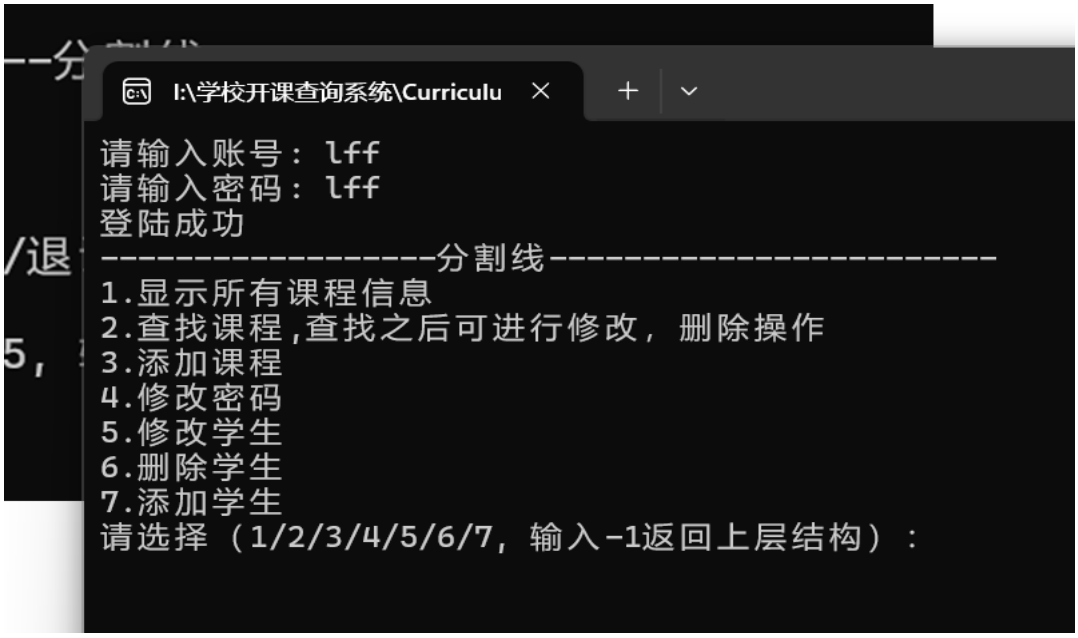


图 11

（4）选课与退课功能实现

学生有 int* mySelectedCourses;属性用于记录所有以及选择的课程编号，选课时会根据是否以及记录来判断时选课还是退选。如果时选课且课程未达到最大容量会记录入数组并更新课程的已选人数，如果已经达到最大人数会选课失败。如果时退选会移除记录并将课程已选人数缩减，课程默认容量 50。课程被删除时会取消所有同学选课记录。

```
int Student::chooseCourse(int id,bool choiceable) {
    int option = 2;
    //没选过
    if (getSelectedCoursePlaceById(id) == -1) {
        cout << "是否要添加这门课,是请输入1，否请输入-1" << endl;
        while (option != 1 && option != -1) {
            cin >> option;
            if (option == 1 || option == -1) {
                if (option == 1) {
                    if(choiceable)
                        add(id);
                    else {
                        cout << "无法选择，已达最大可选人数！\n";
                        return 0;
                    }
                }
                cout << "操作成功" << endl;
                return 1;
            }
        }
    }
    //选过了
    else {
        cout << "您已选了这门课，是否要取消，是请输入1，否请输入-1" << endl;
        while (option != 1 && option != -1) {
            cin >> option;
            if (option == 1 || option == -1) {
                if (option == 1){
                    cancelCourseByPlaceId(getSelectedCoursePlaceById(id));
                }
                cout << "操作成功" << endl;
                return -1;
            }
        }
    }
```

