

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，解决这个问题。



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
- **教师**：审核学生的预约申请
- **管理员**：给学生、教师创建账号

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

- 申请的订单每周由管理员负责清空。
- 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，依据实际情况审核预约通过或者不通过

1.5 系统具体需求

- 首先进入登录界面，可选登录身份有：
 - 学生代表
 - 老师
 - 管理员
 - 退出
- 每个身份都需要进行验证后，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
- 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取消
 - 注销登录 --- 退出登录
- 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
- 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
 - 清空预约 --- 清空所有预约记录
 - 注销登录 --- 退出登录



2、创建项目

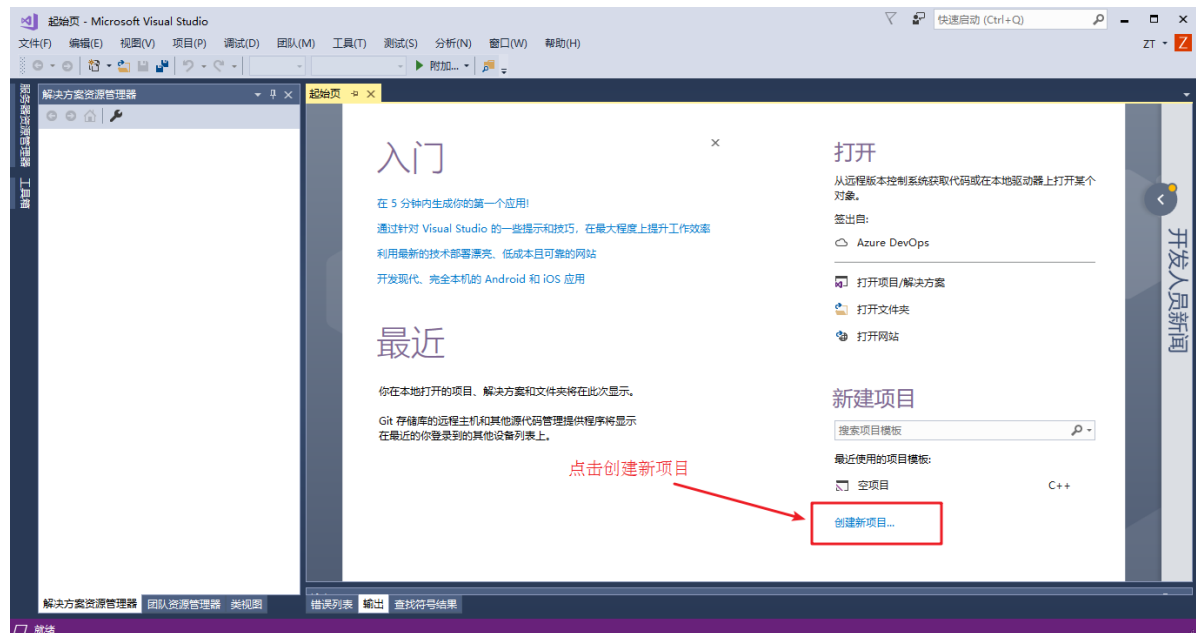
创建项目步骤如下：

- 创建新项目
- 添加文件

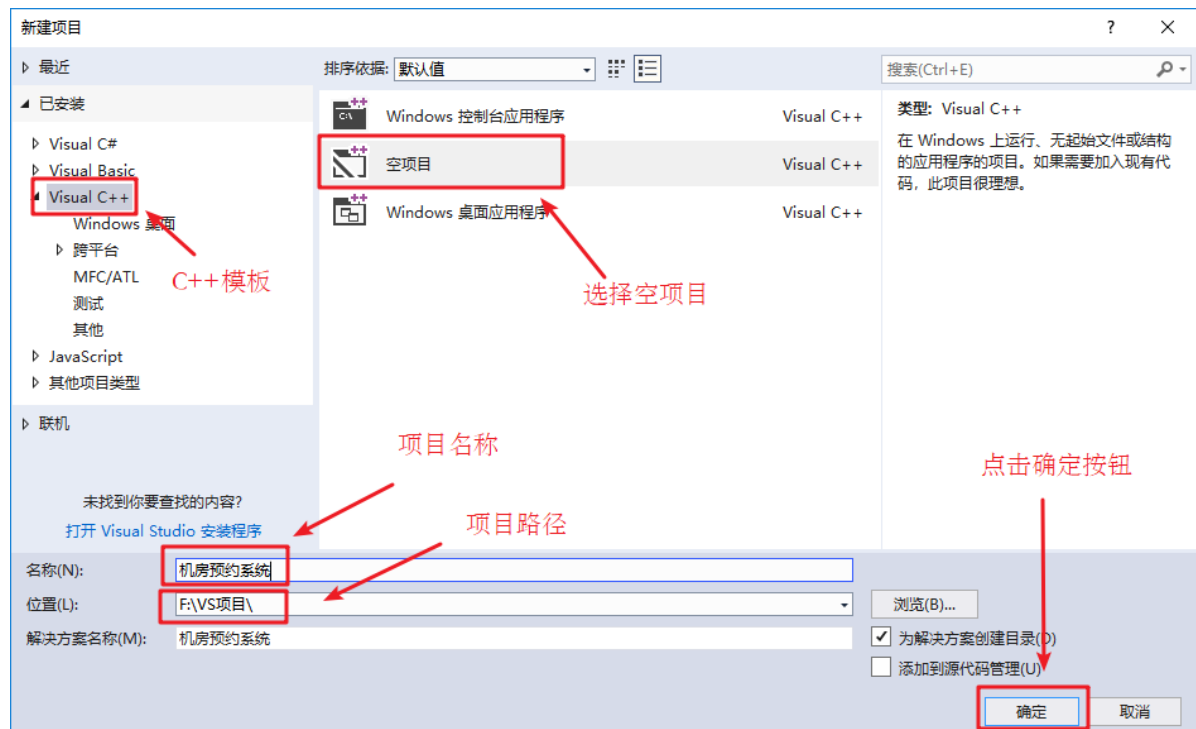
2.1 创建项目

- 打开vs2017后，点击创建新项目，创建新的C++项目

如图：

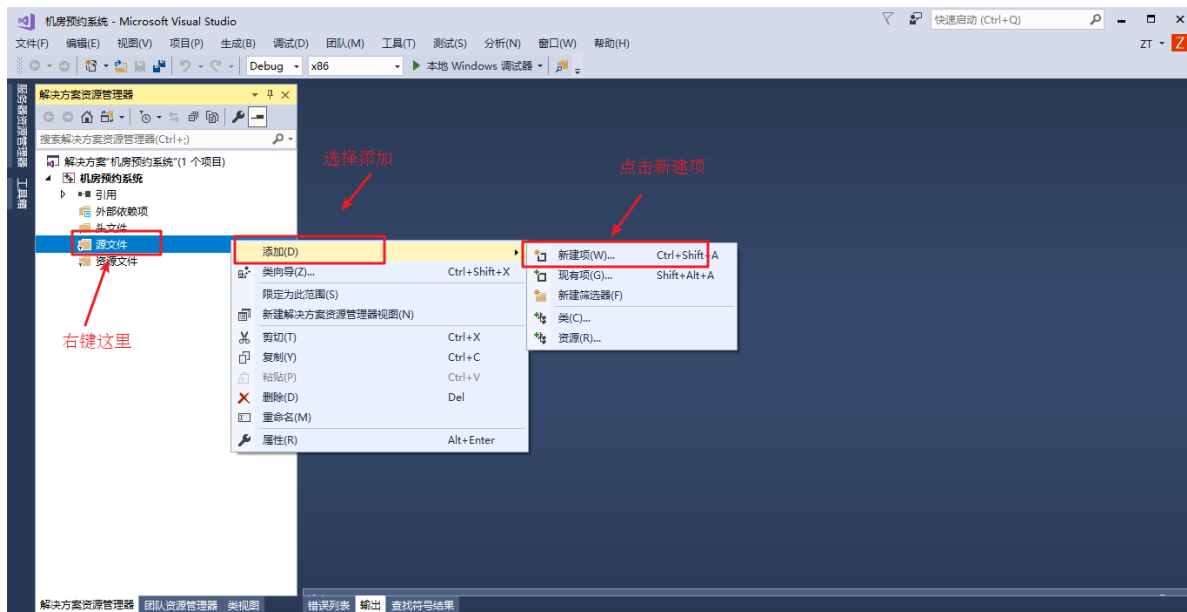


- 填写项目名称以及选取项目路径，点击确定生成项目

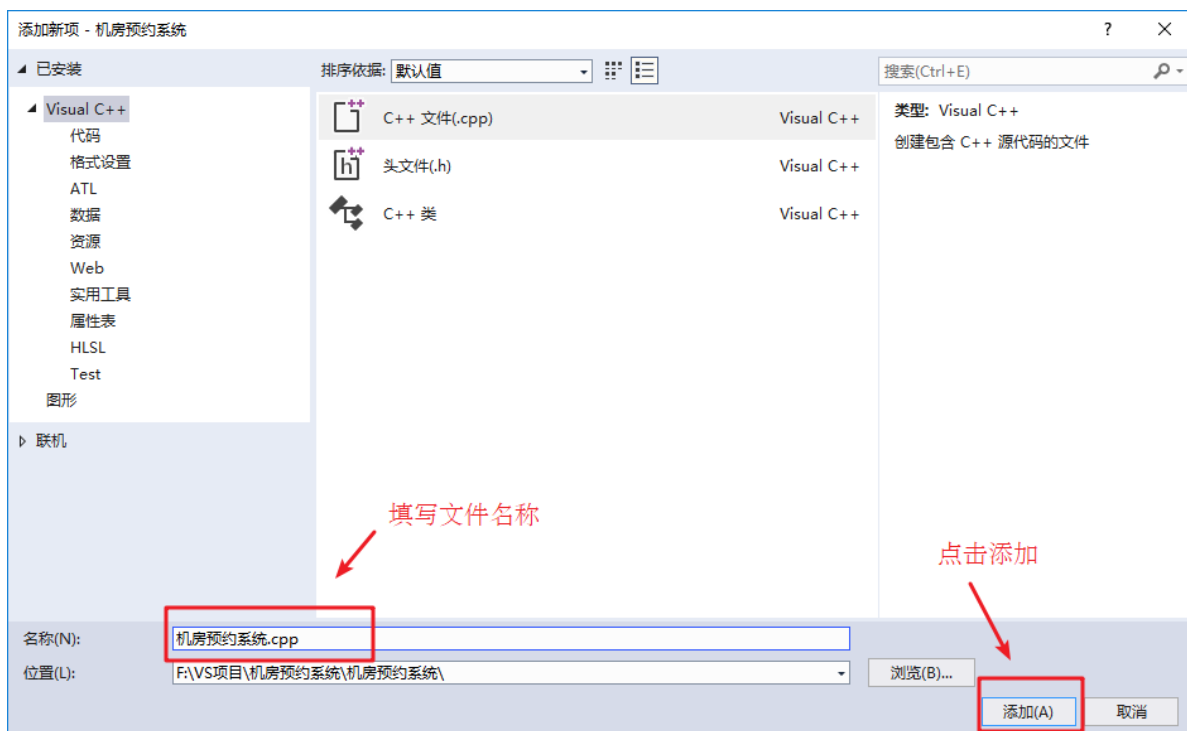


2.2 添加文件

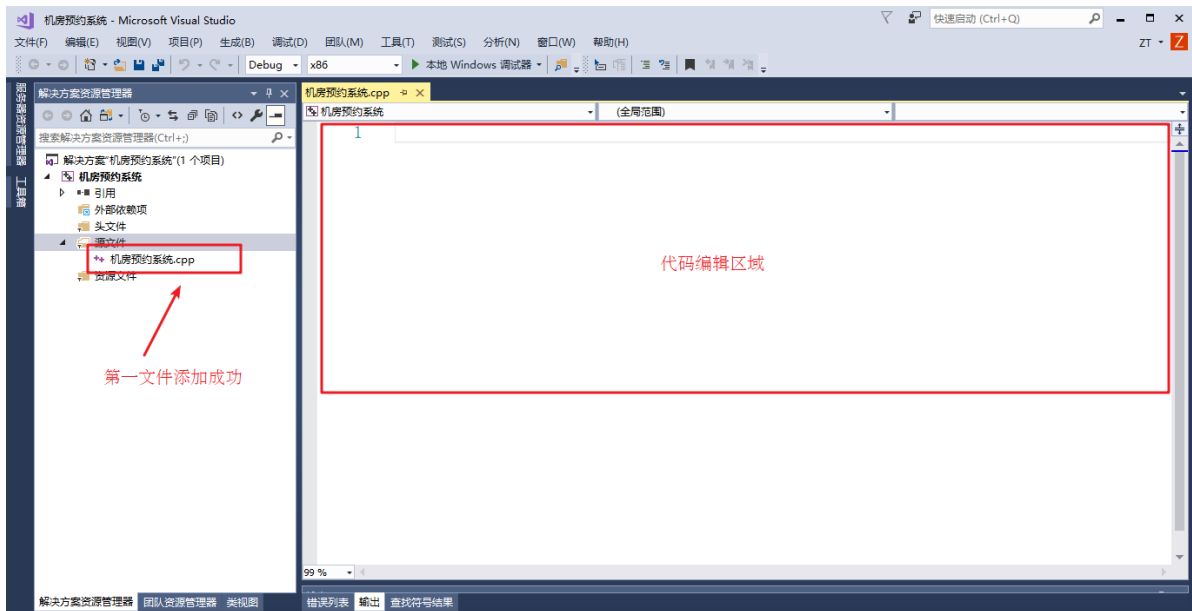
- 右键源文件，进行添加文件操作



- 填写文件名称，点击添加



- 生成文件成功，效果如下图



3、创建主菜单

功能描述：

- 设计主菜单，与用户进行交互

3.1 菜单实现

- 在主函数main中添加菜单提示，代码如下：

```
1  int main() {
2
3      cout << "===== 欢迎来到传智播客机房预约系统
===== "
4          << endl;
5      cout << endl << "请输入您的身份" << endl;
6      cout << "\t\t -----\n";
7      cout << "\t\t|                               |\n";
8      cout << "\t\t|          1.学生代表          |\n";
9      cout << "\t\t|                               |\n";
10     cout << "\t\t|          2.老    师          |\n";
11     cout << "\t\t|                               |\n";
12     cout << "\t\t|          3.管 理 员          |\n";
13     cout << "\t\t|                               |\n";
14     cout << "\t\t|          0.退    出          |\n";
15     cout << "\t\t|                               |\n";
16     cout << "\t\t -----\n";
17     cout << "输入您的选择： ";
18
19     system("pause");
20
21     return 0;
22 }
```

运行效果如图：



3.2 搭建接口

- 接受用户的选择，搭建接口
- 在main中添加代码

```
1  int main() {
2
3      int select = 0;
4
5      while (true)
6      {
7
8          cout << "===== 欢迎来到传智播客机房预约系统
===== " << endl;
9          cout << endl << "请输入您的身份" << endl;
10         cout << "\t\t ----- \n";
11         cout << "\t\t | \n";
12         cout << "\t\t |      1. 学生代表      | \n";
13         cout << "\t\t | \n";
14         cout << "\t\t |      2. 老    师      | \n";
15         cout << "\t\t | \n";
16         cout << "\t\t |      3. 管 理 员      | \n";
17         cout << "\t\t | \n";
18         cout << "\t\t |      0. 退    出      | \n";
19         cout << "\t\t | \n";
20         cout << "\t\t ----- \n";
21         cout << "输入您的选择: ";
22
23         cin >> select; //接受用户选择
24
25         switch (select)
26         {
27             case 1: //学生身份
28                 break;
29             case 2: //老师身份
30                 break;
31             case 3: //管理员身份
```

```

32         break;
33     case 0: //退出系统
34         break;
35     default:
36         cout << "输入有误，请重新选择！" << endl;
37         system("pause");
38         system("cls");
39         break;
40     }
41 }
42 }
43 system("pause");
44 return 0;
45 }

```

测试，输入0、1、2、3会重新回到界面，输入其他提示输入有误，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：

```

1     cout << "欢迎下一次使用"<<endl;
2     system("pause");
3     return 0;

```

```

switch (select)
{
case 1: //学生身份
    break;
case 2: //老师身份
    break;
case 3: //管理员身份
    break;
case 0: //退出系统
    cout << "欢迎下一次使用" << endl;
    system("pause");
    return 0;
    break;
default:
    cout << "输入有误，请重新选择！" << endl;
    system("pause");
    system("cls");
    break;
