



# **数据结构课程设计报告**

**题目：航空客运订票系统**

学 院 计算机学院

专 业 网络工程

年级班别 2018级 4 班

学 号 3118005320

学生姓名 陈智超

指导教师 刘添添

成 绩 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2020 年 1 月

**报告：**

**报告内容：** □详细　 □完整　 □基本完整 □不完整

**设计方案：** □非常合理　 □合理　 □基本合理 □较差

**算法实现：** □全部实现　 □基本实现　 □部分实现 □实现较差

**测试样例：** □完备　 □比较完备　 □基本完备 □不完备

**文档格式：** □规范　 □比较规范　 □基本规范 □不规范

**答辩：**

□理解题目透彻，问题回答流利

□理解题目较透彻，回答问题基本正确

□部分理解题目，部分问题回答正确

□未能完全理解题目，答辩情况较差

**总评成绩：**

□优　　　□良　　　□中　　　□及格　　　□不及格

**航空客运订票系统**

1. **需求分析**

【运行环境】

Devc++

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。

【基本需求】

1. 每条航线所设计的信息有：航班ID、起点站名、终点站名、飞机号、飞行周日（星期几）、成员定额、总余票量、经济仓余票量、商务舱余票量、已订票的客户名单、经济舱候补客户名单、商务舱候补客户名单。

其中，经济舱候补客户名单和商务舱候补客户名单中的包括以下几个域：姓名、身份证、所需票量、预定票的等级

1. 作为示意系统，数据放在内存中；
2. 系统能实现的操作和功能如下：
3. 查看所有航班信息。
4. 查询模块：输入起点站和终点站进行搜索。
5. 订票模块的功能：

A：输入航班ID进行订票，当航班ID不存在或者不合法时要求重新输入航班ID，然后输入订票数量和等级，数量和等级要合法，订票数目要大于0，等级输入1代表经济舱，输入其他数据商务舱。不合法则重新输入，再判断对应等级的票是否足够 ，足够则继续输入客户信息（姓名和身份证），然后订票成功。当对应等级的票数不足时，询问是否改变订票计划，同意改变计划则重新输入航班ID进行相应的订票操作。不同意改变计划则继续询问是否排队等候，不同意排队等候则询问是否需要推荐相同航线的其他航班，询问操作时，输入Y或者y表示同意，输入N，n或者其它数据表示不同意。

B：根据起点和终点查询航班

C：查询所有航班的相关信息

D:：查询航班的订票情况，输入航班ID，当航班ID不存在或者不合法时要求重新输入航班ID，根据ID查询航班，显示出已订票的客户的姓名、订票数目和仓位等级，为了保密，不能显示客户的身份证。

E：返回上一级菜单

1. 退票模块的功能：

A:先输入航班ID，然后输入姓名和身份证号码进行验证，验证成功则办理退票手续。然后查询该航班是否有人排队候补，如果有，而且票数够了，则首先询问排在第一的客户，是否需要订票，是则为他办理订票手续，否则出队，依次询问其他排队候补的客户。若刚刚退票的是经济舱，则询问经济舱排队的客户；若是商务舱，则询问商务舱排队的客户。

B：返回上一级菜单

C：退出系统

1. 管理员模块的功能：

A：管理员密码验证

B：查询所有航班信息

C：根据航班ID查询客户信息（包括身份证号码）

D：增加航班（航班的各个信息输入要合法，不合法的重新输入）

E：删除航班，根据ID进行删除，ID不存在则删除失败。

F：返回上一级菜单

【测试数据】（为了提高可读性，详细测试数据将在测试结果中一一说明）：

//程序一执行就将六个基本航班插入航班链表

//以下两个变量为全局变量

pFlight

Flight flight1[6] {

{"广州", "北京", "1", "B3333", "星期一", 10, 10, 5},

{"广州", "北京", "2", "D5555", "星期二", 8, 8, 4},

{"广州", "北京", "3", "L6666", "星期天", 10, 10, 5},

{"上海", "广州", "4", "K9999", "星期三", 8, 8, 4},

{"广州", "成都", "5", "K7777", "星期三", 10, 10, 5},

{"成都", "广州", "6", "K8888", "星期三", 8, 8, 4}

};

【选做内容】

A：当客户所要订的航班票额不足的时候，系统会根据目的地，输出目的地相同的航线，询问客户是否要订票。（已做）

B：管理员的操作，包括管理员的验证，包括查看航班、增加航班、删除航班和查看订票的客户信息，能查到身份证号码。（已做，自己加上去的）

1. **概要设计**

**1.所要用到的头文件**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <conio.h>

**2.宏定义**

#define OK 1

#define ERROR 0

#define OVERFLOW -1

#define FALSE -1

#define TRUE 1

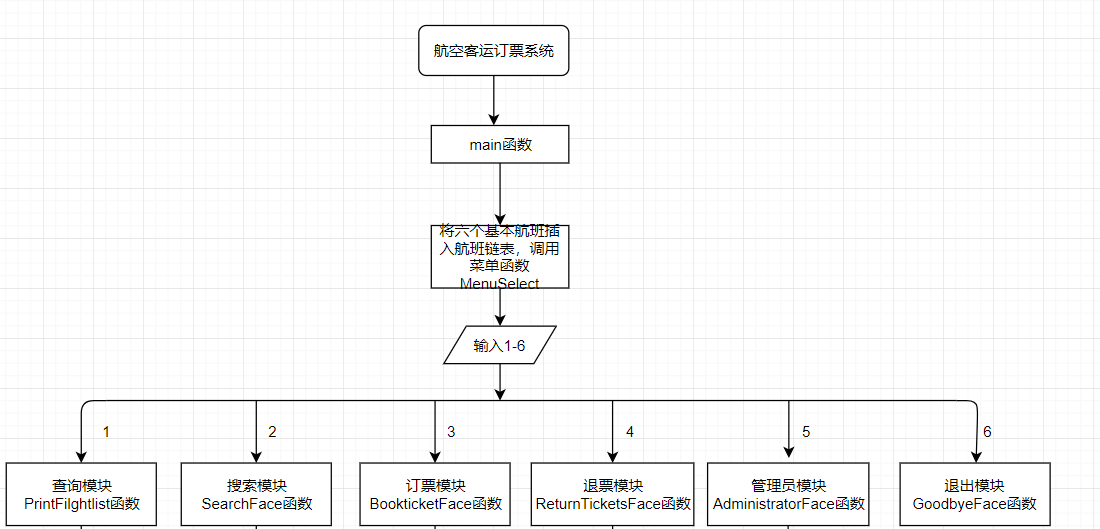
**3.类型定义**typedef int Status;//航班日期枚举类，星期一到星期天enum Week { Mon = 1, Tues = 2, Wed = 3, Thurs = 4, Fri = 5, Sat = 6, Sun = 7};//乘客节点typedef struct CustomerNode { char name[10];//客户姓名 int clientTickets;//客户订票量 char identification[20];//客户身份证号码 int rank;//舱位等级 CustomerNode \*next;} CustomerNode, \*CusLinkList;//候补队列中的节点typedef struct WaitPassenger { char name[10];//姓名 char identification[20]; //身份证 int preTickets;//预定的票量 struct WaitPassenger \*next;} WaitQNode, \*PWait;//候补队列typedef struct Queue { PWait front;//等候替补客户名单域的头指针 PWait rear;//等候替补客户名单域的尾指针} LinkQueue;//封装乘客的姓名和订票量和身份证//用于候补客户出队时把关键字返回

typedef struct NameAndNumAndID { char name[10];//姓名 char identification[20]; //身份证号码 int num;//订票量} NameAndNumAndID;//航班节点typedef struct Flight { char startPoint[20];//起点站名 char destination[20];//终点站名 char flightCodeID[20];//航班ID（相当于主键） char planeNum[20];//飞机号 char day[20];//飞行日期（星期几） int totalTickets;//乘员定额(总票数) int left;//总余票量 int leftEconomicTicket; //经济票剩余量 int leftBusinessTicket; //商务票剩余量 Flight \*next; CusLinkList cusLinkList;//乘员名单域，指向乘员名单链表的头指针 LinkQueue waitQueue1;//经济舱候补，等候替补的客户名单域，指向一个队列 LinkQueue waitQueue2;//商务舱候补，等候替补的客户名单域，指向一个队列} Flight, FlightNode, \*PFlight;

航班链表采用全局变量：pFlight，指向航班链表的头指针

pflight1，一个数组，存有留个基本航班，一个个入pFlight 航班链表中。

**4.基本的模块调用关系：**



**5.接口设计：**

ps：

函数的接口采用多行注释

@param表示该函数的参数

@return 表示函数的返回值

/\*\* \*根据自己输入的航班ID查询并以指针形式返回 \* @return 航班指针 \*/Flight \*find();/\*\* \* 初始化带头结点的链队列 \* @param q \* @return 操作状态

\*/Status InitQueue(LinkQueue &q);/\*\* \* 初始化已订票乘客指针链表 \* @param cusLinkList 航班中乘员链表的头指针 \* @return操作状态

