**1 停车场管理**

**1.1、问题描述**

设停车场是一个可停放 n 辆汽车的狭长通道，且只有一个大门可供汽车进出。汽车在停车场内按车辆到达时间的先后顺序，依次由北向南排列（大门在最南端，最先到达的第一辆车停放在车场的最北端）。若停车场内已经停满 n辆车，那么后来的车只能在门外的便道上等候。一旦有车开走，则排在便道上的第一辆车即可开入。当停车场内某辆车要离开时，在它之后进入的车辆必须先退出车场为它让路，待该辆车开出大门外，其他车辆再按原次序进入车场。每辆停放在车场的车在它离开停车场时必须按它停留的时间长短缴纳费用。试为停车场编制按上述要求进行管理的模拟程序。

使用语言：C语言

编译环境：Codeblocks

**1.2、需求分析**

程序以栈模拟停车场，以队列模拟车场外的便道，按照从终端读入数据的序列进行模拟管理。栈以顺序结构实现，队列以链表结构实现。同时另设一个栈，临时停放为给要离去的汽车让路而从停车场退出来的汽车。

**一个完整的停车场管理系统应该具有以下的功能：**

1. **车辆到达停车场：**从终端读入车牌，建立堆栈（如果文件中保存有上一次运行时，车辆进入停车场的记录，自动写入堆栈）；
2. **车辆离开停车场：**设置另一个栈，用于临时停放为给要离去的汽车让路而从停车场退出来的车辆。离开的车辆的信息保存到停车场使用记录leavedcar.txt文件中。如若候车道上有车，随机把候车队的队头车辆驶入停车场。
3. **查询停车场的状况：**在终端上显示停车场堆栈的车牌，或者显示候车道上排队的车辆牌号。
4. **查询停车场的使用记录：**从leavedcar.txt文件中读取已经离开停车场的车辆的详细信息显示在终端上。
5. **保存信息：**保存停车场内的车辆信息到carlist.txt文件中，保存便道上的车辆的信息到waitqueue.txt文件中。

**1.3、概要设计**

1. 主菜单：用于选择功能；
2. 建立堆栈：文件读入或者终端输入车辆来建立停车堆栈；
3. 建立队列：文件读入或者终端输入车辆基本信息，根据车位是否占满，来建立候车进入停车场队列；
4. 遍历：将停车场以及候车道上的车辆的车牌等信息打印到屏幕上；
5. 保存：将停车场的车辆、候车道上的车辆、离开停车场的车辆的信息写入文件保存。

**1.4、流程图**



**1.5、详细设计**

#include<stdio.h>

#include<stdlib.h >

#include<conio.h >

#include<time.h>

#define NUMB 4

#define MONEY 1

struct tim

{

int sec;

int min;

int hour;

int day;

int mon;

int year;

};

typedef struct stack{

int car\_id;

struct tim arivaltime;

struct tim leavetime;

time\_t start;

time\_t end;

double costtime;

int cost;

struct stack \*next;

}management,\*AD;

typedef struct queue{

int car\_id;

struct queue \*next;

}car\_queue,\*CAR;

void mainmeun()/\*主菜单\*/

{

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

printf("\n\t\t\t\t\t1-申请入库\n\t\t\t\t\t2-车辆出库\n\t\t\t\t\t3-查询停车场状况\n\t\t\t\t\t4-查询停车场使用历史记录\n\t\t\t\t\t0-退出系统\n");

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

}

AD carplace\_initiate()/\*申请车位空间\*/

{

AD newcar;

newcar=(AD)malloc(sizeof(management));

if(newcar==NULL)

{

printf("申请空间失败！");

return NULL;

}

newcar->next=NULL;

return newcar;

}

void input(AD \*cp)

{

FILE \*fp=NULL;

AD pp=NULL;

if((fp=fopen("carlist.txt","r"))==NULL)

{

printf("车辆写入失败！\n");

return;

}

pp=carplace\_initiate();

while(fscanf(fp,"%d",&pp->car\_id)!=EOF)

{

fscanf(fp,"%d%d%d%d%d%d%ld%ld",&pp->arivaltime.year,&pp->arivaltime.mon,&pp->arivaltime.day,&pp->arivaltime.hour,&pp->arivaltime.min,&pp->arivaltime.sec,&pp->start,&pp->end);

if((\*cp)==NULL)

(\*cp)=pp;

else

{

pp->next=(\*cp);

(\*cp)=pp;

}

pp=carplace\_initiate();

}

if(fclose(fp))

printf("关闭文件失败！\n");

}

void save(AD cp)

{

FILE \*fp=NULL;

fp=fopen("carlist.txt","w");

while(cp)

{

fprintf (fp,"%d %d %d %d %d %d %d %ld %ld\n",cp->car\_id,cp->arivaltime.year,cp->arivaltime.mon,cp->arivaltime.day,cp->arivaltime.hour,cp->arivaltime.min,cp->arivaltime.sec,cp->start,cp->end);

cp=cp->next;

}

if(fclose(fp))

printf("关闭文件失败！\n");

return;

}

void savehistory(AD cp)

{

FILE \*f=NULL;

f=fopen("leavedcar.txt","a");

if(f==NULL)

{

printf("保存车辆信息失败！\n");

return;

}

fprintf(f,"%d %d %d %d %d %d %d %d %d %d %d %d %d %.0f %d\n",cp->car\_id,cp->arivaltime.year,cp->arivaltime.mon,cp->arivaltime.day,cp->arivaltime.hour,cp->arivaltime.min,cp->arivaltime.sec,

cp->leavetime.year,cp->leavetime.mon,cp->leavetime.day,cp->leavetime.hour,cp->leavetime.min,cp->leavetime.sec,cp->costtime,cp->cost);

if(fclose(f))

printf("关闭文件失败！\n");

return;

}

void readhistory(AD \*cp)

{

FILE \*f=NULL;

AD pp=NULL;

if((fp=fopen("carlist.txt","r"))==NULL)

{

return;

}

pp=carplace\_initiate();

while(fscanf(f,"%d",&pp->car\_id)!=EOF)

{

fscanf(f,"%d%d%d%d%d%d%d%d%d%d%d%d%lf%d",&pp->arivaltime.year,&pp->arivaltime.mon,&pp->arivaltime.day,&pp->arivaltime.hour,&pp->arivaltime.min,&pp->arivaltime.sec,