图 12

四、 实验调试、测试、运行记录及分析

- 游客功能
  - 显示所有课程信息与排序

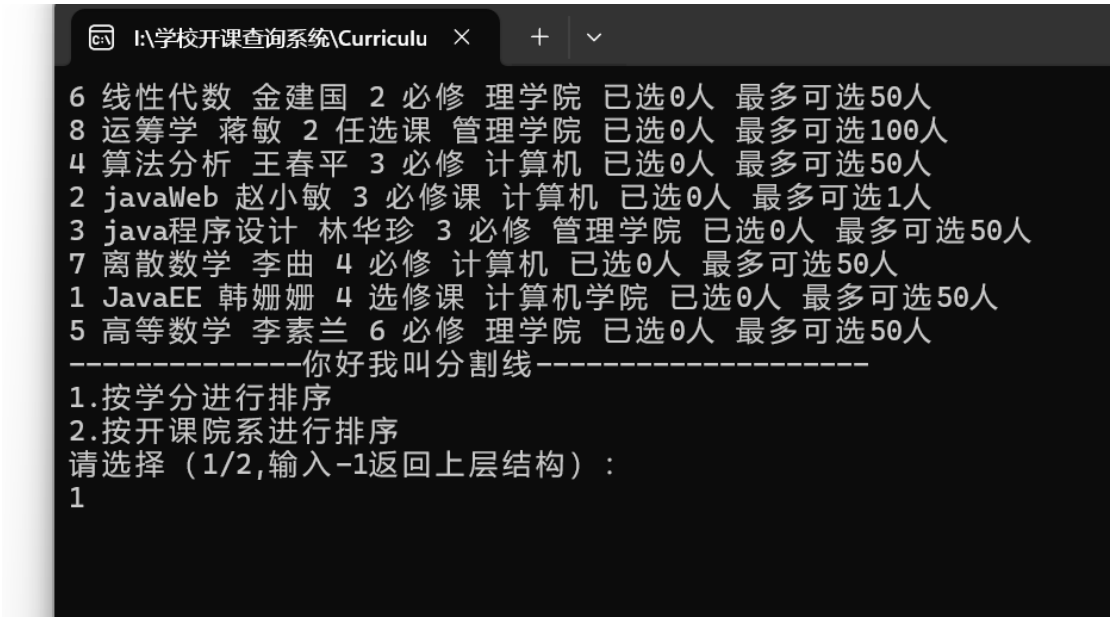


图 13

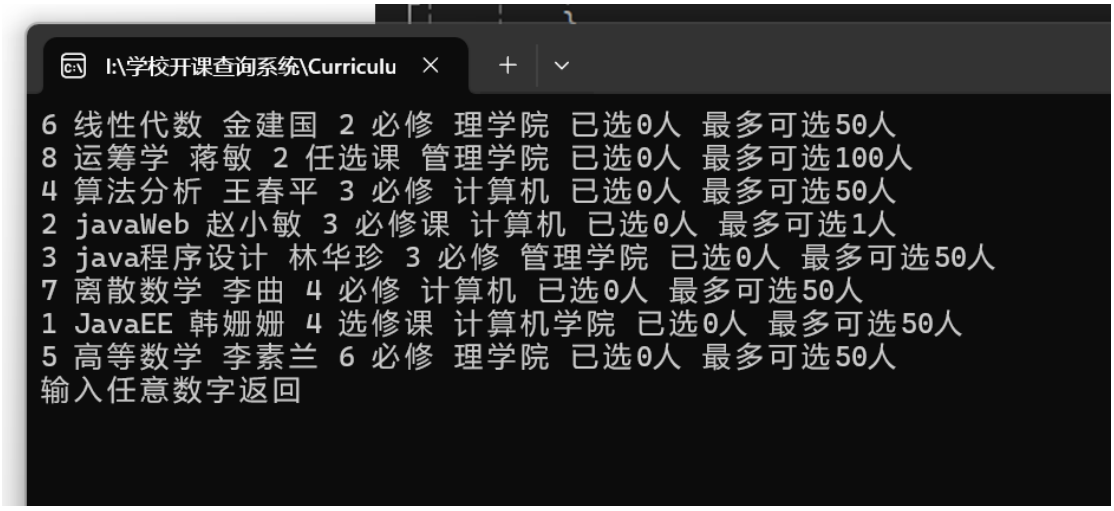


图 14

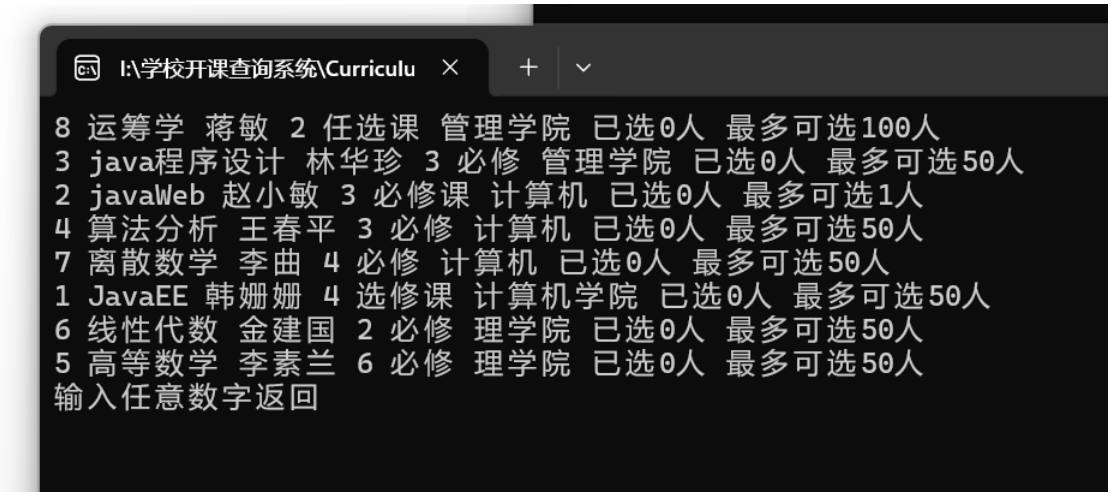


