



青岛大学
QINGDAO UNIVERSITY

《数据结构实践》课程设计

DoDo 航空购票系统

院系名称: 计算机科学技术学院

专业名称: 软件工程（创新班）

班 级: 20 软创

学生姓名: 李姜楠 马润禾 庞雨亭

指导教师: 仇培铭

目录

1 问题背景.....	1
2 需求分析.....	1
2.1 用户需求分析.....	1
2.2 功能需求分析.....	2
2.2.1 用户功能.....	2
2.2.2 管理员功能.....	2
2.3 系统需求分析.....	2
3 逻辑设计.....	3
3.1 数据结构.....	3
3.2 各功能模块间函数调用关系.....	4
3.3 函数说明.....	5
4 详细设计.....	6
4.1 管理员登陆.....	6
4.2 录入航班信息.....	7
4.3 客户订票模块.....	7
4.4 客户退票模块.....	7
4.5 查询航班模块.....	7
4.6 搜索航班模块.....	8
4.7 查询客户信息模块.....	8
4.8 删除航班模块.....	8
4.9 音乐播放模块.....	8
5 程序调试与测试.....	11
5.1 合法数据测试结果.....	11
5.2 非法数据测试结果.....	15
6 结果分析.....	16
6.1 输入.....	16
6.2 输出.....	17
7 附录：程序源代码.....	18

摘 要

DoDo 航空购票系统是使用 C 语言编程技术在 Dev-C++集成开发环境下开发的订票系统。该系统实现了航班管理、航班信息初始化管理、订票管理、退票管理、查询航班管理员系统等功能。

在航班管理中航空公司内部管理员能在系统中完成添加航班或删除航班、将航班信息初始化并保存、更新航班信息、查看数据库等功能。乘客能够在系统中完成查询航班、订票办理和退票办理等功能。乘客可以用多种方式来查询航班信息，也可以根据实际需要来订票(单程、往返、联程)。整个程序分为十个主要模块，即用户下查询航线、查看所有航班、订购机票、退订机票、候补队列查询，管理员界面下的添加航班、删除航班、查看所有航班、查看航线所有订单与查看航线候补订单。此程序经测试后具有良好的健壮性。

Abstract

DoDo air ticket booking system is developed by using C language programming technology in DevC++ integrated development environment. The system has realized flight management, initialization of flight information, booking management, refund management, query flight administrator system and other functions.

In flight management, airline internal administrators can add or delete flights, initialize and save the flight information, update flight information, view the database and other functions in the system. Passengers can complete the flight query, booking and refund functions in the system. Passengers can check flight information in various ways, or book tickets based on actual needs (one-way, round-trip, connection).

The whole program is divided into ten main modules, that is, the user under the query airline, view all flights, order tickets, cancel tickets, waiting queue query, the administrator interface under the add flight, delete flights, view all flights, view all orders and view airline waiting orders. This program has a good robustness after testing.

1 问题背景

航空订票是整个民航客运业务中一个最基本的业务，虽然它只是民航客运业务中一个非常简单的部分，但因为其涉及资金管理以及客户管理所以使其变得非常重要。通常情况下，人们没有多余的时间去购票、退票、改签等。提前去售票处买票，人多又耽误时间，询问售票人员到目的地的飞机票有哪些，时间是几点，票价是多少，是否还有余票等信息，可能不会问得太详细，这样的流程繁琐且容易出错。这时就需要开发一套具有开放体系结构的、容易扩展和维护的，并且有良好人机交互界面的航空售票系统。作为一个航空公司，拥有一个功能完善的售票系统是很重要的。一个售票系统必须及时地将各个航班的起飞和降落时间准确地反映在系统里，以便公司安排其他的航班。其次，该航班售票系统还能够方便旅客，能够及时地了解各个航班的信息，便于选择合适自己的航班并及时预定机票。尤其是在旅游高峰时期，更能体现拥有一个完善的售票系统的重要性。有了这个系统，公司就能及时地调整航班，最大程度的满足客户的要求，以实现提高公司的信誉度的目的。

2 需求分析

2.1 用户需求分析

用户使用此程序所要完成的工作主要为：录入和查询所有航线信息、查看已订票客户信息、查询航班、办理订票业务、办理退票业务。通过此系统可以方便的进行上述工作。每条航线所涉及的信息有：起终点站名、航班 ID、飞机号、飞行日期、乘员定额、经济舱及商务舱的余票量。已订票的客户名单信息包括姓名、身份证号、订票数额、舱位等级（1，2）。查询航线功能可以根据旅客提出的终点站名输出起终点站名、航班 ID、飞机号、飞行日期、乘员定额、经济舱及商务舱的余票量。

订票业务功能根据客户提出的要求查询该航班票额情况，若尚有余票，则为客户办理订票手续，若已满员或余票额少于定票额，则需重新询问客户要求。若需要，可登记排队候补购票。

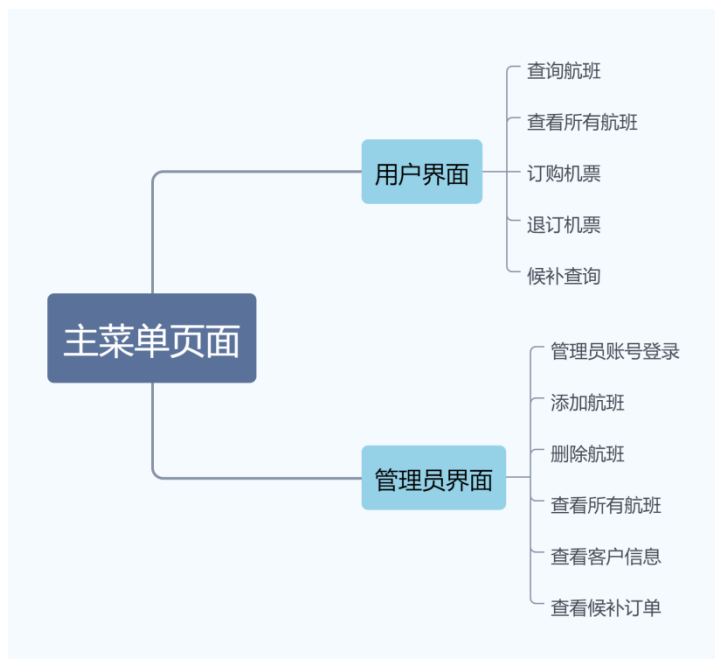
2.2 功能需求分析

2.2.1 用户功能

- (1) 查询所有航班信息
- (2) 搜索所需航班信息
- (3) 输入姓名、身份证号和所要订航班的航班号订票
- (4) 输入姓名、身份证号和已订航班的航班号办理退票