&pp->leavetime.year,&pp->leavetime.mon,&pp->leavetime.day,&pp->leavetime.hour,&pp->leavetime.min,&pp->leavetime.sec,&pp->costtime,&pp->cost);

if((\*cp)==NULL)

(\*cp)=pp;

else

{

pp->next=(\*cp);

(\*cp)=pp;

}

pp=carplace\_initiate();

}

if(fclose(f))

printf("关闭车辆历史记录文件失败！\n");

}

CAR queue\_initiate();

CAR car\_insert(CAR);

void inputqueue(CAR \*cq)/\*读入在停车场的车辆信息\*/

{

FILE \*fpt=NULL;

CAR qq=NULL,p=NULL;

if((fpt=fopen("waitlist.txt","r"))==NULL)

{

printf("车辆写入失败！\n");

return;

}

qq=queue\_initiate();

while(fscanf(fpt,"%d",&qq->car\_id)!=EOF)

{

if((\*cq)==NULL)

(\*cq)=qq;

else

{

p=car\_insert(\*cq);

p->next=qq;

}

qq=queue\_initiate();

}

if(fclose(fpt))

printf("关闭文件失败！\n");

}

void savecar(CAR cq)/\*保存候车道车辆信息\*/

{

FILE \*fpt=NULL;

fpt=fopen("waitlist.txt","w");

while(cq)

{

fprintf (fpt,"%d ",cq->car\_id);

cq=cq->next;

}

if(fclose(fpt))

printf("关闭文件失败！\n");

return;

}

car\_queue \*queue\_initiate()/\*申请等候车位\*/

{

car\_queue \*newcar;

newcar=(car\_queue \*)malloc(sizeof(car\_queue));

if(newcar==NULL)

{

printf("申请空间失败！");

return NULL;

}

newcar->next=NULL;

return newcar;

}

void getatime(AD \*cp)/\*获取到达时间\*/

{

time\_t rawtime;

struct tm \* timeinfo;

time(&(\*cp)->start);

(\*cp)->end=0;

time ( &rawtime );

timeinfo = localtime ( &rawtime );

(\*cp)->arivaltime.year=1900+timeinfo->tm\_year;

(\*cp)->arivaltime.mon=1+timeinfo->tm\_mon;

(\*cp)->arivaltime.day=timeinfo->tm\_mday;

(\*cp)->arivaltime.hour=timeinfo->tm\_hour;

(\*cp)->arivaltime.min=timeinfo->tm\_min;

(\*cp)->arivaltime.sec=timeinfo->tm\_sec;

}

void getltime(AD \*cp)/\*获取车辆离开时间\*/

{

time\_t rawtime;

struct tm \* timeinfo;

time(&(\*cp)->end);

(\*cp)->costtime=difftime((\*cp)->end,(\*cp)->start);

time ( &rawtime );

timeinfo = localtime ( &rawtime );

(\*cp)->leavetime.year=1900+timeinfo->tm\_year;

(\*cp)->leavetime.mon=1+timeinfo->tm\_mon;

(\*cp)->leavetime.day=timeinfo->tm\_mday;

(\*cp)->leavetime.hour=timeinfo->tm\_hour;

(\*cp)->leavetime.min=timeinfo->tm\_min;

(\*cp)->leavetime.sec=timeinfo->tm\_sec;

}

int count(management \*cp)/\*计算进库数量\*/

{

if(cp==NULL)

return 0;

management \*s;

int sum=0;

s=cp;

while(s)

{

s=s->next;

sum++;

}

return sum;

}

int countqueue(CAR cq)/\*计算候车队数量\*/

{

if(cq==NULL)

return 0;

CAR s;

int sum=0;

s=cq;

while(s)

{

s=s->next;

sum++;

}

return sum;

}

car\_queue \*car\_insert(car\_queue \*c)/\*队列找队尾\*/

{

while(c->next)

c=c->next;

return c;

}

int check(AD cp,int n)/\*查重\*/

{

AD p;

int s=1;

p=cp;

while(p&&s)

{

if(p->car\_id==n)

s=0;

else

p=p->next;

}

return s;

}

int check(CAR cq,int n)

{

CAR p;

int s=1;

p=cq;

while(p&&s)

{

if(p->car\_id==n)

s=0;

else

p=p->next;

}

return s;

}

AD push\_car(AD cp,CAR \*cq)/\*车库入库\*/

{

system("cls");

AD car\_place=NULL;

int id,n=1;

CAR newwait=NULL,p=NULL;

if(count(cp)<NUMB)

{

printf("输入车牌号码：\n");

scanf("%d",&id);

n=check(cp,id);

while(!n)

{

printf("该车已在停车场内！请重新输入（输入 0 返回主菜单）： ");

scanf("%d",&id);

if(id==0)

return cp;

n=check(cp,id);

}

car\_place=(AD)malloc(sizeof(management));

car\_place->next=NULL;

car\_place->car\_id=id;

getatime(&car\_place);/\*获取时间\*/

printf("\n成功进入停车场。车场计费为每秒一块钱，请注意停车时间\n");

printf ( "进场时间为 %4d-%02d-%02d %02d:%02d:%02d\n\n",car\_place->arivaltime.year, car\_place->arivaltime.mon,car\_place->arivaltime.day,car\_place->arivaltime.hour,car\_place->arivaltime.min,car\_place->arivaltime.sec);

if(cp==NULL)

cp=car\_place;

else

{

car\_place->next=cp;

cp=car\_place;

}

}

else

{

printf("\n车库已满，请在等候区等候。\n");

printf("请输入车牌号（输入 0 返回主菜单）\n");

scanf("%d",&id);

if(id==0)

return cp;

n=check(cp,id);

while(!n)

{

printf("该车已在停车场内！请重新输入（输入 0 返回主菜单）： ");

scanf("%d",&id);

if(id==0)

return cp;

n=check(cp,id);

}

n=1;

n=check(\*cq,id);

while(!n)

{

printf("该车已在候车道内！请重新输入（输入 0 返回主菜单）： ");

scanf("%d",&id);

if(id==0)

return cp;

n=check(\*cq,id);

}

newwait=queue\_initiate();

newwait->car\_id=id;

newwait->next=NULL;

if((\*cq)==NULL)

(\*cq)=newwait;

else

{

p=car\_insert(\*cq);

p->next=newwait;

}

printf("成功开入等候进入停车场车队，请耐心等候！\n");