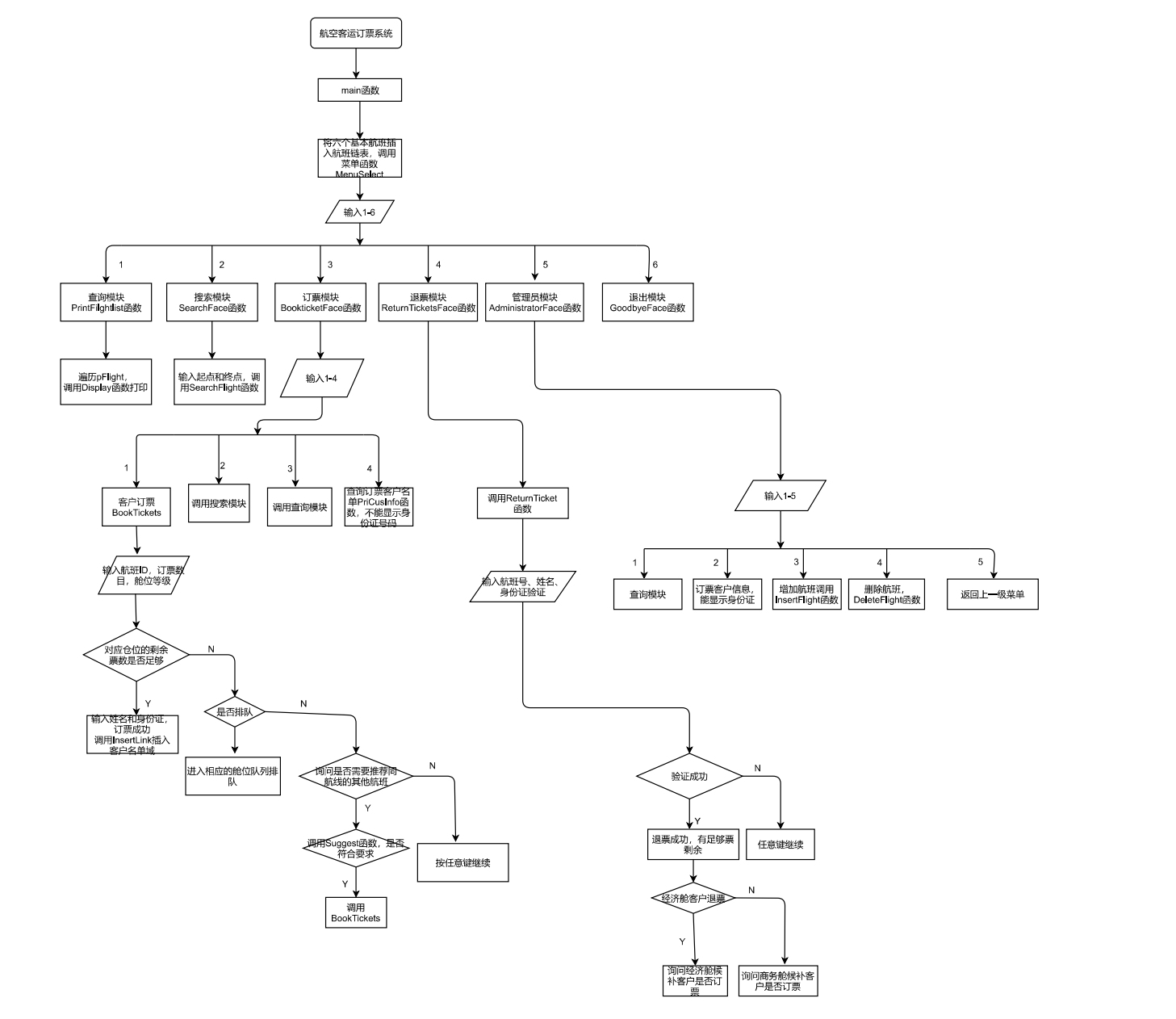
\*/Status InitCusLinkList(CusLinkList &cusLinkList);/\*\* \* 打印info航班的基本信息 \* @param info \*/void Display(struct Flight \*info);/\*\* \* 增加航班时输入日期的辅助函数（1代表星期一，7代表星期日） \* @param day1 传进来的1-7中的一个 \* @param day 数组类变量，可以返回回去给航班的日期变量 \* @return 返回操作状态，输入是否合法 \*/Status IputDay(int day1, char day[]);/\*\* \*根据客户提出的终点站名输出航线信息 \*/void SearchFlight();/\*\* \* 打印订票乘员名单域的客户名单信息(不包括身份证) \*/void PriCusInfo();/\*\* \* 订票功能模块 \*/void BookTickets();/\*\* \* 退票功能模块 \*/void ReturnTicket();/\*\* \* 搜索界面 \*/void SearchFace();/\*\* \* 订票界面 \* 属于订票模块 \*/void BookticketFace();/\*\* \* 退票模块界面 \*/void ReturnTicketsFace();/\*\* \* 退出程序模块 \*/void GoodbyeFace();/\*\* \* 查询模块 \* 打印全部航班信息 \* @param flight 传入的是航班链表的头指针 \*/void PrintFilghtlist(Flight \*flight);/\*\* \*菜单界面函数 \* @return 返回选择的操作 \*/int MenuSelect();/\*\* \* 管理员界面 \*/void AdministratorFace();/\*\* \*根据客户提出的起点，终点站名输出航线信息 \*/void SearchFlight();/\*\* \* 推荐订票 \* @param startPoint 起点 \* @param destination 终点 \* @param flight 原航班，用来辅助选出同一路线但不同航班ID的航班 \*/void Suggest(char startPoint[], char destination[], Flight \*flight);/\*\* \* 将flight1的4个航班用头插入法插入到pFlight的链表中 \* @param flight1 里面存有六个基本航班 \* @return 返回操作是否成功 \*/Status Create(PFlight flight1);/\*\* \* 推荐同一路线的其他航班 \* @param destination \* @param pflight 原航班，用来判断和新搜到的航班是否一样 \* @return 找不到就返回FALSE，否则返回TRUE \*/Status RecommendFlight(char startPoint[], char destination[], Flight \*pflight);/\*\* \* 管理员插入航班时遍历航班，防止航班ID重复（航班ID相当于主键） \* @param flight \* @param flightCodeID \* @return 返回是否重复 \*/Status TraverseFlight(Flight \*flight, char flightCodeID[]);

**6.main函数**

int main() { //初始化pFlight，pFlight为全局变量。 InitFlight(); //用flight1中的数据创建初始航班链表，里面含有四个节点 Create(flight1); //菜单函数 MenuSelect(); return 0;

}

1. **详细设计**
2. **函数之间的调用关系，如下图所示（如果看不清楚可查看流程图文档）：**



**2.算法设计：**

//定义全局指针变量pFlight，航班链表的头指针

Flight \*pFlight;

//六个基本航班

Flight flight1[6] = {

{"广州", "北京", "1", "B3333", "星期一", 10, 10, 5},

{"广州", "北京", "2", "D5555", "星期二", 8, 8, 4},

{"广州", "北京", "3", "L6666", "星期天", 10, 10, 5},

{"上海", "广州", "4", "K9999", "星期三", 8, 8, 4},

{"广州", "成都", "5", "K7777", "星期三", 10, 10, 5},

{"成都", "广州", "6", "K8888", "星期三", 8, 8, 4}

};

/\*\*

\* 初始化航班链表

\* 目的是要初始化移pFlight为头结点的空航班链表,录入航班信息和增加航班后将航班结点插入该链表

\*/

void InitFlight() {

pFlight = (Flight \*) malloc(sizeof(Flight));//申请头结点的空间

if (pFlight == NULL) exit(0);

pFlight->next = NULL;//将头结点h的指针域置为空

}

/\*\*

\* 将flight1的4个航班用头插入法插入到pFlight的链表中

\* @param flight1 里面存有六个基本航班

\* @return 返回操作是否成功

\*/

Status Create(PFlight flight1) {

Flight \*p = pFlight, \*q;

for (int i; i < 6; i++) {

q = (Flight \*) malloc(sizeof(Flight));

if (q == NULL)

return ERROR;

strcpy(q->startPoint, flight1[i].startPoint);

strcpy(q->destination, flight1[i].destination);

strcpy(q->flightCodeID, flight1[i].flightCodeID);

strcpy(q->planeNum, flight1[i].planeNum);

strcpy(q->day, flight1[i].day);

q->totalTickets = flight1[i].totalTickets;

q->left = flight1[i].totalTickets;

q->leftEconomicTicket = flight1[i].leftEconomicTicket;

q->leftBusinessTicket = flight1[i].totalTickets - flight1[i].leftEconomicTicket;

//初始化乘客链表

InitCusLinkList(q->cusLinkList);

//q->cusLinkList = (CustomerNode \*) malloc(sizeof(CustomerNode));

//q->cusLinkList->next = NULL;

InitQueue(q->waitQueue1);

InitQueue(q->waitQueue2);

q->next = p->next;

p->next = q;

}

return OK;

