（报告封面）

合肥工業大學

“数据结构与算法”

课程设计报告

设计题目 机房预约系统

姓　　名 党存远

学　　号 2022217587

专　　业 物联网工程

班　　级 物联网工程 22-2

完成日期 2023.6.30

“数据结构与算法课程设计”项目验收细则			
成绩等级	具体表现	教师评分	
优秀（100-85]	1)能够在规定时间内完成项目，项目具有完整性，交互设计优秀；2）工作量充足；3）设计思路清晰，熟悉所采用的算法，能够进行清晰完整的回答；	☐	
良好（85-75]	1）能够在规定时间内完成项目，且项目具有较好的完整性，交互设计较为优秀；2）工作量较为充足；3）设计思路较为清晰，对相应算法较为熟悉，能够进行较为明确的回答；	☐	
中等（75-65]	1)能够在规定时间内完成项目，项目完整性较好，交互设计一般；2）工作量一般；	☐	
及格（65-60]	1）虽完成项目编码但未能在规定时间内完成的；	☐	
不及格（<60）	1）未进行验收的；	☐	

“数据结构与算法课程设计”报告评分细则			
成绩等级	具体表现	教师评分	
优秀（100-85]	1）实习报告格式完美，充分采用图表、伪代码、流程图多种形式来说明问题，章节设计优秀，工作量饱满；	☐	
良好（85-75]	1)实习报告格式良好，采用了图表等来说明问题，章节设计良好，工作量较为饱满；	☐	
中等（75-65]	1）实习报告格式较好，章节设计较好，工作量一般；	☐	
及格（65-60]	1）虽提交报告但未能在规定时间内完成的；	☐	
不及格（<60）	1）未提交报告的；	☐	

教师评语：   教师签名：　　时间：

1 题目

学校现有几个规格不同的机房，由于使用时经常出现"撞车"现象，开发一套机房预约系统，当通过不同的身份进入此系统可以完成不同操作，实现学生对不同机房进行申请，教师对机房进行预约以及对学生的申请进行允许或拒绝，管理员对系统进行维护等。

2 需求分析与设计

2.1 需求分析

该系统需要设计一个机房预约的管理系统来实现预约机房的操作，同时对该系统的使用成员进行分类，不同身份进入该系统后可以选择不同的操作。

●系统简介
学校有几个规格不同的机房，现开发一套机房预约系统，解决对机房预约使用的问题

●身份简介【有三种身份使用该程序】

▪学生：申请使用机房
▪教师：审核学生的预约申请
▪管理员：给学生、教师创建账号

●机房简介{机房总共有三间【可以根据需求来扩建机房的数量与不同机房的容量】}

▪1 号机房 — 最大容量 20 人
▪2 号机房 — 最大容量 50 人
▪3 号机房 — 最大容量 100 人

●申请简介

▪申请的订单每周由管理员负责清空与删除操作
▪学生可以预约未来一周内的机房使用，预约的日期为周一至周五，预约时需要选择预约时段(上午、下午)
▪教师来审核预约，依赖实际情况审核预约通过或者不通过

2.2 系统设计

首先进入登录界面，可选择的操作有：

▪学生代表
▪老师
▪管理员
▪退出

//每个身份都需要进行验证后，才可以进入子菜单进行机房预约的操作

•学生需要输入：学号、姓名、登录密码
•老师需要输入：职工号、姓名、登录密码
•管理员需要输入：管理员姓名、登录密码

1、学生具体功能

.申请预约 — 预约机房
.查看自身的预约 — 查看自己的预约状态
.查看所有预约 — 查看全部预约信息以及预约状态
.取消预约 — 取消自身的预约，预约成功或审核中的预约均可以取消
.注销登录 — 退出登录

2、教师具体功能

.查看所有预约 — 查看全部预约信息以及预约状态
.审核预约 — 对学生的预约进行审核
.注销登录 — 退出登录

3、管理员具体功能

.添加账号 — 添加学生或教师的账号，需要检测学生编号或教师职工号是否重复
.查看账号 — 可以选择查看学生或教师的全部信息
.查看机房 — 查看所有机房的信息

图一:机房预约系统操作流程图

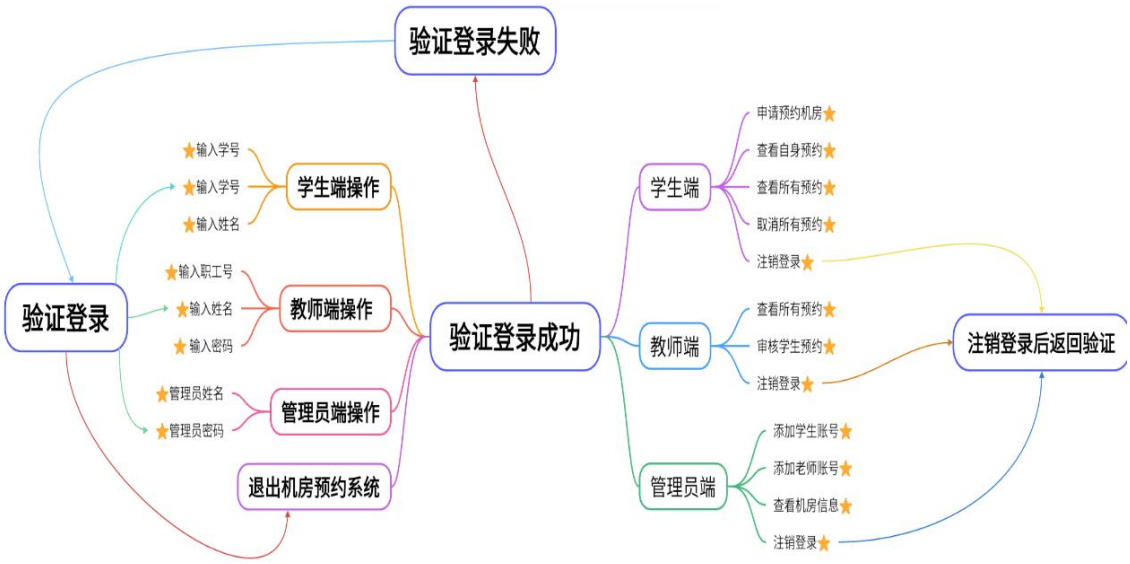


表 1:重要数据结构

数据结构	数据类型	数据项	描述
Identity	string	m_Name	存储用户名
	string	m_Pwd	存储用户密码
Student	int	m_Id	存储学生学号
	Vector<C>	vCom	存储机房信息的容器
Teacher	int	m_EmoId	存储教师职工号码
Manager	Vector<S>	vStu	存放学生容器
	Vector<T>	vTea	存放教师容器
	Vector<C>	vCom	存放机房信息的容器
Computer	int	m_ComId	机房Id号码
	int	m_MaxNum	机房最大容量
GlobalFile	宏类型	File	各类文件的宏定义
OrderFile	int	m_Size	记录预约数量
	map<>	m_OrderData	记录每条预约信息

表 2:重要函数

函数名称	参数	返回值	描述
OpenMenu	virtual	void	不同身份下的菜单
Student	int,string,string	Student	创建学生对象
applyOrder	无参	void	学生申请预约机房函数
showMyOrder	无参	void	查看学生预约信息
cancelOrder	无参	void	取消预约
Teacher	int,string,string	void	取消预约
showAllOrder	无参	void	查看所有预约信息
validOrder	无参	void	审核预约信息
Manager	int,string	Manager	创建管理员对象
addPerson	无参	void	添加账号信息
showPerson	无参	void	查看账号信息
showComputer	无参	void	查看机房信息
cleanFile	无参	void	清空预约记录
initVector	无参	void	初始化容器
studentMenu	Identity*&	void	展示学生端菜单内容
teacherMenu	Identity*&	void	展示教师端菜单内容
managerMenu	Identity*&	void	展示管理员端菜单内容
LoginIn	string,int	void	进行登录与身份验证
OrderFile	无参	void	完成预约信息的存储
updateOrder	无参	void	更新预约记录并存储

表 3:程序结构

一级菜单	二级菜单	三级菜单	四级菜单	五级菜单
开始菜单	用户类型	管理员登入	添加教师学生身份	1、操作完毕返回主菜单
			查看机房信息	2、继续执行管理员操作
		教师登录	查看预约情况	1、返回主菜单
	选择登录		审核预约请求	2、继续执行教师操作
		学生登录	申请、查看、取消机房预约	1、返回主菜单
			查看可预约时段	2、继续执行学生操作
	退出登录	注销登录	/	重新登录
			/	

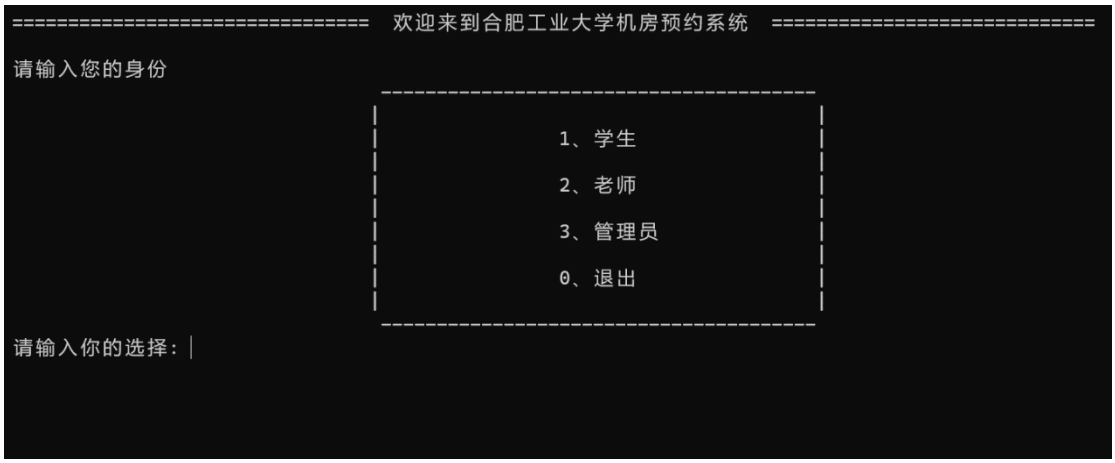
3 系统实现与使用方法

3.1 系统开发环境

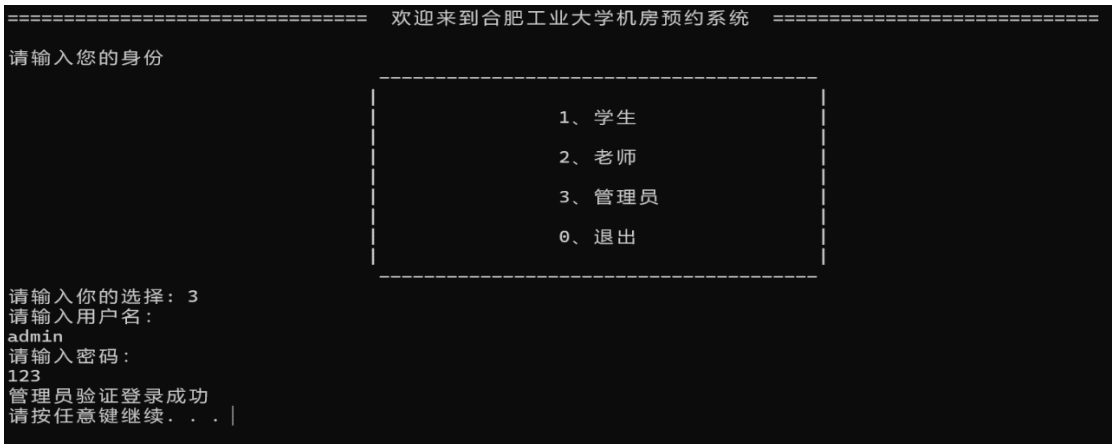
在 windows11 系统下使用 Microsoft Visual Studio 2022 进行该程序系统的开发编写操作。

