

机房预约系统

1、机房预约系统需求

1.1、系统简介

- 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。现开发一套机房预约系统，解决这一问题。



1.2 身份简介

分别有三种身份使用该程序：

- 学生代表： 申请使用机房
- 教室： 审核学生的预约申请
- 管理员： 给学生、教室创建账号

1.3 机房简介

机房总共有3间

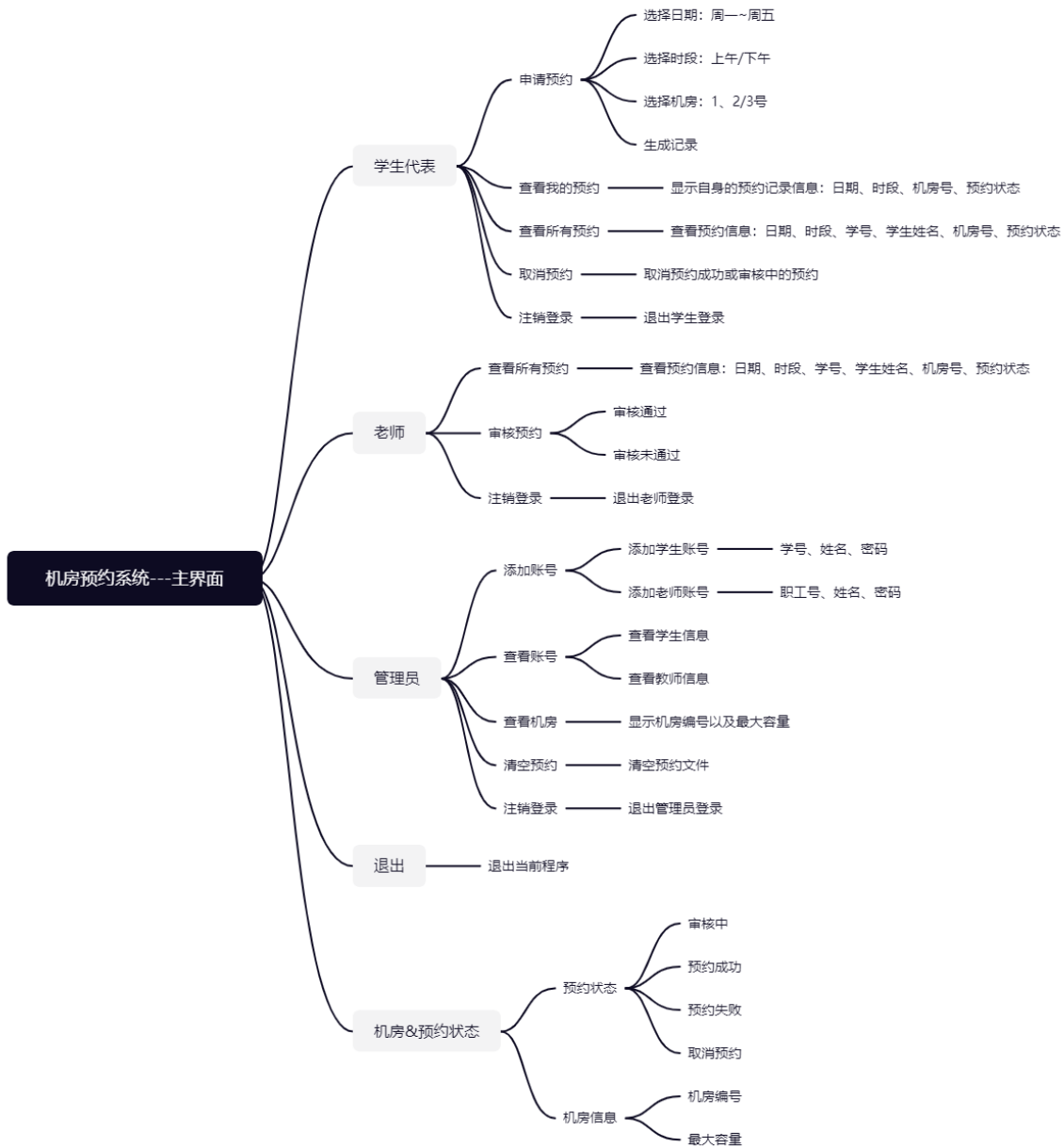
- 1号机房 ---- 最大容量20人
- 2号机房 ---- 最多容量50人
- 3号机房 ---- 最多容量100人

1.4 申请简介

1. 申请的订单每周由管理员负责清空
2. 学生可以预约未来一周内的机房使用，预约的日期为周一到周五，预约时间需要选择预约时段（上午、下午）
3. 教师来审核预约，依据实际情况审核预约通过或者不通过

1.5 系统具体需求

1. 首先进入登录界面，可选择登录身份有：
 - 学生代表
 - 老师
 - 管理员
 - 退出
2. 每个月都需要进行验证后，进入子菜单
 - 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
3. 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 --- 查看全部预约学习以及预约状态
 - 取消预约 --- 取消自身的预约，预约成功或审核中的预约均需要取消
4. 教师具体功能
 - 查看所有预约 ---- 查看全部预约学习以及预约状态
 - 审核预约 --- 对学生的预约进行审核
 - 注销登录 --- 退出登录
5. 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
 - 查看账户 --- 可以选择查看学生或教师的全部信息
 - 查看机构 --- 查看所有机房的信息
 - 情况预约 --- 情况所有预约记录
 - 注销登录 --- 退出登录、



2、 创建项目

创建项目步骤如下：

- 创建新项目
- 添加文件

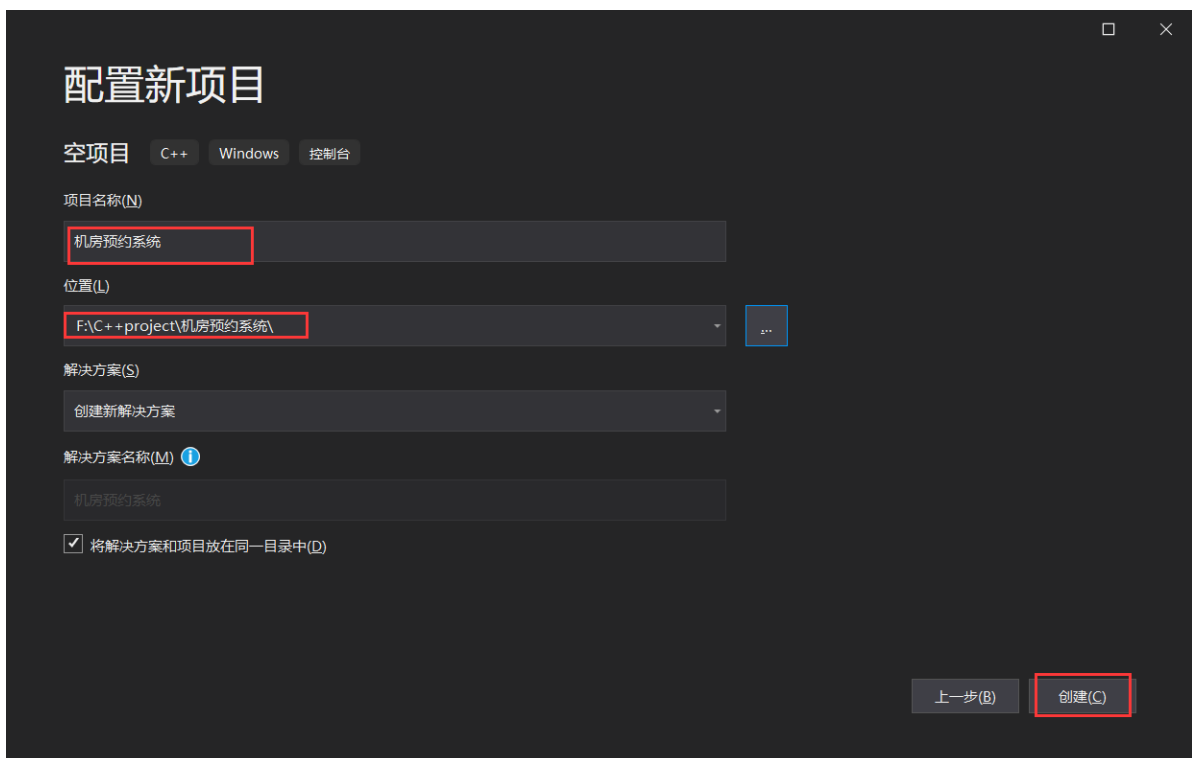
2.1 创建项目

- 打开VS2019后，点击创建新项目，创建新的C++项目

如图：

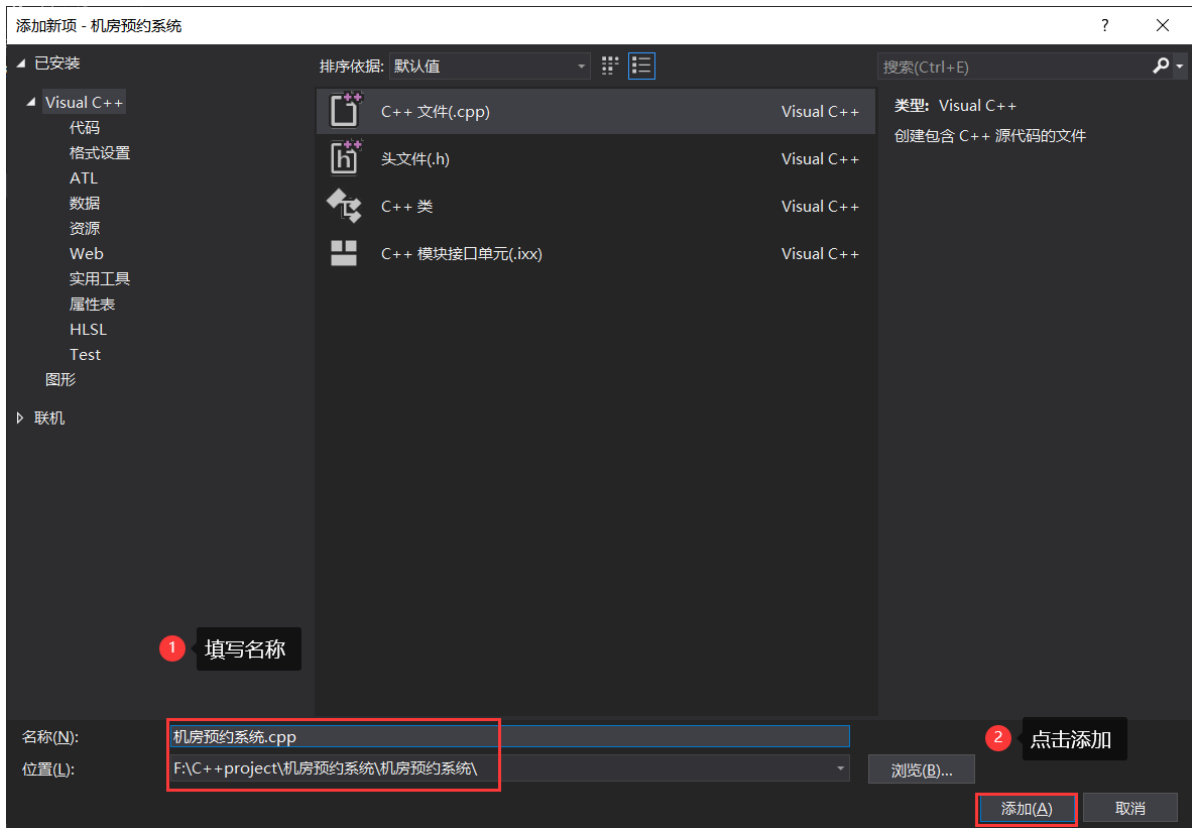


- 填写项目名称以及选取项目路径，点击确定生成项目

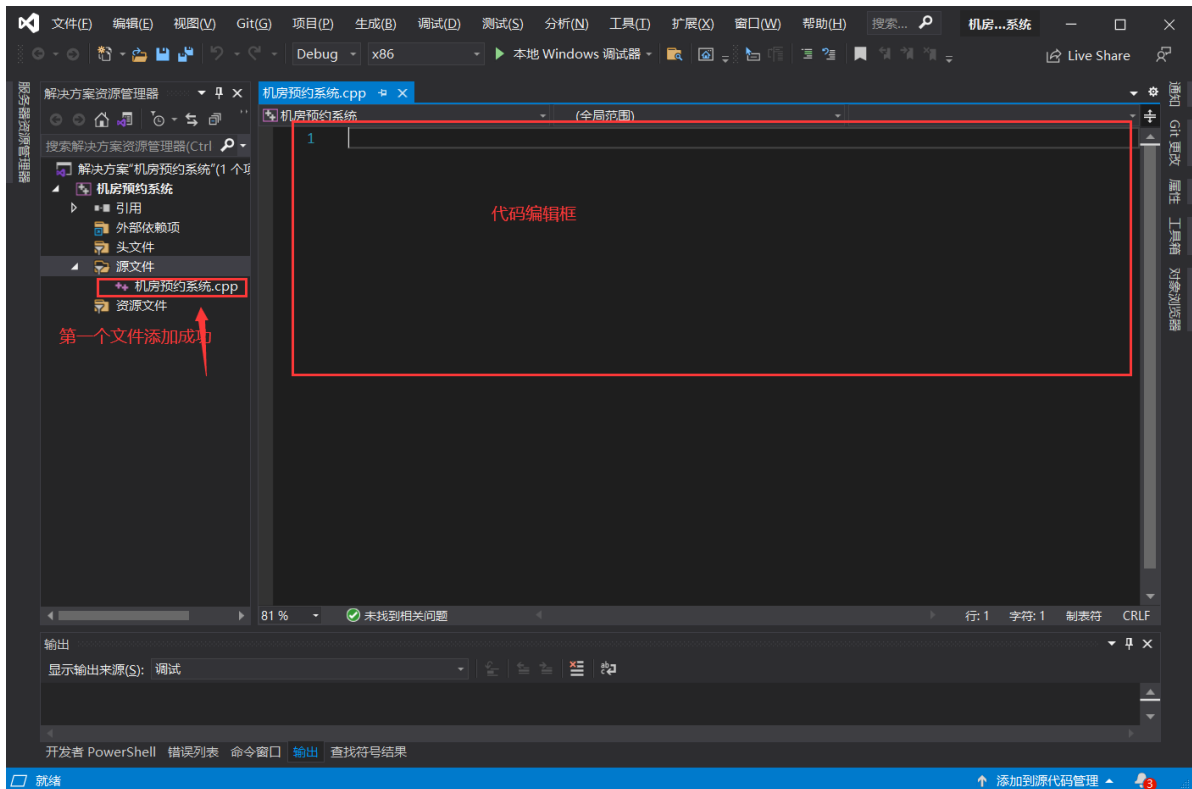


2.2 添加文件

1. 右键点击源文件，进行添加文件操作或者Ctrl + shift + A



2. 填写文件名称，点击添加



3、创建主菜单

功能描述：

- s合计主菜单，与用户进行交互

3.1 菜单实现

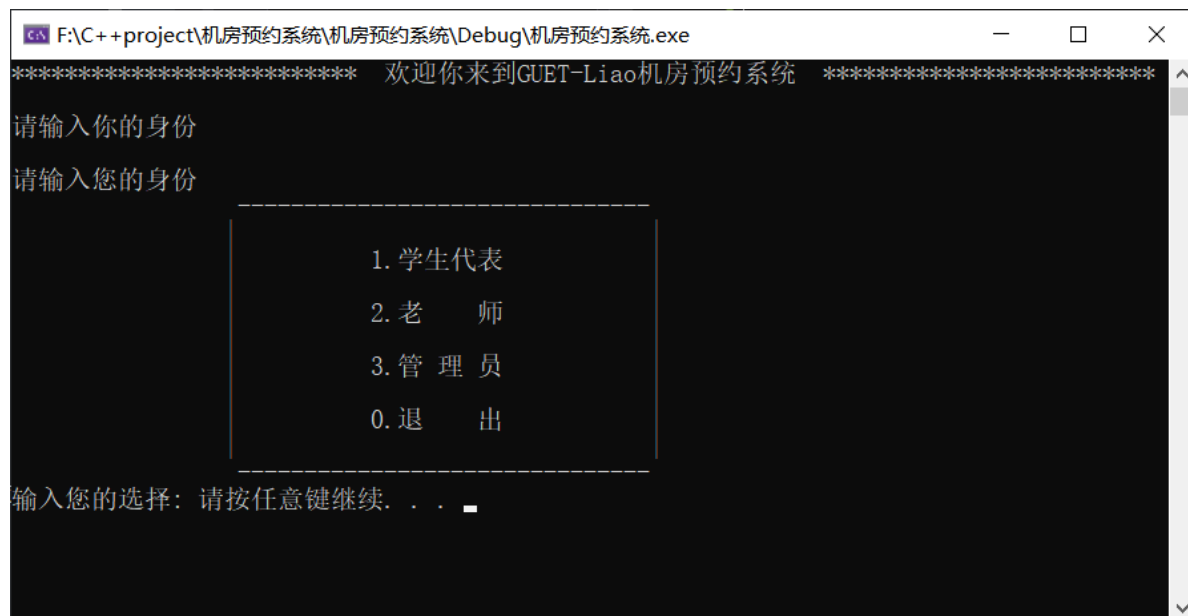
1. 在主函数main中添加菜单提示，代码如下：

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

int main()
{
    cout << "***** 欢迎你来到GUET-Liao机房预约系统 *****" << endl;
    cout << endl << "请输入你的身份" << endl;

    cout << endl << "请输入您的身份" << endl;
    cout << "\t\t -----\n";
    cout << "\t\t | \n";
    cout << "\t\t |      1.学生代表      | \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 << "输入您的选择： ";
    system("pause");
    return 0;
}
```

运行效果如图：



3.2 搭建接口

- 接收用户的选择，搭建接口
- 在main中添加代码