}

/\*\*

\* 管理员操作

\* 将新的航班结点插入到航班链表中，

\* 采用头插入法

\* @return 返回操作是否成功

\*/

Status InsertFlight() {

FlightNode \*q;//定义q为新增加的航班结点的指针的形参

Flight \*p = pFlight;

int mark = 1;

while (mark != 0) {

q = (FlightNode \*) malloc(sizeof(FlightNode));

if (q == NULL)

return ERROR;

printf("\t\t请依次输入以下内容\n");

printf("\t\t请输入航班ID\n");

scanf("%s", q->flightCodeID);

Status status = TraverseFlight(pFlight, q->flightCodeID);

if (status == ERROR) {

printf("该航班ID已经存在，请重新输入航班ID\n");

continue;

}

printf("\t\t请输入起点站名\n");

scanf("%s", q->startPoint);

printf("\t\t请输入终点站名\n");

scanf("%s", q->destination);

printf("\t\t请输入飞机号\n");

scanf("%s", q->planeNum);

printf("\t\t请输入飞行日期(1代表星期,2代表星期二.....7代表星期日)\n");

int day1;

scanf("%d", &day1);

while (ERROR == IputDay(day1, q->day)) {

printf("请输入合法数据(1-7)\n");

scanf("%d", &day1);

};

printf("\t\t请输入乘客定额\n");

scanf("%d", &q->totalTickets);

q->left = q->totalTickets;

printf("\t\t请输入经济票数目（同时也决定了商务票的数目）\n");

scanf("%d", &q->leftEconomicTicket);

//商务票数目 = 总票数目 - 经济票数目

q->leftBusinessTicket = q->totalTickets - q->leftEconomicTicket;

InitCusLinkList(q->cusLinkList);

//初始化

InitQueue(q->waitQueue1);

InitQueue(q->waitQueue2);

q->next = p->next;

p->next = q;

printf("\t\t是否继续录入航班信息（任意数字继续，0表示停止）。\n");

printf("\t\t请输入：");

scanf("%d", &mark);

}

return OK;

}

/\*\*

\* 管理员操作

\* 删除节点

\* @return 返回操作是否成功

\*/

Status DeleteFlight() {

char flightCodeID[20];

printf("请输入航班ID\n");

scanf("%s", flightCodeID);

PFlight pre = pFlight;

PFlight p = pre->next;

while (p != NULL) {

if (!strcmp(flightCodeID, p->flightCodeID)) {

pre->next = p->next;

free(p);

return OK;

}

pre = p;

p = p->next;

}

return ERROR;

}

/\*\*

\* 验证管理员密码

\* @return 返回操作结果

\*/

Status AdministratorVerify() {

char pwd[20] = "admin";//管理员密码

char password[20];//待输入密码，用来验证

printf("请输入管理员密码:\n");

scanf("%s", password);

if (strcmp(pwd, password)) {

printf("密码错误，请重新输入:\n");

return FALSE;

}

printf("验证成功\n");

return TRUE;

}

/\*\*

\* 管理员的查询操作

\* 打印订票乘员名单域的客户名单信息(不包括身份证)

\*/

void AdminPriCusInfo() {

CusLinkList p;

Flight \*info;

info = find();

if (info == NULL) {

printf("没有这个航班\n");

return;

}

//头结点的下一个节点开始遍历

p = info->cusLinkList->next;

if (p != NULL) {

printf("客户姓名 订票数额 舱位等级（1经济舱，2商务舱） 身份证号码\n");

while (p) {

printf("%s\t\t%d\t\t%d\t\t\t%s\n", p->name, p->clientTickets, p->rank, p->identification);

p = p->next;

}

} else

printf("该航班没有客户信息!!\n");

}

/\*\*

\* 管理员模块

\* 管理员界面函数

\*/

void AdministratorFace() {

int a2;

int loop2;//goto语句

loop2:

{

//system("cls");

printf("\n");

printf("\n");

printf("\n");

printf("\n");

printf(" Welcome To 管理员模块\n");

printf(" ------------------------------------\n");

printf(" | 1.航班信息 |\n");

printf(" | 2.客户信息 |\n");

printf(" | 3.增加航班 |\n");

printf(" | 4.删除航班 |\n");

printf(" | 5.返回上一级 |\n");

printf(" ------------------------------------|\n");

printf(" 请输入您的选择:");

scanf("%d", &a2);

}

switch (a2) {

case 1:

PrintFilghtlist(pFlight);

system("PAUSE");

AdministratorFace();

break;

case 2:

AdminPriCusInfo();

AdministratorFace();

case 3:

InsertFlight();

AdministratorFace();

break;

case 4:

if (OK == DeleteFlight()) {

printf("删除成功\n");

} else {

printf("没有这个航班，删除失败！\n");

};

AdministratorFace();

case 5:

MenuSelect();

break;

default:

goto loop2;

}

}

/\*\*

\* 增加航班时输入日期的辅助函数（1代表星期一，7代表星期日）

\* @param day1 传进来的1-7中的一个

\* @param day 数组类变量，可以返回回去给航班的日期变量

\* @return 返回操作状态，输入是否合法

\*/

Status IputDay(int day1, char day[]) {

switch (day1) {

case Mon:

strcpy(day, "星期一");

break;

case Tues:

strcpy(day, "星期二");

break;

case Wed:

strcpy(day, "星期三");

break;

case Thurs:

strcpy(day, "星期四");

break;

case Fri:

strcpy(day, "星期五");

break;

case Sat:

strcpy(day, "星期六");

break;

case Sun:

strcpy(day, "星期日");

break;

default:

return ERROR;

}

return OK;

}

/\*\*

\*菜单界面函数

\* @return 返回选择的操作

\*/

int MenuSelect() {

int select;

Status status;

flag:

{

system("cls");

printf("\n");

printf(" 18级网络工程4班陈智超 3118005320\n");

printf(" 指导老师：刘添添 ");

printf(" 欢迎来到航空订票系统\n");

printf(" ------------------------------------\n");

printf(" | 1. 查询模块 |\n");

printf(" | 2. 搜索模块 |\n");

printf(" | 3. 订票模块 |\n");

printf(" | 4. 退票模块 |\n");

printf(" | 5. 管理员模块 |\n");

printf(" | 6. 离开系统 |\n");

printf(" ------------------------------------\n");

printf(" 请输入（1-5）否则无效！ 请输入您的选择:");

scanf("%d", &select);

}

switch (select) {

case 1:

//查询所有航班信息

//传入航班链表头指针，打印出所有航班信息

PrintFilghtlist(pFlight);

system("PAUSE");

MenuSelect();

break;

case 2:

//进入搜索模块

SearchFace();

system("PAUSE");

MenuSelect();

break;

case 3:

//进入订票模块

BookticketFace();

system("PAUSE");

MenuSelect();

break;

case 4:

//进入退票模块

ReturnTicketsFace();

system("PAUSE");

MenuSelect();

break;

case 5:

//进入管理员模块

while (1) {

status = AdministratorVerify();

if (TRUE == status)

break;

}

AdministratorFace();

system("PAUSE");

MenuSelect();

break;

case 6:

//退出模块

GoodbyeFace();

break;

default:

goto flag;

}

}

/\*\*

\* 打印info航班的基本信息

\* @param info

\*/

void Display(Flight \*info) {

printf("%8s\t%8s\t%3s\t%s\t%4s\t\t%3d\t%10d\t%10d\t%10d\n", info->startPoint, info->destination, info->flightCodeID,

info->planeNum, info->day,

info->totalTickets, info->left, info->leftEconomicTicket, info->leftBusinessTicket);

}

/\*\*

\* 查询模块

\* 打印全部航班信息

\* @param flight 传入的是航班链表的头指针

\*/

void PrintFilghtlist(Flight \*pflight) {

Flight \*p;

//带头结点的头指针，所以从下一个指针还是遍历

p = pflight->next;

system("cls");

printf("起点站名\t终点站名\t航班ID\t飞机号\t飞行日期\t乘员定额\t余票量\t经济票剩余量\t商务票剩余量\n");

while (p != NULL) {

//调用Display函数打印出每个航班节点的信息

Display(p);

p = p->next;

}

printf("\n\n");

}

/\*\*

\*根据客户提出的起点，终点站名输出航线信息

\*/

void SearchFlight() {

char startPonit[10];

char destination[10];

int flag = 0;

system("cls");

printf("请输入起点站名:");

scanf("%s", startPonit);

printf("请输入终点站名:");

scanf("%s", destination);

struct Flight \*p;

p = pFlight->next;

printf("起点站名\t终点站名\t航班ID\t飞机号\t飞行日期\t乘员定额\t余票量\t经济票剩余量\t商务票剩余量\n");

while (p != NULL) {

if ((strcmp(startPonit, p->startPoint) == 0) && (strcmp(destination, p->destination) == 0) ) {

flag = 1;

Display(p);

}

p = p->next;

}

printf("\n\n");

if (flag == 0)

printf("对不起，该航班未找到! \n");

}

/\*\*

\* 推荐同一路线的其他航班

\* @param destination

\* @param pflight 原航班，用来判断和新搜到的航班是否一样

\* @return 找不到就返回FALSE，否则返回TRUE

\*/

Status RecommendFlight(char startPoint[], char destination[], Flight \*pflight) {

