山东大学计算机科学与技术学院

数据结构与算法课程设计报告

学号：202022171214	姓名：刘宇星	班级： 计机 20.1
上机学时： 48	日期：2022/05/23	
课程设计题目：航空客运订票系统		
软件环境：devc++		

报告内容：

1.需求描述

1.1 问题描述

航空客运订票的业务活动包括：查询航线、客票预订和办理退票等。试设计一个航空客运订票系统，以使上述业务可以借助计算机来完成。

1.2 基本要求

民航售票处的计算机系统可以为客户提供下列各项服务：

（1）查询航线：根据旅客提出的终点站名输出下列信息：航班号、飞机号、星期几飞行，最近一天航班的日期和余票额；

（2）承办订票业务：根据客户提出的要求（日期、航班号、订票数额）查询该航班票额情况，若尚有余额，则为客户办理订票手续，输出座位号；若已满员或余票额少于订票额，则需要重新询问客户要求。若需要，可预约登记排队等候。

（3）承办退票业务：根据客户提供的情况（日期、航班、退票数额），为客户办理退票手续，然后查询该航班是否有人预约登记，首先询问排在第一的客户，若所退票额能满足他的要求，则为他办理订票手续，否则依次询问其他排队预约的客户。

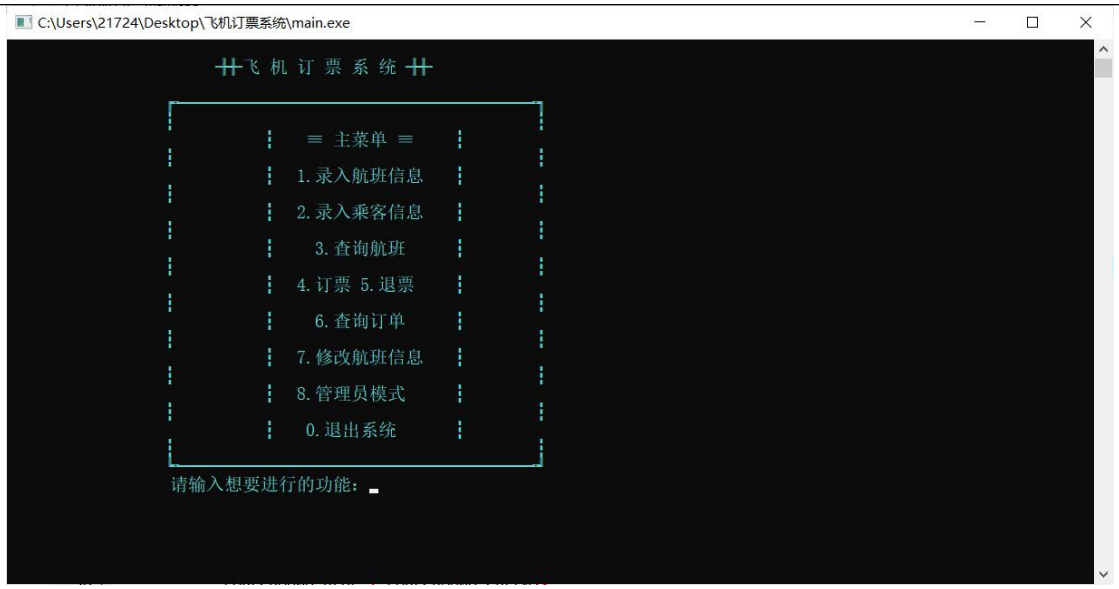
（4）当客户订票要求不能满足时，系统可向客户提供到达同一目的地的其它航线情况。

（5）还可以自己新增其他功能和服务。

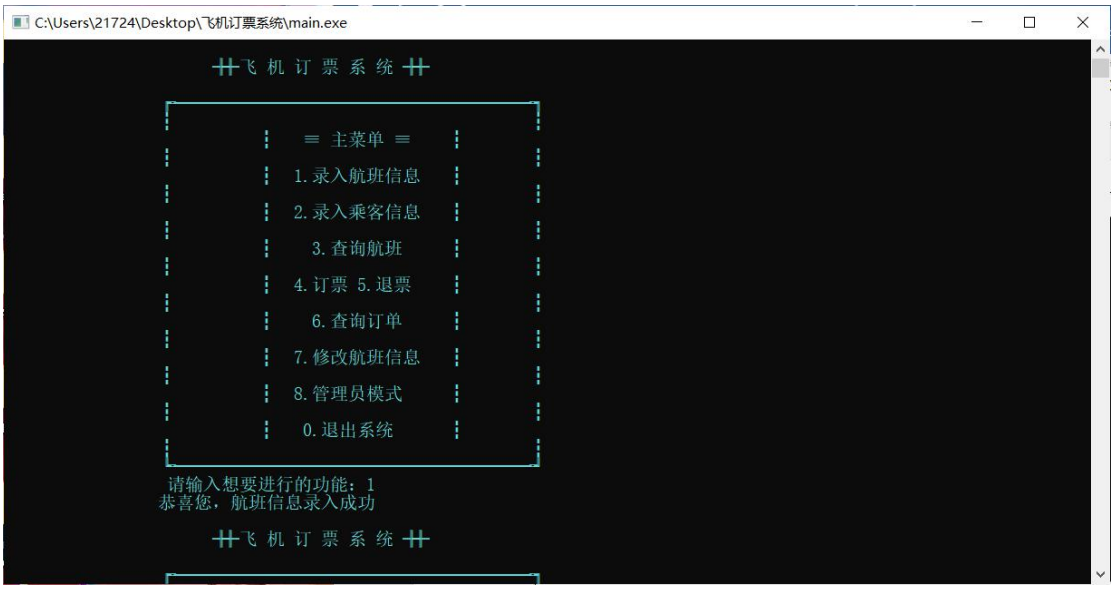
1.3 输入说明

首先是订票系统的界面，如下所示

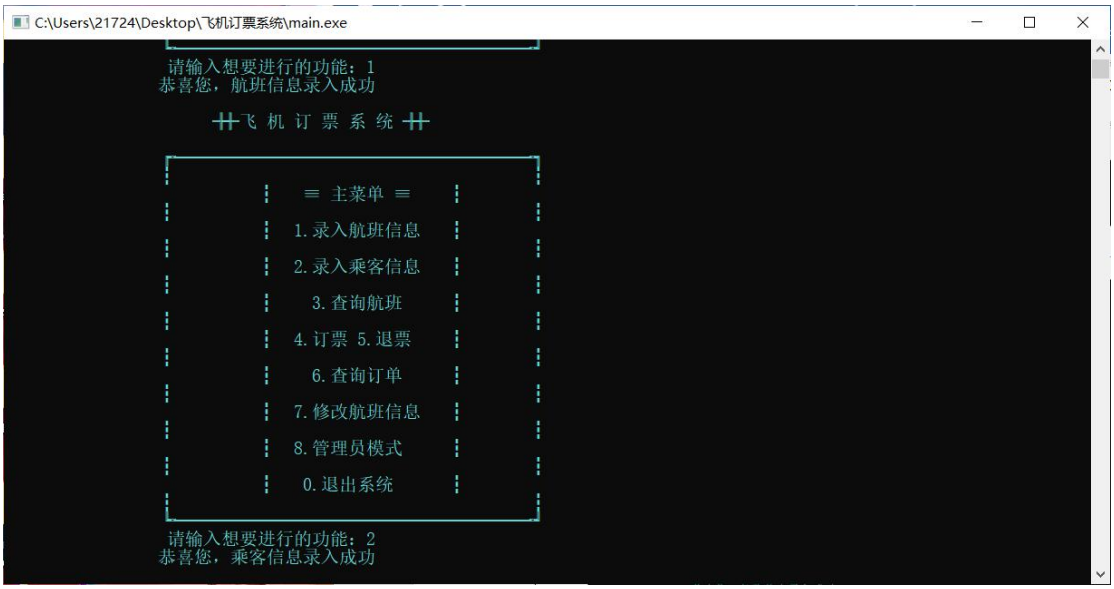
主界面：


C:\Users\21724\Desktop\飞机订票系统\main.exe
飞 机 订 票 系 统
≡ 主菜单 ≡
1. 录入航班信息
2. 录入乘客信息
3. 查询航班
4. 订票 5. 退票
6. 查询订单
7. 修改航班信息
8. 管理员模式
0. 退出系统
请输入想要进行的功能：

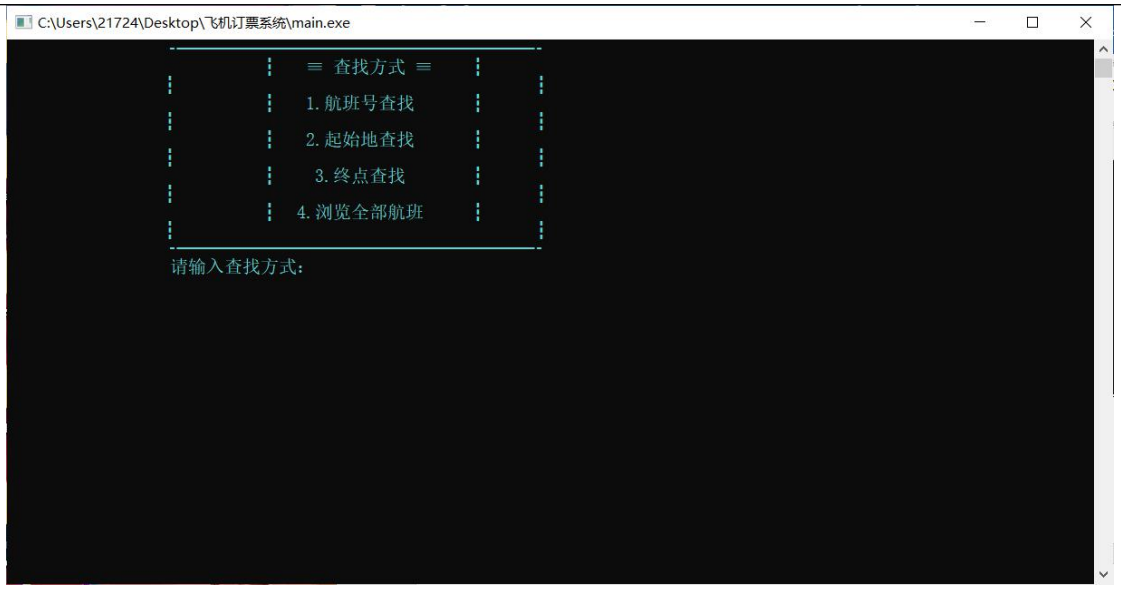
操作 1：


C:\Users\21724\Desktop\飞机订票系统\main.exe
飞 机 订 票 系 统
≡ 主菜单 ≡
1. 录入航班信息
2. 录入乘客信息
3. 查询航班
4. 订票 5. 退票
6. 查询订单
7. 修改航班信息
8. 管理员模式
0. 退出系统
请输入想要进行的功能：1
恭喜您，航班信息录入成功
飞 机 订 票 系 统

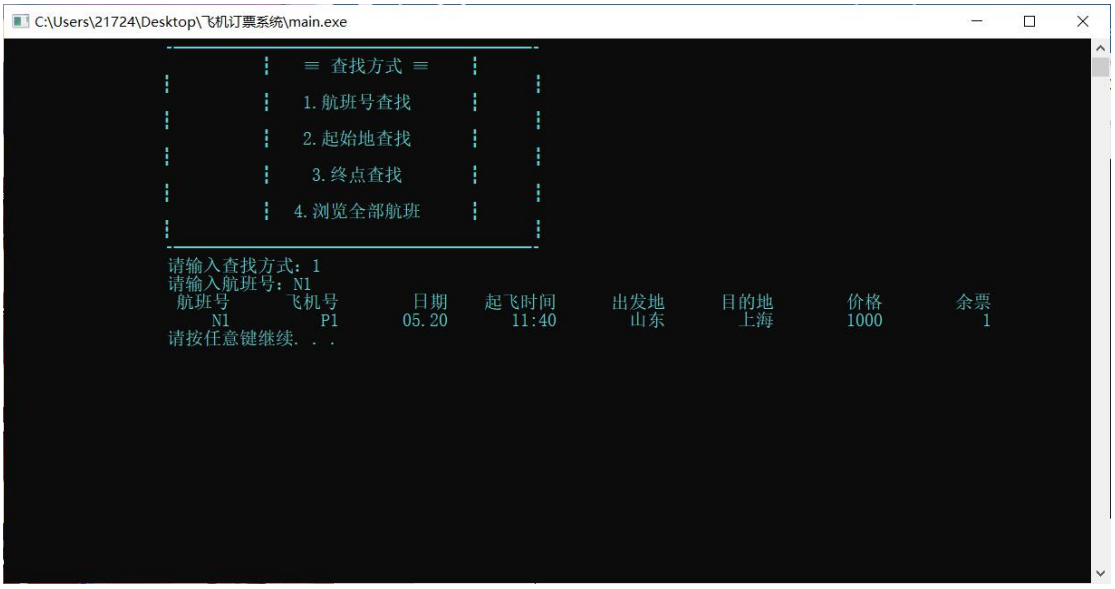
操作 2：

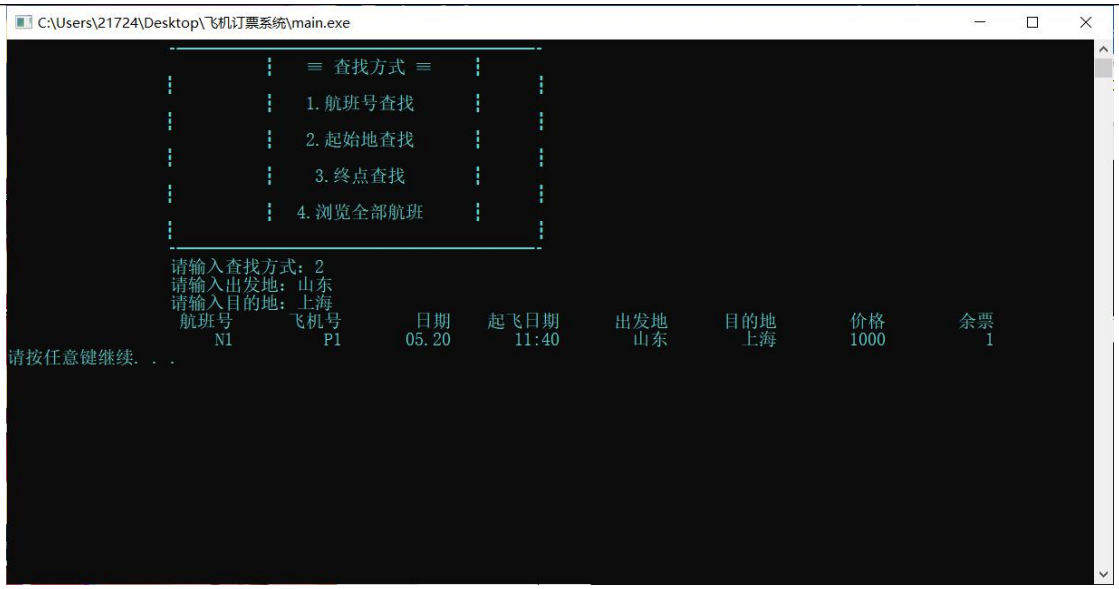

C:\Users\21724\Desktop\飞机订票系统\main.exe
请输入想要进行的功能：1
恭喜您，航班信息录入成功
飞 机 订 票 系 统
≡ 主菜单 ≡
1. 录入航班信息
2. 录入乘客信息
3. 查询航班
4. 订票 5. 退票
6. 查询订单
7. 修改航班信息
8. 管理员模式
0. 退出系统
请输入想要进行的功能：2
恭喜您，乘客信息录入成功

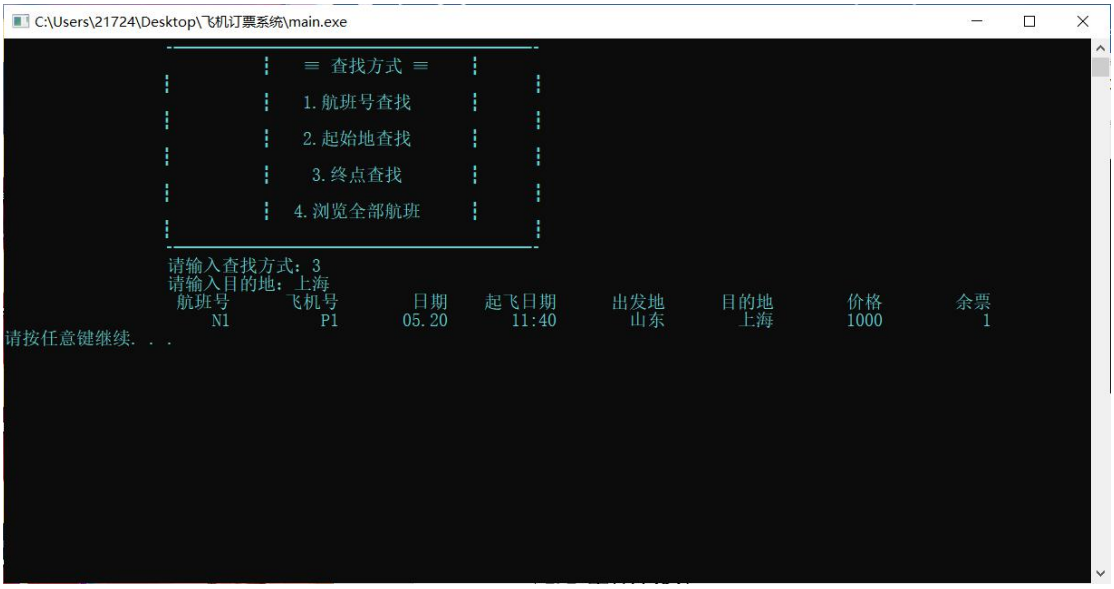
操作 3：


C:\Users\21724\Desktop\飞机订票系统\main.exe
═══ 查找方式 ═══
1. 航班号查找
2. 起始地查找
3. 终点查找
4. 浏览全部航班
请输入查找方式：

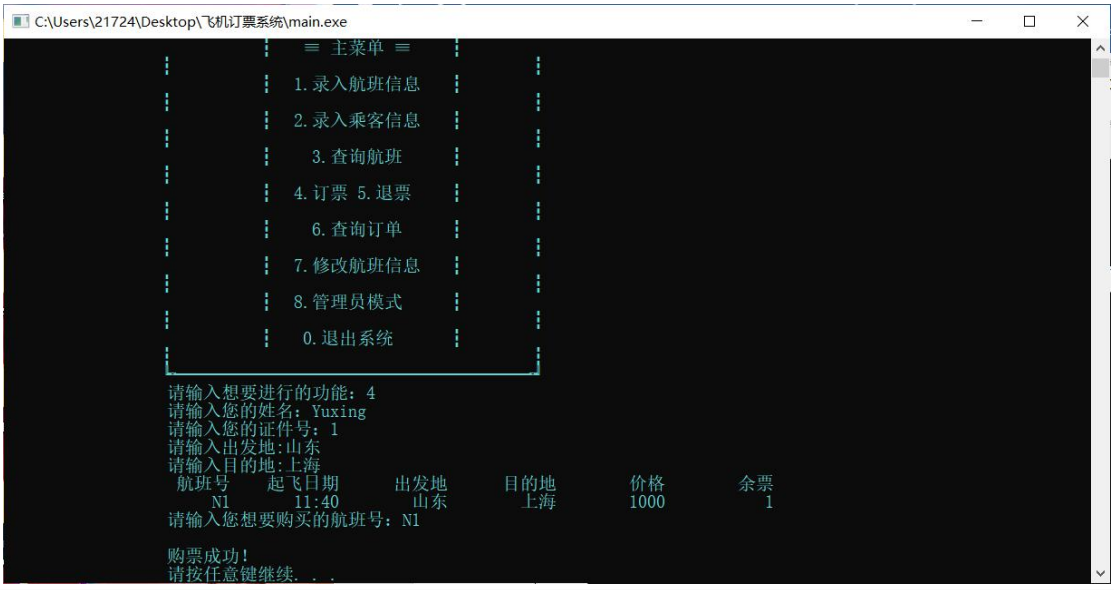

C:\Users\21724\Desktop\飞机订票系统\main.exe
═══ 查找方式 ═══
1. 航班号查找
2. 起始地查找
3. 终点查找
4. 浏览全部航班
请输入查找方式：4
航班号 飞机号 日期 起飞日期 出发地 目的地 价格 余票
N1 P1 05.20 11:40 山东 上海 1000 1
╪飞 机 订 票 系 统╪
═══ 主菜单 ═══
1. 录入航班信息
2. 录入乘客信息
3. 查询航班
4. 订票 5. 退票


C:\Users\21724\Desktop\飞机订票系统\main.exe
═══ 查找方式 ═══
1. 航班号查找
2. 起始地查找
3. 终点查找
4. 浏览全部航班
请输入查找方式：1
请输入航班号：N1
航班号 飞机号 日期 起飞时间 出发地 目的地 价格 余票
N1 P1 05.20 11:40 山东 上海 1000 1
请按任意键继续. . .

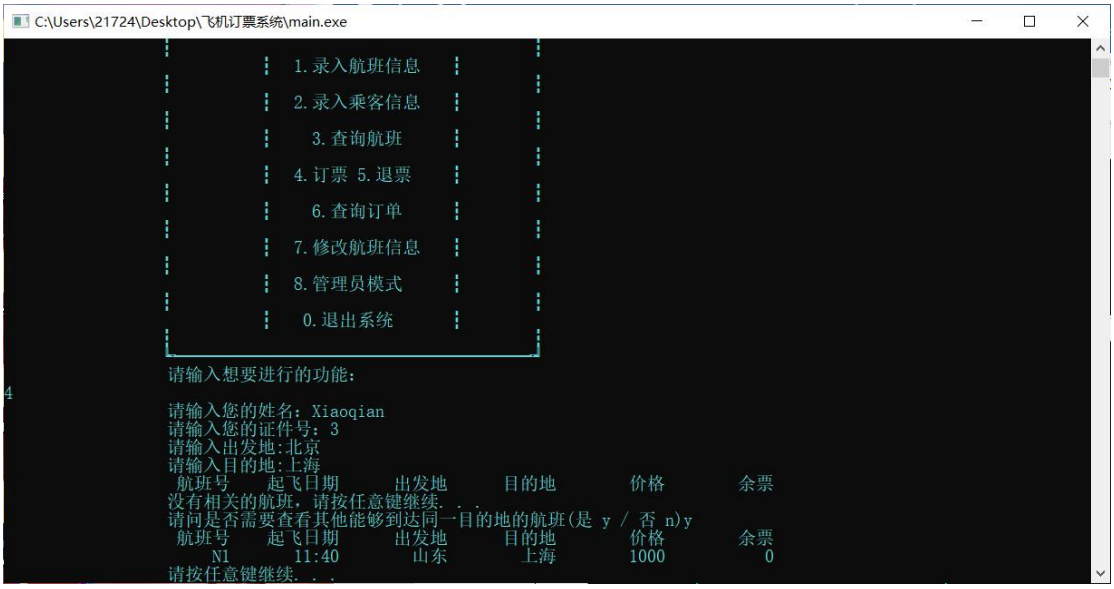

C:\Users\21724\Desktop\飞机订票系统\main.exe
≡ 查找方式 ≡
1. 航班号查找
2. 起始地查找
3. 终点查找
4. 浏览全部航班
请输入查找方式：2
请输入出发地：山东
请输入目的地：上海
航班号 飞机号 日期 起飞日期 出发地 目的地 价格 余票
N1 P1 05.20 11:40 山东 上海 1000 1
请按任意键继续. . .


C:\Users\21724\Desktop\飞机订票系统\main.exe
≡ 查找方式 ≡
1. 航班号查找
2. 起始地查找
3. 终点查找
4. 浏览全部航班
请输入查找方式：3
请输入目的地：上海
航班号 飞机号 日期 起飞日期 出发地 目的地 价格 余票
N1 P1 05.20 11:40 山东 上海 1000 1
请按任意键继续. . .

操作 4：

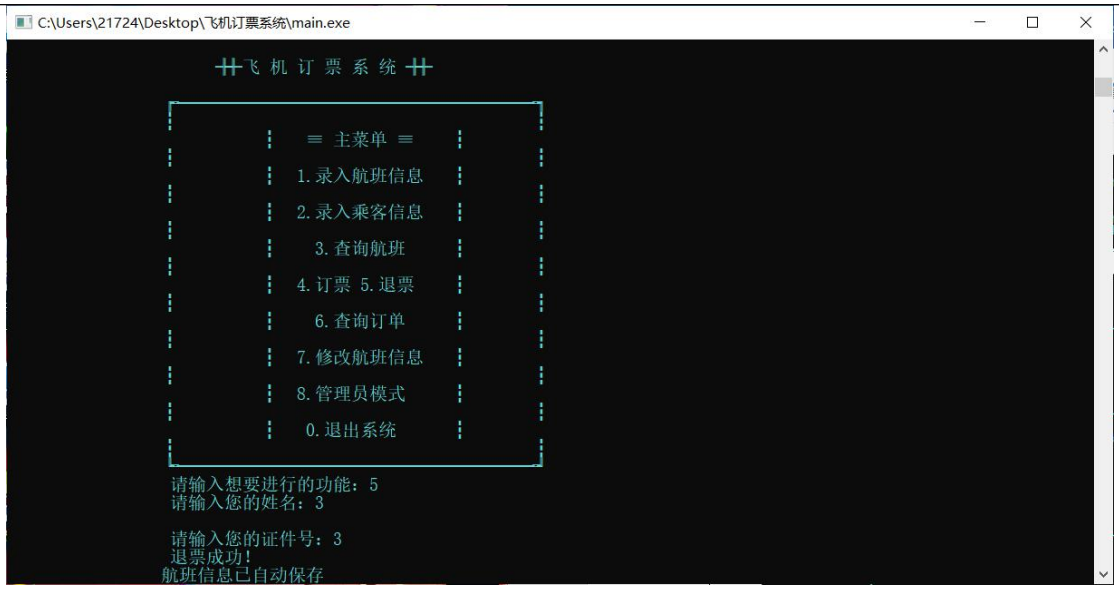

C:\Users\21724\Desktop\飞机订票系统\main.exe
≡ 主菜单 ≡
1. 录入航班信息
2. 录入乘客信息
3. 查询航班
4. 订票 5. 退票
6. 查询订单
7. 修改航班信息
8. 管理员模式
0. 退出系统
请输入想要进行的功能：4
请输入您的姓名：Yuxing
请输入您的证件号：1
请输入出发地:山东
请输入目的地:上海
航班号 起飞日期 出发地 目的地 价格 余票
N1 11:40 山东 上海 1000 1
请输入您想要购买的航班号：N1
购票成功!
请按任意键继续. . .


C:\Users\21724\Desktop\飞机订票系统\main.exe
1.录入航班信息
2.录入乘客信息
3.查询航班
4.订票 5.退票
6.查询订单
7.修改航班信息
8.管理员模式
0.退出系统
请输入想要进行的功能：4
请输入您的姓名：Shaoqing
请输入您的证件号：2
请输入出发地:山东
请输入目的地:上海
航班号 起飞日期 出发地 目的地 价格 余票
N1 11:40 山东 上海 1000 0
请输入您想要购买的航班号：N1
余票不足，请选择是否预约（是 y / 否 n）y
预约成功!
请按任意键继续. . .


C:\Users\21724\Desktop\飞机订票系统\main.exe
1.录入航班信息
2.录入乘客信息
3.查询航班
4.订票 5.退票
6.查询订单
7.修改航班信息
8.管理员模式
0.退出系统
请输入想要进行的功能：
4
请输入您的姓名：Xiaoqian
请输入您的证件号：3
请输入出发地:北京
请输入目的地:上海
航班号 起飞日期 出发地 目的地 价格 余票
没有相关的航班，请按任意键继续. . .
请问是否需要查看其他能够到达同一目的地的航班(是 y / 否 n)y
航班号 起飞日期 出发地 目的地 价格 余票
N1 11:40 山东 上海 1000 0
请按任意键继续. . .

操作 5：

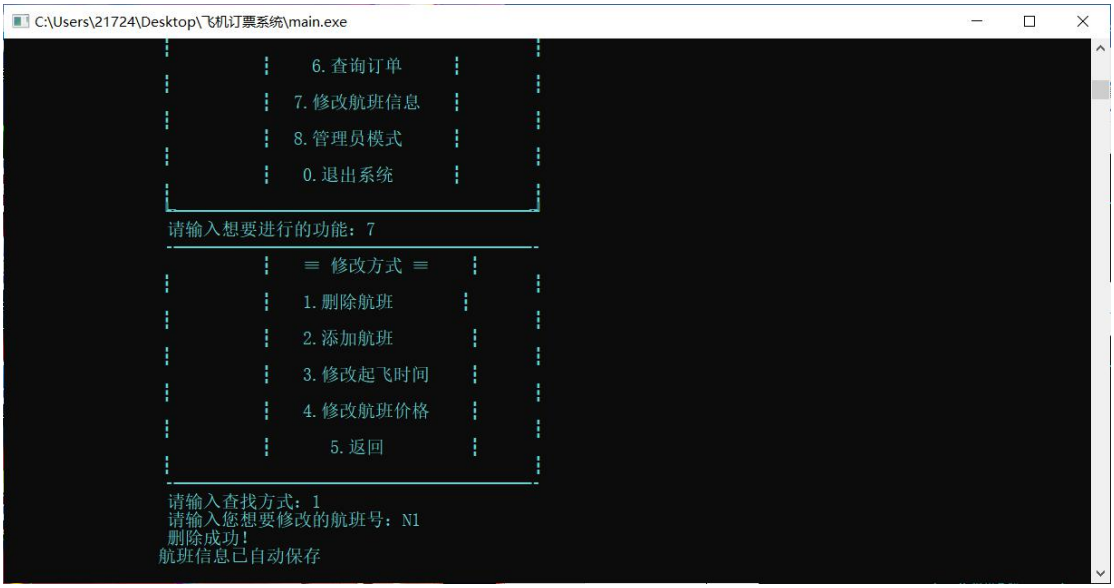

C:\Users\21724\Desktop\飞机订票系统\main.exe
请输入用户名：root1
请输入密码：root1
管理员权限获取成功!
≡ 管理员模式 ≡
1.查询顾客信息
2.返回上一步
输入想要进行的功能：1
预约状态? 乘客姓名 乘客ID 乘坐航班号 座次 航班时间 始发点 终点 票价
12850544 3 3 N1 0 11:40 山东 上海 1000
0 Yuxing 1 N1 1 11:40 山东 上海 1000
1 Shaoqing 2 N1 0 11:40 山东 上海 1000
≡ 管理员模式 ≡
1.查询顾客信息
2.返回上一步
输入想要进行的功能：

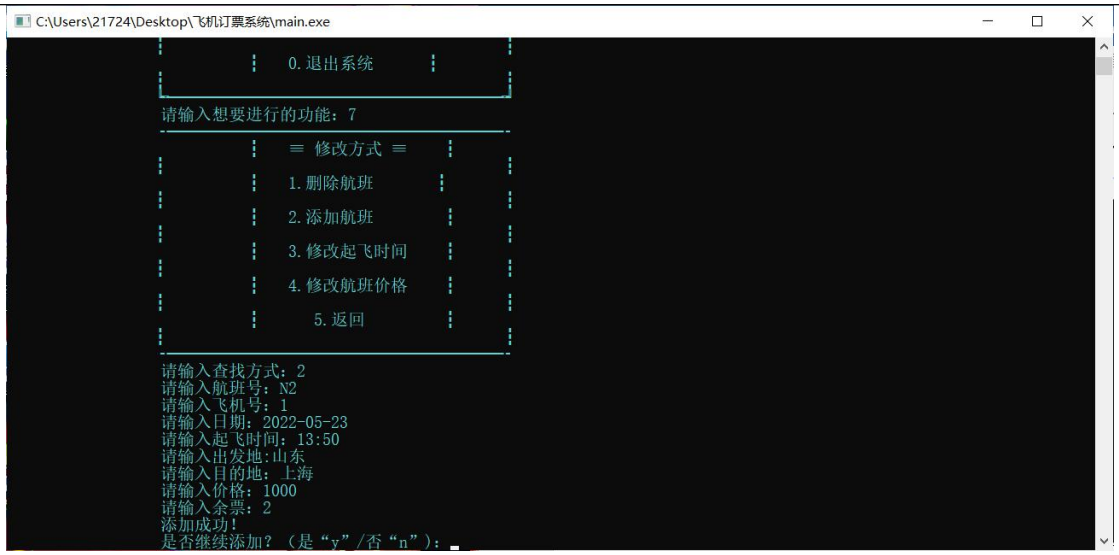

C:\Users\21724\Desktop\飞机订票系统\main.exe
飞 机 订 票 系 统
≡ 主菜单 ≡
1.录入航班信息
2.录入乘客信息
3.查询航班
4.订票 5.退票
6.查询订单
7.修改航班信息
8.管理员模式
0.退出系统
请输入想要进行的功能：5
请输入您的姓名：3
请输入您的证件号：3
退票成功!
航班信息已自动保存

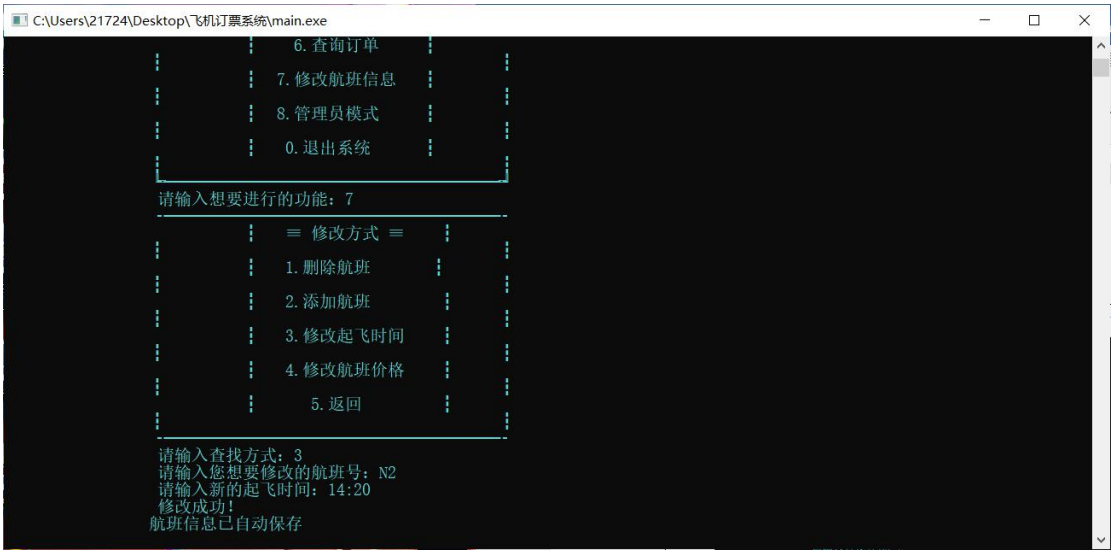
操作 6：

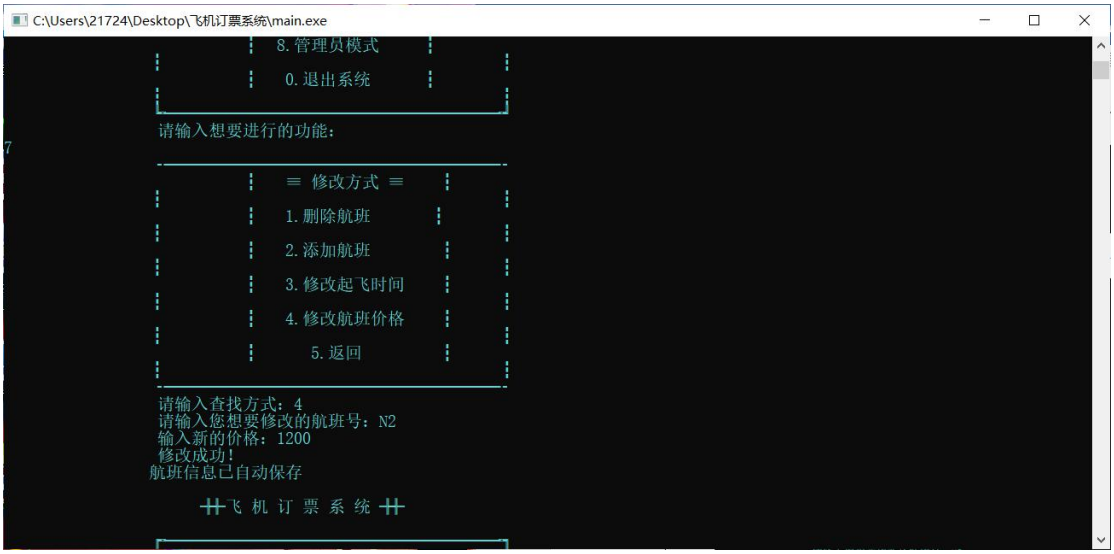

C:\Users\21724\Desktop\飞机订票系统\main.exe
飞 机 订 票 系 统
≡ 主菜单 ≡
1.录入航班信息
2.录入乘客信息
3.查询航班
4.订票 5.退票
6.查询订单
7.修改航班信息
8.管理员模式
0.退出系统
请输入想要进行的功能：6
请输入您的姓名：Shaoqing
请输入您的证件号：2
姓名 航班号 座次 起飞日期 出发地 目的地 价格
目前乘客已成功订票，乘客信息如下所示
Shaoqing N1 0 11:40 山东 上海 1000
请按任意键继续. . .