图 15

- 查找

  见下方管理员演示

- 学生功能
  - 登录

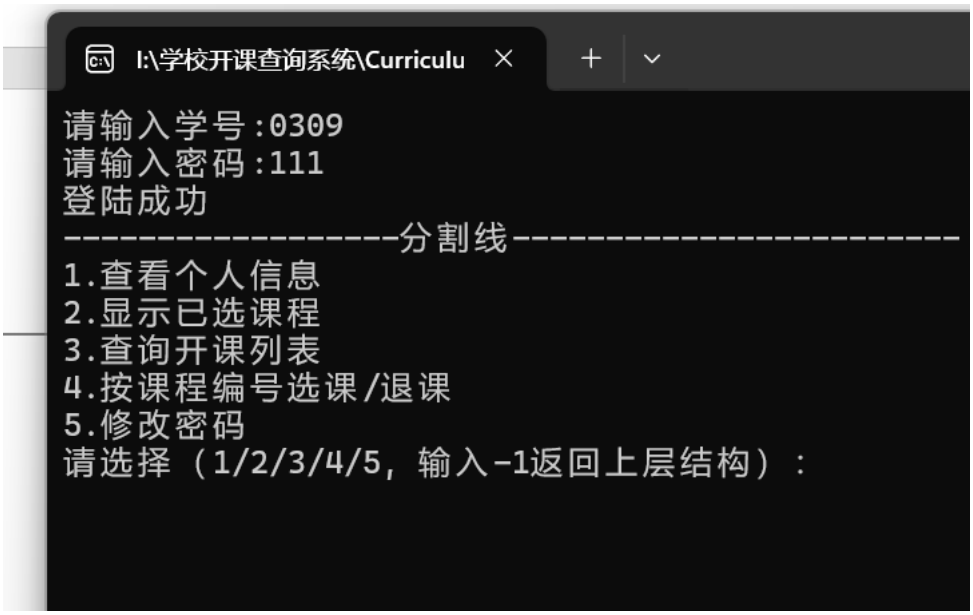


图 16

■ 查看个人信息

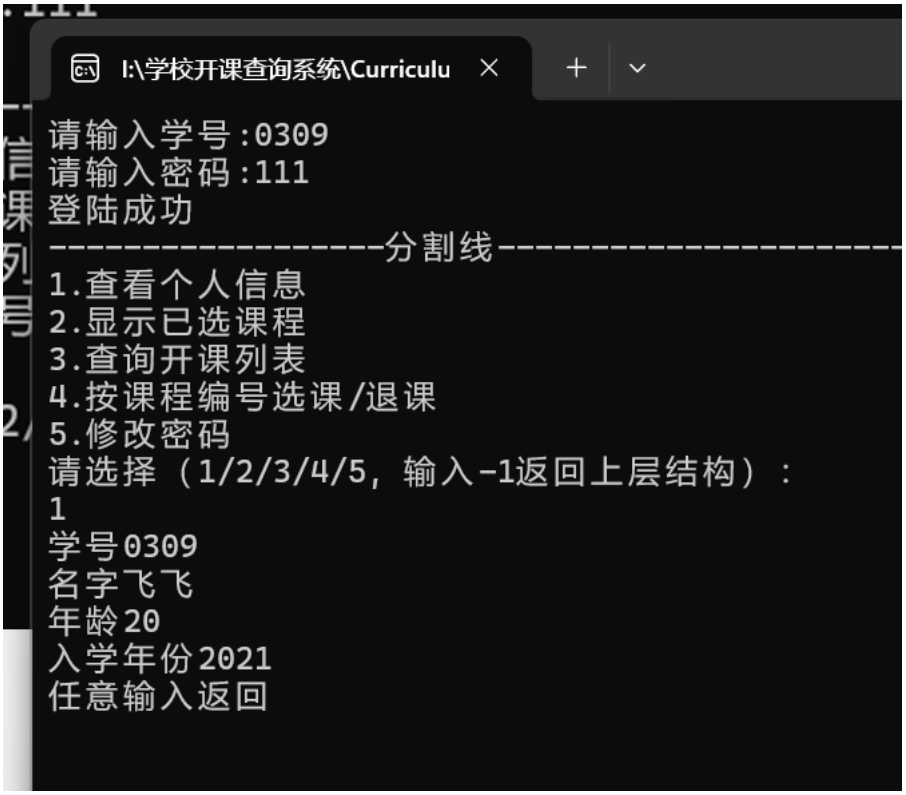


图 17

■ 显示已选课程