}

return cp;

}

AD queuetostack(AD cp,CAR \*cq)

{

AD q=NULL;

CAR w;

if(\*cq)

{

q=(AD)malloc(sizeof(management));

if(q==NULL)

printf("进车失败！\n");

w=\*cq;

(\*cq)=(\*cq)->next;

q->car\_id=w->car\_id;

getatime(&q);

q->next=cp;

free(w);

printf("\n\n等候进入停车场的车队的第一辆车,车牌号为：%d , 成功进入停车场\n",q->car\_id);

printf ( "进场时间为 %4d-%02d-%02d %02d:%02d:%02d\n\n",q->arivaltime.year, q->arivaltime.mon,q->arivaltime.day,q->arivaltime.hour,q->arivaltime.min,q->arivaltime.sec);

return q;

}

}

AD pop\_car(AD cp,CAR \*cq)/\*车库出库\*/

{

if(cp==NULL)

{

printf("停车场里没有车\n");

return NULL;

}

CAR w;

int id,n;

AD p,q,sh,d;

printf("输入要离开的车的车牌号：\n");

scanf("%d",&id);

n=check(cp,id);

while(n)

{

printf("该车不在停车场内！请重新输入 (输入 0 返回主菜单)： ");

scanf("%d",&id);

if(id==0)

return cp;

n=check(cp,id);

}

d=cp;

p=cp;

q=cp;

sh=NULL;

if(cp->car\_id==id)/\*判断栈顶是否为要出库车辆\*/

{

d=cp;

cp=cp->next;

getltime(&d);

printf ( "进场时间为 %4d-%02d-%02d %02d:%02d:%02d\n\n",d->arivaltime.year, d->arivaltime.mon,d->arivaltime.day,d->arivaltime.hour,d->arivaltime.min,d->arivaltime.sec);

printf ( "离开时间为 %4d-%02d-%02d %02d:%02d:%02d\n\n",d->leavetime.year, d->leavetime.mon,d->leavetime.day,d->leavetime.hour,d->leavetime.min,d->leavetime.sec);

printf( "一共耗时： %.2f 秒\n", d->costtime );

p->cost=MONEY\*(p->costtime);

printf( "停车费为： %d 元\n", d->cost );

savehistory(d);

free(d);

}

else

{

while(cp->car\_id!=id)/\*出栈\*/

{

q=cp->next;

cp->next=NULL;

cp->next=sh;

sh=cp;

cp=q;

}

getltime(&cp);

printf ( "进场时间为 %4d-%02d-%02d %02d:%02d:%02d\n",cp->arivaltime.year, cp->arivaltime.mon,cp->arivaltime.day,cp->arivaltime.hour,cp->arivaltime.min,cp->arivaltime.sec);

printf ( "离开时间为 %4d-%02d-%02d %02d:%02d:%02d\n",cp->leavetime.year, cp->leavetime.mon,cp->leavetime.day,cp->leavetime.hour,cp->leavetime.min,cp->leavetime.sec);

printf( "一共耗时： %.2f 秒\n", cp->costtime );

cp->cost=MONEY\*(cp->costtime);

printf( "停车费为： %d 元\n", cp->cost );

savehistory(cp);

free(cp);

if(cp->next==NULL)

return sh;

else

{

cp=cp->next;

d=sh;

while(d)/\*入栈\*/

{

d=sh->next;

sh->next=cp;

cp=sh;

sh=d;

}

return cp;

}

}

return cp;

}

void travalhistory(AD cp)

{

if(cp==NULL)

{printf("历史记录为空！\n");

return;}

else

printf("车牌号\t入场时间： \t\t 离开时间：\t\t停车时间(秒）\t停车费(元)\n");

while(cp)

{

printf("%d\t%d-%d-%d %d:%d:%d\t%d-%d-%d %d:%d:%d\t%.0f\t\t%d\n",cp->car\_id,cp->arivaltime.year,cp->arivaltime.mon,cp->arivaltime.day,cp->arivaltime.hour,cp->arivaltime.min,cp->arivaltime.sec,

cp->leavetime.year,cp->leavetime.mon,cp->leavetime.day,cp->leavetime.hour,cp->leavetime.min,cp->leavetime.sec,cp->costtime,cp->cost);

cp=cp->next;

}

}

void traval(AD cp)/\*遍历停车场\*/

{

printf("停车场共有%d 个车位，停车场内有%d 辆。\n",NUMB,count(cp));

if(count(cp)>NUMB)

if(cp)

printf("分别有：\n");

while(cp)

{

printf("\t%d\n",cp->car\_id);

cp=cp->next;

}

}

void queuetra(CAR cq)/\*遍历候车道\*/

{

printf("在候车道上的车有%d 辆。\n",countqueue(cq));

if(cq)

printf("分别有：\n");

while(cq)

{

printf("\t%d\n",cq->car\_id);

cq=cq->next;

}

}

void output(AD cp,CAR cq)

{

char n='1',ch;

while(n!='0')

{

system("cls");

printf("\n\t\t1-停车场使用情况\n\t\t2-候车道排队状况\n\t\t0-返回主界面\n");

n=getch();

switch(n)

{

case '1':

system("cls");

traval(cp);

printf("\n\n请按任意键返回查询菜单！\n");

ch=getch();

break;

case '2':

system("cls");

queuetra(cq);

printf("\n\n请按任意键返回查询菜单！\n");

ch=getch();

break;

case '0':break;

default:printf("输入有误，请重新输入\n");

}

}

}

int main()

{

char n='1',ch;

management \*car\_prosition,\*history;/\*车位堆栈\*/

car\_queue \*wait;/\*候车位队列\*/

car\_prosition=NULL;

history=NULL;

wait=NULL;

input(&car\_prosition);

inputqueue(&wait);

while(n!='0')

{

system("cls");

mainmeun();

n=getch();

switch(n)

{

case '1':

system("cls");

car\_prosition=push\_car(car\_prosition,&wait);

save(car\_prosition);

printf("\n\n请按任意键返回主菜单！");

ch=getch();

break;

case '2':

system("cls");

car\_prosition=pop\_car(car\_prosition,&wait);

if(wait)

car\_prosition=queuetostack(car\_prosition,&wait);

printf("请按任意键返回主菜单！");

ch=getch();

break;

case '3':

system("cls");

output(car\_prosition,wait);

break;

case '4':

system("cls");

history=NULL;

readhistory(&history);

travalhistory(history);

printf("请按任意键返回主菜单！");

ch=getch();

break;

case '0':

system("cls");

printf("谢谢使用停车场管理系统");

break;

default:printf("输入有误，请重新输入！\n");break;