```
#include<iostream>
```

```

using namespace std;

int main()
{
    int select = 0;
    while (true)
    {
        cout << "***** 欢迎你来到GUET-Liao机房预约系统
*****" << endl;
        cout << endl << "请输入你的身份" << endl;

        cout << endl << "请输入您的身份" << endl;
        cout << "\t\t -----\n";
        cout << "\t\t|                               |\n";
        cout << "\t\t|          1.学生代表          |\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 << "输入您的选择: ";

        cin >> select; //接收用户的选择

        switch (select)
        {
            case 1: //学生身份
                break;

            case 2: //教师身份
                break;

            case 3: //管理员身份
                break;

            case 0: //退出系统
                break;

            default:
                cout << "输入有误，请重新输入你的选择: " << endl;
                system("pause");
                system("cls");
                break;
        }
    }

    system("pause");
    return 0;
}

```

测试:

- 输入0/1/2/3会重新返回到界面，输入其他时提示输入有误，清屏后重新选择

效果如图：



至此，界面搭建完毕！

4、退出功能实现

4.1 退出功能实现

在main函数分支0选项中，添加退出程序的代码；

```
cout << "输入有误，请重新输入你的选择：" << endl;  
system("pause");  
system("cls");
```



```
机房预约系统.cpp  [X]
[+] 机房预约系统 (全局范围) main()
25  cin >> select; //接收用户的选择
26
27  switch (select)
28  {
29      case 1: //学生身份
30          break;
31
32      case 2: //教师身份
33          break;
34
35      case 3: //管理员身份
36          break;
37
38      case 0: //退出系统
39          cout << "欢迎你下一次使用" << endl;
40          system("pause");
41          return 0;
42          break;
43
44      default:
45          cout << "输入有误, 请重新输入你的选择: " << endl;
46          system("pause");
47          ...
```

4.2 测试退出功能

预习程序如下图所示:

```
机房预约系统.cpp  [X]
[+] 机房预约系统 (全局范围) main()
25  cin >> select; //接收用户的选择
26
27  switch (select)
28  {
29      case 1: //学生身份
30          break;
31
32      case 2: //教师身份
33          break;
34
35      case 3: //管理员身份
36          break;
37
38      case 0: //退出系统
39          cout << "欢迎你下一次使用" << endl;
40          system("pause");
41          return 0;
42          break;
43
44      default:
45          cout << "输入有误, 请重新输入你的选择: " << endl;
46          system("pause");
47          ...
```

至此, 退出程序功能实现

5、创建身份类

5.1 身份的基类

- 在整个系统中，有三种身份，分别为：学生代表、老师以及管理员
- 三种身份有共性也有特性，因此我们可以将三种身份抽象出一个身份基类identity
- 在头文件下创建identity.h文件

identity.h文件中添加如下代码：

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

//身份抽象基类:把所有子类共性抽象成一个基类
class Identity
{
public:
    //操作菜单 纯虚函数必须重写
    virtual void operMenu() = 0;
    //用户名
    string m_Name;
    string m_Pwd;
};
```

5.2 学生类

5.2.1 功能分析

- 学生类主要功能是可以通过类中成员函数，实现预约实验室操作
- 学生类中主要功能有：
 - 显示学生操作的菜单界面
 - 申请预约
 - 查看自身预约
 - 查看所有预约
 - 取消预约

5.2.2 类的创建

- 在头文件以及源文件下创建 student.h 和 student.cpp 文件

student.h中添加如下代码：

```
#pragma once
#include<iostream>
using namespace std;
#include"identity.h"

//学生类
```

```

class Student :public Identity
{
    //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;

};

```

student.cpp中添加如下代码：

```

#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)
{

}

//菜单界面
void Student::operMenu()
{

}

//申请预约
void Student::applyOrder()
{

}

```

```

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

}

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

}

//取消预约
void Student::cancelOrder()
{

}

```

5.3 老师类

5.3.1 功能分析

- 教师类主要功能是查看学生的预约，并进行审核
- 教师类中主要功能有：
 - 显示教师操作的菜单界面
 - 查看所有预约
 - 审核预约

5.3.2 类的创建

- 在头文件以及源文件下创建 teacher.h 和 teacher.cpp文件

1. teacher.h中添加如下代码：

```

#pragma once
#include<iostream>
using namespace std;
#include"identity.h"

class Teacher : public Identity
{
public:
    //默认构造
    Teacher();

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

    //菜单界面
    virtual void operMenu() ;

    //查看所有预约

```

```
void showAllOrder();

//审核预约
void validOrder();

//职工号
int t_Id;
};
```

2. teacher.cpp中添加如下代码:

```
#include"teacher.h"

//默认构造
Teacher::Teacher()
{

}

//有参构造
Teacher::Teacher(int t_id, string name, string pwd)
{

}

//菜单界面
void Teacher::operMenu()
{

}

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

}

//审核预约
void Teacher::validOrder()
{

}
```

一开始只需要把这个系统框架搭建起来即可，后面开发将会思路非常清晰、实现容易

5.4 管理员类

5.4.1 功能分析

- 管理员类主要功能是对学生和教师账户进行管理，查看机房信息以及清空预约记录
- 管理员类中主要功能有：
 - 显示管理员操作的菜单界面
 - 添加账号
 - 查看账号
 - 查看机房信息
 - 清空预约记录

5.4.2 类的创建

- 在头文件以及源文件下创建 manager.h 和 manager.cpp 文件

1. manager.h 中添加如下代码：

```
#pragma once
#include<iostream>
using namespace std;
#include"identity.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);

    //显示菜单
    virtual void operMenu() ;

    //添加账号
    void addPerson();

    //查看账号
    void showPerson();

    //查看机房信息
    void showComputer();

    //清空预约记录
    void clearFile();

};
```

2. manager.cpp 中添加如下代码：

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

```

//默认构造
Manager::Manager()
{

}

//有参构造
Manager::Manager(string name, string pwd)
{

}

//显示菜单
void Manager::operMenu()
{

}

//添加账号
void Manager::addPerson()
{

}

//查看账号
void Manager::showPerson()
{

}

//查看机房信息
void Manager::showComputer()
{

}

//清空预约记录
void Manager::clearFile()
{

}

```

6、登录模块

6.1 全局文件添加

功能描述：

- 不同的身份可能会用到不同的文件，我们就要将所有的文件定义到一个全局文件中
- 在头文件中添加 **globalFile.h** 文件
- 并添加如下代码

```
#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"
```

并且在同级目录下，创建这几个文件

> 此电脑 > 新加卷 (F:) > C++project > 机房预约系统 > 机房预约系统					在 机房预约系统 中搜
名称	修改日期	类型	大小		
.vs	2022/9/11 16:08	文件夹			
Debug	2022/9/11 23:41	文件夹			
admin.txt	2022/9/12 9:11	文本文档			
computerRoom.txt	2022/9/12 9:12	文本文档			
globalFile.h	2021/3/18 14:24	C/C++ Header			
identity.h	2022/9/11 23:13	C/C++ Header			
manager.cpp	2022/9/11 23:41	C++ Source			
manager.h	2022/9/11 23:41	C/C++ Header			
order.txt	2022/9/12 9:12	文本文档			
student.cpp	2022/9/11 23:13	C++ Source			
student.h	2022/9/11 23:13	C/C++ Header			
student.txt	2022/9/12 9:12	文本文档			
teacher.cpp	2022/9/11 23:41	C++ Source			
teacher.h	2022/9/11 23:41	C/C++ Header			
teacher.txt	2022/9/12 9:12	文本文档			
机房预约系统.cpp	2022/9/11 16:44	C++ Source			
机房预约系统.sln	2022/9/11 16:08	Visual Studio Sol...			
机房预约系统.vcxproj	2022/9/11 23:41	VC++ Project			
机房预约系统.vcxproj.filters	2022/9/11 23:41	VC++ Project Fil...			
机房预约系统.vcxproj.user	2022/9/11 16:08	Per-User Project...			

6.2 登录函数封装

功能描述：

- 根据用户的选择，进入不同的身份登录

在预约系统的 .cpp 文件中添加全局函数 void LoginIn(string fileName, int type);

参数:

- fileName --- 操作的文件名
- type --- 登录的身份(1代表学生、2代表老师、3代表管理员)

LoginIn 中添加如下代码:

```
//登录功能  参数1  操作文件名 参数2  操作身份类型
void LoginI(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;
    }

    //准备接收用户的信息
    int id = 0;
    string name;
    string pwd;

    //身份判断
    if (type == 1) //学生身份
    {
        cout << "请输入你的学号: " << endl;
        cin >> id;
    }
    else if (type == 2)
    {
        cout << "请输入你的职工号: " << 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
    }
    else if (type == 2)
    {
        //老师身份验证
    }
}
```

```

    }
    else if (type == 3)
    {
        //管理员身份验证

    }

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

    return;
}

```

- 在main函数中的不同分支中，填入不同的登录接口

```

globalFile.h*  manager.h  manager.cpp  teacher.cpp  student.cpp  机房预约系统.cpp*
(全局范围)  Loginl(string fileName, int type)

97  cin >> select; //接收用户的选择
98
99  switch (select)
100 {
101     case 1: //学生身份
102         Loginl(STUDENT_FILE, select);
103         break;
104
105     case 2: //教师身份
106         Loginl(TEACHER_FILE, select);
107         break;
108
109     case 3: //管理员身份
110         Loginl(ADMIN_FILE, select);
111         break;
112
113     case 0: //退出系统
114         cout << "欢迎你下一次使用" << endl;
115         system("pause");
116         return 0;
117         break;
118

```

6.3 学生登录实现

在student.txt 文件中添加两条学生信息，用于测试

添加信息：

```

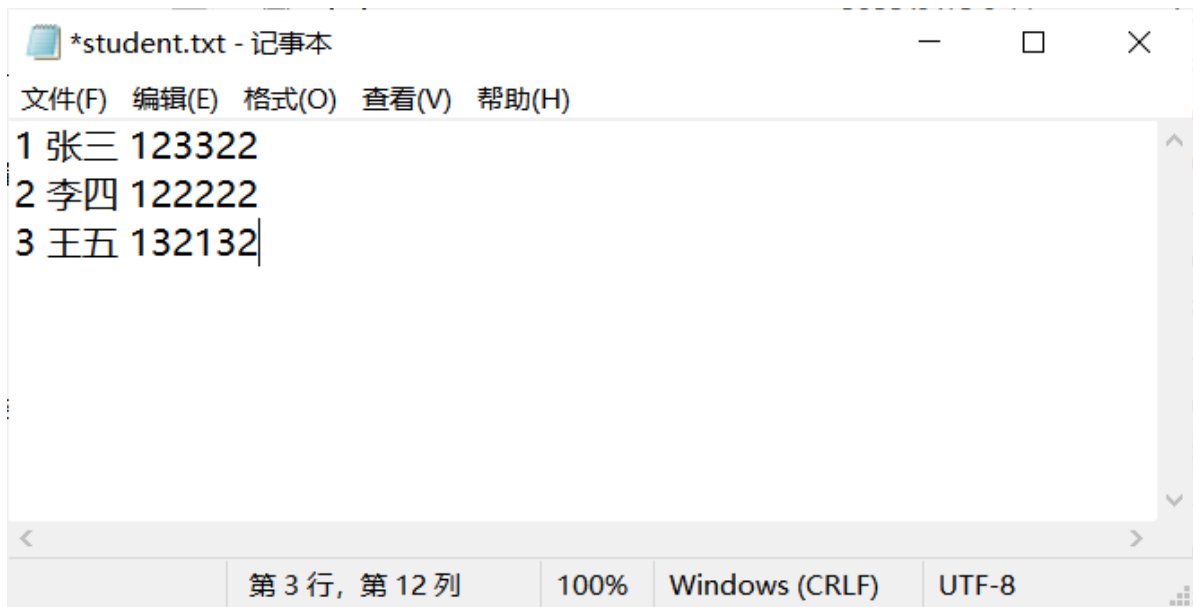
1 张三 123322
2 李四 122222
3 王五 132132

```

其中：

- 第一列 代表 学号
- 第二列 代表 学生姓名
- 第三列 代表 密码

效果图如下：



在Login函数的学生分支中加入如下代码，验证学生身份

```
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");
        //创建一个学生示例对象
        person = new Student(id, name, pwd);

        //进入学生身份的子菜单

        return;
    }
}
```

添加代码效果图：

```
manager.h    manager.cpp    teacher.cpp    student.cpp    机房预约系统.cpp x student.h
机房预约系统 (全局范围) Login(string fileName, int t
49
50     cout << "请输入密码: " << endl;
51     cin >> pwd;
52
53     if (type == 1)
54     {
55         //学生身份验证
56         int fId; //从文件中读取的id号
57         string fName; //从文件中获取的姓名
58         string fPwd; //从文件中获取密码
59         while (ifs>>fId && ifs>>fName && ifs>>fPwd)
60         {
61             //与用户输入的信息做对比
62             if (fId == id && fName == name && fPwd == pwd)
63             {
64                 cout << "学生验证登录成功! " << endl;
65                 system("pause");
66                 system("cls");
67                 //创建一个学生示例对象
68                 person = new Student(id, name, pwd);
69
70                 //进入学生身份的子菜单
71
72                 return;
73             }
74
75
```

测试:

```
F:\C++project\机房预约系统\机房预约系统\Debug\机房预约系统.exe
***** 欢迎你来到GUET-Liao机房预约系统 *****
*****

请输入你的身份
-----
1. 学生代表
2. 老师
3. 管理员
0. 退出
-----

输入您的选择: 1
请输入你的学号:
1
请输入用户名:
张三
请输入密码:
123322
学生验证登录成功!
请按任意键继续. . .
```

6.4 教师登录实现

在teacher.txt 文件中添加一条老师信息，用于测试

添加信息：

```
1 老王 321
2 老廖 666
```

其中：

- 第一列 代表 教师职工编号
- 第二列 代表 教师姓名
- 第三列 代表 密码

效果图：



在Login函数的教师分支中加入如下代码，验证教师身份

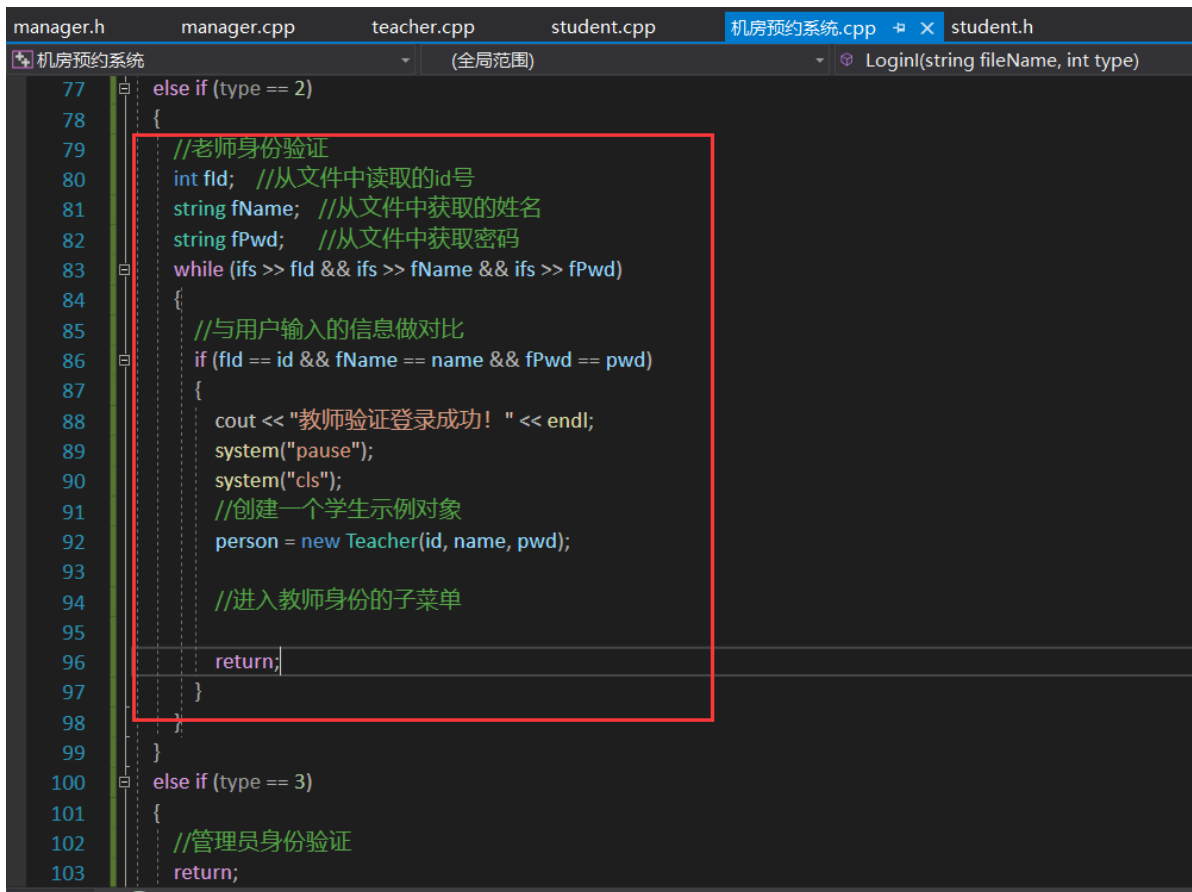
```
//老师身份验证
int fId;           //从文件中读取的id号
string fName;      //从文件中获取的姓名
string fPwd;       //从文件中获取密码
while (ifs >> fId && ifs >> fName && ifs >> fPwd)
{
    //与用户输入的信息做对比
    if (fId == id && fName == name && fPwd == pwd)
    {
        cout << "教师验证登录成功！" << endl;
        system("pause");
        system("cls");
        //创建一个学生示例对象
        person = new Teacher(id, name, pwd);

        //进入教师身份的子菜单

        return;
    }
}
```

```
}
```

添加代码效果图：



```
manager.h  manager.cpp  teacher.cpp  student.cpp  机房预约系统.cpp  student.h
机房预约系统 (全局范围) Login(string fileName, int type)
77 else if (type == 2)
78 {
79     //老师身份验证
80     int fId; //从文件中读取的id号
81     string fName; //从文件中获取的姓名
82     string fPwd; //从文件中获取密码
83     while (ifs >> fId && ifs >> fName && ifs >> fPwd)
84     {
85         //与用户输入的信息做对比
86         if (fId == id && fName == name && fPwd == pwd)
87         {
88             cout << "教师验证登录成功！" << endl;
89             system("pause");
90             system("cls");
91             //创建一个学生示例对象
92             person = new Teacher(id, name, pwd);
93
94             //进入教师身份的子菜单
95
96             return;
97         }
98     }
99 }
100 else if (type == 3)
101 {
102     //管理员身份验证
103     return;
```