2.2.2 管理员功能

- (1) 查看所有航班信息
- (2) 查看某航班的客户信息
- (3) 增加航班
- (4) 删除航班



2.3 系统需求分析

开发环境：PC 机，Windows 10 系统

运行软件：Dev-C++

3 逻辑设计

3.1 数据结构

1. 航班的信息：航班的情况存储结构采用结构体设计，每个元素表示一个航班的情况，包括航班 ID、飞行日期、起点站名、终点站名、飞机号、成员定额、余票量、经济票剩余量及商务票剩余量九个数据项：

C 语言描述如下：

```
typedef struct Flight { //航班节点
```

```
    char startPoint[20];
    char destination[20];
    char flightCodeID[20];
    char planeNum[20];
    char day[20];
    int totalTickets;
    int left;
    int leftEconomicTicket;
    int leftBusinessTicket;
    Flight *next;
    CusLinkList cusLinkList;
    LinkQueue waitQueue1;
    LinkQueue waitQueue2;
```

```
} Flight, FlightNode, *PFlight;
```

数据项	数据类型
航班 ID	字符串 (char)
飞行日期	字符串 (char)
起点站名	字符串 (char)
终点站名	字符串 (char)
飞机号	字符串 (char)
成员定额	整型 (int)
余票量	整型 (int)
经济票剩余量	整型 (int)
商务票剩余量	整型 (int)

2. 客户的资料：为了便于插入、删除和修改，其采用单链表存储结构，每个数据元素包括姓名、证件号、航班号、订票数量及票的舱位等级四个数据项：

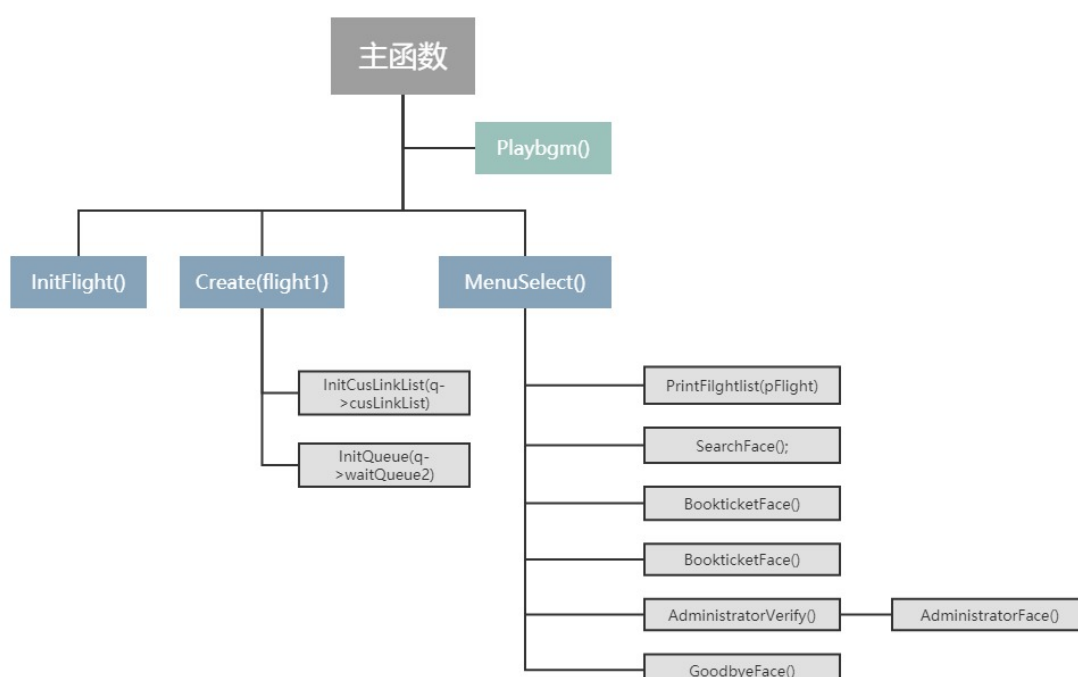
C 语言描述如下：

```
typedef struct CustomerNode { //乘客节点
    char name[10];
    int clientTickets;
    char identification[20];
    int rank;
    CustomerNode *next;
} CustomerNode, *CusLinkList;

typedef struct NameAndNumAndID { //乘客的信息
    char name[10];
    char identification[20];
    int num;
} NameAndNumAndID;
```