}

}

save(car\_prosition);

savecar(wait);

return 0;

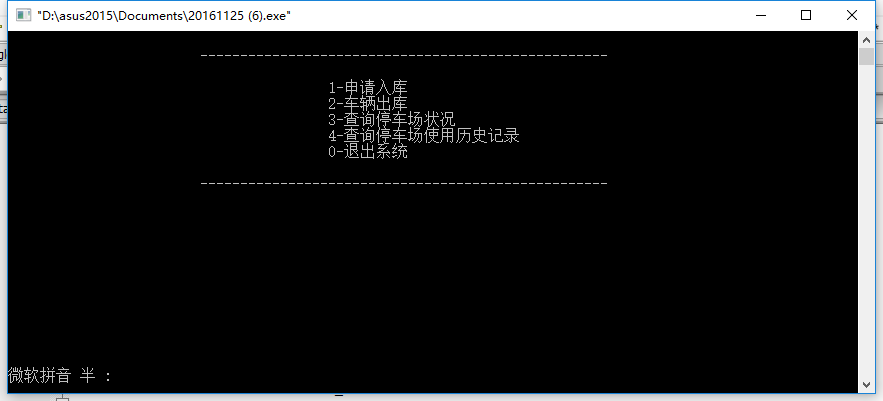
}

**1.6、调试分析**

刚开始编写代码时，不懂得如何获取电脑本地时间，通过上网查询与尝试编写获取时间函数，最后解决这个问题。在程序调试中，程序老是出现莫名其妙的问题，静下心来一个个找的时候，才发现是指针没有赋初值惹的祸。

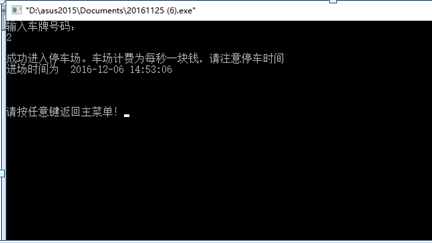
**1.7、运行结果及分析**

主菜单：



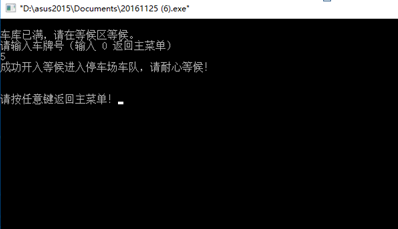
车辆进入停车场：



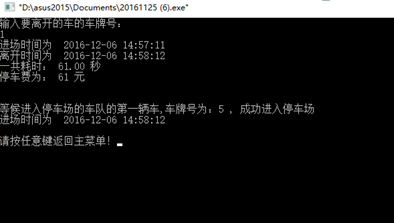




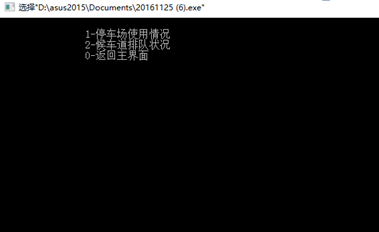




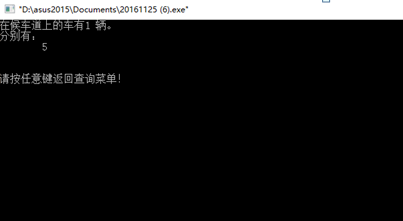
车辆离开停车场：



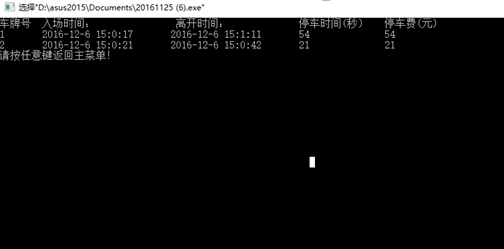
查询停车场：







查询停车场的历史记录：



结果都符合预期。

**2 哈夫曼编码/译码器**

**2.1、问题描述**

哈夫曼编码是一种应用广泛而有效的数据压缩技术。利用哈夫曼编码进行通信可以大大提高信通利用率，加快信息传输速度，降低传输成本。数据压缩的过程称为编码，解压缩的过程称为译码。进行信息传递时，发送端通过一个编码系统对待传数据（明文）预先编码，而接收端将传来的数据（密文）进行译码。要求设计这样一个简单的哈夫曼编码/译码系统。

使用语言：C语言

编译环境：Codeblocks

**2.2、需求分析**

一个完整的系统应该具有以下的功能：

①初始化。从终端读入字符集大小n、n个字符以及n个对应的权值，建立哈夫曼树；利用已经建好的哈夫曼树求每个叶结点的哈夫曼编码。

②编码。利用已构造的哈夫曼编码对输入的明文进行编码，然后将结果存入c://123.txt文件中。

③译码。在终端上显示c://123.txt文件的内容，并进行译码。最后将结果存入c://abc.txt文件中。

④打印。将哈夫曼树以表格的形式显示在终端上，同时将每个字符的哈夫曼编码显示出来。

**2.3、概要设计**

①主菜单：用于选择功能；

②建立哈夫曼树：输入基本内容进行哈夫曼树建立；

③编码：将输入的字符串进行编码输出并保存；

④译码：将编码文件保存的内容读入并进行译码，译码后保存；

⑤输出：输出哈夫曼树及每个字符对应的哈夫曼编码。

**2.4、流程图**

****

**2.5、详细设计**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define MAX 21

#define MAXVALUE 10000

typedef struct {

char data;

int weight;

int parent;

int lchild;

int rchild;

}HNodetype;

typedef struct{

int bit[MAX];

int start;

}HCodetype;

HNodetype \*HuffMTree[2\*MAX-1];

HCodetype \*HuffCode[MAX];

int n;

void CreateHuffMTree();

void HaffmanCode();

void DeCode();

void Output();

int main()

{

int i,a;

while(1)

{

system("cls");

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

printf("请输入0~4\n");

printf("1-建树 2-编码 3-译码\n");

printf("4-输出 0-退出\n");

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

scanf("%d",&a);

if(a==0)

break;

switch(a)