操作 7：

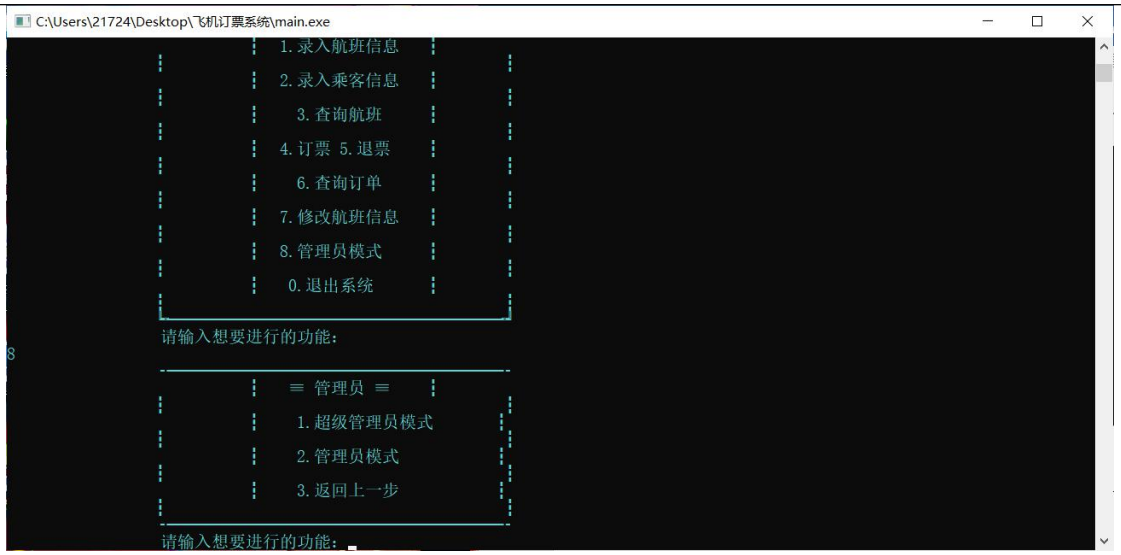

C:\Users\21724\Desktop\飞机订票系统\main.exe
6.查询订单
7.修改航班信息
8.管理员模式
0.退出系统
请输入想要进行的功能：7
≡ 修改方式 ≡
1.删除航班
2.添加航班
3.修改起飞时间
4.修改航班价格
5.返回
请输入查找方式：1
请输入您想要修改的航班号：N1
删除成功!
航班信息已自动保存


C:\Users\21724\Desktop\飞机订票系统\main.exe
0.退出系统
请输入想要进行的功能：7
≡ 修改方式 ≡
1.删除航班
2.添加航班
3.修改起飞时间
4.修改航班价格
5.返回
请输入查找方式：2
请输入航班号：N2
请输入飞机号：1
请输入日期：2022-05-23
请输入起飞时间：13:50
请输入出发地:山东
请输入目的地：上海
请输入价格：1000
请输入余票：2
添加成功!
是否继续添加？（是“y”/否“n”）：

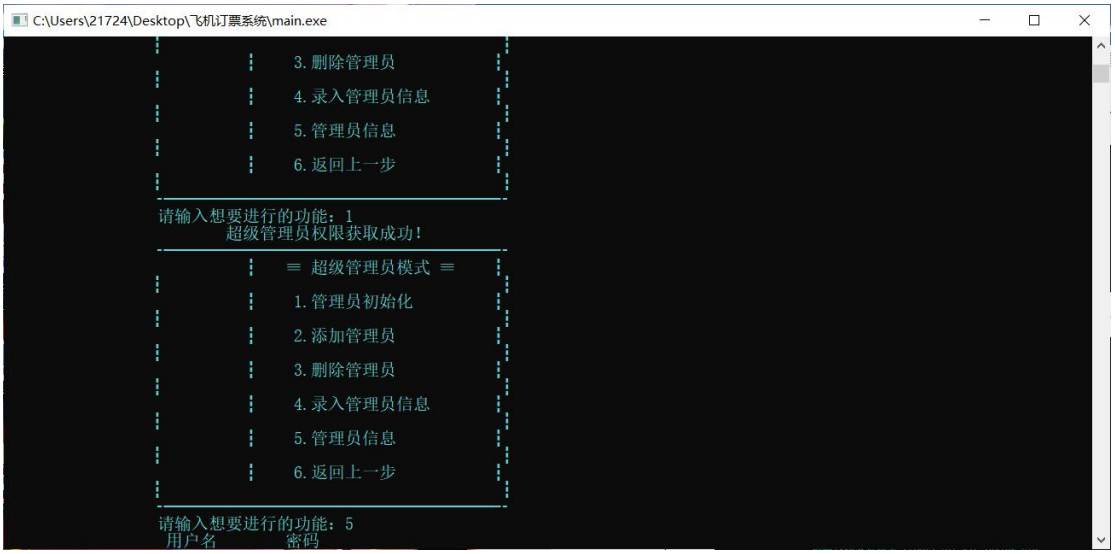

C:\Users\21724\Desktop\飞机订票系统\main.exe
6.查询订单
7.修改航班信息
8.管理员模式
0.退出系统
请输入想要进行的功能：7
≡ 修改方式 ≡
1.删除航班
2.添加航班
3.修改起飞时间
4.修改航班价格
5.返回
请输入查找方式：3
请输入您想要修改的航班号：N2
请输入新的起飞时间：14:20
修改成功!
航班信息已自动保存

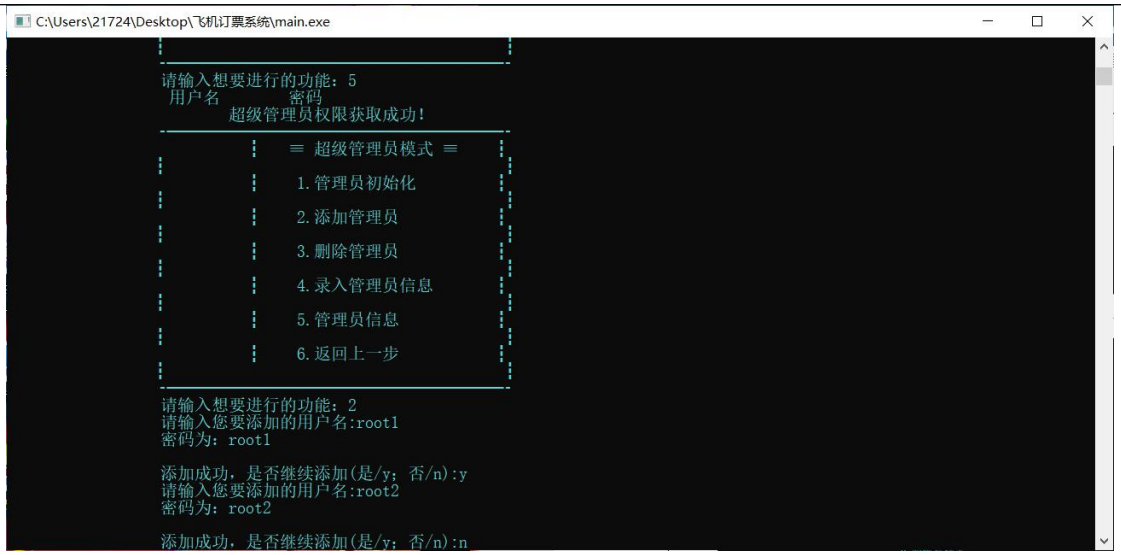

C:\Users\21724\Desktop\飞机订票系统\main.exe
8.管理员模式
0.退出系统
请输入想要进行的功能：
7
≡ 修改方式 ≡
1.删除航班
2.添加航班
3.修改起飞时间
4.修改航班价格
5.返回
请输入查找方式：4
请输入您想要修改的航班号：N2
输入新的价格：1200
修改成功!
航班信息已自动保存
飞机订票系统

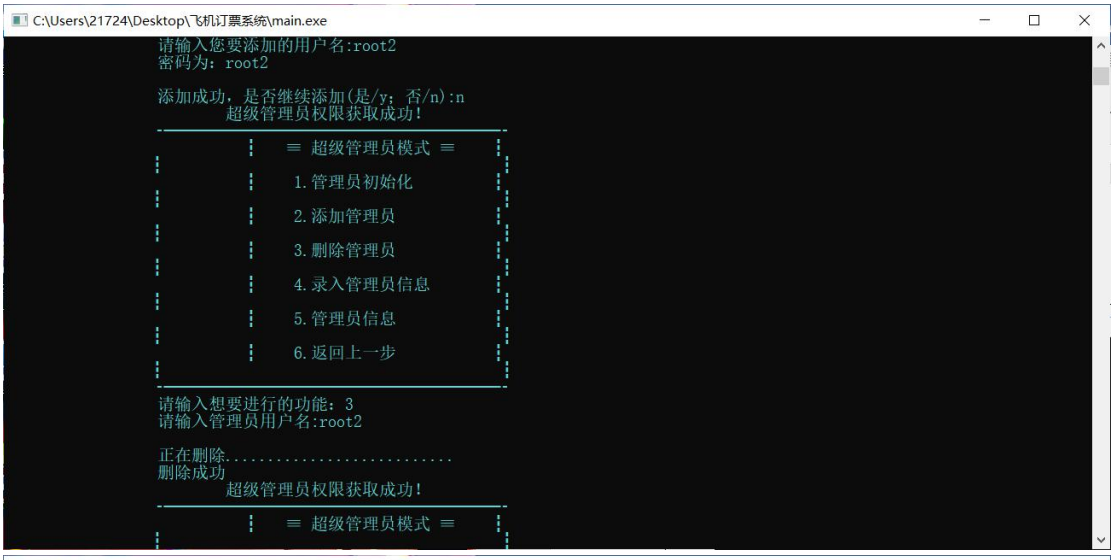
操作 8：

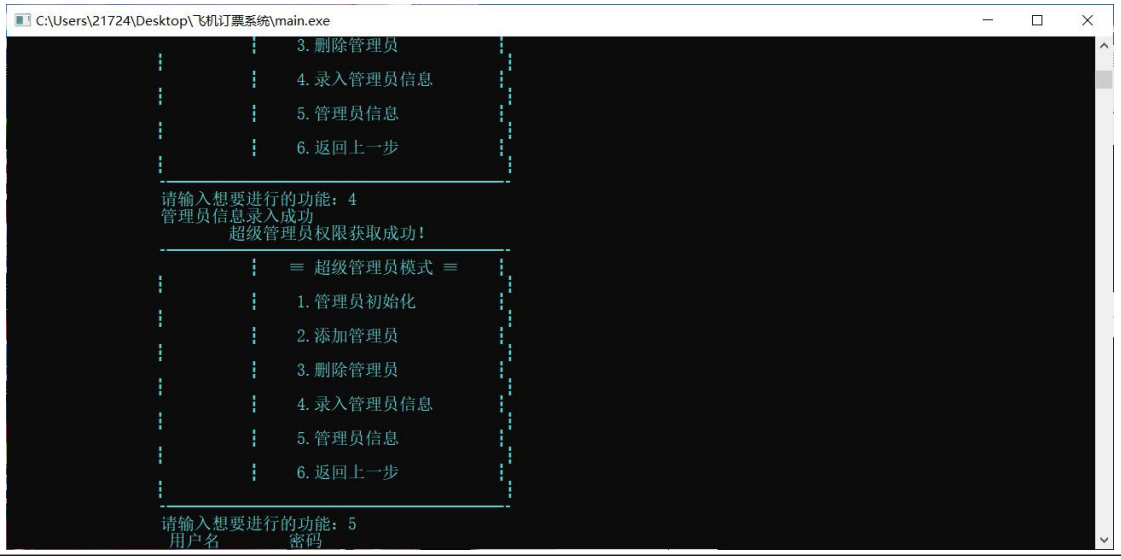

C:\Users\21724\Desktop\飞机订票系统\main.exe
1. 录入航班信息
2. 录入乘客信息
3. 查询航班
4. 订票 5. 退票
6. 查询订单
7. 修改航班信息
8. 管理员模式
0. 退出系统
请输入想要进行的功能：
8
≡ 管理员 ≡
1. 超级管理员模式
2. 管理员模式
3. 返回上一步
请输入想要进行的功能：

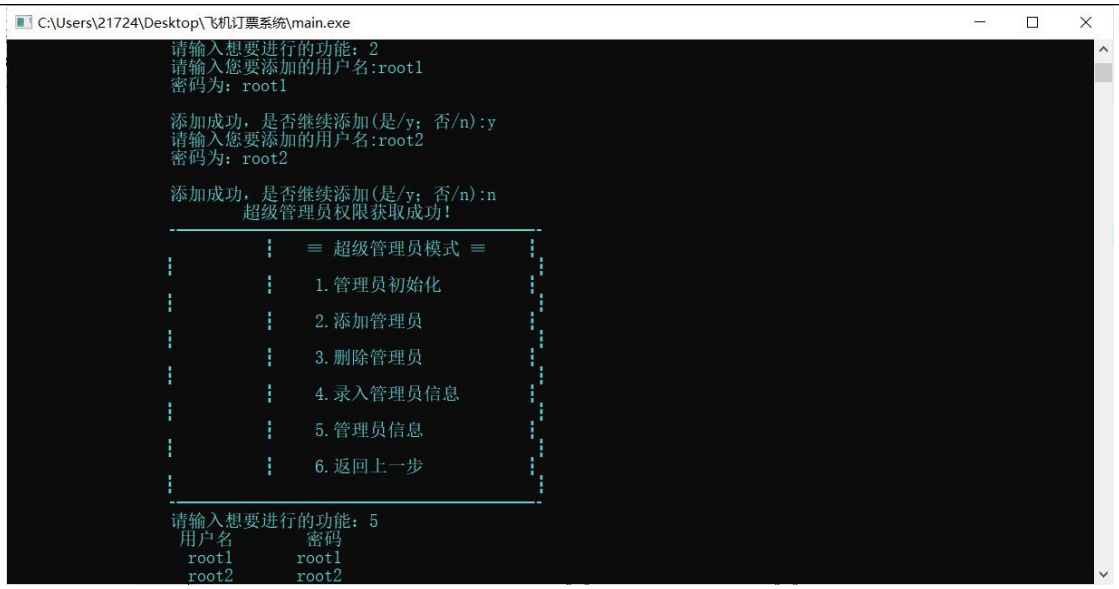

C:\Users\21724\Desktop\飞机订票系统\main.exe
8. 管理员模式
0. 退出系统
请输入想要进行的功能：
8
≡ 管理员 ≡
1. 超级管理员模式
2. 管理员模式
3. 返回上一步
请输入想要进行的功能：1
请输入超级管理员（root）用户名：Yuxing_Liu
请输入密码:20020316
超级管理员权限获取成功!
≡ 超级管理员模式 ≡
1. 管理员初始化

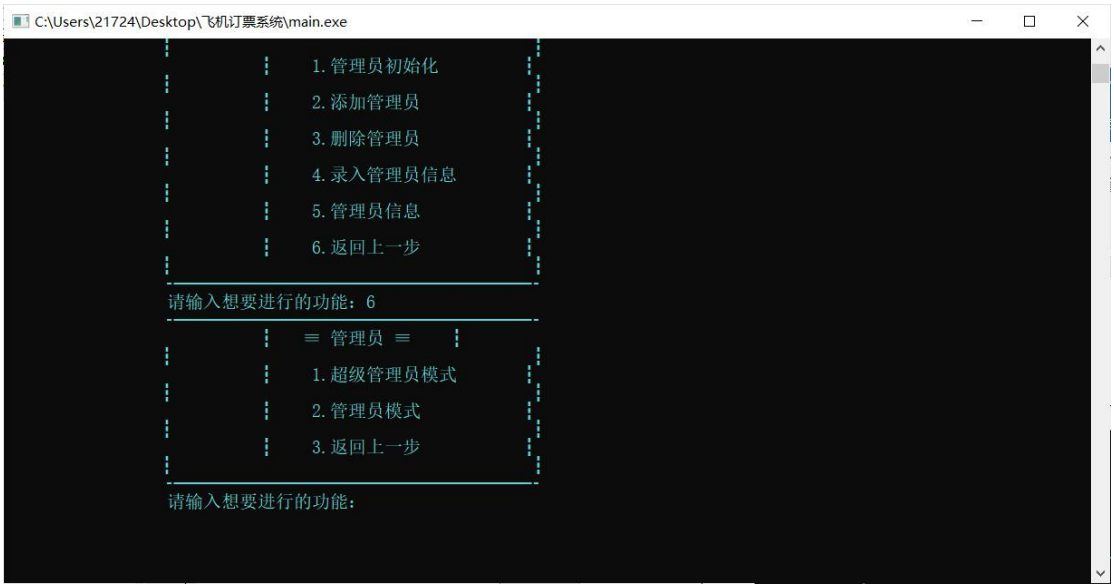

C:\Users\21724\Desktop\飞机订票系统\main.exe
3. 删除管理员
4. 录入管理员信息
5. 管理员信息
6. 返回上一步
请输入想要进行的功能：1
超级管理员权限获取成功!
≡ 超级管理员模式 ≡
1. 管理员初始化
2. 添加管理员
3. 删除管理员
4. 录入管理员信息
5. 管理员信息
6. 返回上一步
请输入想要进行的功能：5
用户名 密码

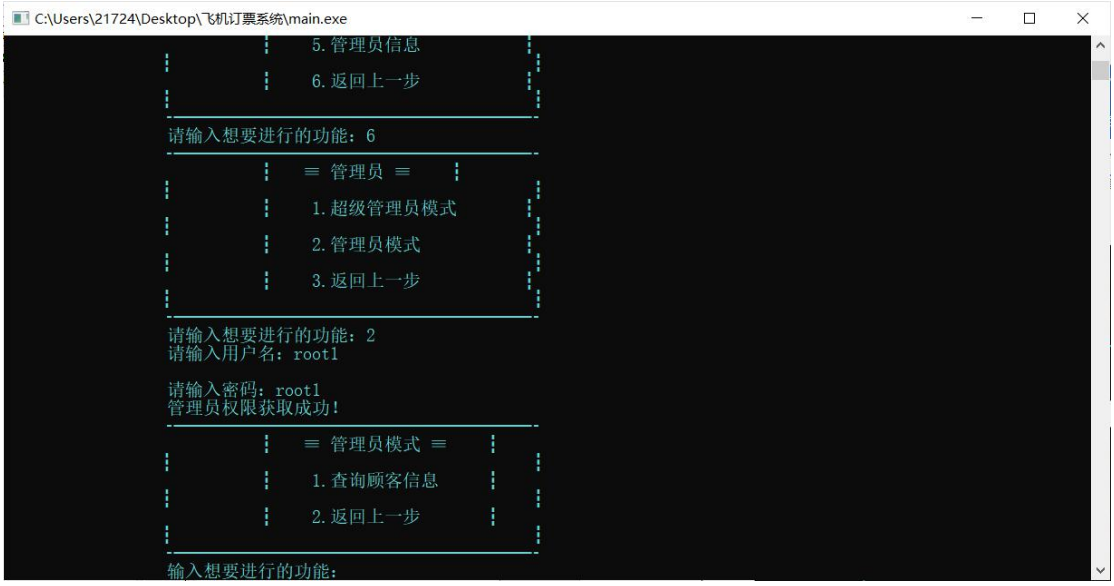

C:\Users\21724\Desktop\飞机订票系统\main.exe
请输入想要进行的功能：5
用户名 密码
超级管理员权限获取成功!
≡ 超级管理员模式 ≡
1.管理员初始化
2.添加管理员
3.删除管理员
4.录入管理员信息
5.管理员信息
6.返回上一步
请输入想要进行的功能：2
请输入您要添加的用户名:root1
密码为：root1
添加成功，是否继续添加(是/y；否/n):y
请输入您要添加的用户名:root2
密码为：root2
添加成功，是否继续添加(是/y；否/n):n

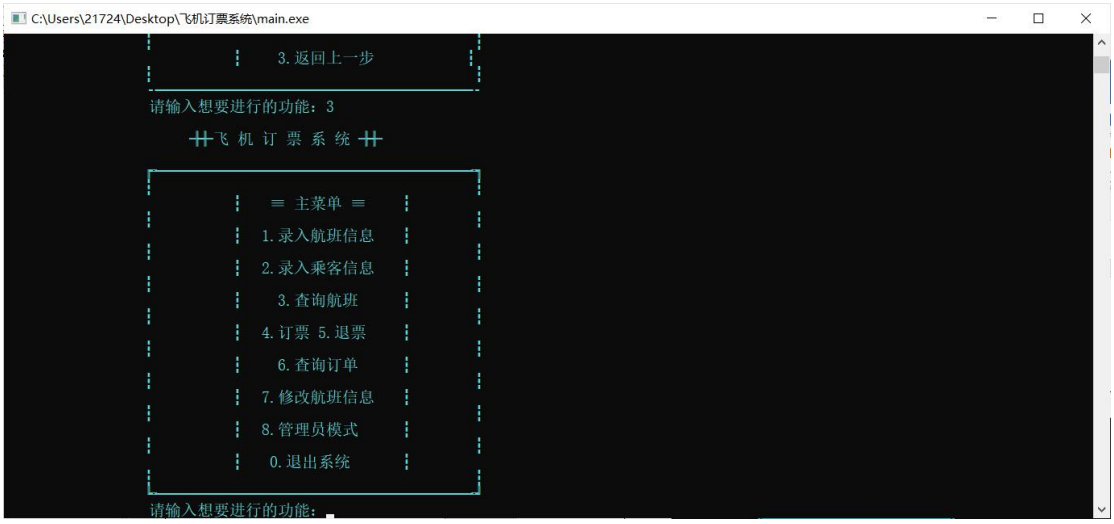

C:\Users\21724\Desktop\飞机订票系统\main.exe
请输入您要添加的用户名:root2
密码为：root2
添加成功，是否继续添加(是/y；否/n):n
超级管理员权限获取成功!
≡ 超级管理员模式 ≡
1.管理员初始化
2.添加管理员
3.删除管理员
4.录入管理员信息
5.管理员信息
6.返回上一步
请输入想要进行的功能：3
请输入管理员用户名:root2
正在删除.....
删除成功
超级管理员权限获取成功!
≡ 超级管理员模式 ≡


C:\Users\21724\Desktop\飞机订票系统\main.exe
3.删除管理员
4.录入管理员信息
5.管理员信息
6.返回上一步
请输入想要进行的功能：4
管理员信息录入成功
超级管理员权限获取成功!
≡ 超级管理员模式 ≡
1.管理员初始化
2.添加管理员
3.删除管理员
4.录入管理员信息
5.管理员信息
6.返回上一步
请输入想要进行的功能：5
用户名 密码

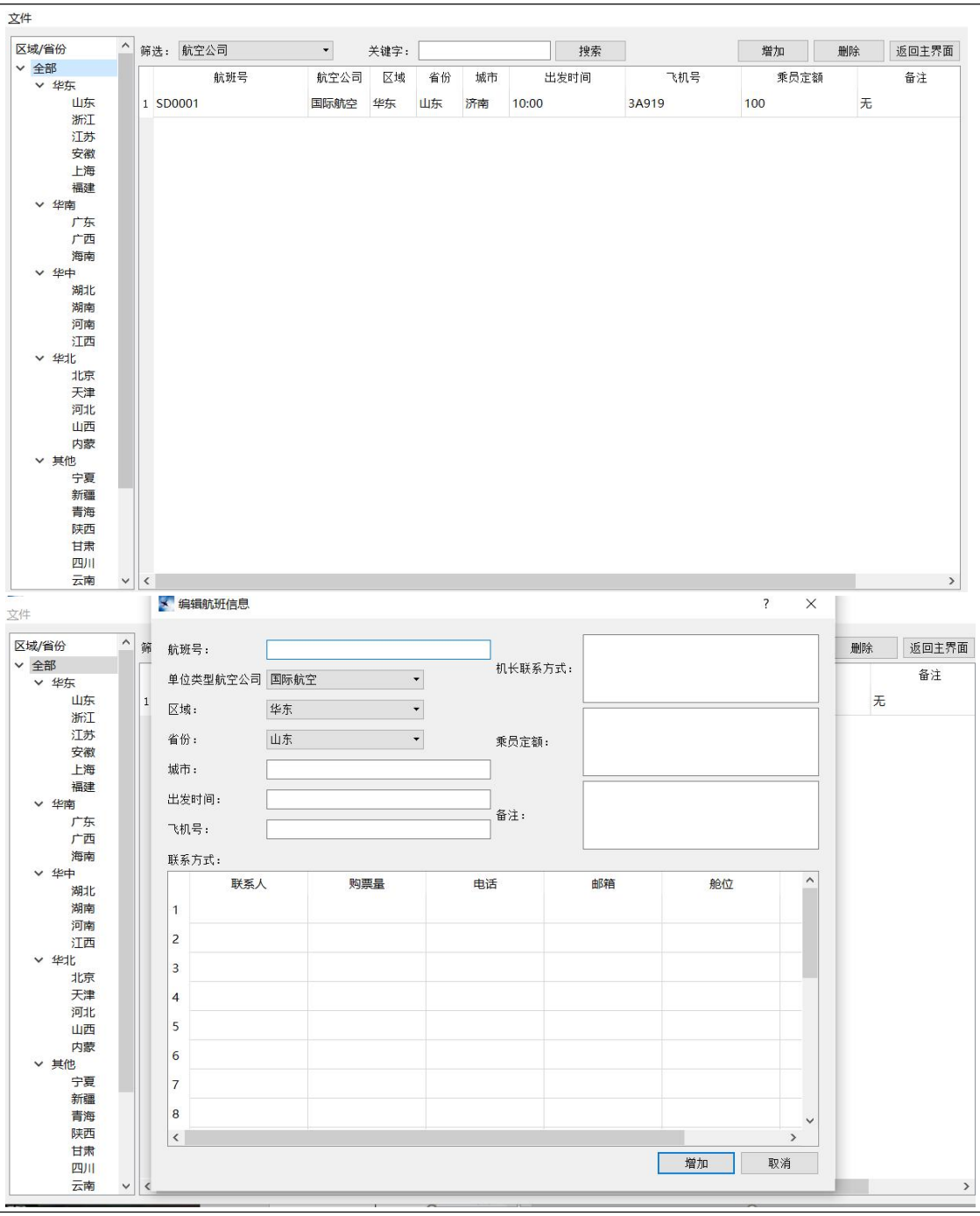

C:\Users\21724\Desktop\飞机订票系统\main.exe
请输入想要进行的功能：2
请输入您要添加的用户名:root1
密码为：root1
添加成功，是否继续添加(是/y；否/n):y
请输入您要添加的用户名:root2
密码为：root2
添加成功，是否继续添加(是/y；否/n):n
超级管理员权限获取成功!
≡ 超级管理员模式 ≡
1.管理员初始化
2.添加管理员
3.删除管理员
4.录入管理员信息
5.管理员信息
6.返回上一步
请输入想要进行的功能：5
用户名 密码
root1 root1
root2 root2

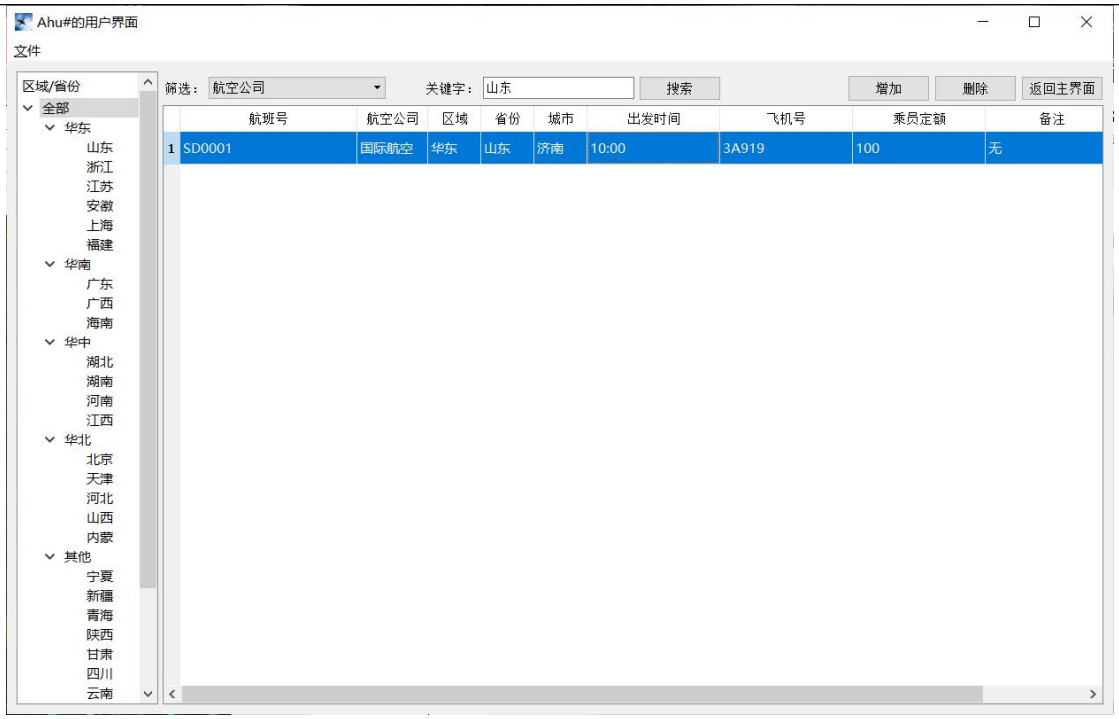

C:\Users\21724\Desktop\飞机订票系统\main.exe
1.管理员初始化
2.添加管理员
3.删除管理员
4.录入管理员信息
5.管理员信息
6.返回上一步
请输入想要进行的功能：6
≡ 管理员 ≡
1.超级管理员模式
2.管理员模式
3.返回上一步
请输入想要进行的功能：


C:\Users\21724\Desktop\飞机订票系统\main.exe
5.管理员信息
6.返回上一步
请输入想要进行的功能：6
≡ 管理员 ≡
1.超级管理员模式
2.管理员模式
3.返回上一步
请输入想要进行的功能：2
请输入用户名：root1
请输入密码：root1
管理员权限获取成功!
≡ 管理员模式 ≡
1.查询顾客信息
2.返回上一步
输入想要进行的功能：



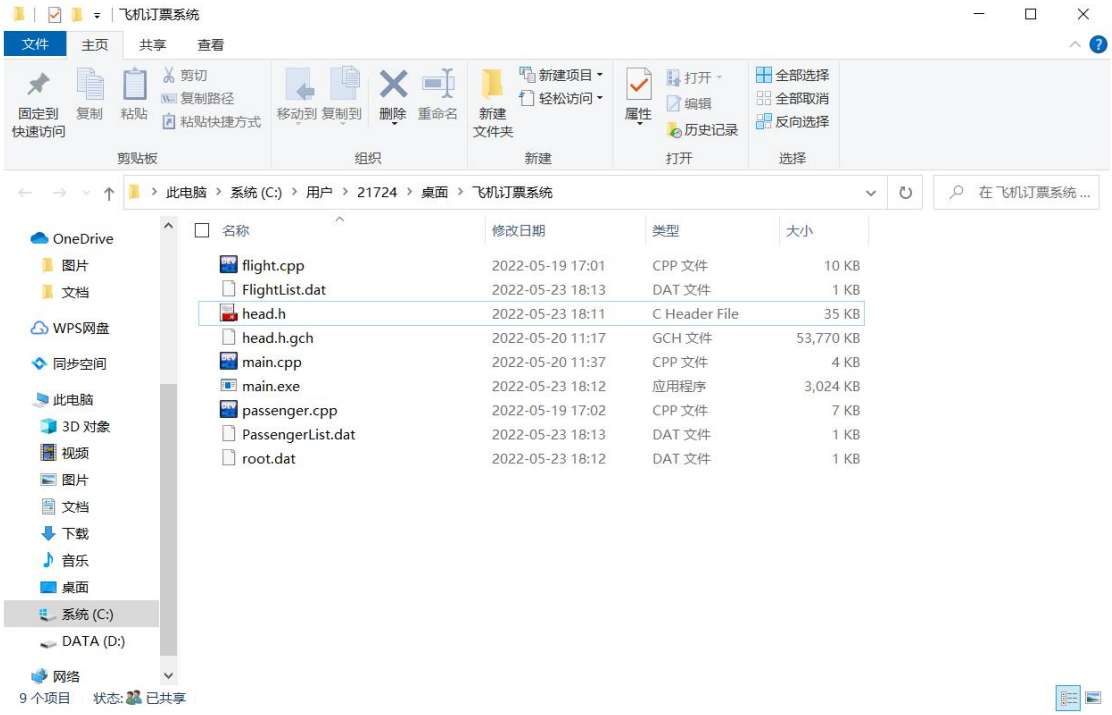
除了订票系统外，还使用 QT 做了一款用来对数据进行可视化的程序。


文件
区域/省份
全部
华东
山东
浙江
江苏
安徽
上海
福建
华南
广东
广西
海南
华中
湖北
湖南
河南
江西
华北
北京
天津
河北
山西
内蒙
其他
宁夏
新疆
青海
陕西
甘肃
四川
云南
筛选：
航空公司
关键字：
搜索
增加
删除
返回主界面
航班号
航空公司
区域
省份
城市
出发时间
飞机号
乘员定额
备注
1
SD0001
国际航空
华东
山东
济南
10:00
3A919
100
无
编辑航班信息
航班号：
单位类型航空公司
国际航空
区域：
华东
省份：
山东
城市：
出发时间：
飞机号：
机长联系方式：
乘员定额：
备注：
联系方式：
联系人
购票量
电话
邮箱
舱位
增加
取消



1.4 输出说明

如图所示。同时还会生成一个.dat 的文件，用来存储上次操作的数据，这样每次还能直接录入之前操作过的信息，更加具有健壮性。



其他在输入中皆有体现。

2. 分析与设计

2.1 问题分析

本题让我们建立一个航空订票系统，需要实现的基本功能是

（1）查询航线：根据旅客提出的终点站名输出下列信息：航班号、飞机号、星期几飞行，最近一天航班的日期和余票额；

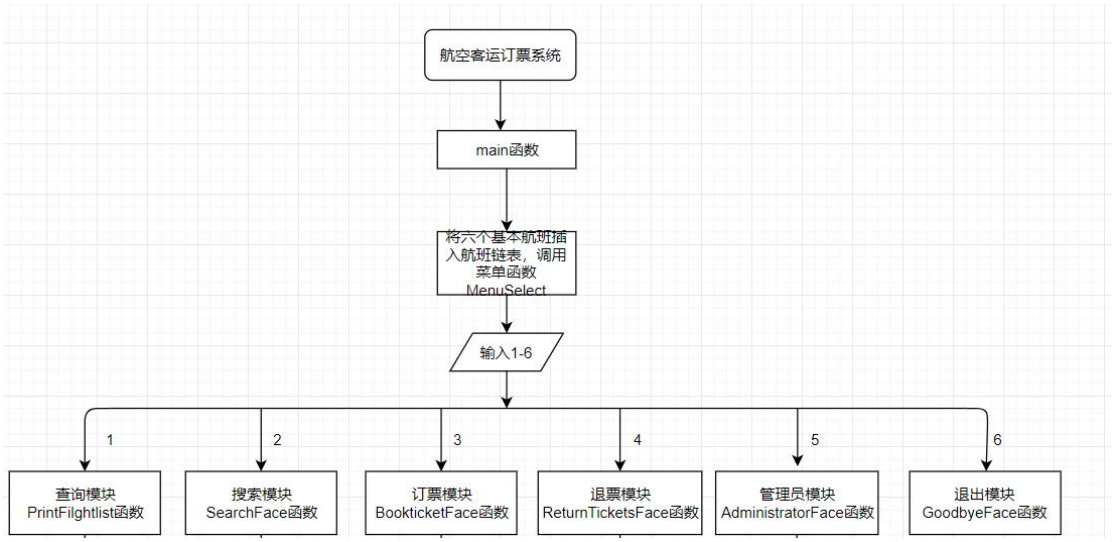
（2）承办订票业务：根据客户提出的要求（日期、航班号、订票数额）查询该航班票额情况，若尚有余额，则为客户办理订票手续，输出座位号；若已满员或余票额少于订票额，则需要重新询问客户要求。若需要，可预约登记排队等候。

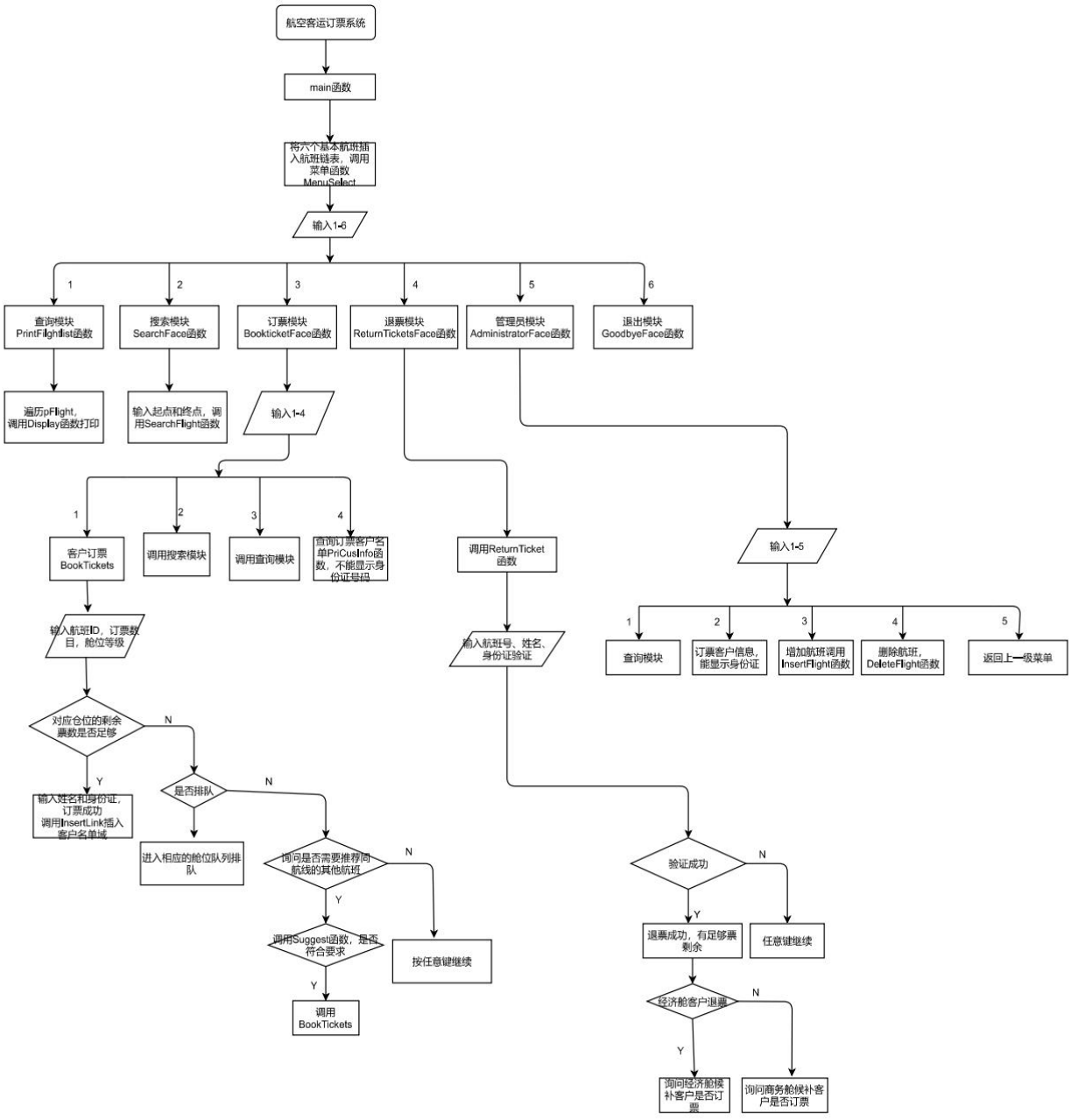
（3）承办退票业务：根据客户提供的情况（日期、航班、退票数额），为客户办理退票手续，然后查询该航班是否有人预约登记，首先询问排在第一的客户，若所退票额能满足他的要求，则为他办理订票手续，否则依次