数据项	数据类型
证件号	字符（char）
航班号	字符（char）
订票数量	整型（int）
票的舱位等级	整型（int）

3.2 各功能模块间函数调用关系



3.3 函数说明

Flight *find(): 根据航班 ID 进行查询航班信息

Status InitQueue(LinkQueue &q): 初始化链队列

Status InitCusLinkList(CusLinkList &cusLinkList): 初始化订票乘客指针

void Display(struct Flight *info): 打印航班信息

Status IputDay(int day1, char day[]): 日期辅助函数

void Playbgm(): 播放音乐函数

void SearchFlight(): 根据起始站查询航班信息

void PriCusInfo(): 查看已订票的客户名单

void BookTickets(): 订票功能

void ReturnTicket(): 退票功能

void SearchFace(): 搜索界面

void BookticketFace(): 订票界面

void ReturnTicketsFace(): 退票界面

void GoodbyeFace(): 退出程序

void PrintFilghtlist(Flight *flight): 打印航班信息

int MenuSelect(): 操作菜单界面

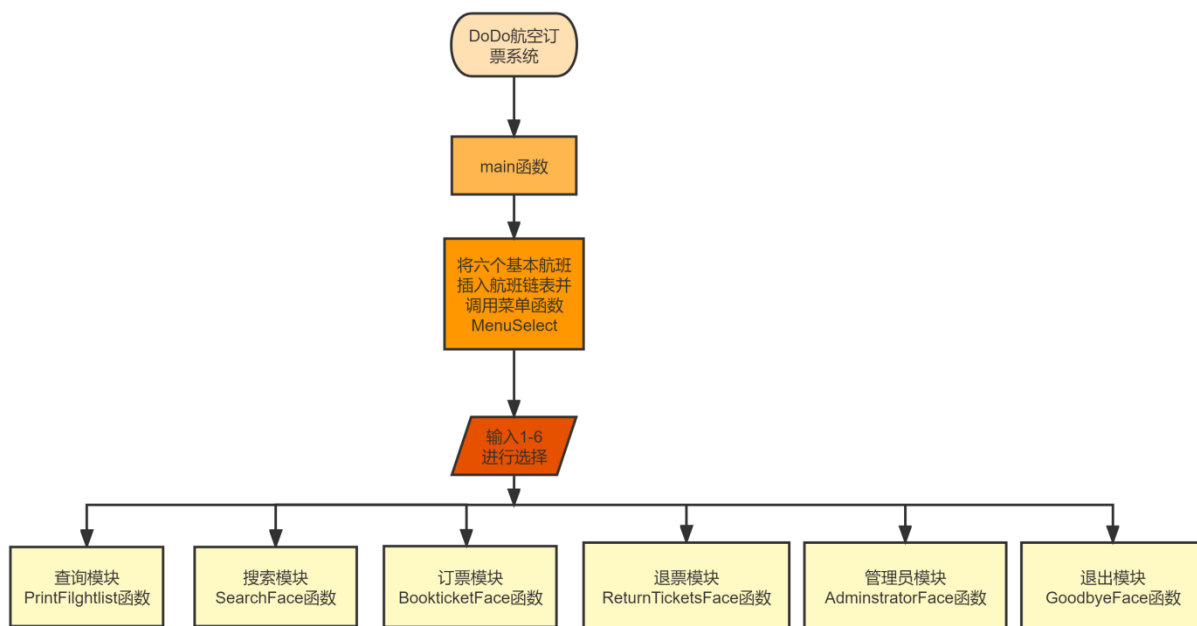
void AdministratorFace(): 管理员界面

void Suggest(char startPoint[], char destination[], Flight *flight): 根据起终点推荐机票

Status Create(PFlight flight1): 初始化航班信息

Status RecommendFlight(char startPoint[], char destination[], Flight *pflight): 推荐同一路线其他航班

Status TraverseFlight(Flight *flight, char flightCodeID[]): 避免航班 ID 重复



4 详细设计

4.1 管理员登陆

在主菜单页面中输入“5”，调用 `status AdministratorVerify()` 函数，输入管理员账号和密码进行验证（账号和密码存在和源代码同文件夹的 `user.txt` 文件中。首先定义 `FILE *fp`，调用 `fopen()` 函数读取 `users.txt` 的内容，`fread()` 从一个文件流中读取数据，最多读取 `count` 个元素，每个元素 `size` 字节，如果调用成功返回实际读取到的元素个数，如果不成功或读到文件末尾返回 0。我们读入一个结构体字符块，写入 `b`。用户输入 ID 与密码后，通过 `strcmp()` 函数进行对比。

值得注意的是，读入密码的同时，我们利用一个星号的输出代替输入的数字，实现了密码的简单加密。若验证通过调用 `void AdministratorFace()` 函数，此函数将打印管理员模块的菜单，随后用户根据自己的需求进行下一步选择。

4.2 录入航班信息

在管理员系统中输入“3”，调用 `Status InsertFlight()` 函数。定义结构指针 *q 为新增加的航班结点的指针的形参，同时创建标记 mark，表示是否要继续录入航班信息，如 `mark != 0`，便会进入 while 循环。在 while 循环中，通过 q 判断结构体指针是否为空，可以进入真正的航班内容输入系统。依照提示输入关于航班的全部信息后，可以自主输入 mark 值，来决定是否要继续输入。如果 `mark==0`，那么将不会进入 while 循环。

4.3 客户订票模块

在主菜单页面中输入“2”，调用 `void BookTickets()` 函数完成客户的订票。在订票模块中，我们首先调用 `find()` 函数返回相应的信息，若返回 NULL，则证明没有这个航班，在询问用户的订票数量、订票等级以及私人信息后，会把这些信息传入相应的变量中，并自动通过（总票数-订票数-余票数+1）这个公式分配座位号。最后我们通过订票数计算出剩余票数，就此完成订票。

4.4 客户退票模块

在主菜单页面中输入“4”，调用 `void ReturnTicket()` 函数完成客户的退票。函数中的两个数组 `cusname[10]` 和 `identification[20]` 用来存客户的姓名和身份证号，指针 *info 可以调用 `find` 函数，通过语句 `info==NULL`，来判断是否存在该航班。用户输入自己的姓名和身份证号后，根据客户姓名搜索，通过 `strcmp()` 函数进行对比，如果该客户已订票，则通过 `info->left += p1->clientTickets;` 语句加回该航班的剩余票，完成退票操作。

4.5 查询航班模块

在主菜单页面中输入“1”，调用 `void PrintFilghtlist(Flight *pflight)` 函数，该函数中定义了一个带头结点的指针 *p，然后开始遍历，调用 `void Display(Flight *info)` 函数打印出每个航班节点的信息。

4.6 搜索航班模块

在主菜单页面中输入“2”，调用 `void SearchFlight()` 函数完成对航班的搜索。函数中的两个数组 `startPonit[10]` 和 `destination[10]` 用来存起飞和抵达城市。提示客户输入相关信息，通过 `strcmp()` 函数进行对比，如果找不到，则输出“没有找到该航班”。定义标记 `flag`，如果路线相同，且不是同一个航班，标记 `flag=1`，表示找到，随机输出所有符合条件的航班信息。

4.7 查询客户信息模块

在管理员系统中输入“2”，调用 `void PriCusInfo()` 函数，此函数创建了 `CusLinkList` 结构体对象 `p`，和指针 `*info`，指针 `*info` 可以调用 `find` 函数；通过语句 `info == NULL`，来判断是否存在该航班。指针 `*p` 指向 `info` 头节点的下一个节点；如果 `p != NULL`，那么将会输出该航班所有的客户信息；反之，则会输出“该航班没有客户信息!!”