{

case 1:CreateHuffMTree(); break;

case 2:HaffmanCode(); break;

case 3:DeCode(); break;

case 4:Output(); break;

}

}

}

void CreateHuffMTree()

{

system("cls");

int m1,x1,m2,x2;

int i,j;

printf("请输入结点个数：");

scanf("%d",&n);

for(i=0;i<2\*n-1;i++)

{

HuffMTree[i]=(HNodetype \*)malloc(sizeof(HNodetype));

HuffMTree[i]->weight=0;

HuffMTree[i]->parent=HuffMTree[i]->lchild=HuffMTree[i]->rchild=-1;

HuffMTree[i]->data='0';

}

for(i=0;i<n;i++){

getchar();

printf("请输入字符：");

scanf("%c",&HuffMTree[i]->data);

printf("请输入字符的权值：");

scanf("%d",&HuffMTree[i]->weight);}

for(i=0;i<n-1;i++)

{

x1=x2=MAXVALUE;

m1=m2=0;

for(j=0;j<n+i;j++)

{

if(HuffMTree[j]->parent==-1&&HuffMTree[j]->weight<x1)

{

x2=x1;m2=m1;

x1=HuffMTree[j]->weight; m1=j;

}

else if(HuffMTree[j]->parent==-1&&HuffMTree[j]->weight<x2)

{

x2=HuffMTree[j]->weight;

m2=j;

}

}

HuffMTree[m2]->parent=n+i; HuffMTree[m1]->parent=n+i;

HuffMTree[n+i]->weight=HuffMTree[m1]->weight+HuffMTree[m2]->weight;

HuffMTree[n+i]->lchild=m2; HuffMTree[n+i]->rchild=m1;

HuffMTree[n+i]->data='0';

}

printf("按回车继续");

getchar();getchar();

}

void HaffmanCode()

{

system("cls");

HCodetype \*cd=(HCodetype \*)malloc(sizeof(HCodetype));

int i,j,c,p;

FILE \*fp;

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

{

cd->start=n-1;

c=i;

p=HuffMTree[c]->parent;

while(p!=-1)

{

if(HuffMTree[p]->lchild==c)

cd->bit[cd->start]=0;

else

cd->bit[cd->start]=1;

cd->start--;

c=p;

p=HuffMTree[c]->parent;

}

HuffCode[i]=(HCodetype \*)malloc(sizeof(HCodetype));

for(j=cd->start+1;j<n;j++)

HuffCode[i]->bit[j]=cd->bit[j];

HuffCode[i]->start=cd->start+1;

}

printf("字符信息--编码信息\n");

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

{

printf("%c:",HuffMTree[i]->data);

for(j=HuffCode[i]->start;j<n;j++)

printf("%d",HuffCode[i]->bit[j]);

printf("\n");

}

printf("请输入请输入要编码的字符串,基本元素为（");

for(i=0;i<n-1;i++)

printf("%c,",HuffMTree[i]->data);

printf("%c",HuffMTree[i]->data);

printf("）\n");

char data[100];

scanf("%s",data);

int f;

f=strlen(data);

printf("字符串编码后为:");

if((fp=fopen("c:\\123.txt","w"))==NULL)

{

printf("文件打不开\n");

return;

}

for(int x=0;x<f;x++)

{

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

{

if(data[x]==HuffMTree[i]->data)

{

for(j=HuffCode[i]->start;j<n;j++)

{

fprintf(fp,"%20d",HuffCode[i]->bit[j]);

printf("%d",HuffCode[i]->bit[j]);

}

}

}

}

printf("\n解码内容文件已经存储在c盘\n");

fclose(fp);

printf("按回车继续");

getchar();getchar();

}

void DeCode()

{

system("cls");

int i;

int tempcode[100];

int num=0;

FILE \*fp,\*fp1;

for(i=0;i<100;i++)

tempcode[i]=-1;

if((fp1=fopen("c:\\abc.txt","w+"))==NULL)

{

printf("文件打不开\n");

return;

}

if((fp=fopen("c:\\123.txt","r"))==NULL)

{

printf("文件打不开\n");

return;

}

printf("从文件中读出的编码为:");

while(!feof(fp))

{

fscanf(fp,"%d",&tempcode[num]);

num++;

}

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

printf("%d",tempcode[i]);

printf("\n");

int m=2\*n-2;

i=0;

printf("译码后为:");

while(i<num)

{

while(HuffMTree[m]->lchild!=-1&&HuffMTree[m]->rchild!=-1)

{

if(tempcode[i]==0)

{

m=HuffMTree[m]->lchild;

i++;

}

else if(tempcode[i]==1)

{

m=HuffMTree[m]->rchild;

i++;

}

}

printf("%c",HuffMTree[m]->data);

fprintf(fp1,"%20c",HuffMTree[m]->data);

m=2\*n-2;

}

printf("\n");

printf("译码内容文件已经存储在c盘\n");

fclose(fp);

fclose(fp1);

printf("按回车继续");

getchar();getchar();

}

void Output()

{

int i,j;

system("cls");

printf("显示哈夫曼树的信息\n");

printf("字符\t权值\t左孩子\t右孩子\t双亲\t\n");

for(i=0;i<2\*n-1;i++)

{

printf("%c\t",HuffMTree[i]->data);

printf("%d\t%d\t%d\t%d\n",HuffMTree[i]->weight,HuffMTree[i]->lchild,HuffMTree[i]->rchild,HuffMTree[i]->parent);

}

printf("字符信息--编码信息\n");

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

{

printf("%c:",HuffMTree[i]->data);

for(j=HuffCode[i]->start;j<n;j++)

printf("%d",HuffCode[i]->bit[j]);

printf("\n");

}

printf("按回车继续");

getchar();

getchar();