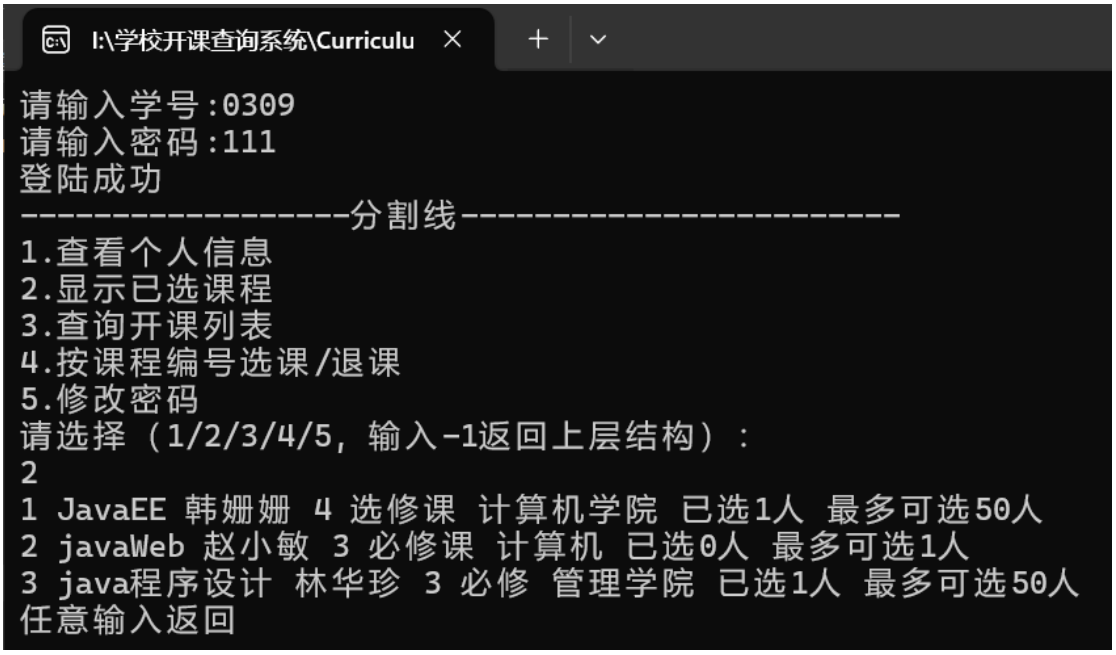


图 18

- 查询开课列表

  见管理员演示

- 按课程编号选课/退课

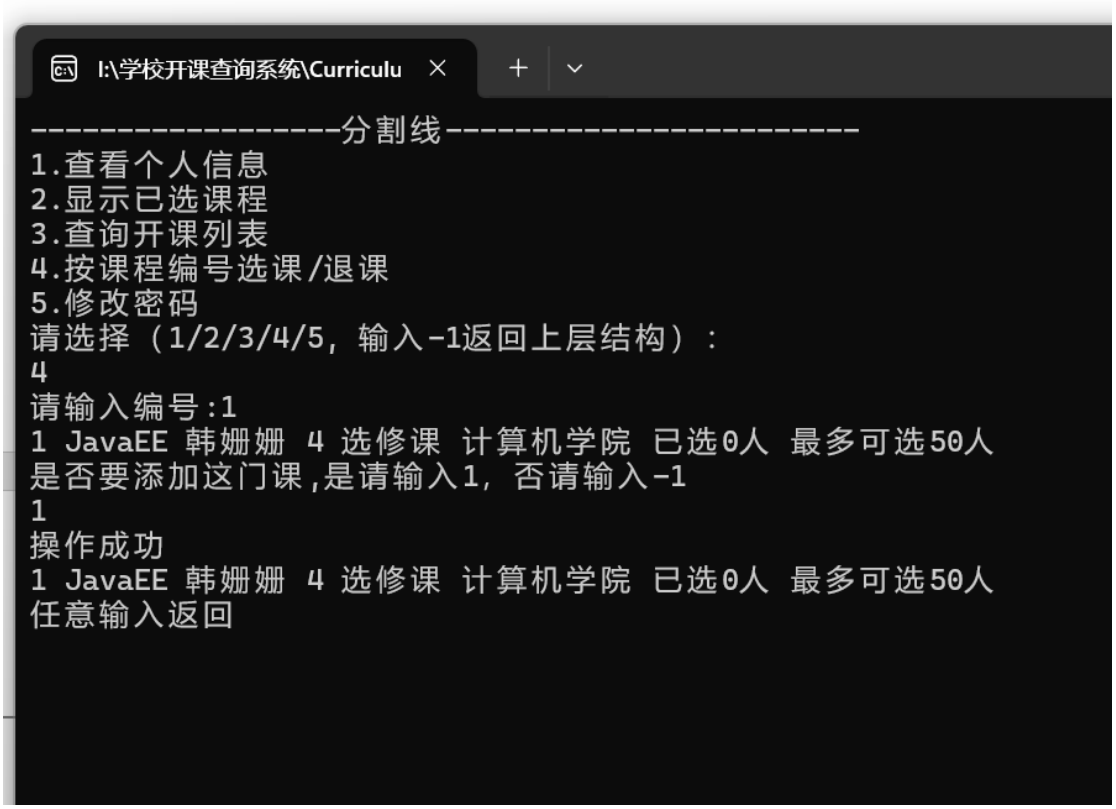


图 19

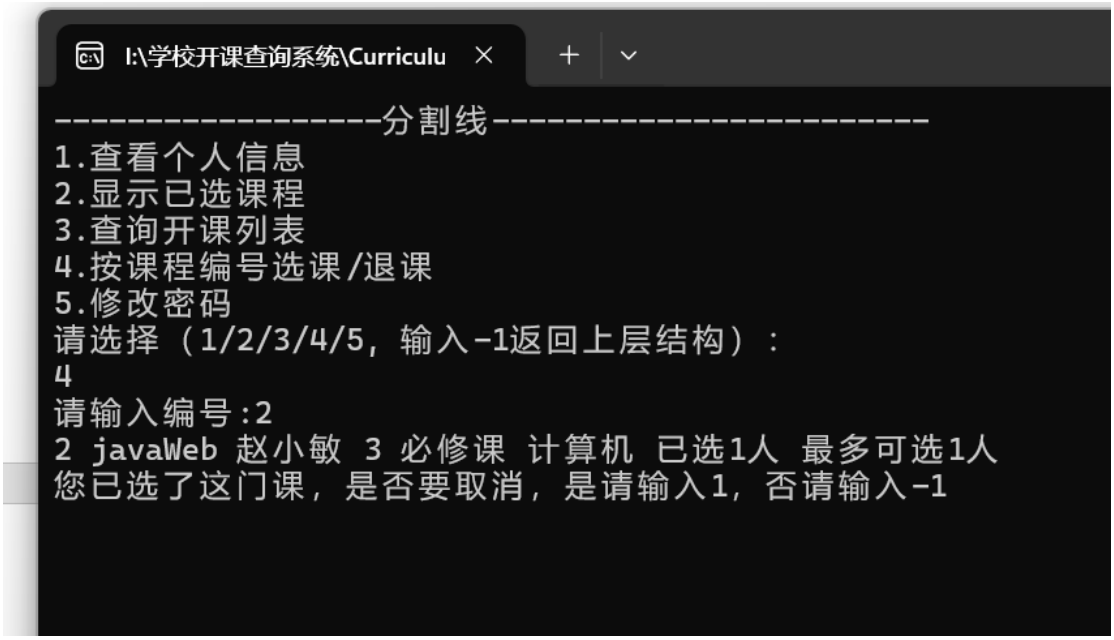


图 20

- 修改密码