3.2 系统界面简介

图二:登录菜单界面

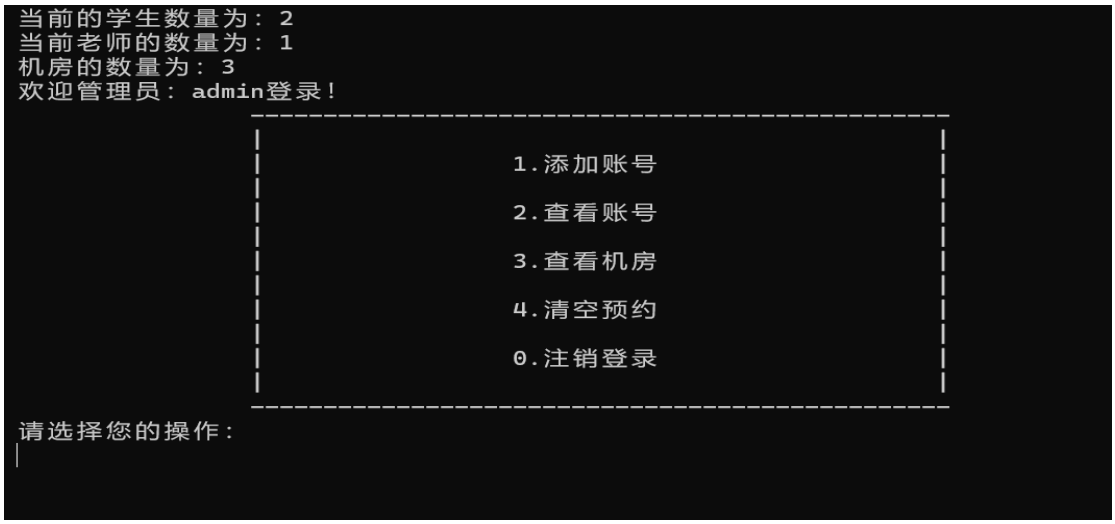


图三:进入管理员菜单界面

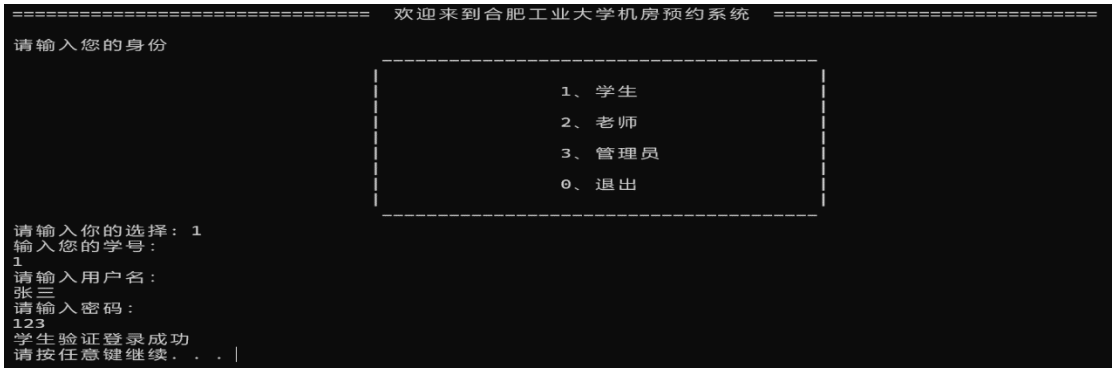


【本系统只设计了一个管理员，添加管理员的方式通过在文件中添加】

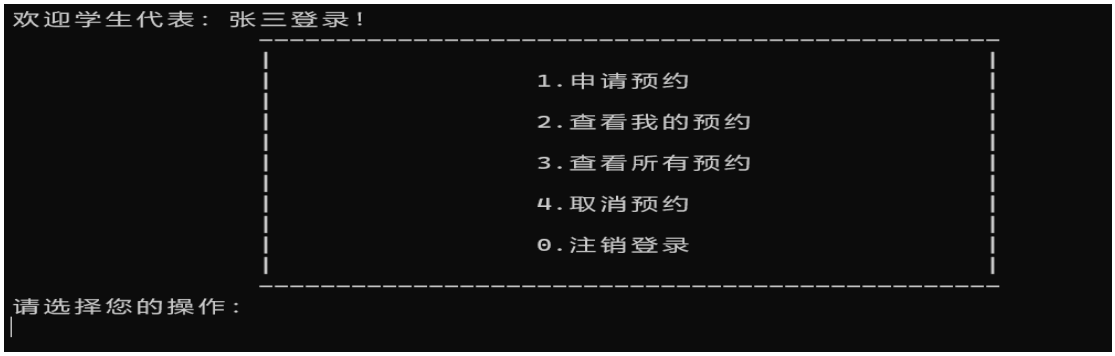
图四:管理员菜单界面



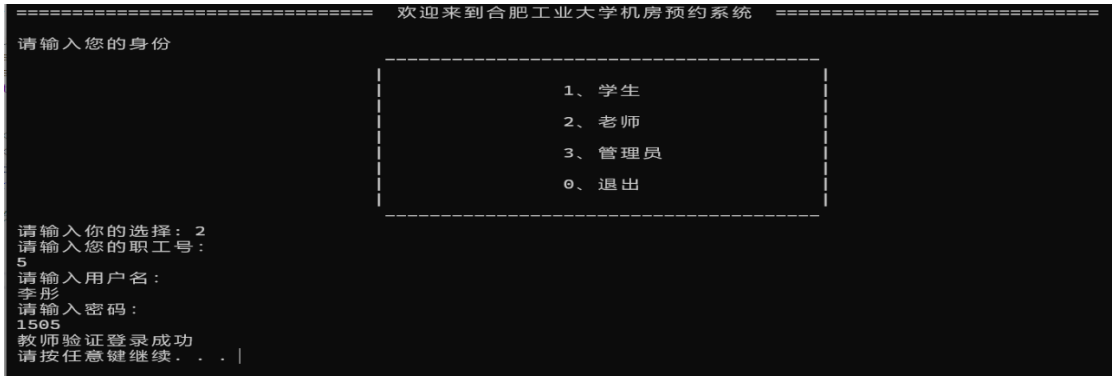
图五:进入学生菜单界面(以学生张三为例)



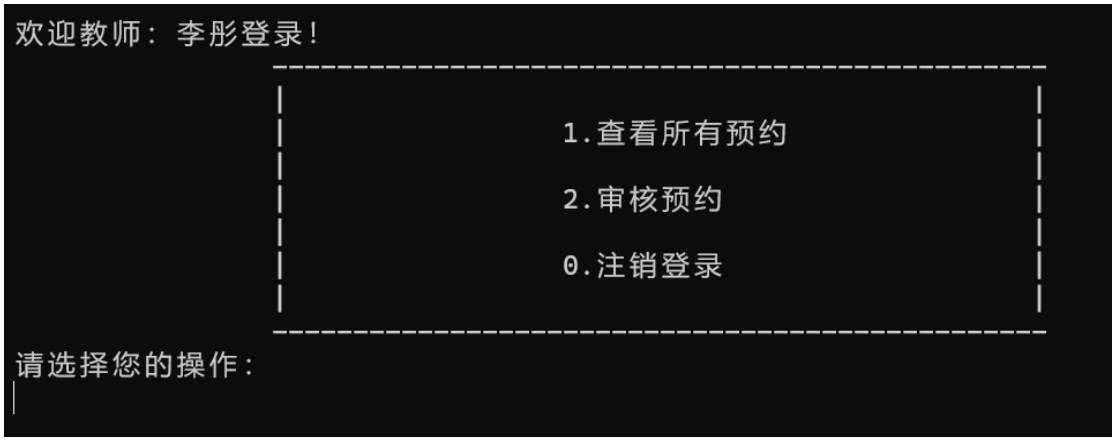
图六:学生菜单界面



图七:进入教师菜单界面(以老师李彤为例)