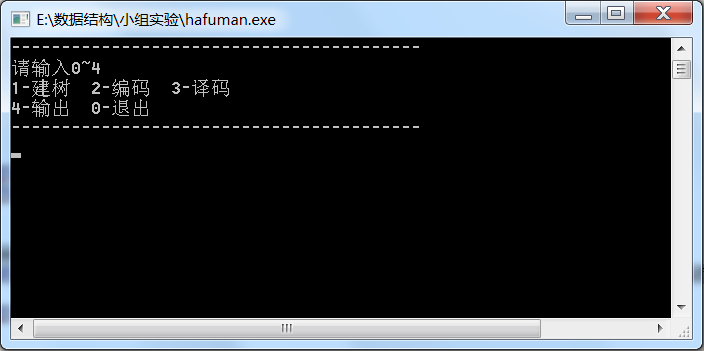
}

**2.6、调试分析**

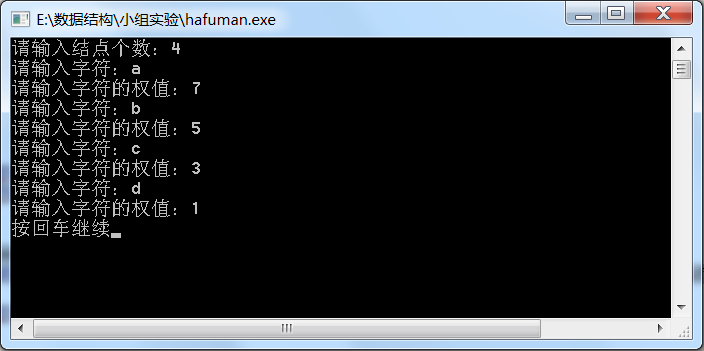
一开始HuffCode[MAX]和cd这些指针没有用申请动态内存分配，程序运行的时候总会崩溃，经过不停的查找资料与询问同学，加上申请动态内存分配的语句，最终程序可以顺利运行。

**2.7、运行结果及分析**

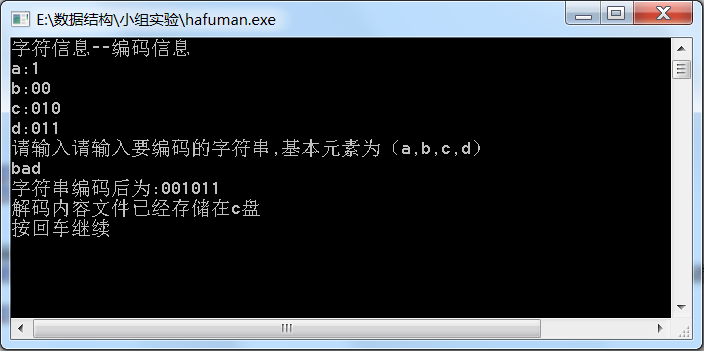
主菜单：



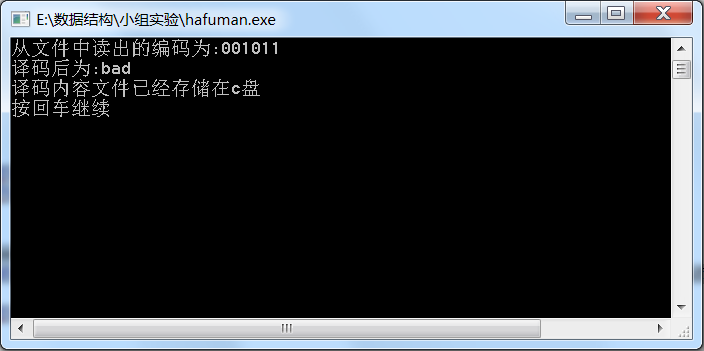
建立哈夫曼树：



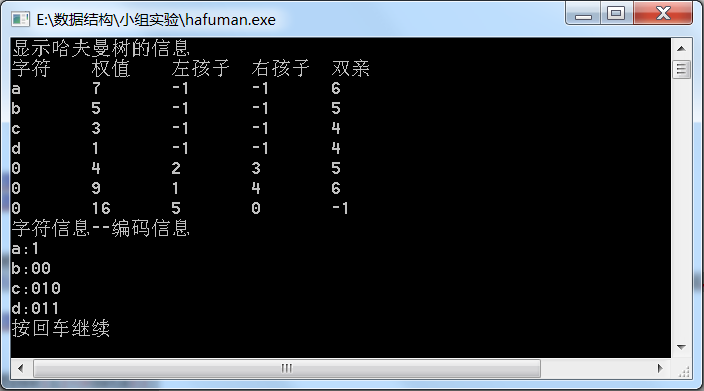
编码：



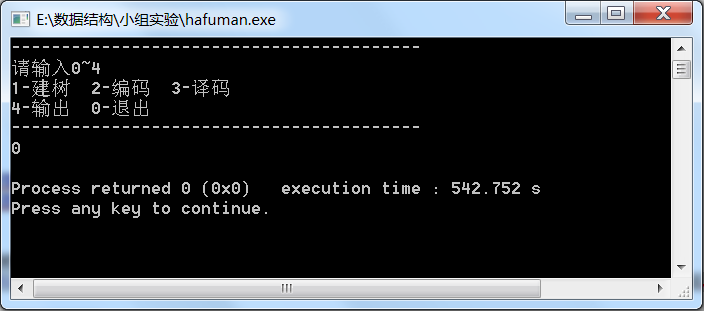
译码：



输出：



退出：



结果都符合预期。

**3 最小生成树的Kruskal算法**

**3.1、问题描述**

Kruskal算法是根据边的权值以递增的方式依次找出权值最小的边来建立最小生成树，规定每次添加的边不能造成生成树有回路。

使用语言：C语言

编译环境：Codeblocks

**3.2、需求分析**

例如，一个图中的所有边按权值递增排序如下：

（4,5）2 （3,4）3 （1,4）5 （1,3）6 （1,2）7 （2,4）8 （2,5）8 （3,5）9

（1,5）12 （2,3）14

步骤1：将（4,5）的边添加到生成树中；

步骤2：将（3,4）的边添加到生成树中；

步骤3：将（1,4）的边添加到生成树中；

步骤4：将（1,3）的边添加到生成树中，但发现这时出现了回路（1,3,4顶点），改用将边（1,2）添加到生成树中，现在已经连通了1、2、3、4、5所有顶点，最小生成树建立完成。

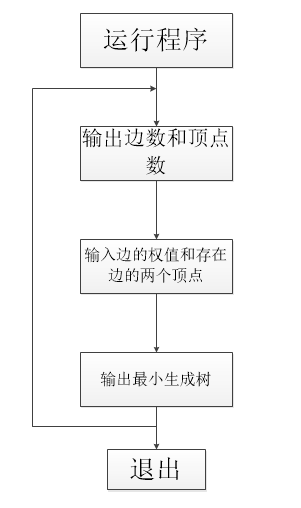
**3.3、概要设计**

①提示输入边数和顶点数

②提示输入边的权值和存在边的两个顶点

③输出最小生成树（Kruskal算法）

**3.4、流程图**



**3.5、详细设计**

#include<stdio.h>

#include<stdlib.h>

#define M 200

#define MAX 200

//权值：边的数

typedef struct

{

int begin;

int end;

int weight;