询问其他排队预约的客户。

（4）当客户订票要求不能满足时，系统可向客户提供到达同一目的地的其它航线情况。

首先，处理这样一个订票系统问题，我们首先应该对客户和航班分别建立一类数据类型，然后我们可以使用链表这样的数据结构来进行存储客户和航班的相关数据。





2.2 主程序设计

```
/*********************
* 版权所有 (C)2022, Yuxing liu。
*
* 文件名称： main.cpp
* 文件标识：无
* 内容摘要：该代码用于完成界面及选择功能
* 其它说明：无
* 当前版本： V1.0
* 作 者：刘宇星
* 完成日期： 2022-05-10
**********************/

#include <iostream>
#include <windows.h> ///改变字体颜色用
#include "head.h"
using namespace std;
int main()
{
    Airline airline;
    airline.Init_passenger();//初始用户信息
    //airline.Load_flight();
    airline.Init_flight();//初始化航班
    //airline.Load_passenger();
    airline.root_init();//管理员初始化
    airline.root_load();
    while (1)
    {
        SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
FOREGROUND_INTENSITY | ///字体颜色
            FOREGROUND_BLUE | FOREGROUND_GREEN);
        cout << "\n";
        cout << "                          ╪╪ 飞 机 订 票 系 统 ╪╪ "
<< endl << endl;
        cout << "                  ╔----------------------------------
----------------------------------------------╗ " << endl;
        cout   <<   "                                                  ┇
┇ " << endl;
        cout << "                                  ┇     ═ 主菜单 ═     ┇
" << endl;
        cout   <<   "                                                  ┇
┇ " << endl;
```

```
        cout << "                                ┆    1.录入航班信息    ┆
" << endl;
        cout   <<   "                                                    ┆
┆ " << endl;
        cout << "                                ┆    2.录入乘客信息    ┆
" << endl;
        cout   <<   "                                                    ┆
┆ " << endl;
        cout << "                                ┆      3.查询航班      ┆
" << endl;
        cout   <<   "                                                    ┆
┆ " << endl;
        cout << "                                ┆    4.订票 5.退票     ┆
" << endl;
        cout   <<   "                                                    ┆
┆ " << endl;
        cout << "                                ┆      6.查询订单      ┆
" << endl;
        cout   <<   "                                                    ┆
┆ " << endl;
        cout << "                                ┆    7.修改航班信息    ┆
" << endl;
        cout   <<   "                                                    ┆
┆ " << endl;
        cout << "                                ┆    8.管理员模式      ┆
" << endl;
        cout   <<   "                                                    ┆
┆ " << endl;
        cout << "                                ┆     0.退出系统       ┆
" << endl;
        cout   <<   "                                                    ┆
┆ " << endl;
        cout << "                          ┗----------------------------------
----------------------------------------------┛ " << endl;
        cout << "                    请输入想要进行的功能：";
        int i;
        cin >> i;
        if (i == 1)
        {
            airline.Load_flight();//录入航班信息
            printf("                    恭喜您，航班信息录入成功\n");
        }
```

```
        else if(i == 2)
        {
            airline.Load_passenger();//录入乘客信息
            printf("                    恭喜您，乘客信息录入成功\n");
        }
        else if (i == 3)

            airline.Check_flight();//查找航班
        else if (i == 4)
        {
            airline.Book();//定票
            airline.Save_flight();//保存航班信息
            airline.Save_passenger();//保存用户信息
            printf("\n                    航班信息已自动保存\n");
            printf("\n                    用户信息已自动保存\n");
        }
        else if (i == 5)
        {
            airline.Qbook();//退票
            airline.Save_flight();//保存航班信息
            airline.Save_passenger();//保存用户信息
            printf("\n                    航班信息已自动保存\n");
            printf("\n                    用户信息已自动保存\n");
        }
        else if (i == 6)
            airline.Check_book();//查询订单
        else if (i == 7)
        {
            airline.Revise_flight();//修改航班信息
            airline.Save_flight();//保存航班信息
            printf("\n                    航班信息已自动保存\n");
        }
        else if(i == 8)
        {
            airline.root_mode();
         }
        else if (i == 0)
            break;
        else
        {
            cout << "                    无效数字，请重新输入！";
            system("pause");
```

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

2.3 设计思路

登录界面分为如下三种情况：

1. 账号密码的验证
2. 用户界面的登录
3. 管理员界面的登录 //账户密码的验证 简单将用户名和密码设置为了固定的字符串
2. 用户界面
1. 航线查询：航线可通过地名，或者飞机号进行查询
2. 航线查看：查看每条航线所涉及的信息，如终点站名、航班号、飞机号、飞机周日（星期几）、乘员定额、余票量、订定票的客户名单（包括姓名、订票量、舱位等级 1，2 或 3）等；
3. 订/退票 3. 管理员界面

1.航线查询：航线可通过地名，或者飞机号进行查询

2.航线查看：查看每条航线所涉及的信息，如终点站名、航班号、飞机号、飞机周日（星期几）、乘员定额、余票量、订定票的客户名单（包括姓名、订票量、舱位等级 1，2 或 3）等；

3.订/退票

4.对航线进行增减

5. 对航线进行编辑 4.航线信息界面 三、 主要模块的算法说明：即实现该模块的思路；

1， 登录界面

1. 在弹出对话框中填写用户名和密码，按下登录按钮，如果用户名和密码均正确则进入主窗口，如果有错则弹出警告对话框。
2. 新建 Qt Widgets Application，项目名称为 LoginDialog，类名和基类保持 MainWindow 和 QMainWindow 不变。
3. 完成项目创建后，向项目中添加新的 Qt 设计师界面类，模板选择 Dialog without Buttons，类名更改为 LoginDialog。完成后向界面上添加两个标签 Label、两个行编辑器 Line Edit 和两个按钮 Push Button，并改名为 usrEdit 与 passEdit。
4. 右击登录按钮，在弹出的菜单中选择“转到槽…”，然后选择 clicked()信号并确定。转到相应的槽以后，添加函数调用。
5. 这时运行程序，按下退出按钮会退出程序，按下登录按钮会关闭登录对话框，并显示主窗口。

2， 用户界面设计

1. 左侧的地域树，因为都是固定信息，直接用一个 QTreeWidget 在 UI 设计器把内容填好了。
2. 中间表格使用 QTableView 来显示记录。
3. 数据使用程序目录下的.db 文件，使用 sqlite，进行左上角文件打开与保存。
4. 关键字的过滤需要把所有列的内容都匹配识别。
5. 窗口高度刚好把地域全部覆盖，省得拖滚动条；
6. 三种过滤方式：点击 tree 里的节点按地域过滤、在筛选的 combobox 里按单位类型过滤、在搜索里按关键字过滤；

7. 双击一行进行修改，但不能对航班信息进行修改，只能对自己信息进行录入更改；
3， 管理员界面设计
1. 左侧的地域树，因为都是固定信息，直接用一个 QTreeWidget 在 UI 设计器把内容填好了。
2. 中间表格使用 QTableView 来显示记录。
3. 数据使用程序目录下的.db 文件，使用 sqlite，进行左上角文件打开与保存。
4. 关键字的过滤需要把所有列的内容都匹配识别。
5. 窗口高度刚好把地域全部覆盖，省得拖滚动条；
6. 三种过滤方式：点击 tree 里的节点按地域过滤、在筛选的 combobox 里按单位类型过滤、在搜索里按关键字过滤；
7. 点按钮增加、删除记录；
8. 双击一行进行修改，可以修改航班信息，包括终点站名、航班号、飞机号、飞机周日（星期几)、乘员定额、余票量、订定票的客户名单（包括姓名、订票量、舱位等级 1，2 或 3）等； 四、
运行结果：包括典型的界面、输入和输出数据等；
1. 关于 ui 界面的设计： 我们需要提示标签来指明输入的数据，也需要输入框来接收用户的输入，QLabel 与 QLineEdit 可以满足我们的要求。QLabel，QLineEdit 有方法 setText(“内容”)来改变其显示的文字，有方法 setGeometry (起始横坐标，起始纵坐标，宽， 高)来设置左上角起始位置与大小。
当然这些都可以在 Qt Designer 中直接设置。
2. 关于实现多个窗口的切换显示。
程序要实现的功能是：运行开始出现一个对话框，按下登录主界面按钮后该对话框消失并进入主窗口，如果直接关闭这个对话框，便不能进入主窗口，整个程序也将退出。进入主窗口后，按下显示对话框按钮，会弹出一个对话框，无论如何关闭这个对话框，都会回到主窗口。 程序里先建立一个工程，设计主界面，然后再建立一个对话框类，将其加入工程中，并在程序中调用自己新建的对话框类来实现多窗口。
3. 关于 icon 的设计
Qt 中可以使用资源文件将各种类型的文件添加到最终生成的可执行文件中，这样就可以避免使用外部文件而出现的一些问题。而且，在编译时 Qt 还会将资源文件进行压缩，我们可能发现生成的可执行文件比我们添加到其中的资源文件还要小。
添加完文件后会自动打开该资源文件，需要先添加前缀，点击“添加”按钮，然后选择“添加前缀”，默认的前缀是“/new/prefix1”，这个可以随意修改（不要出现中文字符），我们这里因为要添加图片，所以修改为“/myimages”。然后再按下添加按钮来添加文件，这里最好将所有要用到的图片放到项目目录中。比如这里在项目目录中新建了一个 images 文件夹，然后将需要的图标文件粘贴进去。添加完文件后，如下图所示。
4. 关于 sqlite 的建立与说明
在这个头文件中我们添加了一个建立连接的函数，使用这个头文件的目的就是要简化主函数中的内容。这里先创建了一个 SQLite 数据库的默认连接，设置数据库名称时使用了“:memory:”，表明这个是建立在内存中的数据库，也就是说该数据库只在程序运行期间有效，等程序运行结束时就会将其销毁。当然，大家也可以将其改为一个具体的数据库名称，比如“my.db”，这样就会在项目目录中创建该数据库文件了。
2.4 数据及数据类(型)定义

```
typedef struct flightnode
{
    char flight_num[10];//航班号
```

```
    char plane_num[10];//飞机号
    char time[20];//起飞时间
    char date[20];//日期
    char start_place[20];//出发地
    char end_place[20];//目的地
    int left;//余票
    int now_seat_num = 0;//当前的座次号
    float price;//价格
    flightnode *next;
}flightnode;
typedef struct administrator
{
    char name[20];
    char password[20];
    administrator *next;
    administrator *pre;
}administrator;
typedef struct passengernode
{
    char name[30];//名字
    char ID_num[30];//证件号
    char flight_num[10];//航班号
    char time[20];//起飞时间
    char start_place[20];//出发地
    char end_place[20];//目的地
    float price;//价格
    int full = 0;//是否处于候补状态
    int seat_num;//座次号
    passengernode *next;
}passengernode;
class Airline
{
    public:
        flightnode *head_of_flight;
        flightnode *tail_of_flight;
        passengernode *head_of_passenger;
        passengernode *tail_of_passenger;
        administrator *head_of_root;
        administrator *tail_of_root;
        char root_name[20] = "Yuxing_Liu";
        char root_password[20] = "20020316";
        void Init_flight();//初始化航班
        void Load_flight();//载入航班
        void Add_flight();//添加航班
        void Check_flight();//查找航班
        void Check_flightnum();//航班号查找
        void Check_seplace();//起始地查找
        void Check_all();//浏览全部航班
        void Check_ed();
```

```
        void Revise_flight();//修改航班信息
        void Delete_flight();//删除航班
        void Revise_time();//修改起飞抵达时间
        void Revise_price();//修改价格
        void Save_flight();//保存航班信息
        ///用户信息
        void passenger_display();//用户信息展示
        void Init_passenger();//初始化用户
        void Load_passenger();//载入用户信息
        void Book();//定票
        void Qbook();//退票
        void Check_book();//查询订单
        void Save_passenger();//保存用户信息
        //管理员模式
        int check();//能否获得超级管理员权限
        void root_display();//管理员展示
        void root_save();//管理员信息保存
        void root_load();//管理员信息载入
        void root_init();//管理员初始化
        void root_reset();//管理员清空
        void root_add();//添加管理员
        void root_del();//删除管理员
        int root_right();//管理员权限
        void root_mode();//管理员
};
```

2.5.算法设计及分析

1.初始化航班

```
void Airline::Init_flight()//初始化航班
{
    flightnode* h = (flightnode *)malloc(sizeof(flightnode));
    head_of_flight = tail_of_flight = h;
    h -> next = NULL;
}
```

2. 载入航班

```
/*******************
* 功能描述：载入航班信息
* 输入参数： 载入 FlightList.dat 信息到链表
* 输出参数： 无
* 返回值： void
* 其它说明：无
*******************/
void Airline::Load_flight()//载入航班
{
    flightnode *s;
    s = (flightnode *)malloc(sizeof(flightnode));
    ifstream infile("FlightList.dat", ios::in);//将文件载入
    if (!infile)
```

```
    {
        cerr << "                    信息错误。";//如果文件名错误的话，填写信息错误
        return;
    }
    while (1)//读入相关的信息
    {
        infile >> s -> flight_num >> s -> plane_num >> s -> date >> s -> time >> s ->
start_place >> s -> end_place >> s -> price >> s -> left >> s -> now_seat_num;
        if (!infile.eof())
        {
            tail_of_flight -> next = s;//不断更新尾结点和中间节点
            tail_of_flight = s;//不断更新尾结点和中间节点
            s = (flightnode *)malloc(sizeof(flightnode));//申请一个新的指针空间
        }
        else
            break;
    }
    tail_of_flight -> next = NULL;//最终让尾指针指向 NULL
    free(s);//将 s 指针的空间释放
}
```

3. 添加航班

```
/*******************
* 功能描述：添加航班
* 输入参数： 航班的相关信息
* 输出参数： 添加成功或者选择继续添加
* 返回值： void
* 其它说明：无
*******************/
void Airline::Add_flight()//添加航班
{
    char mark = 'y';
    while (mark == 'y')
    {
        flightnode* s = (flightnode *)malloc(sizeof(flightnode));//新的航班生命节点空间
        cout << "                    请输入航班号：";
        cin >> s -> flight_num;
        cout << "                    请输入飞机号：";
        cin >> s -> plane_num;
        cout << "                    请输入日期：";
        cin >> s -> date;
        cout << "                    请输入起飞时间：";
        cin >> s -> time;
        cout << "                    请输入出发地:";
        cin >> s -> start_place;
        cout << "                    请输入目的地：";
        cin >> s -> end_place;
        cout << "                    请输入价格：";
        cin >> s -> price;
```

```
        cout << "                    请输入余票：";
        cin >> s -> left;
        s -> now_seat_num = 0;//当前的座次号
        tail_of_flight -> next = s;//更新中间节点
        tail_of_flight = s;//更新尾结点
        cout << "                    添加成功！" << endl;
        cout << "                    是否继续添加？（是“y”/否“n”）：";
        cin >> mark;//判断是否需要继续添加
        cout << endl;
    }
    tail_of_flight -> next = NULL;//最后让尾节点指向 NULL，方便判断终止条件
}
```

4. 查找航班

```
/*******************
* 功能描述：查找航班
* 输入参数： 选择查找方式
* 输出参数： 进入相关函数进行查找功能
* 返回值： void
* 其它说明：无
*******************/
void Airline::Check_flight()//查找航班
{
    system("cls");
    int i;
    cout << "                    ------------------------------------------------------------------
------------------" << endl;
    cout << "                              ┆     ≡ 查找方式 ≡     ┆ " << endl;
    cout << "                  ┆                                    ┆ " << endl;
    cout << "                              ┆     1.航班号查找      ┆ " << endl;
    cout << "                  ┆                                    ┆ " << endl;
    cout << "                              ┆     2.起始地查找      ┆ " << endl;
    cout << "                  ┆                                    ┆ " << endl;
    cout << "                              ┆      3.终点查找       ┆ " << endl;
    cout << "                  ┆                                    ┆ " << endl;
    cout << "                              ┆    4.浏览全部航班     ┆ " << endl;
    cout << "                  ┆                                    ┆ " << endl;
    cout << "                    ------------------------------------------------------------------
------------------" << endl;
    cout << "                    请输入查找方式：";
    cin >> i;
    if (i == 1)
        Check_flightnum();//通过航班号查找
    else if (i == 2)
        Check_seplace();//通过起始地查找
    else if(i == 3)
        Check_ed();//终点查找
    else if (i == 4)
        Check_all();//浏览全部航班
```

```
}
```

5. 航班号查找

```
/*******************
* 功能描述：航班号查找
* 输入参数： 航班号
* 输出参数： 航班的相关信息
* 返回值： void
* 其它说明：无
*******************/
void Airline::Check_flightnum()//航班号查找
{

    char flightnum[10];
    flightnode *p = head_of_flight -> next;//定义 p 为头节点的下一个节点，这样的话就是第一个数据
    cout << setw(32) << "请输入航班号：";
    cin >> flightnum;//读取航班号
    cout << setw(25) << "航班号" << setw(12) << "飞机号" << setw(12) << "日期" << setw(12) << "起飞时间" << setw(12) << "出发地" << setw(12) << "目的地" << setw(12) << "价格" << setw(12) << "余票" << endl;
    while (p != NULL)//如果 p 没有到达终点
    {
        if (strcmp(p->flight_num, flightnum) == 0)//看能否匹配航班号
        {
            cout << setw(25) << p -> flight_num << setw(12) << p -> plane_num << setw(12) << p -> date << setw(12) << p -> time << setw(12) << p -> start_place << setw(12) << p -> end_place << setw(12) << p->price << setw(12) << p->left << endl;
            cout << "                    ";//缩进处理
            system("pause");
            return;
        }
        else
            p = p -> next;//否则就继续往下找
    }
    cout << setw(32) << "未查到任何信息。";//处理缩进
    system("pause");
}
```

6. 起始地查找

```
    /*******************
    * 功能描述：起始地查找
    * 输入参数： 起始地
    * 输出参数： 航班的相关信息
    * 返回值： void
    * 其它说明：无
    *******************/
    void Airline::Check_seplace()//起始地查找
```

```
    {
        char start_place[20], end_place[20];//起始地的字符串
        cout << "                    请输入出发地：";
        cin >> start_place;//输入起点
        cout << "                    请输入目的地：";
        cin >> end_place;//输入终点
        flightnode *p = head_of_flight -> next;//p为头指针的下一个指针，即第一个存放数据的指针
        cout << setw(25) << "航班号" << setw(12) << "飞机号" << setw(12) << "日期" << setw(12) <<  "起飞日期" << setw(12) << "出发地" << setw(12) << "目的地" << setw(12) << "价格" << setw(12) << "余票" << endl;
        int flag = 0;
        while (p != NULL)//如果没有将所有的航班遍历完
        {
            if (strcmp(p -> start_place, start_place) == 0 && strcmp(p -> end_place, end_place) == 0)//看起点和终点能否分别匹配
            {
                flag = 1;//标记为1
                cout << setw(25) << p -> flight_num << setw(12) << p -> plane_num << setw(12) << p -> date << setw(12) << p -> time << setw(12) << p->start_place << setw(12) << p->end_place << setw(12) << p->price << setw(12) << p->left << endl;
            }
            p = p -> next;
        }
        if(!flag)
        cout << setw(32) << "未查到任何信息" << endl;//如果没有被标记的话，就输出
        cout << setw(32);//缩进
        system("pause");
    }
```

7. 根据目的地查询航班

```
/*******************
* 功能描述：根据目的地查询航班
* 输入参数： 无
* 输出参数： 输出航班的相关信息
* 返回值： void
* 其它说明：无
*******************/
void Airline::Check_ed()
{
    char end_place[20];//终点站的字符串名字
    cout << "                    请输入目的地：";
    cin >> end_place;//读入终点站
    cout << setw(25) << "航班号" << setw(12) << "飞机号" << setw(12) << "日期" << setw(12) << "起飞日期" << setw(12) << "出发地" << setw(12) << "目的地" << setw(12) << "价格" << setw(12) << "余票" << endl;
    flightnode *p = head_of_flight -> next;//航班头结点的下一个节点
    int flag = 0;//判断是否能查找到相关信息
```

```
    while (p != NULL)
    {
        if (strcmp(p -> end_place, end_place) == 0)//如果能够匹配的话
        {
            cout << setw(25) << p -> flight_num << setw(12) << p -> plane_num << setw(12) << p -> date << setw(12) << p -> time << setw(12) << p->start_place << setw(12) << p->end_place << setw(12) << p->price << setw(12) << p->left << endl;
            flag = 1;//认为能找到对应的航班
        }
        p = p -> next;//继续往下找
    }
    if(!flag)   cout << setw(32) << "未查到任何信息" << endl;//如果查找不到任何信息的话
    cout << setw(32);//处理缩进
    system("pause");
}
```

8.. 浏览全部航班

```
          /*******************
* 功能描述：浏览全部航班
* 输入参数： 无
* 输出参数： 航班的相关信息
* 返回值： void
* 其它说明：无
*******************/
void Airline::Check_all()//浏览全部航班
{

          flightnode *p = head_of_flight -> next;//p 节点的下一个节点
          cout << setw(25) << "航班号" << setw(12) << "飞机号" << setw(12) << "日期" << setw(12) << "起飞日期" << setw(12) << "出发地" << setw(12) << "目的地" << setw(12) << "价格" << setw(12) << "余票" << endl;
          while (p != NULL)//如果还没到尾结点
          {
           cout << setw(25) << p -> flight_num << setw(12) << p -> plane_num << setw(12) << p -> date << setw(12) << p -> time << setw(12) << p -> start_place << setw(12) << p->end_place << setw(12) << p->price << setw(12) << p->left << endl;
           p = p -> next;//继续往后找
          }
}
```

9. 修改航班信息

```
/*******************
* 功能描述：修改航班信息
* 输入参数：无
* 输出参数： 调用相关函数进行实现
* 返回值： void
* 其它说明：无
*******************/
```

```
void Airline::Revise_flight()//修改航班信息
{
    flightnode *f = head_of_flight -> next;//首节点的下一个节点，第一个存有数据的指针节点
    int i;
    cout << "                    ------------------------------------------------------------
------------------" << endl;
    cout << "                                ┆     ≡ 修改方式 ≡     ┆ " << endl;
    cout << "                ┆                                      ┆ " << endl;
    cout << "                                ┆     1.删除航班       ┆ " << endl;
    cout << "                ┆                                      ┆ " << endl;
    cout << "                                ┆     2.添加航班       ┆ " << endl;
    cout << "                ┆                                      ┆ " << endl;
    cout << "                                ┆     3.修改起飞时间   ┆ " << endl;
    cout << "                ┆                                      ┆ " << endl;
    cout << "                                ┆     4.修改航班价格   ┆ " << endl;
    cout << "                ┆                                      ┆ " << endl;
    cout << "                                ┆        5.返回        ┆ " << endl;
    cout << "                ┆                                      ┆ " << endl;
    cout << "                    ------------------------------------------------------------
------------------" << endl;
    cout << "                    请输入查找方式：";
    cin >> i;
    if (i == 1)
        Delete_flight();//删除航班
    else if (i == 2)
        Add_flight();//添加航班
    else if(i == 3)
        Revise_time();//修改起飞时间
    else if (i == 4)
        Revise_price();//修改航班价格
    else if (i == 5)//否则返回
        return;
}
```

10. 删除航班

```
    /********************
* 功能描述：删除航班
* 输入参数：航班号
* 输出参数： 删除成功
* 返回值： void
* 其它说明：无
*********************/
void Airline::Delete_flight()//删除航班
{
    flightnode *p = head_of_flight -> next, *q = head_of_flight;//q 用来代表当前节点的前一个
节点
    char flightnum[10];//航班号
    cout << "                    请输入您想要修改的航班号：";//输入想要修改的航班号
```

```
        cin >> flightnum;
        while (p != NULL)//从前到后找到这个
        {
          if (strcmp(p -> flight_num, flightnum) == 0)//匹配航班号
          {
              if(p -> next == NULL)//如果尾结点是最后一个节点
              {
                  tail_of_flight = q;//那么就将尾结点往前移一个
                  q -> next = NULL;//尾结点的下一个节点为NULL
              }
              else
              {
                  q -> next = p -> next;//更新当前节点的前一个节点
              }
              vector<passengernode *> v;
              passengernode *ptr1 = head_of_passenger -> next;
              passengernode *ptr2 = head_of_passenger;
              for(; ptr1 != NULL;)//删除这个航班对应的所有乘客信息
              {
                  if(strcmp(ptr1 -> flight_num, flightnum) == 0)//如果航班号能够匹配
                  {
                      if(ptr1 -> next == NULL)//航班节点的尾结点
                      {
                          tail_of_passenger = ptr2;//更新乘客节点的尾结点
                          ptr2 -> next = NULL;//节点的下一个节点为NULL
                      }
                      else
                      {
                          ptr2 -> next = ptr1 -> next;//更新前一个节点的下一个节点为当前节点的下
一个节点
                      }
                      v.push_back(ptr1);//放入动态数组当中
                      ptr1 = ptr2 -> next;//继续往下走
                  }
                  else
                  {
                      ptr1 = ptr1 -> next;//否则两个节点都往后走
                      ptr2 = ptr2 -> next;//继续遍历节点
                  }
              }
              for(auto ptr : v)//枚举所有的指针节点
              {
                  free(ptr);//释放内存空间
              }
              free(p);//释放当前节点的空间
              cout << "                    删除成功！";
              return;
          }
          q = q -> next;//同时往后走
```

```
        p = q -> next;//同时往后走
      }
}
```

11. 修改起飞抵达时间

```
/*******************
* 功能描述：修改起飞抵达时间
* 输入参数：航班号，新的航班号
* 输出参数： 修改成功
* 返回值： void
* 其它说明：无
*******************/
void Airline::Revise_time()//修改起飞抵达时间
{
    char flightnum[10];
    flightnode *p = head_of_flight -> next;//航班头节点的下一个节点
    cout << "                    请输入您想要修改的航班号：";
    cin >> flightnum;//航班号
    while (p != NULL)//如果 p 节点不是尾结点
    {
        if (strcmp(p -> flight_num, flightnum) == 0)//如果能够匹配航班号
        {
            cout << "                    请输入新的起飞时间：";
            cin >> p -> time;
            cout << "                    修改成功！";
            return;
        }
        p = p -> next;//向后遍历节点
    }
    cout << "                    没有您想要修改的航班号!";
    system("pause");
}
```

12. 修改价格

```
/*******************
* 功能描述：修改价格
* 输入参数：航班号，新的价格
* 输出参数： 修改成功
* 返回值： void
* 其它说明：无
*******************/
void Airline::Revise_price()//修改价格
{
    char flightnum[10];
    flightnode *p = head_of_flight -> next;//p 节点是头结点的下一个节点
    cout << "                    请输入您想要修改的航班号：";
    cin >> flightnum;//读入航班号
    while (p != NULL)//如果 p 指针不是尾指针的话
```

```
    {
        if (strcmp(p -> flight_num, flightnum) == 0)//flight_num 和 flightnum 能够对应起来
        {
            cout << "                    输入新的价格：";
            cin >> p -> price;
            cout << "                    修改成功！    ";
            return;
        }
    }
    cout << "                    没有您想要修改的航班号!";
    system("pause");
}
```

13. 保存航班信息

```
/********************
* 功能描述：保存航班信息
* 输入参数：无
* 输出参数：用户的信息到文件 FlightList.dat
* 返回值： void
* 其它说明：
********************/
void Airline::Save_flight()//保存航班信息
{
    flightnode *f = head_of_flight -> next;//f 为头结点的下一个节点
    ofstream outfile("FlightList.dat", ios::trunc);//倒入节点的相关信息
    if (!outfile)//如果打开失败的话
    {
        cerr << "                    存储失败！";//输出存储失败
        return;
    }
    while (f != NULL)//如果 f 节点不是尾结点的话
    {
        outfile << f -> flight_num << " " << f -> plane_num << " " << f -> date << " " << f ->
time << " " << f -> start_place << " " << f -> end_place << " " << f -> price << " " << f ->
left << " " << f -> now_seat_num << endl;
        f = f -> next;//更新 f 节点的下一个节点
    }
    outfile.close();//关闭输出文件
}
```

14. 初始化用户

```
/********************
* 功能描述：初始化用户
* 输入参数： 无
* 输出参数： 无
* 返回值： void
* 其它说明：无
********************/
```

```
void Airline::Init_passenger()//初始化用户
{
    passengernode* c = (passengernode *)malloc(sizeof(passengernode));//新的节点声明
    head_of_passenger = tail_of_passenger = c;//头结点等于尾结点等于 c
    c -> next = NULL;//更新节点的下一个节点为 NULL
}
```

15. 用户信息展示

```
/********************
* 功能描述：用户信息展示
* 输入参数：无
* 输出参数：无
* 其他说明：无
********************/
void Airline::passenger_display()//用户信息展示
{
    passengernode *p = head_of_passenger -> next;//p 节点为头结点的下一个节点
    cout << setw(25) << "预约状态？" << setw(12) << "乘客姓名" << setw(12) << "乘客 ID" << setw(12) << "乘坐航班号" << setw(12) << "座次" << setw(12) << "航班时间" << setw(12) << "始发点" << setw(12) << "终点" << setw(12) << "票价" << endl;
    for(p; p != NULL; p = p -> next)//遍历链表中的节点
    {
        cout << setw(25) << p -> full << setw(12) << p -> name << setw(12) << p -> ID_num << setw(12) << p -> flight_num << setw(12) << p -> seat_num << setw(12) << p -> time << setw(12) << p -> start_place << setw(12) << p -> end_place << setw(12) << p -> price << endl;

    }
    return;//输出后返回即可
}
```

16. 载入用户信息

```
/********************
* 功能描述：载入用户信息
* 输入参数： 载入 PassengerList.dat 信息到链表
* 输出参数： 无
* 返回值： void
* 其它说明：无
********************/
void Airline::Load_passenger()//载入用户信息
{
    passengernode *s;//定义这个乘客节点 s
    s = (passengernode *)malloc(sizeof(passengernode));//为新的节点申请命名空间
    ifstream infile("PassengerList.dat", ios::in);//载入这个信息
    if (!infile)//如果读取不到这个文件的话
    {
        cerr << "                    信息错误。";//认为信息错误
        return;
    }
```

```
    while (1)
    {
        infile >> s -> name >> s -> ID_num >> s -> flight_num >> s -> seat_num >> s -> time >>
s -> start_place >> s -> end_place >> s -> price;
        if (!infile.eof())//读入相关的数据
        {
            tail_of_passenger -> next = s;//乘客节点的下一个节点为 s
            tail_of_passenger = s;//更新尾结点
            s = (passengernode *)malloc(sizeof(passengernode));//声明一个新的节点
        }
        else
            break;
    }
    tail_of_passenger -> next = NULL;//节点的后一个节点为 NULL
    free(s);//释放这个节点
}
```

17. 航班信息

```
/********************
* 功能描述：航班信息
* 输入参数：passenger 的相关信息
            flight 的相关信息
* 输出参数： 购票成功 or 失败
* 返回值： void
* 其它说明：如果余票不足还可以预定，如果有多余的票即可瞬间购买
********************/
void Airline::Book()//订票
{
    char start_place[20], end_place[20], flightnum[10];//航班信息
    flightnode *p = head_of_flight -> next, *q = head_of_flight -> next;//两个节点
    passengernode *s;//客户节点
    char mark, check = '1';//界面是否重复展示
    s = (passengernode *)malloc(sizeof(passengernode));//新的乘客节点
    cout << "                    请输入您的姓名：";
    cin >> s -> name;
    cout << "                    请输入您的证件号：";
    cin >> s -> ID_num;
    cout << "                    请输入出发地:";
    cin >> start_place;
    cout << "                    请输入目的地:";
    cin >> end_place;
    cout << setw(25) << "航班号" << setw(12) << "起飞日期" << setw(12) << "出发地" << setw(12)
<< "目的地" << setw(12) << "价格" << setw(12) << "余票" << endl;
    while (p != NULL)//如果不是终点
    {
        if (strcmp(p -> start_place, start_place) == 0 && strcmp(p -> end_place, end_place) ==
0)
        {
```

```
            cout << setw(25) << p -> flight_num << setw(12) << p -> time << setw(12) << p -> start_place << setw(12) << p->end_place << setw(12) << p->price << setw(12) << p->left << endl;
            check = '0';//说明找到了
        }
        p = p -> next;
    }
    if (check == '1')//否则说明没有找到 qaq
    {
        cout << "                    没有相关的航班，";
        system("pause");
        cout << "                    请问是否需要查看其他能够到达同一目的地的航班(是 y / 否 n)";
        cin >> mark;//如果继续添加的话
        if(mark == 'y')
        {
            p = head_of_flight -> next;//头结点的下一个节点
            cout << setw(25) << "航班号" << setw(12) << "起飞日期" << setw(12) << "出发地" << setw(12) << "目的地" << setw(12) << "价格" << setw(12) << "余票" << endl;
            while(p != NULL)//指针不等于 NULL
            {
                if(strcmp(p -> end_place, end_place) == 0)//匹配一下终点站
                {
                    cout << setw(25) << p -> flight_num << setw(12) << p -> time << setw(12) << p -> start_place << setw(12) << p -> end_place << setw(12) << p -> price << setw(12) << p -> left << endl;
                }
                p = p -> next;//向后遍历节点
            }
            cout << "                    ";//输出缩进
            system("pause");
            return;
        }
        else
        return;//返回节点
    }
    cout << "                    请输入您想要购买的航班号：";
    cin >> flightnum;//航班号
    puts("");
    while (q != NULL)//q 不是 NULL 节点
    {
        if (strcmp(q -> flight_num, flightnum) == 0 && strcmp(q -> start_place, start_place) == 0 && strcmp(q->end_place, end_place) == 0)//匹配起始点是否相同
        {
            if (q -> left == 0)//如果余票为 0 了
            {
                cout << "                    余票不足，请选择是否预约（是 y / 否 n）";//输出余票不足
                cin >> mark;//读入指令
                if (mark == 'y')//如果指令为 y 的话
```

```
            {
                s -> price = q -> price;//价格更新
                s -> seat_num = 0;//座次号更新
                s -> full = 1;//表示在预约状态
                strcpy(s -> start_place, start_place);//复制起点
                strcpy(s -> end_place, end_place);//复制终点
                strcpy(s -> time, q -> time);//复制时间
                strcpy(s -> flight_num, flightnum);//复制航班号
                cout << "                    预约成功！" << endl;
                tail_of_passenger -> next = s;//乘客的下一个节点为 s
                tail_of_passenger = s;//乘客节点更新为 s
                tail_of_passenger -> next = NULL;//乘客节点的尾结点为
                cout << "                    ";
                system("pause");//系统停止一会
                return;
            }
            else
                return;
        }
        /*
            char name[30];//名字
            char ID_num[30];//证件号
            char flight_num[10];//航班号
            char time[20];//起飞时间
            char start_place[20];//出发地
            char end_place[20];//目的地
            float price;//价格
            int full = 0;//是否处于候补状态
            int seat_num;//座次号
        */
        strcpy(s -> flight_num, flightnum);//复制航班号
        strcpy(s -> start_place, start_place);//复制起始地点
        strcpy(s -> end_place, end_place);//复制终止地点
        strcpy(s -> time, q->time);//复制时间
        s -> full = 0; //处于订票状态
        s -> price = q -> price;//价格更新
        s -> seat_num = q -> now_seat_num + 1;//座次号更新
        tail_of_passenger -> next = s;//更新尾节点
        tail_of_passenger = s; //乘客的尾结点
        tail_of_passenger -> next = NULL;//最后一个节点的 next 为 NULL
        //购票成功
        cout << "                    购票成功！" << endl;
        q -> left --;//余票减一
        q -> now_seat_num ++;//当前的座次号+1
        cout << "                    ";
        system("pause");
        return;
    }
    else
```

```
            q = q -> next;
    }
    cout << "                    航班号填入错误\n";
    cout << "                    ";
    system("pause");
}
```