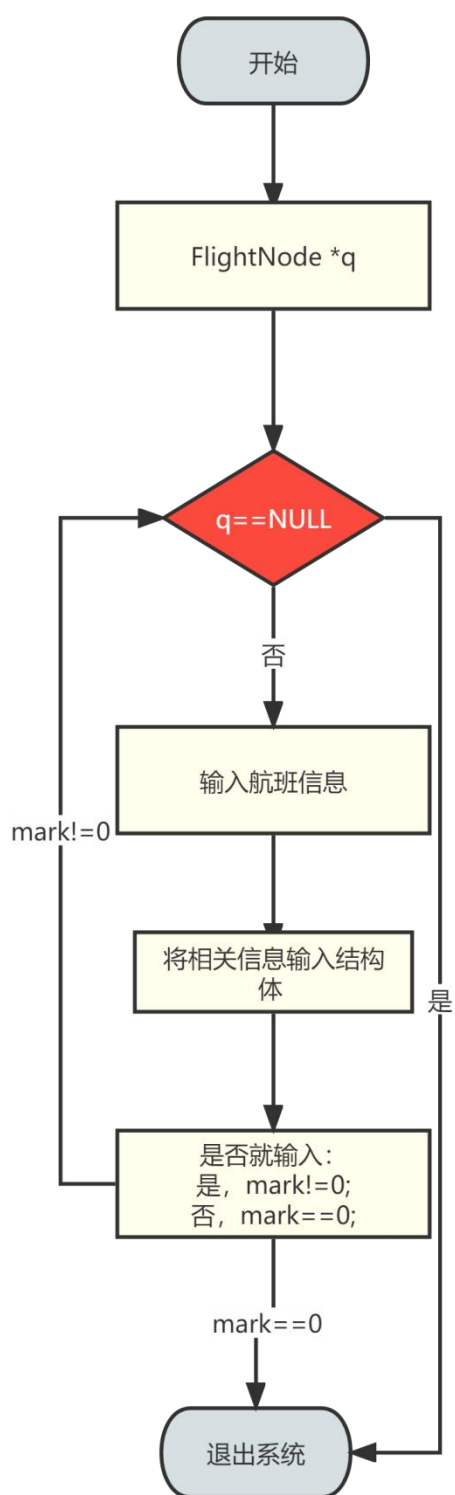
4.8 删除航班模块

在管理员系统中输入“4”，调用 `Status DeleteFlight()` 函数。在函数中首先定义一个长度为 20 的字符型数组用于存放航班 ID。随之创建结构体对象 `pre`，并将全局变量 `pFlight` 赋给 `pre`，使之拥有先前 `pFlight` 变量中的信息。指针 `*p` 指向指针 `*pre` 头节点的下一个节点；通过语句 `p != NULL` 进入 `while` 循环，再通过语句 `!strcmp(flightCodeID, p->flightCodeID)` 进入删除环节，将 `*p` 的下一个节点赋给 `*pre` 的下一个节点，并将 `p` 在内存中占用的空间进行释放；反之，当 `!strcmp(flightCodeID, p->flightCodeID)` 语句没有通过时，循环将继续进行。

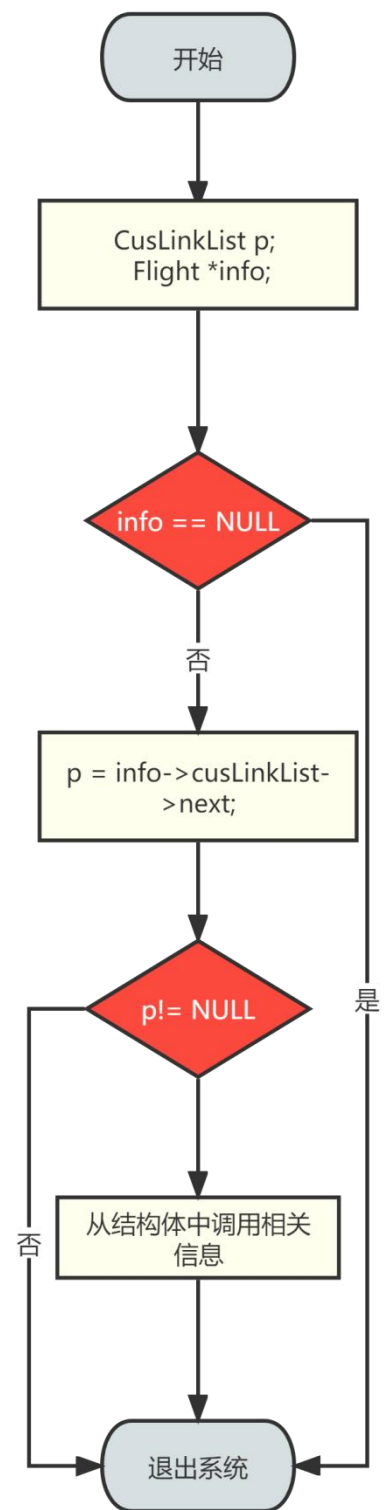
4.9 音乐播放模块

进入 DoDo 航空购票系统后，系统会调用 `Playbgm()` 函数，其中的 `mciSendString` 是用来播放多媒体文件的 API 指令，可以播放 MPEG, AVI, WAV, MP3, 等等，我们使用此语句进行背景音乐的播放。同时我们还增添了订票成功时的提示音，为此采用了 `PlaySound` 语句。`PlaySound` 是 windows 中用来播放声音的 API 函数，这样便可以保证在播放提示音时背景音乐的不中断。

附图:



4.2 录入航班信息函数流程图



4.7 查询客户信息函数流程图

5 程序调试与测试

5.1 合法数据测试结果

(1) 编译连接后显示主菜单页面

20级软创11组
指导老师：仇培铭

欢迎来到DoDo航空售票系统

1.	查询模块
2.	搜索模块
3.	订票模块
4.	退票模块
5.	管理员模块
6.	离开系统

请输入（1-5）否则无效！ 请输入您的选择: _

(2) 输入数字 1，查询系统已有航班

起点站名	终点站名	航班ID	飞机号	飞行日期	乘员定额	余票量	经济票剩余量	商务票剩余量
青岛	南昌	9C7202	波音73F	星期一	275	275	75	200
青岛	南京	H01789	波音738	星期五	350	350	100	250
青岛	哈尔滨	MU5517	波音738	星期二	250	250	50	200
青岛	广州	CZ3800	波音73F	星期一	300	300	100	200
青岛	三亚	SC1151	波音738	星期一	300	300	80	220
青岛	长沙	SC2205	波音738	星期四	285	285	60	225