测试代码：



6.5 管理员登录实现

在admin.txt 文件中添加一条管理员信息，由于我们只有一条管理员，因此本案例中没有添加管理员的功能

添加管理员信息：

```
admin 888
```

其中：

- admin 代表管理员用户名
- 888 代表管理员密码

效果图：



在Login函数的管理员分支中加入如下代码，验证管理员身份

```
//管理员身份验证
string fName;
string fPwd;
while (ifs >> fName && ifs >> fPwd)
{
    if (fName == name && pwd == fPwd)
    {
        cout << "验证登录成功! " << endl;
        //登录成功后，按任意键进入管理员界面
        system("pause");
        system("cls");

        //创建管理员对象
        person = new Manager(name,pwd);

        return;
    }
}
```

添加效果如图：

```
manager.h    manager.cpp    teacher.cpp    student.cpp    机房预约系统.cpp*    student.h
机房预约系统 (全局范围) Login(string fileName, int type)
98 }
99 }
100 else if (type == 3)
101 {
102     //管理员身份验证
103     string fName;
104     string fPwd;
105     while (ifs >> fName && ifs >> fPwd)
106     {
107         if (fName == name && fPwd == fPwd)
108         {
109             cout << "验证登录成功！" << endl;
110             //登录成功后，按任意键进入管理员界面
111             system("pause");
112             system("cls");
113
114             //创建管理员对象
115             person = new Manager(name, fPwd);
116
117             return;
118         }
119     }
120     return;
121 }
122
123 cout << "验证登录失败！" << endl;
124 system("pause");
```

测试代码：

```
F:\C++\project\机房预约系统\机房预约系统\Debug\机房预约系统.exe
***** 欢迎你来到GUET-Liao机房预约系统 *****
请输入你的身份
-----
1. 学生代表
2. 老师
3. 管理员
0. 退出
-----
输入您的选择：3
请输入用户名：
admin
请输入密码：
888
验证登录成功！
请按任意键继续. . .
```

至此，所有身份的登录功能全部实现！

6.6 遇到输出乱码问题

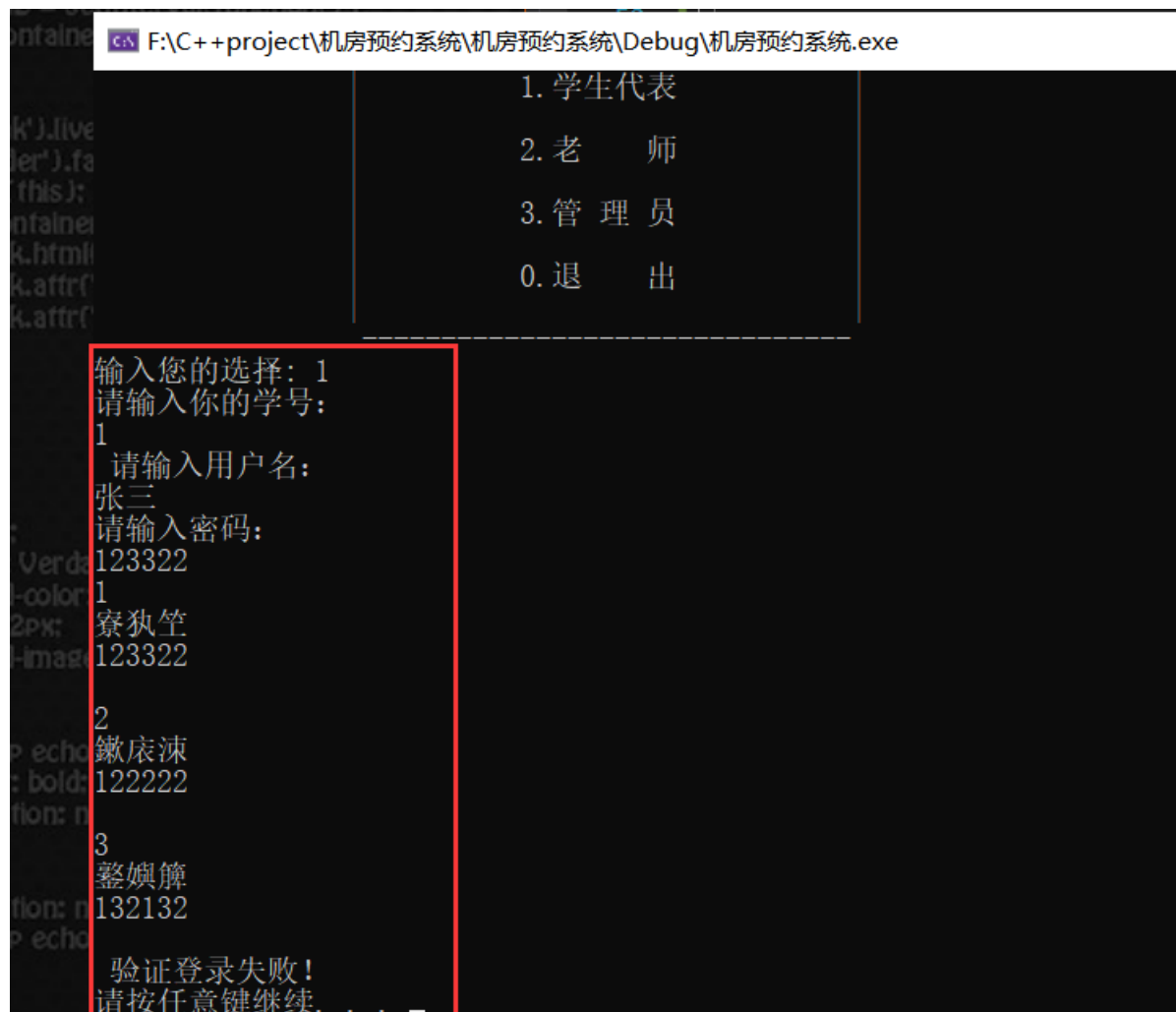
关于Visual Studio 2019控制台输出中文出现乱码问题及解决办法

环境：Windows11 Visual studio 2019

中文输出出现乱码，分为2种情况

1、从.txt文件中读取输出出现乱码

1. 结果如下图所示：



The screenshot shows a console window titled "F:\C++project\机房预约系统\机房预约系统\Debug\机房预约系统.exe". The menu displayed is:

```
1. 学生代表
2. 老师
3. 管理员
0. 退出
```

Below the menu, the user input and system response are shown, with the input area highlighted by a red box:

```
输入您的选择: 1
请输入你的学号:
1
  请输入用户名:
张三
  请输入密码:
123322
1
  寮狄涑
123322
2
  鍬宸涑
122222
3
  鏁愯簮
132132
验证登录失败!
请按任意键继续...
```

2. 分析出现乱码的原因

首先我们先得知道windows下的中文是GBK编码，VS2019也是GBK编码，所以在windows里编程中文最好就是用GBK编码。

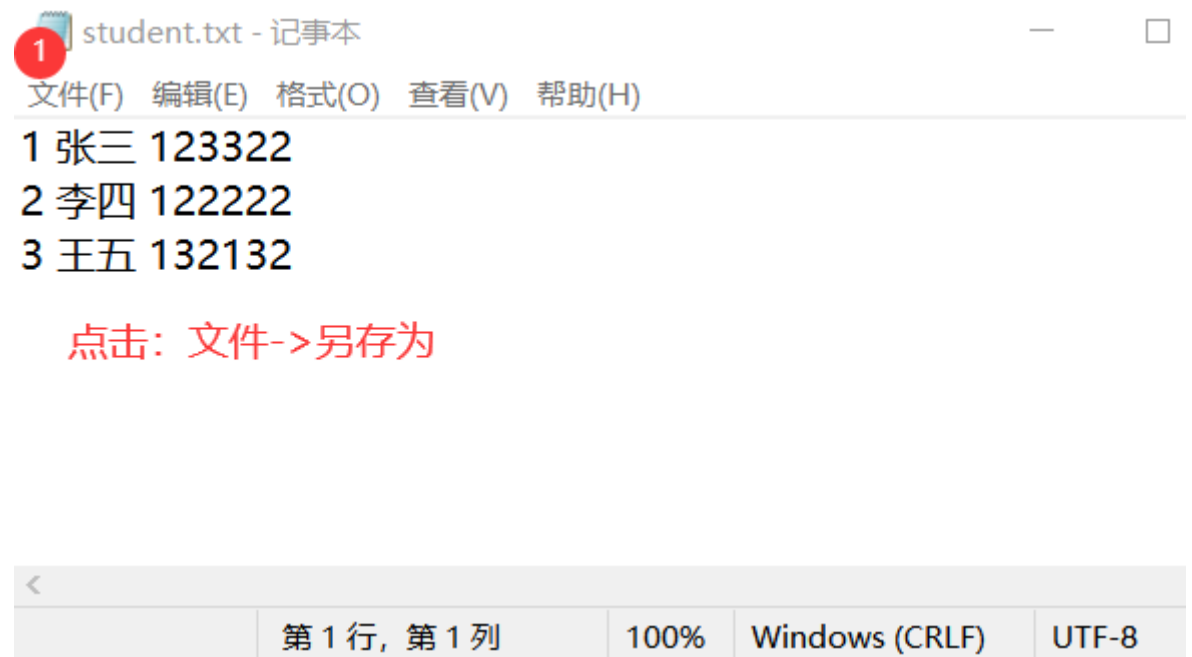
中文乱码的本质就是**编码不匹配的问题**，就好像明明是utf-8的编码你用GBK去理解当然是无法匹配的。



3. 解决办法

我们知道原因就解决办法就显而易见了：将.txt文件utf-8的编码改为和VS2019一样的GBK编码

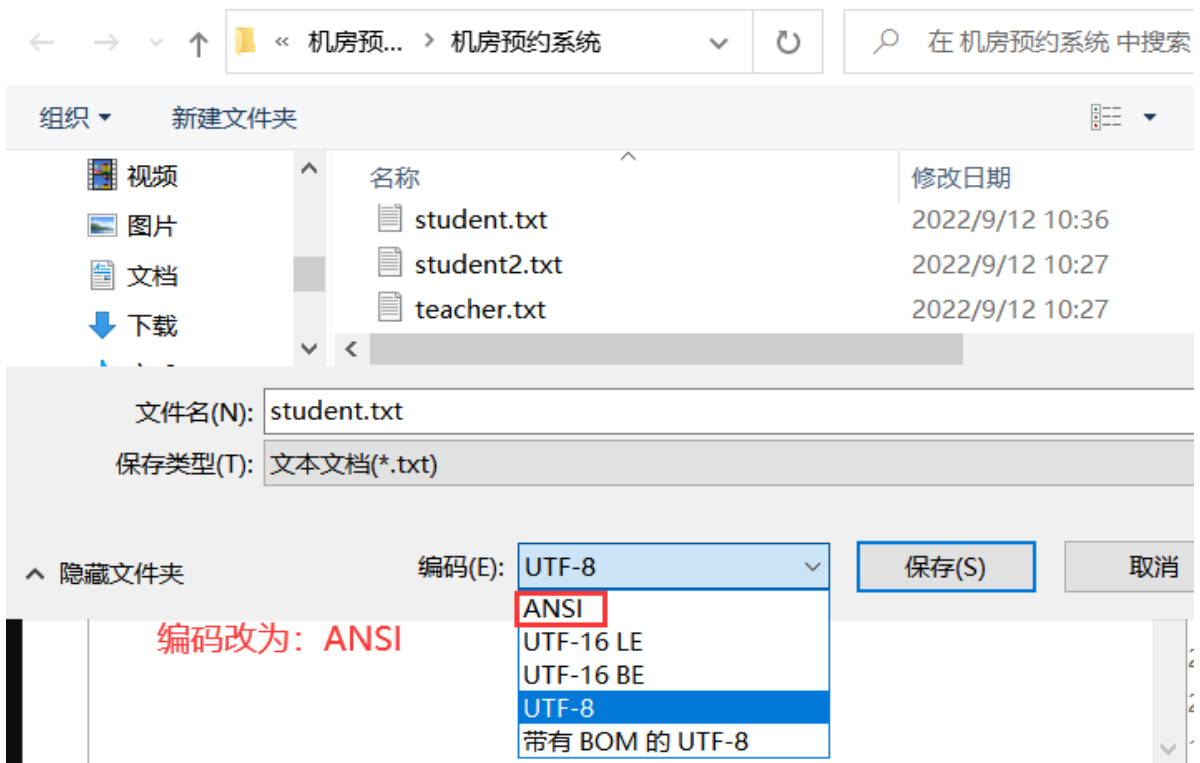
第一步：



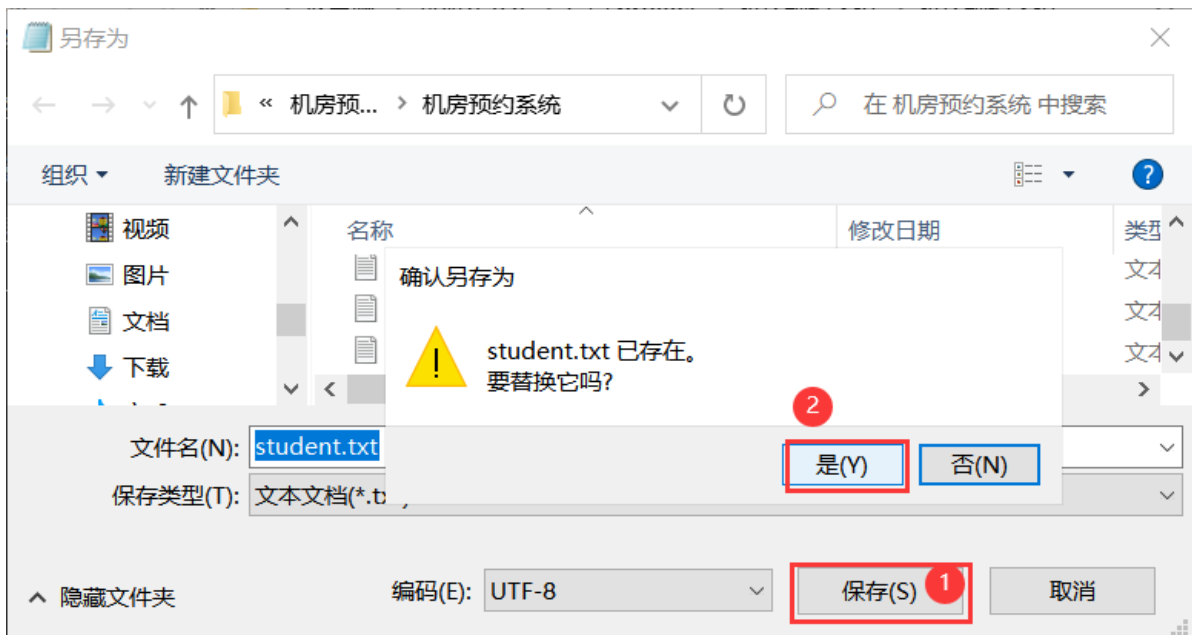
点击：文件->另存为

第二步：

另存为

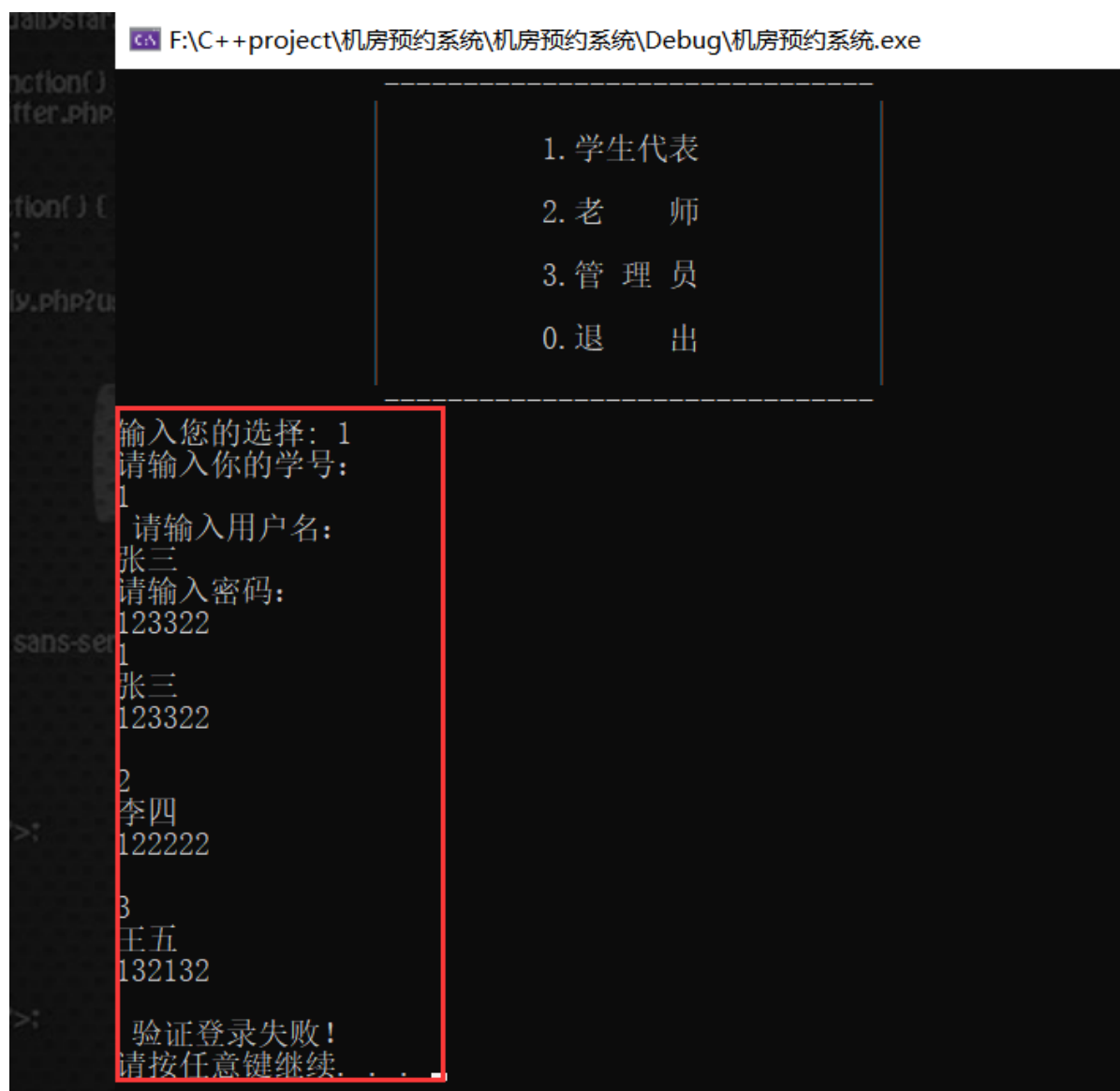


第三步:



第四步:

重新运行代码，进行测试



输出正确

2、从中文字符串中读取输出出现乱码

运行test.c文件出现乱码

```
char s[20] = "中文乱码问题";  
printf("%s\n", s);
```

1. 打开test.c文件所在位置
2. 用右键选择用文本文件方式打开
3. 与上面第一种操作相同，把编码改为GBK编码，请参考上面进行修改

7、管理员模块

7.1 管理员登录和注销

7.1.1 构造函数

- 在Manager类的构造函数中，初始化管理员信息，代码如下：

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

7.1.2 管理员子菜单

- 在机房预约系统.cpp中，当用户登录的是管理员，添加管理员菜单接口
- 将不同的分支提供出来
 - 添加账号
 - 查看账号
 - 查看机房
 - 清空预约
 - 注销登录
- 实现注销功能

添加全局函数 `void managerMenu(Identity * &manager)`，代码如下：

7.1.3 菜单功能实现

- 在实现成员函数 `void Manager::operMenu()` 代码如下：

```
//选择菜单
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;
```

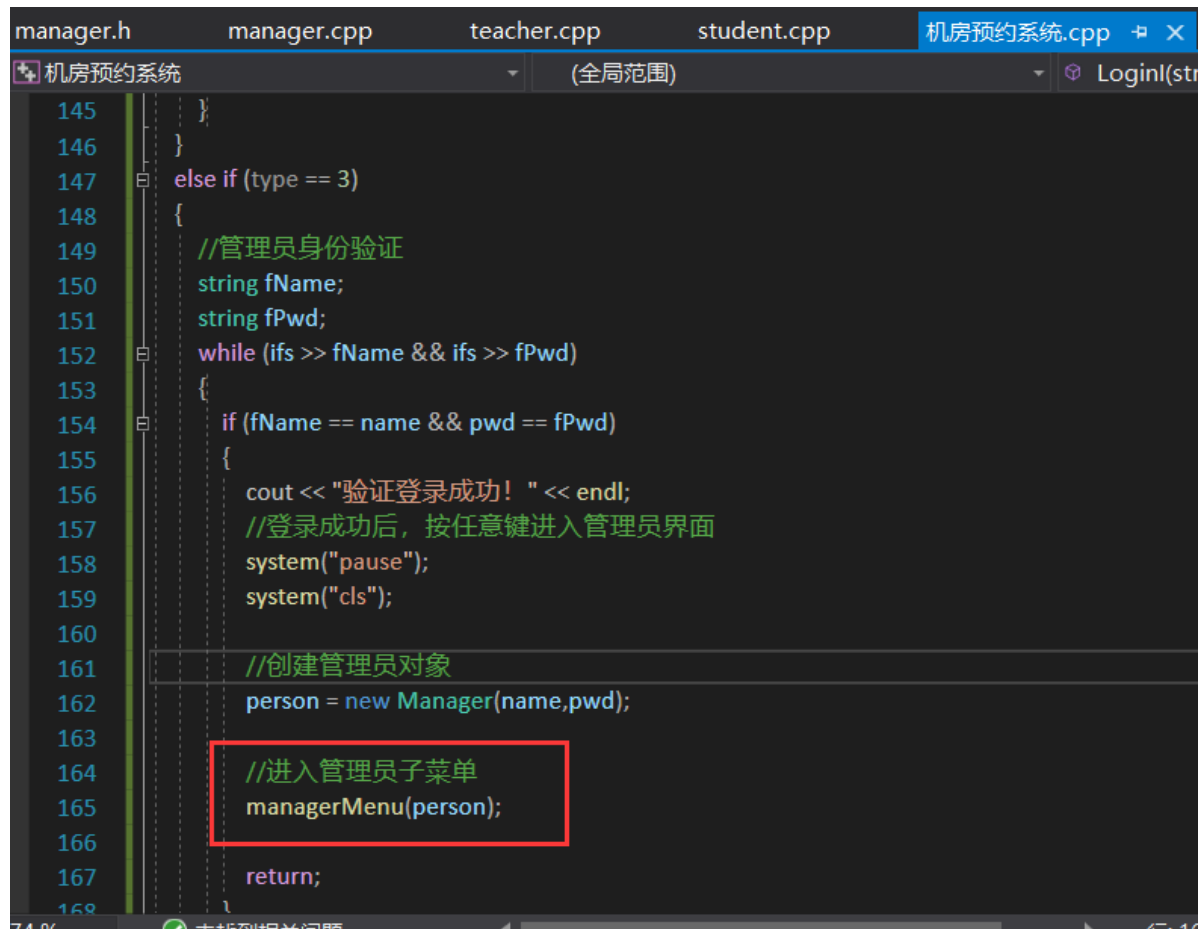
```
}
```

7.1.4 接口对接

- 管理员成功登录后，调用管理员子菜单界面
- 在管理员登录验证分支中，添加代码：

```
//进入管理员子菜单  
managerMenu(person);
```

添加如下：

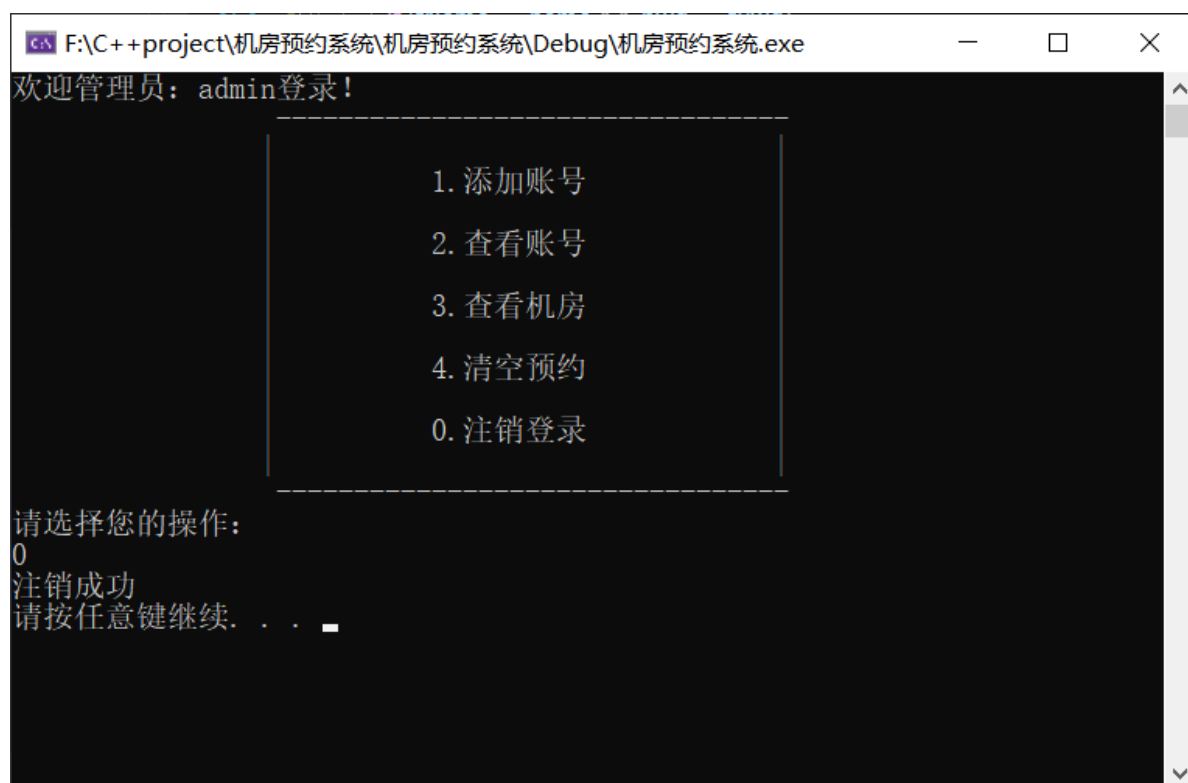


```
manager.h    manager.cpp    teacher.cpp    student.cpp    机房预约系统.cpp  
机房预约系统 (全局范围) LoginI(str  
145     }  
146     }  
147     else if (type == 3)  
148     {  
149         //管理员身份验证  
150         string fName;  
151         string fPwd;  
152         while (ifs >> fName && ifs >> fPwd)  
153         {  
154             if (fName == name && pwd == fPwd)  
155             {  
156                 cout << "验证登录成功! " << endl;  
157                 //登录成功后, 按任意键进入管理员界面  
158                 system("pause");  
159                 system("cls");  
160  
161                 //创建管理员对象  
162                 person = new Manager(name,pwd);  
163  
164                 //进入管理员子菜单  
165                 managerMenu(person);  
166  
167                 return;  
168             }
```

测试对接，效果如图：



登录并注销成功:



至此，管理员身份可以登录以及注销

7.2 添加账号

功能描述:

- 给学生或教师添加新的账号

功能要求:

- 添加是学生学号不能重复、教师职工号不能重复

7.2.1 添加功能实现

在Manager的addPerson成员函数中，实现添加新账号功能，代码如下：

```
//添加账号
void Manager::addPerson()
{
    cout << "请输入添加账号类型: " << 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

    ofstream ofs;        //文件操作对象

    int select = 0;
    cin >> select;    //接收用户的选择

    while (select != 1 && select != 2)
    {
        cout << "输入有误，请重新输入:" << endl;
        cin >> select;
    }

    if (select == 1)
    {
        //添加的是学生
        fileName = STUDENT_FILE;
        tip = "请输入学号: ";
    }
    else
    {
        //添加的是老师
        fileName = TEACHER_FILE;
        tip = "请输入职工号: ";
    }

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

    int id; //学号或职工号
    string name;    //姓名
    string pwd; //密码

    cout << tip << endl;
    cin >> id;

    cout << "请输入姓名: " << endl;
    cin >> name;

    cout << "请输入密码: " << 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();  
}
```

添加学生：



成功在学生文件中统计信息：



添加老师：

```
F:\C++\project\机房预约系统\机房预约系统\Debug\机房预约系统.exe
欢迎管理员：admin登录！

-----
1. 添加账号
2. 查看账号
3. 查看机房
4. 清空预约
0. 注销登录
-----

请选择您的操作：
1
添加账号
请输入添加账号类型：
1、添加学生
2、添加老师
2
请输入职工号：
3
请输入姓名：
老刘
请输入密码：
666
添加成功
请按任意键继续. . .
```

成功在老师文件中添加信息：

```
teacher.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
1 老廖 666
2 老王 888
3 老刘 666

第 4 行, 第 1 列    100%    Windows (CRLF)    ANSI
```

7.2.2 去重操作

功能描述：

- 添加新账号是，如果是重复的学生编号，或是重复的教师职工编号，提示有误

7.2.1.1 获取消息

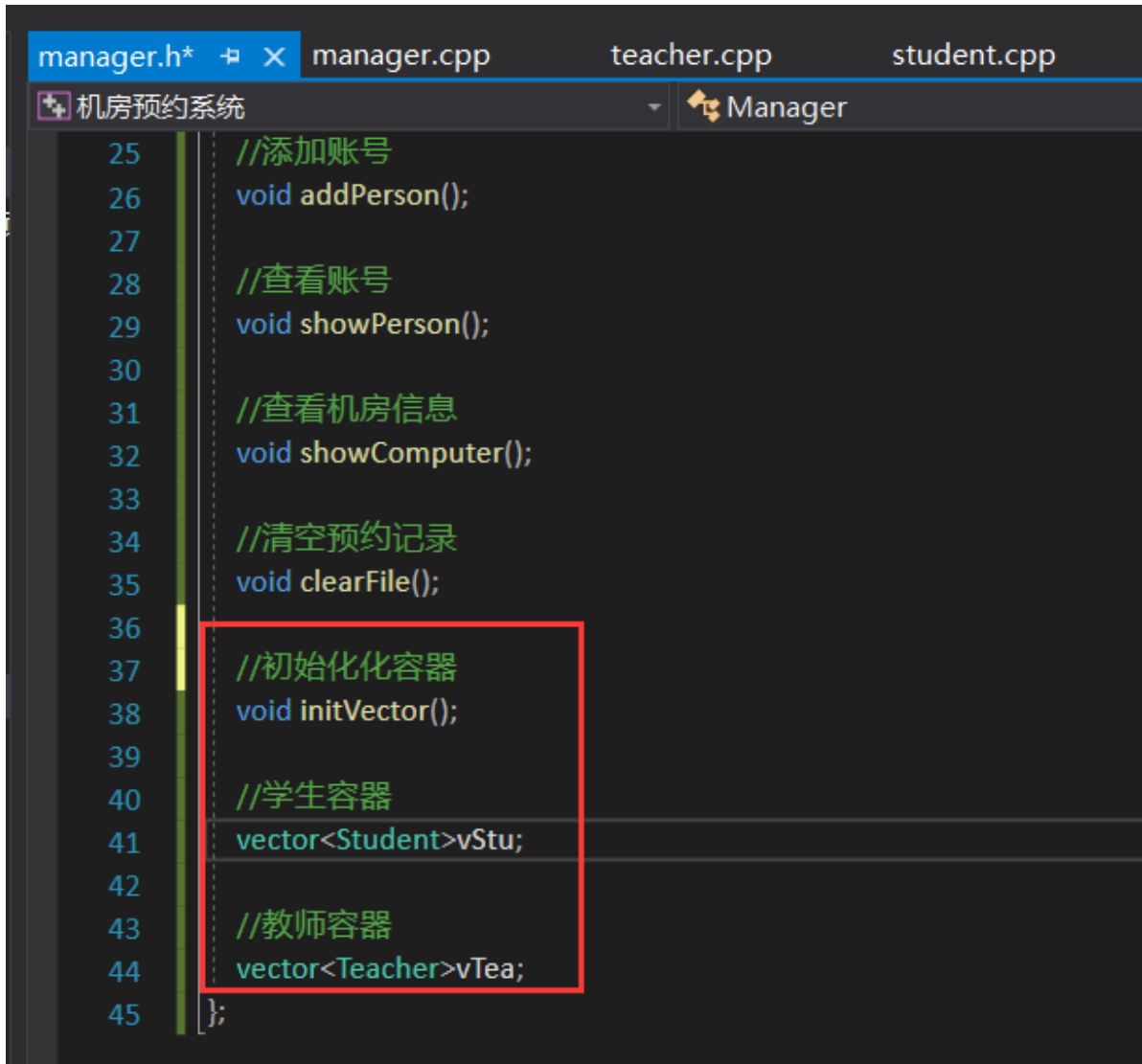
- 要去除重复的账号，首先要先将学生和老师的账号信息获取到程序中，方可检测
- 在manager.h 中，添加两个容器，用于存放学生和教室的信息
- 添加一个新的成员函数， void initVector() 初始化容器

```
//初始化化容器
void initVector();

//学生容器
vector<Student>vStu;

//教师容器
vector<Teacher>vTea;
```

添加位置:



```
manager.h* x manager.cpp teacher.cpp student.cpp
机房预约系统 Manager
25 //添加账号
26 void addPerson();
27
28 //查看账号
29 void showPerson();
30
31 //查看机房信息
32 void showComputer();
33
34 //清空预约记录
35 void clearFile();
36
37 //初始化化容器
38 void initVector();
39
40 //学生容器
41 vector<Student>vStu;
42
43 //教师容器
44 vector<Teacher>vTea;
45 };
```

在Manager的有参构造函数中，获取目前的学生和教师信息

代码如下：

```
void Manager::initVector()
{
    //首先把容器进行清空
    vStu.clear();
    vTea.clear();

    //读取信息 学生
    ifstream ifs;
    ifs.open(STUDENT_FILE, ios::in);
    if (!ifs.is_open())
```

```

{
    cout << "文件读取失败" << endl;
    return;
}
Student s;
while (ifs >> s.m_Id && ifs>> s.m_Name && ifs>>s.m_Pwd)
{
    vStu.push_back(s);
}

cout << "当前学生数量为: " << vStu.size() << endl;
ifs.close();

//读取老师信息
ifs.open(TEACHER_FILE, ios::in);
if (!ifs.is_open())
{
    cout << "文件读取失败" << endl;
    return;
}
Teacher t;
while (ifs>>t.t_Id && ifs>>t.m_Name&&ifs>>t.m_Pwd)
{
    vTea.push_back(t);
}
cout << "当前老师数量为: " << vTea.size() << endl;
ifs.close();
}

```

在有参构造函数中，调用初始化容器函数

```

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

    //初始化容器 获取到所有文件中 学生、老师信息
    this->initVector();
}

```

测试，运行代码可以看到测试代码获取当前学生和老师数量：



7.2.2.2 去重函数封装

在manager.h 文件中添加成员函数 `bool checkRepeat(int id, int type);`

```
//检测重复 参数:(传入id, 传入类型) 返回值: (true 代表有重复, false代表没有重复)
bool checkRepeat(int id, int type);
```