18. 退票

```
/*******************
* 功能描述：退票
* 输入参数：passenger 的相关信息
* 输出参数： 退票成功 or 失败
* 返回值： void
* 其它说明：
*******************/
void Airline::Qbook()//退票
{
    char name[30];
    char ID_num[30];
    flightnode *f = head_of_flight -> next;//退票系统
    passengernode *p = head_of_passenger -> next, *q = head_of_passenger, *t = head_of_passenger -> next;
    cout << "                    请输入您的姓名：";
    cin >> name;//读入姓名
    puts("");
    cout << "                    请输入您的证件号：";
    cin >> ID_num;//读入乘客的身份证号码
    while (p != NULL)
    {
        if (strcmp(p -> name, name) == 0 && strcmp(p -> ID_num, ID_num) == 0)//姓名和身份证号匹配
        {
            while (f != NULL)//如果没到终点的话
            {
                if (strcmp(p -> flight_num, f -> flight_num) == 0)//航班号匹配
                {
                    f -> left ++;//余量+1
                    f -> now_seat_num --;//当前座次号减一
                    while (t != NULL)
                    {
                        if (strcmp(t -> flight_num, p -> flight_num) == 0 && t -> full == 1)
                        {
                            strcpy(t -> start_place, p -> start_place);//复制起始点
                            strcpy(t -> end_place, p -> end_place);//复制终点
                            strcpy(t -> time, p -> time);//复制时间
                            t -> price = p -> price;
                            t -> full = 0;//是否处于预约状态
                            t -> seat_num = f -> now_seat_num + 1;//座次号更新
```

```
                            f -> left --;//余票量减一
                            break;
                        }
                        t = t -> next;
                    }
                    break;
                }
                f = f -> next;
            }
            if(p == tail_of_passenger)
            {
                tail_of_passenger = q;
                tail_of_passenger -> next = NULL;
            }
            else
            {
                q -> next = p -> next;
            }
            for(passengernode *ptr = q -> next; ptr != NULL; ptr = ptr -> next)
            {
                if(strcmp(ptr -> flight_num, p -> flight_num) == 0 && ptr -> full == 0)
                {
                    ptr -> seat_num --;//把对应该航班号的所有处于不是预约状态的座次号更新
                }
            }
            free(p);//释放该节点的内存
            cout << "                    退票成功！";
            return;
        }
        q = q -> next;
        p = q -> next;
    }
}
```

19. 查询订票

```
/*******************
* 功能描述：查询订票
* 输入参数：passenger 的相关信息
* 输出参数： 所查用户订票的信息
* 返回值： void
* 其它说明：
*******************/
void Airline::Check_book()//查询订票
{
    char name[30];
    char ID_num[30];
    passengernode *p = head_of_passenger -> next;//头结点的下一个节点
    passengernode *q = head_of_passenger -> next;
```

```
    cout << "                    请输入您的姓名：";
    cin >> name;
    cout << "                    请输入您的证件号：";
    cin >> ID_num;
    cout << setw(25) << "姓名" << setw(12) << "航班号" << setw(12) << "座次" << setw(12) << "起飞日期" << setw(12) << "出发地" << setw(12) << "目的地" << setw(12) << "价格" << setw(12) << endl;
    int rank = 0;//当前的为此
    while (p != NULL)
    {
        if (strcmp(p -> name, name) == 0 && strcmp(p -> ID_num, ID_num) == 0)
        {
            if(p -> full)
            {
                for(q; q != NULL; q = q -> next)
                {
                    if(q -> full && q -> flight_num == p -> flight_num)
                    {
                        rank ++;//如果处于预约状态那么 rank++
                    }
                }
                cout << "                    目前该乘客正处于预约状态，排在第" << rank + 1 << "位\n";
            }
            else cout << "                    目前乘客已成功订票，乘客信息如下所示\n";
            cout << setw(25) << p -> name << setw(12) << p -> flight_num << setw(12) << p -> seat_num << setw(12) << p->time << setw(12) << p->start_place << setw(12) << p->end_place << setw(12) << p->price << setw(12) << endl;
            cout << "                    ";
            system("pause");
            return;
        }
        else
            p = p -> next;//节点继续往下走啊
    }
    cout << setw(32) << "未查到任何信息。";
    system("pause");
}
```

20. 保存用户信息

```
/********************
* 功能描述：保存用户信息
* 输入参数：无
* 输出参数：用户的信息到文件 PassengerList.dat
* 返回值： void
* 其它说明：
********************/
void Airline::Save_passenger()//保存用户信息
{
```

```
    passengernode *p = head_of_passenger -> next;//头结点往后走
    ofstream outfile("PassengerList.dat", ios::trunc);//乘客信息
    if (!outfile)
    {
        cerr << "                    存储失败！";
        return;
    }
    while (p != NULL)//节点不是最后一个节点的话
    {
        outfile << p -> name << " " << p -> ID_num << " " << p -> flight_num << " " << p -> seat_num << " " << p -> time << " " << p->start_place << " " << p->end_place << " " << p->price << endl;
        p = p -> next;//读出这个节点的下一个节点
    }
    outfile.close();//关闭文件
}
```

21. 用户信息初始化

```
void Airline::root_init()
{
    administrator *s = (administrator *)malloc(sizeof(administrator));//初始化管理员
    head_of_root = tail_of_root = s;//定义一个新的节点后开始更新
    s -> next = NULL;
}
```

22. 超级管理员权限获取

```
int Airline::check()
{
    administrator *ptr = (administrator *)malloc(sizeof(administrator));//申请一个新的指针
    cout << endl << "                    请输入超级管理员（root）用户名：" ;
    cin >> ptr -> name;
    if(strcmp(ptr -> name, root_name) == 1) //看名字是否能够匹配
    {
        cout << endl << "                    用户名错误\n";
        return 0;
    }
    printf("\n");
    cout << endl << "                    请输入密码:";
    cin >> ptr -> password;//读入密码即可
    if(strcmp(ptr -> password, root_password) == 1)//密码能否匹配
    {
        cout << endl << "                    密码错误\n";
        return 0;
    }
    printf("\n");
    return 1;
}
```

23. 管理员信息清零

```
/********************
* 功能描述：管理员信息清零
```

```
* 输入参数： 载入 FlightList.dat 信息到链表
* 输出参数： 无
* 返回值： void
* 其它说明：无
*********************/
void Airline::root_reset()
{
    if(tail_of_root == head_of_root) return;//一次删除所有的管理员信息
    administrator* ptr = tail_of_root -> pre;
    administrator* s;
    for(ptr; ptr != head_of_root; ptr = ptr -> pre)
    {
        s = ptr -> next;
        free(s);//释放这个节点的内存
    }
    if(head_of_root -> next != NULL)
    {
        free(head_of_root -> next);//把初始节点的内存释放
        head_of_root -> next = NULL;
    }
}
```

24. 管理员添加

```
/*********************
* 功能描述：管理员添加
* 输入参数： 无
* 输出参数： 无
* 返回值： void
* 其它说明：无
*********************/
void Airline::root_add()
{
    while(1)
    {
        cout << "                    请输入您要添加的用户名:";//输入要添加的用户名
        administrator* s = (administrator *)malloc(sizeof(administrator));
        cin >> s -> name;//读入用户名的信息
        cout << "                    密码为：";
        cin >> s -> password;//读入用户名的密码
        puts("");
        tail_of_root -> next = s;//更新尾结点
        s -> pre = tail_of_root;//更新前置节点
        tail_of_root = s;//更新尾结点
        cout << "                    添加成功，是否继续添加(是/y；否/n):";
        char p;
        cin >> p;//读入操作
        if(p == 'n') break;
    }
    tail_of_root -> next = NULL;//更新尾结点的下一个节点为 NULL
```

```
}
```

25. 管理员删除

```
/*******************
* 功能描述：管理员删除
* 输入参数： 无
* 输出参数： 无
* 返回值： void
* 其它说明：无
*******************/
void Airline::root_del()//删除管理员
{
    administrator *p = (administrator *)malloc(sizeof(administrator)); //生命一个新的节点命名空间
    cout << "                    请输入管理员用户名:";
    cin >> p -> name;//读入节点的名字
    puts("");
    cout << "                    正在删除.....................\n"; //操作
    for(administrator *ptr = head_of_root -> next; ptr != NULL; ptr = ptr -> next)//遍历管理员链表中的每个管理员
    {
        if(strcmp(ptr -> name, p -> name) == 0)//如果能够匹配名字
        {
            if(ptr -> next == NULL)//如果下一个节点为NULL
            {
                tail_of_root = ptr -> pre;//就第一种更新方式
                tail_of_root -> next = NULL;//尾结点的下一个节点为NULL
                free(ptr);//释放这个节点的空间
                break;
            }
            else
            {
                ptr -> pre -> next = ptr -> next;//更新这个节点前置节点的后继节点
                ptr -> next -> pre = ptr -> pre;//更新后继节点的前驱节点
                free(ptr);//释放这个节点
                break;
            }
        }
    }
    cout << "                    删除成功\n";
}
```

26. 管理员信息载入

```
void Airline::root_load()//管理员信息载入
{
    administrator *s;//定义这个节点
    s = (administrator *)malloc(sizeof(administrator));//给这个节点分配命名空间
    ifstream infile("root.dat", ios::in);//载入管理员的相关信息
    if (!infile)//如果载入失败的话
    {
```

```
        cerr << "                    信息错误。";
        return;
    }
    while (1)
    {
        infile >> s -> name >> s -> password;//读入用户名和密码
        if (!infile.eof())//如果读入失败的话
        {
            tail_of_root -> next = s;//尾结点的下一个节点更新为 s
            s -> pre = tail_of_root;//s 的前驱节点更新为尾结点
            tail_of_root = s;//尾结点更新为 s
            s = (administrator *)malloc(sizeof(administrator));//给 s 分配内存空间
        }
        else
            break;//否则就载入结束了
    }
    tail_of_root -> next = NULL;//尾结点的下一个节点为 NULL
    free(s);            //释放 s 这个节点
}
```

27. 管理员信息保存

```
void Airline::root_save()//管理员信息保存
{
    administrator *f = head_of_root -> next;//定义 f 为头结点的下一个节点
    ofstream outfile("root.dat", ios::trunc);//保存的文件为 root.dat
    if (!outfile)//如果打不开对应的文件
    {
        cerr << "                    存储失败！";//输出存储失败
        return;
    }
    while (f != NULL)//如果 f 不是 NULL 的话
    {
        outfile << f -> name << " " << f -> password << endl; //输出对应的所有信息
        f = f -> next;//移动到下一个指针
    }
    outfile.close();//输出文件结束
}
```

28. 管理员权限

```
int Airline::root_right()//管理员权限
{
    int flag = 0;
    administrator *p = (administrator *)malloc(sizeof(administrator)); //声明节点 p，分配命名空间
    cout << "                    请输入用户名："; //读入用户名
    cin >> p -> name;//读入对应的用户名
    puts("");
    for(administrator *ptr = head_of_root -> next; ptr != NULL; ptr = ptr -> next)//遍历所有的管理员信息
    {
```

```
        if(strcmp(ptr -> name , p -> name) == 0)//如果能够匹配相关的信息的话
        {
            flag = 1;//让 flag=1 认为找到节点了
            cout << "                    请输入密码：";
            cin >> p -> password;//读入密码
            if(strcmp(ptr -> password, p -> password) == 0)
            {
                flag = 2;//如果用户名和密码都能够匹配的话
                cout << "                        管理员权限获取成功！\n";
                break;//获得管理员权限
            }
        }
    }
    if(flag == 0)//如果用户名错误的话
    {
        cout << endl;
        cout << "                    用户名错误\n";
        return 0;
    }
    else if(flag == 1)//如果密码错误的话
    {
        cout << "                    密码错误\n";
        return 0;
    }
    else return 1;//否则认为获得权限
}
```

29. 管理员模式

```
void Airline::root_mode()//管理员模式
{
    while(1)
    {
        cout << "                    ----------------------------------------------------------------
----------------------" << endl;
        cout << "                                ┆     ≡ 管理员 ≡     ┆ " << endl;
        cout << "                    ┆                                    ┆ " << endl;
        cout << "                                ┆      1.超级管理员模式       ┆ " << endl;
        cout << "                    ┆                                    ┆ " << endl;
        cout << "                                ┆      2.管理员模式           ┆ " << endl;
        cout << "                    ┆                                    ┆ " << endl;
        cout << "                                ┆      3.返回上一步           ┆ " << endl;
        cout << "                    ┆                                    ┆ " << endl;
        cout << "                    ----------------------------------------------------------------
----------------------" << endl;
        cout << "                    请输入想要进行的功能：";
        int op;
        cin >> op;
        if(op == 1)
        {
```

```
            if(check())//如果超级管理员权限获得
            {
                while(1)//进入超级管理员模式
                {
                    cout << "                          超级管理员权限获取成功！\n";
                    cout << "                    ----------------------------------------------
---------------------------------" << endl;
                    cout << "                              ┆     ≡ 超级管理员模式 ≡     ┆ " <<
endl;
                    cout << "                    ┆                                    ┆ "
<< endl;
                    cout << "                              ┆      1.管理员初始化        ┆ " <<
endl;
                    cout << "                    ┆                                    ┆ "
<< endl;
                    cout << "                              ┆      2.添加管理员          ┆ " <<
endl;
                    cout << "                    ┆                                    ┆ "
<< endl;
                    cout << "                              ┆      3.删除管理员          ┆ " <<
endl;
                    cout << "                    ┆                                    ┆ "
<< endl;
                    cout << "                              ┆      4.录入管理员信息      ┆ " <<
endl;
                    cout << "                    ┆                                    ┆ "
<< endl;
                    cout << "                              ┆      5.管理员信息          ┆ " <<
endl;
                    cout << "                    ┆                                    ┆ "
<< endl;
                    cout << "                              ┆      6.返回上一步          ┆ " <<
endl;
                    cout << "                    ┆                                    ┆ "
<< endl;
                    cout << "                    ----------------------------------------------
---------------------------------" << endl;
                    cout << "                    请输入想要进行的功能：";
                    int op;
                    cin >> op;
                    if(op == 1)
                    {
                        root_reset();//管理员初始化
                        root_save();    //保存管理员信息
                    }
                    else if(op == 2)
                    {
                        root_add(); //管理员添加
                        root_save();//保存管理员信息
```

```
                    }
                    else if(op == 3)
                    {
                        root_del();//管理员删除
                        root_save();//保存管理员信息
                    }
                    else if(op == 4)
                    {
                        root_load();//管理员加载信息
                        cout << "                    管理员信息录入成功\n";
                    }
                    else if(op == 5)
                    {
                        root_display();//展示所有的管理员信息
                    }
                    else if(op == 6) break;
                    else printf("                                输入有误\n");
                }
            }
        }
        else if(op == 2)
        {
            if(root_right())//如果能够获得管理员权限
            {
                while(1)//进入管理员模式
                {
                    cout << "                    ----------------------------------------------------
----------------------------------" << endl;
                    cout << "                                ┇     ≡ 管理员模式 ≡     ┇ " <<
endl;
                    cout << "                    ┇                                       ┇ "
<< endl;
                    cout << "                                ┇      1.查询顾客信息      ┇ " <<
endl;
                    cout << "                    ┇                                       ┇ "
<< endl;
                    cout << "                                ┇      2.返回上一步        ┇ " <<
endl;
                    cout << "                    ┇                                       ┇ "
<< endl;
                    cout << "                    ----------------------------------------------------
----------------------------------" << endl;
                    cout << "                    输入想要进行的功能：";
                    int op;
                    cin >> op;//读入 op 操作

                    if(op == 1)
                    {
                        passenger_display();//顾客信息展示
```

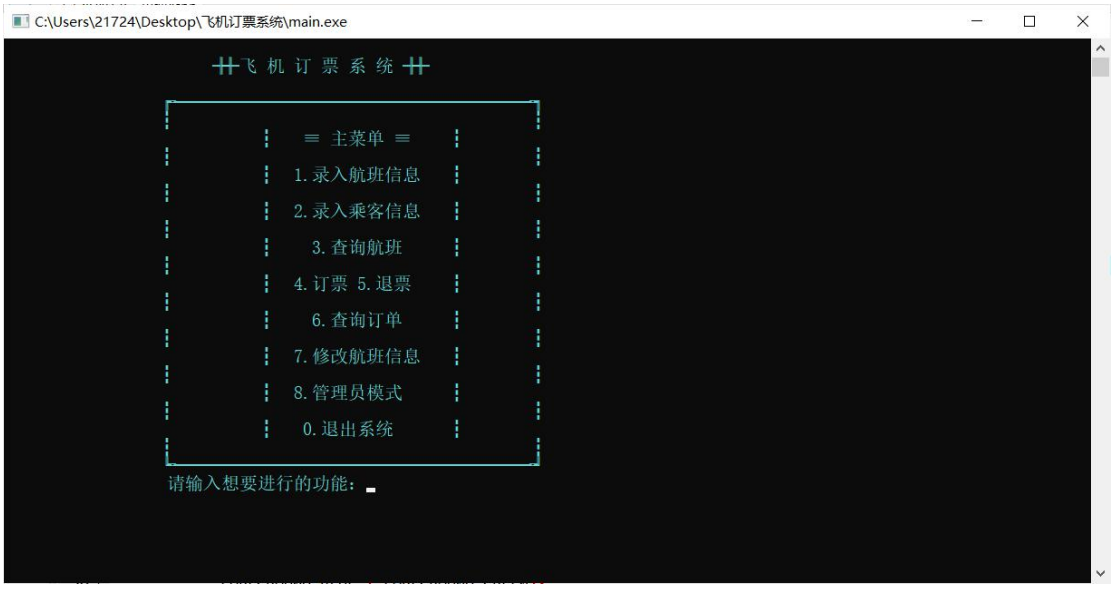
```
                    }
                    else if(op == 2)
                    {
                        break;//退出
                    }
                    else
                    {
                        cout << "                    输入错误\n";
                    }
                }
            }
        }
        else
        {
            break;
        }
    }
}
```

30. 管理员信息展示

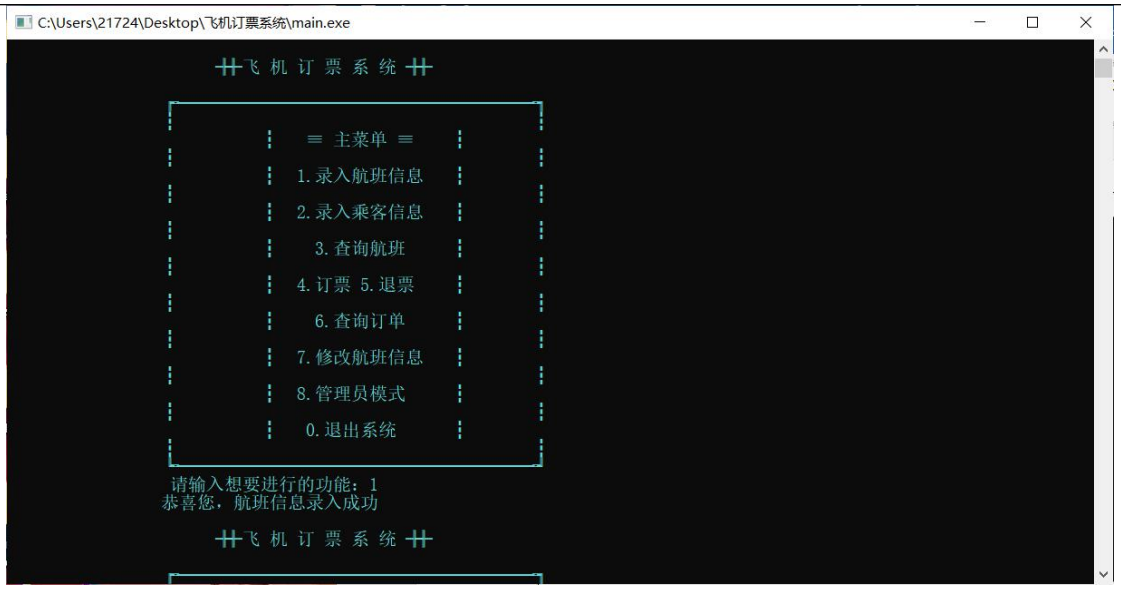
```
void Airline::root_display()
{
    administrator *ptr = head_of_root -> next;//管理员信息展示
    cout << setw(25) << "用户名" << setw(12) << "密码" << endl;
    for(ptr; ptr != NULL; ptr = ptr -> next)//节点从前到后遍历
    {
        cout << setw(25) << ptr -> name << setw(12) << ptr -> password << endl;
    }
}
```

3. 测试

主界面：



操作 1：录入航班信息

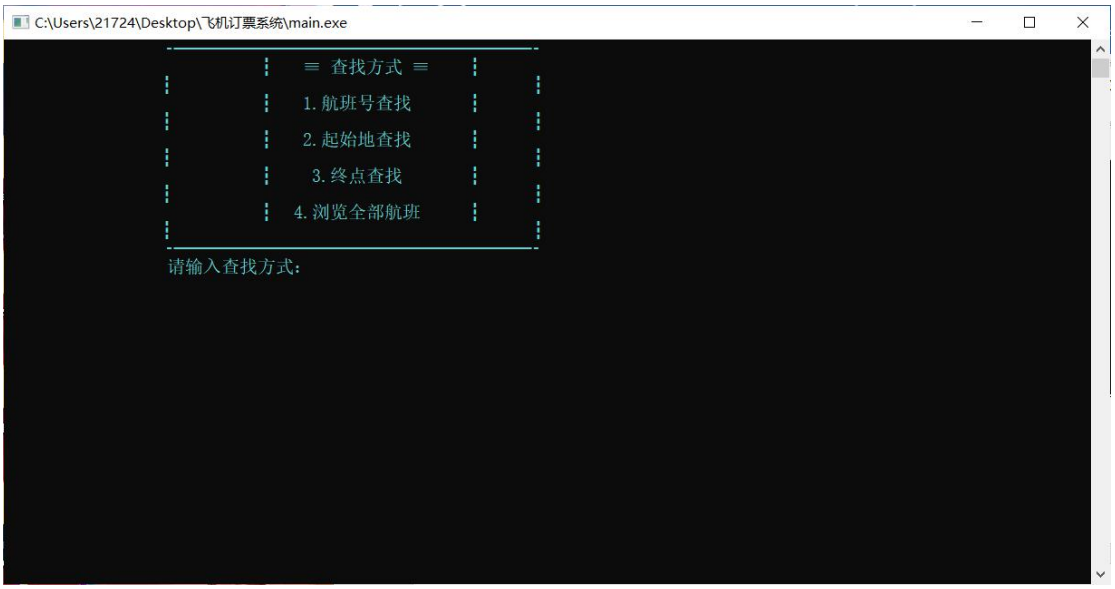


比对航班的信息和.dat 文件中存储的信息，发现结果一致，测试成功。

操作 2：

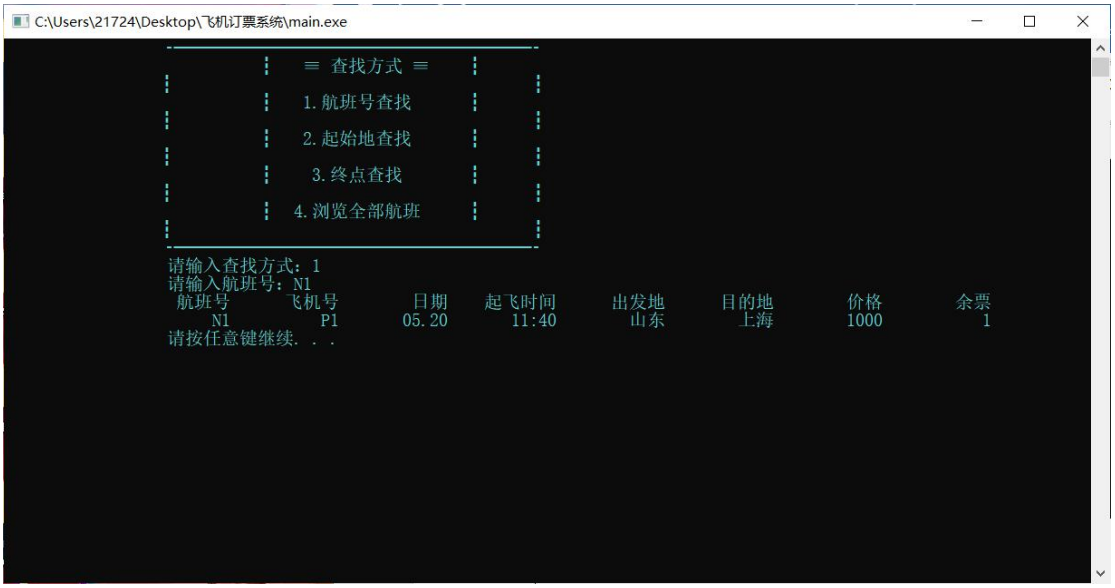
同测试一，成功

操作 3：查询航班

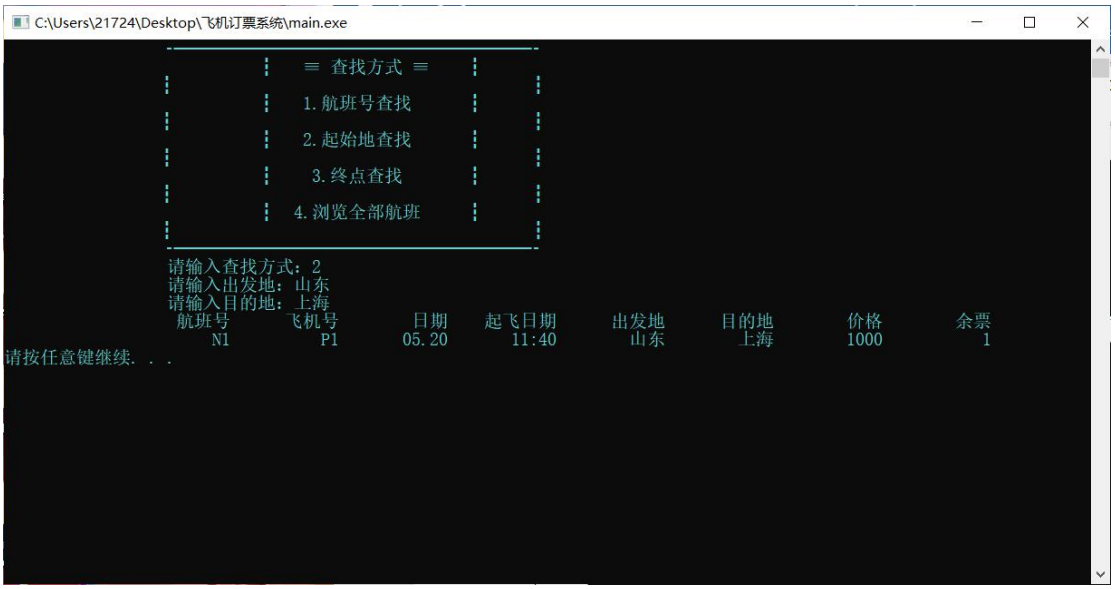


首先浏览现在加入的所有航班

发现只有 N1 这个航班，然后我们在按照航班号查找，发现通过 N1 就可以查找到该航班，测试成功。

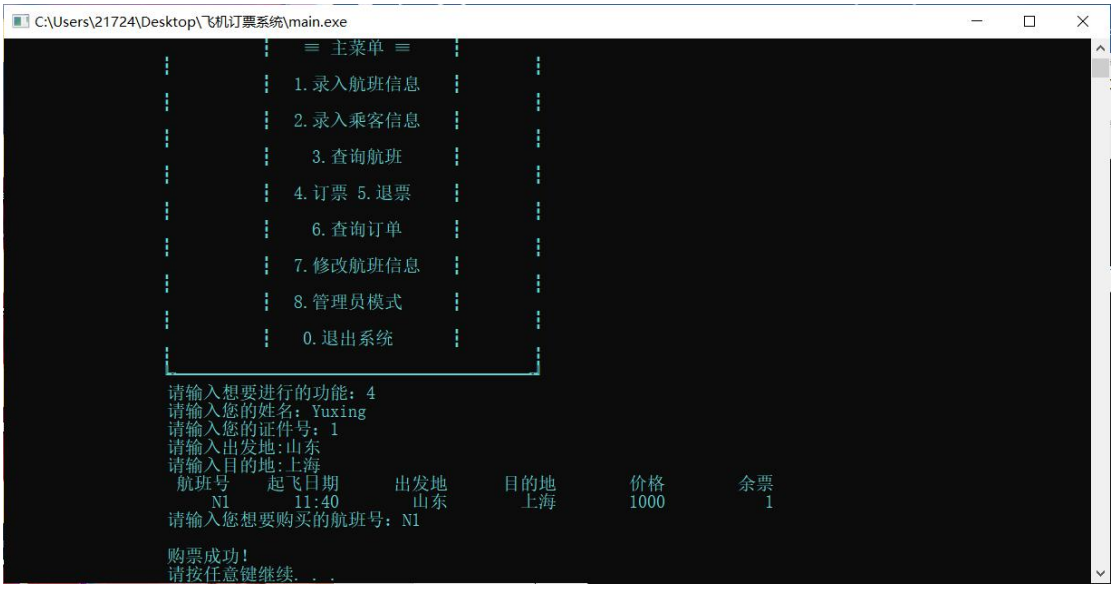


我们按照起始地查找，也能测试成功。输入出发地：山东 目的地：上海，即可查找到 N1 航班。

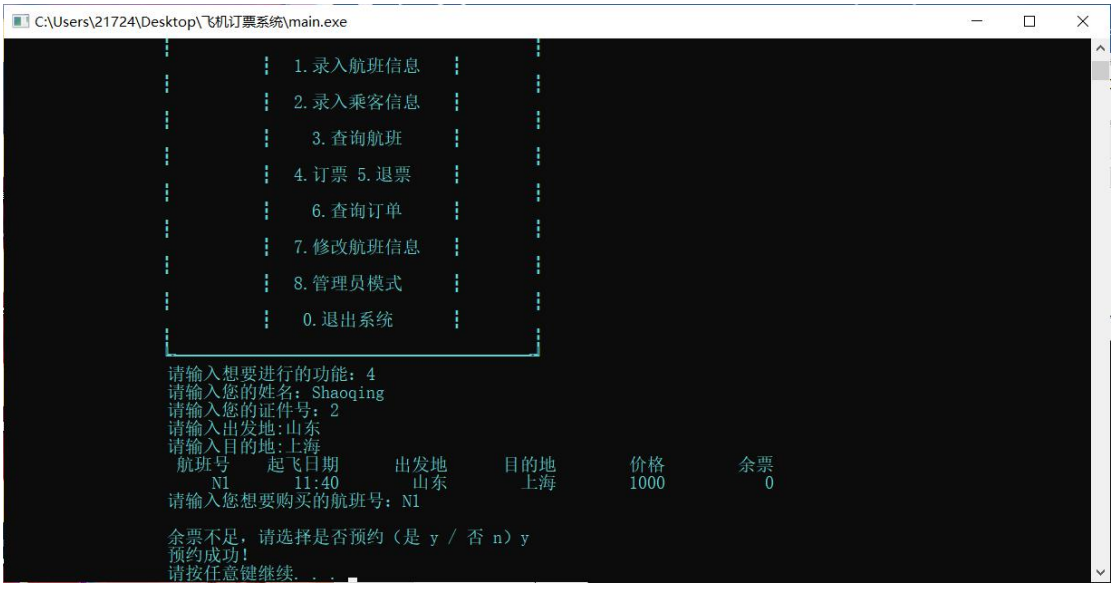


对于查找方式 3，我们输入上海，也可以查找到 N1 航班，测试成功

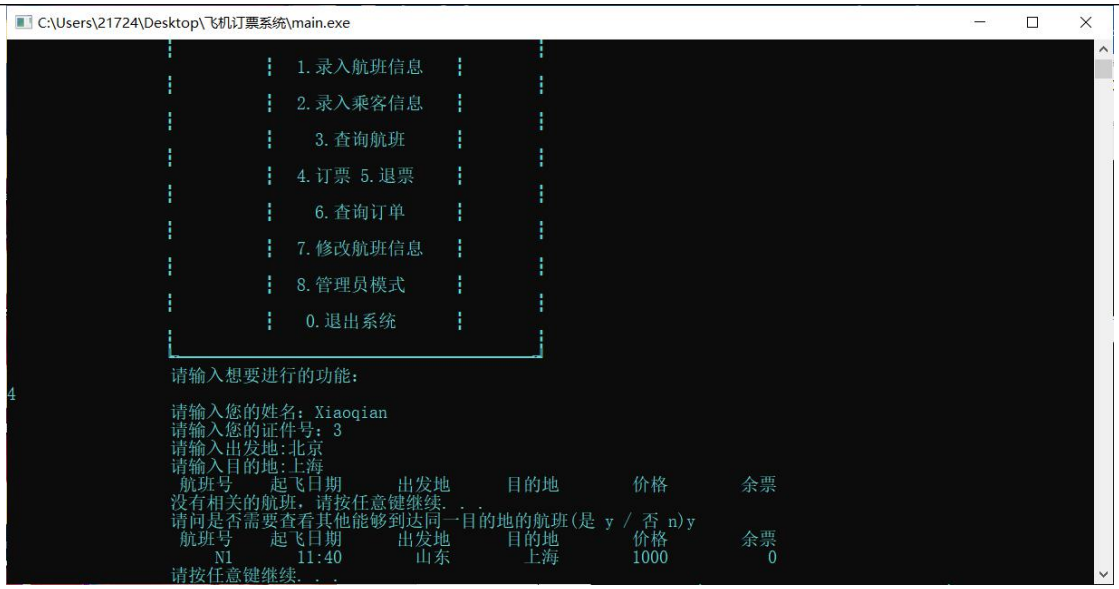
操作 4：订票板块



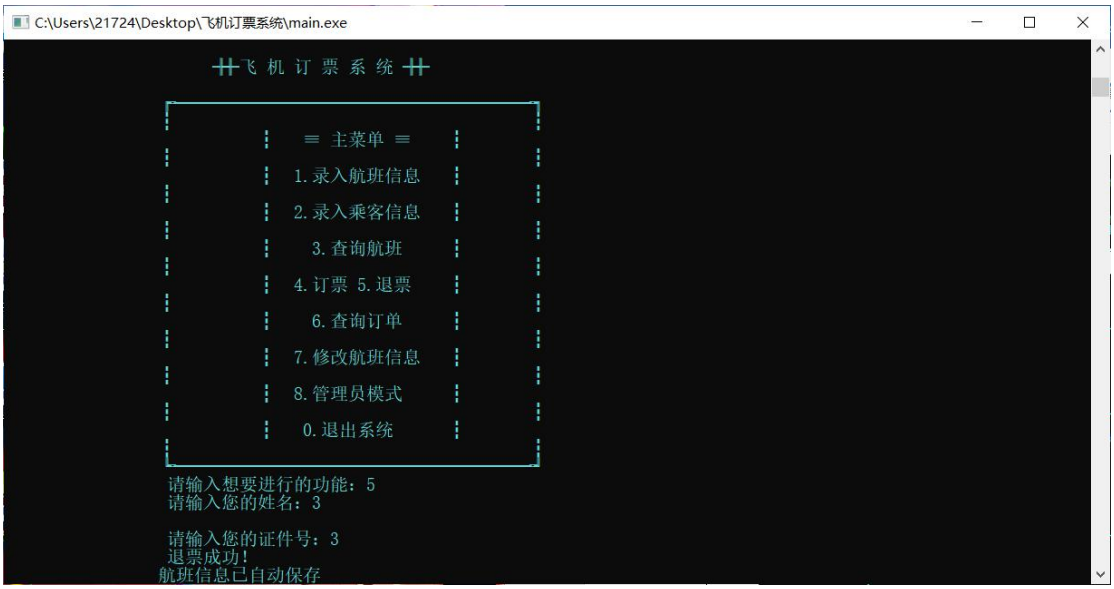
现在我们对 Yuxing 乘客进行订票，证件号为 1，出发地为山东，目的地为上海，首先找到了对应的合法的航班。然后我们再订票，显示订票成功。测试成功。