(3) 输入数字 5，进入管理员模块

请输入（1-5）否则无效！ 请输入您的选择:5

欢迎来到登录界面！
请输入账号：
123
请输入密码
请输入你的登录密码:***

Welcome To 管理员模块

1. 航班信息
2. 客户信息
3. 增加航班
4. 删除航班
5. 返回上一级

请输入您的选择: _

(4) 输入数字 3，进行增加航班操作

```
请输入您的选择:3
请依次输入以下内容
请输入航班ID: 001
请输入起点站名: 青岛
请输入终点站名: 北京
请输入飞机号: QDHK001
请输入飞行日期(1代表星期一, 2代表星期二.....7代表星期日): 1
请输入乘客定额: 120
请输入经济票数目 (同时也决定了商务票的数目): 90
是否继续录入航班信息 (任意数字继续, 0表示停止): 1
请依次输入以下内容
请输入航班ID: _
```

(5) 输入数字 4，进行删除航班操作

```
请输入您的选择:4
请输入航班ID
001
删除成功
```

(6) 输入数字 1，查询所有航班

起点站名	终点站名	航班ID	飞机号	飞行日期	乘员定额	余票量	经济票剩余量	商务票剩余量
青岛	南昌	9C7202	波音73F	星期一	275	275	75	200
青岛	南京	H01789	波音738	星期五	350	350	100	250
青岛	哈尔滨	MU5517	波音738	星期二	250	250	50	200
青岛	广州	CZ3800	波音73F	星期一	300	300	100	200
青岛	三亚	SC1151	波音738	星期一	300	300	80	220
青岛	长沙	SC2205	波音738	星期四	285	285	60	225

请按任意键继续. . . _

(7) 返回主菜单页面，输入 3，进入订票模块

```
Welcome To 订票模块

-----
1. 客户订票
2. 根据起点和终点搜索航班
3. 查询所有航班
4. 返回上一级菜单
-----

请输入您的选择:
```

(8) 输入数字 2，查询青岛到北京的航班信息

```
请输入起点站名:青岛
请输入终点站名:北京
起点站名 终点站名 航班ID 飞机号 飞行日期 乘员定额 余票量 经济票剩余量 商务票剩余量
青岛 北京 001 QDHK001 星期一 120 119 89 30

请按任意键继续. . .
```

(9) 输入数字 3，查询所有航班信息

起点站名	终点站名	航班ID	飞机号	飞行日期	乘员定额	余票量	经济票剩余量	商务票剩余量
青岛	南昌	9C7202	波音73F	星期一	275	275	75	200
青岛	南京	H01789	波音738	星期五	350	350	100	250
青岛	哈尔滨	MU5517	波音738	星期二	250	250	50	200
青岛	广州	CZ3800	波音73F	星期一	300	300	100	200
青岛	三亚	SC1151	波音738	星期一	300	300	80	220
青岛	长沙	SC2205	波音738	星期四	285	285	60	225

请按任意键继续. . .

(10) 输入数字 1，进行订票操作

请输入航班ID: 001
 请正确输入你订票所需要的数量:1
 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):1
 请输入您的姓名:朵拉
 请输入您的身份证号码:3713<-马赛克
 朵拉 的座位号是: 1
 祝您旅途愉快! 欢迎再次光临
 请按任意键继续. . .

(11) 如订票时座位不足

请输入航班ID: 001
 请正确输入你订票所需要的数量:1
 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):1
 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

起点站名	终点站名	航班ID	飞机号	飞行日期	乘员定额	余票量	经济票剩余	商务票剩余
青岛	北京	001	QDHK001	星期一	1	0	0	0

是否改变订票计划? Y/N

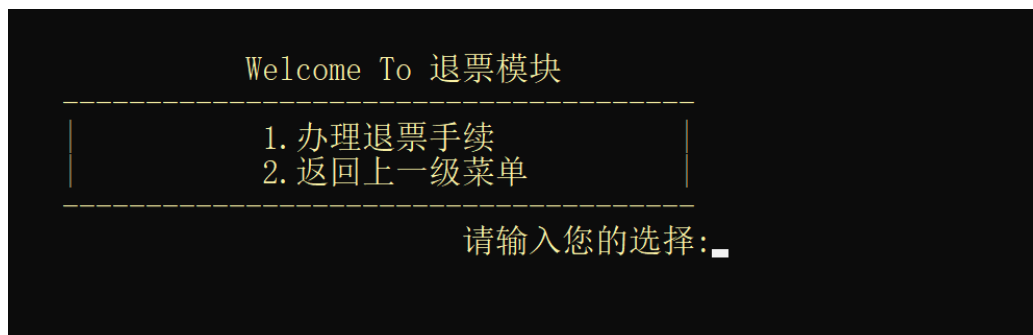
是否改变订票计划? Y/N
 N
 已经没有更多的票, 您需要排队等候吗?(Y/N)N
 是否根据建议订票? 若是, 则推荐相同的起点和终点的航班Y/NNN
 欢迎您下次再次订购!
 请按任意键继续. . .

(12) 返回主菜单页面，输入数字 5，进入管理员模块，再输入数字 2，查询客户信息

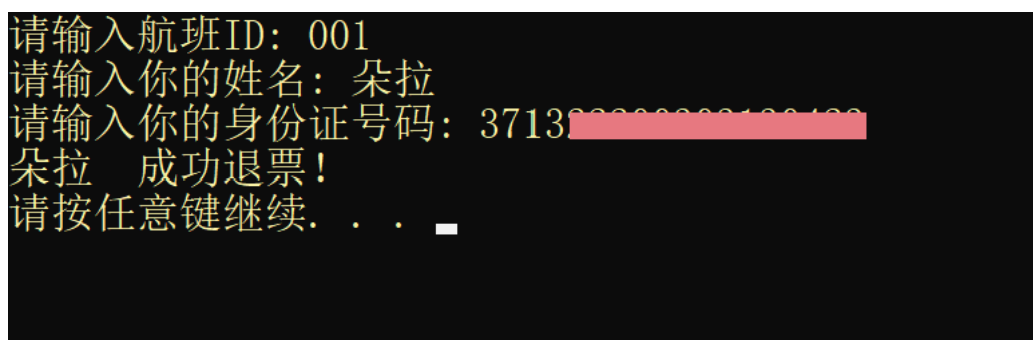
请输入您的选择:2
 请输入航班ID: 001

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
朵拉	1	1	3713<-马赛克

(13) 返回主菜单页面，选择数字 4，进入退票模块



(14) 输入数字 1，根据姓名和身份证号办理退票



(15) 操作完成后，输入数字 6，离开系统



(注：本系统中部分功能实际相同，如主菜单中的 1 查询模块，管理员模块中的 1 航班信息和订票模块中的 3 查询所有航班都是输出所有的航班信息，因此没有一一截图。)