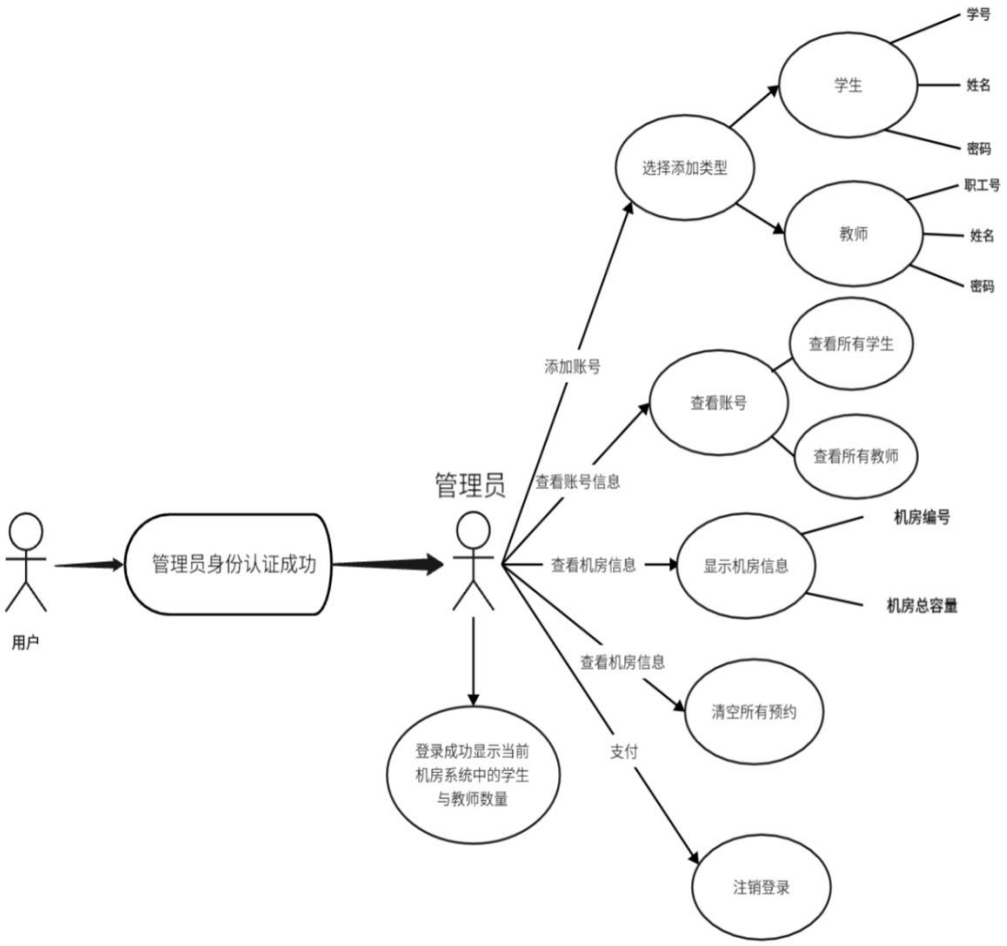


图八:教师菜单界面

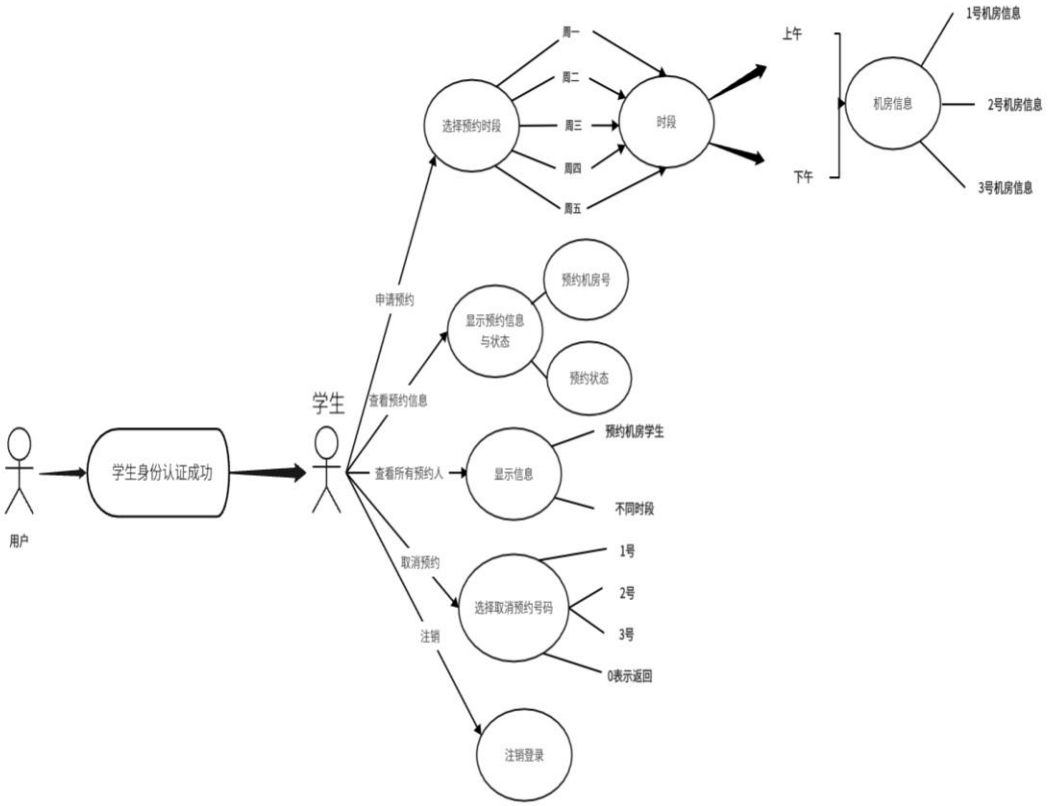


3.3 系统功能模块简介

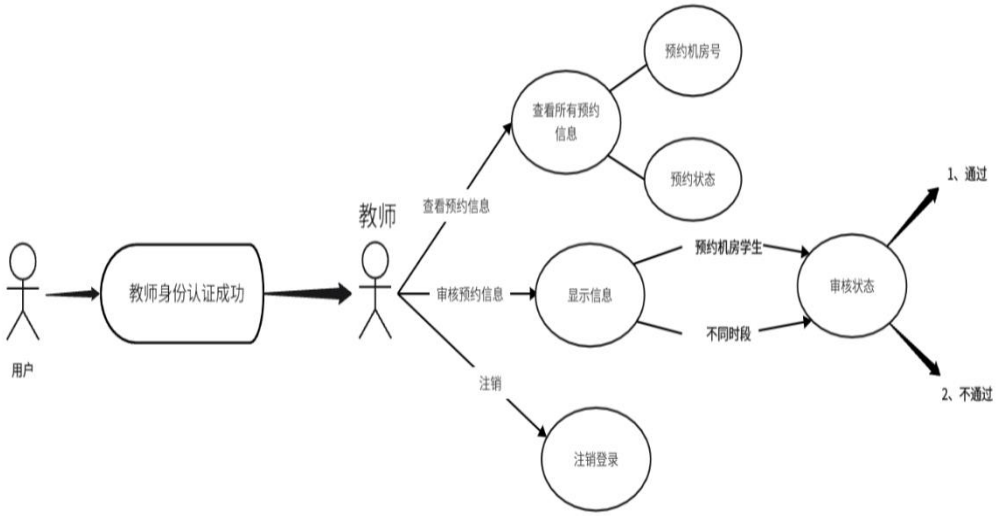
1、管理员模块简介



2、学生模块简介



3、教师模块简介



3.4 重点算法介绍

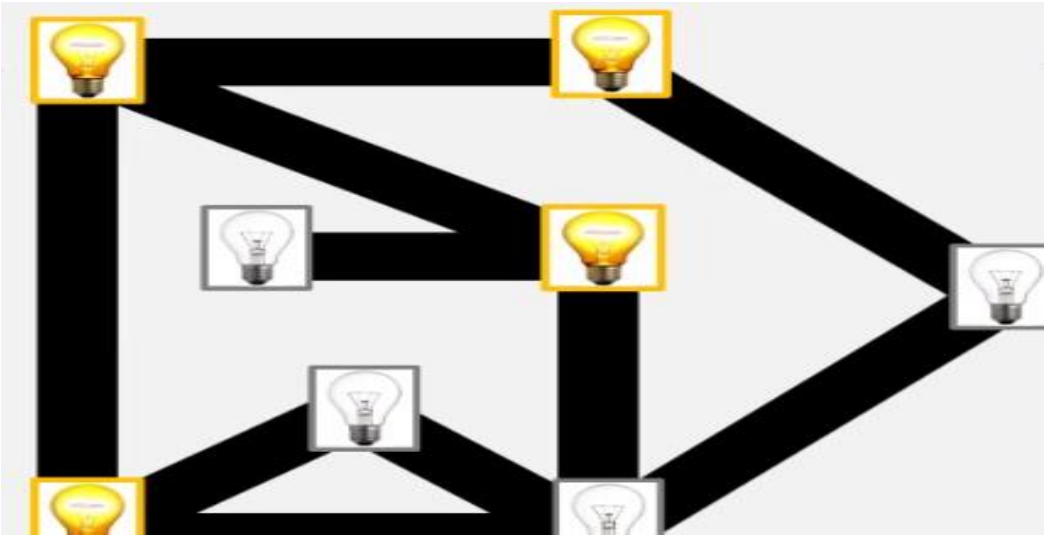
1、判断学生编号是否重复

判断学生的编号是否重复时，将输入进入程序的编号与存储在 student.txt 文件中的学生编号进行比较，判断学生编号是否重复。

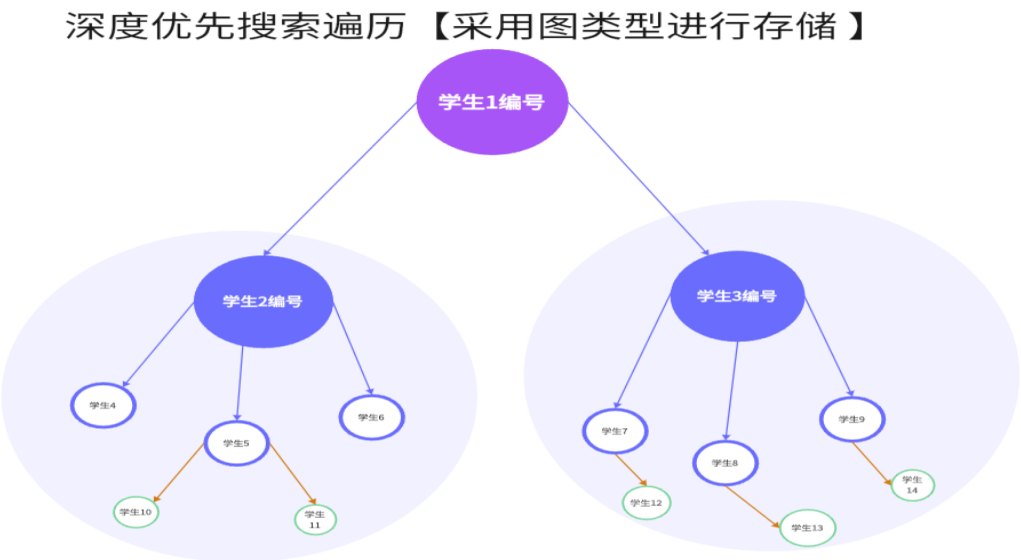
具体实现:

(1) 通过图表形式存储每个学生的编号信息并保存在文件当中,之后设置 visited 数组来记录表的访问情况，通过第一个结点对该表进行访问，每次访问一个结点并将该结点的学生编号与输入编号进行比较，若该编号存在则直接返回提示用户重新输入编号，否则系统正常运行。

图九:深度优先搜索遍历算法介绍



图十:基于本系统的深度优先搜索遍历展示



2、判断老师编号是否重复【使用广度优先搜索遍历】

具体实现:【所有老师信息存储在 teacher.txt 文件】

(1)从第一个职工编号出发，访问该顶点的所有邻接点职工 1，职工 2..职工 N

(2)对被访问后的职工设置为已经访问，同时与所输入的职工编号进行比较，若不同则继续进行广度优先搜索遍历

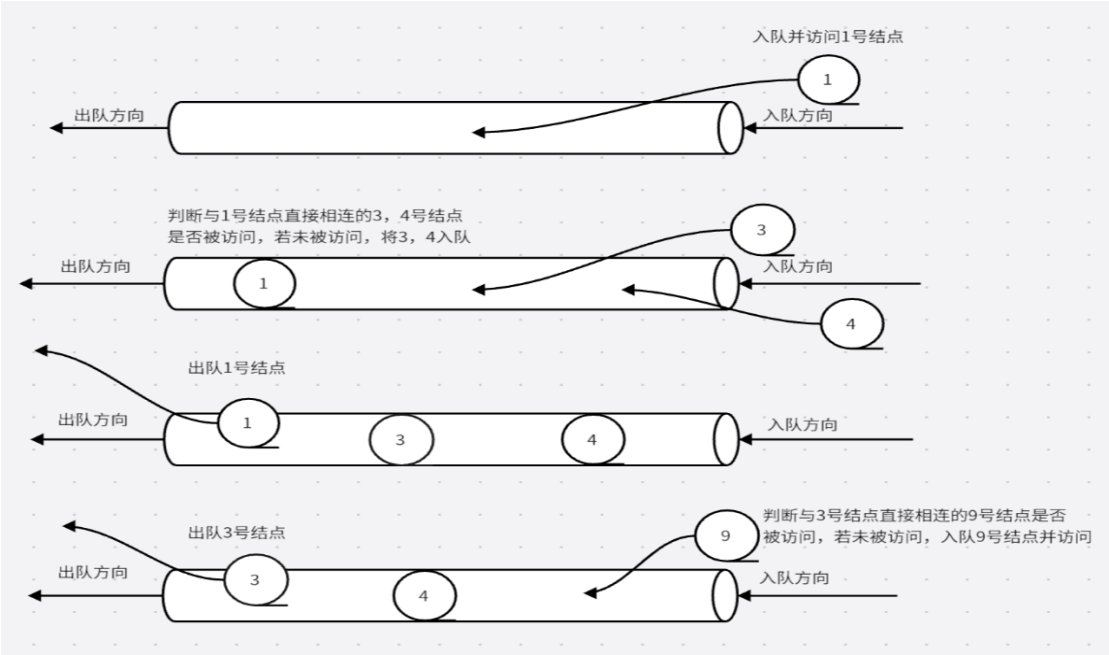
(3)继续从职工 1，职工 2...N 出发，再访问他们各自的所有邻接点

(4)若此时文件中还有顶点未被访问，则在外控算法的控制下，另选一个未曾被访问的顶点作为起始点。

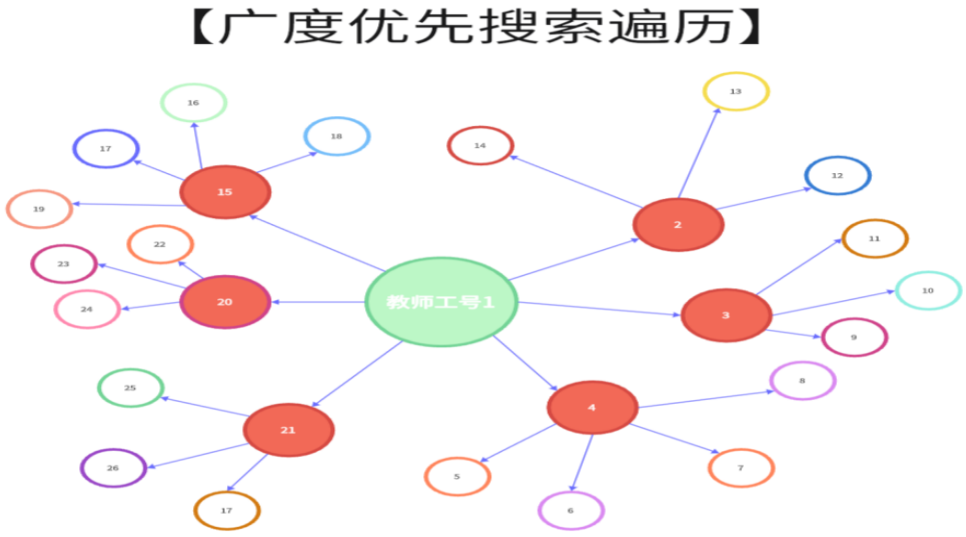
(5)重复上述步骤，直到所有的顶点都被访问过或直到所输入的职工编号被标记为重复则退出该算法让用户重新输入

(6)如果将文件中的所有职工号全部访问后仍没有找到相同的职工编号系统继续进行下步操作。

图十一:广度优先搜索遍历示意



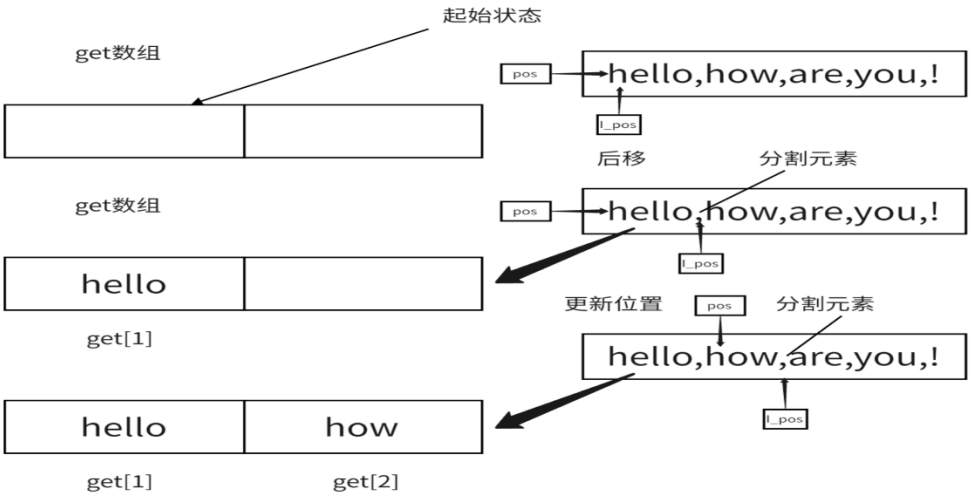
图十二:基于本系统教工编号的广度优先搜索遍历



3、字符串分割算法

当我们在截取预约文件中的信息进行输出时会使用到字符串的分割，对于这个算法，我们可以首先假设有一串字符 hello,how,are you,!我想通过逗号把这个连续的字符串分割成几个字符串，我们可以用两个标志好需要分割的字符的前后位置，pos 指向开始位置，I_pos 指向结束位置，以及一个存储分割后字符串的字符串数组 get[],首先初始化 pos，I_pos 都指向第一个位置，之后将 I_pos 不断向后移动，每移动一步后观察所指向的位置是否为分割的字符，如果不是我们传入的分割字符，继续向后移动，若该位置是我们所传入的需要被分割的字符，则将 pos 和 I_pos 之间的字符串分割存储，每次分割之后记录分割次数之后将 pos 指向 I_pos 也就是指向下一个需要分割的字符，最后重复上述步骤直到将所有的字符串都按需求进行分割。

图十三:分割算法示意图



4、使用了堆排序算法

当对学生和教师信息进行排序以序号编号进行排序，使用了堆排序算法。

堆排序思想:若在输出堆顶的最小值(或最大值)之后，使得剩余 n-1 个元素的序列重新又构成一个堆，则得到 n 个元素的次小值(或次大值)…如此反复，便能得到一个有序序列，这个过程称作堆排序。

小根堆的调整方法:

a. 输出堆顶元素，以堆中最后一个元素【完全二叉树中编号最大的元素】替代

b. 然后将根结点值与左、右子树的根结点进行比较，并与其中小者进行交换

c. 重复上述操作，直至叶子结点，将得到新的堆，称这个堆顶至叶子的调整过程为筛选.