//标记变量，是否找到同一路线的航班

int flag = 0;

system("cls");

struct Flight \*p;

p = pFlight->next;

printf("寻找同一路线的航班\n");

printf("起点站名\t终点站名\t航班ID\t飞机号\t飞行日期\t乘员定额\t余票量\t经济票剩余量\t商务票剩余量\n");

while (p != NULL) {

//路线相同，且不是同一个航班，标记flag = 1，表示找到

if (strcmp(destination, p->destination) == 0 && strcmp(startPoint, p->startPoint) == 0 && p != pflight) {

flag = 1;

Display(p);

}

p = p->next;

}

printf("\n\n");

//没有相同路线的航班

if (flag == 0)

return FALSE;

return TRUE;

}

/\*\*

\* 输出p节点的航班信息

\* @param p 航班节点

\*/

void FlightInfo(Flight \*p) {

printf("起点站名\t终点站名\t航班ID\t飞机号\t飞行日期\t乘员定额\t余票量\t经济票剩余\t商务票剩余\n");

Display(p);

printf("\n\n");

}

/\*\*

\* 管理员插入航班时遍历航班，防止航班ID重复（航班ID相当于主键）

\* @param flight

\* @param flightCodeID

\* @return 返回是否重复

\*/

Status TraverseFlight(Flight \*flight, char flightCodeID[]) {

Flight \*p = flight;

while (p != NULL) {

//当有航班ID重复时候，返回ERROR，

if (!strcmp(flightCodeID, p->flightCodeID)) {

return ERROR;

}

p = p->next;

}

//输入的航班ID不重复

return OK;

}

/\*\*

\*根据自己输入的航班ID查询并以指针形式返回

\* @return 航班指针

\*/

Flight \*find() {

char number[10];

int i = 0;

int loop;

printf("请输入航班ID: ");

scanf("%s", number);

//头结点的下一个节点开始遍历

Flight \*p = pFlight->next;

while (p != NULL) {

if (!strcmp(number, p->flightCodeID))

return p;

p = p->next;

}

return NULL;

}

/\*\*

\* 打印订票乘员名单域的客户名单信息(不包括身份证)

\*/

void PriCusInfo() {

CusLinkList p;

Flight \*info;

info = find();

if (info == NULL) {

printf("没有这个航班\n");

return;

}

//头结点的下一个节点开始遍历

p = info->cusLinkList->next;

if (p != NULL) {

printf("客户姓名 订票数额 舱位等级（1经济舱，2商务舱）\n");

while (p) {

printf("%s\t\t%d\t%d\n", p->name, p->clientTickets, p->rank);

p = p->next;

}

} else

printf("该航班没有客户信息!!\n");

}

/\*\*

\* 订票成功之后，将乘客信息插入到对应航班的订成员名单域中（链表）

\* @param head 乘客名单域头指针

\* @param amount 该乘客订票的数量

\* @param name 乘客的姓名

\* @param rank 订票的等级

\* @return 乘客链表头指针

\*/

CusLinkList insertlink(CusLinkList &head, int amount, char name[], char identification[], int rank) {

//成员名单域新节点new1

CusLinkList new1;

new1 = (CustomerNode \*) malloc(sizeof(CustomerNode));

if (new1 == NULL) {

printf("\n内存不足\n");

return NULL;

}

//将传入乘客信息赋值给new1节点

strcpy(new1->name, name);

strcpy(new1->identification, identification);

new1->clientTickets = amount;

new1->rank = rank;

//头插入法加入成员名单域

new1->next = head->next;

head->next = new1;

// head = new1;

// new1->next =

return head;

}

/\*\*

\* 初始化已订票乘客指针链表

\* @param cusLinkList 航班中乘员链表的头指针

\* @return

\*/

Status InitCusLinkList(CusLinkList &cusLinkList) {

CusLinkList q = cusLinkList;

cusLinkList = (CustomerNode \*) malloc(sizeof(CustomerNode));

cusLinkList->next = NULL;

}

/\*\*

\* 初始化带头结点的链队列

\* @param q

\*/

Status InitQueue(LinkQueue &q) {

WaitQNode \*p;

p = (WaitQNode \*) malloc(sizeof(WaitQNode));

if (p == NULL) {

printf("内存不足\n");

return ERROR;

}

p->next = NULL;

q.front = q.rear = p;

return OK;

}

/\*\*

\*入队，增加排队等候的客户名单域

\* @param q 带头结点的链队列

\* @param name

\* @param amount

\* @return

\*/

LinkQueue Appendqueue(LinkQueue &q, char name[], int amount, char identification[]) {

PWait new1;

new1 = (PWait) malloc(sizeof(WaitQNode));

strcpy(new1->name, name);

strcpy(new1->identification, identification);

new1->preTickets = amount;

new1->next = NULL;

q.rear->next = new1;

q.rear = new1;

return q;

}

/\*\*

\* 出队函数

\* @param Q Q为候补订票客户的队列

\* @param NameAndNumAndIDAndID 候补客户的姓名和订票数目，出队时，将姓名和关键字和身份证返回

\* @return

\*/

Status QueueDelete(LinkQueue &q, NameAndNumAndID &NameAndNumAndID) {

WaitQNode \*p;

p = q.front->next;

//带头结点的链表，当front指针和rear指针相等时，相当于队列为空，没有元素出队

if (q.front == q.rear) {

return ERROR;

}

//出队

q.front->next = p->next;

//出队到空时，rear = front

if (q.front->next == NULL) {

q.rear = q.front;

}

//返回出队元素的关键信息，包括候补客户的姓名和订票量

NameAndNumAndID.num = p->preTickets;

strcpy(NameAndNumAndID.name, p->name);

strcpy(NameAndNumAndID.identification, p->identification);

free(p);

return OK;

}

/\*\*

\* 订票模块

\*/

void BookTickets() {

struct Flight \*info;

int amount, rank;

int tickets;//剩余的商务票数目或者经济票数目

char name[10];

char identification[20];

system("cls");

//调用查找航班函数，返回给

int loop1;

loop1:

{

info = find();

};

if (info == NULL){

printf("没有这个航班, 请重新输入\n");

goto loop1;

}

int loop2;

loop2:

{ printf("请正确输入你订票所需要的数量:");

scanf("%d", &amount);

};

//非法字符，重新输入

if (amount <= 0 ) {

goto loop2;

}

printf("请正确输入您的票的舱位等级（1代表经济舱，2或其他代表商务舱）:");

scanf("%d", &rank);

if (rank == 1)

tickets = info->leftEconomicTicket;

else

tickets = info->leftBusinessTicket;

if (amount <= tickets) {

int i;

printf("请输入您的姓名:");

scanf("%s", name);

printf("请输入您的身份证号码:");

scanf("%s", identification);

CusLinkList head = info->cusLinkList;

//订票成功，插入成员名单链表

insertlink(head, amount, name, identification, rank);

for (i = 0; i < amount; i++)

printf("%s 的座位号是: %d\n", name, info->totalTickets - info->left + i + 1);

info->left -= amount;

if (rank == 1)

info->leftEconomicTicket -= amount;

else

info->leftBusinessTicket -= amount;

printf("\n祝您旅途愉快！欢迎再次光临\n");

} else

loop0:

{

char r;

printf("该等级的票不足，订票失败，以下为该航班乘客信息,希望对您的订票有所帮助\n");

FlightInfo(info);

printf("是否改变订票计划？Y/N\n");

r = getch();

printf("%c", r);

if (r == 'Y' || r == 'y') {//改变计划，重新选择航班

BookTickets();

} else {

printf("\n已经没有更多的票，您需要排队等候吗?(Y/N)");

r = getch();

printf("%c", r);

if (r == 'Y' || r == 'y') {//不改变计划，排队候票

printf("\n请输入您的姓名（排队订票客户）:");

scanf("%s", name);

printf("\n请输入您的身份证（排队订票客户）:");

scanf("%s", identification);

if (rank == 1) {//进入经济舱排队队列

info->waitQueue1 = Appendqueue(info->waitQueue1, name, amount, identification);

} else {//进入商务舱排队队列

info->waitQueue2 = Appendqueue(info->waitQueue2, name, amount, identification);

}

printf("\n排队成功!\n");

} else {//不排队，选择系统提供的其他建议方案

printf("\n是否根据建议订票？若是，则推荐相同的起点和终点的航班Y/N");

r = getch();

printf("%c", r);

printf("%c", r);

if (r == 'Y' || r == 'y') {

//调用推荐函数

Suggest(info->startPoint, info->destination, info);

} else

printf("\n欢迎您下次再次订购！\n");

}

}

}

}

/\*\*

\* 推荐订票

\* @param startPoint 起点

\* @param destination 终点

\* @param flight 原航班，用来辅助选出同一路线但不同航班ID的航班

\*/

void Suggest(char startPoint[], char destination[], Flight \*flight) {

//struct Flight \*info;

//char name[10];

//int amount, rank;

//int i = 0, loop = 0;

//info = pFlight;

Status status = RecommendFlight(startPoint, destination, flight);

if (status == FALSE)

printf("对不起，没有相同起点和终点的航班了");

else {

printf("是否符合订票要求？Y|N");

{

char r;

r = getch();

printf("%c", r);

if (r == 'Y' || r == 'y') {

BookTickets();

system("PAUSE");

} else

printf("不根据建议进行订票，现在将退回主菜单");

system("PAUSE");