5.2 非法数据测试结果

(1) 选择管理员模块后，登陆时输入的用户名或密码不正确

```
欢迎来到登录界面！
请输入账号：
12
此用户名不存在，请重新输入！
欢迎来到登录界面！
请输入账号：
123
请输入密码：
***
密码不正确！请重新输入密码：
_
```

(2) 选择搜索航班服务后，输入了没有开通航班的城市

```
请输入起点站名:北京
请输入终点站名:上海
起点站名      终点站名      航班ID  飞机号  飞行日期      乘员定额      余票量  经济票剩余量  商务票剩余量

对不起，该航班未找到！
请按任意键继续. . .
```

(3) 选择订票服务后，输入了系统中没有的航班号

```
请输入航班ID: 003
没有这个航班，请重新输入
请输入航班ID:
```


(4) 选择退票服务后，输入了没有订票的客户姓名和证件号或姓名和身份证号不匹配

```
请输入航班ID: 001
请输入你的姓名: 熊大
请输入你的身份证号码: 111111200001010101
对不起，你没有订过票或姓名和身份证不对应
请按任意键继续. . .
```

(5) 选择删除航班服务后，输入了系统中不存在的航班号

```
请输入航班ID
002
没有这个航班，删除失败！
```

(6) 选择客户信息后，输入了系统中不存在的航班号



(7) 选择客户信息后，输入的航班号无一张票售出



6 结果分析

6.1 输入

管理员层面：

- 1、管理员登录：当管理员输入正确的账号和密码时，方可进入管理员系统。
- 2、管理员新增航班：管理员需要输入航班 ID、起点站名、终点站名、飞机号、飞机日期、乘客定额、经济票数；如果还需要继续录入航班信息，要输入判别信息，值为整型（“1”表示继续录入航班信息，“0”表示停止录入航班信息）。
- 3、管理员删除航班：只需要管理员输入目标航班 ID 即可删除该航班。

用户层面：

- 1、用户搜索航班信息：需要用户输入起点站名、终点站名。
- 2、用户订票：需要用户输入目标航班 ID、订票数量、航仓等级、用户姓名、用户身份证号码。
- 3、用户退票：需要用户输入航班 ID、用户姓名以及身份证号。

6.2 输出

在所有操作后的输出中都显示操作是否正确以及操作后单链表的内容。

录入航班信息时，输出显示是否继续添加航班。

查询航班时，输出显示航班信息，或者输出提示信息没有航班信息。

查询客户信息时，输出显示对应航班的客户信息，或者输出没有此航班、该航班没有客户信息。

删除航班时，输出提示信息告知管理员删除是否成功。

搜索航班时，输出显示某起点城市至终点城市的所有航班信息。

客户订票时，输入提示信息告知用户订票是否成功。

客户退票时，输出客户退票成功，或者没有订票、姓名和身份证号不匹配。

总结

经过这次航空售票系统的编写，我们收获很大，也遇到很多问题。书本上的知识与老师的讲解都比较容易理解，但是当自己采用学过的知识点编写程序时却感到十分棘手，有时表现在想不到适合需求的算法，有时表现在算法想出来后，只能将书本上原有的程序段誊写到自己的程序中再加以必要的连接以完成程序的编写。撞了几次壁之后，我们决定静下心来，仔细去写程序，经过多次编写调试，我们可以开始运行自己的程序，也在不断的修改和调试下使我们的系统更加完善，我们也更好地掌握了我掌握了链表的各种应用，提升了对代码逻辑和整体分析的能力。

参考文献

- [1]何钦铭 严晖.C 语言程序设计（第三版）.北京:清华大学出版社，2008
- [2]严蔚敏 .数据结构（第二版）.人民邮电出版社，2016
- [3]许真珍，《c 语言课程设计指导教程》，2016

7 附录：程序源代码

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<conio.h>
#include<windows.h>
#include<Mmsystem.h>
#pragma comment(lib,"Winmm.lib")
#include<math.h>
#include<time.h>
#define OK 1
#define ERROR 0
#define OVERFLOW -1
// #define FALSE -1
#define TRUE 1
typedef int Status;
enum Week {
    Mon = 1, Tues = 2, Wed = 3, Thurs = 4, Fri = 5, Sat = 6, Sun = 7
};
typedef struct CustomerNode { // 乘客节点
    char name[10];
    int clientTickets;
    char identification[20];
    int rank;
    CustomerNode *next;
} CustomerNode, *CusLinkList;
typedef struct WaitPassenger { // 候补队列中的节点
    char name[10];
    char identification[20];
    int preTickets;
    struct WaitPassenger *next;
} WaitQNode, *PWait;
typedef struct Queue { // 候补队列
    PWait front;
    PWait rear;
} LinkQueue;
```

```

typedef struct NameAndNumAndID { //乘客的信息
    char name[10];
    char identification[20];
    int num;
} NameAndNumAndID;
typedef struct Flight { //航班节点
    char startPoint[20];
    char destination[20];
    char flightCodeID[20];
    char planeNum[20];
    char day[20];
    int totalTickets;
    int left;
    int leftEconomicTicket;
    int leftBusinessTicket;
    Flight *next;
    CusLinkList cusLinkList;
    LinkQueue waitQueue1;
    LinkQueue waitQueue2;
} Flight, FlightNode, *PFlight;
typedef struct The_users{
    char ID[11];
    char PWD[20];
}users;
Flight *find();//根据航班 iD 进行查询航班信息
Status InitQueue(LinkQueue &q);//初始化链队列
Status InitCusLinkList(CusLinkList &cusLinkList);//初始化订票乘客指针
void Display(struct Flight *info);//打印航班信息
Status IputDay(int day1, char day[]);//日期辅助函数
void Playbgm();//播放音乐函数
void SearchFlight();//根据起始站查询航班信息
void PriCusInfo();//查看已订票的客户名单
void BookTickets();//订票功能
void ReturnTicket();//退票功能
void SearchFace();//搜索界面
void BookticketFace();//订票界面
void ReturnTicketsFace();//退票界面

```