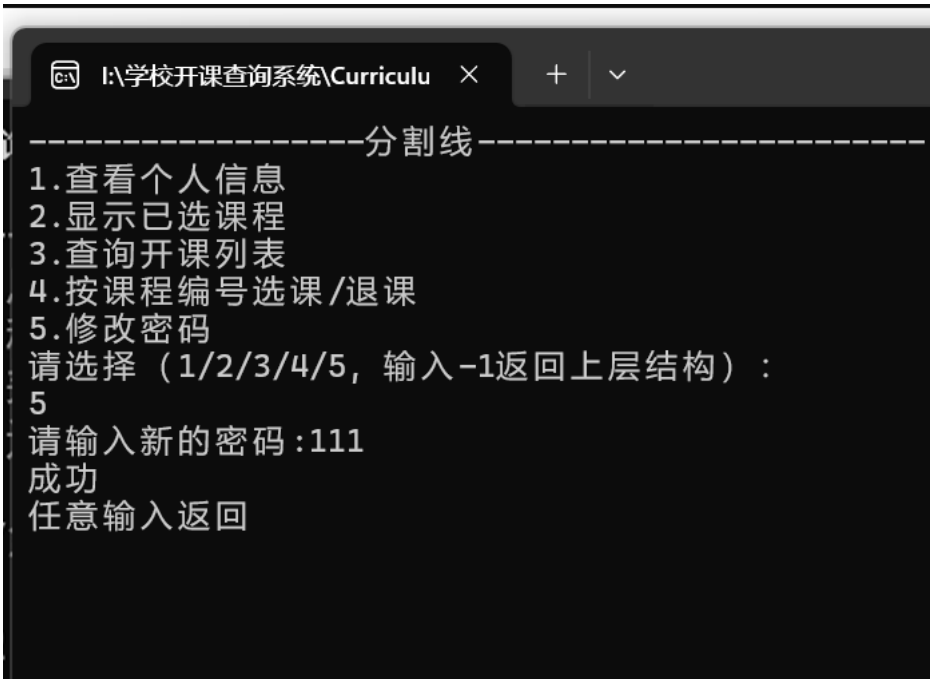


图 21

- 管理员功能
  - 展示所有课程

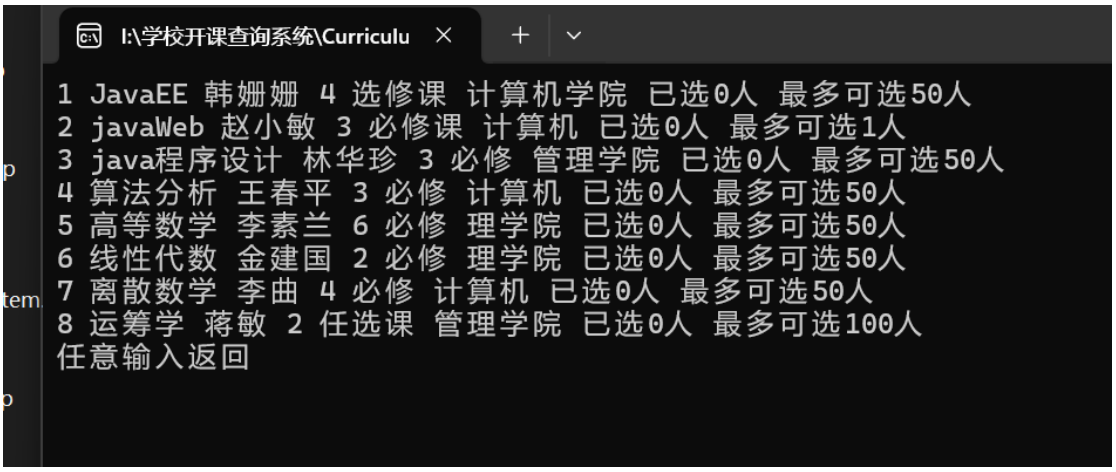


图 22

  - 查找课程并修改

◆ 模糊查询

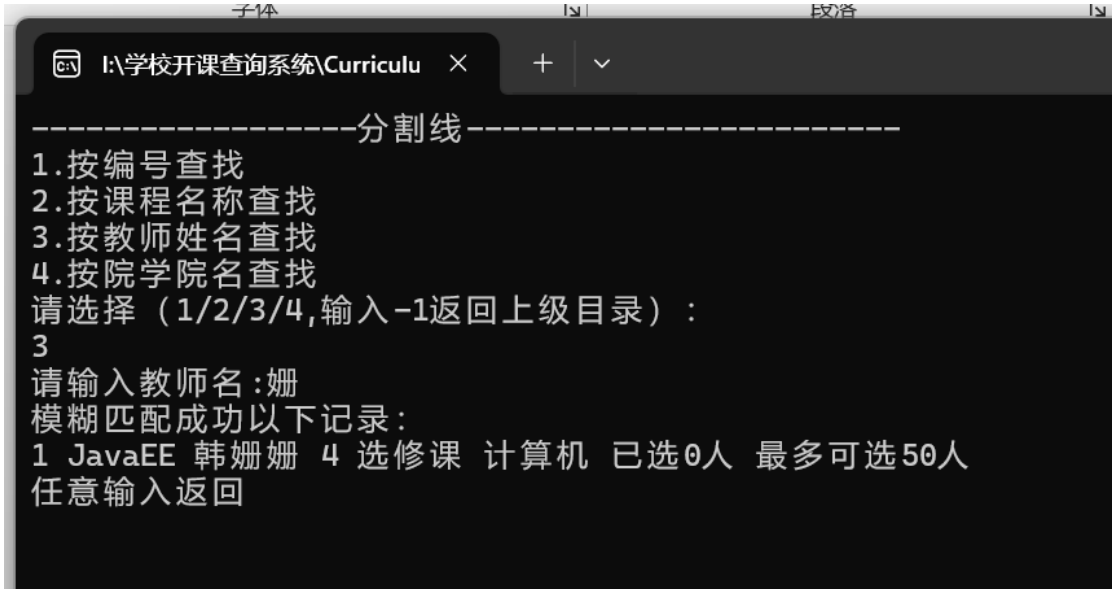

I:\学校开课查询系统\Curriculu
----------分割线----------
1.按编号查找
2.按课程名称查找
3.按教师姓名查找
4.按院学院名查找
请选择（1/2/3/4,输入-1返回上级目录）：
3
请输入教师名:姗
模糊匹配成功以下记录:
1 JavaEE 韩姗姗 4 选修课 计算机 已选0人 最多可选50人
任意输入返回