在manager.cpp 文件中实现成员函数: `bool ccheckRepeat(int id, int type);`

```
//检测重复 参数:(传入id, 传入类型) 返回值: (true 代表有重复, false代表没有重复)
bool Manager::checkRepeat(int id, int type)
{
    if (type == 1)
    {
        for (vector<Student>::iterator it = vstu.begin(); it != vstu.end(); it++)
        {
            if (id == it->m_Id)
            {
                return true;
            }
        }
    }
    else
    {
        for (vector<Teacher>::iterator it = vTea.begin(); it !=vTea.end(); it++)
        {
            if (id == it->t_Id)
            {
                return true;
            }
        }
    }
    return false;
}
```

7.2.2.3 添加去重操作

在void addPerson()函数中添加学生编号或者教师职工号时，检测是否有重复，代码如下：

```
//添加账号
void Manager::addPerson()
{
    cout << "请输入添加账号类型: " << 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;     //提示错误

    ofstream ofs;        //文件操作对象

    int select = 0;
    cin >> select;    //接收用户的选择

    while (select != 1 && select != 2)
    {
        cout << "输入有误，请重新输入:" << endl;
        cin >> select;
    }

    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
    {
        //添加的是老师
        fileName = TEACHER_FILE;
        tip = "请输入职工号: ";
        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);

    int id; //学号或职工号
    string name; //姓名
    string pwd; //密码
    cout << tip << endl;

    while (true)
    {
        cin >> id;
        bool res = checkRepeat(id, select);
        if (res)
        {
            cout << errorTip << endl;
        }
    }
}
```

```
    }  
    else  
    {  
        break; //退出循环  
    }  
}
```

代码添加位置如下：

```
manager.h  manager.cpp  teacher.cpp  student.cpp  机房预约系统.cpp  student.h
机房预约系统  → Manager  addPerson()

46
47 string fileName; //操作文件名
48 string tip; //提示id号
49 string errorTip; //提示错误
50
51 ofstream ofs; //文件操作对象
52
53 int select = 0;
54 cin >> select; //接收用户的选择
55
56 while (select != 1 && select != 2)
57 {
58     cout << "输入有误, 请重新输入:" << endl;
59     cin >> select;
60 }
61
62 if (select == 1)
63 {
64     //添加的是学生
65     fileName = STUDENT_FILE;
66     tip = "请输入学号: ";
67     errorTip = "学号重复, 请重新输入";
68 }
69 else
70 {
71     //添加的是老师
72     fileName = TEACHER_FILE;
73     tip = "请输入职工号: ";
74     errorTip = "职工号重复, 请重新输入";
75 }
76
77 //利用追加的方式写文件
78 ofs.open(fileName, ios::out | ios::app);
79
80 int id; //学号或职工号
81 string name; //姓名
82 string pwd; //密码
83 cout << tip << endl;
84
85 while (true)
86 {
87     cin >> id;
88     bool res = checkRepeat(id, select);
89     if (res)
90     {
91         cout << errorTip << endl;
92     }
93     else
94     {
95         break; //退出循环
96     }
97 }
98
99 cout << "请输入姓名: " << endl;
100 cin >> name;
101
102 cout << "请输入密码: " << endl;
103 cin >> pwd;
104
105 //向文件中添加数据
106 ofs << id << " " << name << " " << pwd << " " << endl;
107 cout << "添加成功" << endl;
108
```

检测效果:



7.2.2.4 bug解决

bug描述:

- 虽然可以检测重复的账号，但是刚添加的账号由于没有更新到容器中，因此不会做进程
- 导致刚加入的账号的学生或者职工编号，再次添加时依然可以重复

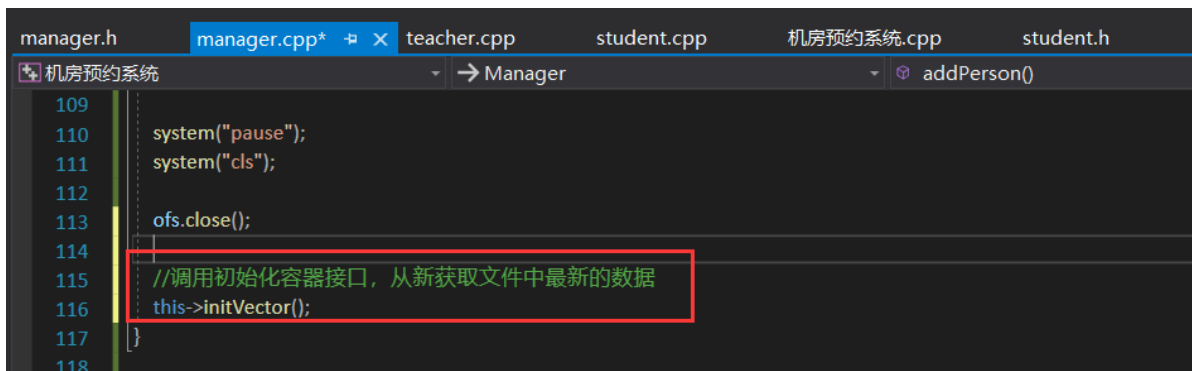
解决方案:

- 在每次添加新账号是，重新初始化容器

在void addPerson()函数中，添加完毕后，加入代码：

```
//调用初始化容器接口，从新获取文件中最新的数据  
this->initVector();
```

位置如图：



```
manager.h  manager.cpp  teacher.cpp  student.cpp  机房预约系统.cpp  student.h
机房预约系统  → Manager  addPerson()

46
47 string fileName; //操作文件名
48 string tip; //提示id号
49 string errorTip; //提示错误
50
51 ofstream ofs; //文件操作对象
52
53 int select = 0;
54 cin >> select; //接收用户的选择
55
56 while (select != 1 && select != 2)
57 {
58     cout << "输入有误, 请重新输入:" << endl;
59     cin >> select;
60 }
61
62 if (select == 1)
63 {
64     //添加的是学生
65     fileName = STUDENT_FILE;
66     tip = "请输入学号: ";
67     errorTip = "学号重复, 请重新输入";
68 }
69 else
70 {
71     //添加的是老师
72     fileName = TEACHER_FILE;
73     tip = "请输入职工号: ";
74     errorTip = "职工号重复, 请重新输入";
75 }
76
77 //利用追加的方式写文件
78 ofs.open(fileName, ios::out | ios::app);
79
80 int id; //学号或职工号
81 string name; //姓名
82 string pwd; //密码
83 cout << tip << endl;
84
85 while (true)
86 {
87     cin >> id;
88     bool res = checkRepeat(id, select);
89     if (res)
90     {
91         cout << errorTip << endl;
92     }
93     else
94     {
95         break; //退出循环
96     }
97 }
98
99 cout << "请输入姓名: " << endl;
100 cin >> name;
101
102 cout << "请输入密码: " << endl;
103 cin >> pwd;
104
105 //向文件中添加数据
106 ofs << id << " " << name << " " << pwd << " " << endl;
107 cout << "添加成功" << endl;
108
```

测试代码:



查看txt文档内容:



再次测试, 刚加入的账号不会重复添加了!

7.3 显示账号

功能描述: 显示学生信息或老师信息

7.3.1 显示功能实现

在Manager的showPerson成员函数中，实现显示账号功能，代码如下：

```
//查看账号
void Manager::showPerson()
{
    cout << "请选择查看内容: " << endl;
    cout << "1、查看所有学生" << endl;
    cout << "2、查看所有老师" << endl;

    int select = 0; //接收用户选择
    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);
    }
    else
    {
        //查看老师信息
        cout << "所有老师信息如下: " << endl;
        for_each(vTea.begin(), vTea.end(), printTeacher);
    }

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

7.3.2 测试

测试查看学生效果

```
F:\C++project\机房预约系统\机房预约系统\Debug\机房预约系统.exe
当前学生数量为: 2
当前老师数量为: 4
欢迎管理员: admin登录!

-----
1. 添加账号
2. 查看账号
3. 查看机房
4. 清空预约
0. 注销登录
-----

请选择您的操作:
2
查看账号
请选择查看内容:
1、查看所有学生
2、查看所有老师
1
所有学生信息如下:
学号: 1 姓名: 张三 密码: 333
学号: 2 姓名: 李四 密码: 444
请按任意键继续. . .
```

测试查看老师效果:

```
选择 F:\C++project\机房预约系统\机房预约系统\Debug\机房预约系统.exe
欢迎管理员: admin登录!

-----
1. 添加账号
2. 查看账号
3. 查看机房
4. 清空预约
0. 注销登录
-----

请选择您的操作:
2
查看账号
请选择查看内容:
1、查看所有学生
2、查看所有老师
2
所有老师信息如下:
职工号: 1 姓名: 老廖 密码: 666
职工号: 2 姓名: 老王 密码: 888
职工号: 3 姓名: 老刘 密码: 666
职工号: 4 姓名: 老李 密码: 444
请按任意键继续. . .
```

至此，显示账号功能实现完毕

7.4 查看机房

7.4.1 添加机房信息

案例需求中，机房一共有三个，其中1号机房容量20台机器，2号50台，3号100台

我们可以将信息录入到computerRoom.txt 中



7.4.2 机房创建

在头文件下，创建新的文件 computerRoom.h

并添加如下代码：

```
#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
};
```

7.4.3 初始化机房信息

在Manager管理员类下，添加机房的容器，用于保存机房信息

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

在Manager有参构造函数中，追加如下代码，初始化机房信息

```

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

ComputerRoom com;
while (ifs >> com.m_ComId && ifs >> com.m_MaxNum)
{
    vCom.push_back(com);
}

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

```

添加位置如下:

```

computerRoom.h  manager.h  manager.cpp  teacher.cpp  student.
+ 机房预约系统  -> Manager

7  }
8
9  //有参构造
10 Manager::Manager(string name, string pwd)
11 {
12     //初始化管理员信息
13     this->m_Name = name;
14     this->m_Pwd = pwd;
15
16     //初始化容器 获取到所有文件中 学生、老师信息
17     this->initVector();
18
19     //初始化机房信息
20     ifstream ifs;
21     ifs.open(COMPUTER_FILE, ios::in);
22
23     ComputerRoom com;
24     while (ifs >> com.m_ComId && ifs >> com.m_MaxNum)
25     {
26         vCom.push_back(com);
27     }
28     ifs.close();
29     cout << "机房的数量为: " << vCom.size() << endl;
30 }

```

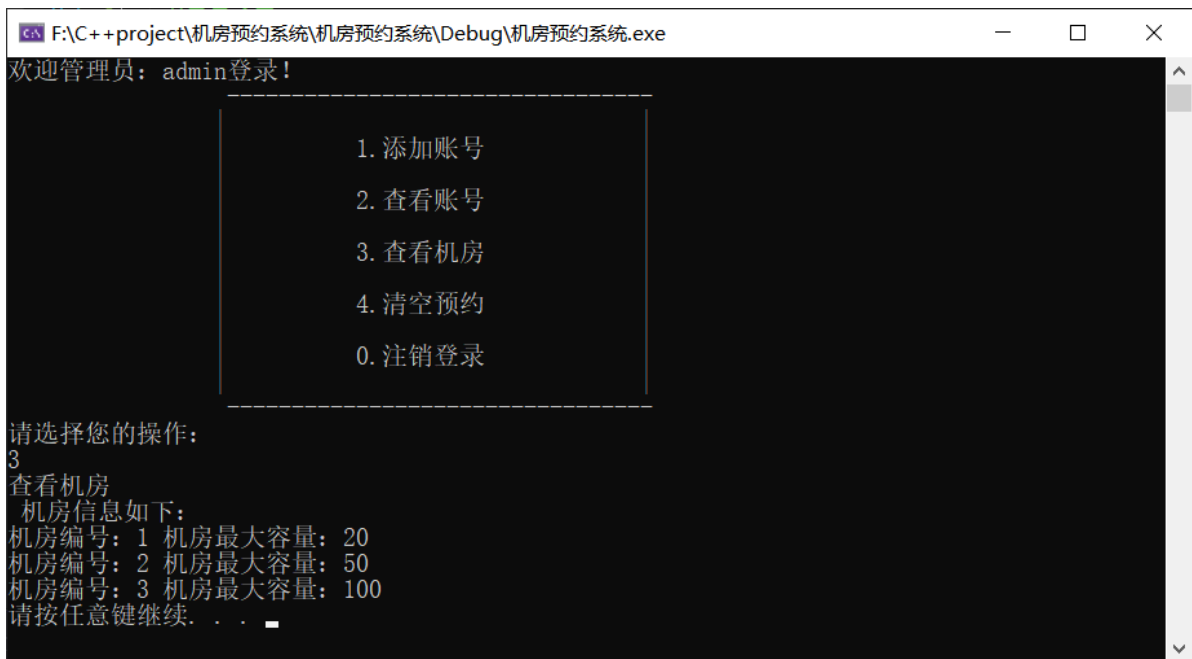
因为机房信息目前版本不会有所改动，如果后期有修改功能，最好封装到一个函数中，方便维护。

7.4.4 显示机房信息

在Manager类的showComputer成员函数中添加如下代码:

```
//查看机房信息
void Manager::showComputer()
{
    cout << "机房信息如下：" << 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");
}
```

测试显示机房信息功能：



7.5 清空预约

功能描述：

- 清空生成的order.txt 预约文件

7.5.1 清空功能实现

在Manager的 cleanFile () 成员函数中添加如下代码：

```
//清空预约记录
void Manager::clearFile()
{
    int select;
    cout << "是否确定要清空预约记录" << endl;
    cout << "1、确定" << " " << "2、取消" << endl;
    cin >> select;
    if (select == 1)
    {
        ofstream ofs(ORDER_FILE, ios::trunc);
        ofs.close();
    }
}
```



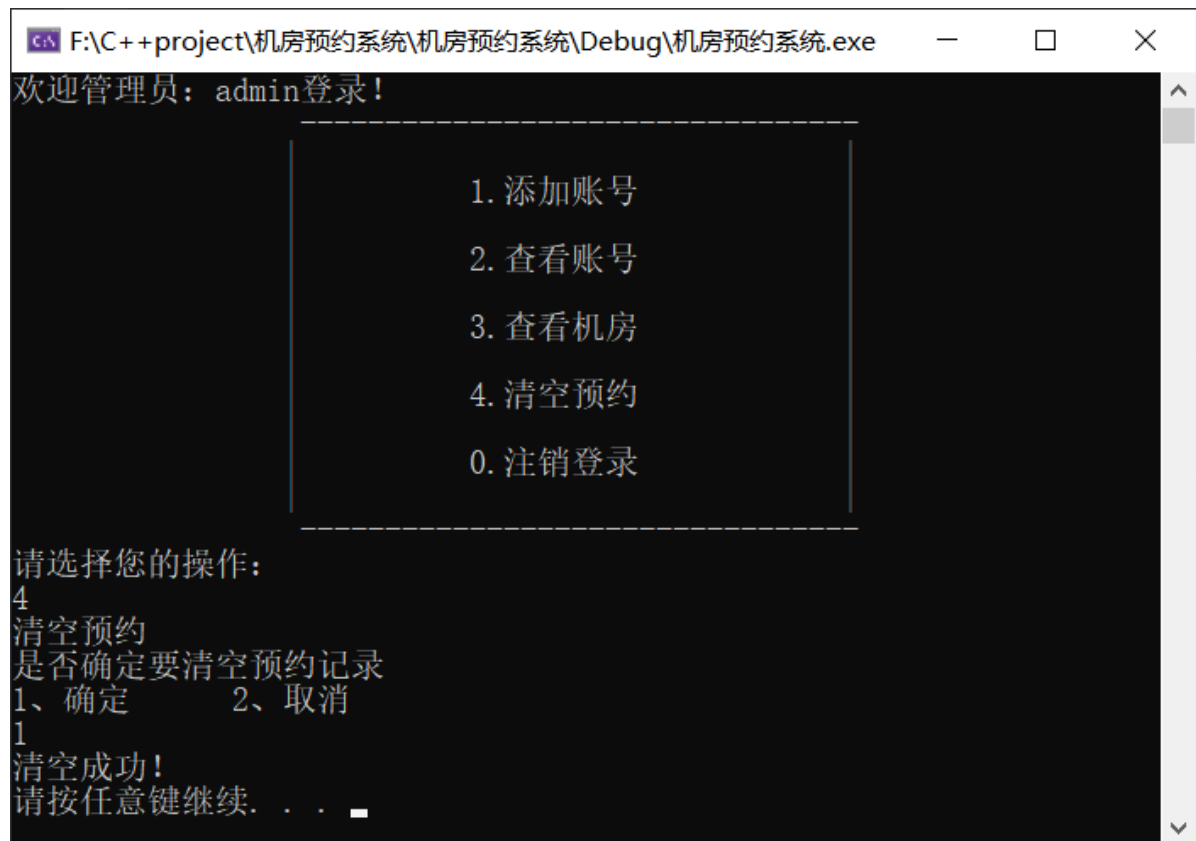
```

        cout << "清空成功!" << endl;
    }

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

```

测试清空，可以随意写入一些信息在 order.txt 中，然后调用cleanFile清空文件接口，查看是否清空干净。



8、学生模块

8.1 学生登录和注销

8.1.1 构造函数

- 在Student类的构造函数中，初始化学生信息，代码如下：

```

//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;
}

```

8.1.2 管理员子菜单

- 在机房预约系统.cpp中，当用户登录是的学生，添加学生菜单接口
- 将不同的分支提供出来
 - 申请预约
 - 查看我的预约
 - 查看所有预约
 - 取消预约
 - 注销登录
- 实现注销功能

添加全局函数 void studentMenu(Identity* &manager); 代码如下：

```
//进入学生子菜单界面
void studentMenu(Identity* &student)
{
    while (true)
    {
        //调用学生子菜单
        student->operMenu();
        Student* stu = (Student*)student;

        int select = 0;
        cin >> select;        //接收用户选择
        if (select == 1) //申请预约
        {
            stu->applyOrder();
        }
        else if (select == 2) //查看自身的预约
        {
            stu->showMyOrder();
        }
        else if (select == 3) //查看所有人的预约
        {
            stu->showAllOrder();
        }
        else if (select == 4) //取消预约
        {
            stu->cancelOrder();
        }
        else
        {
            //注销登录
            delete student;
            cout << "注销成功" << endl;
            system("pause");
            system("cls");
            return;
        }
    }
}
```

8.1.3 菜单功能实现

- 在实现成员函数 void Student::operMenu() 代码如下:

```
//菜单界面
void Student::operMenu()
{
    cout << "欢迎学生代表: " << 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;
}
```

8.1.4 接口对接

- 学生成功登录后, 调用学生的子菜单界面
- 在学生登录分支后, 添加代码

```
//进入学生身份的子菜单
studentMenu(person);
```

添加效果如图:

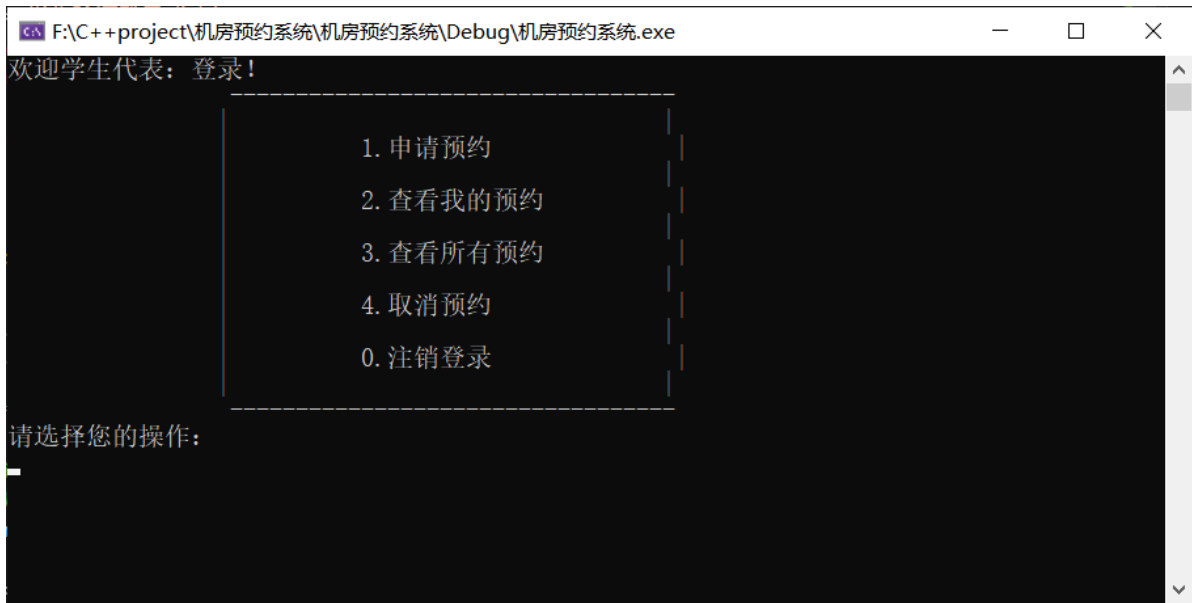
```
manager.h    manager.cpp    teacher.cpp    student.cpp*    机房预约系统.cpp    student.h
机房预约系统 (全局范围) Login(string fileName,
133      cout << " 请输入用户名: " << endl;
134      cin >> name;
135
136      cout << "请输入密码: " << endl;
137      cin >> pwd;
138
139      if (type == 1)
140      {
141          //学生身份验证
142          int fId; //从文件中读取的id号
143          string fName; //从文件中获取的姓名
144          string fPwd; //从文件中获取密码
145          while (ifs >> fId && ifs >> fName && ifs >> fPwd)
146          {
147              //与用户输入的信息做对比
148              if (fId == id && fName == name && fPwd == pwd)
149              {
150                  cout << "学生验证登录成功! " << endl;
151                  system("pause");
152                  system("cls");
153                  //创建一个学生示例对象
154                  person = new Student(id, name, pwd);
155
156                  //进入学生身份的子菜单
157                  studentMenu(person);
158                  return;
159              }
160          }
161      }
162  }
```

测试对接，效果如图：

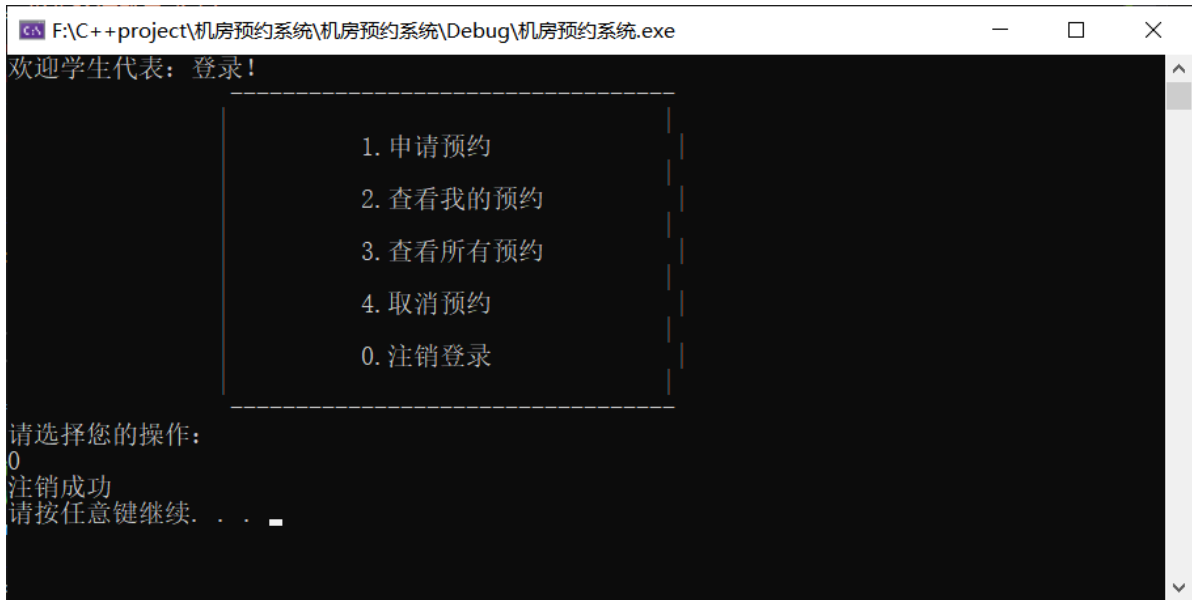
登录验证通过：

```
F:\C++\project\机房预约系统\机房预约系统\Debug\机房预约系统.exe
-----
1. 学生代表
2. 老  师
3. 管 理 员
0. 退  出
-----
输入您的选择: 1
请输入你的学号:
1
  请输入用户名:
张三
  请输入密码:
333
学生验证登录成功!
请按任意键继续. . .
```

学生子菜单：



注销登录：



8.2 申请预约

8.2.1 获取机房信息

- 在申请预约时，学生可以看到机房的信息，因此我们需要让学生获取到机房的信息

在student.h中添加新的成员函数如下：

```
//机房容器  
vector<ComputerRoom> vCom;
```

在学生的有参构造函数中追加如下代码：

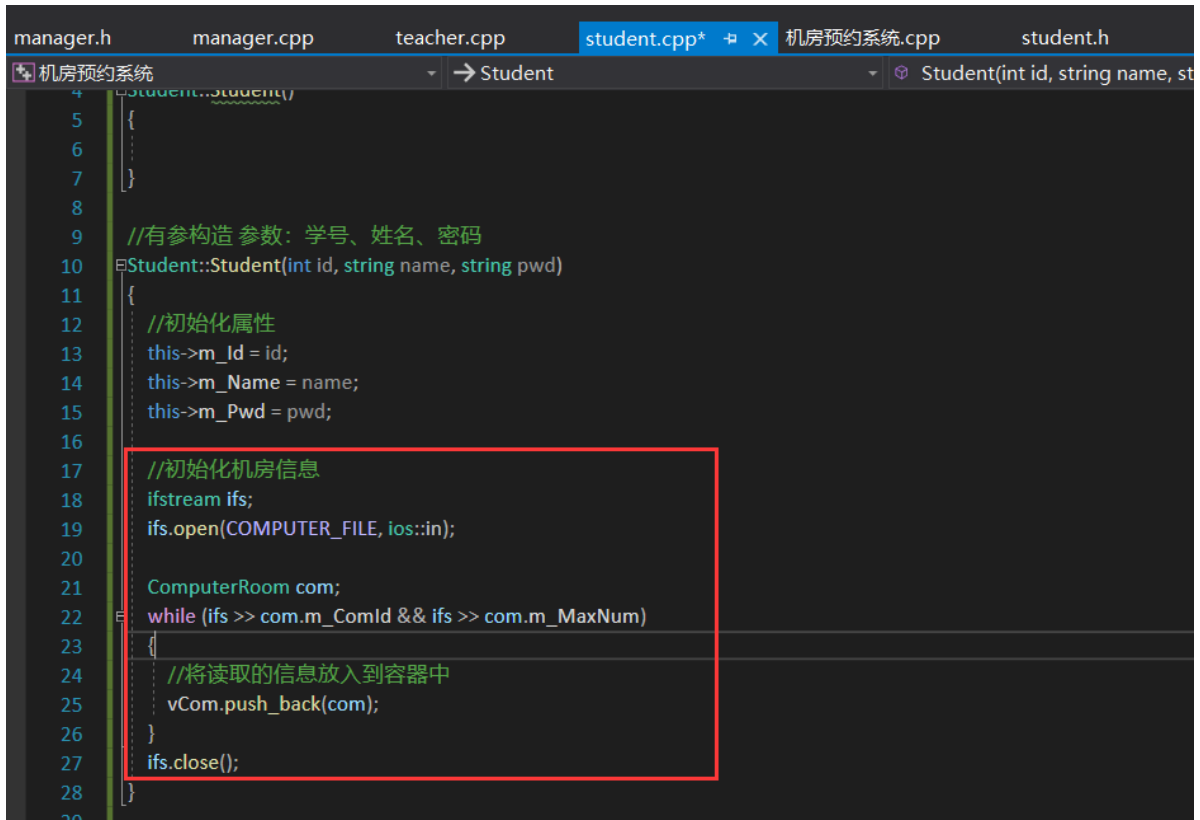
```

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

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();

```

追加位置如图：



至此，vCom容器中保存了所有机房的信息

8.2.2 预约功能实现

在student.cpp 中实现成员函数 void Student::applyOrder()

```

//申请预约
void Student::applyOrder()
{
    cout << "机房开发时间为周一至周五！" << 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)
    {
        break;
    }
    cout << "输入有误，请重新输入" << endl;
}

cout << "请输入申请预约时间段: " << endl;
cout << "1、上午" << endl;
cout << "2、下午" << endl;
cin >> interval;
while (interval != 1&&interval !=2)
{
    cout << "输入有误，请重新输入" << endl;
    cin >> interval;
}

cout << "请选择机房: " << endl;
for (int i = 0; i < vCom.size(); i++)
{
    cout << vCom[i].m_ComId << " 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;
ofs.open(ORDER_FILE, ios::app);

ofs << "date:" << date << " ";
ofs << "interval:" << interval << " ";
ofs << "stuId:" << this->m_Id << " ";
ofs << "sutName:" << this->m_Name << " ";
ofs << "roomId:" << room << " ";
ofs << "status:" << 1 << " "<<endl;

ofs.close();
system("pause");
system("cls");
}

```

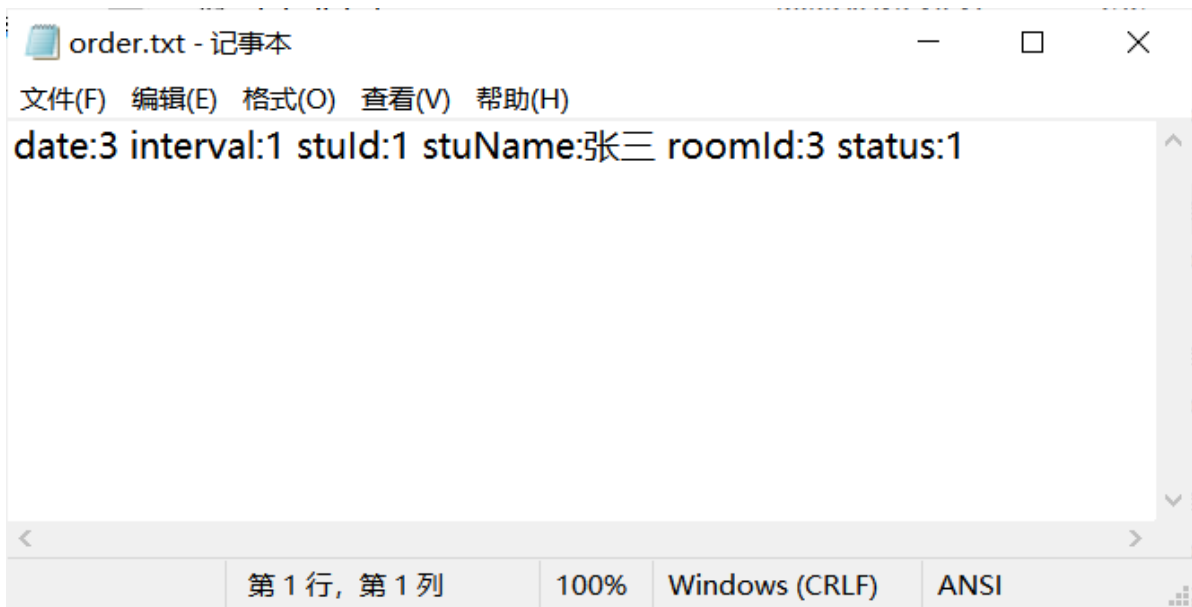
运行程序，测试代码：

```
选择 F:\C++project\机房预约系统\机房预约系统\Debug\机房预...
欢迎学生代表：张三登录！

1. 申请预约
2. 查看我的预约
3. 查看所有预约
4. 取消预约
0. 注销登录

请选择您的操作：
1
机房开发时间为周一至周五！
请输入申请预约的时间：
1、周一
2、周二
3、周三
4、周四
5、周五
6
输入有误，请重新输入
5
请输入申请预约时间段：
1、上午
2、下午
5
输入有误，请重新输入
2
请选择机房：
1 机房容量为：20
2 机房容量为：50
3 机房容量为：100
2
预约成功！审核中
```

在order.txt 文件中生成如下内容：



8.3 显示预约

8.3.1 创建预约类

功能描述：显示预约记录时，需要从文件中获取到所有记录，用来显示，创建预约的类来管理记录以及更新

在头文件以及源文件下分别创建**orderFile.h** 和 **orderFile.cpp**文件

orderFile.h中添加如下代码：

```
#pragma once
#include<iostream>
using namespace std;
#include"globalFile.h"
#include<fstream>
#include<map>
#include<string>
#include"orderFile.h"
#include<vector>

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

    //更新预约记录
    void updateOrder();

    //记录预约信息条数
    int m_Size;

    //记录所有预约信息的容器 key记录条数 value 具体记录每条是信息

};
```

构造函数中获取所有信息，并存放在容器中，添加如下代码：

```
//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; //记录预约条数

    while (ifs>>date && ifs>>interval && ifs>>stuId && ifs>>stuName &&
ifs>>roomId && ifs>>status)
    {
        //cout << date << endl;
        //cout << interval << endl;
        //cout << stuId << endl;
        //cout << stuName << endl;
        //cout << roomId << endl;
        //cout << status << endl;
        //cout << endl;

        //截取出key和value值
        string key;
        string value;
        map<string, string>m;

        //date:1111
        //日期
        int pos = date.find(":");
        if (pos !=-1)
        {
            key = date.substr(0, pos);
            value = date.substr(pos + 1, date.size() - pos - 1); //size = 9,
pos = 4 ,size - pos -1= 9-4-1=4

            //将截取出来信息放到map
            m.insert(make_pair(key, value));
        }

        //时间段
        pos = interval.find(":");
        if (pos != -1)
        {
            key = interval.substr(0, pos);
            value = interval.substr(pos + 1, interval.size() - pos - 1); //size
= 9, pos = 4 ,size - pos -1= 9-4-1=4

            //将截取出来信息放到map
            m.insert(make_pair(key, value));
        }
    }
}
```

```

//cout << "key = " << key << endl;
//cout << "value = " << value << endl;
//学生编号
pos = stuId.find(":");
if (pos != -1)
{
    key = stuId.substr(0, pos);
    value = stuId.substr(pos + 1, stuId.size() - pos - 1); //size = 9,
pos = 4 ,size - pos -1= 9-4-1=4

    //将截取出来信息放到map
    m.insert(make_pair(key, value));
}

//学生姓名
pos = stuName.find(":");
if (pos != -1)
{
    key = stuName.substr(0, pos);
    value = stuName.substr(pos + 1, stuName.size() - pos - 1); //size =
9, pos = 4 ,size - pos -1= 9-4-1=4

    //将截取出来信息放到map
    m.insert(make_pair(key, value));
}

//机房编号
pos = roomId.find(":");
if (pos != -1)
{
    key = roomId.substr(0, pos);
    value = roomId.substr(pos + 1, roomId.size() - pos - 1); //size =
9, pos = 4 ,size - pos -1= 9-4-1=4

    //将截取出来信息放到map
    m.insert(make_pair(key, value));
}

//预约状态
pos = status.find(":");
if (pos != -1)
{
    key = status.substr(0, pos);
    value = status.substr(pos + 1, status.size() - pos - 1); //size =
9, pos = 4 ,size - pos -1= 9-4-1=4

    //将截取出来信息放到map
    m.insert(make_pair(key, value));
}

//将小map容器放入到大的map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
        //for (map<int,map<string,string>>::iterator it=m_orderData.begin(); it !=
m_orderData.end(); it++)
        //{
        //    cout << "信息条数为:  = " << it->first << " value = " << endl;
        //    for (map<string, string>::iterator mit = (*it).second.begin(); mit !=
it->second.end(); mit++)
        //    {
        //        cout << " key = " << mit->first << " value = " << mit->second << "
";
        //    }
        //    cout << endl;
        //}
    }
}

```

更新预约记录的成员函数 updateOrder 代码如下：