}

```

4.2 测试退出功能

运行程序，效果如图：



至此，退出程序功能实现

5、创建身份类

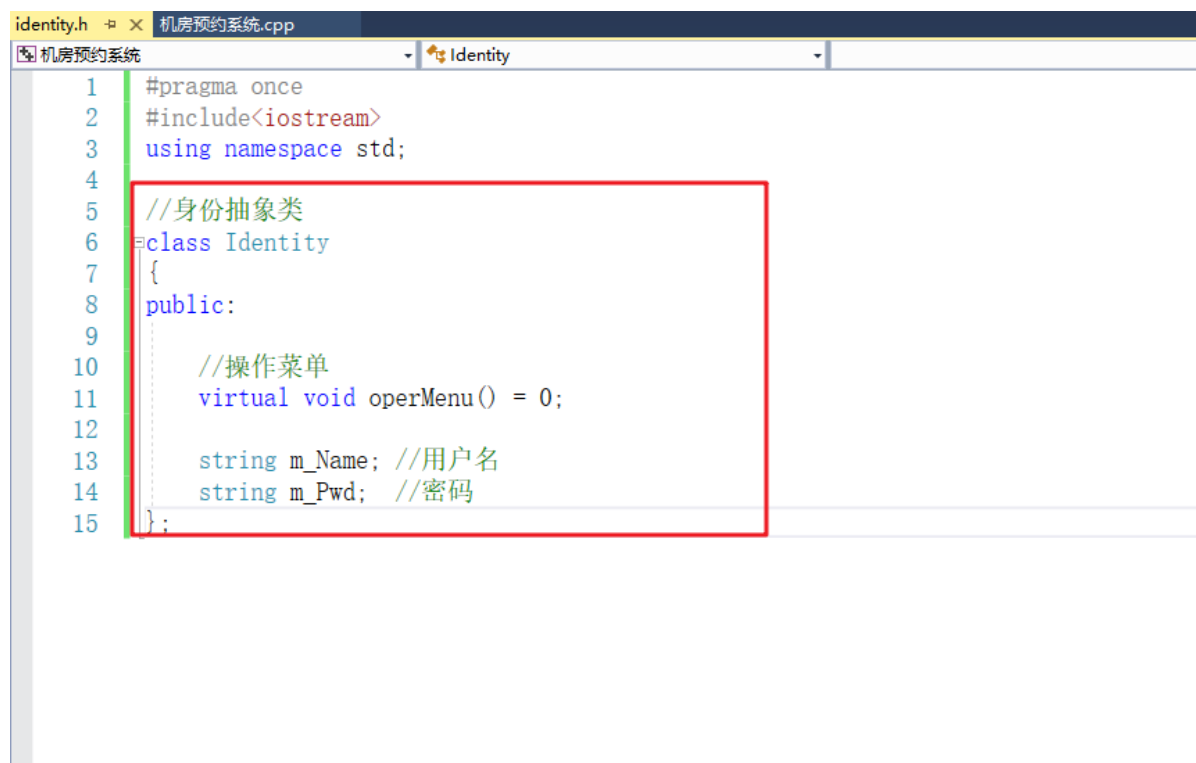
5.1 身份的基类

- 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
- 三种身份有其共性也有其特性，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中添加如下代码：

```
1  #pragma once
2  #include<iostream>
3  using namespace std;
4
5  //身份抽象类
6  class Identity
7  {
8  public:
9
10     //操作菜单
11     virtual void operMenu() = 0;
12
13     string m_Name; //用户名
14     string m_Pwd;  //密码
15 };
```

效果如图：



The screenshot shows a code editor with two tabs: 'identity.h' and '机房预约系统.cpp'. The 'Identity' class definition in 'identity.h' is highlighted with a red box. The code is as follows:

```
1  #pragma once
2  #include<iostream>
3  using namespace std;
4
5  //身份抽象类
6  class Identity
7  {
8  public:
9
10     //操作菜单
11     virtual void operMenu() = 0;
12
13     string m_Name; //用户名
14     string m_Pwd;  //密码
15 };
```

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：

```
1  #pragma once
2  #include<iostream>
3  using namespace std;
4  #include "identity.h"
5
6  //学生类
7  class Student :public Identity
8  {
9  public:
10     //默认构造
11     Student();
12
13     //有参构造(学号、姓名、密码)
14     Student(int id, string name, string pwd);
15
16     //菜单界面
17     virtual void operMenu();
18
19     //申请预约
20     void applyOrder();
21
22     //查看我的预约
23     void showMyOrder();
24
25     //查看所有预约
26     void showAllOrder();
27
28     //取消预约
29     void cancelOrder();
30
31     //学生学号
32     int m_Id;
33
34 };
```

student.cpp中添加如下代码：

```
1  #include "student.h"
2
3  //默认构造
4  Student::Student()
5  {
6  }
7
8  //有参构造(学号、姓名、密码)
9  Student::Student(int id, string name, string pwd)
10 {
11 }
12
13 //菜单界面
14 void Student::operMenu()
15 {
16 }
17
18 //申请预约
19 void Student::applyOrder()
20 {
21 }
22 }
23
24 //查看我的预约
25 void Student::showMyOrder()
26 {
27 }
28 }
29
30 //查看所有预约
31 void Student::showAllOrder()
32 {
33 }
34 }
35
36 //取消预约
37 void Student::cancelOrder()
38 {
39 }
40 }
41
```

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

teacher.h 中添加如下代码：

```
1  #pragma once
2  #define _CRT_SECURE_NO_WARNINGS
3  #include<iostream>
4  using namespace std;
5  #include "identity.h"
6
7  class Teacher :public Identity
8  {
9  public:
10
11      //默认构造
12      Teacher();
13
14      //有参构造（职工编号，姓名，密码）
15      Teacher(int empId, string name, string pwd);
16
17      //菜单界面
18      virtual void operMenu();
19
20      //查看所有预约
21      void showAllOrder();
22
23      //审核预约
24      void validOrder();
25
26      int m_EmpId; //教师编号
27
28  };
```

- teacher.cpp 中添加如下代码：

```
1  #include"teacher.h"
2
3  //默认构造
4  Teacher::Teacher()
5  {
6  }
7
```

```

8 //有参构造（职工编号，姓名，密码）
9 Teacher::Teacher(int empId, string name, string pwd)
10 {
11 }
12
13 //菜单界面
14 void Teacher::operMenu()
15 {
16 }
17
18 //查看所有预约
19 void Teacher::showAllOrder()
20 {
21 }
22
23 //审核预约
24 void Teacher::validOrder()
25 {
26 }

```

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老师账户进行管理，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

manager.h 中添加如下代码：

```

1 #pragma once
2 #include<iostream>
3 using namespace std;
4 #include "identity.h"
5
6 class Manager :public Identity
7 {
8 public:

```

```

9
10     //默认构造
11     Manager();
12
13     //有参构造  管理员姓名, 密码
14     Manager(string name, string pwd);
15
16     //选择菜单
17     virtual void operMenu();
18
19     //添加账号
20     void addPerson();
21
22     //查看账号
23     void showPerson();
24
25     //查看机房信息
26     void showComputer();
27
28     //清空预约记录
29     void cleanFile();
30
31 };

```

- manager.cpp中添加如下代码:

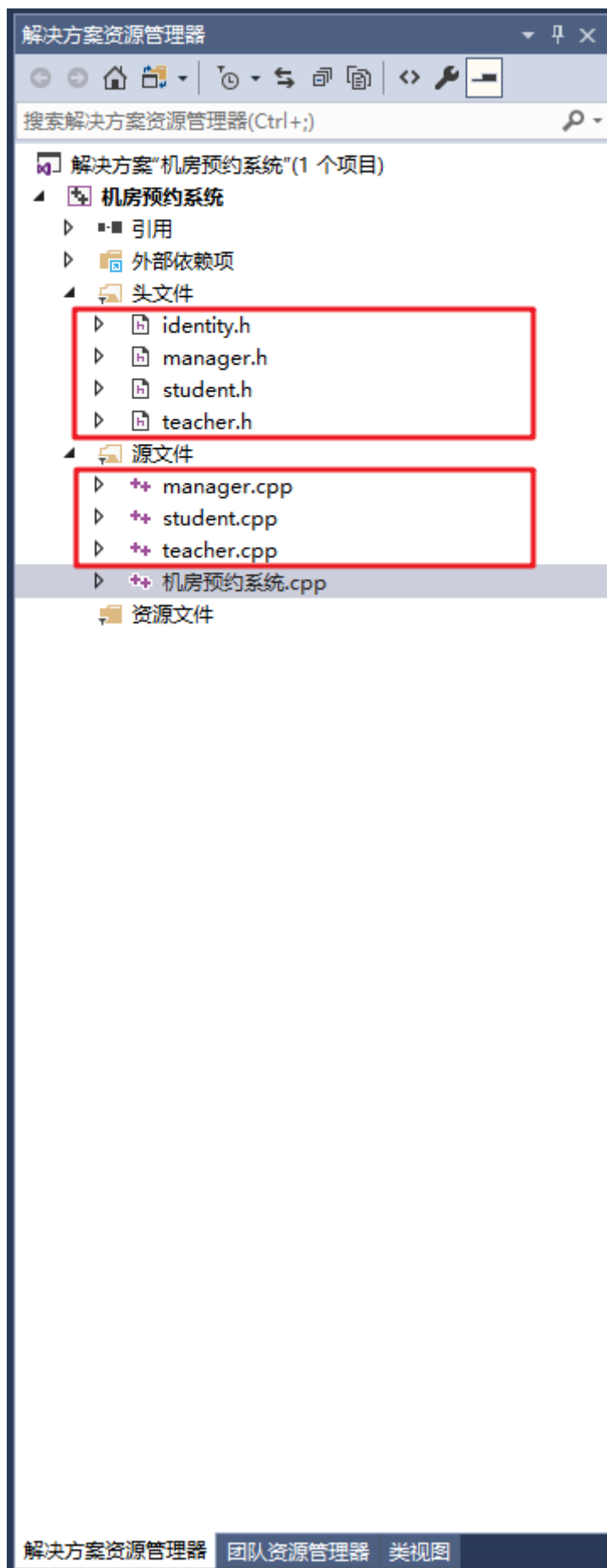
```

1  #include "manager.h"
2
3  //默认构造
4  Manager::Manager()
5  {
6  }
7
8  //有参构造
9  Manager::Manager(string name, string pwd)
10 {
11 }
12
13 //选择菜单
14 void Manager::operMenu()
15 {
16 }
17
18 //添加账号
19 void Manager::addPerson()
20 {
21 }
22
23 //查看账号
24 void Manager::showPerson()
25 {
26 }
27
28 //查看机房信息
29 void Manager::showComputer()

```

```
30 {  
31 }  
32  
33 //清空预约记录  
34 void Manager::cleanFile()  
35 {  
36 }
```

至此，所有身份类创建完毕，效果如图：



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操作，我们可以将所有的文件名定义到一个全局的文件中
- 在头文件中添加 **globalFile.h** 文件
- 并添加如下代码：

```
1  #pragma once
2
3  //管理员文件
4  #define ADMIN_FILE      "admin.txt"
5  //学生文件
6  #define STUDENT_FILE    "student.txt"
7  //教师文件
8  #define TEACHER_FILE    "teacher.txt"
9  //机房信息文件
10 #define COMPUTER_FILE   "computerRoom.txt"
11 //订单文件
12 #define ORDER_FILE      "order.txt"
```

并且在同级目录下，创建这几个文件

| 名称 | 修改日期 | 类型 | 大小 |
|------------------------|-----------------|---------------------|------|
| Debug | 2019/1/27 11:14 | 文件夹 | |
| globalFile.h | 2019/1/27 15:51 | C/C++ Header | 1 KB |
| identity.h | 2019/1/27 15:14 | C/C++ Header | 1 KB |
| manager.h | 2019/1/27 15:30 | C/C++ Header | 1 KB |
| student.h | 2019/1/27 15:20 | C/C++ Header | 1 KB |
| teacher.h | 2019/1/27 15:23 | C/C++ Header | 1 KB |
| manager.cpp | 2019/1/27 15:31 | C++ Source | 1 KB |
| student.cpp | 2019/1/27 15:21 | C++ Source | 1 KB |
| teacher.cpp | 2019/1/27 15:25 | C++ Source | 1 KB |
| 机房预约系统.cpp | 2019/1/27 15:52 | C++ Source | 4 KB |
| 机房预约系统.vcxproj.user | 2019/1/27 10:19 | Per-User Project... | 1 KB |
| 机房预约系统.vcxproj | 2019/1/27 10:40 | VC++ Project | 6 KB |
| 机房预约系统.vcxproj.filters | 2019/1/27 10:40 | VC++ Project Fil... | 1 KB |
| admin.txt | 2019/1/27 15:53 | 文本文档 | 0 KB |
| computerRoom.txt | 2019/1/27 15:53 | 文本文档 | 0 KB |
| order.txt | 2019/1/26 13:54 | 文本文档 | 0 KB |
| student.txt | 2019/1/27 15:53 | 文本文档 | 0 KB |
| teacher.txt | 2019/1/27 15:53 | 文本文档 | 0 KB |

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：

```
1  #include "globalFile.h"
2  #include "identity.h"
3  #include <fstream>
4  #include <string>
5
6
7  //登录功能
8  void LoginIn(string fileName, int type)
9  {
10
11     Identity * person = NULL;
12
13     ifstream ifs;
14     ifs.open(fileName, ios::in);
15
16     //文件不存在情况
17     if (!ifs.is_open())
18     {
19         cout << "文件不存在" << endl;
20         ifs.close();
21         return;
22     }
23
24     int id = 0;
25     string name;
26     string pwd;
27
28     if (type == 1) //学生登录
29     {
30         cout << "请输入你的学号" << endl;
31         cin >> id;
32     }
33     else if (type == 2) //教师登录
34     {
35         cout << "请输入你的职工号" << endl;
36         cin >> id;
37     }
38
39     cout << "请输入用户名: " << endl;
40     cin >> name;
41
42     cout << "请输入密码: " << endl;
```

```

43     cin >> pwd;
44
45
46     if (type == 1)
47     {
48         //学生登录验证
49     }
50     else if (type == 2)
51     {
52         //教师登录验证
53     }
54     else if (type == 3)
55     {
56         //管理员登录验证
57     }
58
59     cout << "验证登录失败!" << endl;
60
61     system("pause");
62     system("cls");
63     return;
64 }

```

- 在main函数的不同分支中，填入不同的登录接口

```

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: //管理员身份
    LoginIn(ADMIN_FILE, 3);
    break;
case 0: //退出系统
    cout << "欢迎下一次使用" << endl;
    system("pause");
    return 0;
    break;
default:
    cout << "输入有误，请重新选择!" << endl;
    system("pause");
    system("cls");
    break;
}

```

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 1 张三 123
2 2 李四 123456
```

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:



在Login函数的学生分支中加入如下代码，验证学生身份

```
1 //学生登录验证
2 int fId;
3 string fName;
4 string fPwd;
5 while (ifs >> fId && ifs >> fName && ifs >> fPwd)
6 {
7     if (id == fId && name == fName && pwd == fPwd)
8     {
9         cout << "学生验证登录成功!" << endl;
10        system("pause");
11        system("cls");
12        person = new Student(id, name, pwd);
13
14        return;
15    }
16 }
```

添加代码效果图

```

if (type == 1)
{
    //学生登录验证
    int fId;
    string fName;
    string fPwd;
    while (ifs >> fId && ifs >> fName && ifs >> fPwd)
    {
        if (id == fId && name == fName && pwd == fPwd)
        {
            cout << "学生验证登录成功!" << endl;
            system("pause");
            system("cls");
            person = new Student(id, name, pwd);

            return;
        }
    }
}
else if (type == 2)
{
    //教师登录验证
}
else if (type == 3)
{
    //管理员登录验证
}

```

测试:

```

F:\VS项目\机房预约系统\Debug\机房预约系统.exe
===== 欢迎来到传智播客机房预约系统 =====
请输入您的身份
-----
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:
123
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 1 老王 123
```

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

```
1 //教师登录验证
2 int fId;
3 string fName;
4 string fPwd;
5 while (ifs >> fId && ifs >> fName && ifs >> fPwd)
6 {
7     if (id == fId && name == fName && pwd == fPwd)
8     {
9         cout << "教师验证登录成功!" << endl;
10        system("pause");
11        system("cls");
12        person = new Teacher(id, name, pwd);
13        return;
14    }
15 }
```

添加代码效果图

```

if (type == 1)
{
    //学生登录验证
    int fId;
    string fName;
    string fPwd;
    while (ifs >> fId && ifs >> fName && ifs >> fPwd)
    {
        if (id == fId && name == fName && pwd == fPwd)
        {
            cout << "学生验证登录成功!" << endl;
            system("pause");
            system("cls");
            person = new Student(id, name, pwd);

            return;
        }
    }
}
else if (type == 2)
{
    //教师登录验证
    int fId;
    string fName;
    string fPwd;
    while (ifs >> fId && ifs >> fName && ifs >> fPwd)
    {
        if (id == fId && name == fName && pwd == fPwd)
        {
            cout << "教师验证登录成功!" << endl;
            system("pause");
            system("cls");
            person = new Teacher(id, name, pwd);
            return;
        }
    }
}
}

```

测试:

```

F:\VS项目\机房预约系统(Debug)\机房预约系统.exe
===== 欢迎来到传智播客机房预约系统 =====
请输入您的身份
    1. 学生代表
    2. 老师
    3. 管理员
    0. 退出
输入您的选择: 2
请输入你的职工号
1
请输入用户名:
老师
请输入密码:
123
教师验证登录成功!
请按任意键继续...

```

6.5 管理员登录实现

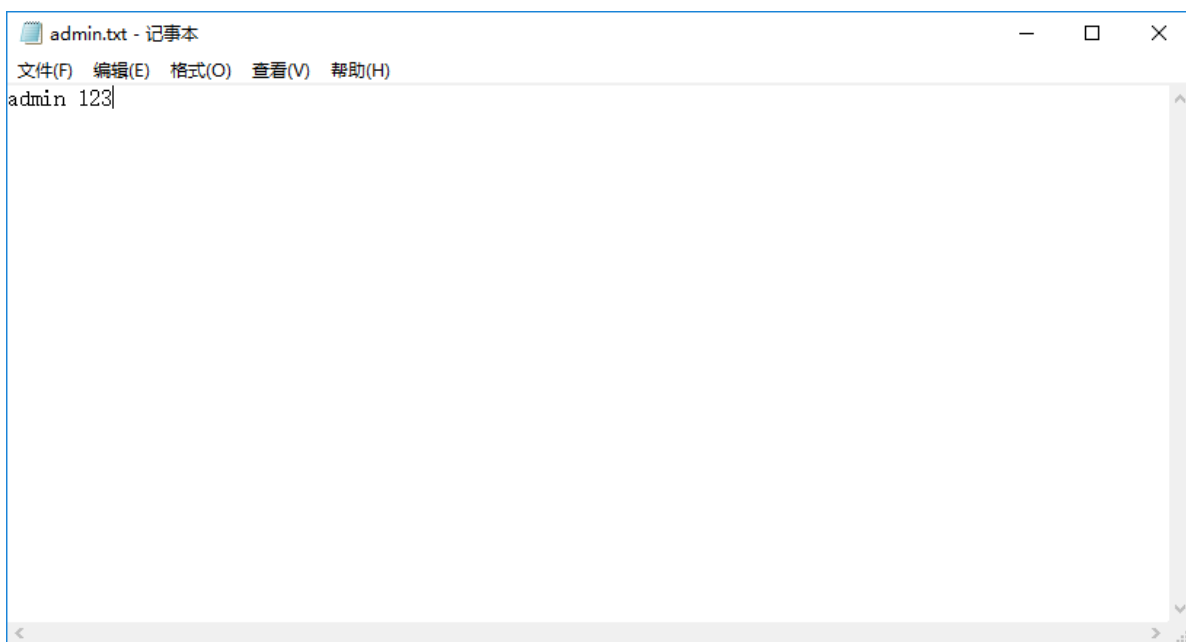
在admin.txt文件中添加一条管理员信息，由于我们只有一条管理员，因此本案例中没有添加管理员的功能

添加信息:

```
1 admin 123
```

其中: admin 代表管理员用户名, 123 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

```
1 //管理员登录验证
2     string fName;
3     string fPwd;
4     while (ifs >> fName && ifs >> fPwd)
5     {
6         if (name == fName && pwd == fPwd)
7         {
8             cout << "验证登录成功!" << endl;
9             //登录后，按任意键进入管理员界面
10            system("pause");
11            system("cls");
12            //创建管理员对象
13            person = new Manager(name, pwd);
14            return;
15        }
16    }
```

添加效果如图:


```

else if (type == 2)
{
    //教师登录验证
    int fId;
    string fName;
    string fPwd;
    while (ifs >> fId && ifs >> fName && ifs >> fPwd)
    {
        if (id == fId && name == fName && pwd == fPwd)
        {
            cout << "教师验证登录成功!" << endl;
            system("pause");
            system("cls");
            person = new Teacher(id, name, pwd);
            return;
        }
    }
}

else if(type == 3)
{
    //管理员登录验证
    string fName;
    string fPwd;
    while (ifs >> fName && ifs >> fPwd)
    {
        if (name == fName && pwd == fPwd)
        {
            cout << "管理员验证登录成功!" << 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;
        }
    }
}

```

测试效果如图：



至此，所有身份的登录功能全部实现！

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：

```
1 //有参构造
2 Manager::Manager(string name, string pwd)
3 {
4     this->m_Name = name;
5     this->m_Pwd = pwd;
6 }
```

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：

```
1 //管理员菜单
2 void managerMenu(Identity * &manager)
3 {
4     while (true)
5     {
6         //管理员菜单
7         manager->operMenu();
8
9         Manager* man = (Manager*)manager;
10        int select = 0;
11
12        cin >> select;
13
14        if (select == 1) //添加账号
15        {
16            cout << "添加账号" << endl;
```

```

17         man->addPerson();
18     }
19     else if (select == 2) //查看账号
20     {
21         cout << "查看账号" << endl;
22         man->showPerson();
23     }
24     else if (select == 3) //查看机房
25     {
26         cout << "查看机房" << endl;
27         man->showComputer();
28     }
29     else if (select == 4) //清空预约
30     {
31         cout << "清空预约" << endl;
32         man->cleanFile();
33     }
34     else
35     {
36         delete manager;
37         cout << "注销成功" << endl;
38         system("pause");
39         system("cls");
40         return;
41     }
42 }
43 }

```

7.1.3 菜单功能实现

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

```

1 //选择菜单
2 void Manager::operMenu()
3 {
4     cout << "欢迎管理员: "<<this->m_Name << "登录! " << endl;
5     cout << "\t\t -----\n";
6     cout << "\t\t|                                     |\n";
7     cout << "\t\t|          1. 添加账号          |\n";
8     cout << "\t\t|                                     |\n";
9     cout << "\t\t|          2. 查看账号          |\n";
10    cout << "\t\t|                                     |\n";
11    cout << "\t\t|          3. 查看机房          |\n";
12    cout << "\t\t|                                     |\n";
13    cout << "\t\t|          4. 清空预约          |\n";
14    cout << "\t\t|                                     |\n";
15    cout << "\t\t|          0. 注销登录          |\n";
16    cout << "\t\t|                                     |\n";
17    cout << "\t\t -----\n";
18    cout << "请选择您的操作:  " << endl;
19 }

```

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：

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

添加效果如：

```
else if(type == 3)
{
    //管理员登录验证
    string fName;
    string fPwd;
    while (ifs >> fName && ifs >> fPwd)
    {
        if (name == fName && pwd == fPwd)
        {
            cout << "管理员验证登录成功!" << 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);
            //进入管理员子菜单
            managerMenu(person);
            return;
        }
    }
}
```

测试对接，效果如图：



登录成功



注销登录:



至此，管理员身份可以成功登录以及注销

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:

- 添加时学生学号不能重复、教师职工号不能重复

7.2.1 添加功能实现

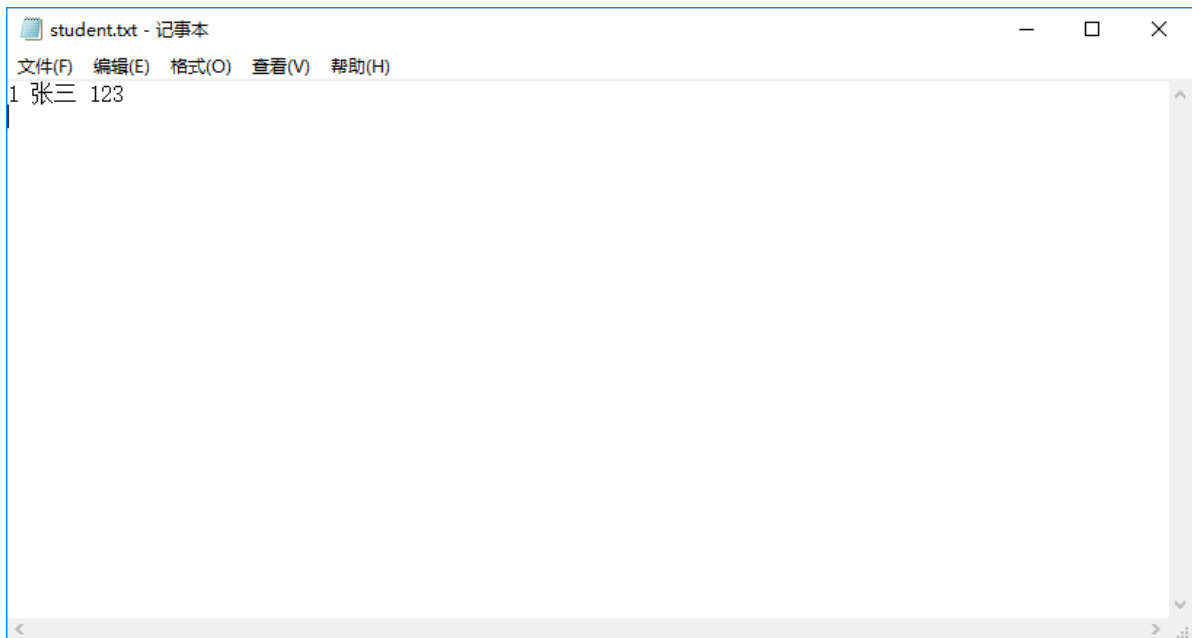
在Manager的**addPerson**成员函数中，实现添加新账号功能，代码如下：

```
1  //添加账号
2  void Manager::addPerson()
3  {
4
5      cout << "请输入添加账号的类型" << endl;
6      cout << "1、添加学生" << endl;
7      cout << "2、添加老师" << endl;
8
9      string fileName;
10     string tip;
11     ofstream ofs;
12
13     int select = 0;
14     cin >> select;
15
16     if (select == 1)
17     {
18         fileName = STUDENT_FILE;
19         tip = "请输入学号: ";
20     }
21     else
22     {
23         fileName = TEACHER_FILE;
24         tip = "请输入职工编号: ";
25     }
26
27     ofs.open(fileName, ios::out | ios::app);
28     int id;
29     string name;
30     string pwd;
31     cout << tip << endl;
32     cin >> id;
33
34     cout << "请输入姓名: " << endl;
35     cin >> name;
36
37     cout << "请输入密码: " << endl;
38     cin >> pwd;
39
40     ofs << id << " " << name << " " << pwd << " " << endl;
41     cout << "添加成功" << endl;
42
43     system("pause");
44     system("cls");
45
46     ofs.close();
47 }
```

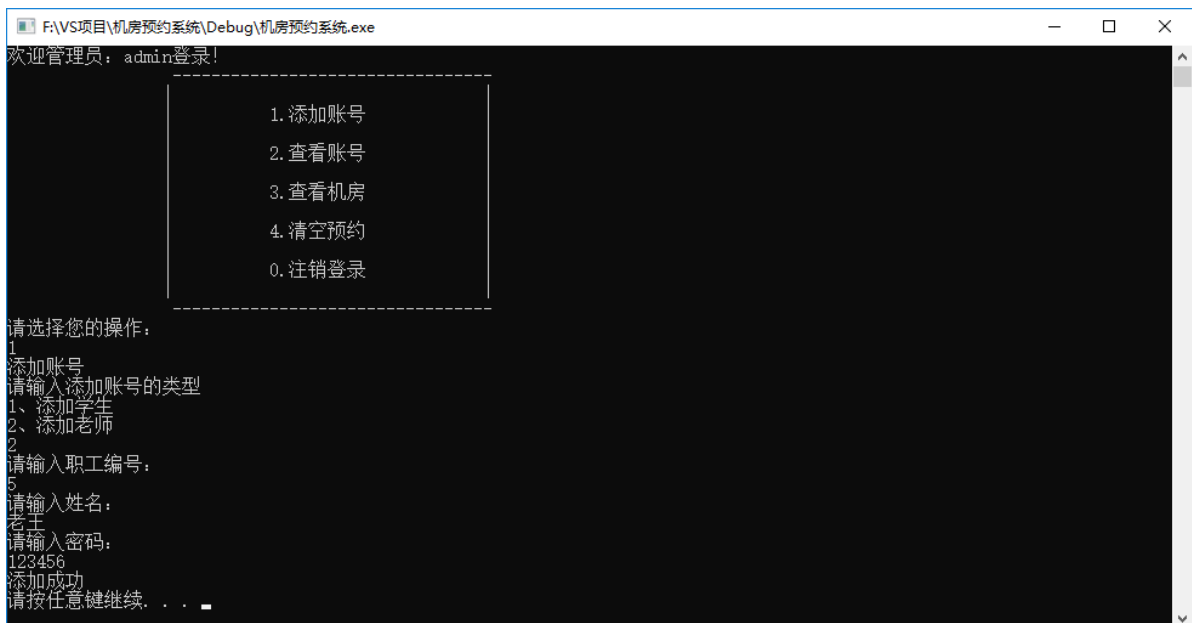
测试添加学生：



成功在学生文件中添加了一条信息



测试添加教师:



成功在教师文件中添加了一条信息



7.2.2 去重操作

功能描述：添加新账号时，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

7.2.2.1 读取信息

- 要去除重复的账号，首先要先将学生和教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，用于存放学生和教师的信息
- 添加一个新的成员函数 `void initVector()` 初始化容器

```
1 //初始化容器
2 void initVector();
3
4 //学生容器
5 vector<Student> vStu;
6
7 //教师容器
8 vector<Teacher> vTea;
```

添加位置如图：


```

class Manager :public Identity
{
public:

    //默认构造
    Manager();

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

    //选择菜单
    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 cleanFile();

    //初始化容器
    void initVector();

    //学生容器
    vector<Student> vStu;

    //教师容器
    vector<Teacher> vTea;
};

```

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：

```

1 void Manager::initVector()
2 {
3     //读取学生文件中信息
4     ifstream ifs;
5     ifs.open(STUDENT_FILE, ios::in);
6     if (!ifs.is_open())
7     {
8         cout << "文件读取失败" << endl;
9         return;
10    }
11
12    vStu.clear();
13    vTea.clear();
14

```

```

15     Student s;
16     while (ifs >> s.m_Id && ifs >> s.m_Name && ifs >> s.m_Pwd)
17     {
18         vstu.push_back(s);
19     }
20     cout << "当前学生数量为: " << vstu.size() << endl;
21     ifs.close(); //学生初始化
22
23     //读取老师文件信息
24     ifs.open(TEACHER_FILE, ios::in);
25
26     Teacher t;
27     while (ifs >> t.m_EmpId && ifs >> t.m_Name && ifs >> t.m_Pwd)
28     {
29         vTea.push_back(t);
30     }
31     cout << "当前教师数量为: " << vTea.size() << endl;
32
33     ifs.close();
34 }

```

在有参构造函数中，调用初始化容器函数

```

1 //有参构造
2 Manager::Manager(string name, string pwd)
3 {
4     this->m_Name = name;
5     this->m_Pwd = pwd;
6
7     //初始化容器
8     this->initVector();
9 }

```

测试，运行代码可以看到测试代码获取当前学生和教师数量



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);`

```
1 //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)
2 bool checkRepeat(int id, int type);
```

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