```

void GoodbyeFace();//退出程序
void PrintFlightlist(Flight *flight);//打印航班信息
void ini();//内嵌航班信息
int MenuSelect();//操作菜单界面
void AdministratorFace();//管理员界面
void Suggest(char startPoint[], char destination[], Flight *flight);//根据起终点推荐机票
Status Create(PFlight flight1);//初始化航班信息
Status RecommendFlight(char startPoint[], char destination[], Flight *pflight);//推荐同一路
线其他航班
Status TraverseFlight(Flight *flight, char flightCodeID[]);//避免航班 ID 重复
Flight *pFlight;
Flight flight1[6] = { };
void ini(){
    strcpy(flight1[5].startPoint, "青岛");
    printf("%s",flight1[5].startPoint);
    strcpy(flight1[5].destination, "南昌");
    strcpy(flight1[5].day, "星期一");
    strcpy(flight1[5].flightCodeID, "9C7202");
    strcpy(flight1[5].planeNum, "波音 73F");
    flight1[5].leftBusinessTicket = 220;
    flight1[5].leftEconomicTicket = 75;
    flight1[5].totalTickets=275;
    strcpy(flight1[4].startPoint, "青岛");
    printf("%s",flight1[4].startPoint);
    strcpy(flight1[4].destination, "南京");
    strcpy(flight1[4].day, "星期五");
    strcpy(flight1[4].flightCodeID, "HO1789");
    strcpy(flight1[4].planeNum, "波音 738");
    flight1[4].leftBusinessTicket = 250;
    flight1[4].leftEconomicTicket = 100;
    flight1[4].totalTickets=350;
    strcpy(flight1[3].startPoint, "青岛");
    printf("%s",flight1[3].startPoint);
    strcpy(flight1[3].destination, "哈尔滨");
    strcpy(flight1[3].day, "星期二");
    strcpy(flight1[3].flightCodeID, "MU5517");
    strcpy(flight1[3].planeNum, "波音 738");
    flight1[3].leftBusinessTicket = 200;

```

```

    flight1[3].leftEconomicTicket = 50;
    flight1[3].totalTickets=250;
    strcpy(flight1[2].startPoint , "青岛");
    printf("%s",flight1[2].startPoint);
    strcpy(flight1[2].destination , "广州");
    strcpy(flight1[2].day , "星期一");
    strcpy(flight1[2].flightCodeID , "CZ3800");
    strcpy(flight1[2].planeNum , "波音 73F");
    flight1[2].leftBusinessTicket = 200;
    flight1[2].leftEconomicTicket = 100;
    flight1[2].totalTickets=300;
    strcpy(flight1[1].startPoint , "青岛");
    printf("%s",flight1[1].startPoint);
    strcpy(flight1[1].destination , "三亚");
    strcpy(flight1[1].day , "星期一");
    strcpy(flight1[1].flightCodeID , "SC1151");
    strcpy(flight1[1].planeNum , "波音 738");
    flight1[1].leftBusinessTicket = 220;
    flight1[1].leftEconomicTicket = 80;
    flight1[1].totalTickets=300;
    strcpy(flight1[0].startPoint , "青岛");
    printf("%s",flight1[0].startPoint);
    strcpy(flight1[0].destination , "长沙");
    strcpy(flight1[0].day , "星期四");
    strcpy(flight1[0].flightCodeID , "SC2205");
    strcpy(flight1[0].planeNum , "波音 738");
    flight1[0].leftBusinessTicket = 220;
    flight1[0].leftEconomicTicket = 60;
    flight1[0].totalTickets=285;
}

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
}

```



```

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);
        InitQueue(q->waitQueue1);
        InitQueue(q->waitQueue2);

        q->next = p->next;
        p->next = q;
    }
    return OK;
}

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\ : ");
        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: ");

```

```

        continue;
    }
    printf("\t\t 请输入起点站名: ");
    scanf("%s", q->startPoint);
    printf("\t\t 请输入终点站名: ");
    scanf("%s", q->destination);
    printf("\t\t 请输入飞机号: ");
    scanf("%s", q->planeNum);
    printf("\t\t 请输入飞行日期(1 代表星期一,2 代表星期二.....7 代表星期日): ");
    int day1;
    scanf("%d", &day1);
    while (ERROR == IputDay(day1, q->day)) {
        printf("请输入合法数据(1-7): ");
        scanf("%d", &day1);
    };
    printf("\t\t 请输入乘客定额: ");
    scanf("%d", &q->totalTickets);
    q->left = q->totalTickets;
    printf("\t\t 请输入经济票数（同时也决定了商务票的数目）: ");
    scanf("%d", &q->leftEconomicTicket);
    //商务票数 = 总票数 - 经济票数
    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）: ");
    scanf("%d", &mark);
}
return OK;
}

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;
}

Status AdministratorVerify() {
    users a,b;//定义结构体 The_users 别名
    FILE *fp;
    printf("欢迎来到登录界面！ \n");
    Sleep(1000);
    fp = fopen("users.txt","r");
    fread(&b, sizeof(struct The_users), 1, fp); //读入一个结构体字符块 写入 b
    printf("请输入账号： \n");
    scanf("%s",&a.ID);
    while (1) {
        if (strcmp(a.ID, b.ID)==0) {    //如果有此用户名
            break;
        }
        else
        {
            if (!feof(fp)) { //如果文件没有读完
                fread(&b, sizeof(struct The_users), 1, fp);
            }
            else {
                printf("此用户名不存在，请重新输入！ \n");
                Sleep(500);
                fclose(fp);
                return false;
            }
        }
    }
    printf("请输入密码： \n");
    int i1=0;
    char c;

```