图 23

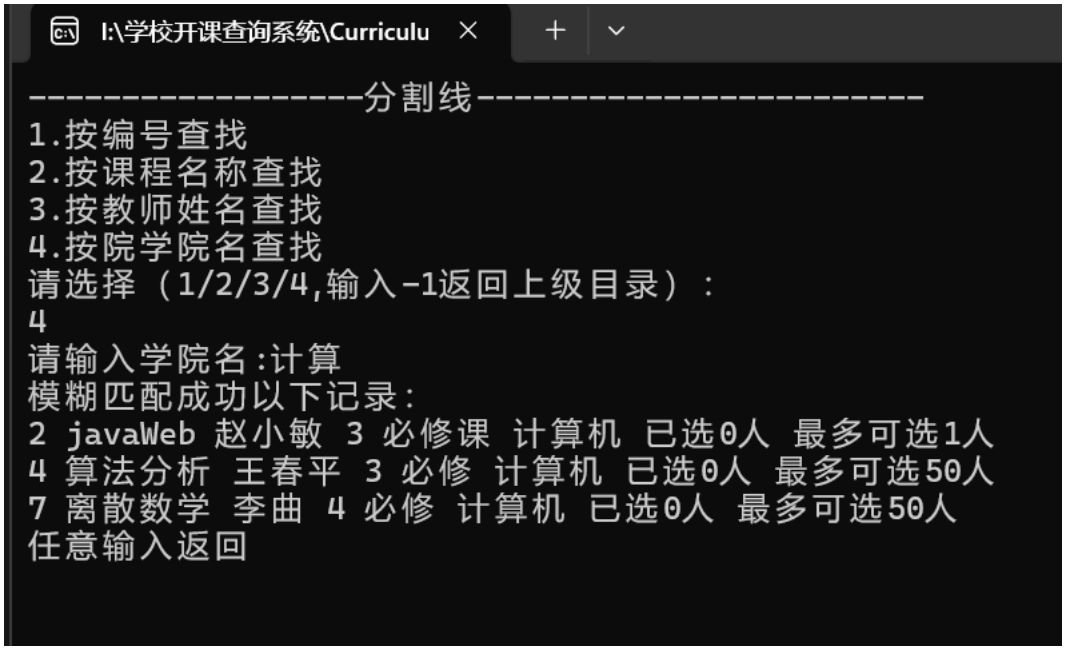

I:\学校开课查询系统\Curriculu
----------分割线----------
1.按编号查找
2.按课程名称查找
3.按教师姓名查找
4.按院学院名查找
请选择（1/2/3/4,输入-1返回上级目录）：
4
请输入学院名:计算
模糊匹配成功以下记录:
2 javaWeb 赵小敏 3 必修课 计算机 已选0人 最多可选1人
4 算法分析 王春平 3 必修 计算机 已选0人 最多可选50人
7 离散数学 李曲 4 必修 计算机 已选0人 最多可选50人
任意输入返回