```
1 bool Manager::checkRepeat(int id, int type)
2 {
3     if (type == 1)
4     {
5         for (vector<Student>::iterator it = vStu.begin(); it != vStu.end();
6 it++)
7         {
8             if (id == it->m_Id)
9             {
10                return true;
11            }
12        }
13    }
14    else
15    {
16        for (vector<Teacher>::iterator it = vTea.begin(); it != vTea.end();
17 it++)
18        {
19            if (id == it->m_EmpId)
20            {
21                return true;
22            }
23        }
24        return false;
25    }
26 }
```

7.2.2.3 添加去重操作

在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：

```
1 string errorTip; //重复错误提示
2
3 if (select == 1)
4 {
5     fileName = STUDENT_FILE;
6     tip = "请输入学号: ";
7     errorTip = "学号重复, 请重新输入";
8 }
9 else
10 {
11     fileName = TEACHER_FILE;
```

```
12         tip = "请输入职工编号: ";
13         errorTip = "职工号重复, 请重新输入";
14     }
15     ofs.open(fileName, ios::out | ios::app);
16     int id;
17     string name;
18     string pwd;
19     cout << tip << endl;
20
21     while (true)
22     {
23         cin >> id;
24
25         bool ret = this->checkRepeat(id, 1);
26
27         if (ret) //有重复
28         {
29             cout << errorTip << endl;
30         }
31         else
32         {
33             break;
34         }
35     }
```

代码位置如图:

```
string errorTip; //重复错误提示
```

```
if (select == 1)
{
    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";
}
ofs.open(fileName, ios::out | ios::app);
int id;
string name;
string pwd;
cout << tip << endl;
```

```
while (true)
{
    cin >> id;

    bool ret = this->checkRepeat(id, 1);

    if (ret) //有重复
    {
        cout << errorTip << endl;
    }
    else
    {
        break;
    }
}
```

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, 因此不会做检测
- 导致刚加入的账号的学生号或者职工编号, 再次添加时依然可以重复

解决方案:

- 在每次添加新账号时, 重新初始化容器

在添加完毕后, 加入代码:

```
1 //初始化容器
2 this->initVector();
```

位置如图:



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：显示学生信息或教师信息

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：

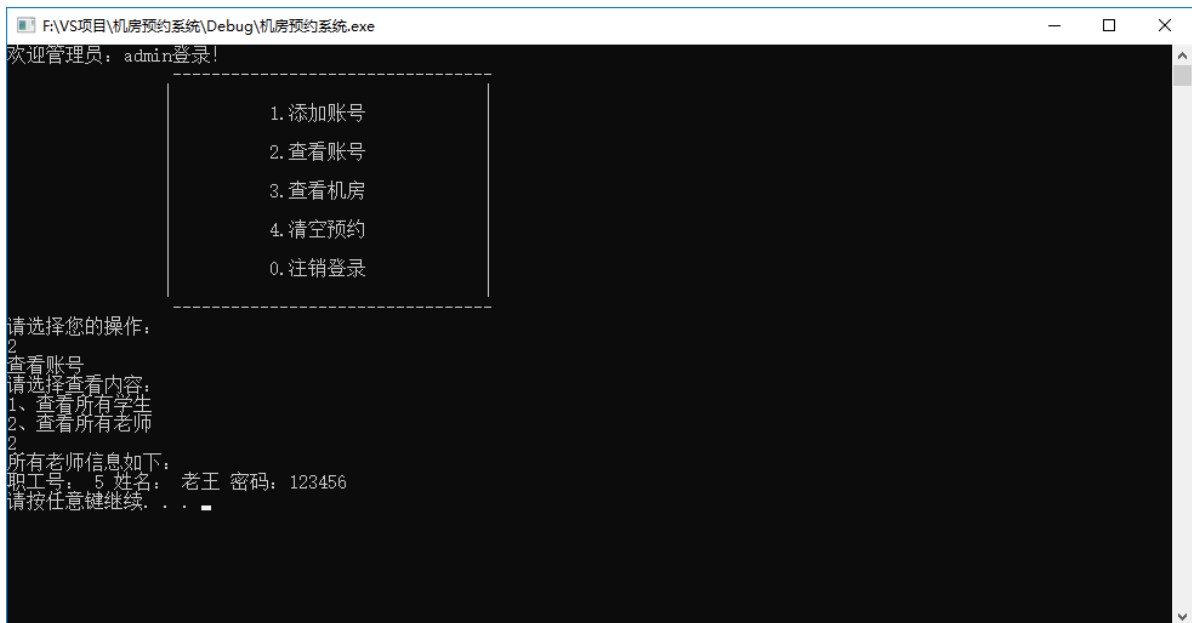
```
1 void printStudent(Student & s)
2 {
3     cout << "学号: " << s.m_Id << " 姓名: " << s.m_Name << " 密码: " <<
s.m_Pwd << endl;
4 }
5 void printTeacher(Teacher & t)
6 {
7     cout << "职工号: " << t.m_EmpId << " 姓名: " << t.m_Name << " 密码: " <<
t.m_Pwd << endl;
8 }
9
10 void Manager::showPerson()
11 {
12     cout << "请选择查看内容: " << endl;
13     cout << "1、查看所有学生" << endl;
14     cout << "2、查看所有老师" << endl;
15
16     int select = 0;
17
18     cin >> select;
19
20     if (select == 1)
21     {
22         cout << "所有学生信息如下: " << endl;
23         for_each(vStu.begin(), vStu.end(), printStudent);
24     }
25     else
26     {
27         cout << "所有老师信息如下: " << endl;
28         for_each(vTea.begin(), vTea.end(), printTeacher);
29     }
30     system("pause");
31     system("cls");
32 }
```

7.3.2 测试

测试查看学生效果



测试查看教师效果



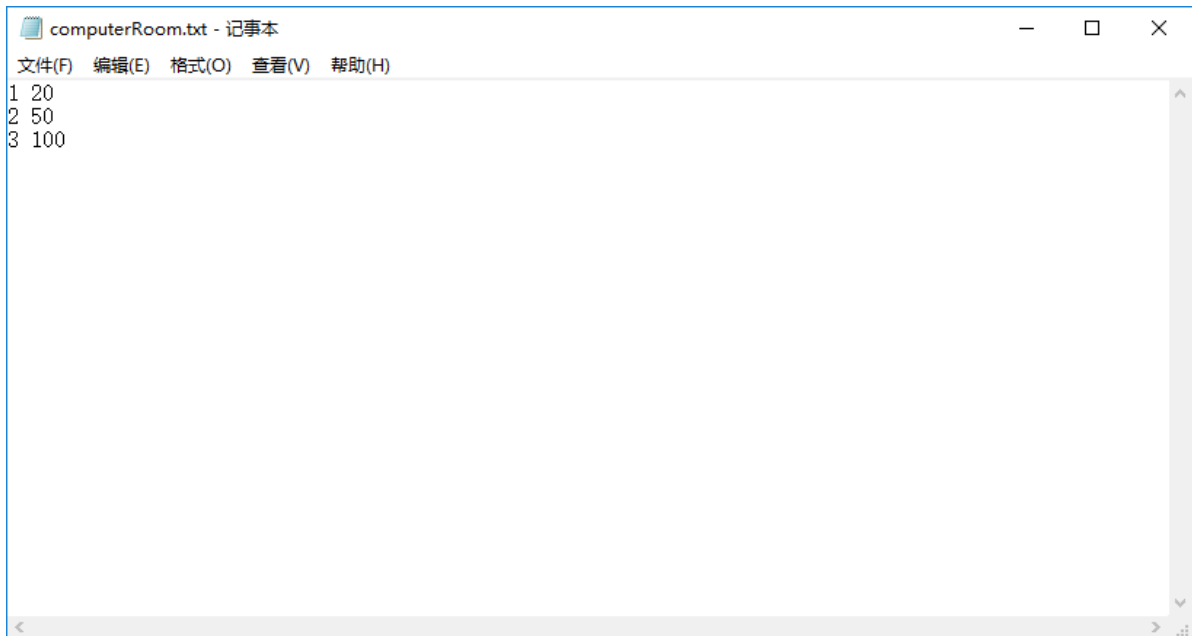
至此，显示账号功能实现完毕

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 机房类创建

在头文件下，创建新的文件 computerRoom.h

并添加如下代码：

```
1  #pragma once
2  #include<iostream>
3  using namespace std;
4  //机房类
5  class ComputerRoom
6  {
7  public:
8
9      int m_ComId; //机房id号
10
11     int m_MaxNum; //机房最大容量
12 };
```

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

```
1  //机房容器
2  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

```

1      //获取机房信息
2      ifstream ifs;
3
4      ifs.open(COMPUTER_FILE, ios::in);
5
6      ComputerRoom c;
7      while (ifs >> c.m_ComId && ifs >> c.m_MaxNum)
8      {
9          vCom.push_back(c);
10     }
11     cout << "当前机房数量为: " << vCom.size() << endl;
12
13     ifs.close();

```

位置如图:

```

//有参构造
Manager::Manager(string name, string pwd)
{
    this->m_Name = name;
    this->m_Pwd = pwd;
    //初始化容器
    this->initVector();

    //获取机房信息
    ifstream ifs;

    ifs.open(COMPUTER_FILE, ios::in);

    ComputerRoom c;
    while (ifs >> c.m_ComId && ifs >> c.m_MaxNum)
    {
        vCom.push_back(c);
    }
    cout << "当前机房数量为: " << vCom.size() << endl;

    ifs.close();
}

```

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:

```

1 //查看机房信息
2 void Manager::showComputer()
3 {
4     cout << "机房信息如下: " << endl;
5     for (vector<ComputerRoom>::iterator it = vCom.begin(); it != vCom.end();
6         it++)
7     {
8         cout << "机房编号: " << it->m_ComId << " 机房最大容量: " << it-
9         >m_MaxNum << endl;
10    }
11    system("pause");
12    system("cls");
13 }

```

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:

```

1 //清空预约记录
2 void Manager::cleanFile()
3 {
4     ofstream ofs(ORDER_FILE, ios::trunc);
5     ofs.close();
6
7     cout << "清空成功! " << endl;
8     system("pause");
9     system("cls");
10 }

```

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：

```

1 //有参构造(学号、姓名、密码)
2 Student::Student(int id, string name, string pwd)
3 {
4     //初始化属性
5     this->m_Id = id;
6     this->m_Name = name;
7     this->m_Pwd = pwd;
8 }

```

8.1.2 管理员子菜单

- 在机房预约系统.cpp中，当用户登录的是学生，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：

```

1 //学生菜单
2 void studentMenu(Identity * &student)
3 {
4     while (true)

```

```

5      {
6          //学生菜单
7          student->operMenu();
8
9          Student* stu = (Student*)student;
10         int select = 0;
11
12         cin >> select;
13
14         if (select == 1) //申请预约
15         {
16             stu->applyOrder();
17         }
18         else if (select == 2) //查看自身预约
19         {
20             stu->showMyOrder();
21         }
22         else if (select == 3) //查看所有预约
23         {
24             stu->showAllOrder();
25         }
26         else if (select == 4) //取消预约
27         {
28             stu->cancelOrder();
29         }
30         else
31         {
32             delete student;
33             cout << "注销成功" << endl;
34             system("pause");
35             system("cls");
36             return;
37         }
38     }
39 }

```

8.1.3 菜单功能实现

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

```

1  //菜单界面
2  void Student::operMenu()
3  {
4      cout << "欢迎学生代表: " << this->m_Name << "登录! " << endl;
5      cout << "\t\t -----\n";
6      cout << "\t\t | \n";
7      cout << "\t\t |      1. 申请预约      | \n";
8      cout << "\t\t | \n";
9      cout << "\t\t |      2. 查看我的预约    | \n";
10     cout << "\t\t | \n";
11     cout << "\t\t |      3. 查看所有预约    | \n";
12     cout << "\t\t | \n";
13     cout << "\t\t |      4. 取消预约      | \n";
14     cout << "\t\t | \n";
15     cout << "\t\t |      0. 注销登录      | \n";

```

```

16     cout << "\\t\\t|                                     |\\n";
17     cout << "\\t\\t -----\\n";
18     cout << "请选择您的操作： " << endl;
19 }

```

8.1.4 接口对接

- 学生成功登录后，调用学生的子菜单界面
- 在学生登录分支中，添加代码：

```

1 //进入学生子菜单
2 studentMenu(person);