MenuSelect();

}

}

}

/\*\*

\* 退票功能模块

\*/

void ReturnTicket() {

struct Flight \*info;

int rank;

//p1为遍历指针，p2为辅助指针，指向p1的前驱

CustomerNode \*p1, \*p2, \*head;

//客户姓名

char cusname[10];

//客户身份证

char identification[20];

system("cls");

//info指针，find函数根据航班ID返回该航班节点的指针

int loop;

loop:

{

info = find();

};

if (info == NULL){

printf("没有这个航班, 请重新输入\n");

goto loop;

}

//head为该航班的的乘员名单域的头指针

head = info->cusLinkList;

//带头结点的指针，head->next 开始遍历

p1 = head->next;

printf("请输入你的姓名: ");

scanf("%s", cusname);

printf("请输入你的身份证号码: ");

scanf("%s", identification);

//根据客户姓名搜索客户是否订票

p2 = head;

while (p1 != NULL) {

if ((strcmp(cusname, p1->name) == 0) && (strcmp(identification, p1->identification) == 0)) break;

p2 = p1;

p1 = p1->next;

}

if (p1 == NULL) {

printf("对不起，你没有订过票或姓名和身份证不对应\n");

return;

} else {//退票成功

//从乘客名单域中移除该节点

rank = p1->rank;

p2->next = p1->next;

//加回该航班的剩余票

info->left += p1->clientTickets;

if (rank == 1) {

info->leftEconomicTicket += p1->clientTickets;

} else {

info->leftBusinessTicket += p1->clientTickets;

}

printf("%s 成功退票！\n", p1->name);

free(p1);

}

LinkQueue queue1 = info->waitQueue1;

LinkQueue queue2 = info->waitQueue2;

NameAndNumAndID nameAndNumAndID = {0, 0};

if (rank == 1) { //有经济舱退票，询问经济舱排队的客户

for (; queue1.front->next != NULL && queue1.front->next->preTickets <= info->leftEconomicTicket;) {

//从候补客户队列中，出队客户的姓名和订票量用NameAndNumAndID返回

QueueDelete(info->waitQueue1, nameAndNumAndID);

int y;

printf("有 经济舱票 剩余 , 尊敬的%s ：\n", nameAndNumAndID.name);

printf("是否确认订票（1确认订票， 其他数字拒绝订票\n");

scanf("%d", &y);

if (y == 1) {

//排队订票成功

for (int i = 0; i < nameAndNumAndID.num; i++)

printf("排队订票成功 %s 的座位号是:%d\n", nameAndNumAndID.name, (info->left) - i);

//剩余票减少

info->left -= nameAndNumAndID.num;

info->leftEconomicTicket -= nameAndNumAndID.num;

//乘员名单链表插入排队订票成功的客户

info->cusLinkList = insertlink(info->cusLinkList, nameAndNumAndID.num, nameAndNumAndID.name,

nameAndNumAndID.identification, rank);

}

}

} else { //有商务舱客户退票，询问商务仓排队的客户

for (; queue2.front->next != NULL && queue2.front->next->preTickets <= info->leftBusinessTicket;) {

//从候补客户队列中，出队客户的姓名和订票量用NameAndNumAndID返回

QueueDelete(info->waitQueue2, nameAndNumAndID);

int y;

printf("有 商务舱票 剩余 , 尊敬的%s ：\n", nameAndNumAndID.name);

printf("是否确认订票（1确认订票， 其他数字拒绝订票\n");

scanf("%d", &y);

if (y == 1) {

//排队订票成功

for (int i = 0; i < nameAndNumAndID.num; i++)

printf("排队订票成功 %s 的座位号是:%d\n", nameAndNumAndID.name, (info->left) - i);

//剩余票减少

info->left -= nameAndNumAndID.num;

info->leftBusinessTicket -= nameAndNumAndID.num;

//乘员名单链表插入排队订票成功的客户

info->cusLinkList = insertlink(info->cusLinkList, nameAndNumAndID.num, nameAndNumAndID.name,

nameAndNumAndID.identification, rank);

}

}

}

}

/\*\*

\* 搜索界面

\*/

void SearchFace() {

int a2;

int loop2;//goto语句

loop2:

{

system("cls");

printf("\n");

printf("\n");

printf("\n");

printf("\n");

printf(" Welcome To 搜索模块\n");

printf(" ------------------------------------\n");

printf(" | 1.查询航班信息 | \n");

printf(" | 2.返回上一层菜单 | \n");

printf(" | 3.退出系统 |\n");

printf(" ------------------------------------|\n");

printf(" 请输入您的选择:");

scanf("%d", &a2);

}

switch (a2) {

case 1:

SearchFlight();

system("PAUSE");

SearchFace();

break;

case 2:

MenuSelect();

break;

case 3:

GoodbyeFace();

break;

default:

goto loop2;

}

}

/\*\*

\* 订票界面

\* 属于订票模块

\*/

void BookticketFace() {

int a3;

int loop3;

loop3:

{

system("cls");

printf("\n");

printf("\n");

printf(" Welcome To 订票模块\n");

printf(" -----------------------------------------------\n");

printf(" | 1.客户订票 |\n");

printf(" | 2.根据起点和终点搜索航班 |\n");

printf(" | 3.查询所有航班 |\n");

printf(" | 4.通过航班ID查询订票客户名单 |\n");

printf(" | 5.返回上一级菜单 |\n");

printf(" ------------------------------------------------\n");

printf(" 请输入您的选择:");

scanf("%d", &a3);

}

switch (a3) {

case 1:

//订票

BookTickets();

system("PAUSE");

BookticketFace();

break;

case 2:

//输入起点和终点查询

SearchFlight();

system("PAUSE");

BookticketFace();

break;

case 3:

PrintFilghtlist(pFlight);

BookticketFace();

break;

case 4:

//为了保密，只能查到客户的姓名和订票量，查不到身份证

PriCusInfo();

system("PAUSE");

BookticketFace();

break;

case 5:

MenuSelect();

break;

default:

goto loop3;

}

}

/\*\*

\* 退票模块界面

\*/

void ReturnTicketsFace() {

int a3;

int loop4;

loop4:

{

system("cls");

printf("\n");

printf("\n");

printf(" Welcome To 退票模块\n");

printf(" -------------------------------------\n");

printf(" | 1.办理退票手续 |\n");

printf(" | 2.返回上一级菜单 |\n");

printf(" | 3.退出系统 |\n");

printf(" --------------------------------------\n");

printf(" 请输入您的选择:");

scanf("%d", &a3);

}

switch (a3) {

case 1:

ReturnTicket();

system("PAUSE");

ReturnTicketsFace();

break;

case 2:

MenuSelect();

break;

case 3:

GoodbyeFace();

break;

default:

goto loop4;

}

}

/\*\*

\* 退出程序模块

\*/

void GoodbyeFace() {

system("cls");

printf("\n");

printf("\n");

printf(" ---------------------------------------------\n");

printf(" | 感谢使用航空航班订票系统 \n");

printf(" | |\n");

printf(" | GoodBye！ \n");

printf(" --------------------------------------------\n");

}

/\*\*

\* pFlight为全局变量，航班链表的头指针

\* 先初始化pFlight航班链表，再把flight1的四个你接班数据插入pFlight链表

\* 调用菜单函数

\*/

int main() {

//初始化pFlight，pFlight为全局变量。

InitFlight();

//用flight1中的数据创建初始航班链表，里面含有四个节点

Create(flight1);

//菜单函数

MenuSelect();

return 0;

}

1. **调试分析**

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O(1)，

删除和查询的时间复杂度为O(n),空间复杂度都为O（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O（1）。

1. 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。
2. 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。
3. 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。
4. 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。
5. 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。
6. 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。
7. 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。
8. 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。
9. 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。
10. **用户说明**

1. 用户运行程序后可以看到菜单。

2. 菜单列出了六个模块，用户可以根据提示输入对应编号进行系统的相关操作。

3. 普通用户可以进行全部航班查询、搜索航班、订票、退票以及查看航班的订票情况，但是，为了保密，对于普通用户，将不能查看到其他用户的身份证号码。

3. 管理员操作需要密码验证（密码为addmin），管理员可以对航班进行增加删除操作，还可以对客户订票情况进行查询，而且管理员能查到用户的身份证号码。

4.退票需要验证航班ID，姓名和身份证，验证成功才能退票。

5.当票数不足时可以选择排队或者根据提示选择其他相同起点和终点的航班，符合要求则可以根据推荐进行订票。

.