}edge;//用于权值排序

typedef struct

{

int adj;//是否存在关系的标识

int weight;

}AdjMatrix[MAX][MAX];//邻接矩阵

typedef struct

{

AdjMatrix arc;

int vexnum,arcnum;//顶点数，边数

}Mgraph;//图的构造

void Creatgraph(Mgraph \*);

void sort(edge \*,Mgraph \*);

void Minispantree(Mgraph \*);

int Find(int \*,int );

void Swapn(edge \*,int ,int );

int main()

{

Mgraph \* G;

int op;

G = (Mgraph \*)malloc(sizeof(Mgraph));

if(G == NULL)

{

printf("内存分配失败\n");

exit(1);

}

while(1)

{

printf("程序开始运行,请按1继续，按0退出\n");

scanf("%d",&op);

if(op == 1)

{

Creatgraph(G);

Minispantree(G);

}

else

exit(1);

}

//system("pause");//按任意键进行返回

}

void Creatgraph(Mgraph \* G)//构造图

{

int i,j,n,m;

printf("请输入边数和顶点数:");

scanf("%d %d",&G -> arcnum,&G -> vexnum);

for(i = 0;i < G -> vexnum;i++)//顶点数

{

for(j = 0;j < G -> vexnum;j++);

{

G -> arc[i][j].adj = 0;

}

}

for(i = 0;i < G -> arcnum; i++)

{

printf("请输入有边的两个顶点:");

scanf("%d%d",&n,&m);

while(n<0||n>G->vexnum||m<0||n>G->vexnum)

{

printf("输入的数字不符合要求，请重新输入:");

scanf("%d%d",&n,&m);

}

G -> arc[n][m].adj = 1;

G -> arc[m][n].adj = 1;

//getchar();

printf("请输入%d与%d之间的权值:",n,m);

scanf("%d",&G -> arc[n][m].weight);

}

printf("邻接矩阵为:\n");

for(i = 0;i < G -> vexnum;i++)

{

for(j = 0;j < G -> vexnum;j++)

{

printf("%d ",G -> arc[i][j].adj);

}

printf("\n");

}

}

void sort(edge edges[],Mgraph \* G)//权值排序函数

{

int i,j;

for(i = 1; i < G -> arcnum;i++)

{

for(j = i+1; j <= G -> arcnum;j++)

{

if(edges[i].weight > edges[j].weight)

{

Swapn(edges,i,j);

}

}

}

printf("权值排序之后为:\n");

for( i = 1;i <= G -> arcnum;i++)

{

printf("<< %d,%d >> %d\n",edges[i].begin,edges[i].end,edges[i].weight);

}

}

void Swapn(edge \* edges,int i,int j)//权值交换

{

int temp;

temp = edges[i].begin;

edges[i].begin = edges[j].begin;

edges[j].begin = temp;

temp = edges[i].end;

edges[i].end = edges[j].end;

edges[j].end = temp;

temp = edges[i].weight;

edges[i].weight = edges[j].weight;

edges[j].weight = temp;

}

void Minispantree(Mgraph \* G)//生成最小生成树

{

int i,j,n,m;

int k = 1;

int parent[M];

edge edges[M];//结构体数组

for( i = 1; i < G -> vexnum; i++)//vexnum顶点数 arcnum边数

{

for( j = i+1 ; j <= G -> vexnum; j++)

{

if(G -> arc[i][j].adj == 1)//如果边存在关系

{

edges[k].begin = i;

edges[k].end = j;

edges[k].weight = G -> arc[i][j].weight;

k++;

}

}

}

sort(edges,G);//对最小生成树进行权值排序

for(i = 1; i <= G -> arcnum; i++)

{

parent[i] = 0;

}

printf("最小生成树为:\n");

for(i = 1;i <= G -> arcnum; i++)//kruckal算法

{

n = Find(parent,edges[i].begin);

m = Find(parent,edges[i].end);

if(n != m)

{

parent[n] = m;

printf("%d\n",parent[n]);

printf("<< %d, %d >> %d\n",edges[i].begin,edges[i].end,edges[i].weight);

}

}

}

int Find(int \* parent,int f)

{

while(parent[f] > 0)

{

f = parent[f];

}

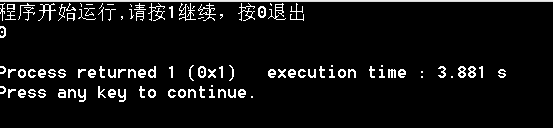
return f;

}

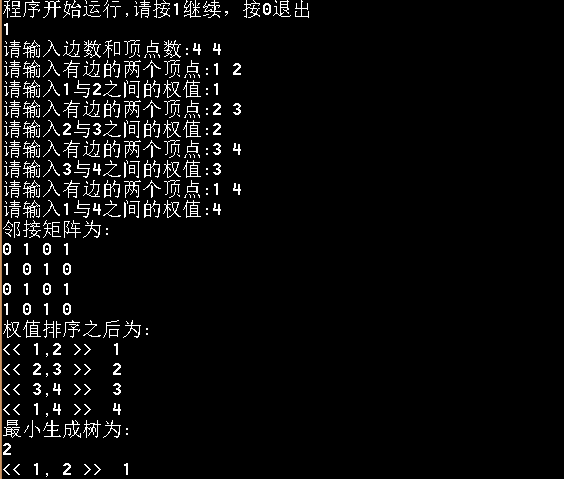
**3.6、调试分析**

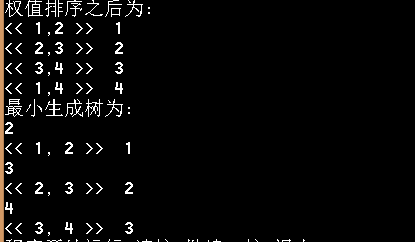
处理邻接矩阵的输出时，在处理for循环的问题上不够清晰，导致输出错误并带有乱码。经过小组成员的共同努力，仔细对for循环内的代码进行排查，成功解决了问题