```

添加效果如图：

```

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

if (type == 1)
{
    //学生登录验证
    int fId;
    string fName;
    string fPwd;
    while (ifs >> fId && ifs >> fName && ifs >> fPwd)
    {
        if (id == fId && name == fName && pwd == fPwd)
        {
            cout << "学生验证登录成功!" << endl;
            system("pause");
            system("cls");
            person = new Student(id, name, pwd);
            //进入学生子菜单
            studentMenu(person);
            return;
        }
    }
}
else if (type == 2)
{
    //教师登录验证
    int fId;
    string fName;
    string fPwd;
    while (ifs >> fId && ifs >> fName && ifs >> fPwd)
    {
        if (id == fId && name == fName && pwd == fPwd)
        {
            cout << "教师验证登录成功!" << endl;
            system("pause");
            system("cls");
            person = new Teacher(id, name, pwd);
            return;
        }
    }
}

```

测试对接，效果如图：

登录验证通过：



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：

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

在学生的有参构造函数中追加如下代码：

```
1 //获取机房信息
2 ifstream ifs;
3 ifs.open(COMPUTER_FILE, ios::in);
4
5 ComputerRoom c;
6 while (ifs >> c.m_ComId && ifs >> c.m_MaxNum)
7 {
8     vCom.push_back(c);
9 }
10
11 ifs.close();
```

追加位置如图：


```

//有参构造(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;

    //获取机房信息
    ifstream ifs;
    ifs.open(COMPUTER_FILE, ios::in);

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

    ifs.close();
}
}

```

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()`

```

1  //申请预约
2  void Student::applyOrder()
3  {
4      cout << "机房开放时间为周一至周五!" << endl;
5      cout << "请输入申请预约的时间: " << endl;
6      cout << "1、周一" << endl;
7      cout << "2、周二" << endl;
8      cout << "3、周三" << endl;
9      cout << "4、周四" << endl;
10     cout << "5、周五" << endl;
11     int date = 0;
12     int interval = 0;
13     int room = 0;
14
15     while (true)
16     {
17         cin >> date;
18         if (date >= 1 && date <= 5)
19         {
20             break;
21         }
22         cout << "输入有误，请重新输入" << endl;
23     }
24
25
26     cout << "请输入申请预约的时间段: " << endl;
27     cout << "1、上午" << endl;

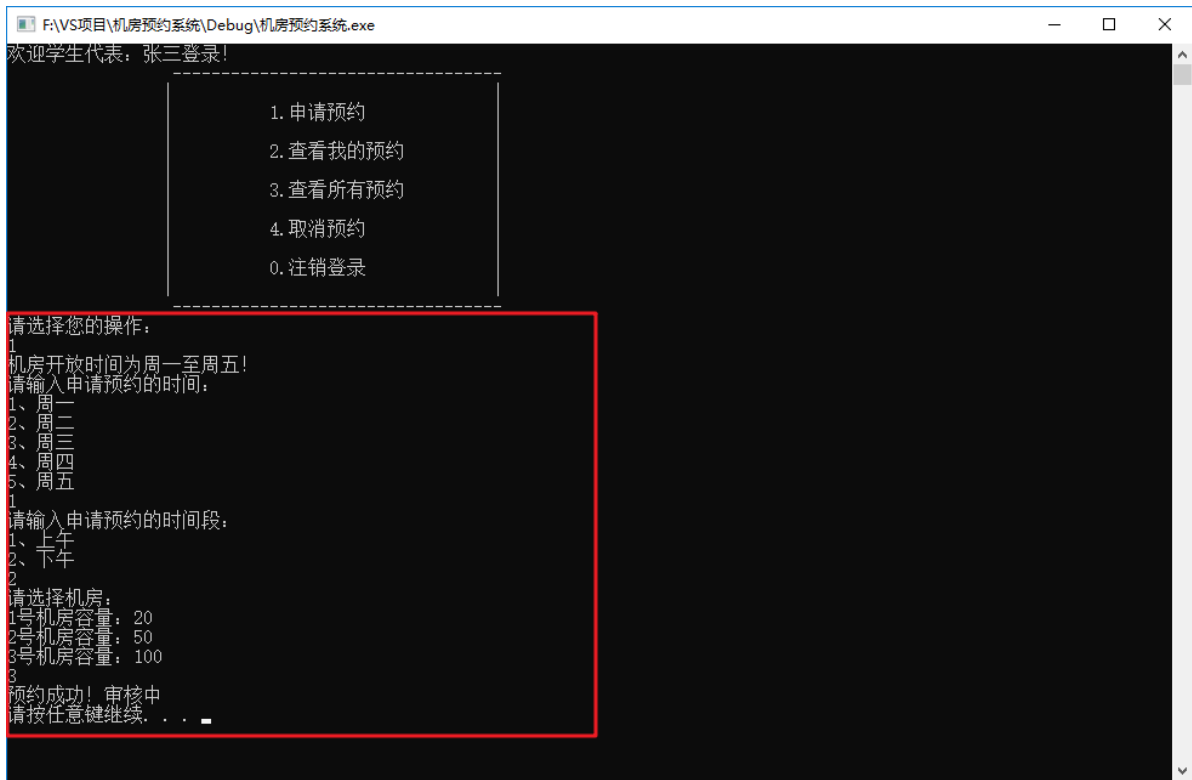
```

```

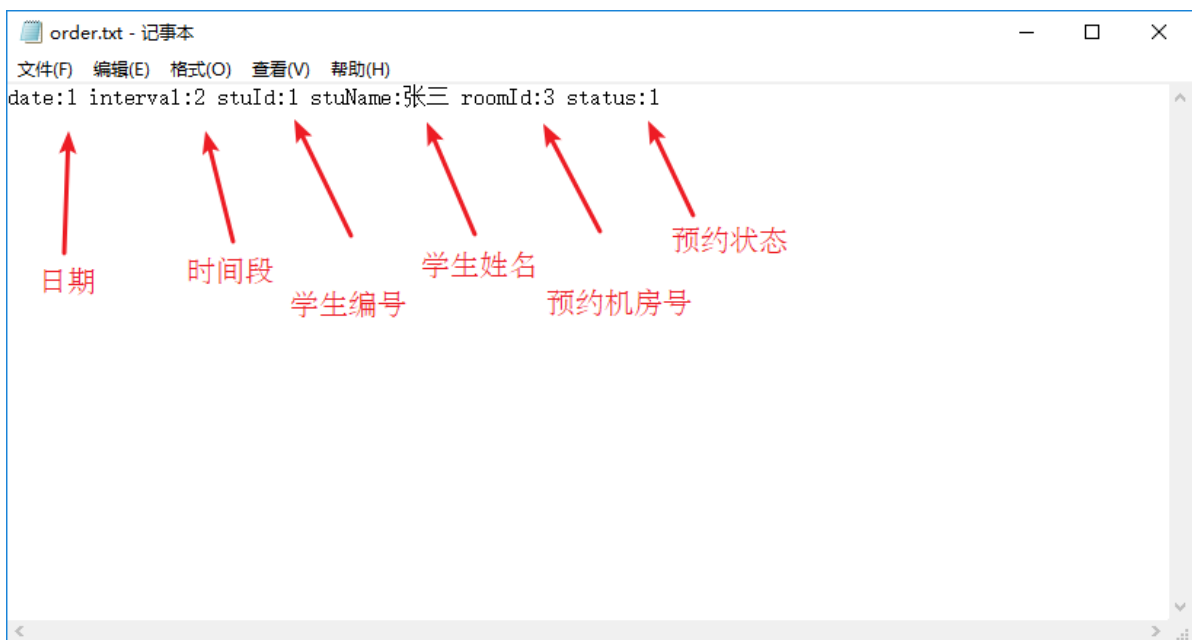
28     cout << "2、下午" << endl;
29
30     while (true)
31     {
32         cin >> interval;
33         if (interval >= 1 && interval <= 2)
34         {
35             break;
36         }
37         cout << "输入有误，请重新输入" << endl;
38     }
39
40     cout << "请选择机房：" << endl;
41     cout << "1号机房容量：" << vCom[0].m_MaxNum << endl;
42     cout << "2号机房容量：" << vCom[1].m_MaxNum << endl;
43     cout << "3号机房容量：" << vCom[2].m_MaxNum << endl;
44
45     while (true)
46     {
47         cin >> room;
48         if (room >= 1 && room <= 3)
49         {
50             break;
51         }
52         cout << "输入有误，请重新输入" << endl;
53     }
54
55     cout << "预约成功！审核中" << endl;
56
57     ofstream ofs(ORDER_FILE, ios::app);
58     ofs << "date:" << date << " ";
59     ofs << "interval:" << interval << " ";
60     ofs << "stuId:" << this->m_Id << " ";
61     ofs << "stuName:" << this->m_Name << " ";
62     ofs << "roomId:" << room << " ";
63     ofs << "status:" << 1 << endl;
64
65     ofs.close();
66
67     system("pause");
68     system("cls");
69 }

```

运行程序，测试代码:



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：

```
1 #pragma once
```

```

2  #include<iostream>
3  using namespace std;
4  #include <map>
5  #include "globalFile.h"
6
7  class OrderFile
8  {
9  public:
10
11     //构造函数
12     OrderFile();
13
14     //更新预约记录
15     void updateOrder();
16
17     //记录的容器  key --- 记录的条数  value --- 具体记录的键值对信息
18     map<int, map<string, string>> m_orderData;
19
20     //预约记录条数
21     int m_Size;
22 };

```

构造函数中获取所有信息，并存放在容器中，添加如下代码：

```

1  OrderFile::OrderFile()
2  {
3      ifstream ifs;
4      ifs.open(ORDER_FILE, ios::in);
5
6      string date;      //日期
7      string interval;  //时间段
8      string stuId;     //学生编号
9      string stuName;   //学生姓名
10     string roomId;    //机房编号
11     string status;    //预约状态
12
13
14     this->m_Size = 0; //预约记录个数
15
16     while (ifs >> date && ifs >> interval && ifs >> stuId && ifs >> stuName
&& ifs >> roomId && ifs >> status)
17     {
18         //测试代码
19         /*
20         cout << date << endl;
21         cout << interval << endl;
22         cout << stuId << endl;
23         cout << stuName << endl;
24         cout << roomId << endl;
25         cout << status << endl;
26         */
27
28         string key;
29         string value;
30         map<string, string> m;

```

```

31
32     int pos = date.find(":");
33     if (pos != -1)
34     {
35         key = date.substr(0, pos);
36         value = date.substr(pos + 1, date.size() - pos - 1);
37         m.insert(make_pair(key, value));
38     }
39
40     pos = interval.find(":");
41     if (pos != -1)
42     {
43         key = interval.substr(0, pos);
44         value = interval.substr(pos + 1, interval.size() - pos - 1);
45         m.insert(make_pair(key, value));
46     }
47
48     pos = stuId.find(":");
49     if (pos != -1)
50     {
51         key = stuId.substr(0, pos);
52         value = stuId.substr(pos + 1, stuId.size() - pos - 1);
53         m.insert(make_pair(key, value));
54     }
55
56     pos = stuName.find(":");
57     if (pos != -1)
58     {
59         key = stuName.substr(0, pos);
60         value = stuName.substr(pos + 1, stuName.size() - pos - 1);
61         m.insert(make_pair(key, value));
62     }
63
64     pos = roomId.find(":");
65     if (pos != -1)
66     {
67         key = roomId.substr(0, pos);
68         value = roomId.substr(pos + 1, roomId.size() - pos - 1);
69         m.insert(make_pair(key, value));
70     }
71
72     pos = status.find(":");
73     if (pos != -1)
74     {
75         key = status.substr(0, pos);
76         value = status.substr(pos + 1, status.size() - pos - 1);
77         m.insert(make_pair(key, value));
78     }
79
80
81     this->m_orderData.insert(make_pair(this->m_Size, m));
82     this->m_Size++;
83 }
84
85 //测试代码
86 //for (map<int, map<string, string>>::iterator it = m_orderData.begin();
87 it != m_orderData.end(); it++)
88     //{

```

```

88     // cout << "key = " << it->first << " value = " << endl;
89     // for (map<string, string>::iterator mit = it->second.begin(); mit !=
it->second.end(); mit++)
90     // {
91     //     cout << mit->first << " " << mit->second << " ";
92     // }
93     // cout << endl;
94     //}
95
96     ifs.close();
97 }
98