```

//更新预约记录
void OrderFile::updateOrder()
{
    if (this->m_Size == 0)
    {
        return; //预约记录为0条，直接return
    }
    ofstream ofs(ORDER_FILE, ios::out | ios::trunc);
    for (int i = 0; i < this->m_Size; i++)
    {
        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();
}

```

8.3.2 显示自身预约

首先我们先添加几条预约记录，可以用程序添加或者直接修改 order.txt 文件

order.txt 文件内容如下：比如我们有三名同学分别产生了3条预约记录



```
order.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
date:3 interval:1 stuld:1 stuName:张三 roomId:3 status:1
date:1 interval:1 stuld:1 stuName:张三 roomId:2 status:1
date:3 interval:2 stuld:1 stuName:张三 roomId:3 status:1
date:4 interval:1 stuld:2 stuName:李四 roomId:1 status:1
第 1 行, 第 1 列 100% Windows (CRLF) ANSI
```

在Student类的 void Student::showMyorder() 成员函数中，添加如下代码：

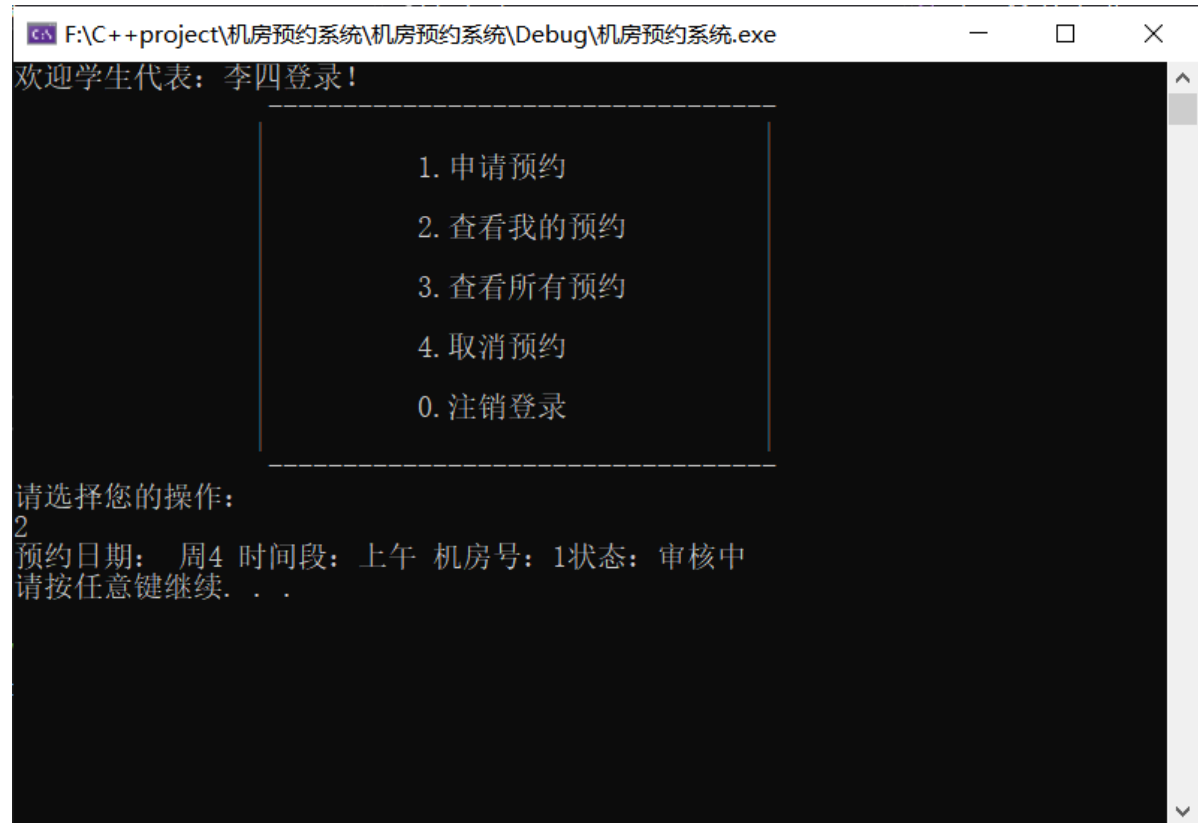
```
//查看自身预约
void Student::showMyOrder()
{
    OrderFile of;
    if (of.m_Size == 0)
    {
        cout << "无预约记录！" << endl;
        system("pause");
        system("cls");
        return;
    }

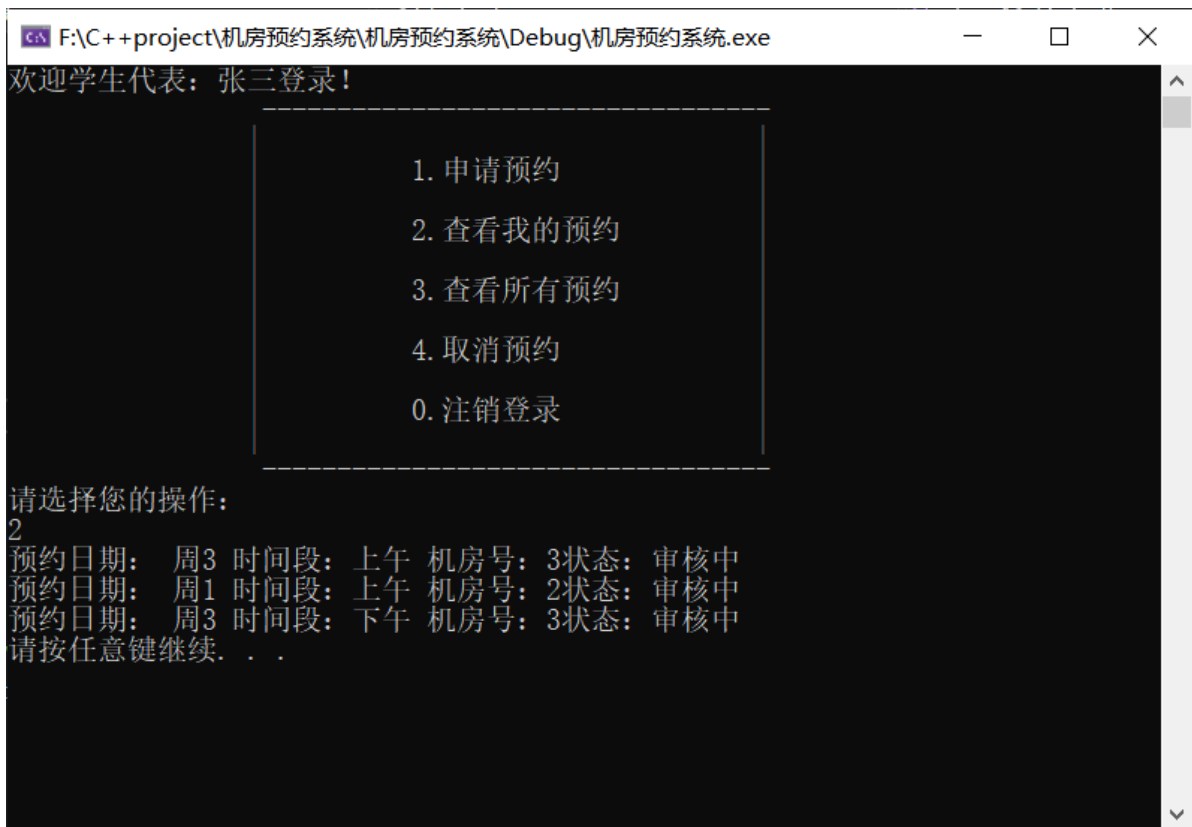
    for (int i = 0; i < of.m_Size; i++)
    {
        //string 转int
        //string 利用 .c_str() 转 const char *
        //利用atoi (const char *) 转 int
        if (this->m_Id == atoi(of.m_orderData[i][ "stuId" ].c_str())) //找到自身
        预约
        {
            cout << "预约日期: 周" << of.m_orderData[i]["date"];
            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");  
}
```

测试效果：





8.3.3 显示所有预约

在Student类的 void Student::showAllOrder() 成员函数中，添加如下代码：

```
//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: 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 << " 机房编号: " << 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");
}

```

测试效果如图：

8.4 取消预约

在Student类的 `void Student::cancelOrder()` 成员函数中，添加如下代码：

```

//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可以取消，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 << " 机房编号: " << of.m_orderData[i]["roomId"];
            string status = " 状态: ";
            if (of.m_orderData[i]["status"] == "1")
            {
                status += "审核中";
            }
            else if (of.m_orderData[i]["status"] == "2")
            {
                status += "预约成功";
            }
            cout << status << endl;
        }
    }
}

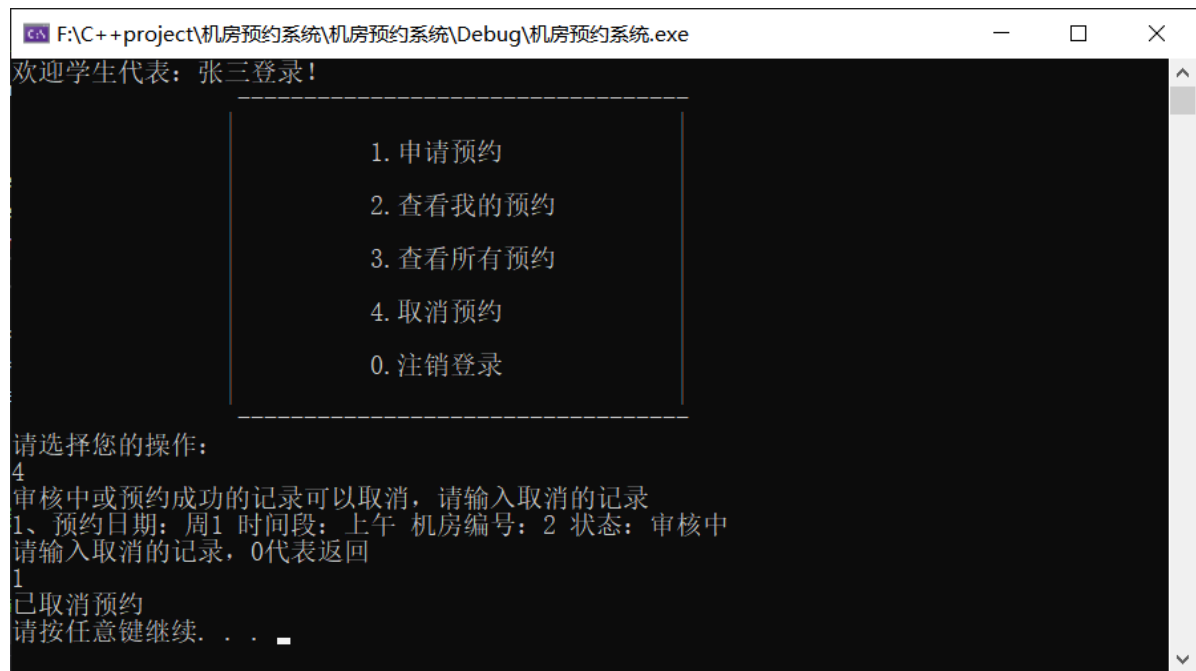
cout << "请输入取消的记录, 0代表返回" << endl;
int select = 0;
while (true)
{
    cin >> select;

    if (select >= 0&&select <= v.size())
    {
        if (select == 0)
        {
            break;
        }
        else
        {
            of.m_orderData[v[select - 1]]["status"] = "0";

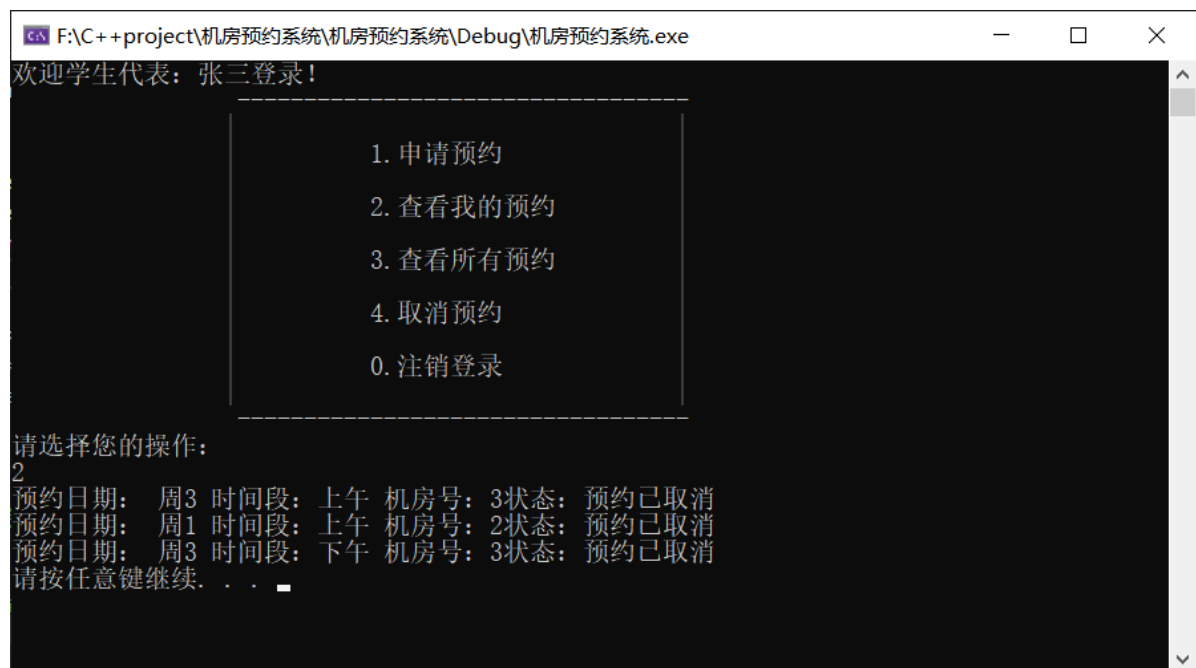
            of.updateOrder(); //更新文件信息
            cout << "已取消预约" << endl;
            break;
        }
    }
    cout << "输入有误, 请重新输入" << endl;
}
system("pause");
system("cls");
}

```

测试取消预约：



再次查看个人预约记录：



查看order.txt 预约文件



至此，学生模块功能全部实现

9、教师模块

9.1 教师登录和注销

9.1.1 构造函数

- Teacher类的构造函数中，初始化教师信息，代码如下：

```
//有参构造(职工编号，姓名，密码)
Teacher::Teacher(int t_id, string name, string pwd)
{
    //初始化
    this->t_Id = t_id;
    this->m_Name = name;
    this->m_Pwd = pwd;
}
```

9.1.2 教师子菜单

- 在机房预约系统.cpp中，当用户登录的是教师，添加教师菜单接口
- 将不同的分支提供出来
 - 查看所有预约
 - 审核预约
 - 注销登录
- 实现注销功能

添加全局函数 `void TeacherMenu(Person * &manager)` 代码如下：

```
//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; //接收用户选择
cin >> select;
if (select == 1) //查看所有预约
{
    tea->showAllOrder();
}
else if (select == 2)
{
    tea->validOrder();
}
else
{
    delete teacher;
    cout << "注销成功" << endl;
    system("pause");
    system("cls");
    return;
}
}
}

```

9.1.3 菜单功能实现

- 在实现成员函数 `void Teacher::operMenu()` 代码如下:

```

//教师菜单界面
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;
}

```

9.1.4 接口对接

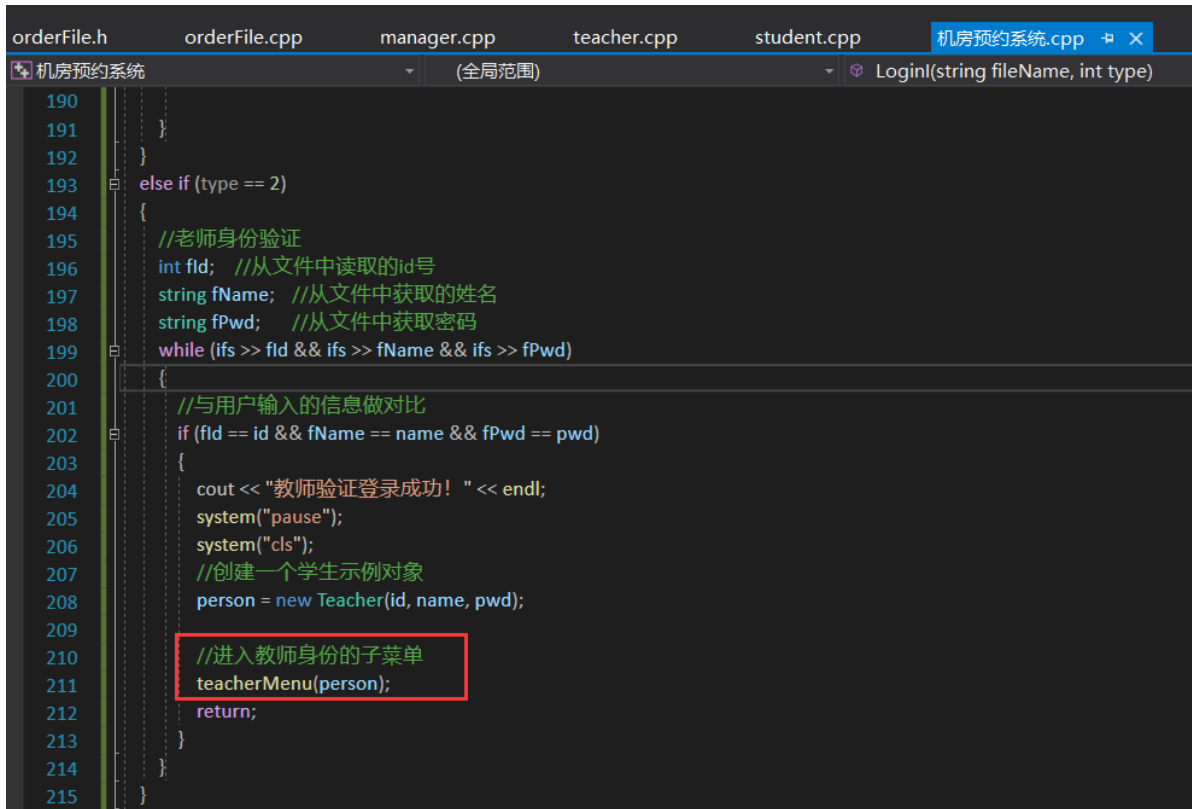
- 教师成功登录后，调用教师的子菜单界面
- 在教师登录分支中，添加代码：