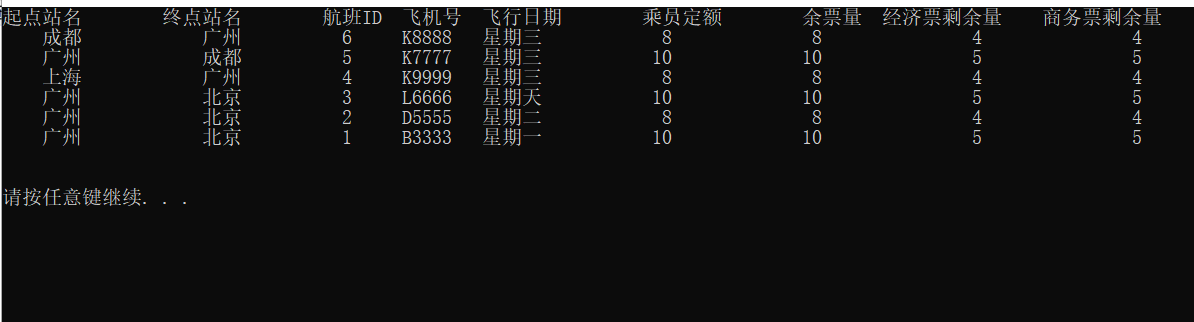
1. **测试结果**

菜单界面：输入1-6选择相应的功能，输入其他数字要求重新输入。



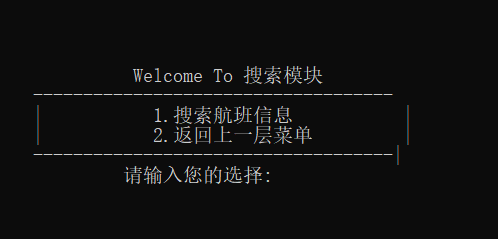
模块一：

从菜单中输入1：调用了PrintFilghtlist函数，进入查询模块，将显示所有航班的相关信息，查看完之后按任意键会返回菜单界面。

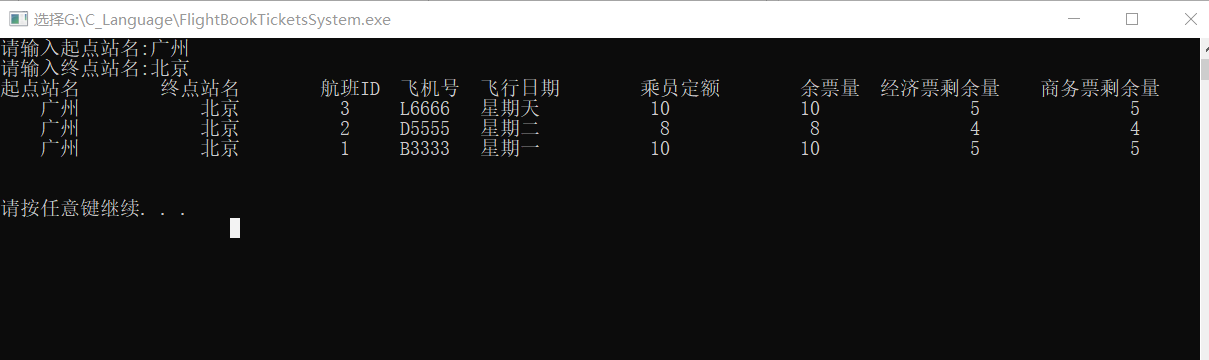


模块二：

从主菜单中输入2：调用了SearchFace函数，进入搜索模块



* 1. 输入1，搜索航班信息，需要输入起点站和终点站名字

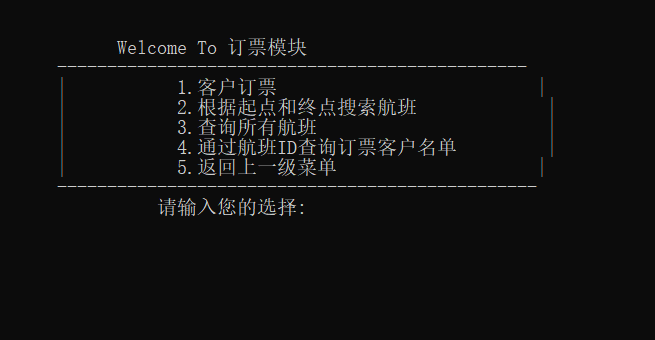


* 1. 输入起点站名和终点站名搜索航班，若是在该航班链表中不存在，则提示该航班未找到。

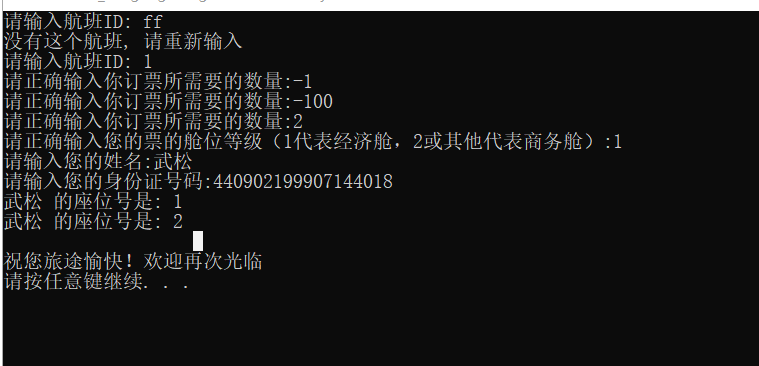


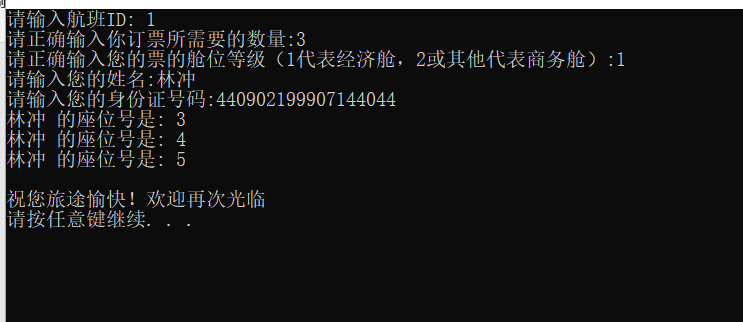
模块三：

从主菜单中输入3，调用BookTicketsFace函数，进入订票模块。

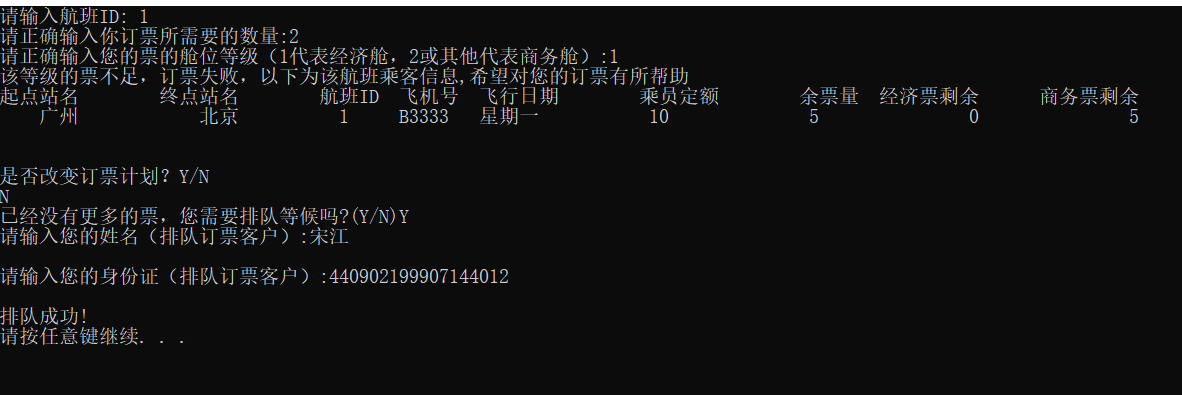


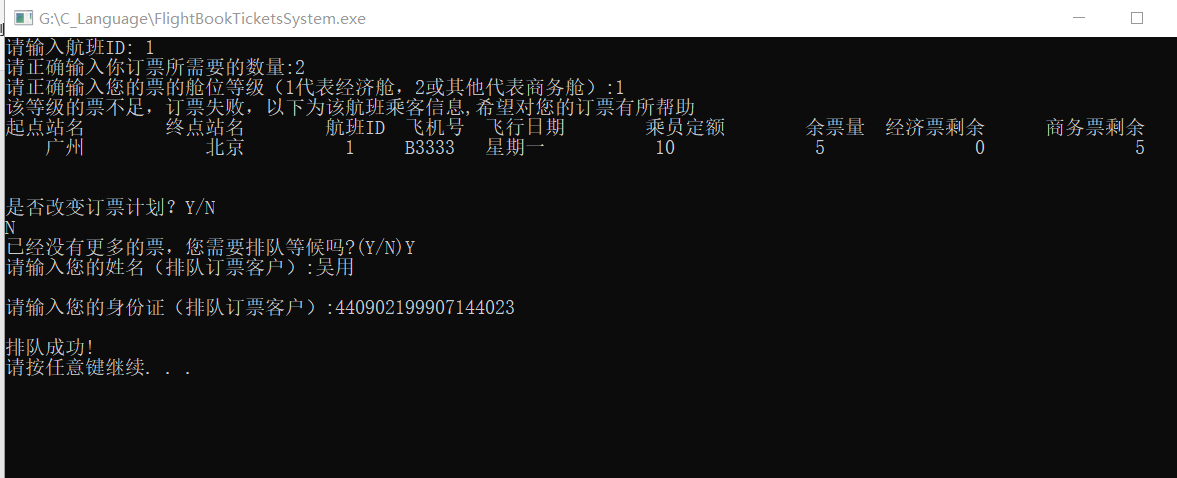
1. 输入1，进行订票，航班ID要正确输入，不存在该航班ID会提示没有这个航班（例如输入ff），则要求重新输入。当航班ID正确时输入订票数目，数目要合法，输入负数要求重新输入（例如下图输入了-1和-100就不行），正确输入票数之后，需要输入舱位等级，此时，如果对应仓位剩余的票数足够，则显示订票成功，并告诉你相应的座位号。