```

更新预约记录的成员函数updateOrder代码如下：

```

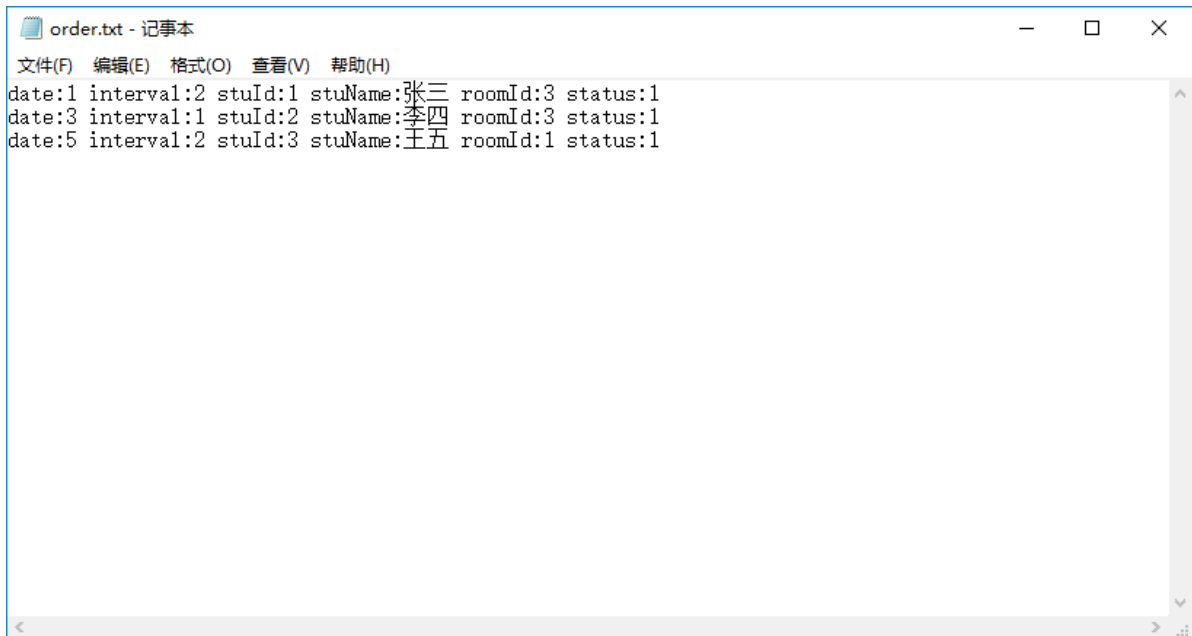
1  void OrderFile::updateOrder()
2  {
3      if (this->m_Size == 0)
4      {
5          return;
6      }
7
8      ofstream ofs(ORDER_FILE, ios::out | ios::trunc);
9      for (int i = 0; i < m_Size; i++)
10     {
11         ofs << "date:" << this->m_orderData[i]["date"] << " ";
12         ofs << "interval:" << this->m_orderData[i]["interval"] << " ";
13         ofs << "stuId:" << this->m_orderData[i]["stuId"] << " ";
14         ofs << "stuName:" << this->m_orderData[i]["stuName"] << " ";
15         ofs << "roomId:" << this->m_orderData[i]["roomId"] << " ";
16         ofs << "status:" << this->m_orderData[i]["status"] << endl;
17     }
18     ofs.close();
19 }

```

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:1 interval:2 stuId:1 stuName:张三 roomId:3 status:1
date:3 interval:1 stuId:2 stuName:李四 roomId:3 status:1
date:5 interval:2 stuId:3 stuName:王五 roomId:1 status:1
```

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

```
1 //查看我的预约
2 void Student::showMyOrder()
3 {
4     OrderFile of;
5     if (of.m_Size == 0)
6     {
7         cout << "无预约记录" << endl;
8         system("pause");
9         system("cls");
10        return;
11    }
12    for (int i = 0; i < of.m_Size; i++)
13    {
14        if (atoi(of.m_orderData[i]["stuId"].c_str()) == this->m_Id)
15        {
16            cout << "预约日期: 周" << of.m_orderData[i]["date"];
17            cout << " 时段: " << (of.m_orderData[i]["interval"] == "1" ? "上
18            午" : "下午");
19            cout << " 机房: " << of.m_orderData[i]["roomId"];
20            string status = " 状态: "; // 0 取消的预约 1 审核中 2 已预约 -1
21            预约失败
22            if (of.m_orderData[i]["status"] == "1")
23            {
24                status += "审核中";
25            }
26            else if (of.m_orderData[i]["status"] == "2")
27            {
28                status += "预约成功";
29            }
30            else if (of.m_orderData[i]["status"] == "-1")
31            {
32                status += "审核未通过, 预约失败";
33            }
34            else
35            {
36                status += "预约已取消";
37            }
38        }
39    }
40}
```

```

36         cout << status << endl;
37
38     }
39 }
40
41     system("pause");
42     system("cls");
43 }

```

测试效果如图：



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

```

1  //查看所有预约
2  void Student::showAllOrder()
3  {
4      OrderFile of;
5      if (of.m_Size == 0)
6      {
7          cout << "无预约记录" << endl;
8          system("pause");
9          system("cls");
10         return;
11     }
12
13     for (int i = 0; i < of.m_Size; i++)
14     {
15         cout << i + 1 << "、 ";
16
17         cout << "预约日期: 周" << of.m_orderData[i]["date"];
18         cout << " 时段: " << (of.m_orderData[i]["interval"] == "1" ? "上午" :
19         "下午");
20         cout << " 学号: " << of.m_orderData[i]["stuId"];
21         cout << " 姓名: " << of.m_orderData[i]["stuName"];
22         cout << " 机房: " << of.m_orderData[i]["roomId"];

```



```

22     string status = " 状态: "; // 0 取消的预约    1 审核中    2 已预约 -1 预约
    失败
23     if (of.m_orderData[i]["status"] == "1")
24     {
25         status += "审核中";
26     }
27     else if (of.m_orderData[i]["status"] == "2")
28     {
29         status += "预约成功";
30     }
31     else if (of.m_orderData[i]["status"] == "-1")
32     {
33         status += "审核未通过，预约失败";
34     }
35     else
36     {
37         status += "预约已取消";
38     }
39     cout << status << endl;
40 }
41
42 system("pause");
43 system("cls");
44 }

```

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

```

1 //取消预约
2 void Student::cancelOrder()
3 {
4     OrderFile of;
5     if (of.m_Size == 0)
6     {

```

```

7         cout << "无预约记录" << endl;
8         system("pause");
9         system("cls");
10        return;
11    }
12    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;
13
14    vector<int>v;
15    int index = 1;
16    for (int i = 0; i < of.m_Size; i++)
17    {
18        if (atoi(of.m_orderData[i]["stuId"].c_str()) == this->m_Id)
19        {
20            if (of.m_orderData[i]["status"] == "1" || of.m_orderData[i]
21["status"] == "2")
22            {
23                v.push_back(i);
24                cout << index ++ << ", ";
25                cout << "预约日期: 周" << of.m_orderData[i]["date"];
26                cout << " 时段: " << (of.m_orderData[i]["interval"] == "1" ?
27"上午" : "下午");
28                cout << " 机房: " << of.m_orderData[i]["roomId"];
29                string status = " 状态: "; // 0 取消的预约 1 审核中 2 已预
30约 -1 预约失败
31                if (of.m_orderData[i]["status"] == "1")
32                {
33                    status += "审核中";
34                }
35                else if (of.m_orderData[i]["status"] == "2")
36                {
37                    status += "预约成功";
38                }
39                cout << status << endl;
40            }
41        }
42    }
43
44    cout << "请输入取消的记录,0代表返回" << endl;
45    int select = 0;
46    while (true)
47    {
48        cin >> select;
49        if (select >= 0 && select <= v.size())
50        {
51            if (select == 0)
52            {
53                break;
54            }
55            else
56            {
57                // cout << "记录所在位置: " << v[select - 1] << endl;
58                of.m_orderData[v[select - 1]]["status"] = "0";
59                of.updateOrder();
60                cout << "已取消预约" << endl;
61                break;
62            }
63        }
64    }

```

```

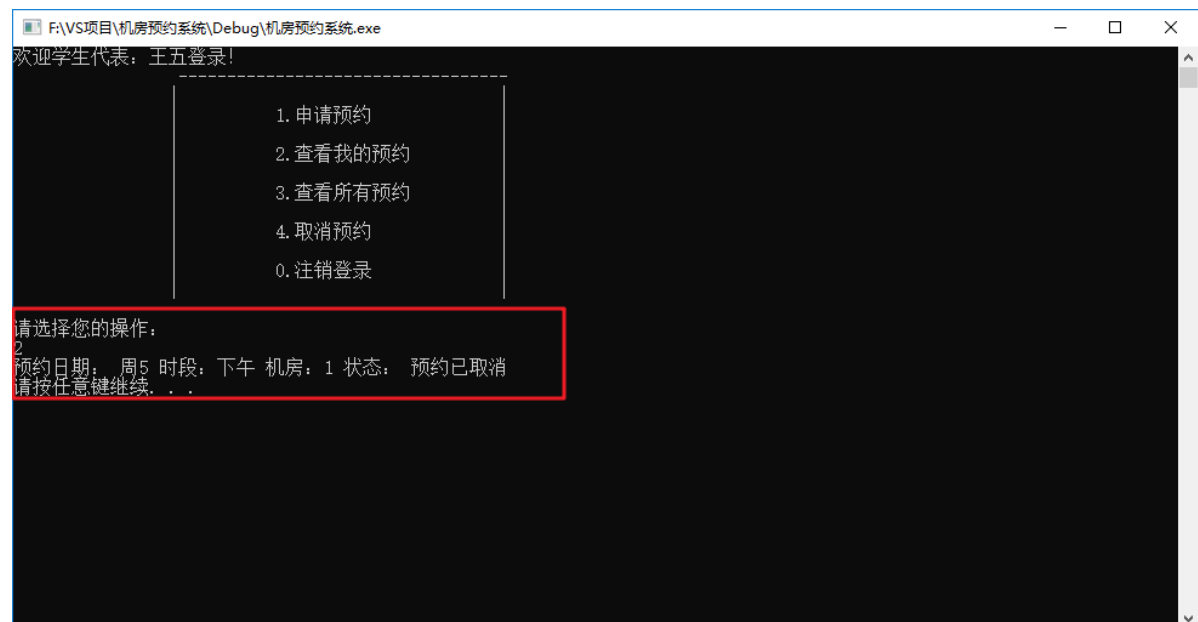
62     }
63     cout << "输入有误，请重新输入" << endl;
64 }
65
66 system("pause");
67 system("cls");
68 }
69

```

测试取消预约：



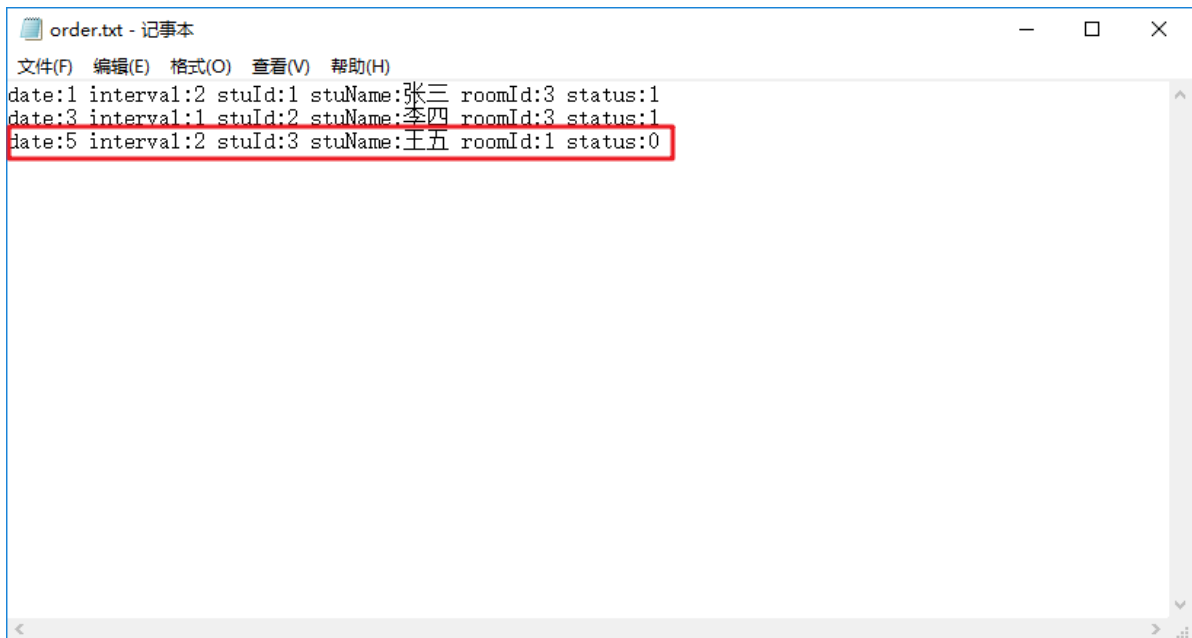
再次查看个人预约记录：



查看所有预约



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

- 在Teacher类的构造函数中，初始化教师信息，代码如下：