```

//进入教师子菜单
TeacherMenu(person);

```

添加位置如下：



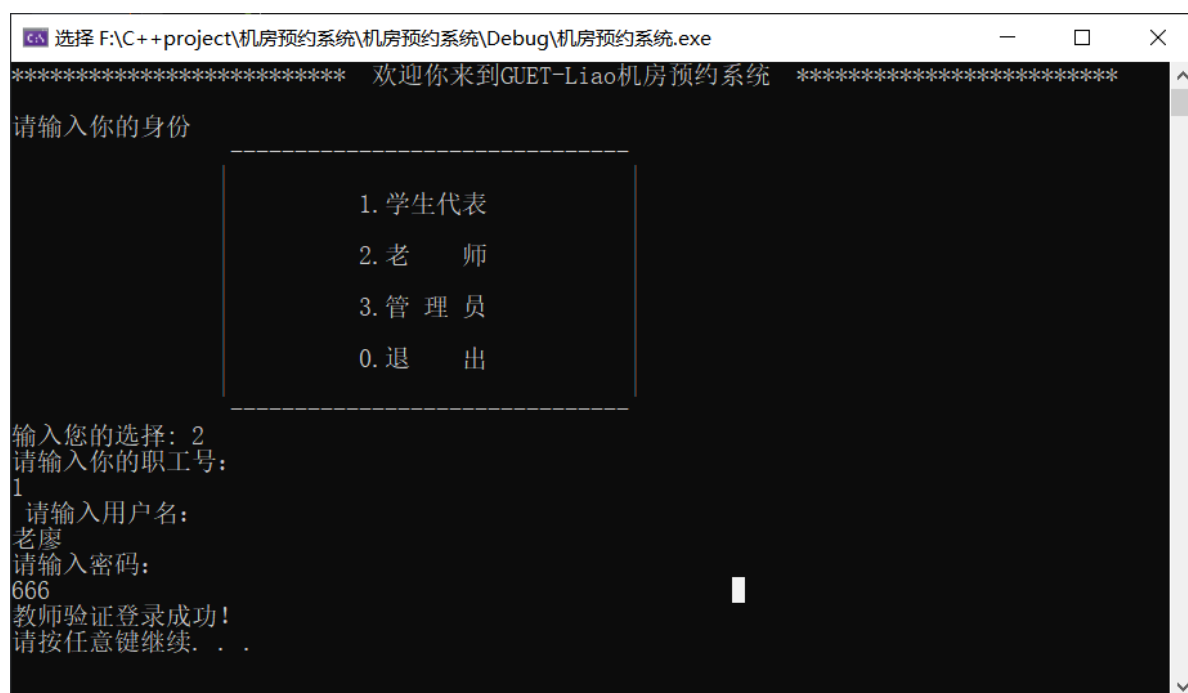
```
190 }
191 }
192 }
193 else if (type == 2)
194 {
195     //老师身份验证
196     int fId; //从文件中读取的id号
197     string fName; //从文件中获取的姓名
198     string fPwd; //从文件中获取密码
199     while (ifs >> fId && ifs >> fName && ifs >> fPwd)
200     {
201         //与用户输入的信息做对比
202         if (fId == id && fName == name && fPwd == pwd)
203         {
204             cout << "教师验证登录成功! " << endl;
205             system("pause");
206             system("cls");
207             //创建一个学生示例对象
208             person = new Teacher(id, name, pwd);
209
210             //进入教师身份的子菜单
211             teacherMenu(person);
212             return;
213         }
214     }
215 }
```

查看teacher.txt信息：

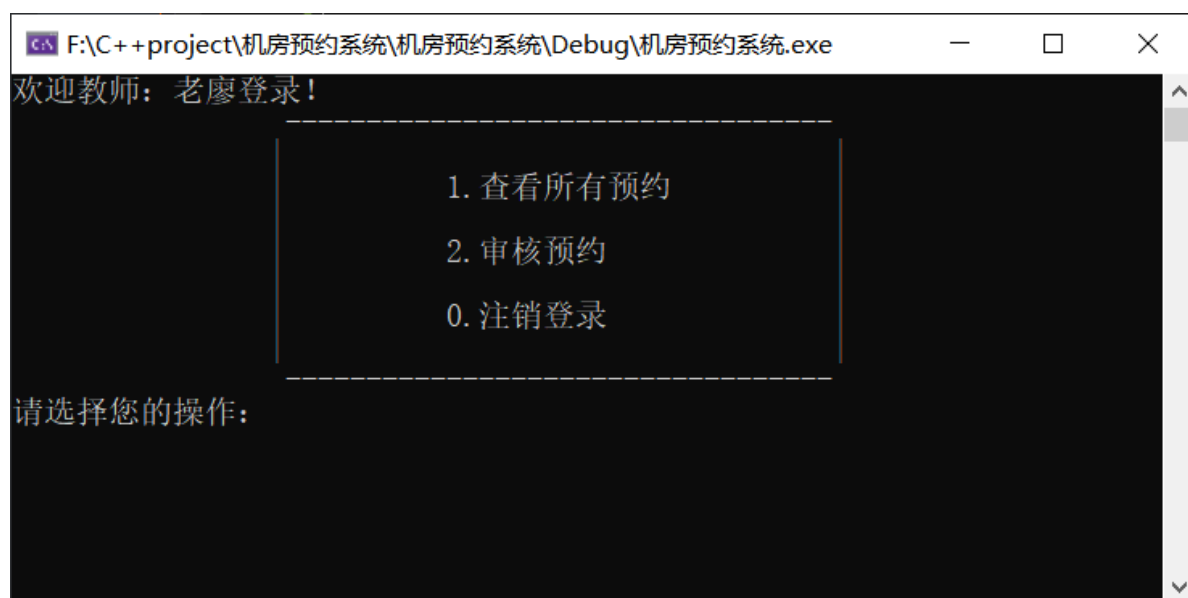


```
1 老廖 666
2 老王 888
3 老刘 666
4 老李 444
```

测试对接，效果图如下：



进入教师子菜单:



注销登录:



9.2 查看所有预约

9.2.1 所有预约功能实现

该功能与学生身份的查看所有预约功能相似，用于显示所有预约记录

在Teacher.cpp中实现成员函数 `void Teacher::showAllOrder()`

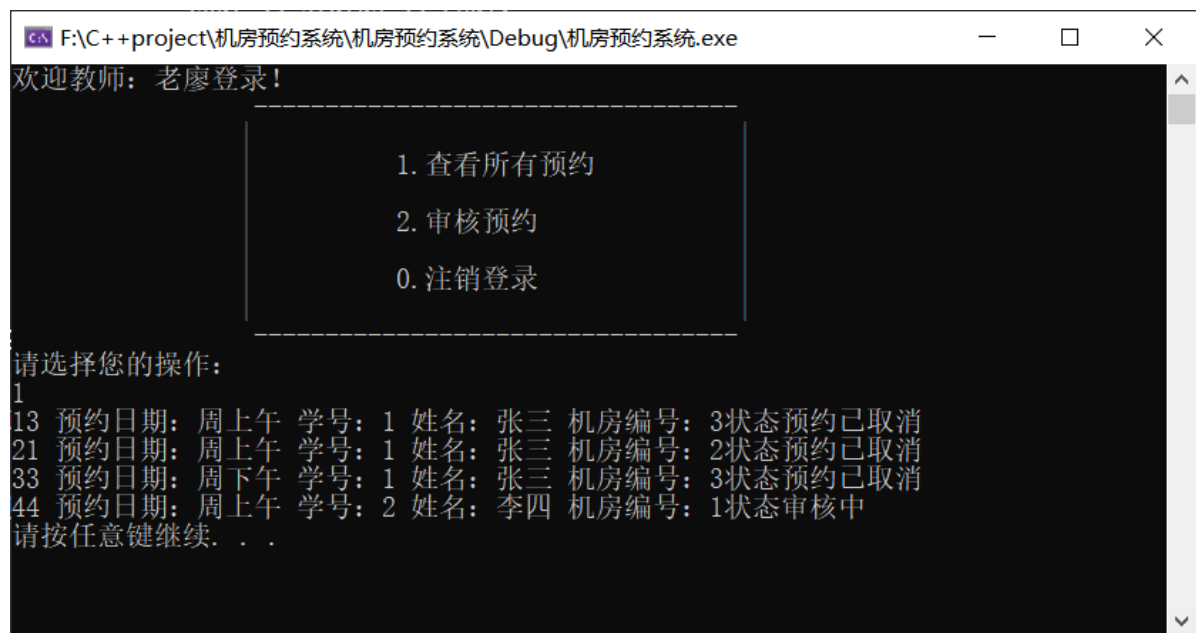
```
//查看所有预约
void Teacher::showAllOrder()
{
    OrderFile of;
    if (of.m_Size == 0)
    {
        cout << "无预约记录" << endl;
        system("pause");
        system("cls");
        return;
    }

    for (int i = 0; i < of.m_Size ; i++)
    {
        cout << i + 1 << of.m_orderData[i]["date"];
        cout << " 预约日期: 周" << (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 << " 机房编号: " << 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");
}
```

9.2.2 测试功能

运行测试教师身份的查看所有预约功能

测试效果如图：



9.3 审核预约

9.3.1 审核功能实现

功能描述：教师审核学生的预约，依据实际情况审核预约

在Teacher.cpp中实现成员函数 `void Teacher::validOrder()`

代码如下：

```
//审核预约
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; //接收用户的选择的预约记录
int ret = 0;    //接收预约结果记录

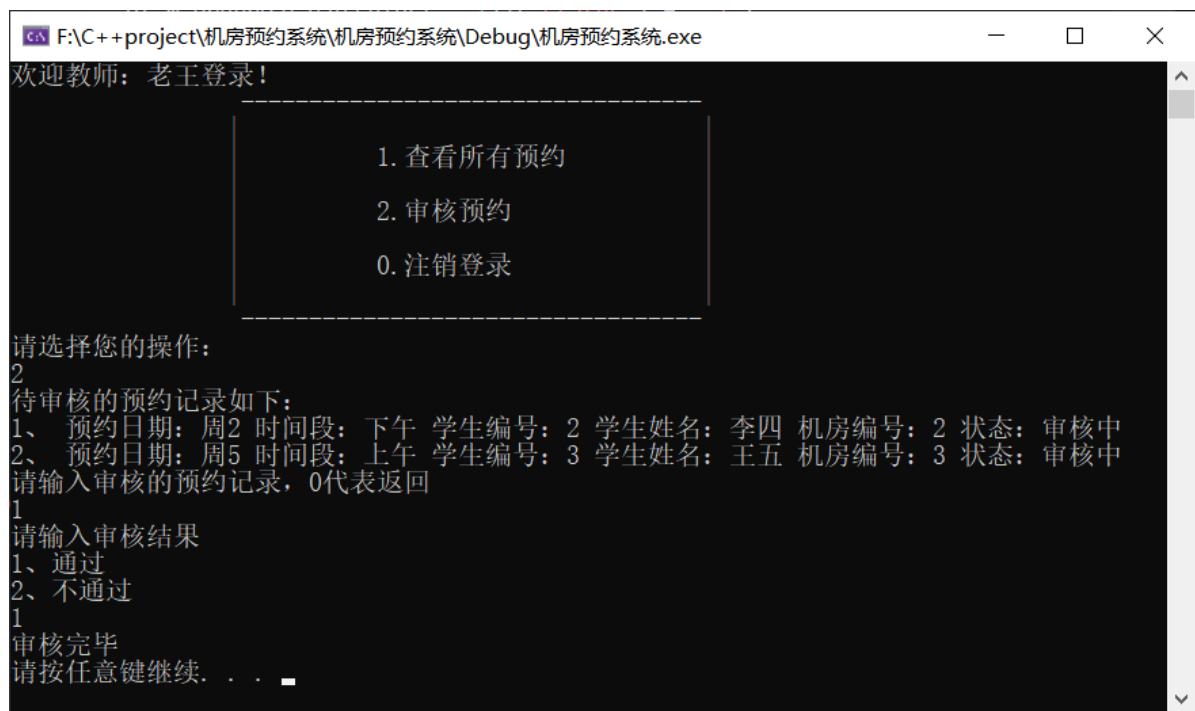
while (true)
{
    cin >> select;
    if (select >= 0 && select <= v.size())
    {
        if(select == 0)
        {
            break;
        }
        else
        {
            cout << "请输入审核结果" << endl;
            cout << "1、通过" << endl;
            cout << "2、不通过" << endl;
            cin >> ret;

            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";
            }

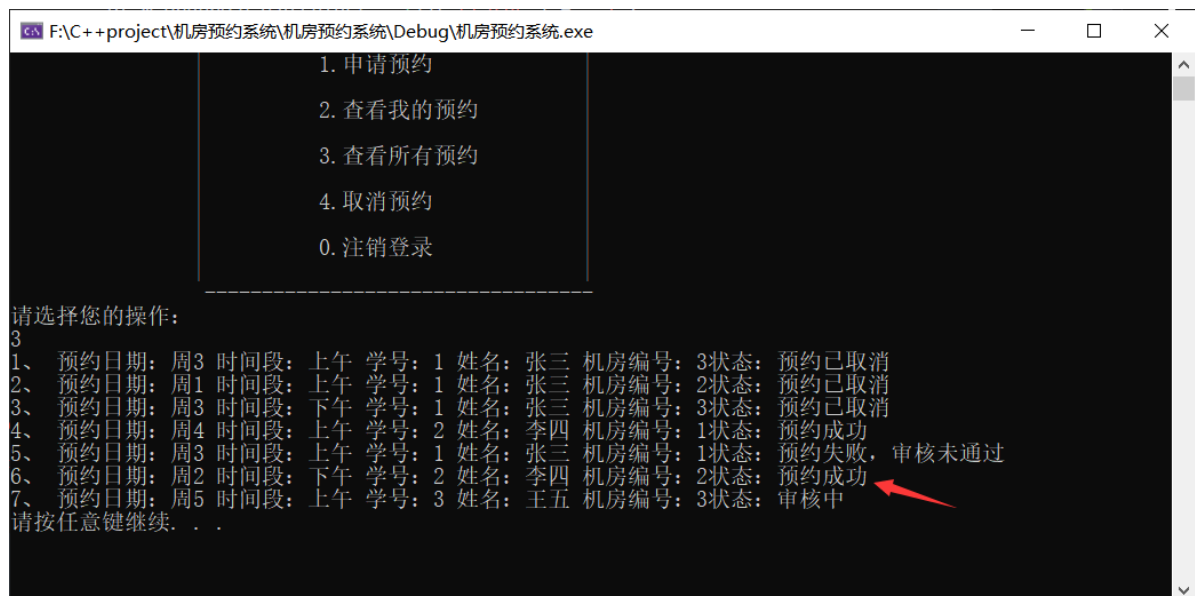
            //更新预约记录信息
            of.updateOrder();
            cout << "审核完毕" << endl;
            break;
        }
    }
    cout << "输入有误, 请重新输入" << endl;
}
system("pause");
system("cls");
}

```

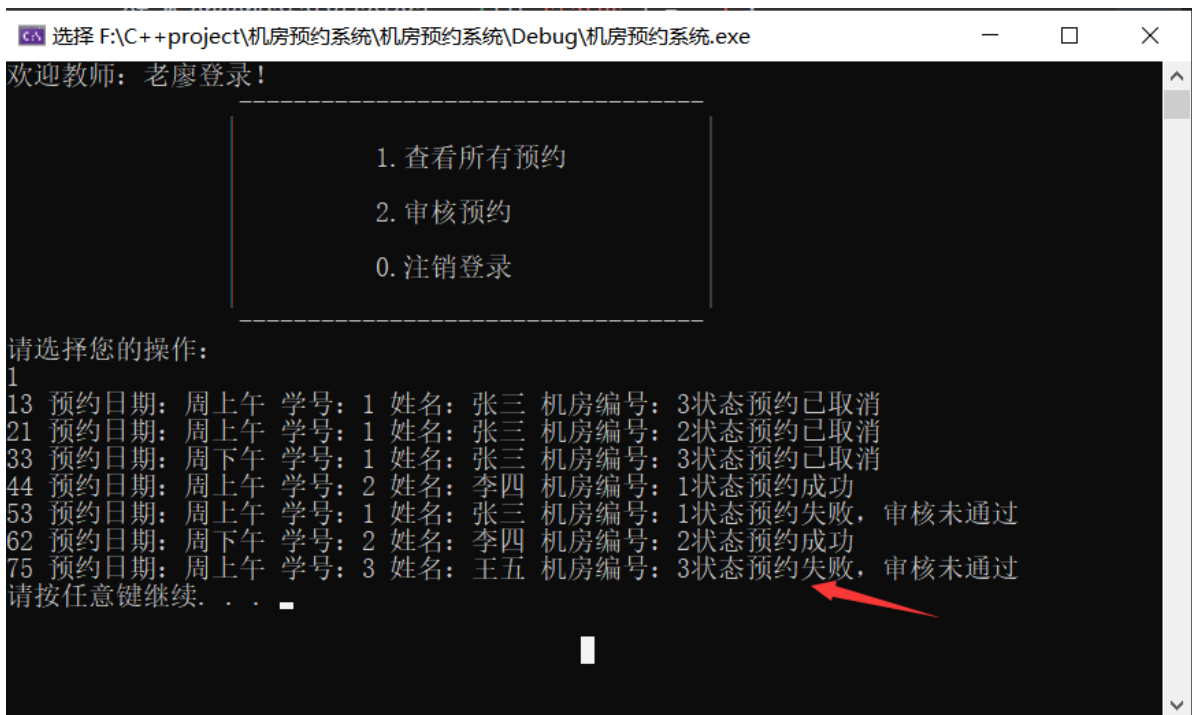
测试, 审核通过



学生身份下查看预约情况：



审核不通过情况：



本项目案例到此结束

加入集成电路QQ群