```

while(1){
    c=getch();
    if(c!=13){
        a.PWD[i1]=c;
        i1++;
        printf("*");
    }
    else{
        break;
    }
}
printf("\n");
do{
    if (strcmp(a.PWD, b.PWD)==0) {           //如果密码匹配*
        fclose(fp);
        system("cls");
        printf("登录成功,欢迎使用!");
        //sfdlcg=1;
        Sleep(500);
        AdministratorFace();
        return true;
    }
    else { printf("密码不正确!请重新输入密码: \n");
        int i1=0;
char c;
        while(1){
            c=getch();
            if(c!=13){
                a.PWD[i1]=c;
                i1++;
                printf("*");
            }
            else{
                break;
            }
        }

        if (strcmp(a.PWD, b.PWD)!=0)
            printf("密码错误! ");
    }
}while(strcmp(a.PWD, b.PWD)==0);

```



```

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:
        PrintFlightlist(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");
            Sleep(1000);
        } else {
            printf("没有这个航班，删除失败！ \n");
            Sleep(1000);
        };
        AdministratorFace();
    case 5:
        MenuSelect();
        break;
    default:
        goto loop2;
}
}
Status InputDay(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;
}

int MenuSelect() {
    int select;
    Status status;
    flag:
    {
        system("cls");
        printf("\n");
        printf("                20 级软创 11 组\n");
        printf("                指导老师： 仇培铭                ");
        printf("\n");
        printf("\n");
        printf("\n");

        printf("\n");
        printf("                欢迎来到 DoDo 航空售票系统\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
    PrintFlightlist(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;
}
}

void Display(Flight *info) {
    printf("%8s\t%8s\t%3s\t%s\t%4s\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);
}

void PrintFlightlist(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");
}

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");
}
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;
}
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");
}
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;
}
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;
}
void PriCusInfo() {
    CusLinkList p;

```

```

Flight *info;
info = find();
if (info == NULL) {
    printf("没有这个航班\n");
    Sleep(1000);
    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);
        Sleep(1000);
        p = p->next;
        Sleep(1000);
    }
} else{
    printf("该航班没有客户信息!!\n");
    Sleep(1000);
}

}

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;
    }
    strcpy(new1->name, name);
    strcpy(new1->identification, identification);

    new1->clientTickets = amount;
    new1->rank = rank;
    new1->next = head->next;
    head->next = new1;
    return head;
}

```

```

}
Status InitCusLinkList(CusLinkList &cusLinkList) {
    CusLinkList q = cusLinkList;
    cusLinkList = (CustomerNode *) malloc(sizeof(CustomerNode));
    cusLinkList->next = NULL;
}
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;
}
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;
}
Status QueueDelete(LinkQueue &q, NameAndNumAndID &NameAndNumAndID) {
    WaitQNode *p;
    p = q.front->next;
    if (q.front == q.rear) {
        return ERROR;
    }
    q.front->next = p->next;
    if (q.front->next == NULL) {
        q.rear = q.front;
    }
}

```

入队，增加排队等候的客户名单域

出队函数

```

NameAndNumAndID.num = p->preTickets;
strcpy(NameAndNumAndID.name, p->name);
strcpy(NameAndNumAndID.identification, p->identification);
free(p);
return OK;
}

void BookTickets() { //订票模块
    struct Flight *info;
    int amount, rank;
    int tickets; //剩余的商务票数或者经济票数
    char name[10];
    char identification[20];
    system("cls");
    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);
    };
    if (amount <= 0 ) { //非法字符, 重新输入
        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");
PlaySound("15275.wav", NULL, SND_ASYNC | SND_NODEFAULT);
Playbgm();
} 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");
    }
}
}
}

void Suggest(char startPoint[], char destination[], Flight *flight) { //推荐订票
    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();
        }
    }
}
}

```



```

}
void ReturnTicket() { //退票功能模块
    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");
    int loop;
    loop:
    {
        info = find();
    };
    if (info == NULL){
        printf("没有这个航班, 请重新输入\n");
        goto loop;
    }
    head = info->cusLinkList;
    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

```

```

rank = p1->rank;
p2->next = p1->next;
info->left += p1->clientTickets;//加回该航班的剩余票
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;) {
        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);
            info->left -= nameAndNumAndID.num;
            info->leftEconomicTicket -= nameAndNumAndID.num;
            info->cusLinkList = insertlink(info->cusLinkList, nameAndNumAndID.num,
nameAndNumAndID.name, nameAndNumAndID.identification, rank);
        }
    }
} else { //有商务舱客户退票, 询问商务舱排队的客户
    for (; queue2.front->next != NULL && queue2.front->next->preTickets <= info->leftBusinessTicket;) {
        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) {
    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);
    info->left -= nameAndNumAndID.num;
    info->leftBusinessTicket -= nameAndNumAndID.num;
    info->cusLinkList = insertlink(info->cusLinkList, nameAndNumAndID.num,
nameAndNumAndID.name,nameAndNumAndID.identification, rank);
    }
}
}
}

void SearchFace() { //搜索界面
    int a2;
    int loop2; //goto 语句
    loop2:
    {
        system("cls");
        printf("\n");
        printf("\n");
        printf("\n");
        printf("\n");
        printf("\n");
        printf("          Welcome To 搜索模块\n");
        printf("          ----- \n");
        printf("      |      1.搜索航班信息      | \n");
        printf("      |      2.返回上一层菜单    | \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;
    default:
        goto loop2;
    }
}

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.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);
            system("PAUSE");
            BookticketFace();

```

```

        break;
    case 4:
        MenuSelect();
        break;
    default:
        goto loop3;
    }
}

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(" ----- \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;

        default:
            goto loop4;
    }
}

void GoodbyeFace() {
    system("cls");

```

```

printf("\n");
printf("\n");
printf("-----\n");
printf("      |      感谢使用 DoDo 航空售票系统      |\n");
printf("      |                                  |\n");
printf("      |      GoodBye!                          |\n");
printf("-----\n");

}

void Playbgm()
{
    mciSendString("play G:\\数据结构\\DoDo 航空购票系统\\bgm.wav",
        NULL,
        0,
        NULL);
}

int main() {
    system("color 0E");
    Playbgm();
    ini();
    InitFlight();
    ini();
    Create(flight1);
    MenuSelect();
    return 0;
}

```