图 24

◆ 修改课程

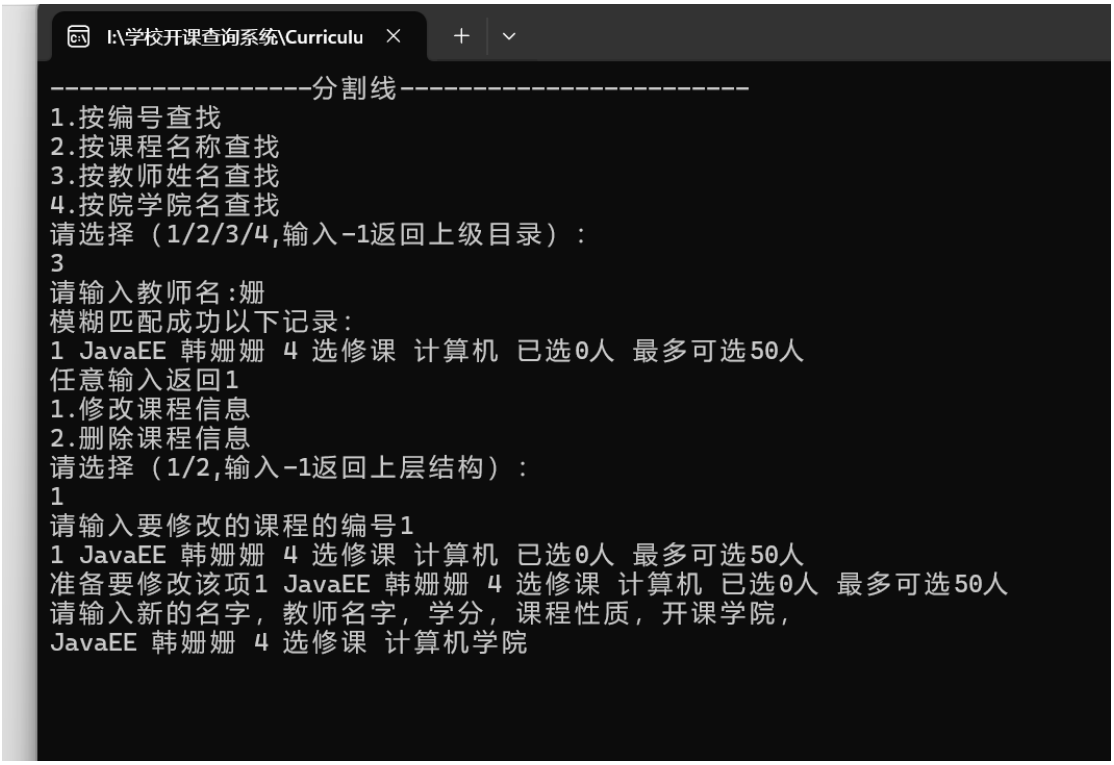


图 25

■ 添加课程

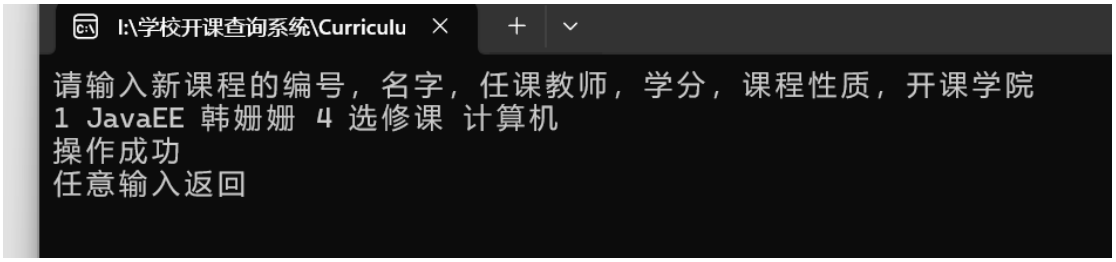


图 26

■ 修改密码

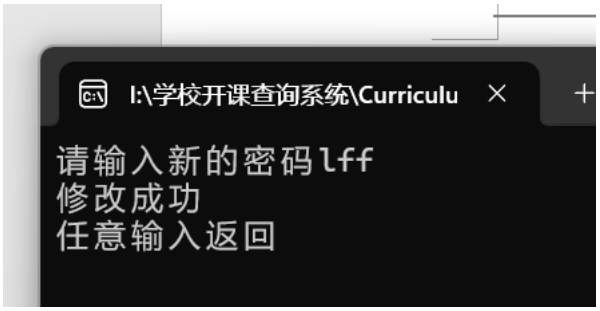


图 27

■ 修改学生

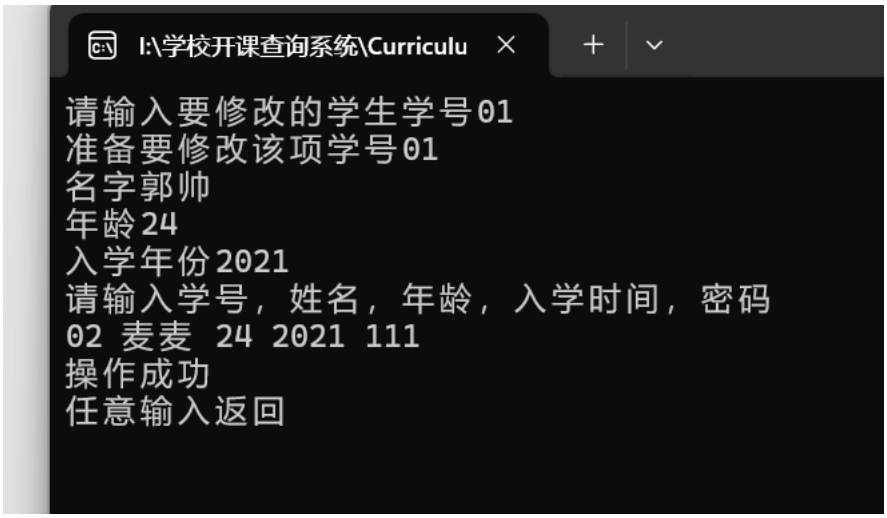


图 28

■ 删除学生

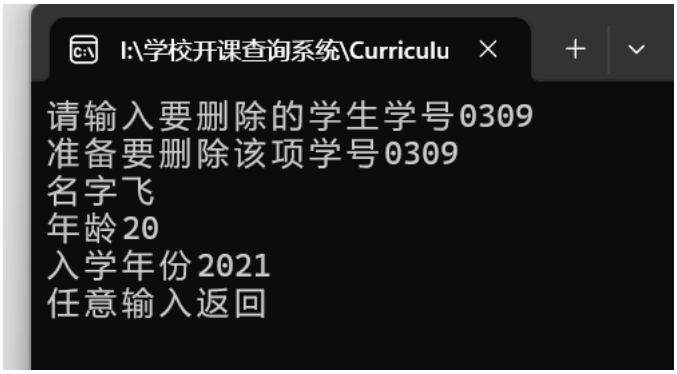


图 29

- 添加学生

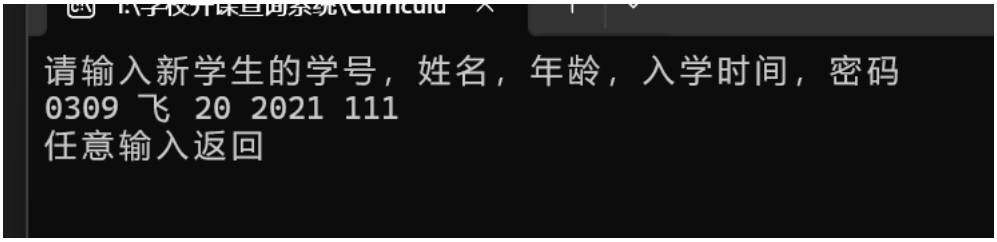


图 30

五、 遇到的问题及解决方法如下:

- **问题 1：**

问题描述： 跟新选课人数时将 CourseList.hpp 加入 Student.hpp 后报错

解决方法：刚开始以为时循环依赖了，结果看了半天也没发现哪里循环，无奈改了改名结构，将课程是否还能选择放入主函数判断，结果还是编辑报错，最后发现是一个构造函数没实现导致的 QAQ

- **问题 2**

问题描述：在 switchCase 下声明变量报错不能声明临时变量

解决方法：搜索后发现要加{}才行，加了括号解决

- **问题 3**

问题描述：在按照导师名匹配时输入“计算机“会错误的结果（应该无返回才对）

解决方法：中文占用了两个 char，匹配时同时匹配两个字符即可

六、 附录：源代码