```
#include  <stdio.h>
#include  <Windows.h>
#include <time.h>
#include <stdlib.h>
#include "flight.h"

#define SIZE 50


int g_num = 0; //全局变量，数组里的有效名片数
FLIGHT  flightary[SIZE] ;//全局变量，名片数组


/*对从文件里读出的所有航班信息做特殊排序后进行显示*/
/*
输入: 无 使用全局数组 flightary
返回:无
本函数是否需要修改：否
 */
void displayall()
{
    int i;
    //调用排序函数做特殊排序
    bubblesortflight() ;

    puts("No  航班号  出发城市    到达城市    离港时间    到达时间\t\t 余票额    单
价\n");
    for(i=0;i<g_num;i++)
    {
        displaysingle(i);
    }
    system("pause");
}


/*显示某一条航班信息*/
/*
输入: n，表示是结构体数组中的第几条信息（即下标）
返回:无
本函数是否需要修改：否
 */
void displaysingle(int n)
{
    int temphour=0;
```

```
    printf("%2d                                                    %2d
\t%c\t  %c\t  %2d\t\t",n,flightary[n].fltno,flightary[n].cityfrom,flightary[n].c
ityto,flightary[n].depttime);
    temphour=flightary[n].depttime+flightary[n].flttime;
    if(temphour>24)
        printf("%2d+1day\t\t",temphour%24);
    else
        printf("%2d\t\t",temphour);

    printf("%2d\t   %2d\n",flightary[n].tcktnum,flightary[n].price);
}

/*完成航班信息的特殊排序*/
/*
排序要求说明：按出发城市字母从小到大排序，若出发城市字母相等则进一步按到达城市字
母从小到大排序，
若出发和到达城市都相等则进一步按航班出发时刻由小到大排序，航班出发时刻相等情况不
再进行特殊处理
不限制使用哪种排序算法，但不得使用库函数进行排序
如不能理解以上说明请观察题目输出，或询问监考老师
输入：无，使用全局数组 flightary
返回：无
本函数需要自行编写，并在 displayall 开始执行处被调用，调用指令已经写好
 */
 void bubblesortflight()
  {
       unsigned int pass;
       unsigned int i;
       for (pass = 1; pass < g_num; ++pass) {

          // loop to control number of comparisons per pass
          for (i = 0; i < g_num - 1; ++i) {

             // compare adjacent elements and swap them if first
             // element is greater than second element
              if (flightary[i].cityfrom > flightary[i + 1].cityfrom ) {
                FLIGHT hold = flightary[i];
                flightary[i] = flightary[i + 1];
                flightary[i + 1] = hold;
             }

              if  (flightary[i].cityfrom  ==  flightary[i  +  1].cityfrom  &&
flightary[i].cityto > flightary[i + 1].cityto ) {
                FLIGHT hold = flightary[i];
```

```
                flightary[i] = flightary[i + 1];
                flightary[i + 1] = hold;
            }

             if  (flightary[i].cityfrom  ==  flightary[i  +  1].cityfrom  &&
flightary[i].cityto  ==  flightary[i  +  1].cityto  &&  flightary[i].depttime  >
flightary[i + 1].depttime ) {
                FLIGHT hold = flightary[i];
                flightary[i] = flightary[i + 1];
                flightary[i + 1] = hold;
            }
          }
       }
  }

/*用于根据出发城市和到达城市查询有哪些航班*/
/*
要求用户输入出发城市字母和到达城市字母，然后将满足条件航班信息全部输出，此处不需要考虑排序，仅依据对数组的搜索顺序进行结果显示，不考虑票额是否为0
显示格式：  puts("No  航班号  出发城市  到达城市  离港时间  到达时间\t\t 余票额  单价\n");
输入: 无，使用全局数组 flightary
返回: 无
本函数需要自行编写，注意防错设计
 */
void search()
{
    int i,j;
    char from,to;
    do{
        puts("请输入出发城市<C for Chendu  B for Beijing  X for Xian>: ");
        scanf("%c",&from);
        getchar();
    }while(from != 'C'&& from !='B'&& from !='X');


    do{
        puts("请输入到达城市<C for Chendu  B for Beijing  X for Xian>: ");
        scanf("%c",&to);
        getchar();
    }while(to != 'C' && to !='B'&& to !='X');


    printf("\n");
```

```
    puts("No  航班号  出发城市    到达城市    离港时间    到达时间\t\t 余票额    单价\n");
    for(i=0;i<g_num;i++)
    {
        if(flightary[i].cityfrom==from && flightary[i].cityto==to)
        {
            displaysingle(i);
        }
    }

    system("pause");
}


/*预定机票*/
/*
要求用户输入航班的 2 位编号（注意不是 NO）显示该航班的余票数
并且要求用户输入所需购票数目，
如果购票数大于余票，则显示“购票数大于余票，购票失败”
如果购票数小于等于余票，随后输出购票数，从起飞城市到到达城市，起飞时间和应付的购票总金额
如果未查询到航班号，输出未找到航班号，返回-1
输入: 无，使用全局数组 flightary
返回: 整型：成功返回购票数；失败返回-1
本函数需要自行编写，注意防错设计
 */
int bookticket()
{
    int i=0,flightno=0;
    int booknum = 0;

    do{
        printf("请输入航班编号: ");
        scanf("%d",&flightno);
    }while(flightno <10 || flightno > 99);

    for(i=0;i<g_num;i++)
    {
        if(flightno == flightary[i].fltno && flightary[i].tcktnum > 0)
        {
            printf("目前余票%d，请输入购票数<x x>: ",flightary[i].tcktnum);
            scanf("%d",&booknum);
            if(booknum > flightary[i].tcktnum)
            {
```

```
                printf("购票数目大于余额，本次购票失败\n");
                system("pause");
                return -1;
            }
            else
            {
                printf("您已成功购票%d 张：%c 到%c，起飞时间%d，应付总金额%d\n",booknum,flightary[i].cityfrom,flightary[i].cityto,flightary[i].depttime,booknum*flightary[i].price);
                flightary[i].tcktnum -= booknum;
                system("pause");
                return booknum;
            }
        }
    }

    printf("未查找到该航班号！\n");
    system("pause");
    return -1;
}

/*从文件中读取航班信息*/
/*
以结构体元素为单位从指定文件读取航班信息
输入：数据文件名，该文件已经和代码放置在一起
返回：整型：成功返回 1；失败返回 0
本函数仅需要补充一行代码，请完成 while 中的表达式
 */
int  ReadDataFromFile( const char *filename )
{
    //清空数组
    memset(flightary,0,sizeof(FLIGHT)*SIZE);
    g_num=0;

    FILE   *pfile;         //文件指针
    //打开保存信息的数据文件
    pfile = fopen(filename, "rb" );
    if(pfile == NULL)
    {
        printf("文件打开失败!\n" );
        return 0;
    }

    // 从文件中读取一个成员的信息，放到数组尾部
```

```
    while(fread(&flightary[g_num], sizeof(FLIGHT), 1, pfile) == 1 )
    {
        g_num++;
        if(g_num >= SIZE)
        {
            printf("Reach max storage capability! Stop reading from file!\n");
            break;
        }
    }

    fclose(pfile );
    return 1;

}
```