我们再为 xiaoqian 订票，他的证件号为 2，出发地相同，但是此时余票为 0 了，这时会询问是否预约，我们确定预约以后，显示预约成功。测试成功。

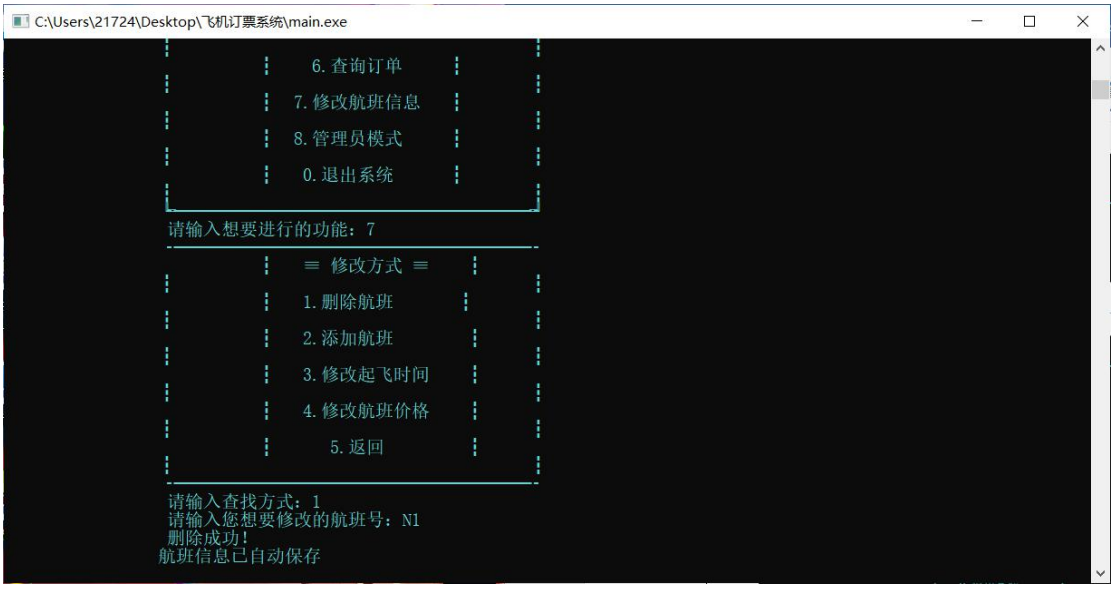


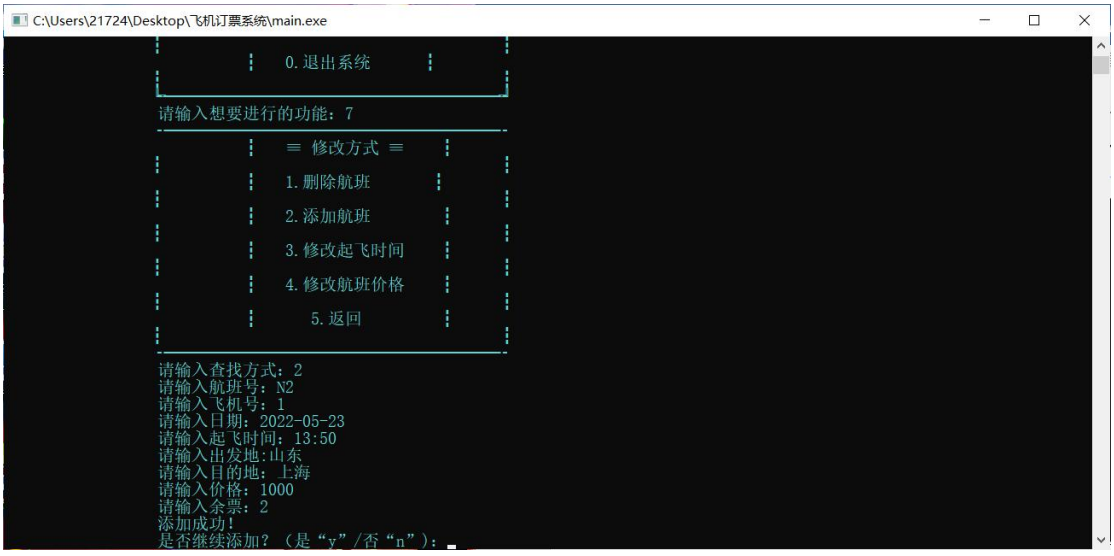
操作 5：//通过管理员模式查看乘客信息，发现除了之前导入的错误数据外其他数据呈现正确的状态，shaoqing 的预约状态为 1 显示为候补状态，而 yuxing 则为 0 表示已经订上票了



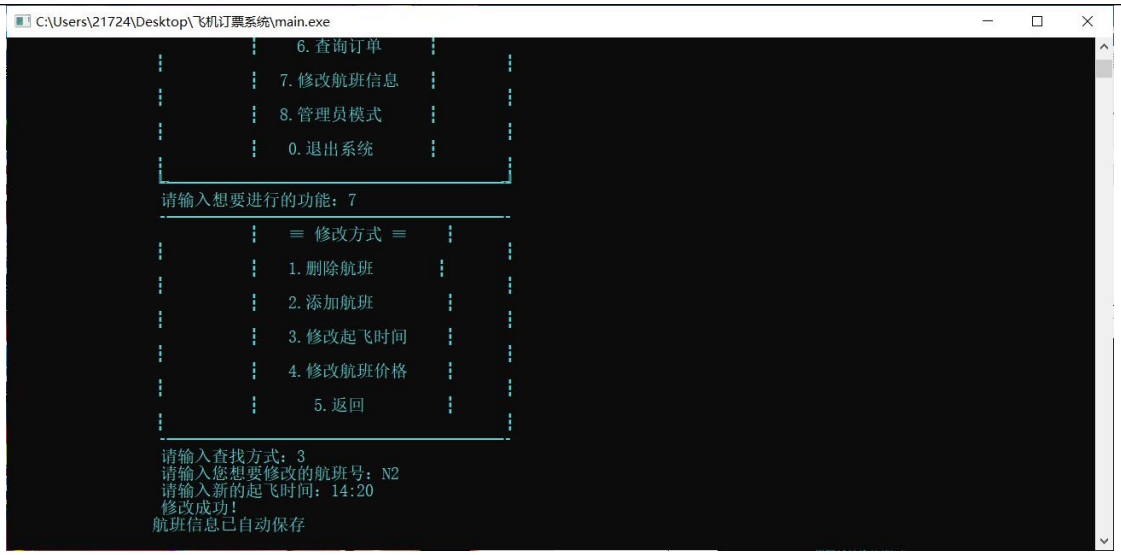
操作 6：我们对之前的乘客退票以后，发现之前候补的 shaoqing 订票成功了，查询订单和预约功能测试成功

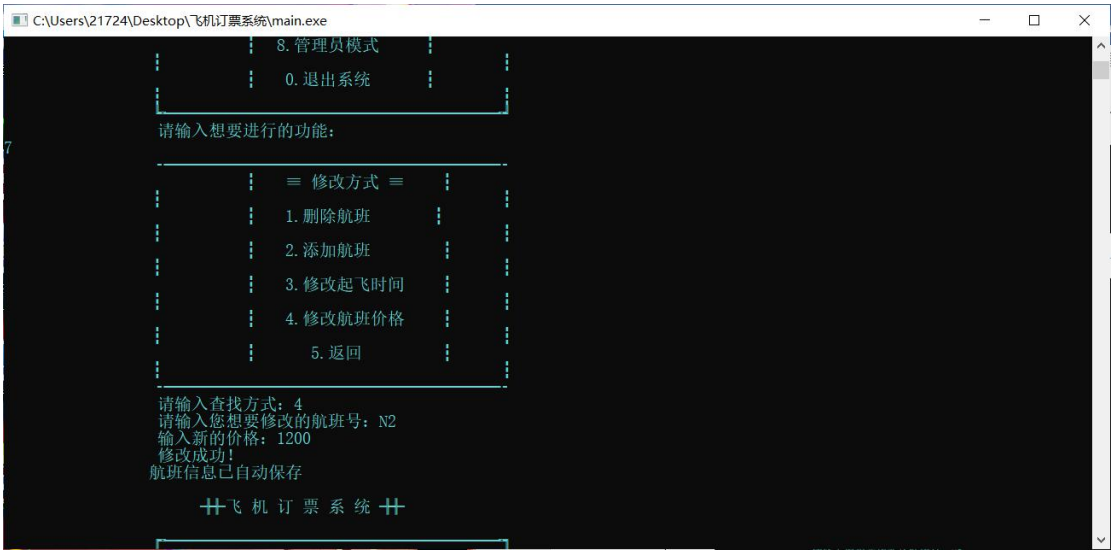
操作 7：修改航班信息功能



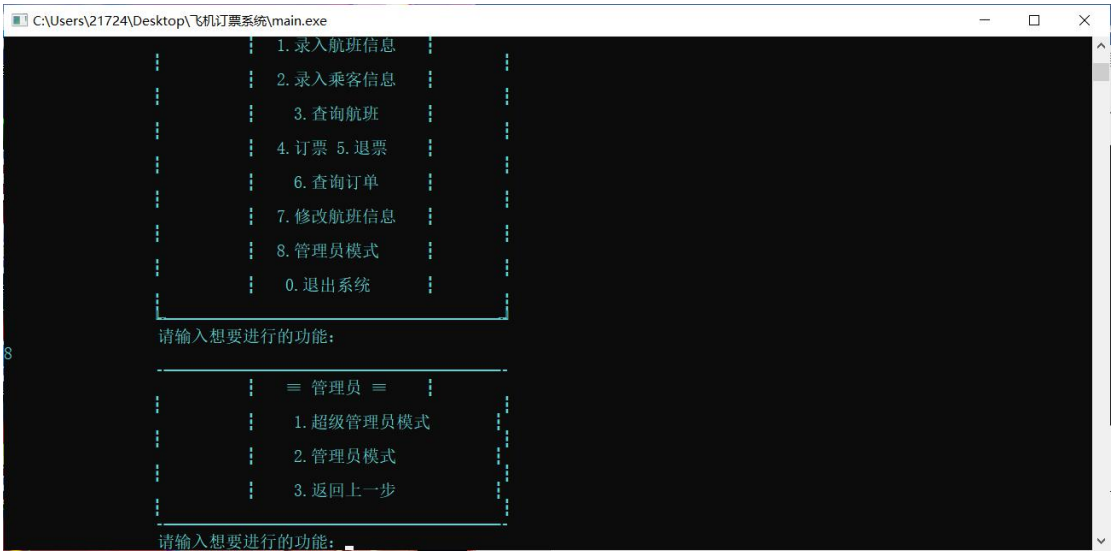


有好几种操作，通过测试分别成功



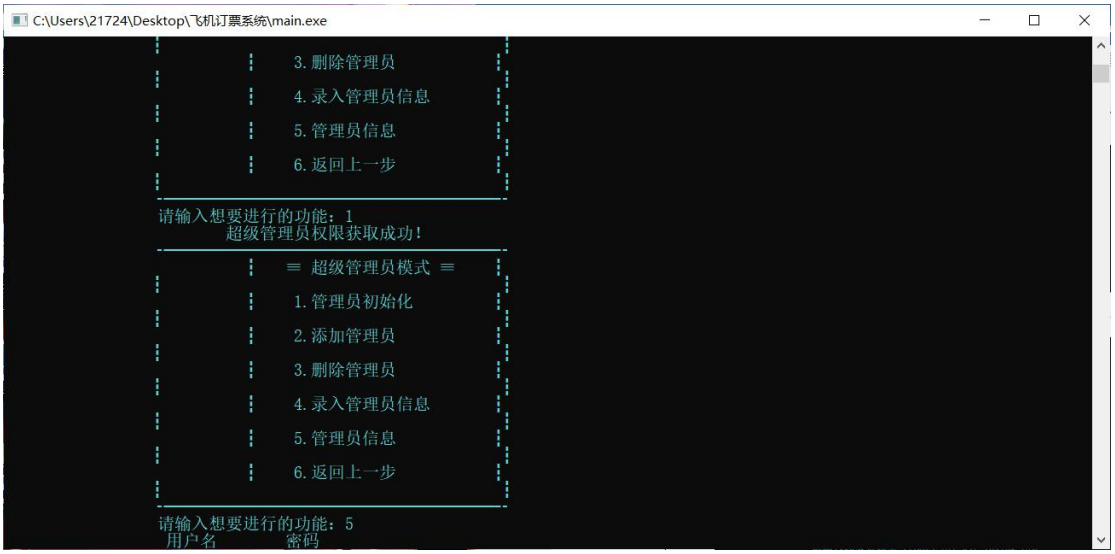


操作 8：进入管理员模式

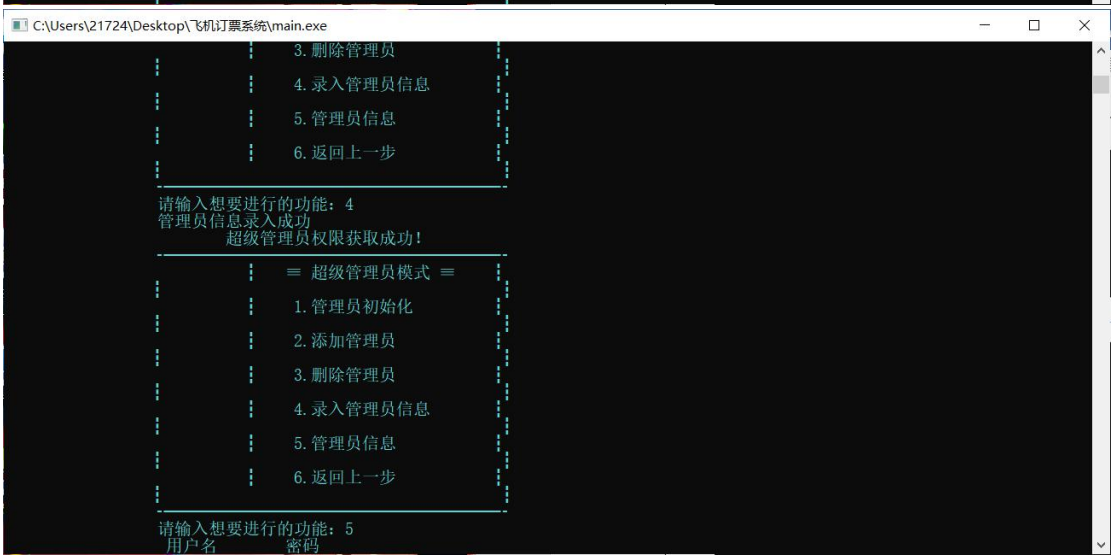


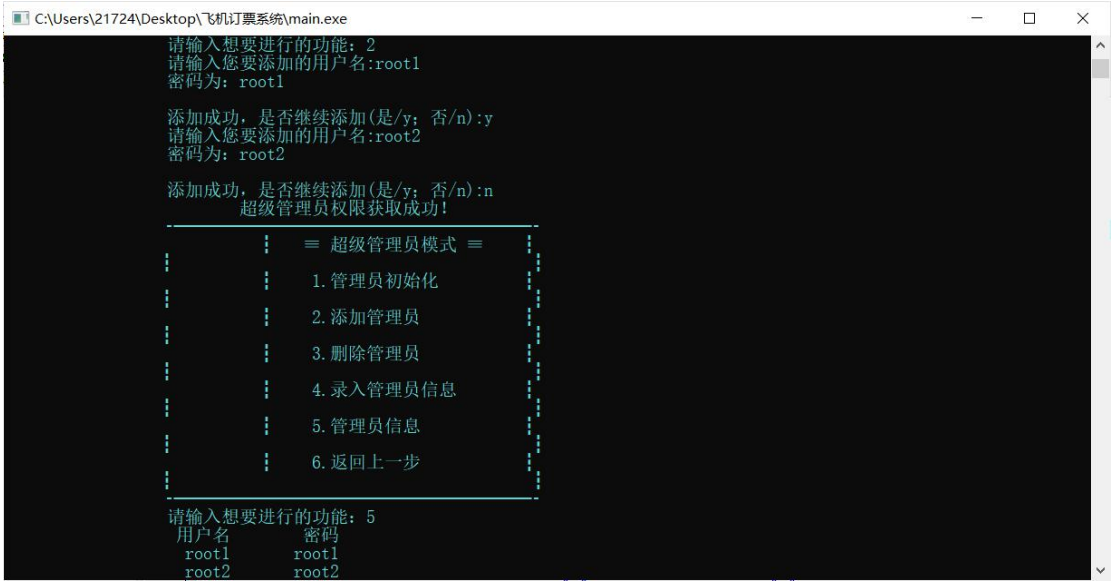
首先是超级管理员权限需要验证，输入正确的用户名和密码以后就可以获得超级管理员权限，看到只有超级 root 才能看到的界面。

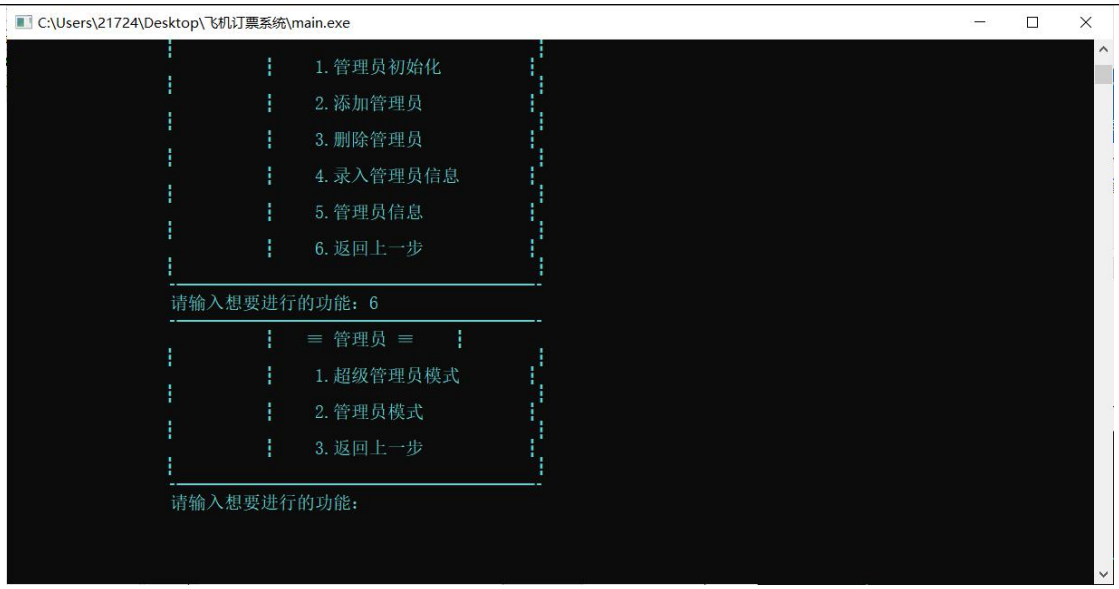
//超级管理员模式有 6 个功能，分别是管理员初始化，管理员添加，管理员删除，管理员信息录入，管理员信息展示，返回上一步，经过测试无误。

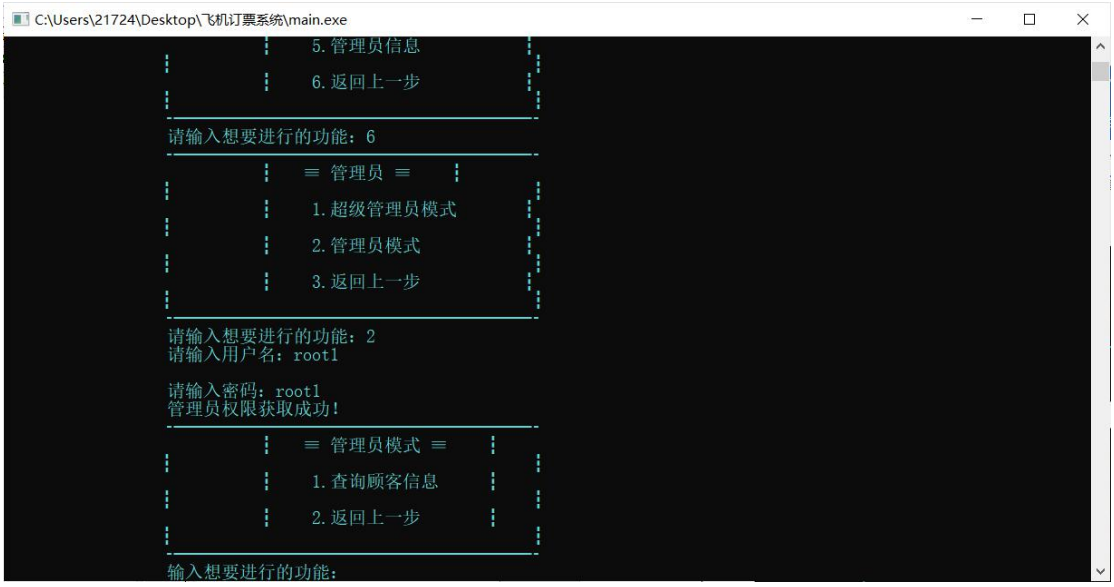



C:\Users\21724\Desktop\飞机订票系统\main.exe
请输入您要添加的用户名:root2
密码为：root2
添加成功，是否继续添加(是/y；否/n):n
超级管理员权限获取成功!
≡ 超级管理员模式 ≡
1.管理员初始化
2.添加管理员
3.删除管理员
4.录入管理员信息
5.管理员信息
6.返回上一步
请输入想要进行的功能：3
请输入管理员用户名:root2
正在删除.....
删除成功
超级管理员权限获取成功!
≡ 超级管理员模式 ≡


C:\Users\21724\Desktop\飞机订票系统\main.exe
3.删除管理员
4.录入管理员信息
5.管理员信息
6.返回上一步
请输入想要进行的功能：4
管理员信息录入成功
超级管理员权限获取成功!
≡ 超级管理员模式 ≡
1.管理员初始化
2.添加管理员
3.删除管理员
4.录入管理员信息
5.管理员信息
6.返回上一步
请输入想要进行的功能：5
用户名 密码


C:\Users\21724\Desktop\飞机订票系统\main.exe
请输入想要进行的功能：2
请输入您要添加的用户名:root1
密码为：root1
添加成功，是否继续添加(是/y；否/n):y
请输入您要添加的用户名:root2
密码为：root2
添加成功，是否继续添加(是/y；否/n):n
超级管理员权限获取成功!
≡ 超级管理员模式 ≡
1.管理员初始化
2.添加管理员
3.删除管理员
4.录入管理员信息
5.管理员信息
6.返回上一步
请输入想要进行的功能：5
用户名 密码
root1 root1
root2 root2





管理员模式也需要对应的管理员账号和密码

可以查看顾客信息等功能。

测试具体情况。

功能	测试
航班信息存储	√
航班信息录入	√
乘客信息存储	√
乘客信息录入	√

航班查询	√
航班起始点查询	√
航班号查询	√
终点查找	√
浏览全部航班	√
订票	√
有余票正常订票	√
没有余票咨询是否预约	√
如果没有对应航班，是否展示其他到达目的地航班	√
退票	√
处于订票状态退票后座次号和余票量是否更新	√
处于订票状态退票后是否按照递补顺序一一递补	√
处于候补状态退票后，是否不影响座次号和余票量	√
查询订单	√
给姓名和身份证号能否查到乘客信息	√
座次号是否显示正确	√
预约状态是否显示正确	√
预约的排位号是否显示正确	√
航班信息修改	√
删除航班	√
添加航班	√
修改起飞时间	√
修改航班时间	√
返回操作	√
管理员模式	√
超级管理员模式	√
管理员初始化	√
添加管理员	√
删除管理员	√
管理员信息录入	√
管理员信息展示	√
返回操作	√
管理员模式	√
乘客信息展示	√
返回操作	√

4. 分析与探讨

1.航班采用单链表连接起来，采用头插入法，插入的时间复杂度为0(1)，删除和查询的时间复杂度为0(n),空间复杂度都为0（1）；候补客户使用含有头指针和尾指针的链队列实现，所以入队和出队的时间复杂度和空间复杂度都为0（1）。

2. 航班和候补队列都要用到单链表，一开始我使用尾插入法的，后来觉得头插入法更简单，所以最后均采用了头插入法插入，为了插入和删除节点方便，使用了带头结点的链表和链队列。

3. 关于舱位等级和剩余票问题，一开始我是采用了票数和舱位等级无关，自己随意输入舱位等级都行，但是后来发现这样并不合理，后来就将总剩余票数分为两部分，一部分是经济舱剩余票数，一部分是商务舱剩余票，这样，在候补客户排队订票时，就也需要分开两条队列，一条是在经济舱队列排队，一条是在商务舱队列排队。

4. 在链表或者队列的最后一个节点的 next 指针域一定要置为 NULL，否则就会出现野指针，可能会导致实验出现错误。
5. 在输入的过程中，要注意参数值的合法性，比如票数不能出现负数，不合法的参数要求重新输入。
6. 在退票的时候原本我是采用了输入航班 ID 和姓名就能退票了，后来觉得不合理，增加了身份证号码验证，验证成功才能退票。
7. 在身份证的输入时有一个明显的不足，因为身份证不是随意输入一系列数字就行了，这里本来应该采用正则表达式进行判断的，但是由于编译器不带<regex.h>头文件，无法直接调用里面的函数进行判断，又受限于时间问题，因此身份证号码不够充分。
8. 当票数不足时，要采取多种判断来为客户服务，这就涉及到了很多 if else 的嵌套。
9. 在这次写代码的过程中，出现过很多问题，如指针错误，空指针异常，野指针等等，在处理这些 Bug 的时候，我深刻认识到了断点调试的重要性，只有断点调试，才能快速找出 Bug 并加以改正。
10. 通过这次课设，我加深了对指针、链表和队列的理解，同时也自我感觉动手能力和算法能力有了明显的提升。