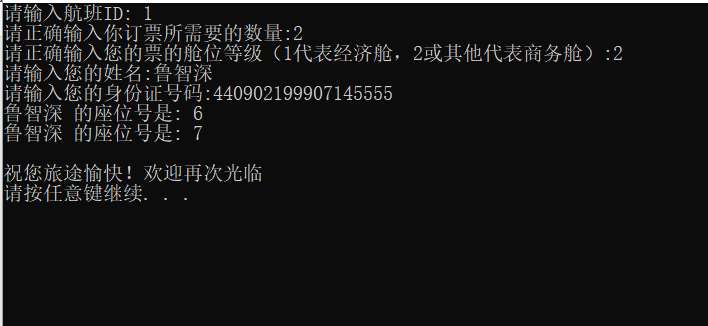


以下是订票失败选择排队的案例：1号航班的经济票已经售完，然后两个客户，宋江和吴用先后再去订1号航班的经济舱的票，订票失败，但是宋江和吴用都选择了排队。



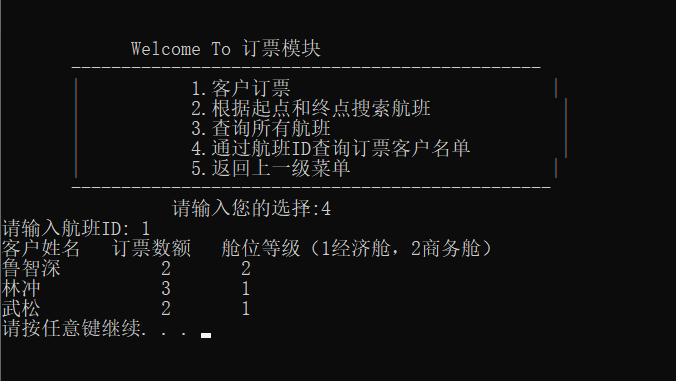


1号的经济舱票数已经为0，但是商务舱票数仍然有剩余，此时，鲁智深前去订1号航班的商务舱的机票，则订票成功。



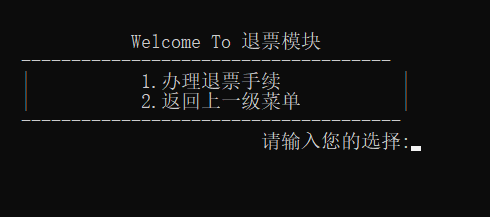
1. 订票模块中输入2和3分别时调用搜索和查询模块，前面已经测试过了，这里不再重复测试。
2. 订票模块输入4，可以通过航班ID查询该航班的订票情况，不显示身份证。

.

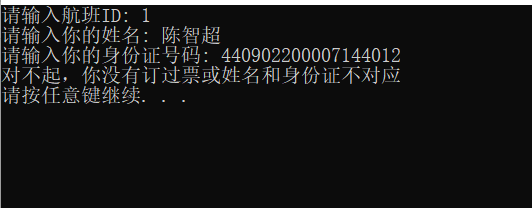


模块四：

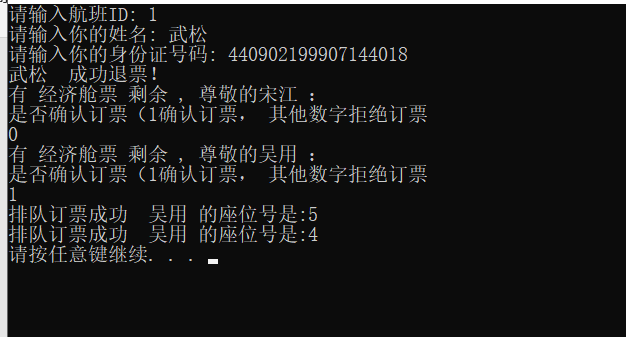
从主菜单中输入4，进入退票模块：

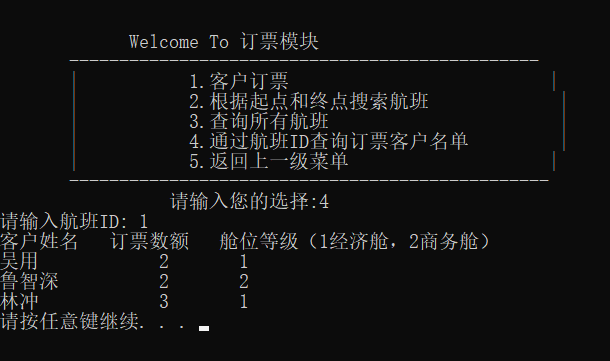


输入1，办理瑞票手续，当没有订票或者输入的航班号、姓名和身份证号码没有一一对应，都将退票失败。

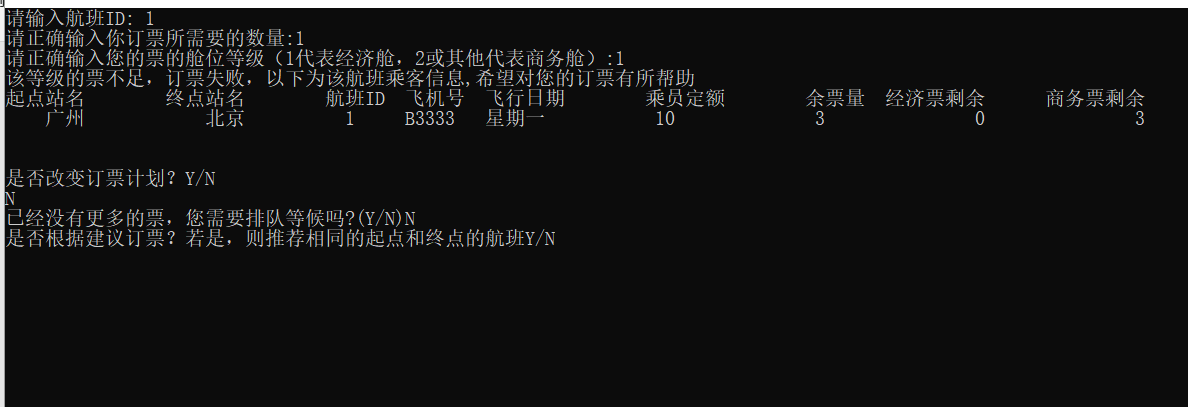


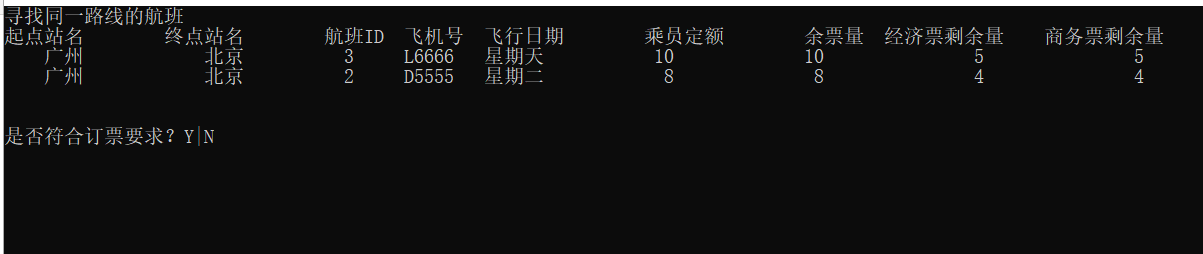
武松订了2张1号航班的经济舱的票，然后，退票，退票成功后，1号航班的经济舱已经有剩余的票数了，则按先后顺序询问正在候补的宋江和吴用，宋江选择拒绝订票，则继续询问吴用是否订票，吴用选择了订票，则吴用排队订票成功。此时，再返回订票模块查看订票客户，可以看到武松已经退票，不在订票客户名单中了，而吴用已经订票成功。