图十四:小根堆示意图

3.5 使用手册

程序运行开始时可以通过键盘选择您的身份：1、学生 2、老师 3、管理员 或 0、退出机房预约系统。

选择身份完毕后需要您输入您的个人信息进行验证登录操作。

A.当您选择学生身份进入该系统时，系统会提示您输入学号信息，用户名称信息，以及您的登录密码，如果验证登录失败则会返回主菜单界面。若验证成功，将会进入学生类菜单界面。此时您可以选择:

1、申请预约:对学校中的不同机房进行预约的申请，您可以选择本周的周一、周二，…，周五这五天之内的早上或者下午进行使用。同时，您可以选择您预约的机房房间号，每个机房的容量不同，您可以根据您的个人需求进行选择。

2、查看本人预约信息:您可以通过该项操作查看您的预约时间段以及此刻您的预约状态，并根据预约状态进行下一步操作。

3、查看所有预约信息:通过该项操作，您可以了解所有人对机房的预约使用情况依次来选择您想要对哪个机房在哪个时间段进行使用。

4、取消预约:您可以通过该项操作来取消您对机房的预约。

5、注销登录:返回登录界面或退出程序。

B.当您选择教师身份进入该系统时，系统会提示您输入职工编号，用户名称信息，以及您的登录密码，如果验证登录失败则会返回主菜单界面。若验证成功，将会进入教师类菜单界面。此时您可以选择:

1、查看所有预约:通过该项操作，您可以查看到您所有学生对于机房使用的预约情况，以此来督促您的学生进行学习或合理分配学生使用机房。

2、审核预约:您可以通过该项操作对学生的预约进行审核，您可以通过此时机房的使用状态来通过或否决学生的预约申请。

3、注销登录:返回登录界面或退出程序。

C.当您选择管理员身份进入该系统时，系统会提示您输入用户名以及您的密码，如果验证登录失败则会返回主菜单界面。若验证成功，将会进入管理员类菜单界面。此时您可以选择:

1、添加账号:您可以通过该项操作向本系统中添加用户信息，同时可以通过该操作对学生或者教师的信息进行修改与审核。

2、查看账号:通过此项操作您可以查看本系统中除您以外所有用户的账号信息，其中账号信息包括:编号，用户名称，以及用户密码。

3、查看机房:您可以通过此选择查看本校的机房使用情况以及机房容量。

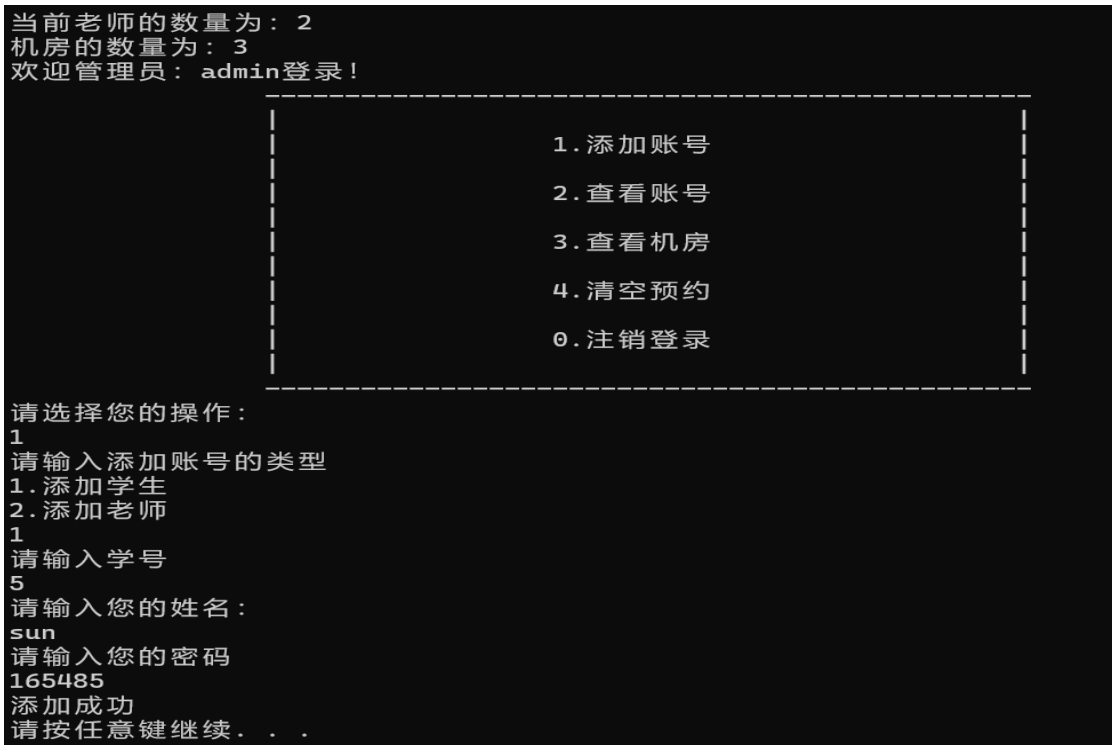
4、清空预约:若需要清空所有人的预约以对机房进行清理或维护，您可以通过此项操作完成。