5. 附录：实现源代码

```
//head.h
    /**********************
    * 版权所有 (C)2022, Yuxing Liu。
    *
    * 文件名称： head.cpp
    * 文件标识：无
    * 内容摘要：定义航班和用户的信息，完成各个函数的声明
    * 其它说明：无
    * 当前版本： V1.0
    * 作 者：刘宇星
    * 完成日期： 2022-05-10
    **********************/
    #include <bits/stdc++.h>
    using namespace std;
    typedef struct flightnode
    {
            char flight_num[10];//航班号
            char plane_num[10];//飞机号
            char time[20];//起飞时间
            char date[20];//日期
            char start_place[20];//出发地
            char end_place[20];//目的地
            int left;//余票
            int now_seat_num = 0;//当前的座次号
            float price;//价格
```

```
        flightnode *next;
}flightnode;
typedef struct administrator
{
        char name[20];
        char password[20];
        administrator *next;
        administrator *pre;
}administrator;
typedef struct passengernode
{
        char name[30];//名字
        char ID_num[30];//证件号
        char flight_num[10];//航班号
        char time[20];//起飞时间
        char start_place[20];//出发地
        char end_place[20];//目的地
        float price;//价格
        int full = 0;//是否处于候补状态
        int seat_num;//座次号
        passengernode *next;
}passengernode;
class Airline
{
        public:
          flightnode *head_of_flight;
          flightnode *tail_of_flight;
          passengernode *head_of_passenger;
          passengernode *tail_of_passenger;
          administrator *head_of_root;
          administrator *tail_of_root;
          char root_name[20] = "Yuxing_Liu";
          char root_password[20] = "20020316";
          void Init_flight();//初始化航班
          void Load_flight();//载入航班
          void Add_flight();//添加航班
          void Check_flight();//查找航班
          void Check_flightnum();//航班号查找
          void Check_seplace();//起始地查找
          void Check_all();//浏览全部航班
          void Check_ed();
          void Revise_flight();//修改航班信息
```

```
        void Delete_flight();//删除航班
        void Revise_time();//修改起飞抵达时间
        void Revise_price();//修改价格
        void Save_flight();//保存航班信息
        ///用户信息
        void passenger_display();//用户信息展示
        void Init_passenger();//初始化用户
        void Load_passenger();//载入用户信息
        void Book();//定票
        void Qbook();//退票
        void Check_book();//查询订单
        void Save_passenger();//保存用户信息
        //管理员模式
        int check();//能否获得超级管理员权限
        void root_display();//管理员展示
        void root_save();//管理员信息保存
        void root_load();//管理员信息载入
        void root_init();//管理员初始化
        void root_reset();//管理员清空
        void root_add();//添加管理员
        void root_del();//删除管理员
        int root_right();//管理员权限
        void root_mode();//管理员
};
/*
////航班类信息
void Init_flight(flightnode *&);//初始化航班
void Load_flight(flightnode *&);//载入航班
void Add_flight(flightnode *&);//添加航班
void Check_flight(flightnode *&);//查找航班
void Check_flightnum(flightnode *&);//航班号查找
void Check_seplace(flightnode *&);//起始地查找
void Check_all(flightnode *&);//浏览全部航班
void Revise_flight(flightnode *&);//修改航班信息
void Delete_flight(flightnode *&);//删除航班
void Revise_time(flightnode*&);//修改起飞抵达时间
void Revise_price(flightnode *&);//修改价格
void Save_flight(flightnode *&);//保存航班信息
///用户信息
void Init_passenger(passengernode *&);//初始化用户
void Load_passenger(passengernode *&);//载入用户信息
void Book(flightnode *&, passengernode *&);//定票
```

```
    void Qbook(flightnode *&, passengernode *&);//退票
    void Check_book(passengernode *&);//查询订单
    void Save_passenger(passengernode *&);//保存用户信息
    */
    /*******************
    * 功能描述：初始化航班
    * 输入参数： 无
    * 输出参数： 无
    * 返回值： void
    * 其它说明：无
    ********************/
    void Airline::Init_flight()//初始化航班
    {
            flightnode* h = (flightnode *)malloc(sizeof(flightnode));
            head_of_flight = tail_of_flight = h;
            h -> next = NULL;
    }
    /*******************
    * 功能描述：载入航班信息
    * 输入参数： 载入 FlightList.dat 信息到链表
    * 输出参数： 无
    * 返回值： void
    * 其它说明：无
    ********************/
    void Airline::Load_flight()//载入航班
    {
            flightnode *s;
            s = (flightnode *)malloc(sizeof(flightnode));
            ifstream infile("FlightList.dat", ios::in);//将文件载入
            if (!infile)
            {
              cerr << "                    信息错误。";//如果文件名错误的话，填写信
息错误
              return;
            }
            while (1)//读入相关的信息
            {
              infile >> s -> flight_num >> s -> plane_num >> s -> date >> s ->
time >> s -> start_place >> s -> end_place >> s -> price >> s -> left >> s ->
now_seat_num;
              if (!infile.eof())
              {
```

```
                tail_of_flight -> next = s;//不断更新尾结点和中间节点
                tail_of_flight = s;//不断更新尾结点和中间节点
                s = (flightnode *)malloc(sizeof(flightnode));//申请一个新的指针
空间
            }
            else
                break;
        }
        tail_of_flight -> next = NULL;//最终让尾指针指向 NULL
        free(s);//将 s 指针的空间释放
    }
    /*******************
    * 功能描述：添加航班
    * 输入参数： 航班的相关信息
    * 输出参数： 添加成功或者选择继续添加
    * 返回值： void
    * 其它说明：无
    *******************/
    void Airline::Add_flight()//添加航班
    {
        char mark = 'y';
        while (mark == 'y')
        {
          flightnode* s = (flightnode *)malloc(sizeof(flightnode));//新的航班
生命节点空间
          cout << "                    请输入航班号：";
          cin >> s -> flight_num;
          cout << "                    请输入飞机号：";
          cin >> s -> plane_num;
          cout << "                    请输入日期：";
          cin >> s -> date;
          cout << "                    请输入起飞时间：";
          cin >> s -> time;
          cout << "                    请输入出发地:";
          cin >> s -> start_place;
          cout << "                    请输入目的地：";
          cin >> s -> end_place;
          cout << "                    请输入价格：";
          cin >> s -> price;
          cout << "                    请输入余票：";
          cin >> s -> left;
          s -> now_seat_num = 0;//当前的座次号
```

```
            tail_of_flight -> next = s;//更新中间节点
            tail_of_flight = s;//更新尾结点
            cout << "                    添加成功！" << endl;
            cout << "                    是否继续添加？（是“y”/否“n”）：";
            cin >> mark;//判断是否需要继续添加
            cout << endl;
          }
          tail_of_flight -> next = NULL;//最后让尾节点指向 NULL，方便判断终止条
件
    }
    /*******************
    * 功能描述：查找航班
    * 输入参数： 选择查找方式
    * 输出参数： 进入相关函数进行查找功能
    * 返回值： void
    * 其它说明：无
    *******************/
    void Airline::Check_flight()//查找航班
    {
          system("cls");
          int i;
          cout << "                    ----------------------------------------
----------------------------------" << endl;
          cout << "                                ┆    ≡ 查找方式 ≡    ┆ " <<
endl;
          cout << "                ┆                                ┆
" << endl;
          cout << "                                ┆    1.航班号查找    ┆ " <<
endl;
          cout << "                ┆                                ┆
" << endl;
          cout << "                                ┆    2.起始地查找    ┆ " <<
endl;
          cout << "                ┆                                ┆
" << endl;
          cout << "                                ┆     3.终点查找     ┆ " <<
endl;
          cout << "                ┆                                ┆
" << endl;
          cout << "                                ┆   4.浏览全部航班   ┆ " <<
endl;
          cout << "                ┆                                ┆
```

```
" << endl;
            cout << "                    ---------------------------------------------------------
----------------------------------------- " << endl;
            cout << "                    请输入查找方式：";
            cin >> i;
            if (i == 1)
              Check_flightnum();//通过航班号查找
            else if (i == 2)
              Check_seplace();//通过起始地查找
            else if(i == 3)
              Check_ed();//终点查找
            else if (i == 4)
              Check_all();//浏览全部航班
    }
    /*******************
    * 功能描述：航班号查找
    * 输入参数： 航班号
    * 输出参数： 航班的相关信息
    * 返回值： void
    * 其它说明：无
    *******************/
    void Airline::Check_flightnum()//航班号查找
    {


            char flightnum[10];
            flightnode *p = head_of_flight -> next;//定义 p 为头节点的下一个节点，
这样的话就是第一个数据
            cout << setw(32) << "请输入航班号：";
            cin >> flightnum;//读取航班号
            cout << setw(25) << "航班号" << setw(12) << "飞机号" << setw(12) << "
日期" << setw(12) << "起飞时间" << setw(12) << "出发地" << setw(12) << "目的地" <<
setw(12) << "价格" << setw(12) << "余票" << endl;
            while (p != NULL)//如果 p 没有到达终点
            {
              if (strcmp(p->flight_num, flightnum) == 0)//看能否匹配航班号
              {
                 cout << setw(25) << p -> flight_num << setw(12) << p ->
plane_num << setw(12) << p -> date << setw(12) << p -> time << setw(12) << p ->
start_place << setw(12) << p -> end_place << setw(12) << p->price << setw(12) <<
p->left << endl;
                 cout << "                  ";//缩进处理
                 system("pause");
```

```
                return;
            }
            else
                p = p -> next;//否则就继续往下找
        }
        cout << setw(32) << "未查到任何信息。";//处理缩进
        system("pause");
    }
    /*******************
    * 功能描述：起始地查找
    * 输入参数： 起始地
    * 输出参数： 航班的相关信息
    * 返回值： void
    * 其它说明：无
    *******************/
    void Airline::Check_seplace()//起始地查找
    {
        char start_place[20], end_place[20];//起始地的字符串
        cout << "                    请输入出发地：";
        cin >> start_place;//输入起点
        cout << "                    请输入目的地：";
        cin >> end_place;//输入终点
        flightnode *p = head_of_flight -> next;//p 为头指针的下一个指针，即第一个存放数据的指针
        cout << setw(25) << "航班号" << setw(12) << "飞机号" << setw(12) << "日期" << setw(12) << "起飞日期" << setw(12) << "出发地" << setw(12) << "目的地" << setw(12) << "价格" << setw(12) << "余票" << endl;
        int flag = 0;
        while (p != NULL)//如果没有将所有的航班遍历完
        {
            if (strcmp(p -> start_place, start_place) == 0 && strcmp(p -> end_place, end_place) == 0)//看起点和终点能否分别匹配
            {
                flag = 1;//标记为 1
                cout << setw(25) << p -> flight_num << setw(12) << p -> plane_num << setw(12) << p -> date << setw(12) << p -> time << setw(12) << p->start_place << setw(12) << p->end_place << setw(12) << p->price << setw(12) << p->left << endl;
            }
            p = p -> next;
        }
        if(!flag)
```

```
            cout << setw(32) << "未查到任何信息" << endl;//如果没有被标记的话，就输出
            cout << setw(32);//缩进
            system("pause");
    }
    /*******************
    * 功能描述：根据目的地查询航班
    * 输入参数： 无
    * 输出参数： 输出航班的相关信息
    * 返回值： void
    * 其它说明：无
    *******************/
    void Airline::Check_ed()
    {
            char end_place[20];//终点站的字符串名字
            cout << "                    请输入目的地：";
            cin >> end_place;//读入终点站
            cout << setw(25) << "航班号" << setw(12) << "飞机号" << setw(12) << "日期" << setw(12) << "起飞日期" << setw(12) << "出发地" << setw(12) << "目的地" << setw(12) << "价格" << setw(12) << "余票" << endl;
            flightnode *p = head_of_flight -> next;//航班头结点的下一个节点
            int flag = 0;//判断是否能查找到相关信息
            while (p != NULL)
            {
              if (strcmp(p -> end_place, end_place) == 0)//如果能够匹配的话
              {
                  cout << setw(25) << p -> flight_num << setw(12) << p -> plane_num << setw(12) << p -> date << setw(12) << p -> time << setw(12) << p->start_place << setw(12) << p->end_place << setw(12) << p->price << setw(12) << p->left << endl;
                  flag = 1;//认为能找到对应的航班
              }
              p = p -> next;//继续往下找
            }
            if(!flag)    cout << setw(32) << "未查到任何信息" << endl;//如果查找不到任何信息的话
            cout << setw(32);//处理缩进
            system("pause");
    }
    /*******************
    * 功能描述：浏览全部航班
    * 输入参数： 无
```

```
    * 输出参数： 航班的相关信息
    * 返回值： void
    * 其它说明：无
    ********************/
    void Airline::Check_all()//浏览全部航班
    {

            flightnode *p = head_of_flight -> next;//p 节点的下一个节点
            cout << setw(25) << "航班号" << setw(12) << "飞机号" << setw(12) << "日期" << setw(12) <<  "起飞日期" << setw(12) << "出发地" << setw(12) << "目的地" << setw(12) << "价格" << setw(12) << "余票" << endl;
            while (p != NULL)//如果还没到尾结点
            {
              cout << setw(25) << p -> flight_num << setw(12) << p -> plane_num << setw(12) << p -> date << setw(12) << p -> time << setw(12) << p -> start_place << setw(12) << p->end_place << setw(12) << p->price << setw(12) << p->left << endl;
              p = p -> next;//继续往后找
            }
    }
    /********************
    * 功能描述：修改航班信息
    * 输入参数：无
    * 输出参数： 调用相关函数进行实现
    * 返回值： void
    * 其它说明：无
    ********************/
    void Airline::Revise_flight()//修改航班信息
    {
            flightnode *f = head_of_flight -> next;//首节点的下一个节点，第一个存有数据的指针节点
            int i;
            cout << "                    ----------------------------------------
----------------------------------" << endl;
            cout << "                                ┆     ≡ 修改方式 ≡     ┆ " << endl;
            cout << "                    ┆                                  ┆ " << endl;
            cout << "                                ┆     1.删除航班       ┆ " << endl;
            cout << "                    ┆                                  ┆ " << endl;
            cout << "                                ┆     2.添加航班       ┆ " <<
```

```
endl;
            cout << "                    ┆                                    ┆
" << endl;
            cout << "                                ┆    3.修改起飞时间    ┆ " <<
endl;
            cout << "                    ┆                                    ┆
" << endl;
            cout << "                                ┆    4.修改航班价格    ┆ " <<
endl;
            cout << "                    ┆                                    ┆
" << endl;
            cout << "                                ┆       5.返回         ┆ " <<
endl;
            cout << "                    ┆                                    ┆
" << endl;
            cout << "                     ----------------------------------------
----------------------------------- " << endl;
            cout << "                    请输入查找方式：";
            cin >> i;
            if (i == 1)
              Delete_flight();//删除航班
            else if (i == 2)
              Add_flight();//添加航班
            else if(i == 3)
              Revise_time();//修改起飞时间
            else if (i == 4)
              Revise_price();//修改航班价格
            else if (i == 5)//否则返回
              return;
    }
    /*******************
    * 功能描述：删除航班
    * 输入参数：航班号
    * 输出参数： 删除成功
    * 返回值： void
    * 其它说明：无
    *******************/
    void Airline::Delete_flight()//删除航班
    {
            flightnode *p = head_of_flight -> next, *q = head_of_flight;//q 用来代
表当前节点的前一个节点
            char flightnum[10];//航班号
```

```
            cout << "                    请输入您想要修改的航班号：";//输入想要修改的航班号
            cin >> flightnum;
            while (p != NULL)//从前到后找到这个
            {
              if (strcmp(p -> flight_num, flightnum) == 0)//匹配航班号
              {
                  if(p -> next == NULL)//如果尾结点是最后一个节点
                  {
                      tail_of_flight = q;//那么就将尾结点往前移一个
                      q -> next = NULL;//尾结点的下一个节点为NULL
                  }
                  else
                  {
                      q -> next = p -> next;//更新当前节点的前一个节点
                  }
                  vector<passengernode *> v;
                  passengernode *ptr1 = head_of_passenger -> next;
                  passengernode *ptr2 = head_of_passenger;
                  for(; ptr1 != NULL;)//删除这个航班对应的所有乘客信息
                  {
                      if(strcmp(ptr1 -> flight_num, flightnum) == 0)//如果航班号能够匹配
                      {
                          if(ptr1 -> next == NULL)//航班节点的尾结点
                          {
                              tail_of_passenger = ptr2;//更新乘客节点的尾结点
                              ptr2 -> next = NULL;//节点的下一个节点为NULL
                          }
                          else
                          {
                              ptr2 -> next = ptr1 -> next;//更新前一个节点的下一个节点为当前节点的下一个节点
                          }
                          v.push_back(ptr1);//放入动态数组当中
                          ptr1 = ptr2 -> next;//继续往下走
                      }
                      else
                      {
                          ptr1 = ptr1 -> next;//否则两个节点都往后走
                          ptr2 = ptr2 -> next;//继续遍历节点
                      }
```

```
                }
                for(auto ptr : v)//枚举所有的指针节点
                {
                    free(ptr);//释放内存空间
                }
                free(p);//释放当前节点的空间
                cout << "                    删除成功！";
                return;
            }
            q = q -> next;//同时往后走
            p = q -> next;//同时往后走
        }
    }
    /*******************
    * 功能描述：修改起飞抵达时间
    * 输入参数：航班号，新的航班号
    * 输出参数： 修改成功
    * 返回值： void
    * 其它说明：无
    *******************/
    void Airline::Revise_time()//修改起飞抵达时间
    {
        char flightnum[10];
        flightnode *p = head_of_flight -> next;//航班头节点的下一个节点
        cout << "                    请输入您想要修改的航班号：";
        cin >> flightnum;//航班号
        while (p != NULL)//如果 p 节点不是尾结点
        {
          if (strcmp(p -> flight_num, flightnum) == 0)//如果能够匹配航班号
          {
              cout << "                    请输入新的起飞时间：";
              cin >> p -> time;
              cout << "                    修改成功！";
              return;
          }
          p = p -> next;//向后遍历节点
        }
        cout << "                    没有您想要修改的航班号!";
        system("pause");
    }
    /*******************
    * 功能描述：修改价格
```

```
    * 输入参数：航班号，新的价格
    * 输出参数： 修改成功
    * 返回值： void
    * 其它说明：无
    ********************/
    void Airline::Revise_price()//修改价格
    {
            char flightnum[10];
            flightnode *p = head_of_flight -> next;//p 节点是头结点的下一个节点
            cout << "                    请输入您想要修改的航班号：";
            cin >> flightnum;//读入航班号
            while (p != NULL)//如果 p 指针不是尾指针的话
            {
              if (strcmp(p -> flight_num, flightnum) == 0)//flight_num 和 flightnum
能够对应起来
              {
                  cout << "                    输入新的价格：";
                  cin >> p -> price;
                  cout << "                    修改成功！  ";
                  return;
              }
            }
            cout << "                    没有您想要修改的航班号!";
            system("pause");
    }
    /********************
    * 功能描述：保存航班信息
    * 输入参数：无
    * 输出参数：用户的信息到文件 FlightList.dat
    * 返回值： void
    * 其它说明：
    ********************/
    void Airline::Save_flight()//保存航班信息
    {
            flightnode *f = head_of_flight -> next;//f 为头结点的下一个节点
            ofstream outfile("FlightList.dat", ios::trunc);//倒入节点的相关信息
            if (!outfile)//如果打开失败的话
            {
              cerr << "                    存储失败！";//输出存储失败
              return;
            }
            while (f != NULL)//如果 f 节点不是尾结点的话
```

```
            {
              outfile << f -> flight_num << " " << f -> plane_num << " " << f -> date << " " << f -> time << " " << f -> start_place << " " << f -> end_place << " " << f -> price << " " << f -> left << " " << f -> now_seat_num << endl;
              f = f -> next;//更新 f 节点的下一个节点
            }
            outfile.close();//关闭输出文件
    }
    /********************
    * 功能描述：初始化用户
    * 输入参数： 无
    * 输出参数： 无
    * 返回值： void
    * 其它说明：无
    ********************/
    void Airline::Init_passenger()//初始化用户
    {
            passengernode* c = (passengernode *)malloc(sizeof(passengernode));//新的节点声明
            head_of_passenger = tail_of_passenger = c;//头结点等于尾结点等于 c
            c -> next = NULL;//更新节点的下一个节点为 NULL
    }
    /********************
    * 功能描述：用户信息展示
    * 输入参数：无
    * 输出参数：无
    * 其他说明：无
    ********************/
    void Airline::passenger_display()//用户信息展示
    {
            passengernode *p = head_of_passenger -> next;//p 节点为头结点的下一个节点
            cout << setw(25) << "预约状态？" << setw(12) << "乘客姓名" << setw(12) << "乘客 ID" << setw(12) << "乘坐航班号" << setw(12) << "座次" << setw(12) << "航班时间" << setw(12) << "始发点" << setw(12) << "终点" << setw(12) << "票价" << endl;
            for(p; p != NULL; p = p -> next)//遍历链表中的节点
            {
              cout << setw(25) << p -> full << setw(12) << p -> name << setw(12) << p -> ID_num << setw(12) << p -> flight_num << setw(12) << p -> seat_num << setw(12) << p -> time << setw(12) << p -> start_place << setw(12) << p -> end_place << setw(12) << p -> price << endl;
```

```
            }
            return;//输出后返回即可
    }
    /*******************
    * 功能描述：载入用户信息
    * 输入参数： 载入 PassengerList.dat 信息到链表
    * 输出参数： 无
    * 返回值： void
    * 其它说明：无
    *******************/
    void Airline::Load_passenger()//载入用户信息
    {
            passengernode *s;//定义这个乘客节点 s
            s = (passengernode *)malloc(sizeof(passengernode));//为新的节点申请命
名空间
            ifstream infile("PassengerList.dat", ios::in);//载入这个信息
            if (!infile)//如果读取不到这个文件的话
            {
              cerr << "                    信息错误。";//认为信息错误
              return;
            }
            while (1)
            {
              infile >> s -> name >> s -> ID_num >> s -> flight_num >> s ->
seat_num >> s -> time >> s -> start_place >> s -> end_place >> s -> price;
              if (!infile.eof())//读入相关的数据
              {
                  tail_of_passenger -> next = s;//乘客节点的下一个节点为 s
                  tail_of_passenger = s;//更新尾结点
                  s = (passengernode *)malloc(sizeof(passengernode));//声明一个新
的节点
              }
              else
                  break;
            }
            tail_of_passenger -> next = NULL;//节点的后一个节点为 NULL
            free(s);//释放这个节点
    }
    /*******************
    * 功能描述：航班信息
    * 输入参数：passenger 的相关信息
                flight 的相关信息
```

```
    * 输出参数： 购票成功 or 失败
    * 返回值： void
    * 其它说明：如果余票不足还可以预定，如果有多余的票即可瞬间购买
    ********************/
    void Airline::Book()//订票
    {
            char start_place[20], end_place[20], flightnum[10];//航班信息
            flightnode *p = head_of_flight -> next, *q = head_of_flight -> next;//
两个节点
            passengernode *s;//客户节点
            char mark, check = '1';//界面是否重复展示
            s = (passengernode *)malloc(sizeof(passengernode));//新的乘客节点
            cout << "                    请输入您的姓名：";
            cin >> s -> name;
            cout << "                    请输入您的证件号：";
            cin >> s -> ID_num;
            cout << "                    请输入出发地:";
            cin >> start_place;
            cout << "                    请输入目的地:";
            cin >> end_place;
            cout << setw(25) << "航班号" << setw(12) << "起飞日期" << setw(12) <<
"出发地" << setw(12) << "目的地" << setw(12) << "价格" << setw(12) << "余票" <<
endl;
            while (p != NULL)//如果不是终点
            {
              if (strcmp(p -> start_place, start_place) == 0 && strcmp(p ->
end_place, end_place) == 0)
              {
                  cout << setw(25) << p -> flight_num << setw(12) << p -> time <<
setw(12) << p -> start_place << setw(12) << p->end_place << setw(12) << p->price
<< setw(12) << p->left << endl;
                  check = '0';//说明找到了
              }
              p = p -> next;
            }
            if (check == '1')//否则说明没有找到qaq
            {
              cout << "                    没有相关的航班，";
              system("pause");
              cout << "                    请问是否需要查看其他能够到达同一目的地的
航班(是 y / 否 n)";
              cin >> mark;//如果继续添加的话
```

```
            if(mark == 'y')
            {
                p = head_of_flight -> next;//头结点的下一个节点
                cout << setw(25) << "航班号" << setw(12) << "起飞日期" << setw(12) << "出发地" << setw(12) << "目的地" << setw(12) << "价格" << setw(12) << "余票" << endl;
                while(p != NULL)//指针不等于 NULL
                {
                    if(strcmp(p -> end_place, end_place) == 0)//匹配一下终点站
                    {
                        cout << setw(25) << p -> flight_num << setw(12) << p -> time << setw(12) << p -> start_place << setw(12) << p -> end_place << setw(12) << p -> price << setw(12) << p -> left << endl;
                    }
                    p = p -> next;//向后遍历节点
                }
                cout << "                    ";//输出缩进
                system("pause");
                return;
            }
            else
            return;//返回节点
          }
          cout << "                    请输入您想要购买的航班号：";
          cin >> flightnum;//航班号
          puts("");
          while (q != NULL)//q 不是 NULL 节点
          {
            if (strcmp(q -> flight_num, flightnum) == 0 && strcmp(q -> start_place, start_place) == 0 && strcmp(q->end_place, end_place) == 0)//匹配起始点是否相同
            {
                if (q -> left == 0)//如果余票为 0 了
                {
                    cout << "                    余票不足，请选择是否预约（是 y / 否 n）";//输出余票不足
                    cin >> mark;//读入指令
                    if (mark == 'y')//如果指令为 y 的话
                    {
                        s -> price = q -> price;//价格更新
                        s -> seat_num = 0;//座次号更新
                        s -> full = 1;//表示在预约状态
```

```
            strcpy(s -> start_place, start_place);//复制起点
            strcpy(s -> end_place, end_place);//复制终点
            strcpy(s -> time, q -> time);//复制时间
            strcpy(s -> flight_num, flightnum);//复制航班号
            cout << "                    预约成功！" << endl;
            tail_of_passenger -> next = s;//乘客的下一个节点为 s
            tail_of_passenger = s;//乘客节点更新为 s
            tail_of_passenger -> next = NULL;//乘客节点的尾结点为
            cout << "                    ";
            system("pause");//系统停止一会
            return;
        }
        else
            return;
    }
    /*
        char name[30];//名字
        char ID_num[30];//证件号
        char flight_num[10];//航班号
        char time[20];//起飞时间
        char start_place[20];//出发地
        char end_place[20];//目的地
        float price;//价格
        int full = 0;//是否处于候补状态
        int seat_num;//座次号
    */
    strcpy(s -> flight_num, flightnum);//复制航班号
    strcpy(s -> start_place, start_place);//复制起始地点
    strcpy(s -> end_place, end_place);//复制终止地点
    strcpy(s -> time, q->time);//复制时间
    s -> full = 0; //处于订票状态
    s -> price = q -> price;//价格更新
    s -> seat_num = q -> now_seat_num + 1;//座次号更新
    tail_of_passenger -> next = s;//更新尾节点
    tail_of_passenger = s; //乘客的尾结点
    tail_of_passenger -> next = NULL;//最后一个节点的 next 为 NULL
    //购票成功
    cout << "                    购票成功！" << endl;
    q -> left --;//余票减一
    q -> now_seat_num ++;//当前的座次号+1
    cout << "                    ";
    system("pause");
```

```
                return;
            }
            else
                q = q -> next;
        }
        cout << "                    航班号填入错误\n";
        cout << "                    ";
        system("pause");
    }
    /*******************
    * 功能描述：退票
    * 输入参数：passenger 的相关信息
    * 输出参数： 退票成功 or 失败
    * 返回值： void
    * 其它说明：
    *******************/
    void Airline::Qbook()//退票
    {
        char name[30];
        char ID_num[30];
        flightnode *f = head_of_flight -> next;//退票系统
        passengernode *p = head_of_passenger -> next, *q = head_of_passenger,
*t = head_of_passenger -> next;
        cout << "                    请输入您的姓名：";
        cin >> name;//读入姓名
        puts("");
        cout << "                    请输入您的证件号：";
        cin >> ID_num;//读入乘客的身份证号码
        while (p != NULL)
        {
            if (strcmp(p -> name, name) == 0 && strcmp(p -> ID_num, ID_num) ==
0)//姓名和身份证号匹配
            {
                while (f != NULL)//如果没到终点的话
                {
                    if (strcmp(p -> flight_num, f -> flight_num) == 0)//航班号匹
配
                    {
                        f -> left ++;//余量+1
                        f -> now_seat_num --;//当前座次号减一
                        while (t != NULL)
                        {
```

```
                        if (strcmp(t -> flight_num, p -> flight_num) == 0 && t
-> full == 1)
                        {
                            strcpy(t -> start_place, p -> start_place);//复制
起始点
                            strcpy(t -> end_place, p -> end_place);//复制终点
                            strcpy(t -> time, p -> time);//复制时间
                            t -> price = p -> price;
                            t -> full = 0;//是否处于预约状态
                            t -> seat_num = f -> now_seat_num + 1;//座次号更新
                            f -> left --;//余票量减一
                            break;
                        }
                        t = t -> next;
                    }
                    break;
                }
                f = f -> next;
            }
            if(p == tail_of_passenger)
            {
                tail_of_passenger = q;
                tail_of_passenger -> next = NULL;
            }
            else
            {
                q -> next = p -> next;
            }
            for(passengernode *ptr = q -> next; ptr != NULL; ptr = ptr ->
next)
            {
                if(strcmp(ptr -> flight_num, p -> flight_num) == 0 && ptr ->
full == 0)
                {
                    ptr -> seat_num --;//把对应该航班号的所有处于不是预约状态
的座次号更新
                }
            }
            free(p);//释放该节点的内存
            cout << "                    退票成功！";
            return;
        }
```

```
            q = q -> next;
            p = q -> next;
          }
    }
    /*******************
    * 功能描述：查询订票
    * 输入参数：passenger 的相关信息
    * 输出参数： 所查用户订票的信息
    * 返回值： void
    * 其它说明：
    *******************/
    void Airline::Check_book()//查询订票
    {
          char name[30];
          char ID_num[30];
          passengernode *p = head_of_passenger -> next;//头结点的下一个节点
          passengernode *q = head_of_passenger -> next;
          cout << "                    请输入您的姓名：";
          cin >> name;
          cout << "                    请输入您的证件号：";
          cin >> ID_num;
          cout << setw(25) << "姓名" << setw(12) << "航班号" << setw(12) << "座次" << setw(12) << "起飞日期" << setw(12) << "出发地" << setw(12) << "目的地" << setw(12) << "价格" << setw(12) << endl;
          int rank = 0;//当前的为此
          while (p != NULL)
          {
            if (strcmp(p -> name, name) == 0 && strcmp(p -> ID_num, ID_num) == 0)
            {
                if(p -> full)
                {
                    for(q; q != NULL; q = q -> next)
                    {
                        if(q -> full && q -> flight_num == p -> flight_num)
                        {
                            rank ++;//如果处于预约状态那么 rank++
                        }
                    }
                    cout << "                    目前该乘客正处于预约状态，排在第" << rank + 1 << "位\n";
                }
```

```
                else cout << "                    目前乘客已成功订票，乘客信息如下所示\n";
                cout << setw(25) << p -> name << setw(12) << p -> flight_num << setw(12) << p -> seat_num << setw(12) << p->time << setw(12) << p->start_place << setw(12) << p->end_place << setw(12) << p->price << setw(12) << endl;
                cout << "                    ";
                system("pause");
                return;
              }
              else
                p = p -> next;//节点继续往下走啊
            }
            cout << setw(32) << "未查到任何信息。";
            system("pause");
    }
    /*******************
    * 功能描述：保存用户信息
    * 输入参数：无
    * 输出参数：用户的信息到文件 PassengerList.dat
    * 返回值： void
    * 其它说明：
    *******************/
    void Airline::Save_passenger()//保存用户信息
    {
            passengernode *p = head_of_passenger -> next;//头结点往后走
            ofstream outfile("PassengerList.dat", ios::trunc);//乘客信息
            if (!outfile)
            {
              cerr << "                    存储失败！";
              return;
            }
            while (p != NULL)//节点不是最后一个节点的话
            {
              outfile << p -> name << " " << p -> ID_num << " " << p -> flight_num << " " << p -> seat_num << " " << p -> time << " " << p->start_place << " " << p->end_place << " " << p->price << endl;
              p = p -> next;//读出这个节点的下一个节点
            }
            outfile.close();//关闭文件
    }
    void Airline::root_init()
    {
```

```
        administrator *s = (administrator *)malloc(sizeof(administrator));//初始化管理员
        head_of_root = tail_of_root = s;//定义一个新的节点后开始更新
        s -> next = NULL;
    }
    int Airline::check()
    {
        administrator *ptr = (administrator *)malloc(sizeof(administrator));//申请一个新的指针
        cout << endl << "                    请输入超级管理员（root）用户名：" ;
        cin >> ptr -> name;
        if(strcmp(ptr -> name, root_name) == 1) //看名字是否能够匹配
        {
          cout << endl << "                     用户名错误\n";
          return 0;
        }
        printf("\n");
        cout << endl << "                    请输入密码:";
        cin >> ptr -> password;//读入密码即可
        if(strcmp(ptr -> password, root_password) == 1)//密码能否匹配
        {
          cout << endl << "                     密码错误\n";
          return 0;
        }
        printf("\n");
        return 1;
    }
    /*******************
    * 功能描述：管理员信息清零
    * 输入参数： 载入 FlightList.dat 信息到链表
    * 输出参数： 无
    * 返回值： void
    * 其它说明：无
    *******************/
    void Airline::root_reset()
    {
        if(tail_of_root == head_of_root) return;//一次删除所有的管理员信息
        administrator* ptr = tail_of_root -> pre;
        administrator* s;
        for(ptr; ptr != head_of_root; ptr = ptr -> pre)
        {
          s = ptr -> next;
```

```
            free(s);//释放这个节点的内存
          }
          if(head_of_root -> next != NULL)
          {
            free(head_of_root -> next);//把初始节点的内存释放
            head_of_root -> next = NULL;
          }
    }
    /*******************
    * 功能描述：管理员添加
    * 输入参数： 无
    * 输出参数： 无
    * 返回值： void
    * 其它说明：无
    *******************/
    void Airline::root_add()
    {
          while(1)
          {
            cout << "                    请输入您要添加的用户名:";//输入要添加的用
户名
            administrator* s = (administrator *)malloc(sizeof(administrator));
            cin >> s -> name;//读入用户名的信息
            cout << "                    密码为：";
            cin >> s -> password;//读入用户名的密码
            puts("");
            tail_of_root -> next = s;//更新尾结点
            s -> pre = tail_of_root;//更新前置节点
            tail_of_root = s;//更新尾结点
            cout << "                    添加成功，是否继续添加(是/y；否/n):";
            char p;
            cin >> p;//读入操作
            if(p == 'n') break;
          }
          tail_of_root -> next = NULL;//更新尾结点的下一个节点为 NULL
    }
    /*
    void Airline::Add_flight()//添加航班
    {
          char mark = 'y';
          while (mark == 'y')
          {
```

```
            flightnode* s = (flightnode *)malloc(sizeof(flightnode));
            cout << "                    请输入航班号：";
            cin >> s -> flight_num;
            cout << "                    请输入起飞时间：";
            cin >> s -> time;
            cout << "                    请输入出发地:";
            cin >> s -> start_place;
            cout << "                    请输入目的地：";
            cin >> s -> end_place;
            cout << "                    请输入价格：";
            cin >> s -> price;
            cout << "                    请输入余票：";
            cin >> s -> left;
            tail_of_flight -> next = s;
            tail_of_flight = s;
            cout << "                    添加成功！" << endl;
            cout << "                    是否继续添加？（是“y”/否“n”）：";
            cin >> mark;
          }
          tail_of_flight -> next = NULL;
    }*/
    /********************
    * 功能描述：管理员删除
    * 输入参数： 无
    * 输出参数： 无
    * 返回值： void
    * 其它说明：无
    ********************/
    void Airline::root_del()//删除管理员
    {
          administrator *p = (administrator *)malloc(sizeof(administrator)); //
生命一个新的节点命名空间
          cout << "                    请输入管理员用户名:";
          cin >> p -> name;//读入节点的名字
          puts("");
          cout << "                    正在删除.....................\n"; //
操作
          for(administrator *ptr = head_of_root -> next; ptr != NULL; ptr = ptr
-> next)//遍历管理员链表中的每个管理员
          {
            if(strcmp(ptr -> name, p -> name) == 0)//如果能够匹配名字
            {
```

```
                if(ptr -> next == NULL)//如果下一个节点为NULL
                {
                    tail_of_root = ptr -> pre;//就第一种更新方式
                    tail_of_root -> next = NULL;//尾结点的下一个节点为NULL
                    free(ptr);//释放这个节点的空间
                    break;
                }
                else
                {
                    ptr -> pre -> next = ptr -> next;//更新这个节点前置节点的后继节点
                    ptr -> next -> pre = ptr -> pre;//更新后继节点的前驱节点
                    free(ptr);//释放这个节点
                    break;
                }
            }
        }
        cout << "                    删除成功\n";
    }
    void Airline::root_load()//管理员信息载入
    {
        administrator *s;//定义这个节点
        s = (administrator *)malloc(sizeof(administrator));//给这个节点分配命名空间
        ifstream infile("root.dat", ios::in);//载入管理员的相关信息
        if (!infile)//如果载入失败的话
        {
          cerr << "                    信息错误。";
          return;
        }
        while (1)
        {
          infile >> s -> name >> s -> password;//读入用户名和密码
          if (!infile.eof())//如果读入失败的话
          {
              tail_of_root -> next = s;//尾结点的下一个节点更新为s
              s -> pre = tail_of_root;//s的前驱节点更新为尾结点
              tail_of_root = s;//尾结点更新为s
              s = (administrator *)malloc(sizeof(administrator));//给s分配内存空间
          }
          else
```

```
                break;//否则就载入结束了
            }
            tail_of_root -> next = NULL;//尾结点的下一个节点为 NULL
            free(s);        //释放 s 这个节点
    }
    void Airline::root_save()//管理员信息保存
    {
            administrator *f = head_of_root -> next;//定义 f 为头结点的下一个节点
            ofstream outfile("root.dat", ios::trunc);//保存的文件为 root.dat
            if (!outfile)//如果打不开对应的文件
            {
              cerr << "                    存储失败！";//输出存储失败
              return;
            }
            while (f != NULL)//如果 f 不是 NULL 的话
            {
              outfile << f -> name << " " << f -> password << endl; //输出对应的所
有信息
              f = f -> next;//移动到下一个指针
            }
            outfile.close();//输出文件结束
    }
    int Airline::root_right()//管理员权限
    {
            int flag = 0;
            administrator *p = (administrator *)malloc(sizeof(administrator)); //
声明节点 p，分配命名空间
            cout << "                    请输入用户名："; //读入用户名
            cin >> p -> name;//读入对应的用户名
            puts("");
            for(administrator *ptr = head_of_root -> next; ptr != NULL; ptr = ptr
-> next)//遍历所有的管理员信息
            {
              if(strcmp(ptr -> name , p -> name) == 0)//如果能够匹配相关的信息的话
              {
                  flag = 1;//让 flag=1 认为找到节点了
                  cout << "                    请输入密码：";
                  cin >> p -> password;//读入密码
                  if(strcmp(ptr -> password, p -> password) == 0)
                  {
                      flag = 2;//如果用户名和密码都能够匹配的话
                      cout << "                    管理员权限获取成功！\n";
```

```
                    break;//获得管理员权限
                }
            }
        }
        if(flag == 0)//如果用户名错误的话
        {
            cout << endl;
            cout << "                    用户名错误\n";
            return 0;
        }
        else if(flag == 1)//如果密码错误的话
        {
            cout << "                    密码错误\n";
            return 0;
        }
        else return 1;//否则认为获得权限
    }
    void Airline::root_mode()//管理员模式
    {
        while(1)
        {
            cout << "                    ----------------------------------------
----------------------------------------" << endl;
            cout << "                                ┆    ≡ 管理员 ≡    ┆ " <<
endl;
            cout   <<   "                                                  ┆
┆ " << endl;
            cout << "                                ┆       1.超级管理员模式
┆ " << endl;
            cout   <<   "                                                  ┆
┆ " << endl;
            cout << "                                ┆       2.管理员模式
┆ " << endl;
            cout   <<   "                                                  ┆
┆ " << endl;
            cout << "                                ┆       3.返回上一步
┆ " << endl;
            cout   <<   "                                                  ┆
┆ " << endl;
            cout << "                    ----------------------------------------
----------------------------------------" << endl;
            cout << "                    请输入想要进行的功能：";
```

```
            int op;
            cin >> op;
            if(op == 1)
            {
                if(check())//如果超级管理员权限获得
                {
                    while(1)//进入超级管理员模式
                    {
                        cout << "                                超级管理员权限获取成
功! \n";
                        cout << "                    ------------------------------
------------------------------------------------" << endl;
                        cout << "                                ┆     ≡ 超级管理员
模式 ≡     ┆ " << endl;
                        cout   <<   "                                                ┆
┆ " << endl;
                        cout << "                                ┆      1.管理员初始
化          ┆ " << endl;
                        cout   <<   "                                                ┆
┆ " << endl;
                        cout << "                                ┆      2.添加管理员
┆ " << endl;
                        cout   <<   "                                                ┆
┆ " << endl;
                        cout << "                                ┆      3.删除管理员
┆ " << endl;
                        cout   <<   "                                                ┆
┆ " << endl;
                        cout << "                                ┆      4.录入管理员
信息        ┆ " << endl;
                        cout   <<   "                                                ┆
┆ " << endl;
                        cout << "                                ┆      5.管理员信息
┆ " << endl;
                        cout   <<   "                                                ┆
┆ " << endl;
                        cout << "                                ┆      6.返回上一步
┆ " << endl;
                        cout   <<   "                                                ┆
┆ " << endl;
                        cout << "                    ------------------------------
------------------------------------------------" << endl;
```

```
                    cout << "                    请输入想要进行的功能：";
                    int op;
                    cin >> op;
                    if(op == 1)
                    {
                        root_reset();//管理员初始化
                        root_save();  //保存管理员信息