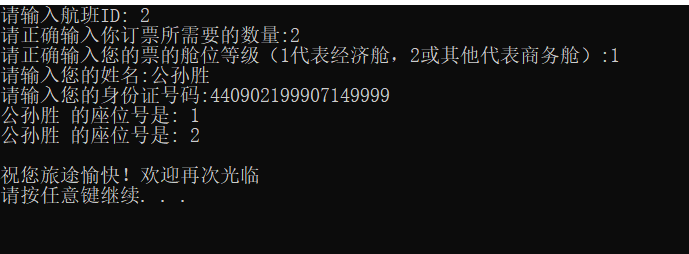


下面测试订票失败，不选择排队而选择推荐航班的例子。



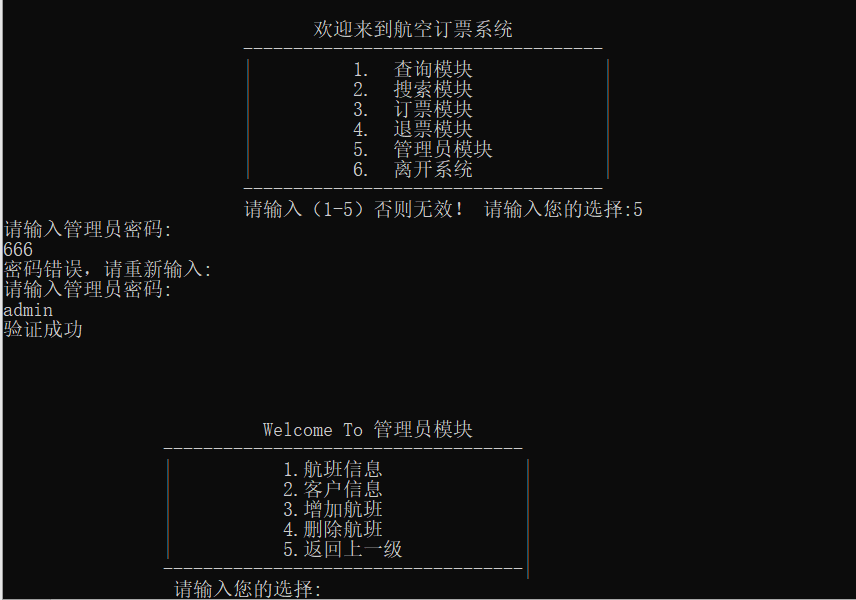


根据推荐，选择2号航班，回到BookTickets函数继续订票。

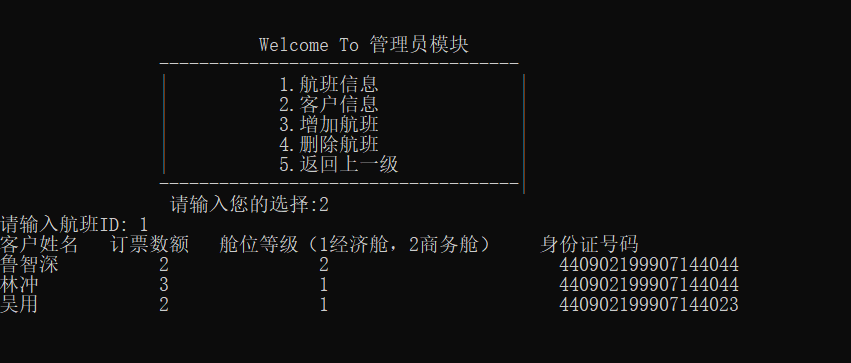


模块五：

从主菜单中输入5：进入管理员模块，先验证密码（admin）

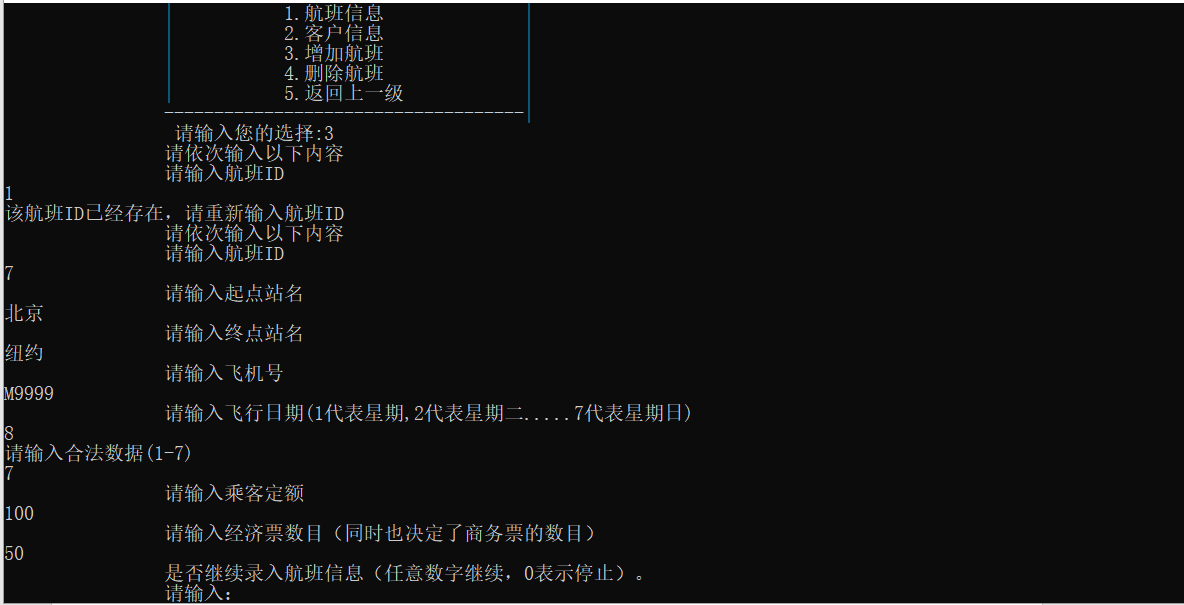


1. 选择1，调用航班查询模块，不在那重复测试。
2. 选择2：通过航班号查询客户信息，能显示身份证号码。

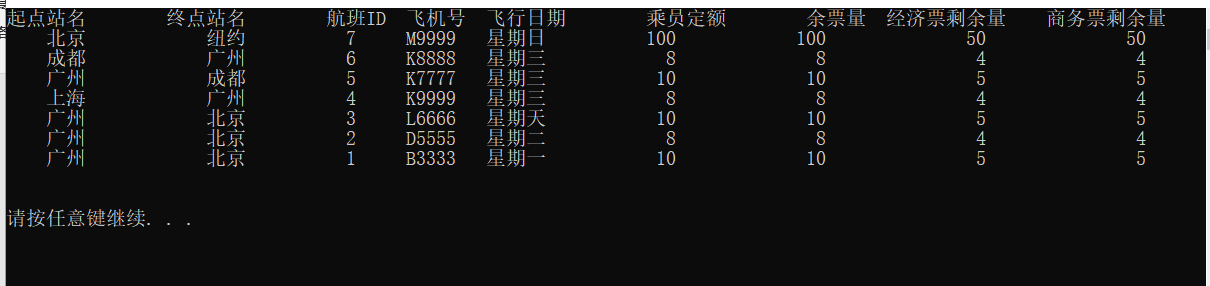


1. 选择3：增加航班，由于主键是航班ID，不能重复，重复要求重新输入。

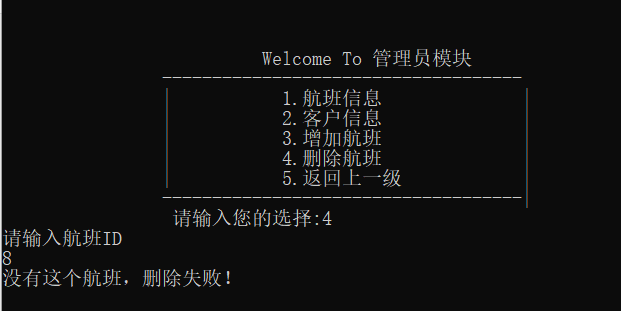
飞行日期要输入（1-7）代表星期一到星期天，否则重新输入

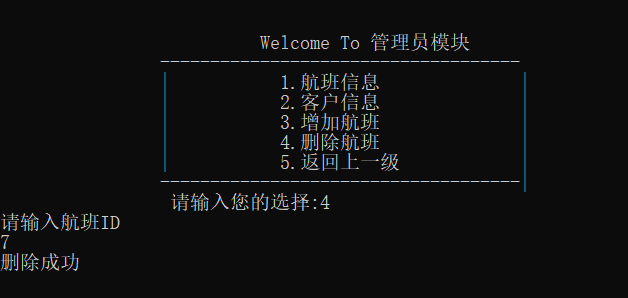


插入成功后，调用查询模块查询，可以看到刚刚添加的7号航班。

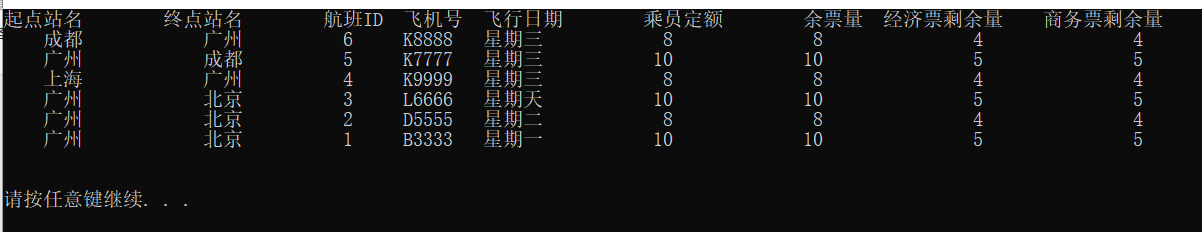


1. 删除航班：如果航班不存在，则删除失败





删成功后查询，可以看到7号航班已经不存在了。



模块六：退出系统



1. **附录**
2. 源代码：FlightBookTicketsSystem.cpp
3. 可执行文件：FlightBookTicketsSystem.exe
4. 课程设计报告电子版：陈智超数据结构课程设计报告.docx
5. 流程图电子版：航空客运订票系统流程图.pdf