5、注销登录:返回登录界面或退出程序。

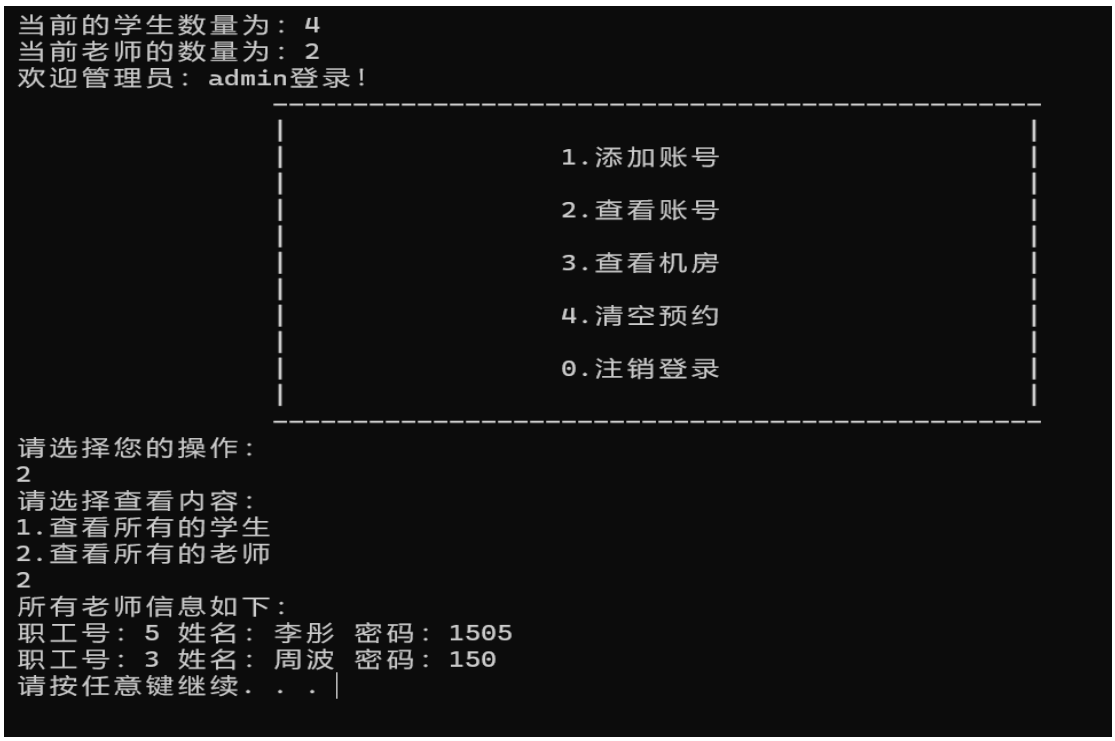
4 运行实例与系统功能测试

4.1 系统功能测试

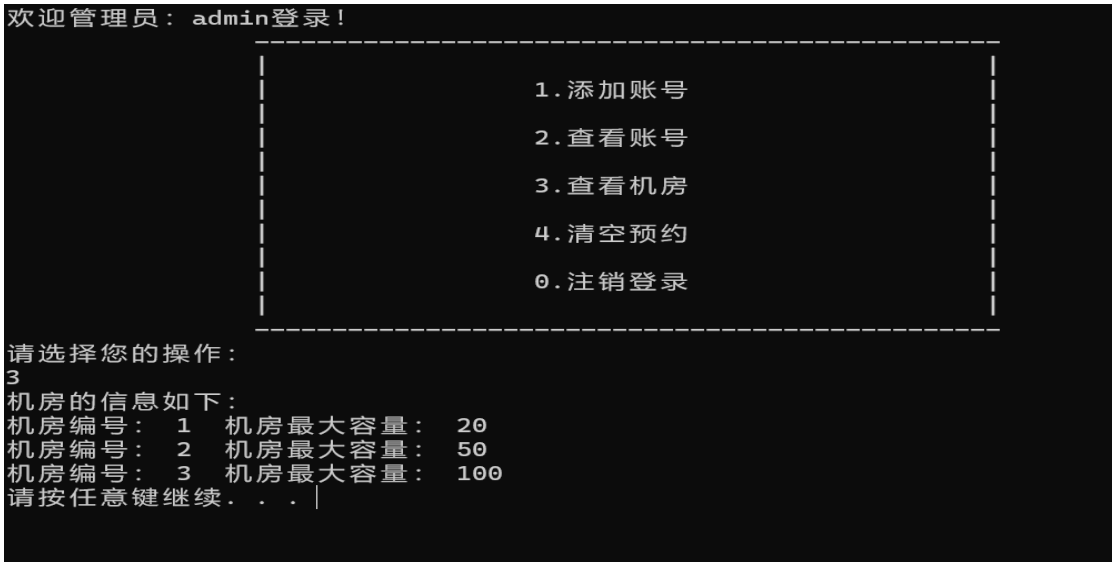
图十五:添加账号功能实现



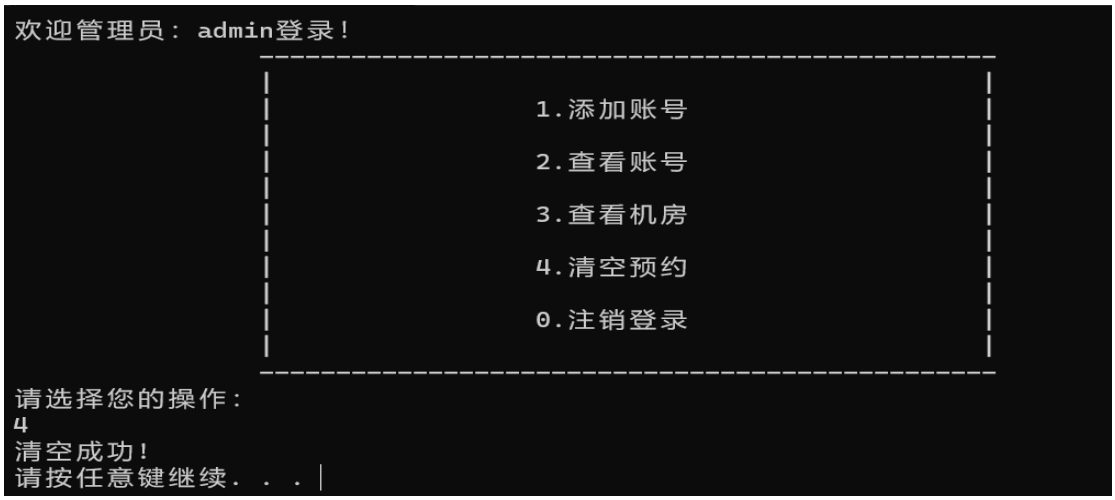
图十六:查看账号信息功能



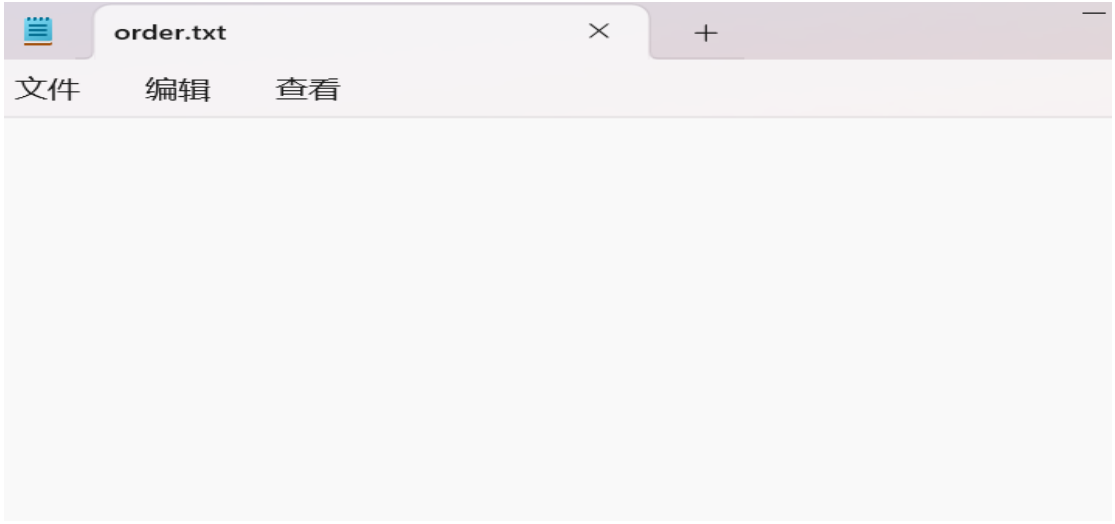
图十七:查看机房信息功能



图十八:清空所有预约信息

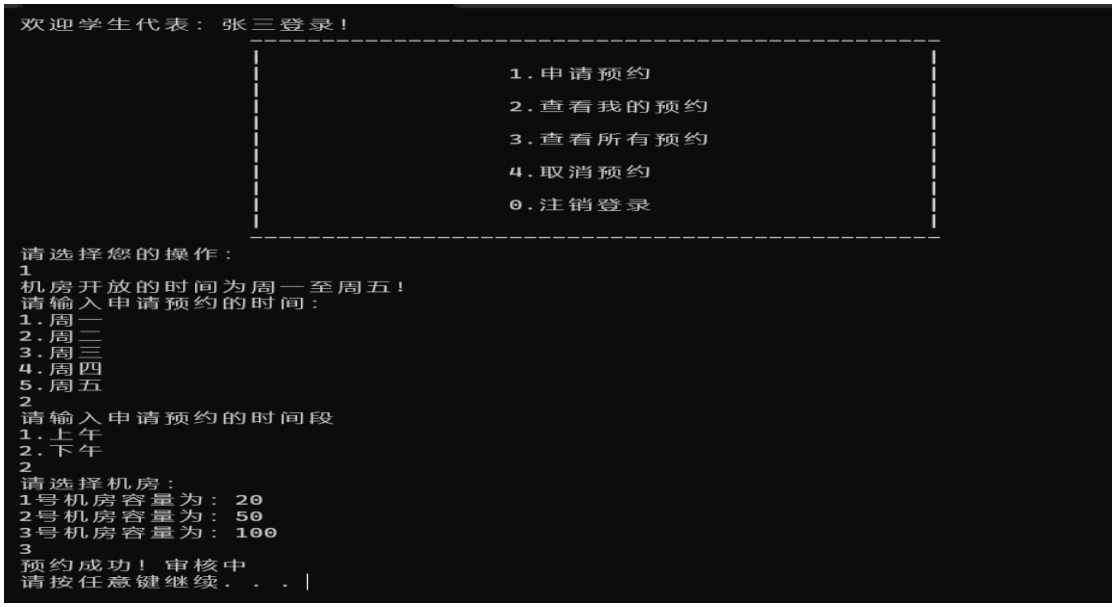


图十九:预约信息文件展示

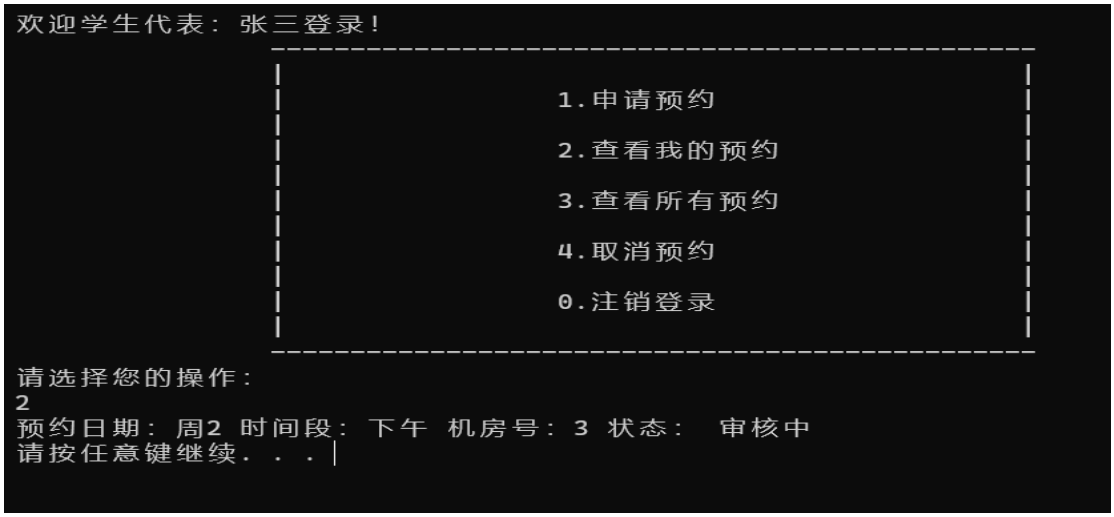


上图文件中信息为空，即预约信息已被全部清空

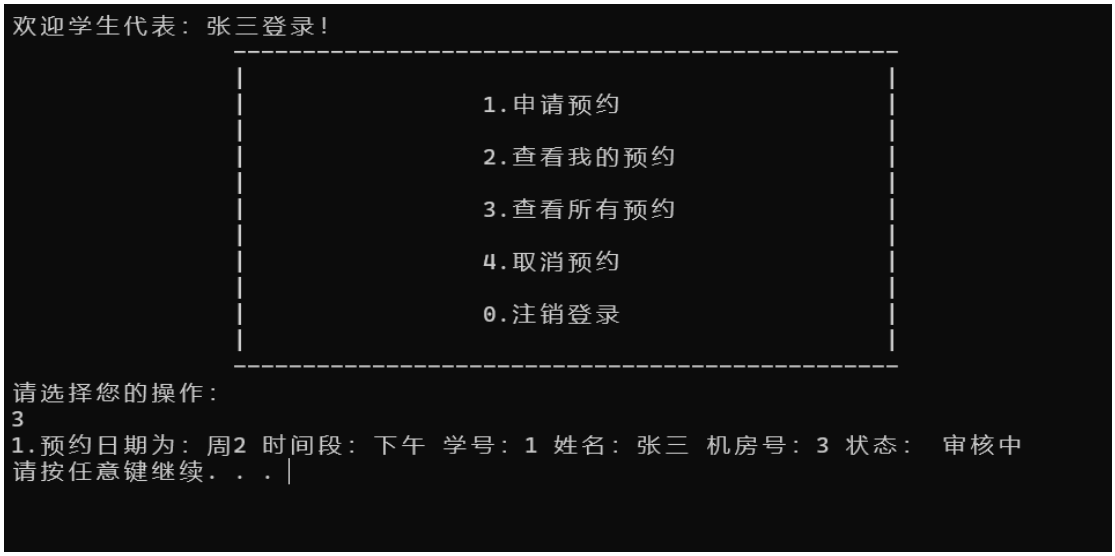
图二十:申请预约机房



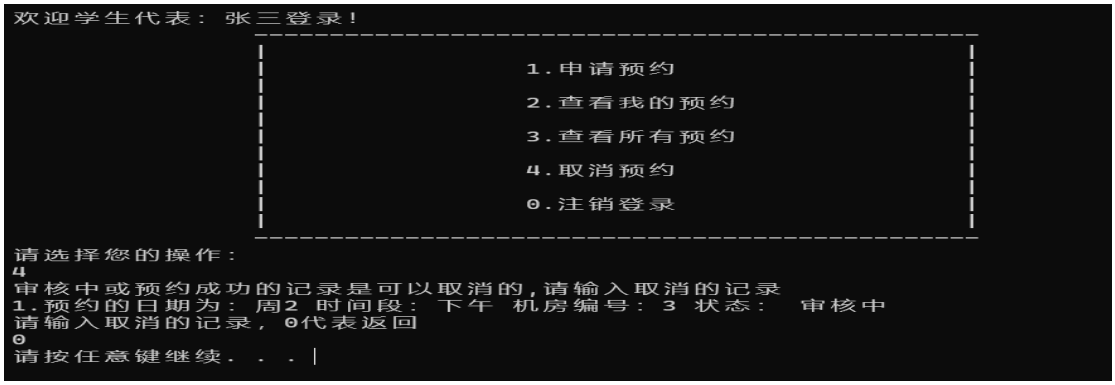
图二十一:查看本人的所有预约



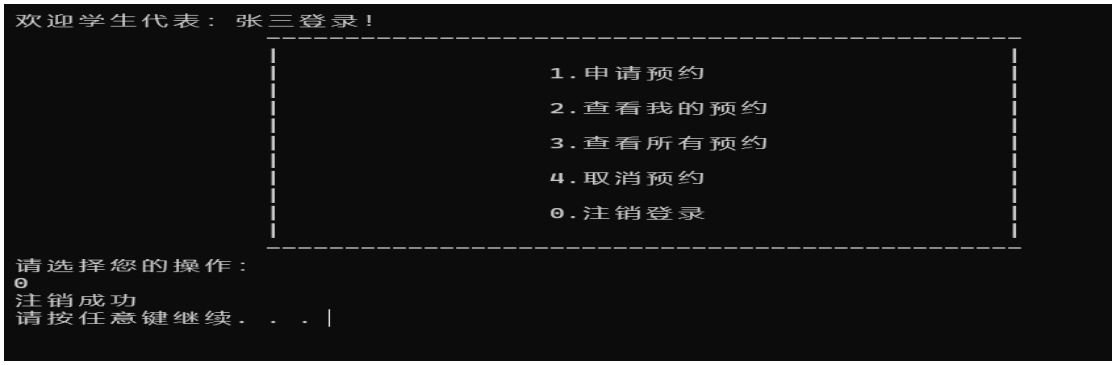
图二十二:查看所有学生的预约



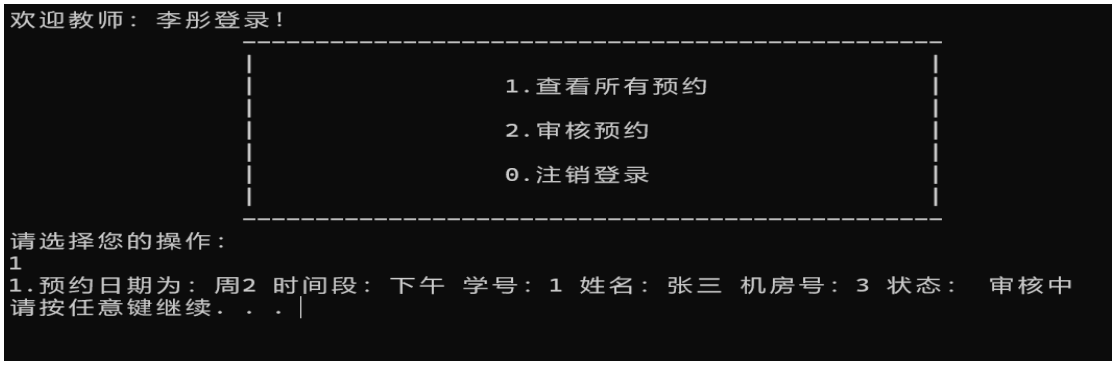
图二十三:取消本人的预约订单



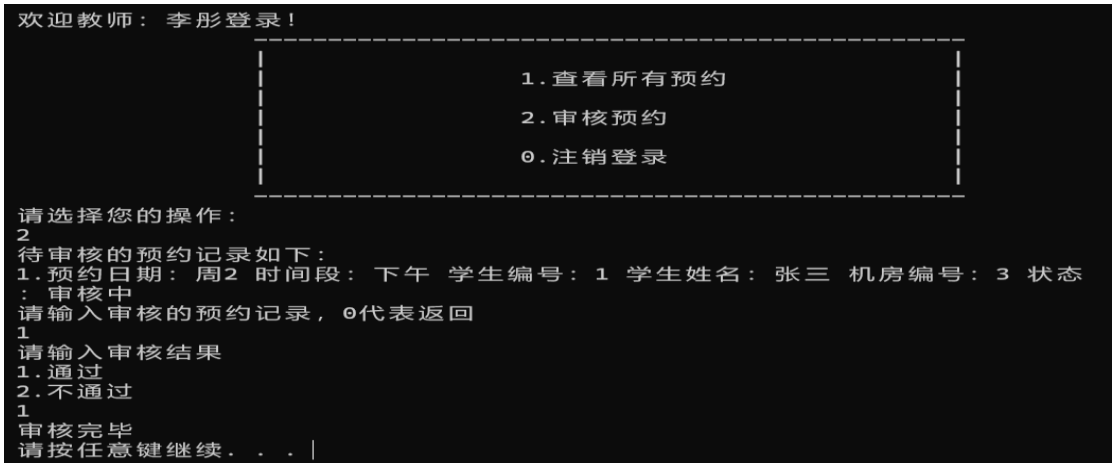
图二十四:注销登录



图二十五:查看所有预约情况

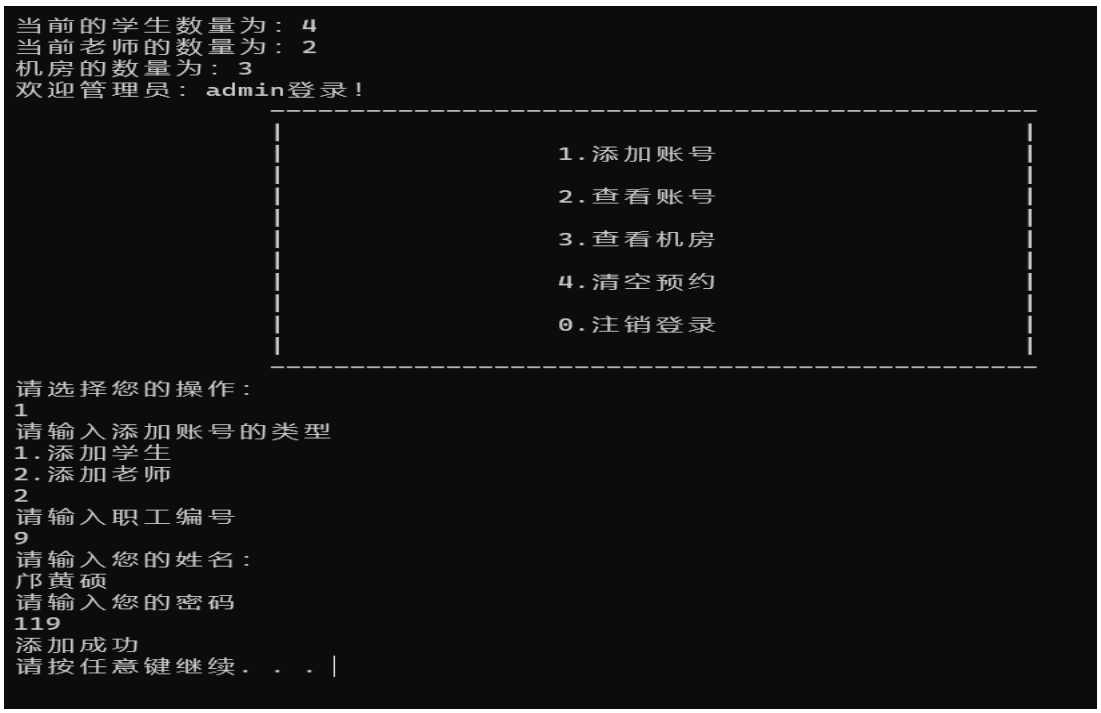


图二十六:审核预约

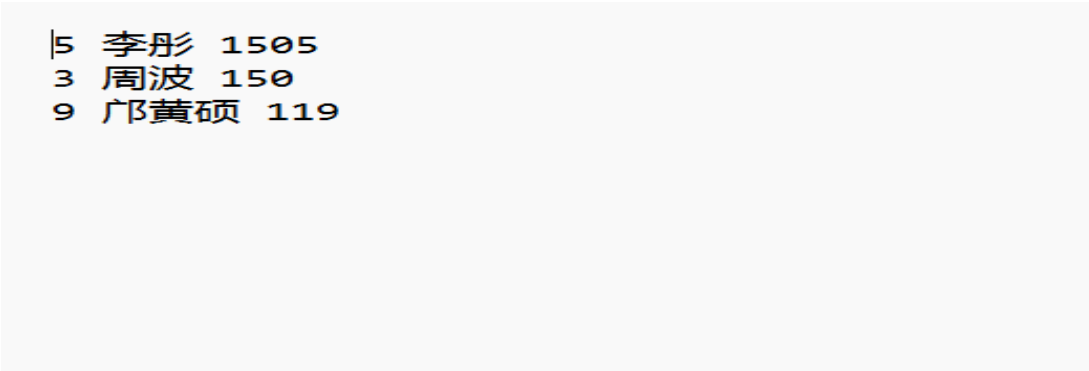


4.2 运行实例

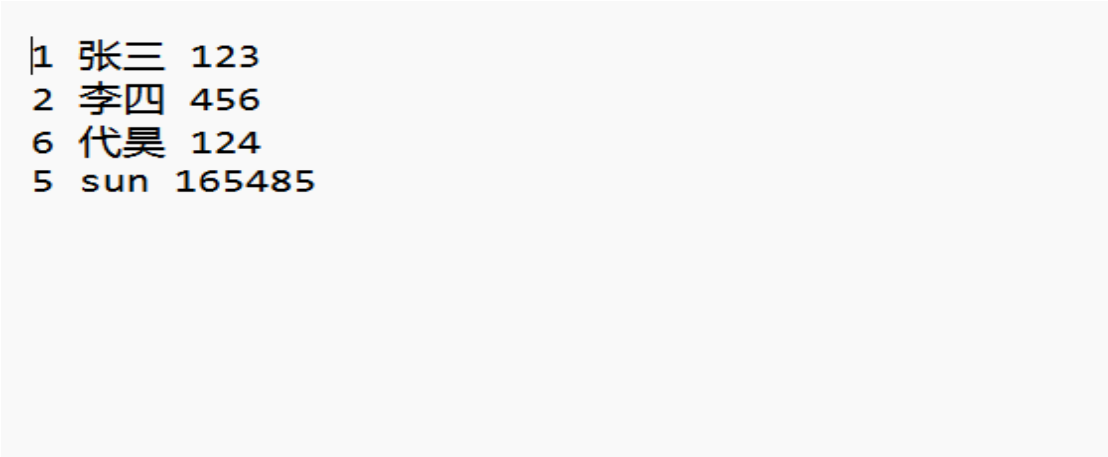
图二十七:添加账号



图二十八:教师信息存储文件



图二十九:学生信息存储文件



5 总结与进一步改进

5.1 总结（心得体会）

在本次数据结构的课程设计中，我选择了编写机房预约管理系统，因为我觉得这是一个既实用而又有挑战性的项目。 本次所写的机房管理系统的目的是为了方便机房管理员、教师和学生使用机房资源。系统需要实现以下功能：