                    }
                    else if(op == 2)
                    {
                        root_add();   //管理员添加
                        root_save();//保存管理员信息
                    }
                    else if(op == 3)
                    {
                        root_del();//管理员删除
                        root_save();//保存管理员信息
                    }
                    else if(op == 4)
                    {
                        root_load();//管理员加载信息
                        cout << "                    管理员信息录入成功\n";
                    }
                    else if(op == 5)
                    {
                        root_display();//展示所有的管理员信息
                    }
                    else if(op == 6) break;
                    else printf("                                    输入有误\n");
                }
            }
        }
        else if(op == 2)
        {
            if(root_right())//如果能够获得管理员权限
            {
                while(1)//进入管理员模式
                {
                    cout << "                    ------------------------------
------------------------------------------------" << endl;
                    cout << "                                        ┇     ≡ 管理员模式
≡     ┇ " << endl;
```

```
                        cout   <<   "                                                    ┆
┆ " << endl;
                        cout << "                                  ┆      1.查询顾客信
息      ┆ " << endl;
                        cout   <<   "                                                    ┆
┆ " << endl;
                        cout << "                                  ┆      2.返回上一步
┆ " << endl;
                        cout   <<   "                                                    ┆
┆ " << endl;
                        cout << "                    ------------------------------
------------------------------------------------- " << endl;
                        cout << "                    输入想要进行的功能：";
                        int op;
                        cin >> op;//读入 op 操作

                        if(op == 1)
                        {
                            passenger_display();//顾客信息展示
                        }
                        else if(op == 2)
                        {
                            break;//退出
                        }
                        else
                        {
                            cout << "                    输入错误\n";
                        }
                    }
                }
            }
            else
            {
                break;
            }
        }
    }
    void Airline::root_display()
    {
            administrator *ptr = head_of_root -> next;//管理员信息展示
            cout << setw(25) << "用户名" << setw(12) << "密码" << endl;
            for(ptr; ptr != NULL; ptr = ptr -> next)//节点从前到后遍历
```

```
            {
              cout << setw(25) << ptr -> name << setw(12) << ptr -> password <<
endl;
            }
}
//main.cpp
    /*******************
    * 版权所有 (C)2022, Yuxing liu。
    *
    * 文件名称： main.cpp
    * 文件标识：无
    * 内容摘要：该代码用于完成界面及选择功能
    * 其它说明：无
    * 当前版本： V1.0
    * 作 者：刘宇星
    * 完成日期： 2022-05-10
    *******************/

    #include <iostream>
    #include <windows.h> ///改变字体颜色用
    #include "head.h"
    using namespace std;
    int main()
    {
            Airline airline;
            airline.Init_passenger();//初始用户信息
            //airline.Load_flight();
            airline.Init_flight();//初始化航班
            //airline.Load_passenger();
            airline.root_init();//管理员初始化
            airline.root_load();
            while (1)
            {
              SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
FOREGROUND_INTENSITY | ///字体颜色
                  FOREGROUND_BLUE | FOREGROUND_GREEN);
              cout << "\n";
              cout << "                          ╪╪ 飞 机 订 票 系 统 ╪╪ " <<
endl << endl;
              cout << "                  ╔-----------------------------------
-----------------------------------╗ " << endl;
              cout   <<   "                                                    ┇
```

```
┆ " << endl;
            cout << "                          ┆      ≡ 主菜单 ≡      ┆ " <<
endl;
            cout   <<   "                                                ┆
┆ " << endl;
            cout << "                          ┆    1.录入航班信息    ┆ " <<
endl;
            cout   <<   "                                                ┆
┆ " << endl;
            cout << "                          ┆    2.录入乘客信息    ┆ " <<
endl;
            cout   <<   "                                                ┆
┆ " << endl;
            cout << "                          ┆      3.查询航班      ┆ " <<
endl;
            cout   <<   "                                                ┆
┆ " << endl;
            cout << "                          ┆    4.订票 5.退票     ┆ " <<
endl;
            cout   <<   "                                                ┆
┆ " << endl;
            cout << "                          ┆      6.查询订单      ┆ " <<
endl;
            cout   <<   "                                                ┆
┆ " << endl;
            cout << "                          ┆    7.修改航班信息    ┆ " <<
endl;
            cout   <<   "                                                ┆
┆ " << endl;
            cout << "                          ┆    8.管理员模式      ┆ " <<
endl;
            cout   <<   "                                                ┆
┆ " << endl;
            cout << "                          ┆      0.退出系统      ┆ " <<
endl;
            cout   <<   "                                                ┆
┆ " << endl;
            cout << "                          ┗------------------------------
------------------------------------┛ " << endl;
            cout << "                          请输入想要进行的功能：";
            int i;
            cin >> i;
```

```
            if (i == 1)
            {
                airline.Load_flight();//录入航班信息
                printf("                    恭喜您，航班信息录入成功\n");
            }
            else if(i == 2)
            {
                airline.Load_passenger();//录入乘客信息
                printf("                    恭喜您，乘客信息录入成功\n");
            }
            else if (i == 3)

                airline.Check_flight();//查找航班
            else if (i == 4)
            {
                airline.Book();//定票
                airline.Save_flight();//保存航班信息
                airline.Save_passenger();//保存用户信息
                printf("\n                    航班信息已自动保存\n");
                printf("\n                    用户信息已自动保存\n");
            }
            else if (i == 5)
            {
                airline.Qbook();//退票
                airline.Save_flight();//保存航班信息
                airline.Save_passenger();//保存用户信息
                printf("\n                    航班信息已自动保存\n");
                printf("\n                    用户信息已自动保存\n");
            }
            else if (i == 6)
                airline.Check_book();//查询订单
            else if (i == 7)
            {
                airline.Revise_flight();//修改航班信息
                airline.Save_flight();//保存航班信息
                printf("\n                    航班信息已自动保存\n");
            }
            else if(i == 8)
            {
                airline.root_mode();
             }
            else if (i == 0)
```

```
                break;
            else
            {
                cout << "                无效数字，请重新输入！";
                system("pause");
                system("cls");
            }
        }
        return 0;
}
//Flight.cpp
    /*******************
    * 版权所有 (C)2022, Yuxing Liu。
    *
    * 文件名称： flight.cpp
    * 文件标识：无
    * 内容摘要：完成关于航班类事件的文件
    * 其它说明：无
    * 当前版本： V1.0
    * 作 者：刘宇星
    * 完成日期： 2022-05-10
    *******************/
    #include <iostream>
    #include"head.h"
    #include <cstring>
    #include <iomanip>
    #include <fstream>
    using namespace std;
    /*******************
    * 功能描述：初始化航班
    * 输入参数： 无
    * 输出参数： 无
    * 返回值： void
    * 其它说明：无
    *******************/
    void Init_flight(flightnode *&h)//初始化航班
    {
            h = (flightnode *)malloc(sizeof(flightnode));
            h->next = NULL;
    }
    /*******************
    * 功能描述：载入航班信息
```

```
    * 输入参数： 载入 FlightList.dat 信息到链表
    * 输出参数： 无
    * 返回值： void
    * 其它说明：无
    ********************/
    void Load_flight(flightnode *&h)//载入航班
    {
            flightnode *f = h;
            flightnode *s;
            s = (flightnode *)malloc(sizeof(flightnode));
            ifstream infile("FlightList.dat", ios::in);
            if (!infile)
            {
              cerr << "                    信息错误。";
              return;
            }
            while (1)
            {
              infile  >>  s->flight_num  >>  s->time  >>  s->start_place  >>  s-
>end_place >> s->price >> s->left;
              if (!infile.eof())
              {
                  f->next = s;
                  f = f->next;
                  s = (flightnode *)malloc(sizeof(flightnode));
              }
              else
                  break;
            }
            f->next = NULL;
            free(s);
    }
    /********************
    * 功能描述：添加航班
    * 输入参数： 航班的相关信息
    * 输出参数： 添加成功或者选择继续添加
    * 返回值： void
    * 其它说明：无
    ********************/
    void Add_flight(flightnode *&h)//添加航班
    {
            char mark = 'y';
```

```
        flightnode *s, *r;
        r = h;
        for (; r->next != NULL; r = r->next) {}
        while (mark == 'y')
        {
          s = (flightnode *)malloc(sizeof(flightnode));
          cout << "                    请输入航班号：";
          cin >> s->flight_num;
          cout << "                    请输入起飞时间：";
          cin >> s->time;
          cout << "                    请输入出发地:";
          cin >> s->start_place;
          cout << "                    请输入目的地：";
          cin >> s->end_place;
          cout << "                    请输入价格：";
          cin >> s->price;
          cout << "                    请输入余票：";
          cin >> s->left;

          r->next = s;
          r = s;
          cout << "                    添加成功！" << endl;
          cout << "                    是否继续添加？（是“y”/否“n”）：";
          cin >> mark;
        }
        r->next = NULL;
    }
    /*******************
    * 功能描述：查找航班
    * 输入参数： 选择查找方式
    * 输出参数： 进入相关函数进行查找功能
    * 返回值： void
    * 其它说明：无
    *******************/
    void Check_flight(flightnode *&h)//查找航班
    {
        system("cls");
        int i;
        cout << "                    ----------------------------------------
---------------------------------- " << endl;
        cout << "                                ┆    ≡ 查找方式 ≡    ┆ " <<
endl;
```

```
            cout << "                    ┆                                    ┆
" << endl;
            cout << "                              ┆      1.航班查找      ┆ " <<
endl;
            cout << "                    ┆                                    ┆
" << endl;
            cout << "                              ┆     2.起始地查找     ┆ " <<
endl;
            cout << "                    ┆                                    ┆
" << endl;
            cout << "                              ┆   3.浏览全部航班     ┆ " <<
endl;
            cout << "                    ┆                                    ┆
" << endl;
            cout << "                     ----------------------------------------
----------------------------------- " << endl;
            cout << "                    请输入查找方式：";
            cin >> i;
            if (i == 1)
              Check_flightnum(h);
            else if (i == 2)
              Check_seplace(h);
            else if (i == 3)
              Check_all(h);
    }
    /********************
    * 功能描述：航班号查找
    * 输入参数： 航班号
    * 输出参数： 航班的相关信息
    * 返回值： void
    * 其它说明：无
    ********************/
    void Check_flightnum(flightnode *&h)//航班号查找
    {

            char flightnum[10];
            flightnode *p = h;
            cout << setw(32) << "请输入航班号：";
            cin >> flightnum;
            cout << setw(25) << "航班号" << setw(15) << "起飞日期" << setw(15) <<
"出发地" << setw(15) << "目的地" << setw(15) << "价格" << setw(15) << "余票" <<
endl;
```

```
            while (p != NULL)
            {
              if (strcmp(p->flight_num, flightnum) == 0)
              {
                  cout << setw(25) << p->flight_num << setw(15) << p->time <<
setw(15) << p->start_place << setw(15) << p->end_place << setw(15) << p->price <<
setw(15) << p->left << endl;
                  cout << "                    ";
                  system("pause");
                  return;
              }
              else
                  p = p->next;
            }
            cout << setw(32) << "未查到任何信息。";
            system("pause");
    }
    /*******************
    * 功能描述：起始地查找
    * 输入参数： 起始地
    * 输出参数： 航班的相关信息
    * 返回值： void
    * 其它说明：无
    *******************/
    void Check_seplace(flightnode *&h)//起始地查找
    {
            char start_place[20], end_place[20];
            cout << "                    请输入出发地：";
            cin >> start_place;
            cout << "                    请输入目的地：";
            cin >> end_place;
            flightnode *p = h->next;
            cout << setw(25) << "航班号" << setw(15) << "起飞日期" << setw(15) <<
"出发地" << setw(15) << "目的地" << setw(15) << "价格" << setw(15) << "余票" <<
endl;
            while (p != NULL)
            {
              if (strcmp(p->start_place, start_place) == 0 && strcmp(p->end_place,
end_place) == 0)
              {
                  cout << setw(25) << p->flight_num << setw(15) << p->time <<
setw(15) << p->start_place << setw(15) << p->end_place << setw(15) << p->price <<
```

```
setw(15) << p->left << endl;
                cout << setw(32);
                system("pause");
                return;
              }
              p = p->next;
            }
            cout << setw(32) << "未查到任何信息";
            system("pause");
    }
    /*******************
    * 功能描述：浏览全部航班
    * 输入参数：
    * 输出参数： 航班的相关信息
    * 返回值： void
    * 其它说明：无
    *******************/
    void Check_all(flightnode *&h)//浏览全部航班
    {

            flightnode *p = h->next;
            cout << setw(25) << "航班号" << setw(15) << "起飞日期" << setw(15) <<
"出发地" << setw(15) << "目的地" << setw(15) << "价格" << setw(15) << "余票" <<
endl;
            while (p != NULL)
            {
              cout << setw(25) << p->flight_num << setw(15) << p->time << setw(15)
<< p->start_place << setw(15) << p->end_place << setw(15) << p->price << setw(15)
<< p->left << endl;
              p = p->next;
            }
    }
    /*******************
    * 功能描述：修改航班信息
    * 输入参数：选择修改方式
    * 输出参数： 调用相关函数进行实现
    * 返回值： void
    * 其它说明：无
    *******************/
    void Revise_flight(flightnode *&h)//修改航班信息
    {
            flightnode *f = h->next;
```

```
            int i;
            cout << "                    ------------------------------------------------
------------------------------------------ " << endl;
            cout << "                               ┇      ≡ 修改方式 ≡      ┇ " <<
endl;
            cout << "                  ┇                                              ┇
" << endl;
            cout << "                               ┇       1.删除航班           ┇ " <<
endl;
            cout << "                  ┇                                              ┇
" << endl;
            cout << "                               ┇      2.修改起飞时间        ┇ " <<
endl;
            cout << "                  ┇                                              ┇
" << endl;
            cout << "                               ┇      3.修改航班价格        ┇ " <<
endl;
            cout << "                  ┇                                              ┇
" << endl;
            cout << "                               ┇         4.返回             ┇ " <<
endl;
            cout << "                  ┇                                              ┇
" << endl;
            cout << "                    ------------------------------------------------
------------------------------------------ " << endl;
            cout << "                  请输入查找方式：";
            cin >> i;
            if (i == 1)
              Delete_flight(h);
            else if (i == 2)
              Revise_time(h);
            else if (i == 3)
              Revise_price(h);
            else if (i == 4)
              return;
    }
    /*******************
    * 功能描述：删除航班
    * 输入参数：航班号
    * 输出参数： 删除成功
    * 返回值： void
    * 其它说明：无
```

```
*******************/
void Delete_flight(flightnode *&h)//删除航班
{
        flightnode *p = h->next, *q = h;
        char flightnum[10];
        cout << "                    请输入您想要修改的航班号：";
        cin >> flightnum;
        while (p != NULL)
        {
          if (strcmp(p->flight_num, flightnum) == 0)
          {
              q->next = p->next;
              free(p);
              cout << "                    删除成功！";
              return;
          }
          q = q->next;
          p = q->next;

        }
}
/*******************
* 功能描述：修改起飞抵达时间
* 输入参数：航班号，新的航班号
* 输出参数： 修改成功
* 返回值： void
* 其它说明：无
*******************/
void Revise_time(flightnode *&h)//修改起飞抵达时间
{
        char flightnum[10];
        flightnode *p = h->next;
        cout << "                    请输入您想要修改的航班号：";
        cin >> flightnum;
        while (p != NULL)
        {
          if (strcmp(p->flight_num, flightnum) == 0)
          {
              cout << "          请输入新的起飞时间：";
              cin >> p->time;
              cout << "          修改成功！";
              return;
```

```
            }
        }
        cout << "                    没有您想要修改的航班号!";
        system("pause");
}
/*******************
* 功能描述：修改价格
* 输入参数：航班号，新的价格
* 输出参数： 修改成功
* 返回值： void
* 其它说明：无
*******************/
void Revise_price(flightnode *&h)//修改价格
{
        char flightnum[10];
        flightnode *p = h->next;
        cout << "                    请输入您想要修改的航班号：";
        cin >> flightnum;
        while (p != NULL)
        {
          if (strcmp(p->flight_num, flightnum) == 0)
          {
              cout << "          请输入新的价格：";
              cin >> p->price;
              cout << "          修改成功!    ";
              return;
          }
        }
        cout << "                    没有您想要修改的航班号!";
        system("pause");
}
/*******************
* 功能描述：保存航班信息
* 输入参数：无
* 输出参数：用户的信息到文件 FlightList.dat
* 返回值： void
* 其它说明：
*******************/
void Save_flight(flightnode *&h)//保存航班信息
{
        flightnode *f = h->next;
        ofstream outfile("FlightList.dat", ios::trunc);
```

```
            if (!outfile)
            {
              cerr << "                    存储失败！";
              return;
            }
            while (f != NULL)
            {
              outfile << f->flight_num << " " << f->time << " " << f->start_place
<< " " << f->end_place << " " << f->price << " " << f->left << endl;
              f = f->next;
            }
            outfile.close();
}
//passenger.cpp
    /*********************
    * 版权所有 (C)2022, Yuxing Liu。
    *
    * 文件名称： passenger.cpp
    * 文件标识：无
    * 内容摘要：完成关于用户类事件的文件
    * 其它说明：无
    * 当前版本： V1.0
    * 作 者：刘宇星
    * 完成日期： 2022-05-10
    *********************/
    #include <iostream>
    #include "head.h"
    #include <iomanip>
    #include <string>
    #include <fstream>
    using namespace std;
    /*********************
    * 功能描述：初始化用户
    * 输入参数： 无
    * 输出参数： 无
    * 返回值： void
    * 其它说明：无
    *********************/
    void Init_passenger(passengernode *&c)//初始化用户
    {
            c = (passengernode *)malloc(sizeof(passengernode));
            c->next = NULL;
```

```
    }
    /*******************
    * 功能描述：载入用户信息
    * 输入参数： 载入 PassengerList.dat 信息到链表
    * 输出参数： 无
    * 返回值： void
    * 其它说明：无
    *******************/
    void Load_passenger(passengernode *&c)//载入用户信息
    {
            passengernode *p = c;
            passengernode *s;
            s = (passengernode *)malloc(sizeof(passengernode));
            ifstream infile("PassengerList.dat", ios::in);
            if (!infile)
            {
              cerr << "                    信息错误。";
              return;
            }
            while (1)
            {
              infile >> s->name >> s->ID_num >> s->flight_num >> s->time >> s-
>start_place >> s->end_place >> s->price;
              if (!infile.eof())
              {
                  p->next = s;
                  p = p->next;
                  s = (passengernode *)malloc(sizeof(passengernode));
              }
              else
                  break;
            }
            p->next = NULL;
            free(s);
    }
    /*******************
    * 功能描述：航班信息
    * 输入参数：passenger 的相关信息
                flight 的相关信息
    * 输出参数： 购票成功 or 失败
    * 返回值： void
    * 其它说明：如果余票不足还可以预定，如果有多余的票即可瞬间购买
```

```
    ********************/
    void Book(flightnode *&h, passengernode *&c)//订票
    {
            char start_place[20], end_place[20], flightnum[10];//航班信息
            flightnode *p = h->next, *q = h->next;
            passengernode *s, *r = c;
            char mark, check = '1';
            for (; r->next != NULL; r = r->next) {}
            s = (passengernode *)malloc(sizeof(passengernode));
            cout << "                    请输入您的姓名：";
            cin >> s->name;
            cout << "                    请输入您的证件号：";
            cin >> s->ID_num;

            cout << "                    请输入出发地：";
            cin >> start_place;
            cout << "                    请输入目的地：";
            cin >> end_place;
            cout << setw(25) << "航班号" << setw(15) << "起飞日期" << setw(15) <<
"出发地" << setw(15) << "目的地" << setw(15) << "价格" << setw(15) << "余票" <<
endl;
            while (p != NULL)
            {
              if (strcmp(p->start_place, start_place) == 0 && strcmp(p->end_place,
end_place) == 0)
              {
                  cout << setw(25) << p->flight_num << setw(15) << p->time <<
setw(15) << p->start_place << setw(15) << p->end_place << setw(15) << p->price <<
setw(15) << p->left << endl;
                  check = '0';
              }
              p = p->next;
            }
            if (check == '1')
            {
              cout << "                      没想相关的航班，";
              system("pause");
              return;
            }
            cout << "                    请输入您想要购买的航班号：";
            cin >> flightnum;
            while (q != NULL)
```

```
            {
              if (strcmp(q->flight_num, flightnum) == 0 && strcmp(q->start_place,
start_place) == 0 && strcmp(q->end_place, end_place) == 0)
              {
                  if (q->left == 0)
                  {
                      cout << "                    余票不足，请选择是否预约（是“y”/
否“n”）";
                      cin >> mark;
                      if (mark == 'y')
                      {
                          s->full = 1;
                          strcpy(s->flight_num, flightnum);
                          cout << "                        预约成功！";
                          r->next = s;
                          r = s;
                          r->next = NULL;
                          system("pause");
                          return;
                      }
                      else
                          return;
                  }
                  strcpy(s->flight_num, flightnum);
                  strcpy(s->start_place, start_place);
                  strcpy(s->end_place, end_place);
                  strcpy(s->time, q->time);
                  s->price = q->price;
                  r->next = s;
                  r = s;
                  r->next = NULL;
                  //购票成功
                  cout << "                    购票成功！" << endl;
                  q->left--;
                  cout << "                    ";
                  system("pause");
                  return;
              }
              else
                  q = q->next;
            }
            cout << "                    航班号填入错误！" << endl;
```

```
            cout << "                    ";
            system("pause");
    }
    /*******************
    * 功能描述：退票
    * 输入参数：passenger 的相关信息
    * 输出参数： 退票成功 or 失败
    * 返回值： void
    * 其它说明：
    ********************/
    void Qbook(flightnode *&h, passengernode *&c)//退票
    {
            char name[30];
            char ID_num[30];
            flightnode *f = h->next;
            passengernode *p = c->next, *q = c, *t = c->next;
            cout << "                    请输入您的姓名：";
            cin >> name;
            cout << "                    请输入您的证件号：";
            cin >> ID_num;
            while (p != NULL)
            {
              if (strcmp(p->name, name) == 0 && strcmp(p->ID_num, ID_num) == 0)
              {
                  while (f != NULL)
                  {
                      if (strcmp(p->flight_num, f->flight_num) == 0)
                      {
                          f->left++;
                          while (t != NULL)
                          {
                              if (strcmp(t->flight_num, p->flight_num) == 0 && t->full == 1)
                              {
                                  strcpy(t->start_place, p->start_place);
                                  strcpy(t->end_place, p->end_place);
                                  strcpy(t->time, p->time);
                                  t->price = p->price;
                                  f->left--;
                                  break;
                              }
                              t = t->next;
```

```
                        }
                        break;
                    }
                    f = f->next;
                }
                q->next = p->next;
                free(p);
                cout << "                    退票成功！";
                return;
            }
            q = q->next;
            p = q->next;
          }


    }
    /*******************
    * 功能描述：查询订票
    * 输入参数：passenger 的相关信息
    * 输出参数： 所查用户订票的信息
    * 返回值： void
    * 其它说明：
    *******************/
    void Check_book(passengernode *&c)//查询订票
    {
            char name[30];
            char ID_num[30];
            passengernode *p = c->next;
            cout << "                    请输入您的姓名：";
            cin >> name;
            cout << "                    请输入您的证件号：";
            cin >> ID_num;
            cout << setw(25) << "姓名" << setw(15) << "航班号" << setw(15) << "起飞日期" << setw(15) << "出发地" << setw(15) << "目的地" << setw(15) << "价格" << setw(15) << endl;
            while (p != NULL)
            {
              if (strcmp(p->name, name) == 0 && strcmp(p->ID_num, ID_num) == 0)
              {
                  cout << setw(25) << p->name << setw(15) << p->flight_num << setw(15) << p->time << setw(15) << p->start_place << setw(15) << p->end_place << setw(15) << p->price << setw(15) << endl;
```

```
                cout << "                    ";
                system("pause");
                return;
            }
            else
                p = p->next;
        }
        cout << setw(32) << "未查到任何信息。";
        system("pause");
    }
    /*******************
    * 功能描述：保存用户信息
    * 输入参数：无
    * 输出参数：用户的信息到文件 PassengerList.dat
    * 返回值： void
    * 其它说明：
    *******************/
    void Save_passenger(passengernode *&c)//保存用户信息
    {
        passengernode *p = c->next;
        ofstream outfile("PassengerList.dat", ios::trunc);
        if (!outfile)
        {
            cerr << "                存储失败！";
            return;
        }
        while (p != NULL)
        {
            outfile << p->name << " " << p->ID_num << " " << p->flight_num << "
" << p->time << " " << p->start_place << " " << p->end_place << " " << p->price <<
endl;
            p = p->next;
        }
        outfile.close();
}
//airlogindialog.h
#ifndef AIRLOGINDIALOG_H
#define AIRLOGINDIALOG_H

#include <QDialog>
#include <QSqlTableModel>
namespace Ui {
```

```
class AirLoginDialog;
}

class AirLoginDialog : public QDialog
{
    Q_OBJECT

public:
    explicit AirLoginDialog(QWidget *parent = nullptr);
    ~AirLoginDialog();

private slots:
    void on_pushButton_clicked();

    void on_pushButton_2_clicked();

    void on_pushButton_3_clicked();

    void on_pushButton_4_clicked();

    void on_pushButton_5_clicked();

    void on_pushButton_6_clicked();

    void on_pushButton_7_clicked();

    void on_pushButton_8_clicked();



    void on_pushButton_9_clicked();

private:
    Ui::AirLoginDialog *ui;
    QSqlTableModel *model;
};

#endif // AIRLOGINDIALOG_H
```

```
//connection.h
#ifndef CONNECTION_H
#define CONNECTION_H

#include <QSqlDatabase>
#include <QSqlQuery>
static bool createConnection()
{
    QSqlDatabase db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName("ahu.db");
    if(!db.open()) return false;
    QSqlQuery query;
    query.exec("create table student (id int primary key, name vchar)");
    query.exec("insert into student values (0,'刘明')");
    query.exec("insert into student values (1,'陈刚')");
    query.exec("insert into student values (2,'王红')");
    return true;
}

#endif // CONNECTION_H
//dlgitem.h
#ifndef DLGITEM_H
#define DLGITEM_H

#include <QDialog>

namespace Ui {
class DlgItem;
}
typedef struct _Customer{
    QString name;
    int type;
    int area;
    int province;
    QString city;
    QString address;
    QString site;
    QString contact;
    QString buy;
    QString research;
    QString remark;
}Customer,*PCustomer;
```

```
class DlgItem : public QDialog
{
    Q_OBJECT

public:
    explicit DlgItem(Customer* item,QWidget *parent = 0);
    ~DlgItem();

signals:
    void sigAddItem(Customer*);

private slots:
    void on_btnAdd_clicked();
    void on_btnCancel_clicked();
    void on_combo2_currentTextChanged(const QString &arg1);
    void on_edit1_textChanged(const QString &arg1);

private:
    int String2Province(QString s);
    QString Province2String(int province);

private:
    Ui::DlgItem *ui;
    Customer* mItem;
};

#endif // DLGITEM_H
//logindialog.h
#ifndef LOGINDIALOG_H
#define LOGINDIALOG_H
#include <QDialog>

namespace Ui {
class LoginDialog;
}

class LoginDialog : public QDialog
{
    Q_OBJECT

public:
```

```
    explicit LoginDialog(QWidget *parent = nullptr);
    ~LoginDialog();

private slots:
    void on_loginBtn_clicked();


    void on_exitBtn_clicked();

    void on_usrLineEdit_cursorPositionChanged(int arg1, int arg2);

    void on_pwdLineEdit_cursorPositionChanged(int arg1, int arg2);

private:
    Ui::LoginDialog *ui;
};

#endif // LOGINDIALOG_H
//mainwindow.h
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QMainWindow>
#include <QSortFilterProxyModel>
#include <QStandardItemModel>
#include <QTreeWidget>
#include <QVector>
#include "dlgitem.h"

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

    void InitMenu();
```

```
    void InitModel();
    void InitTable();
    void AddItem(Customer* item);
    void ClearData();

    void OpenDb(QString fileName);
    void SaveToDb(QString fileName);
    void InsertToDb(Customer* item);
    void DeleteFromDb(QString name);

private slots:
    void onMenuOpen(bool checked);
    void onMenuSave(bool checked);

    void on_btnSearch_clicked();        //搜索按钮，关键字过滤
    void on_btnAdd_clicked();           //添加
    void on_btnDelete_clicked();        //删除

    void slotAddItem(Customer* item);   //DlgItem 里添加记录后需要不关闭对话框，因
此发个信号
    void on_comboBox_currentTextChanged(const QString &arg1);   //按单位类别过滤
    void on_treeWidget_currentItemChanged(QTreeWidgetItem *current,
QTreeWidgetItem *previous); //改变选择区域
    void on_tableView_doubleClicked(const QModelIndex &index);  //数据一行双击修改

    void on_pushButton_clicked();

private:
    QString Type2String(int type);
    QString Area2String(int area);
    QString Province2String(int province);
    Customer* FindItem(QString name);
    void DeleteItem(QString name);

private:
    Ui::MainWindow *ui;
    QString mDefaultDB;          //db 文件路径
    QStandardItemModel* mModel; //tableview 的 model
    QVector<Customer*> mItems;  //所有客户数据

    QStringList mAllType;        //所有单位类型
    QStringList mAllArea;        //所有区域
```

```
    QStringList mAllProvince;    //所有省份

    QString mFilterType;         //单位类型过滤
    QString mFilterArea;         //地域过滤
    QString mFilterProvince;     //省份过滤
    QString mFilterKey;          //关键字过滤
};

#endif // MAINWINDOW_H
//useritem.h
#ifndef USERITEM_H
#define USERITEM_H
#include <QMainWindow>
#include <QSortFilterProxyModel>
#include <QStandardItemModel>
#include <QTreeWidget>
#include <QVector>
#include "userticket.h"


namespace Ui {
class useritem;
}

class useritem : public QMainWindow
{
    Q_OBJECT

public:
    explicit useritem(QWidget *parent = 0);
    ~useritem();

    void InitMenu();

    void InitModel();
    void InitTable();
    void AddItem(Customer1* item);
    void ClearData();

    void OpenDb(QString fileName);
    void SaveToDb(QString fileName);
    void InsertToDb(Customer1* item);
```

```
    void DeleteFromDb(QString name);

private slots:
    void onMenuOpen(bool checked);
    void onMenuSave(bool checked);

    void on_btnSearch_clicked();        //搜索按钮，关键字过滤
    void on_btnAdd_clicked();           //添加
    void on_btnDelete_clicked();        //删除

    void slotAddItem(Customer1* item);   //userticket 里添加记录后需要不关闭对话
框，因此发个信号
    void on_comboBox_currentTextChanged(const QString &arg1);   //按单位类别过滤
    void on_treeWidget_currentItemChanged(QTreeWidgetItem *current,
QTreeWidgetItem *previous); //改变选择区域
    void on_tableView_doubleClicked(const QModelIndex &index);  //数据一行双击修改

    void on_pushButton_clicked();

private:
    QString Type2String(int type);
    QString Area2String(int area);
    QString Province2String(int province);
    Customer1* FindItem(QString name);
    void DeleteItem(QString name);

private:
    Ui::useritem *ui;
    QString mDefaultDB;          //db 文件路径
    QStandardItemModel* mModel; //tableview 的 model
    QVector<Customer1*> mItems;  //所有客户数据

    QStringList mAllType;        //所有单位类型
    QStringList mAllArea;        //所有区域
    QStringList mAllProvince;    //所有省份

    QString mFilterType;         //单位类型过滤
    QString mFilterArea;         //地域过滤
    QString mFilterProvince;     //省份过滤
    QString mFilterKey;          //关键字过滤
};
```

```
#endif // useritem_H

//userticket.h
#ifndef USERTICKET_H
#define USERTICKET_H
#include <QDialog>


namespace Ui {
class userticket;
}

typedef struct Customer_{
    QString name;
    int type;
    int area;
    int province;
    QString city;
    QString address;
    QString site;
    QString contact;
    QString buy;
    QString research;
    QString remark;
}Customer1,*PCustomer1;

class userticket : public QDialog
{
    Q_OBJECT

public:
    explicit userticket(Customer1* item,QWidget *parent = 0);
    ~userticket();

signals:
    void sigAddItem(Customer1*);

private slots:
    void on_btnAdd_clicked();
    void on_btnCancel_clicked();
    void on_combo2_currentTextChanged(const QString &arg1);
    void on_edit1_textChanged(const QString &arg1);
```

```
private:
    int String2Province(QString s);
    QString Province2String(int province);

private:
    Ui::userticket *ui;
    Customer1* mItem;
};

#endif // userticket_H
//airlogindialog.cpp
#include "airlogindialog.h"
#include "ui_airlogindialog.h"
#include <QMessageBox>
#include <QSqlError>
#include "logindialog.h"
AirLoginDialog::AirLoginDialog(QWidget *parent) :
    QDialog (parent),
    ui(new Ui::AirLoginDialog)
{
    ui->setupUi(this);
    model = new QSqlTableModel(this);
    model->setTable("student");
    model->setEditStrategy(QSqlTableModel::OnManualSubmit);
    model->select(); //选取整个表的所有行

    //不显示 name 属性列,如果这时添加记录，则该属性的值添加不上
    // model->removeColumn(1);

    ui->tableView->setModel(model);

    //使其不可编辑
    // ui->tableView->setEditTriggers(QAbstractItemView::NoEditTriggers);
}

AirLoginDialog::~AirLoginDialog()
{
    delete ui;
}

void AirLoginDialog::on_pushButton_clicked()
```

```
{
    model->database().transaction(); //开始事务操作
    if (model->submitAll()) {
        model->database().commit(); //提交
    } else {
        model->database().rollback(); //回滚
        QMessageBox::warning(this, tr("tableModel"),
                             tr("数据库错误: %1")
                             .arg(model->lastError().text()));
    }
}


void AirLoginDialog::on_pushButton_2_clicked()
{
    model->revertAll();
}

void AirLoginDialog::on_pushButton_5_clicked()
{
    QString name = ui->lineEdit->text();
    //根据姓名进行筛选
    model->setFilter(QString("name = '%1'").arg(name));
    //显示结果
    model->select();
}

void AirLoginDialog::on_pushButton_6_clicked()
{
    model->setTable("student");   //重新关联表
    model->select();   //这样才能再次显示整个表的内容
}

// 升序
void AirLoginDialog::on_pushButton_7_clicked()
{
    //id 属性即第 0 列，升序排列
    model->setSort(0, Qt::AscendingOrder);
    model->select();
}
// 降序
void AirLoginDialog::on_pushButton_8_clicked()
```

```
{
    model->setSort(0, Qt::DescendingOrder);
    model->select();
}

void AirLoginDialog::on_pushButton_4_clicked()
{
    //获取选中的行
    int curRow = ui->tableView->currentIndex().row();

    //删除该行
    model->removeRow(curRow);

    int ok = QMessageBox::warning(this,tr("删除当前行!"),
                                  tr("你确定删除当前行吗？"),
                                  QMessageBox::Yes,QMessageBox::No);
    if(ok == QMessageBox::No)
    {
        model->revertAll(); //如果不删除，则撤销
    }
    else model->submitAll(); //否则提交，在数据库中删除该行
}

void AirLoginDialog::on_pushButton_3_clicked()
{
    int rowNum = model->rowCount(); //获得表的行数
    int id = 10;
    model->insertRow(rowNum); //添加一行
    model->setData(model->index(rowNum,0),id);
    //model->submitAll(); //可以直接提交
}

void AirLoginDialog::on_pushButton_9_clicked()
{
    LoginDialog *dlg = new LoginDialog;
    dlg->show();
    this->close();
}
//dlgitem.cpp
#include "dlgitem.h"
#include "ui_dlgitem.h"
DlgItem::DlgItem(Customer* item,QWidget *parent) :
```

```
    QDialog(parent),
    ui(new Ui::DlgItem)
{
    ui->setupUi(this);

    mItem = item;

    QFont font( "Microsoft YaHei", 10, 75);
    ui->labelTip->setFont(font);
    ui->labelTip->setStyleSheet("QLabel{color:red;}");
    ui->tableWidget->setRowCount(20);
    for(int i=0;i<20;i++)
    {
        ui->tableWidget->setItem(i,0,new QTableWidgetItem(""));
        ui->tableWidget->setItem(i,1,new QTableWidgetItem(""));
        ui->tableWidget->setItem(i,2,new QTableWidgetItem(""));
        ui->tableWidget->setItem(i,3,new QTableWidgetItem(""));
        ui->tableWidget->setItem(i,4,new QTableWidgetItem(""));
        ui->tableWidget->setItem(i,5,new QTableWidgetItem(""));
    }

    on_combo2_currentTextChanged("华东");
    if(mItem)
    {
        //修改
        ui->btnAdd->setText("修改");

        ui->edit1->setText(mItem->name);
        ui->combo1->setCurrentIndex(mItem->type);
        ui->combo2->setCurrentIndex(mItem->area);
        on_combo2_currentTextChanged(ui->combo2->currentText());
        ui->combo3->setCurrentText(Province2String(mItem->province));
        ui->edit2->setText(mItem->city);
        ui->edit3->setText(mItem->address);
        ui->edit4->setText(mItem->site);
        ui->textEdit->setText(mItem->buy);
        ui->textEdit2->setText(mItem->research);
        ui->textEdit3->setText(mItem->remark);

        QStringList all = mItem->contact.split(";");
        for(int i=0;i<all.size();i++)
        {
```

```
            QStringList row = all[i].split(",");
            if(row.size() >= 6)
            {
                bool bRed = !row[5].isEmpty();
                ui->tableWidget->item(i,0)->setText(row[0]);
                ui->tableWidget->item(i,1)->setText(row[1]);
                ui->tableWidget->item(i,2)->setText(row[2]);
                ui->tableWidget->item(i,3)->setText(row[3]);
                ui->tableWidget->item(i,4)->setText(row[4]);
                ui->tableWidget->item(i,5)->setText(row[5]);
                if(!row[5].isEmpty())
                {
                    ui->tableWidget->item(i,0)->setBackground(Qt::red);
                    ui->tableWidget->item(i,1)->setBackground(Qt::red);
                    ui->tableWidget->item(i,2)->setBackground(Qt::red);
                    ui->tableWidget->item(i,3)->setBackground(Qt::red);
                    ui->tableWidget->item(i,4)->setBackground(Qt::red);
                    ui->tableWidget->item(i,5)->setBackground(Qt::red);
                }
            }
        }
    }
}

DlgItem::~DlgItem()
{
    delete ui;
}

void DlgItem::on_btnAdd_clicked()
{
    if(ui->edit1->text().isEmpty())
    {
        ui->labelTip->setText("航班号名称不能为空!");
        return;
    }
    Customer* p = new Customer;
    p->name = ui->edit1->text();
    p->type = ui->combo1->currentIndex();
    p->area = ui->combo2->currentIndex();
    p->province = String2Province(ui->combo3->currentText());
    p->city = ui->edit2->text();
```

```
    p->address = ui->edit3->text();
    p->site = ui->edit4->text();
    p->buy = ui->textEdit->toPlainText();
    p->research = ui->textEdit2->toPlainText();
    p->remark = ui->textEdit3->toPlainText();

    QString sAll = "";
    QString s = "";
    int columns = ui->tableWidget->columnCount();
    for(int i=0;i<ui->tableWidget->rowCount();i++)
    {
        if(ui->tableWidget->item(i,0)->text().isEmpty())
        {
            continue;
        }
        for(int j=0;j<columns;j++)
        {
            //,和;是用来分隔的标记，禁止用户输入这 2 个符号，如果不小心输了，程序也
替换成空格
            s = ui->tableWidget->item(i,j)->text();
            s.replace(",","，");
            s.replace(";","；");
            sAll += s;
            if(j == columns-1)
            {
                sAll += ";";
            }
            else
            {
                sAll += ",";
            }
        }
    }
    p->contact = sAll;

    if(mItem)
    {
        //修改
       mItem->name = p->name;
        mItem->type = p->type;
        mItem->area = p->area;
        mItem->province = p->province;
```

```
        mItem->city = p->city;
        mItem->address = p->address;
        mItem->site = p->site;
        mItem->contact = p->contact;
        mItem->buy = p->buy;
        mItem->research = p->research;
        mItem->remark = p->remark;
        accept();
    }
    else
    {
        //新增
        emit sigAddItem(p);

        ui->edit1->setText("");
        ui->edit2->setText("");
        ui->edit3->setText("");
        ui->edit4->setText("");
        ui->textEdit->setText("");
        ui->textEdit2->setText("");
        ui->textEdit3->setText("");
        for(int i=0;i<20;i++)
        {
            ui->tableWidget->item(i,0)->setText("");
            ui->tableWidget->item(i,1)->setText("");
            ui->tableWidget->item(i,2)->setText("");
            ui->tableWidget->item(i,3)->setText("");
            ui->tableWidget->item(i,4)->setText("");
            ui->tableWidget->item(i,5)->setText("");
        }
        ui->edit1->setFocus();