```

1 //有参构造 (职工编号, 姓名, 密码)
2 Teacher::Teacher(int empId, string name, string pwd)
3 {
4     //初始化属性
5     this->m_EmpId = empId;
6     this->m_Name = name;
7     this->m_Pwd = pwd;
8 }

```

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:

```

1 //教师菜单
2 void TeacherMenu(Identity * &teacher)
3 {
4     while (true)
5     {
6         //教师菜单
7         teacher->operMenu();
8
9         Teacher* tea = (Teacher*)teacher;
10        int select = 0;
11
12        cin >> select;
13
14        if (select == 1)
15        {
16            //查看所有预约
17            tea->showAllOrder();
18        }
19        else if (select == 2)
20        {
21            //审核预约
22            tea->validOrder();
23        }
24        else
25        {
26            delete teacher;
27            cout << "注销成功" << endl;
28            system("pause");
29            system("cls");
30            return;
31        }
32    }
33 }
34 }

```

9.1.3 菜单功能实现

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

```
1 //教师菜单界面
2 void Teacher::operMenu()
3 {
4     cout << "欢迎教师: " << this->m_Name << "登录! " << endl;
5     cout << "\t\t -----\n";
6     cout << "\t\t|                                     |\n";
7     cout << "\t\t|          1. 查看所有预约          |\n";
8     cout << "\t\t|                                     |\n";
9     cout << "\t\t|          2. 审核预约              |\n";
10    cout << "\t\t|                                     |\n";
11    cout << "\t\t|          0. 注销登录              |\n";
12    cout << "\t\t|                                     |\n";
13    cout << "\t\t -----\n";
14    cout << "请选择您的操作:  " << endl;
15 }
```

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:

```
1 //进入教师子菜单
2 TeacherMenu(person);
```

添加效果如图:

```
orderFile.h student.cpp 机房预约系统.cpp globalFile.h computerRoom.h teacher.cpp*
机房预约系统 (全局范围) LoginIn(string fileName, int type)

192 else if (type == 2)
193 {
194     //教师登录验证
195     int fId;
196     string fName;
197     string fPwd;
198     while (ifs >> fId && ifs >> fName && ifs >> fPwd)
199     {
200         if (id == fId && name == fName && pwd == fPwd)
201         {
202             cout << "教师验证登录成功!" << endl;
203             system("pause");
204             system("cls");
205             person = new Teacher(id, name, pwd);
206             //进入教师子菜单
207             TeacherMenu(person);
208             return;
209         }
210     }
211 }
212 else if (type == 3)
213 {
214     //管理员登录验证
215     string fName;
216     string fPwd;
217     while (ifs >> fName && ifs >> fPwd)
218     {
219         if (name == fName && pwd == fPwd)
220         {
221             cout << "管理员验证登录成功!" << endl;
222             //登录成功后，按任意键进入管理员界面
223             system("pause");
224             system("cls");
```

测试对接，效果如图：

登录验证通过：



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()`

```
1 void Teacher::showAllOrder()  
2 {  
3     OrderFile of;
```



```

4      if (of.m_Size == 0)
5      {
6          cout << "无预约记录" << endl;
7          system("pause");
8          system("cls");
9          return;
10     }
11     for (int i = 0; i < of.m_Size; i++)
12     {
13         cout << i + 1 << "、 ";
14
15         cout << "预约日期: 周" << of.m_orderData[i]["date"];
16         cout << "时段: " << (of.m_orderData[i]["interval"] == "1" ? "上午" :
"下午");
17         cout << "学号: " << of.m_orderData[i]["stuId"];
18         cout << "姓名: " << of.m_orderData[i]["stuName"];
19         cout << "机房: " << of.m_orderData[i]["roomId"];
20         string status = "状态: "; // 0 取消的预约 1 审核中 2 已预约 -1 预约
失败
21         if (of.m_orderData[i]["status"] == "1")
22         {
23             status += "审核中";
24         }
25         else if (of.m_orderData[i]["status"] == "2")
26         {
27             status += "预约成功";
28         }
29         else if (of.m_orderData[i]["status"] == "-1")
30         {
31             status += "审核未通过, 预约失败";
32         }
33         else
34         {
35             status += "预约已取消";
36         }
37         cout << status << endl;
38     }
39
40     system("pause");
41     system("cls");
42 }

```

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：

```
1 //审核预约
2 void Teacher::validOrder()
3 {
4     OrderFile of;
5     if (of.m_Size == 0)
6     {
7         cout << "无预约记录" << endl;
8         system("pause");
9         system("cls");
10        return;
11    }
12    cout << "待审核的预约记录如下: " << endl;
13
14    vector<int>v;
15    int index = 0;
16    for (int i = 0; i < of.m_Size; i++)
17    {
18        if (of.m_orderData[i]["status"] == "1")
19        {
20            v.push_back(i);
21            cout << ++index << "、 ";
22            cout << "预约日期: 周" << of.m_orderData[i]["date"];
23            cout << " 时段: " << (of.m_orderData[i]["interval"] == "1" ? "上
午" : "下午");
24            cout << " 机房: " << of.m_orderData[i]["roomId"];
```

```

25         string status = " 状态: "; // 0取消的预约  1 审核中  2 已预约  -1
    预约失败
26         if (of.m_orderData[i]["status"] == "1")
27         {
28             status += "审核中";
29         }
30         cout << status << endl;
31     }
32 }
33 cout << "请输入审核的预约记录,0代表返回" << endl;
34 int select = 0;
35 int ret = 0;
36 while (true)
37 {
38     cin >> select;
39     if (select >= 0 && select <= v.size())
40     {
41         if (select == 0)
42         {
43             break;
44         }
45         else
46         {
47             cout << "请输入审核结果" << endl;
48             cout << "1、通过" << endl;
49             cout << "2、不通过" << endl;
50             cin >> ret;
51
52             if (ret == 1)
53             {
54                 of.m_orderData[v[select - 1]]["status"] = "2";
55             }
56             else
57             {
58                 of.m_orderData[v[select - 1]]["status"] = "-1";
59             }
60             of.updateOrder();
61             cout << "审核完毕!" << endl;
62             break;
63         }
64     }
65     cout << "输入有误, 请重新输入" << endl;
66 }
67
68 system("pause");
69 system("cls");
70 }

```

9.3.2 测试审核预约

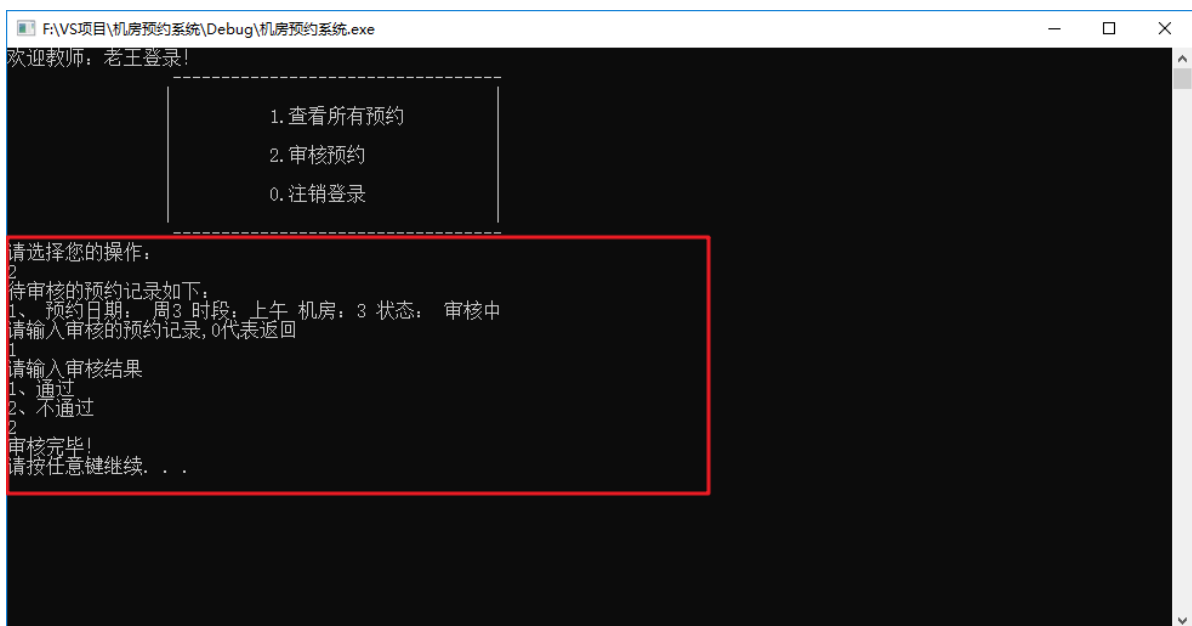
测试 - 审核通过



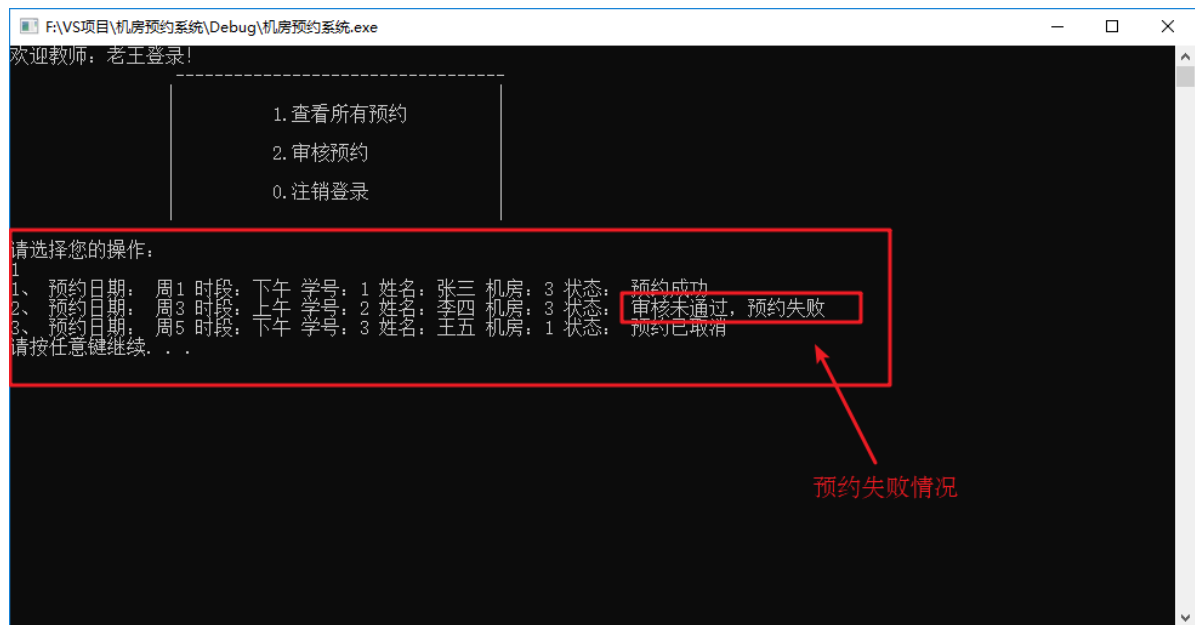
审核通过情况



测试-审核未通过



审核未通过情况:



学生身份下查看记录:



审核预约成功!

至此本案例制作完毕! ^_^