- 管理员可以对机房、教师和学生的信息进行增删改查，以及清空和删除预约订单。
- 教师可以对学生的预约申请进行审核，以及查看所有预约信息和状态。
- 学生可以申请预约机房，选择预约日期和时段，以及查看自己的预约信息和状态，还可以取消预约。

为了实现这些功能，我首先设计了系统的类图，确定了系统的主要类和它们之间的关系。我使用了面向对象的思想，将系统分为三个层次：数据层、业务逻辑层和表示层。数据层负责存储和读取数据文件，业务逻辑层负责处理各种操作和逻辑判断，表示层负责与用户交互和显示界面。我使用了多态和继承的特性，将管理员、教师和学生抽象为一个基类 Identity，然后分别派生出 Admin、Teacher 和 Student 三个子类。这样可以减少代码的重复和冗余，也方便了后期的扩展和维护。我还使用了文件流来读写数据文件，使用了容器类来存储数据结构，使用了 STL 算法来简化操作。

在编写代码的过程中，我遇到了一些困难和问题。例如，如何保证数据文件的完整性和安全性？如何保证学生以及老师的编号是唯一的？如何将机房预约的信息进行更新同时进行保存？这些都是在编写程序中碰到的难题，但通过从网上搜集资料以及看书学习慢慢解决了这些问题，完成算法编写。 通过本次课程设计，我不仅巩固了数据结构的知识，也提高了我的编程能力和解决问题的能力。我感受到了编程的乐趣和挑战，也对自己有了更多的信心和动力。我还意识到了自己在编程方面还有很多不足之处，需要不断地学习和进步。我希望能够在今后的学习中继续努力，做出更好的作品。

同时在编写算法时我遇到了很多的问题，以下是在编写过程中遇到的问题以及解决方法。

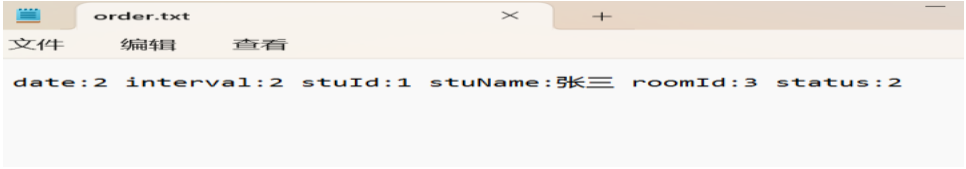
1、 添加账号

当通过管理员添加教师和学生身份之后，未能及时对文件中的信息进行更新，由于每次系统更新文件中的信息是在整个系统运行结束之后才能更新文件信息，因此为了解决这个 Bug 的出现，在每一次添加完学生或教师信息以后立刻调用 Init 函数【该函数的作用是对两个容器中的信息进行更新，同时在更新完信息之后重新打开文件进行数据的读取，在读取的同时将所有信息保存在容器之中】，通过这个函数的功能可以很好的解决这个 Bug 的出现。

2、 获取预约信息并更新

当我们通过该系统获取机房信息的时候，会发现无法正常的解析出文件中的信息，这是因为存储在文件中的信息是以对组的形式进行存储，不能简单的直接获取，如果使用 ifs 对象进行读取会导致显示信息不全同时信息错乱的情况

图三十:预约信息文件的存储显示



如上图所示文件的存储是按照对组方式存储，为了解决在这个 Bug,使用容器嵌套容器的方式来进行存储，具体存储方式如下。

图三十一:存储预约信息的容器

其中每个容器中 key 值表示预约记录的条数，而 value 值代表第 i 条记录中的所有信息即 map<string,string>。但当使用这种方式读取出机房信息时，所得到的结果带有“:”，因此需要截取:前后的信息进行打印输出，具体请见重要算法描述。

3、 更新预约记录

注意仅仅需要更新 status【状态】的数值，其余信息都无需进行更新，但是难点就是如何从文件中读取到 status 的数值并且在不更改其它信息的条件下对所有信息进行更新操作。为了解决这个 Bug，我们采用将原来文件中存储的信息全部删除，然后重新创建新的文件向文件中重新添加信息，虽然这种方法比较笨拙，但是使用这种方法可以保证信息的正确与完整性，同时若未来此系统可以升级更改机房信息或者预约时间信息后更方便操作，具体的更新机房预约信息的方法如下。

图三十二:更新预约信息代码实现

```
//更新预约记录
void OrderFile::updateOrder()
{
    if (this->m_Size == 0)
    {
        return;    //预约记录0条，直接return
    }

    ofstream ofs(ORDER_FILE, ios::out | ios::trunc);//删除并且重新创建新文件
    for (int i = 0; i < this->m_Size; i++)//通过m_Size得到所需要写进文件的个数之后重新向文件中写入数据
    {
        //通过第一个[i]找到需要访问的记录的条数，
        //此时获得map组中的value值，通过["date"]获取日期信息

        ofs << "date:" << this->m_OrderData[i]["date"] << " ";
        ofs << "interval:" << this->m_OrderData[i]["interval"] << " ";
        ofs << "stuId:" << this->m_OrderData[i]["stuId"] << " ";
        ofs << "stuName:" << this->m_OrderData[i]["stuName"] << " ";
        ofs << "roomId:" << this->m_OrderData[i]["roomId"] << " ";
        ofs << "status:" << this->m_OrderData[i]["status"] << endl;

    }
    ofs.close();
}
```

5.2 进一步改进及建议

在这次课设的编写过程中我学会了如何利用 STL 容器来存放数据与内容，但是由于编程能力有限目前无法实现预约的准确时间段即具体时间点，尝试此种编写时无法正确的输出预约情况的信息，在这个课设的基础上增加了防止学生学号，教职工职工号的重复操作，同时增加了对所有人的账号进行访问操作防止学生和职工忘

记密码导致无法登入系统。在未来可以根据自身需要继续向该系统中增加操作，比如如果教职工离职或者学生毕业删除该系统中的对应的信息。

感谢课程设计提供锻炼的机会，尽管时间紧张，但在课程设计的过程中收获了很多。课程设计中涉及到了很多课内没有接触过的算法和思想，有时理解起来会有些困难，如果在课程设计期间能安排答疑，可能会对课程设计的进展有一些帮助。同时若老师能随时跟进课设情况，在有 Bug 时候可以给予合适的帮助会帮助我们学生更好的完成课设。

6 附录——源程序

机房预约系统.cpp 文件

```
#include<iostream>
using namespace std;
#include"identity.h"
#include<fstream>
#include<string>
#include"globalFile.h"
#include"student.h"
#include"manager.h"
#include"teacher.h"

//进入学生子菜单界面
void studentMenu(Identity*& student)//使用父类的指针指向子类的对象，传入学生本体方便操作
{
    while (true)
    {
        //调用学生的子菜单
        student->operMenu();

        Student* stu = (Student*)student;//需要使用Student类型来使用这个指针对象

        int select = 0;
        cin >> select;        //接受用户的选择

        if (select == 1)      //申请预约
        {
            stu->applyOrder();
        }
        else if (select == 2)  //查看自身预约
        {
            stu->showMyOrder();
        }
        else if (select == 3)   //查看所有人预约
        {
```

```
            stu->showAllOrder();
        }
        else if (select == 4)    //取消预约
        {
            stu->cancelOrder();
        }
        else
        {
            //注销登录
            delete student;//删去对象
            cout << "注销成功" << endl;
            system("pause");
            system("cls");
            return;
        }
    }
}

//进入教师子菜单界面
void teacherMenu(Identity*& teacher)
{
    while (true)
    {
        //调用子菜单界面
        teacher->operMenu();

        Teacher* tea = (Teacher*)teacher;

        int select = 0;       //接受用户的选择

        cin >> select;

        if (select == 1)      //查看所有的预约
        {
            tea->showAllOrder();
        }
        else if (select == 2) //审核预约
        {
            tea->validOrder();
        }
        else
        {
            delete teacher;
            cout << "注销成功!" << endl;
```

```
            system("pause");
            system("cls");
            return;
        }
    }
}

//进入管理员子菜单界面
void managerMenu(Identity*& manager)
{
    while (true)
    {
        //调用管理员的子菜单
        manager->operMenu();

        //将父类的指针 转为子类指针，调用子类里的其他接口
        Manager* man = (Manager*)manager;

        int select = 0;
        //接受用户的选择
        cin >> select;

        if (select == 1)       //添加账号
        {
            //cout << "添加账号" << endl;
            man->addPerson();
        }
        else if (select == 2)  //查看账号
        {
            //cout << "查看账号" << endl;
            man->showPerson();
        }
        else if (select == 3)  //查看机房
        {
            //cout << "查看机房" << endl;
            man->showComputer();
        }
        else if (select == 4)   //清空预约
        {
            //cout << "清空预约" << endl;
            man->cleanFile();
        }
        else
        {
```

```
            //注销
            delete manager;     //销毁堆区对象
            cout << "注销成功" << endl;
            system("pause");
            system("cls");
            return;
        }


    }
}



//登录的功能    //参数1 操作文件名  参数2  操作身份类型
void LoginIn(string fileName, int type)
{
    //父类指针 用于指向子类对象
    Identity* person = NULL;


    //读文件
    ifstream ifs;
    ifs.open(fileName, ios::in);
    //判断文件是否存在
    if (!ifs.is_open())
    {
        cout << "文件不存在" << endl;
        ifs.close();
        return;
    }


    //准备接受用户的信息
    int id;
    string name;
    string pwd;

    if (type == 1)   //学生身份
    {
        cout << "输入您的学号：" << endl;
        cin >> id;
    }
    else if (type == 2)   //老师身份
```

```
{
    cout << "请输入您的职工号：" << endl;
    cin >> id;
}

cout << "请输入用户名：" << endl;
cin >> name;

cout << "请输入密码：" << endl;
cin >> pwd;

if (type == 1)
{
    //学生身份验证
    int fId;        //从文件中读取的id号
    string fName;   //从文件中获取的姓名
    string fPwd;    //从文件中获取的密码
    while (ifs >> fId && ifs >> fName && ifs >> fPwd)
    {
        //与用户输入的信息做对比
        if (fId == id && fName == name && fPwd == pwd)
        {
            cout << "学生验证登录成功" << endl;
            system("pause");
            system("cls");

            person = new Student(id, name, pwd);

            //进入学生身份的子菜单
            studentMenu(person);//只有当身份验证成功以后才可以执行下一步操作
            return;

        }
    }


}
else if (type == 2)
{
    //教师身份验证
    int fId;        //从文件中读取的id号
    string fName;   //从文件中获取的姓名
    string fPwd;    //从文件中获取的密码
    while (ifs >> fId && ifs >> fName && ifs >> fPwd)
```

```
    {
        //与用户输入的信息做对比
        if (fId == id && fName == name && fPwd == pwd)
        {
            cout << "教师验证登录成功" << endl;
            system("pause");
            system("cls");

            person = new Teacher(id, name, pwd);

            //进入教师身份的子菜单
            teacherMenu(person);
            return;

        }
    }


}
else if (type == 3)
{
    //管理员身份验证
    string fName;   //从文件中获取的姓名
    string fPwd;    //从文件中获取的密码

    while (ifs >> fName && ifs >> fPwd)
    {
        //与用户输入的信息做对比
        if (fName == name && fPwd == pwd)
        {
            cout << "管理员验证登录成功" << endl;
            system("pause");
            system("cls");

            person = new Manager(name, pwd);

            //进入管理员身份的子菜单
            managerMenu(person);
            return;

        }
    }
}
```

```
    cout << "验证登录失败!" << endl;
    system("pause");
    system("cls");

}


int main()
{
    int select = 0;   //用于接收用户的选择


    while (true)
    {

        cout << "==============================  欢迎来到合肥工业大学机房预约系统
============================" << endl;
        cout << endl << "请输入您的身份" << endl;
        cout << "                    " << "\t\t ------------------------------------
\n";
        cout << "                    " << "\t\t|
|\n";
        cout << "                    " << "\t\t|               1、学生
|\n";
        cout << "                    " << "\t\t|
|\n";
        cout << "                    " << "\t\t|               2、老师
|\n";
        cout << "                    " << "\t\t|
|\n";
        cout << "                    " << "\t\t|               3、管理员
|\n";
        cout << "                    " << "\t\t|
|\n";
        cout << "                    " << "\t\t|               0、退出
|\n";
        cout << "                    " << "\t\t|
|\n";
        cout << "                    " << "\t\t ------------------------------------
\n";

        cout << "请输入你的选择: ";

        cin >> select;   //接受用户选择
```

```
        switch (select)
        {
        case 1:         //学生身份
            LoginIn(STUDENT_FILE, 1);
            break;
        case 2:         //老师身份
            LoginIn(TEACHER_FILE, 2);
            break;
        case 3:         //管理员系统
            LoginIn(ADMIN_FILE, 3);
            break;
        case 0:         //退出系统
            cout << "欢迎您下次使用" << endl;
            system("pause");
            return 0;
            break;
        default:
            cout << "输入有误，请重新选择" << endl;
            system("pause");
            system("cls");
            break;
        }

    }

    system("pause");
    return 0;
    }
```