        ui->labelTip->setText("增加成功!");
    }

}

void DlgItem::on_btnCancel_clicked()
{
    reject();
}

int DlgItem::String2Province(QString s)
```

```
{
    QStringList AllProvince;
    AllProvince<<"山东"<<"浙江"<<"江苏"<<"安徽"<<"上海"<<"福建"
               <<"广东"<<"广西"<<"海南"
               <<"湖北"<<"湖南"<<"河南"<<"江西"
               <<"北京"<<"天津"<<"河北"<<"山西"<<"内蒙"
               <<"宁夏"<<"新疆"<<"青海"<<"陕西"<<"甘肃"<<"四川"<<"云南"<<"贵州
"<<"西藏"<<"重庆"<<"辽宁"<<"吉林"<<"黑龙江";
    return AllProvince.indexOf(s);
}

QString DlgItem::Province2String(int n)
{
    QStringList AllProvince;
    AllProvince<<"山东"<<"浙江"<<"江苏"<<"安徽"<<"上海"<<"福建"
               <<"广东"<<"广西"<<"海南"
               <<"湖北"<<"湖南"<<"河南"<<"江西"
               <<"北京"<<"天津"<<"河北"<<"山西"<<"内蒙"
               <<"宁夏"<<"新疆"<<"青海"<<"陕西"<<"甘肃"<<"四川"<<"云南"<<"贵州
"<<"西藏"<<"重庆"<<"辽宁"<<"吉林"<<"黑龙江";
    if(n>=0 && n<AllProvince.size())
    {
        return AllProvince[n];
    }
    return "";
}

void DlgItem::on_combo2_currentTextChanged(const QString &text)
{
    ui->combo3->clear();
    if(text == "华东")
    {
        ui->combo3->addItems(QStringList()<<"山东"<<"浙江"<<"江苏"<<"安徽"<<"上海
"<<"福建");
    }
    else if(text == "华南")
    {
        ui->combo3->addItems(QStringList()<<"广东"<<"广西"<<"海南");
    }
    else if(text == "华中")
    {
        ui->combo3->addItems(QStringList()<<"湖北"<<"湖南"<<"河南"<<"江西");
```

```
    }
    else if(text == "华北")
    {
        ui->combo3->addItems(QStringList()<<"北京"<<"天津"<<"河北"<<"山西"<<"内蒙
");
    }
    else if(text == "其他")
    {
        ui->combo3->addItems(QStringList()<<"宁夏"<<"新疆"<<"青海"<<"陕西"<<"甘肃
"<<"四川"<<"云南"<<"贵州"<<"西藏"<<"重庆"<<"辽宁"<<"吉林"<<"黑龙江");
    }
}

void DlgItem::on_edit1_textChanged(const QString &arg1)
{
    ui->labelTip->setText("");
}
//logindialog.cpp
#include "logindialog.h"
#include "ui_logindialog.h"
#include <QMessageBox>
int yonghu=0;

LoginDialog::LoginDialog(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::LoginDialog)
{
    ui->setupUi(this);
}

LoginDialog::~LoginDialog()
{
    delete ui;
}

void LoginDialog::on_loginBtn_clicked()
{
    // 判断用户名和密码是否正确,
    // 如果错误则弹出警告对话框
    if((ui->usrLineEdit->text() == tr("Yuxing_Liu") &&
        ui->pwdLineEdit->text() == tr("372321"))    ||
            (ui->usrLineEdit->text() == tr("Yuxing") &&
```

```
            ui->pwdLineEdit->text() == tr("372321"))   )
    {
        if(ui->usrLineEdit->text() == tr("Yuxing_Liu") &&
            ui->pwdLineEdit->text() == tr("372321"))
        {
                yonghu=1;
                accept();
        }
        if(ui->usrLineEdit->text() == tr("Yuxing") &&
            ui->pwdLineEdit->text() == tr("372321"))
        {
                yonghu=2;
                accept();
        }
    } else {
       QMessageBox::warning(this, QObject::tr("警告!"),
                  QObject::tr("用户名或密码错误!"),
                  QMessageBox::Yes);
    }
    //清除内容 并定位光标
    ui->usrLineEdit->clear();
    ui->pwdLineEdit->clear();
    ui->usrLineEdit->setFocus();
}



void LoginDialog::on_exitBtn_clicked()
{
    this->close();
}

void LoginDialog::on_usrLineEdit_cursorPositionChanged(int arg1, int arg2)
{

}

void LoginDialog::on_pwdLineEdit_cursorPositionChanged(int arg1, int arg2)
{

}
//main.cpp
```

```
#include "mainwindow.h"
#include <QApplication>
#include "logindialog.h"
#include "airlogindialog.h"
#include "connection.h"
#include "useritem.h"
extern int yonghu;

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

    if(!createConnection())
        return 1;

    MainWindow w;
    useritem u;
    AirLoginDialog ald;
    LoginDialog dlg;

    if (dlg.exec() == QDialog::Accepted && (yonghu==1 || yonghu==2))
    {
        if(yonghu==1){
            w.show();
            return a.exec();
        }
        if(yonghu==2){
            u.show();
            return a.exec();
        }
    }

/*
    if (dlg.exec() == QDialog::Accepted)
    {
       if(!createConnection())
           return 1;

       u.show();
       return a.exec();
    }
 */
```

```
    else return 0;

    //return a.exec();
}
//#include "mainwindow.h"
#include "ui_mainwindow.h"

#include <QListView>
#include <QFileDialog>
#include <QMessageBox>
#include <QSqlDatabase>
#include <QSqlError>
#include <QSqlQuery>
#include <QDebug>
#include <QSortFilterProxyModel>
#include "logindialog.h"
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    setWindowIcon(QIcon(":/myapp.ico"));

    mDefaultDB = QCoreApplication::applicationDirPath() + "\\crm.db";
    mAllType<<"国际航空"<<"南方航空"<<"海南航空"<<"长沙航空"<<"深圳航空"<<"东方航
空"<<"厦门航空"<<"四川航空";
    mAllArea<<"华东"<<"华南"<<"华中"<<"华北"<<"其他"<<"外";
    mAllProvince<<"山东"<<"浙江"<<"江苏"<<"安徽"<<"上海"<<"福建"
                <<"广东"<<"广西"<<"海南"
                <<"湖北"<<"湖南"<<"河南"<<"江西"
                <<"北京"<<"天津"<<"河北"<<"山西"<<"内蒙"
                <<"宁夏"<<"新疆"<<"青海"<<"陕西"<<"甘肃"<<"四川"<<"云南"<<"贵州
"<<"西藏"<<"重庆"<<"辽宁"<<"吉林"<<"黑龙江";
    mFilterType = "";
    mFilterArea = "";
    mFilterProvince = "";
    mFilterKey = "";

    InitMenu();

    OpenDb(mDefaultDB);
```

```
    InitTable();

    ui->treeWidget->expandAll();
    ui->treeWidget->setCurrentItem(ui->treeWidget->itemAt(0,0));

    ui->comboBox->setView(new QListView());
}

MainWindow::~MainWindow()
{
    ClearData();
    delete ui;
}

void MainWindow::InitMenu()
{
    QMenu *file = menuBar()->addMenu("&文件");
    file->addAction("打开...",this,SLOT(onMenuOpen(bool)));
    file->addAction("保存",this,SLOT(onMenuSave(bool)));
}

void MainWindow::InitModel()
{
    mModel->removeRows(0,mModel->rowCount());
    for(int i=0;i<mItems.size();i++)
    {
        Customer* p = mItems[i];
        QString sArea = Area2String(p->area);
        QString sProvince = Province2String(p->province);
        QString sType = Type2String(p->type);

        //区域过滤
        if(!mFilterArea.isEmpty())
        {
           if(mFilterArea != sArea)
           {
               continue;
           }
        }
        //省份过滤
        if(!mFilterProvince.isEmpty())
        {
```

```
            if(mFilterProvince != sProvince)
            {
                continue;
            }
        }
        //单位类别过滤
        if(!mFilterType.isEmpty())
        {
            if(mFilterType != sType)
            {
                continue;
            }
        }
        //关键字过滤
        if(!mFilterKey.isEmpty())
        {
            QStringList strs;
            strs<<p->name<<sType<<sArea<<sProvince<<p->city<<p->site<<p-
>contact<<p->buy<<p->research<<p->remark;
            bool bFind = false;
            foreach (QString s, strs)
            {
                if(s.indexOf(mFilterKey) != -1)
                {
                    bFind = true;
                    break;
                }
            }
            if(!bFind)
            {
                continue;
            }
        }

        QList<QStandardItem*> items;
        QStandardItem* item1 = new QStandardItem(p->name);
        QStandardItem* item2 = new QStandardItem(Type2String(p->type));
        QStandardItem* item3 = new QStandardItem(Area2String(p->area));
        QStandardItem* item4 = new QStandardItem(Province2String(p->province) );
        QStandardItem* item5 = new QStandardItem(p->city);
        QStandardItem* item6 = new QStandardItem(p->address);
        QStandardItem* item7 = new QStandardItem(p->site);
```

```
        QStandardItem* item8 = new QStandardItem(p->research);
        QStandardItem* item9 = new QStandardItem(p->remark);
        items.append(item1);
        items.append(item2);
        items.append(item3);
        items.append(item4);
        items.append(item5);
        items.append(item6);
        items.append(item7);
        items.append(item8);
        items.append(item9);
        mModel->appendRow(items);
    }
}

void MainWindow::InitTable()
{
    QTableView* t = ui->tableView;
    t->setEditTriggers(QTreeView::NoEditTriggers);                    //不能编辑
    t->setSelectionBehavior(QTreeView::SelectRows);                   //一次选中整
行
    t->setAlternatingRowColors(true);

    QString strTreeStyle = "QTableView::item {height: 25px}";
    t->setStyleSheet(strTreeStyle);

    QStringList headers;
    headers<< QStringLiteral("航班号")<<QStringLiteral("航空公司")
            << QStringLiteral("区域")<<QStringLiteral("省份")
            << QStringLiteral("城市")<<QStringLiteral("出发时间")
            << QStringLiteral("飞机号")<<QStringLiteral("乘员定额")
            << QStringLiteral("备注");
    mModel = new QStandardItemModel(t);
    mModel->setHorizontalHeaderLabels( headers );

    InitModel();
    t->setModel(mModel);
    for(int i=0;i<headers.size();i++)
    {
        if(i==0)
        {
            t->horizontalHeader()->resizeSection(i,200);
```

```
        }
        else if(i==1)
        {
            t->horizontalHeader()->resizeSection(i,80);
        }
        else if(i>=2 && i<=4)
        {
            t->horizontalHeader()->resizeSection(i,60);
        }
        else
        {
            t->horizontalHeader()->resizeSection(i,150);
        }
    }

    QModelIndex rootIndex = t->rootIndex();
    QModelIndex selIndex = mModel->index(0,0,rootIndex);
    t->setCurrentIndex(selIndex);
//    t->expandAll();
}

void MainWindow::AddItem(Customer *item)
{
    mItems.push_back(item);
}

void MainWindow::ClearData()
{
    for(int i=0;i<mItems.size();i++)
    {
        Customer* p = mItems[i];
        if(p)
        {
            delete p;
            p = NULL;
        }
    }
    mItems.clear();
}

void MainWindow::OpenDb(QString fileName)
{
```

```
    QSqlDatabase db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(fileName);
    if(db.open())
    {
        qDebug()<<"Database Opened";

        mDefaultDB = fileName;

        QSqlQuery sql_query;
        QString select_all_sql = "select * from Customer";
        //查询所有记录
        sql_query.prepare(select_all_sql);
        if(!sql_query.exec())
        {
            qDebug()<<sql_query.lastError();
        }
        else
        {
            while(sql_query.next())
            {
                Customer* p = new Customer;
                p->name = sql_query.value(0).toString();
                p->type = sql_query.value(1).toInt();
                p->area = sql_query.value(2).toInt();
                p->province = sql_query.value(3).toInt();
                p->city = sql_query.value(4).toString();
                p->address = sql_query.value(5).toString();
                p->site = sql_query.value(6).toString();
                p->contact = sql_query.value(7).toString();
                p->buy = sql_query.value(8).toString();
                p->research = sql_query.value(9).toString();
                p->remark = sql_query.value(10).toString();
                mItems.push_back(p);
            }
        }
    }
    db.close();
}

void MainWindow::SaveToDb(QString fileName)
{
    QSqlDatabase db = QSqlDatabase::addDatabase("QSQLITE");
```

```
    db.setDatabaseName(fileName);
    if(db.open())
    {
        qDebug()<<"Database Opened";

        QSqlQuery sql_query;
        QString create_sql = "create table if not exists Customer (name
varchar(100) primary key, \
                                                    type int ,\
                                                    area int,\
                                                    province int,\
                                                    city varchar(20),\
                                                    address varchar(100),\
                                                    site varchar(200),\
                                                    contact varchar(500),\
                                                    buy varchar(200),\
                                                    research varchar(100),\
                                                    remark varchar(200))"; //创建
数据表
//        QString insert_sql = "insert into Customer values(\"公司 1\",1,1,1,\"武
汉\",\"武汉洪山区 XXX\",\"www.xxx.com\",\"aaaaa\",\"bbbbb\",\"ccccc\",\"ddddd\")";
//插入数据
        QString insert_sql = "insert into Customer
values(\"%1\",%2,%3,%4,\"%5\",\"%6\",\"%7\",\"%8\",\"%9\",\"%10\",\"%11\")";    //
插入数据
        QString select_all_sql = "select * from Customer";

        sql_query.prepare(create_sql); //创建表
        if(!sql_query.exec()) //查看创建表是否成功
        {
            qDebug()<<QObject::tr("Table Create failed");
            qDebug()<<sql_query.lastError();
        }
        else
        {
            qDebug()<< "Table Created" ;

            for(int i=0;i<mItems.size();i++)
            {
                Customer* p = mItems[i];
                QString sql = insert_sql;
                sql.replace("%10",p->research);
```

```
                sql.replace("%11",p->remark);
                sql = sql.arg(p->name).arg(p->type).arg(p->area).arg(p->province).arg(p->city).arg(p->address).arg(p->site).arg(p->contact).arg(p->buy);
                sql_query.prepare(sql);
                if(!sql_query.exec())
                {
                    qDebug()<<sql_query.lastError();
                }
                else
                {
                    qDebug()<<"插入记录成功";
                }
            }
        }
    }
    db.close();
}

void MainWindow::InsertToDb(Customer *p)
{
    QSqlDatabase db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(mDefaultDB);
    if(db.open())
    {
        qDebug()<<"Database Opened";

        QSqlQuery sql_query;
        QString create_sql = "create table if not exists Customer (name varchar(100) primary key, \
                                                                   type int ,\
                                                                   area int,\
                                                                   province int,\
                                                                   city varchar(20),\
                                                                   address varchar(100),\
                                                                   site varchar(200),\
                                                                   contact varchar(500),\
                                                                   buy varchar(200),\
                                                                   research varchar(100),\
                                                                   remark varchar(200))"; //创建数据表
        sql_query.prepare(create_sql);
        sql_query.exec();                    //创建表
```

```
        QString sql = "insert into Customer
values(\"%1\",%2,%3,%4,\"%5\",\"%6\",\"%7\",\"%8\",\"%9\",\"%10\",\"%11\")";    //
插入数据
        sql.replace("%10",p->research);
        sql.replace("%11",p->remark);
        sql = sql.arg(p->name).arg(p->type).arg(p->area).arg(p->province).arg(p-
>city).arg(p->address).arg(p->site).arg(p->contact).arg(p->buy);
        sql_query.prepare(sql);
        if(!sql_query.exec())
        {
            qDebug()<<sql_query.lastError();
        }
        else
        {
            qDebug()<<"插入记录成功";
        }
    }
    db.close();
}

void MainWindow::DeleteFromDb(QString name)
{
    QSqlDatabase db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(mDefaultDB);
    if(db.open())
    {
        qDebug()<<"Database Opened";

        QSqlQuery sql_query;
        QString sql = QString("delete from Customer where name='%1'").arg(name);
        sql_query.prepare(sql);
        if(!sql_query.exec())
        {
            qDebug()<<sql_query.lastError();
        }
        else
        {
            qDebug()<<"删除记录成功";
        }
    }
    db.close();
```

```
}

void MainWindow::onMenuOpen(bool checked)
{
    QString fileName = QFileDialog::getOpenFileName(this,"打开文件","","Database Files(*.db)");
    if (fileName.isNull())
    {
        return;
    }
    if(fileName.indexOf(".db") != -1)
    {
        ClearData();
        OpenDb(fileName);
        InitTable();
    }
}

void MainWindow::onMenuSave(bool checked)
{
    QString fileName = QFileDialog::getSaveFileName(this,"保存到文件","","Database Files(*.db)");
    if (fileName.isNull())
    {
        return;
    }
    if(QFile::exists(fileName))
    {
        QFile::remove(fileName);
    }
    SaveToDb(fileName);
}

void MainWindow::on_btnSearch_clicked()
{
    QString s = ui->lineEdit->text();
    mFilterKey = s;
    InitModel();
}

void MainWindow::on_btnAdd_clicked()
{
```

```
    DlgItem dlg(NULL);
    connect(&dlg,&DlgItem::sigAddItem,this,MainWindow::slotAddItem);
    dlg.exec();
}

void MainWindow::on_btnDelete_clicked()
{
//    int row = ui->treeView->currentIndex().row();
    int row = ui->tableView->currentIndex().row();
    QString name = mModel->item(row,0)->text();
    DeleteItem(name);

    mModel->removeRow(row);

    DeleteFromDb(name);
}

void MainWindow::slotAddItem(Customer *p)
{
    mItems.push_back(p);

    QList<QStandardItem*> items;
    QStandardItem* item1 = new QStandardItem(p->name);
    QStandardItem* item2 = new QStandardItem(Type2String(p->type));
    QStandardItem* item3 = new QStandardItem(Area2String(p->area));
    QStandardItem* item4 = new QStandardItem(Province2String(p->province) );
    QStandardItem* item5 = new QStandardItem(p->city);
    QStandardItem* item6 = new QStandardItem(p->address);
    QStandardItem* item7 = new QStandardItem(p->site);
    QStandardItem* item8 = new QStandardItem(p->research);
    QStandardItem* item9 = new QStandardItem(p->remark);
    items.append(item1);
    items.append(item2);
    items.append(item3);
    items.append(item4);
    items.append(item5);
    items.append(item6);
    items.append(item7);
    items.append(item8);
    items.append(item9);
    mModel->appendRow(items);
```

```
    InsertToDb(p);
}

QString MainWindow::Type2String(int n)
{
    return mAllType[n];
}

QString MainWindow::Area2String(int n)
{
    return mAllArea[n];
}

QString MainWindow::Province2String(int n)
{
    if(n>=0 && n<mAllProvince.size() )
    {
        return mAllProvince[n];
    }
    return "";
}

Customer *MainWindow::FindItem(QString name)
{
    for(int i=0;i<mItems.size();i++)
    {
        if(mItems[i]->name == name)
        {
            return mItems[i];
        }
    }
    return NULL;
}

void MainWindow::DeleteItem(QString name)
{
    for(int i=0;i<mItems.size();i++)
    {
        if(mItems[i]->name == name)
        {
            mItems.removeAt(i);
            return;
```

```
        }
    }
}

void MainWindow::on_comboBox_currentTextChanged(const QString &text)
{
    if(text == "航空公司")
    {
        mFilterType = "";
    }
    else
    {
        mFilterType = text;
    }
    InitModel();
}

void MainWindow::on_treeWidget_currentItemChanged(QTreeWidgetItem *current,
QTreeWidgetItem *previous)
{
    QString s = current->text(0);
    if(s == "全部")
    {
        mFilterArea = "";
        mFilterProvince = "";
    }
    else if(mAllArea.contains(s))
    {
        //区域
        mFilterArea = s;
        mFilterProvince = "";
    }
    else if(mAllProvince.contains(s))
    {
        //省份
        mFilterArea = "";
        mFilterProvince = s;
    }
    InitModel();
}

void MainWindow::on_tableView_doubleClicked(const QModelIndex &index)
```

```
{
    QString name = mModel->item(index.row(),0)->text();
    Customer *p = FindItem(name);
    if(p)
    {
        DlgItem dlg(p);
        if(QDialog::Accepted == dlg.exec())
        {
            mModel->item(index.row(),0)->setText(p->name);
            mModel->item(index.row(),1)->setText(Type2String(p->type));
            mModel->item(index.row(),2)->setText(Area2String(p->area));
            mModel->item(index.row(),3)->setText(Province2String(p->province));
            mModel->item(index.row(),4)->setText(p->city);
            mModel->item(index.row(),5)->setText(p->address);
            mModel->item(index.row(),6)->setText(p->site);
            mModel->item(index.row(),7)->setText(p->research);
            mModel->item(index.row(),8)->setText(p->remark);

            DeleteFromDb(name);
            InsertToDb(p);
        }
    }
}

void MainWindow::on_pushButton_clicked()
{
    LoginDialog *dlg = new LoginDialog;
    dlg->show();
    this->close();
}
//useitem.cpp
#include "mainwindow.h"
#include "ui_mainwindow.h"

#include <QListView>
#include <QFileDialog>
#include <QMessageBox>
#include <QSqlDatabase>
#include <QSqlError>
#include <QSqlQuery>
#include <QDebug>
#include <QSortFilterProxyModel>
```

```
#include "logindialog.h"
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    setWindowIcon(QIcon(":/myapp.ico"));

    mDefaultDB = QCoreApplication::applicationDirPath() + "\\crm.db";
    mAllType<<"国际航空"<<"南方航空"<<"海南航空"<<"长沙航空"<<"深圳航空"<<"东方航
空"<<"厦门航空"<<"四川航空";
    mAllArea<<"华东"<<"华南"<<"华中"<<"华北"<<"其他"<<"外";
    mAllProvince<<"山东"<<"浙江"<<"江苏"<<"安徽"<<"上海"<<"福建"
                <<"广东"<<"广西"<<"海南"
                <<"湖北"<<"湖南"<<"河南"<<"江西"
                <<"北京"<<"天津"<<"河北"<<"山西"<<"内蒙"
                <<"宁夏"<<"新疆"<<"青海"<<"陕西"<<"甘肃"<<"四川"<<"云南"<<"贵州
"<<"西藏"<<"重庆"<<"辽宁"<<"吉林"<<"黑龙江";
    mFilterType = "";
    mFilterArea = "";
    mFilterProvince = "";
    mFilterKey = "";

    InitMenu();

    OpenDb(mDefaultDB);
    InitTable();

    ui->treeWidget->expandAll();
    ui->treeWidget->setCurrentItem(ui->treeWidget->itemAt(0,0));

    ui->comboBox->setView(new QListView());
}

MainWindow::~MainWindow()
{
    ClearData();
    delete ui;
}

void MainWindow::InitMenu()
{
```

```
    QMenu *file = menuBar()->addMenu("&文件");
    file->addAction("打开...",this,SLOT(onMenuOpen(bool)));
    file->addAction("保存",this,SLOT(onMenuSave(bool)));
}

void MainWindow::InitModel()
{
    mModel->removeRows(0,mModel->rowCount());
    for(int i=0;i<mItems.size();i++)
    {
        Customer* p = mItems[i];
        QString sArea = Area2String(p->area);
        QString sProvince = Province2String(p->province);
        QString sType = Type2String(p->type);

        //区域过滤
        if(!mFilterArea.isEmpty())
        {
           if(mFilterArea != sArea)
           {
               continue;
           }
        }
        //省份过滤
        if(!mFilterProvince.isEmpty())
        {
           if(mFilterProvince != sProvince)
           {
               continue;
           }
        }
        //单位类别过滤
        if(!mFilterType.isEmpty())
        {
           if(mFilterType != sType)
           {
               continue;
           }
        }
        //关键字过滤
        if(!mFilterKey.isEmpty())
        {
```

```
            QStringList strs;
            strs<<p->name<<sType<<sArea<<sProvince<<p->city<<p->site<<p-
>contact<<p->buy<<p->research<<p->remark;
            bool bFind = false;
            foreach (QString s, strs)
            {
                if(s.indexOf(mFilterKey) != -1)
                {
                    bFind = true;
                    break;
                }
            }
            if(!bFind)
            {
                continue;
            }
        }

        QList<QStandardItem*> items;
        QStandardItem* item1 = new QStandardItem(p->name);
        QStandardItem* item2 = new QStandardItem(Type2String(p->type));
        QStandardItem* item3 = new QStandardItem(Area2String(p->area));
        QStandardItem* item4 = new QStandardItem(Province2String(p->province) );
        QStandardItem* item5 = new QStandardItem(p->city);
        QStandardItem* item6 = new QStandardItem(p->address);
        QStandardItem* item7 = new QStandardItem(p->site);
        QStandardItem* item8 = new QStandardItem(p->research);
        QStandardItem* item9 = new QStandardItem(p->remark);
        items.append(item1);
        items.append(item2);
        items.append(item3);
        items.append(item4);
        items.append(item5);
        items.append(item6);
        items.append(item7);
        items.append(item8);
        items.append(item9);
        mModel->appendRow(items);
    }
}

void MainWindow::InitTable()
```

```
{
    QTableView* t = ui->tableView;
    t->setEditTriggers(QTreeView::NoEditTriggers);                    //不能编辑
    t->setSelectionBehavior(QTreeView::SelectRows);                   //一次选中整
行
    t->setAlternatingRowColors(true);

    QString strTreeStyle = "QTableView::item {height: 25px}";
    t->setStyleSheet(strTreeStyle);

    QStringList headers;
    headers<< QStringLiteral("航班号")<<QStringLiteral("航空公司")
           << QStringLiteral("区域")<<QStringLiteral("省份")
           << QStringLiteral("城市")<<QStringLiteral("出发时间")
           << QStringLiteral("飞机号")<<QStringLiteral("乘员定额")
           << QStringLiteral("备注");
    mModel = new QStandardItemModel(t);
    mModel->setHorizontalHeaderLabels( headers );

    InitModel();
    t->setModel(mModel);
    for(int i=0;i<headers.size();i++)
    {
        if(i==0)
        {
            t->horizontalHeader()->resizeSection(i,200);
        }
        else if(i==1)
        {
            t->horizontalHeader()->resizeSection(i,80);
        }
        else if(i>=2 && i<=4)
        {
            t->horizontalHeader()->resizeSection(i,60);
        }
        else
        {
            t->horizontalHeader()->resizeSection(i,150);
        }
    }

    QModelIndex rootIndex = t->rootIndex();
```

```
    QModelIndex selIndex = mModel->index(0,0,rootIndex);
    t->setCurrentIndex(selIndex);
//    t->expandAll();
}

void MainWindow::AddItem(Customer *item)
{
    mItems.push_back(item);
}

void MainWindow::ClearData()
{
    for(int i=0;i<mItems.size();i++)
    {
        Customer* p = mItems[i];
        if(p)
        {
            delete p;
            p = NULL;
        }
    }
    mItems.clear();
}

void MainWindow::OpenDb(QString fileName)
{
    QSqlDatabase db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(fileName);
    if(db.open())
    {
        qDebug()<<"Database Opened";

        mDefaultDB = fileName;

        QSqlQuery sql_query;
        QString select_all_sql = "select * from Customer";
        //查询所有记录
        sql_query.prepare(select_all_sql);
        if(!sql_query.exec())
        {
            qDebug()<<sql_query.lastError();
        }
```

```
        else
        {
            while(sql_query.next())
            {
                Customer* p = new Customer;
                p->name = sql_query.value(0).toString();
                p->type = sql_query.value(1).toInt();
                p->area = sql_query.value(2).toInt();
                p->province = sql_query.value(3).toInt();
                p->city = sql_query.value(4).toString();
                p->address = sql_query.value(5).toString();
                p->site = sql_query.value(6).toString();
                p->contact = sql_query.value(7).toString();
                p->buy = sql_query.value(8).toString();
                p->research = sql_query.value(9).toString();
                p->remark = sql_query.value(10).toString();
                mItems.push_back(p);
            }
        }
    }
    db.close();
}

void MainWindow::SaveToDb(QString fileName)
{
    QSqlDatabase db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(fileName);
    if(db.open())
    {
        qDebug()<<"Database Opened";

        QSqlQuery sql_query;
        QString create_sql = "create table if not exists Customer (name
varchar(100) primary key, \
                                                    type int ,\
                                                    area int,\
                                                    province int,\
                                                    city varchar(20),\
                                                    address varchar(100),\
                                                    site varchar(200),\
                                                    contact varchar(500),\
                                                    buy varchar(200),\
```

```
                                                    research varchar(100),\
                                                    remark varchar(200))"; //创建
数据表
//        QString insert_sql = "insert into Customer values(\"公司1\",1,1,1,\"武
汉\",\"武汉洪山区XXX\",\"www.xxx.com\",\"aaaaa\",\"bbbbb\",\"ccccc\",\"ddddd\")";
//插入数据
        QString insert_sql = "insert into Customer
values(\"%1\",%2,%3,%4,\"%5\",\"%6\",\"%7\",\"%8\",\"%9\",\"%10\",\"%11\")";    //
插入数据
        QString select_all_sql = "select * from Customer";

        sql_query.prepare(create_sql); //创建表
        if(!sql_query.exec()) //查看创建表是否成功
        {
            qDebug()<<QObject::tr("Table Create failed");
            qDebug()<<sql_query.lastError();
        }
        else
        {
            qDebug()<< "Table Created" ;

            for(int i=0;i<mItems.size();i++)
            {
                Customer* p = mItems[i];
                QString sql = insert_sql;
                sql.replace("%10",p->research);
                sql.replace("%11",p->remark);
                sql = sql.arg(p->name).arg(p->type).arg(p->area).arg(p-
>province).arg(p->city).arg(p->address).arg(p->site).arg(p->contact).arg(p->buy);
                sql_query.prepare(sql);
                if(!sql_query.exec())
                {
                    qDebug()<<sql_query.lastError();
                }
                else
                {
                    qDebug()<<"插入记录成功";
                }
            }
        }
    }
    db.close();
```

```
}

void MainWindow::InsertToDb(Customer *p)
{
    QSqlDatabase db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(mDefaultDB);
    if(db.open())
    {
        qDebug()<<"Database Opened";

        QSqlQuery sql_query;
        QString create_sql = "create table if not exists Customer (name
varchar(100) primary key, \
                                                    type int ,\
                                                    area int,\
                                                    province int,\
                                                    city varchar(20),\
                                                    address varchar(100),\
                                                    site varchar(200),\
                                                    contact varchar(500),\
                                                    buy varchar(200),\
                                                    research varchar(100),\
                                                    remark varchar(200))"; //创建
数据表
        sql_query.prepare(create_sql);
        sql_query.exec();                   //创建表

        QString sql = "insert into Customer
values(\"%1\",%2,%3,%4,\"%5\",\"%6\",\"%7\",\"%8\",\"%9\",\"%10\",\"%11\")";    //
插入数据
        sql.replace("%10",p->research);
        sql.replace("%11",p->remark);
        sql = sql.arg(p->name).arg(p->type).arg(p->area).arg(p->province).arg(p-
>city).arg(p->address).arg(p->site).arg(p->contact).arg(p->buy);
        sql_query.prepare(sql);
        if(!sql_query.exec())
        {
            qDebug()<<sql_query.lastError();
        }
        else
        {
            qDebug()<<"插入记录成功";
```

```
        }
    }
    db.close();
}

void MainWindow::DeleteFromDb(QString name)
{
    QSqlDatabase db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(mDefaultDB);
    if(db.open())
    {
        qDebug()<<"Database Opened";

        QSqlQuery sql_query;
        QString sql = QString("delete from Customer where name='%1'").arg(name);
        sql_query.prepare(sql);
        if(!sql_query.exec())
        {
            qDebug()<<sql_query.lastError();
        }
        else
        {
            qDebug()<<"删除记录成功";
        }
    }
    db.close();
}

void MainWindow::onMenuOpen(bool checked)
{
    QString fileName = QFileDialog::getOpenFileName(this,"打开文件","","Database
Files(*.db)");
    if (fileName.isNull())
    {
        return;
    }
    if(fileName.indexOf(".db") != -1)
    {
        ClearData();
        OpenDb(fileName);
        InitTable();
    }
```

```
}

void MainWindow::onMenuSave(bool checked)
{
    QString fileName = QFileDialog::getSaveFileName(this,"保存到文件","","Database Files(*.db)");
    if (fileName.isNull())
    {
        return;
    }
    if(QFile::exists(fileName))
    {
        QFile::remove(fileName);
    }
    SaveToDb(fileName);
}

void MainWindow::on_btnSearch_clicked()
{
    QString s = ui->lineEdit->text();
    mFilterKey = s;
    InitModel();
}

void MainWindow::on_btnAdd_clicked()
{
    DlgItem dlg(NULL);
    connect(&dlg,&DlgItem::sigAddItem,this,MainWindow::slotAddItem);
    dlg.exec();
}

void MainWindow::on_btnDelete_clicked()
{
//    int row = ui->treeView->currentIndex().row();
    int row = ui->tableView->currentIndex().row();
    QString name = mModel->item(row,0)->text();
    DeleteItem(name);

    mModel->removeRow(row);

    DeleteFromDb(name);
}
```

```
void MainWindow::slotAddItem(Customer *p)
{
    mItems.push_back(p);

    QList<QStandardItem*> items;
    QStandardItem* item1 = new QStandardItem(p->name);
    QStandardItem* item2 = new QStandardItem(Type2String(p->type));
    QStandardItem* item3 = new QStandardItem(Area2String(p->area));
    QStandardItem* item4 = new QStandardItem(Province2String(p->province) );
    QStandardItem* item5 = new QStandardItem(p->city);
    QStandardItem* item6 = new QStandardItem(p->address);
    QStandardItem* item7 = new QStandardItem(p->site);
    QStandardItem* item8 = new QStandardItem(p->research);
    QStandardItem* item9 = new QStandardItem(p->remark);
    items.append(item1);
    items.append(item2);
    items.append(item3);
    items.append(item4);
    items.append(item5);
    items.append(item6);
    items.append(item7);
    items.append(item8);
    items.append(item9);
    mModel->appendRow(items);

    InsertToDb(p);
}

QString MainWindow::Type2String(int n)
{
    return mAllType[n];
}

QString MainWindow::Area2String(int n)
{
    return mAllArea[n];
}

QString MainWindow::Province2String(int n)
{
    if(n>=0 && n<mAllProvince.size() )
```

```
    {
        return mAllProvince[n];
    }
    return "";
}

Customer *MainWindow::FindItem(QString name)
{
    for(int i=0;i<mItems.size();i++)
    {
        if(mItems[i]->name == name)
        {
            return mItems[i];
        }
    }
    return NULL;
}

void MainWindow::DeleteItem(QString name)
{
    for(int i=0;i<mItems.size();i++)
    {
        if(mItems[i]->name == name)
        {
            mItems.removeAt(i);
            return;
        }
    }
}

void MainWindow::on_comboBox_currentTextChanged(const QString &text)
{
    if(text == "航空公司")
    {
        mFilterType = "";
    }
    else
    {
        mFilterType = text;
    }
    InitModel();
}
```

```
void MainWindow::on_treeWidget_currentItemChanged(QTreeWidgetItem *current,
QTreeWidgetItem *previous)
{
    QString s = current->text(0);
    if(s == "全部")
    {
        mFilterArea = "";
        mFilterProvince = "";
    }
    else if(mAllArea.contains(s))
    {
        //区域
        mFilterArea = s;
        mFilterProvince = "";
    }
    else if(mAllProvince.contains(s))
    {
        //省份
        mFilterArea = "";
        mFilterProvince = s;
    }
    InitModel();
}

void MainWindow::on_tableView_doubleClicked(const QModelIndex &index)
{
    QString name = mModel->item(index.row(),0)->text();
    Customer *p = FindItem(name);
    if(p)
    {
        DlgItem dlg(p);
        if(QDialog::Accepted == dlg.exec())
        {
            mModel->item(index.row(),0)->setText(p->name);
            mModel->item(index.row(),1)->setText(Type2String(p->type));
            mModel->item(index.row(),2)->setText(Area2String(p->area));
            mModel->item(index.row(),3)->setText(Province2String(p->province));
            mModel->item(index.row(),4)->setText(p->city);
            mModel->item(index.row(),5)->setText(p->address);
            mModel->item(index.row(),6)->setText(p->site);
            mModel->item(index.row(),7)->setText(p->research);
```

```
            mModel->item(index.row(),8)->setText(p->remark);

            DeleteFromDb(name);
            InsertToDb(p);
        }
    }
}

void MainWindow::on_pushButton_clicked()
{
    LoginDialog *dlg = new LoginDialog;
    dlg->show();
    this->close();
}
//userticket.cpp
#include "userticket.h"
#include "ui_userticket.h"
userticket::userticket(Customer1* item,QWidget *parent) :
    QDialog(parent),
    ui(new Ui::userticket)
{
    ui->setupUi(this);

    mItem = item;

    QFont font( "Microsoft YaHei", 10, 75);
    ui->labelTip->setFont(font);
    ui->labelTip->setStyleSheet("QLabel{color:red;}");
    ui->tableWidget->setRowCount(20);
    for(int i=0;i<20;i++)
    {
        ui->tableWidget->setItem(i,0,new QTableWidgetItem(""));
        ui->tableWidget->setItem(i,1,new QTableWidgetItem(""));
        ui->tableWidget->setItem(i,2,new QTableWidgetItem(""));
        ui->tableWidget->setItem(i,3,new QTableWidgetItem(""));
        ui->tableWidget->setItem(i,4,new QTableWidgetItem(""));
        ui->tableWidget->setItem(i,5,new QTableWidgetItem(""));
    }

    on_combo2_currentTextChanged("华东");
    if(mItem)
    {
```

```
        //修改
        ui->btnAdd->setText("修改");

        ui->edit1->setText(mItem->name);
        ui->combo1->setCurrentIndex(mItem->type);
        ui->combo2->setCurrentIndex(mItem->area);
        on_combo2_currentTextChanged(ui->combo2->currentText());
        ui->combo3->setCurrentText(Province2String(mItem->province));
        ui->edit2->setText(mItem->city);
        ui->edit3->setText(mItem->address);
        ui->edit4->setText(mItem->site);
        ui->textEdit->setText(mItem->buy);
        ui->textEdit2->setText(mItem->research);
        ui->textEdit3->setText(mItem->remark);

        QStringList all = mItem->contact.split(";");
        for(int i=0;i<all.size();i++)
        {
            QStringList row = all[i].split(",");
            if(row.size() >= 6)
            {
                bool bRed = !row[5].isEmpty();
                ui->tableWidget->item(i,0)->setText(row[0]);
                ui->tableWidget->item(i,1)->setText(row[1]);
                ui->tableWidget->item(i,2)->setText(row[2]);
                ui->tableWidget->item(i,3)->setText(row[3]);
                ui->tableWidget->item(i,4)->setText(row[4]);
                ui->tableWidget->item(i,5)->setText(row[5]);
                if(!row[5].isEmpty())
                {
                    ui->tableWidget->item(i,0)->setBackground(Qt::red);
                    ui->tableWidget->item(i,1)->setBackground(Qt::red);
                    ui->tableWidget->item(i,2)->setBackground(Qt::red);
                    ui->tableWidget->item(i,3)->setBackground(Qt::red);
                    ui->tableWidget->item(i,4)->setBackground(Qt::red);
                    ui->tableWidget->item(i,5)->setBackground(Qt::red);
                }
            }
        }
    }
}
```

```
userticket::~userticket()
{
    delete ui;
}

void userticket::on_btnAdd_clicked()
{
    if(ui->edit1->text().isEmpty())
    {
        ui->labelTip->setText("航班号名称不能为空!");
        return;
    }
    Customer1* p = new Customer1;
    p->name = ui->edit1->text();
    p->type = ui->combo1->currentIndex();
    p->area = ui->combo2->currentIndex();
    p->province = String2Province(ui->combo3->currentText());
    p->city = ui->edit2->text();
    p->address = ui->edit3->text();
    p->site = ui->edit4->text();
    p->buy = ui->textEdit->toPlainText();
    p->research = ui->textEdit2->toPlainText();
    p->remark = ui->textEdit3->toPlainText();

    QString sAll = "";
    QString s = "";
    int columns = ui->tableWidget->columnCount();
    for(int i=0;i<ui->tableWidget->rowCount();i++)
    {
        if(ui->tableWidget->item(i,0)->text().isEmpty())
        {
            continue;
        }
        for(int j=0;j<columns;j++)
        {
            //,和;是用来分隔的标记，禁止用户输入这 2 个符号，如果不小心输了，程序也
替换成空格
            s = ui->tableWidget->item(i,j)->text();
            s.replace(",","，");
            s.replace(";","；");
            sAll += s;
            if(j == columns-1)
```

```
            {
                sAll += ";";
            }
            else
            {
                sAll += ",";
            }
        }
    }
    p->contact = sAll;

    if(mItem)
    {
        //修改
/*
        mItem->name = p->name;
        mItem->type = p->type;
        mItem->area = p->area;
        mItem->province = p->province;
        mItem->city = p->city;
        mItem->address = p->address;
        mItem->site = p->site;
        mItem->contact = p->contact;
        mItem->buy = p->buy;
        mItem->research = p->research;
        mItem->remark = p->remark;
*/
        mItem->contact = p->contact;
        accept();
    }
    else
    {
        //新增
        emit sigAddItem(p);

        ui->edit1->setText("");
        ui->edit2->setText("");
        ui->edit3->setText("");
        ui->edit4->setText("");
        ui->textEdit->setText("");
        ui->textEdit2->setText("");
        ui->textEdit3->setText("");
```

```
        for(int i=0;i<20;i++)
        {
            ui->tableWidget->item(i,0)->setText("");
            ui->tableWidget->item(i,1)->setText("");
            ui->tableWidget->item(i,2)->setText("");
            ui->tableWidget->item(i,3)->setText("");
            ui->tableWidget->item(i,4)->setText("");
            ui->tableWidget->item(i,5)->setText("");
        }
        ui->edit1->setFocus();
        ui->labelTip->setText("增加成功!");
    }

}

void userticket::on_btnCancel_clicked()
{
    reject();
}

int userticket::String2Province(QString s)
{
    QStringList AllProvince;
    AllProvince<<"山东"<<"浙江"<<"江苏"<<"安徽"<<"上海"<<"福建"
               <<"广东"<<"广西"<<"海南"
               <<"湖北"<<"湖南"<<"河南"<<"江西"
               <<"北京"<<"天津"<<"河北"<<"山西"<<"内蒙"
               <<"宁夏"<<"新疆"<<"青海"<<"陕西"<<"甘肃"<<"四川"<<"云南"<<"贵州
"<<"西藏"<<"重庆"<<"辽宁"<<"吉林"<<"黑龙江";
    return AllProvince.indexOf(s);
}

QString userticket::Province2String(int n)
{
    QStringList AllProvince;
    AllProvince<<"山东"<<"浙江"<<"江苏"<<"安徽"<<"上海"<<"福建"
               <<"广东"<<"广西"<<"海南"
               <<"湖北"<<"湖南"<<"河南"<<"江西"
               <<"北京"<<"天津"<<"河北"<<"山西"<<"内蒙"
               <<"宁夏"<<"新疆"<<"青海"<<"陕西"<<"甘肃"<<"四川"<<"云南"<<"贵州
"<<"西藏"<<"重庆"<<"辽宁"<<"吉林"<<"黑龙江";
    if(n>=0 && n<AllProvince.size())
```

```
    {
        return AllProvince[n];
    }
    return "";
}

void userticket::on_combo2_currentTextChanged(const QString &text)
{
    ui->combo3->clear();
    if(text == "华东")
    {
        ui->combo3->addItems(QStringList()<<"山东"<<"浙江"<<"江苏"<<"安徽"<<"上海
"<<"福建");
    }
    else if(text == "华南")
    {
        ui->combo3->addItems(QStringList()<<"广东"<<"广西"<<"海南");
    }
    else if(text == "华中")
    {
        ui->combo3->addItems(QStringList()<<"湖北"<<"湖南"<<"河南"<<"江西");
    }
    else if(text == "华北")
    {
        ui->combo3->addItems(QStringList()<<"北京"<<"天津"<<"河北"<<"山西"<<"内蒙
");
    }
    else if(text == "其他")
    {
        ui->combo3->addItems(QStringList()<<"宁夏"<<"新疆"<<"青海"<<"陕西"<<"甘肃
"<<"四川"<<"云南"<<"贵州"<<"西藏"<<"重庆"<<"辽宁"<<"吉林"<<"黑龙江");
    }
}

void userticket::on_edit1_textChanged(const QString &arg1)
{
    ui->labelTip->setText("");
}
```

见文件夹