```
#include  <stdio.h>
#include  <Windows.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


/*将一条数据增加到数组尾部， 不做重复性检查，不排序*/
/*
起飞城市只有 C,B,X,到达城市只有 C,B,X
起飞时间为整点 1~24 点
飞行时长为整数 1~12 小时
输入: 无 使用全局数组 flightary
返回:成功返回 g_num，失败返回-1
本函数需要自行编写，注意放错设计
 */
int adddata2arry()
{
    if(g_num >= SIZE)
    {
        printf("Reach max storage capability! Adding fail!\n");
        return -1;
    }
    do{
        puts("请输入出发城市<C for Chendu  B for Beijing  X for Xian>: ");
        scanf("%c",&flightary[g_num].cityfrom);
        getchar();
    }while(flightary[g_num].cityfrom != 'C' && flightary[g_num].cityfrom !='B'&&
flightary[g_num].cityfrom !='X');


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

    //24 小时制，整点
    do{
```

```
        puts("请输入离港时间 <1 to 24 >: ");
        scanf("%d",&flightary[g_num].depttime);
    }while(flightary[g_num].depttime <1 || flightary[g_num].depttime >24);

    //飞行时长小于等于 12 小时，整点
    do{
        puts("请输入飞行时间 <1 to 12 >: ");
        scanf("%d",&flightary[g_num].flttime);
    }while(flightary[g_num].flttime <1 || flightary[g_num].flttime >12);

    g_num++ ;

    printf("添加成功\n");
    system("pause");
    return g_num;
}

/*根据 NO 编号将一条数据从数组中删除，并将后方数据前移，注意 g_num 修改后，最后一个元素即尾部数据可以不作清空处理*/
/*
1.首先显示所有航班信息
2.要求用户输入需要删除的航班号（整数），航班号范围需要显示出来 0~当前最后一个编号
3.后续航班信息整体前移
输入: 无 使用全局数组 flightary
返回:成功返回 g_num，失败返回-1
本函数需要自行编写，注意放错设计
 */
void deletedatafromarry()
{
    //调用显示所以航班信息及其在数组中的下标号
    displayall();

    //根据显示信息编号，删除信息，后续元素依次上移
    int index=0;
    do{
        printf("输入需要删除的航班号：0~%d\t",g_num-1);
        scanf("%d",&index);
    }while(index <0 || index >g_num-1);

    //删除，即：将后面的元素往前挪一步
    for(int i = index; i<g_num-1; i++)
    {
        flightary[i] =flightary[i + 1]  ;
     }
```

```
     g_num--;
     printf("删除成功\n");
     system("pause");
}

/*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;
    //puts("No\tCityFrom\t   CityTo\tDepartureTime\tFlightTime\n");
    puts("No\t 出发城市\t   到达城市\t 离港时间\t 到达时间\n");
    for(i=0;i<g_num;i++)
    {
        displaysingle(i);
    }
    printf("\n");
    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\t\t%c\t\t%c\t\t%2d\t\t",n,flightary[n].cityfrom,flightary[n].cit
yto,flightary[n].depttime);
    temphour=flightary[n].depttime+flightary[n].flttime;
    if(temphour>24)
        printf("%2d+1day\n",temphour%24);
    else
        printf("%2d\n",temphour);
}

/*根据出发城市和到达城市查询航班,无需排序,请将查询到的直飞航班和经停航班分别显示
出来*/
```

```
/*
1.读入出发城市读入到达城市
2.查询后先显示全部直飞航线信息，直飞航班显示格式请参考或调用 displaysingle（）
3.显示全部经停航线信息,经停航线显示格式如下 puts("No\t 出发城市  经停城市  到达城市\t 离港时间\t 到达时间\n");
其中 NO 包括了出发（首发）城市的航班 NO 和经停起飞航班的 NO，离港时间为首发航班离港时间，到达时间为经停航班的到达时间
4.如果没有航班则不显示航班信息
输入: 无 使用全局数组 flightary
返回:无
本函数需要自行编写，注意放错设计
 */
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 !='c' &&from!='b' && from !='B'&& from !='X' && 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 != 'c' && to!='b' && to !='B'&& to !='X' && to != 'x');

    //搜索直飞航线
    printf("直飞航线：\n");
    puts("No\t 出发城市\t   到达城市\t 离港时间\t 到达时间\n");
    for(i=0;i<g_num;i++)
    {
        if(flightary[i].cityfrom==from && flightary[i].cityto==to)
        {
            displaysingle(i);
        }

    }
    printf("经停航线：\n");
    puts("No\t 出发城市  经停城市  到达城市\t 离港时间\t 到达时间\n");
    for(i=0;i<g_num;i++)
```

```
    {
        //搜索经停
        if(flightary[i].cityfrom==from && flightary[i].cityto!=to)
        {
            for(j=0;j<g_num;j++)
            {
                if(flightary[j].cityfrom==flightary[i].cityto                    &&
flightary[j].cityto==to)
                {

    if(flightary[i].depttime+flightary[i].flttime+1<=flightary[j].depttime)
                    {

    //printf("%2d\t%c\t%c\t%2d\t%2d--",i,flightary[i].cityfrom,flightary[i].city
to,flightary[i].depttime,flightary[i].flttime);

    //printf("%2d\t%c\t%c\t%2d\t%2d\n",j,flightary[j].cityfrom,flightary[j].city
to,flightary[j].depttime,flightary[j].flttime);

                        int temphour=0;

    printf("%2d-%2d\t\t%c\t%c\t%c\t\t%2d\t\t",i,j,flightary[i].cityfrom,flightar
y[i].cityto,flightary[j].cityto,flightary[i].depttime);
                        temphour=flightary[j].depttime+flightary[j].flttime;
                        if(temphour>24)
                            printf("%2d+1day\n",temphour%24);
                        else
                            printf("%2d\n",temphour);
                    }
                }
            }
        }
    }
    system("pause");
}

/*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;

}
```