Student.h 文件

```
#pragma once
#include<iostream>
using namespace std;
#include"identity.h"
#include<string>
#include<vector>
#include"computerRoom.h"
#include<fstream>
#include"globalFile.h"
#include"orderFile.h"
//设计学生类
class Student :public Identity
{
```

```
public:

    //默认构造
    Student();

    //有参构造   参数：学号 ，姓名，密码
    Student(int id, string name, string pwd);

    //菜单界面
    virtual void operMenu();

    //申请预约
    void applyOrder();

    //查看自身预约
    void showMyOrder();

    //查看所有预约
    void showAllOrder();

    //取消预约
    void cancelOrder();

    //学生学号
    int m_Id;

    //机房信息容器
    vector<ComputerRoom>vCom;//存放机房信息的容器
    };
```

Student.cpp 文件

```
#include"student.h"


//默认构造
Student::Student()
{
}

//有参构造   参数：学号 ，姓名，密码
Student::Student(int id, string name, string pwd)
{
    //初始化属性
    this->m_Id = id;
```

```
    this->m_Name = name;
    this->m_Pwd = pwd;

    //初始化机房信息
    ifstream ifs;
    ifs.open(COMPUTER_FILE, ios::in);//打开机房信息的存储容器

    //创建机房对象用来存放机房信息
    ComputerRoom com;
    while (ifs >> com.m_ComId && ifs >> com.m_MaxNum)
    {
        //将读取的信息放入到容器中
        vCom.push_back(com);
    }
    ifs.close();
}

//菜单界面
void Student::operMenu()
{
    cout << "欢迎学生：" << this->m_Name << "登录！ " << endl;
    cout << "\t\t-------------------------------------------\n";
    cout << "\t\t|                                         |\n";
    cout << "\t\t|                1.申请预约               |\n";
    cout << "\t\t|                                         |\n";
    cout << "\t\t|                2.查看我的预约           |\n";
    cout << "\t\t|                                         |\n";
    cout << "\t\t|                3.查看所有预约           |\n";
    cout << "\t\t|                                         |\n";
    cout << "\t\t|                4.取消预约               |\n";
    cout << "\t\t|                                         |\n";
    cout << "\t\t|                0.注销登录               |\n";
    cout << "\t\t|                                         |\n";
    cout << "\t\t-------------------------------------------\n";
    cout << "请选择您的操作：" << endl;
}


//申请预约
void Student::applyOrder()
{
    cout << "机房开放的时间为周一至周五!" << endl;
    cout << "请输入申请预约的时间：" << endl;
    cout << "1.周一" << endl;
```

```
cout << "2.周二" << endl;
cout << "3.周三" << endl;
cout << "4.周四" << endl;
cout << "5.周五" << endl;

int date = 0;          //日期
int interval = 0;      //时间段
int room = 0;          //机房编号

while (true)
{
    cin >> date;
    if (date >= 1 && date <= 5)//规定时间范围
    {
        break;
    }
    cout << "输入有误，请重新输入" << endl;
}

cout << "请输入申请预约的时间段" << endl;
cout << "1.上午" << endl;
cout << "2.下午" << endl;
while (true)
{
    cin >> interval;
    if (interval >= 1 && interval <= 2)
    {
        break;
    }
    cout << "输入有误，请重新输入" << endl;
}

cout << "请选择机房：" << endl;
for (int i = 0; i < vCom.size(); i++)//将所有机房信息进行输出
{
    cout << vCom[i].m_ComId << "号机房容量为：" << vCom[i].m_MaxNum << endl;
}

while (true)
{
    cin >> room;
    if (room >= 1 && room <= 3)
    {
        break;
```

```
        }
        cout << "输入有误，请重新输入" << endl;
    }

    cout << "预约成功！审核中" << endl;

    ofstream ofs;//向预约的文件中对信息进行添加操作
    ofs.open(ORDER_FILE, ios::app);//追加信息

    ofs << "date:" << date << " ";
    ofs << "interval:" << interval << " ";//时间段
    ofs << "stuId:" << this->m_Id << " ";
    ofs << "stuName:" << this->m_Name << " ";
    ofs << "roomId:" << room << " ";
    ofs << "status:" << 1 << endl;          //状态 1表示审核中

    ofs.close();

    system("pause");
    system("cls");

}

//查看自身预约
void Student::showMyOrder()
{
    OrderFile of;

    if (of.m_Size == 0)//判断文件中是否存在记录
    {
        cout << "无预约记录!" << endl;
        system("pause");
        system("cls");
        return;
    }
    for (int i = 0; i < of.m_Size; i++)
    {
        //string 转 int
        //string 利用  .c_str()转 const char *
        //利用 atoi (const char*)转int
        if (this->m_Id == atoi(of.m_OrderData[i]["stuId"].c_str())) //找到自身的预约,//
由于自身的学号信息是字符串类型，利用atoi将信息转换成int类型
        {
            cout << "预约日期：周" << of.m_OrderData[i]["date"];
```

```
            cout << " 时间段：" << (of.m_OrderData[i]["interval"] == "1" ? "上午" : "下
午");
            cout << " 机房号：" << of.m_OrderData[i]["roomId"];

            string status = " 状态：";
            //1.审核中    2.已预约    -1.预约失败  0.取消预约
            if (of.m_OrderData[i]["status"] == "1")
            {
                status += " 审核中";
            }
            else if (of.m_OrderData[i]["status"] == "2")
            {
                status += " 预约成功";
            }
            else if (of.m_OrderData[i]["status"] == "-1")
            {
                status += " 预约失败，审核未通过";
            }
            else
            {
                status += " 预约已取消";
            }
            cout << status << endl;

        }
    }
    system("pause");
    system("cls");

}

//查看所有预约
void Student::showAllOrder()
{
    OrderFile of;
    if (of.m_Size == 0)
    {
        cout << "无预约记录" << endl;
        system("pause");
        system("cls");
        return;
    }

    for (int i = 0; i < of.m_Size; i++)
```

```
    {
        cout << i + 1 << ".";
        cout << "预约日期为：周" << of.m_OrderData[i]["date"];
        cout << " 时间段：" << (of.m_OrderData[i]["interval"] == "1" ? "上午" : "下午");
        cout << " 学号：" << of.m_OrderData[i]["stuId"];
        cout << " 姓名：" << of.m_OrderData[i]["stuName"];
        cout << " 机房号：" << of.m_OrderData[i]["roomId"];

        string status = " 状态：";
        //1.审核中    2.已预约    -1.预约失败  0.取消预约
        if (of.m_OrderData[i]["status"] == "1")
        {
            status += " 审核中";
        }
        else if (of.m_OrderData[i]["status"] == "2")
        {
            status += " 预约成功";
        }
        else if (of.m_OrderData[i]["status"] == "-1")
        {
            status += " 预约失败，审核未通过";
        }
        else
        {
            status += " 预约已取消";
        }
        cout << status << endl;
    }
    system("pause");
    system("cls");
}

//取消预约
void Student::cancelOrder()
{
    OrderFile of;
    if (of.m_Size == 0)
    {
        cout << "无预约记录!" << endl;
        system("pause");
        system("cls");
        return;
    }
```

```
    cout << "审核中或预约成功的记录是可以取消的,请输入取消的记录" << endl;
    vector<int>v;      //存放在最大容器中的下标编号
    int index = 1;
    for (int i = 0; i < of.m_Size; i++)
    {
        //先判断是自身的学号
        if (this->m_Id == atoi(of.m_OrderData[i]["stuId"].c_str()))
        {
            //再筛选状态  审核中或预约成功
            if (of.m_OrderData[i]["status"] == "1" || of.m_OrderData[i]["status"] == "2")
            {
                v.push_back(i);
                cout << index++ << ".";
                cout << "预约的日期为：周" << of.m_OrderData[i]["date"];//date中存储的是
汉字字符
                cout << " 时间段：" << (of.m_OrderData[i]["interval"] == "1" ? "上午" : "
下午");//不是1就输出下午
                cout << " 机房编号：" << of.m_OrderData[i]["roomId"];
                string status = " 状态：";
                if (of.m_OrderData[i]["status"] == "1")
                {
                    status += " 审核中";//使用字符串的拼接操作
                }
                else if (of.m_OrderData[i]["status"] == "2")
                {
                    status += " 预约成功";
                }

                cout << status << endl;

            }
        }
    }

    cout << "请输入取消的记录，0代表返回" << endl;
    int  select = 0;

    while (true)
    {
        cin >> select;
        if (select >= 0 && select <= v.size())
        {
            if (select == 0)
            {
```

```
                break;
            }
            else
            {
                of.m_OrderData[v[select - 1]]["status"] = "0";

                of.updateOrder();
                cout << "已取消预约" << endl;
                break;
            }
        }
        else
        {
            cout << "输入有误，请重新输入" << endl;
        }
    }
    system("pause");
    system("cls");
    }
```

teacher.h 文件

```
#pragma once
#include<iostream>
using namespace std;
#include"identity.h"
#include<string>
#include"orderFile.h"
#include<vector>
//教师类设计
class Teacher :public Identity
{
public:
    //默认构造
    Teacher();

    //有参构造
    Teacher(int empId, string name, string pwd);

    //菜单界面
    virtual void operMenu();

    //查看所有预约
    void showAllOrder();
```

```
    //审核所有预约
    void validOrder();

    //职工号
    int m_EmpId;

    };
```

teacher.cpp 文件

```
#include"teacher.h"

//默认构造
Teacher::Teacher()
{
}

//有参构造
Teacher::Teacher(int empId, string name, string pwd)
{
    //初始化属性
    this->m_EmpId = empId;
    this->m_Name = name;
    this->m_Pwd = pwd;
}

//菜单界面
void Teacher::operMenu()
{
    cout << "欢迎教师：" << this->m_Name << "登录！ " << endl;
    cout << "\t\t-----------------------------------------\n";
    cout << "\t\t|                                       |\n";
    cout << "\t\t|                1.查看所有预约         |\n";
    cout << "\t\t|                                       |\n";
    cout << "\t\t|                2.审核预约             |\n";
    cout << "\t\t|                                       |\n";
    cout << "\t\t|                0.注销登录             |\n";
    cout << "\t\t|                                       |\n";
    cout << "\t\t-----------------------------------------\n";
    cout << "请选择您的操作：" << endl;
}

//查看所有预约
void Teacher::showAllOrder()
{
```

```
    OrderFile of;
    if (of.m_Size == 0)
    {
        cout << "无预约记录" << endl;
        system("pause");
        system("cls");
        return;
    }

    for (int i = 0; i < of.m_Size; i++)
    {
        cout << i + 1 << ".";
        cout << "预约日期为：周" << of.m_OrderData[i]["date"];
        cout << " 时间段：" << (of.m_OrderData[i]["interval"] == "1" ? "上午" : "下午");
        cout << " 学号：" << of.m_OrderData[i]["stuId"];
        cout << " 姓名：" << of.m_OrderData[i]["stuName"];
        cout << " 机房号：" << of.m_OrderData[i]["roomId"];

        string status = " 状态：";
        //1.审核中    2.已预约    -1.预约失败  0.取消预约
        if (of.m_OrderData[i]["status"] == "1")
        {
            status += " 审核中";
        }
        else if (of.m_OrderData[i]["status"] == "2")
        {
            status += " 预约成功";
        }
        else if (of.m_OrderData[i]["status"] == "-1")
        {
            status += " 预约失败，审核未通过";
        }
        else
        {
            status += " 预约已取消";
        }
        cout << status << endl;
    }
    system("pause");
    system("cls");
}

//审核所有预约
void Teacher::validOrder()
```

```
{
    OrderFile of;
    if (of.m_Size == 0)
    {
        cout << "无预约记录" << endl;
        system("pause");
        system("cls");
        return;
    }

    vector<int>v;
    int index = 0;
    cout << "待审核的预约记录如下：" << endl;

    for (int i = 0; i < of.m_Size; i++)
    {
        if (of.m_OrderData[i]["status"] == "1")
        {
            v.push_back(i);
            cout << ++index << ".";
            cout << "预约日期：周" << of.m_OrderData[i]["date"];
            cout << " 时间段：" << (of.m_OrderData[i]["interval"] == "1" ? "上午" : "下
午");
            cout << " 学生编号：" << of.m_OrderData[i]["stuId"];
            cout << " 学生姓名：" << of.m_OrderData[i]["stuName"];
            cout << " 机房编号：" << of.m_OrderData[i]["roomId"];
            string status = " 状态：审核中 ";
            cout << status << endl;
        }
    }

    cout << "请输入审核的预约记录，0代表返回" << endl;
    int select = 0;     //接受用户选择的预约记录
    int ret = 0;        //接受预约结果记录

    while (true)
    {
        cin >> select;
        if (select >= 0 && select <= v.size())
        {
            if (select == 0)
            {
                break;
            }
```

```
            else
            {
                cout << "请输入审核结果" << endl;
                cout << "1.通过" << endl;
                cout << "2.不通过" << endl;
                cin >> ret;

                if (ret == 1)
                {
                    //通过情况
                    of.m_OrderData[v[select - 1]]["status"] = "2";

                }
                else
                {
                    //不通过情况
                    of.m_OrderData[v[select - 1]]["status"] = "-1";
                }
                of.updateOrder();
                cout << "审核完毕" << endl;
                break;
            }
        }
        cout << "输入有误，请重新输入" << endl;
    }

    system("pause");
    system("cls");
    }
```

manager.h 文件

```
#pragma once
#include<iostream>
using namespace std;
#include"identity.h"
#include<string>
#include<fstream>
#include"globalFile.h"
#include<vector>
#include"student.h"
#include"teacher.h"
#include<algorithm>
#include"computerRoom.h"
```

```
//管理员类设计
class Manager :public Identity
{
public:
    //默认构造
    Manager();

    //有参构造
    Manager(string name, string pwd);

    //菜单界面
    virtual void operMenu();

    //添加账号
    void addPerson();

    //查看账号
    void showPerson();

    //查看机房信息
    void showComputer();

    //清空所有的预约记录
    void cleanFile();

    //初始化容器
    void initVector();

    //检测重复    参数1 检测学号/职工号    参数2 检测类型
    bool checkRepeat(int id, int type);

    //存放学生容器
    vector<Student>vStu;

    //存放教师容器
    vector<Teacher>vTea;

    //存放机房信息容器
    vector<ComputerRoom>vCom;

    };
```

manager.cpp 文件

```
#include"manager.h"
```

```
//默认构造
Manager::Manager()
{
}

//有参构造
Manager::Manager(string name, string pwd)
{
    //初始化管理员的信息
    this->m_Name = name;
    this->m_Pwd = pwd;

    //初始化容器，获取到所有文件中学生和老师的信息
    this->initVector();

    //初始化机房信息
    ifstream ifs;
    ifs.open(COMPUTER_FILE, ios::in);//读文件

    ComputerRoom com;//创建Com对象
    while (ifs >> com.m_ComId && ifs >> com.m_MaxNum)
    {
        vCom.push_back(com);
    }
    ifs.close();

    cout << "机房的数量为：" << vCom.size() << endl;
}


//菜单界面
void Manager::operMenu()
{
    cout << "欢迎管理员：" << this->m_Name << "登录！ " << endl;
    cout << "\t\t-------------------------------------------\n";
    cout << "\t\t|                                         |\n";
    cout << "\t\t|                1.添加账号               |\n";
    cout << "\t\t|                                         |\n";
    cout << "\t\t|                2.查看账号               |\n";
    cout << "\t\t|                                         |\n";
    cout << "\t\t|                3.查看机房               |\n";
    cout << "\t\t|                                         |\n";
    cout << "\t\t|                4.清空预约               |\n";
```

```
    cout << "\t\t|                                                |\n";
    cout << "\t\t|                 0.注销登录                     |\n";
    cout << "\t\t|                                                |\n";
    cout << "\t\t--------------------------------------------------\n";
    cout << "请选择您的操作：" << endl;

}

//添加账号
void Manager::addPerson()
{
    cout << "请输入添加账号的类型" << endl;
    cout << "1.添加学生" << endl;
    cout << "2.添加老师" << endl;

    string fileName;   //操作文件名
    string tip;        //提示id号
    string errorTip;   //错误重复提示

    ofstream ofs;      //文件操作对象，写文件

    int select = 0;
    cin >> select;     //接受用户的选择

    if (select == 1)
    {
        //添加的是学生
        fileName = STUDENT_FILE;
        tip = "请输入学号";
        errorTip = "学号重复，请重新输入";

    }
    else
    {
        fileName = TEACHER_FILE;
        tip = "请输入职工编号";
        errorTip = "职工号重复，请重新输入";

    }

    //利用追加的方式写文件
    ofs.open(fileName, ios::out | ios::app);//向文件中追加信息

    int id;        //学号或者职工号
```

```
    string name;   //姓名
    string pwd;    //密码

    cout << tip << endl;

    while (true)
    {
        cin >> id;
        //首先输入一个id号然后判断是否有重复
        bool ret = checkRepeat(id, select);//select中存放的就是传入的信息类型
        if (ret)  //有重复
        {
            cout << errorTip << endl;
        }
        else
        {
            break;//没有重复则继续向下执行
        }
    }

    cout << "请输入您的姓名：" << endl;
    cin >> name;
    cout << "请输入您的密码" << endl;
    cin >> pwd;

    //向文件中添加数据
    ofs << id << " " << name << " " << pwd << " " << endl;
    cout << "添加成功" << endl;

    system("pause");
    system("cls");

    ofs.close();

    //调用初始化容器的接口，重新获取接口中的数据
    this->initVector();
}

void printStudent(Student& s)
{
    cout << "学号：" << s.m_Id << " 姓名：" << s.m_Name << " 密码：" << s.m_Pwd << endl;

}
```

```
void printTeacher(Teacher& t)
{
    cout <<"职工号：" << t.m_EmpId <<" 姓名：" << t.m_Name <<" 密码：" << t.m_Pwd << endl;
}


//查看账号
void Manager::showPerson()
{
    cout << "请选择查看内容：" << endl;
    cout << "1.查看所有的学生" << endl;
    cout << "2.查看所有的老师" << endl;

    int select = 0;     //接受用户的选择
    cin >> select;

    if (select == 1)
    {
        //查看学生
        cout << "所有学生信息如下：" << endl;
        for_each(vStu.begin(), vStu.end(), printStudent);//使用Vector容器中的算法进行遍
历操作
    }
    else
    {
        //查看学生
        cout << "所有老师信息如下：" << endl;
        for_each(vTea.begin(), vTea.end(), printTeacher);
    }

    system("pause");
    system("cls");

}

//查看机房信息
void Manager::showComputer()
{
    cout << "机房的信息如下：" << endl;
    for (vector<ComputerRoom>::iterator it = vCom.begin(); it != vCom.end(); it++)
    {
        cout <<"机房编号：  " << it->m_ComId <<"  机房最大容量：  " << it->m_MaxNum << endl;
    }
    system("pause");
```

```
    system("cls");
}


//清空所有的预约记录
void Manager::cleanFile()
{
    ofstream ofs(ORDER_FILE, ios::trunc);//如果文件存在则重新创建
    ofs.close();

    cout << "清空成功! " << endl;
    system("pause");
    system("cls");
}

//初始化容器
void Manager::initVector()
{
    //确保两个容器全为空才进行初始化操作
    //清空操作
    vStu.clear();
    vTea.clear();
    //读取信息  学生
    ifstream ifs;
    ifs.open(STUDENT_FILE, ios::in);
    if (!ifs.is_open())
    {
        cout << "文件打开失败" << endl;
        return;
    }

    Student s;
    while (ifs >> s.m_Id && ifs >> s.m_Name && ifs >> s.m_Pwd)
    {
        vStu.push_back(s);//全部放入容器中
    }

    cout << "当前的学生数量为：" << vStu.size() << endl;//vector容器中统计学生的个数函数
    ifs.close();

    //读取信息  老师
    ifs.open(TEACHER_FILE, ios::in);
    if (!ifs.is_open())
    {
```

```
        cout << "文件打开失败" << endl;
    }

    Teacher t;
    while (ifs >> t.m_EmpId && ifs >> t.m_Name && ifs >> t.m_Pwd)
    {
        vTea.push_back(t);
    }

    cout << "当前老师的数量为：" << vTea.size() << endl;
    ifs.close();
}

//检测重复    参数1 检测学号/职工号    参数2 检测类型
bool Manager::checkRepeat(int id, int type)
{
    if (type == 1)//检测学生
    {
        //遍历操作
        for (vector<Student>::iterator it = vStu.begin(); it != vStu.end(); it++)
        {
            if (id == it->m_Id)
            {
                return true;//发生了重复,返回真
            }
        }
    }
    else//检测老师
    {

        for (vector<Teacher>::iterator it = vTea.begin(); it != vTea.end(); it++)
        {
            if (id == it->m_EmpId)//职工id号
            {
                return true;//发生了重复，返回真
            }
        }
    }
    return false;//未发生重复
}
```

Identity.h 文件

```
#pragma once          //防止头文件重复包含
#include<iostream>
```

```
using namespace std;

//身份的抽象的基类
class Identity
{
public:

    //操作菜单 纯虚函数
    virtual void operMenu() = 0;

    //用户名
    string m_Name;
    //密码
    string m_Pwd;
};
```

orderFile.h 文件

```
using namespace std;
#include"globalFile.h"
#include<fstream>
#include<map>
#include<string>

class OrderFile
{
public:
    //构造函数
    OrderFile();

    //更新预约记录
    void updateOrder();

    //记录预约条数
    int m_Size;

    //记录所有预约信息的容器    key代表记录条数  value具体记录键值对信息
    map<int, map<string, string>>m_OrderData;
    };
```

orderFile.cpp 文件

```
#include"orderFile.h"

//构造函数
OrderFile::OrderFile()
{
```

```
    ifstream ifs;
    ifs.open(ORDER_FILE, ios::in);

    string date;          //日期
    string interval;      //时间段
    string stuId;         //学生编号
    string stuName;       //学生姓名
    string roomId;        //机房编号
    string status;        //预约状态

    this->m_Size = 0;     //记录的条数

    while (ifs >> date && ifs >> interval && ifs >> stuId && ifs >> stuName && ifs >> roomId
&& ifs >> status)
    {
        //测试数据是否可以获取
        /*cout << date << endl;
        cout << interval << endl;
        cout << stuId << endl;
        cout << stuName << endl;
        cout << roomId << endl;
        cout << status << endl;
        cout << endl;*/

        //date:1      example  date:1111  输出情况与所需不符，需要进行截取
        string key;
        string value;
        map<string, string>m;

        //截取日期
        int pos = date.find(":");   //4
        if (pos != -1)//防止超出
        {
            key = date.substr(0, pos);//截取子字符串
            value = date.substr(pos + 1, date.size() - pos - 1);     //size = 9  pos = 4
size - pos - 1 = 5 -1 //截取value值

            m.insert(make_pair(key, value));//每一个为对组形式
        }

        //cout << "key = " << key << endl;
        //cout << "value = " << value << endl;

        //截取时间段
```

```
pos = interval.find(":");
if (pos != -1)
{
    key = interval.substr(0, pos);
    value = interval.substr(pos + 1, interval.size() - pos - 1);

    m.insert(make_pair(key, value));
}



//截取学号
pos = stuId.find(":");
if (pos != -1)
{
    key = stuId.substr(0, pos);
    value = stuId.substr(pos + 1, stuId.size() - pos - 1);

    m.insert(make_pair(key, value));
}


//截取姓名
pos = stuName.find(":");
if (pos != -1)
{
    key = stuName.substr(0, pos);
    value = stuName.substr(pos + 1, stuName.size() - pos - 1);

    m.insert(make_pair(key, value));
}


//截取机房编号
pos = roomId.find(":");
if (pos != -1)
{
    key = roomId.substr(0, pos);
    value = roomId.substr(pos + 1, roomId.size() - pos - 1);

    m.insert(make_pair(key, value));
}
```

```
        //截取预约状态
        pos = status.find(":");
        if (pos != -1)
        {
            key = status.substr(0, pos);
            value = status.substr(pos + 1, status.size() - pos - 1);

            m.insert(make_pair(key, value));
        }


        //将小map容器放入到大的容器中
        this->m_OrderData.insert(make_pair(this->m_Size, m));
        this->m_Size++;

    }

    ifs.close();

    //测试最大map容器是否存储了所有的信息
    /*for (map<int, map<string, string>>::iterator it = m_OrderData.begin(); it !=
m_OrderData.end(); it++)
    {
        cout << "条数： " << it->first << "value = " << endl;
        for (map<string, string>::iterator mit = (*it).second.begin(); mit !=
(*it).second.end(); it++)
        {
            cout << "key = " << mit->first << "value = " << mit->second << " ";
        }
        cout << endl;
    }*/

}

//更新预约记录
void OrderFile::updateOrder()
{
    if (this->m_Size == 0)
    {
        return;    //预约记录0条，直接return
    }

    ofstream ofs(ORDER_FILE, ios::out | ios::trunc);//删除并且重新创建新文件
    for (int i = 0; i < this->m_Size; i++)//通过m_Size得到所需要写进文件的个数之后重新向
```

文件中写入数据

```
    {
        //通过第一个[i]找到需要访问的记录的条数，
        //此时获得map组中的value值，通过["date"]获取日期信息

        ofs << "date:" << this->m_OrderData[i]["date"] << " ";
        ofs << "interval:" << this->m_OrderData[i]["interval"] << " ";
        ofs << "stuId:" << this->m_OrderData[i]["stuId"] << " ";
        ofs << "stuName:" << this->m_OrderData[i]["stuName"] << " ";
        ofs << "roomId:" << this->m_OrderData[i]["roomId"] << " ";
        ofs << "status:" << this->m_OrderData[i]["status"] << endl;

    }
    ofs.close();
    }
```

globalFile.h 文件

```
#pragma once

//管理员文件
#define ADMIN_FILE     "admin.txt"
//学生文件
#define STUDENT_FILE   "student.txt"
//教师文件
#define TEACHER_FILE   "teacher.txt"
//机房信息文件
#define COMPUTER_FILE  "computerRoom.txt"
//订单文件
    #define ORDER_FILE     "order.txt"
```

ComputerRoom.h 文件

```
#pragma once
#pragma once
#include<iostream>
using namespace std;

//机房类
class ComputerRoom
{
public:
    int m_ComId;     //机房Id号

    int m_MaxNum;    //机房最大容量
};
    //机房类中存储机房的各类信息【包括机房 Id 号以及机房容量】
```