**3.7、运行结果及分析**



继续运行程序





结果都符合预期。

**4 航空客运订票系统**

**4.1、问题描述**

航空客运订票的业务活动包括查询航线和客票预定的信息、客票预订和办理退票等，设计一个程序使上述任务借助计算机来完成

使用语言：C语言

编译环境：Codeblocks

**4.2、需求分析**

1. 系统必须存储的数据信息

⑴航班信息：航班号、飞机号、飞机抵达的城市、起飞日期、起飞时间、降落时间、航班票价、总位置和剩余位置

⑵客户信息：客户姓名、客户证件号、客户乘坐的航班号、客户所乘坐的飞机号、座位号

1. 系统能实现的操作和功能

⑴ 查询功能：

① 查询航线信息：输入所要抵达的城市，若满足要求，则会输出航班号、飞机号、起飞日期、起飞时间、降落时间、航班票价和剩余位置等信息

② 查询客户预订信息：根据客户证件号输出客户姓名、航班号、飞机号和座位号

⑵ 承办订票业务：先通过查询功能查询客户所要抵达的城市在该系统中是否存在，若满足要求，则为客户办理订票手续，输出座位号

⑶承办退票业务：根据客户的姓名和证件号输出客户所有的订票信息，选择相关的订票编号，为客户办理退票手续

**4.3、概要设计**

typedef struct airline /\*飞机航班的结构定义\*/

{

char air\_num[8];

char plane\_num[8];

char end\_place[20];

char start\_data[20];

char start\_time[8];

char end\_time[8];

double price;

int total;

int left;

struct airline \*next;

}airline;

typedef struct customer /\*顾客信息的结构定义\*/

{

char name[8];

char cus\_num[20];

char air\_num[8];

char plane\_num[8];

int seat\_num;

struct customer \*next;

}customer;

airline \*start\_air() /\*创建航班链表\*/

customer \*start\_cus() /\*创建顾客链表\*/

int insert\_air(airline \*\*p,char \*air\_num,char \*plane\_num,char \*end\_place,char \*start\_data,char \*start\_time,char \*end\_time,double price,int total,int left) /\*增加航班的信息\*/

int insert\_cus(customer \*p,char \*name,char \*air\_num,char \*plane\_num,char \*cus\_num,int seat\_num) /\*增加某航班的顾客信息\*/

int book(airline \*a,char \*end\_place,customer \*c,char \*name,char \*cus\_num,char \*start\_data) /\*订票操作\*/

int delete\_cus(airline \*air, customer \*cus) /\*取消订票信息操作\*/

int search\_air(airline \*head) /\*查询航班信息操作\*/

int search\_cus(customer \*head) /\*查找顾客信息操作\*/

int creat\_air(airline \*\*l) /\*预先设置航班信息\*/

int creat\_cus(customer \*\*l) /\*预先设置已订票的顾客信息\*/

**4.4、流程图**

开始

主菜单

查询服务

退出

退票业务

订票业务

查询客户预订信息

查询航线信息

结束

**4.5、详细设计**

/\*

1.想了解座位是否满额的情况，可先通过查询功能直接了解航线情况

\*/

#include <stdio.h>

#include <malloc.h>

#include <string.h>

#include <stdlib.h>

#define OK 1

#define ERROR 0

typedef struct airline /\*飞机航班的结构定义\*/

{

char air\_num[8];

char plane\_num[8];

char end\_place[20];

char start\_data[20];

char start\_time[8];

char end\_time[8];

double price;

int total;

int left;

struct airline \*next;

}airline;

typedef struct customer /\*顾客信息的结构定义\*/

{

char name[8];

char cus\_num[20];

char air\_num[8];

char plane\_num[8];

int seat\_num;

struct customer \*next;

}customer;

airline \*start\_air() /\*创建航班链表\*/

{

airline \*a;

a = (airline\*)malloc(sizeof(customer));/\*airline空间占用\*/

if(a==NULL)

printf("空间不足");

return a;

}

customer \*start\_cus() /\*创建顾客链表\*/

{

customer \*c;

c = (customer\*)malloc(sizeof(customer));/\*customer空间占用\*/

if(c==NULL)

printf("空间不足");

return c;

}

int insert\_air(airline \*\*p,char \*air\_num,char \*plane\_num,char \*end\_place,char \*start\_data,char \*start\_time,char \*end\_time,double price,int total,int left) /\*增加航班信息\*/

{

airline \*q;

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

strcpy(q->air\_num,air\_num);

strcpy(q->plane\_num,plane\_num);

strcpy(q->end\_place,end\_place);

strcpy(q->start\_data,start\_data);

strcpy(q->start\_time,start\_time);

strcpy(q->end\_time,end\_time);

q->price = price;

q->total = total;

q->left = left;

q->next = NULL;

(\*p)->next = q;

(\*p) = (\*p)->next;

return OK;

}

int insert\_cus(customer \* p,char \*name,char \*air\_num,char \*plane\_num,char \*cus\_num,int seat\_num) /\*增加某航班的顾客信息\*/

{

customer \*q;

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

strcpy(q->name,name);

strcpy(q->air\_num,air\_num);

strcpy(q->cus\_num,cus\_num);

strcpy(q->plane\_num,plane\_num);

q->seat\_num = seat\_num;

q->next = NULL;

//(\*p)->next = q;

//(\*p) = (\*p)->next;

//若用课本里面的代码，则无法将链表一个一个连接下去

customer \*l= p;

for(; l->next != NULL; l =l->next){}

l->next = q;

l = l->next;

return OK;

}

int book(airline \*a,char \*end\_place,customer \*c,char \*name,char \*cus\_num,char \*start\_data) /\*订票操作\*/

{

airline \*p = a;

customer \*q = c ->next;

p = a->next;

while(p!=NULL){

if(strcmp(end\_place,p ->end\_place)==0&&strcmp(start\_data,p->start\_data)==0)

{

if(p->left>0)

{ //如果座位大于0下一步

char \*r;

r= p->air\_num;

printf("\t\t您的座位号为：%d\n",(p->total - p->left+1));

insert\_cus(c,name,r,p->plane\_num,cus\_num,p->total - p->left+1);

p ->left--;

return OK;

}

else

printf("\t\t该航班座位已满！\n");

}

p = p->next;}

return 0;

}

int delete\_cus(airline \*air, customer \*cus) /\*取消订票信息操作\*/

{

char name[8];

char id[20];

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

printf("\n");

printf("\t请输入需要退票乘客姓名：");

scanf("%s",name);

printf("\n");

printf("\t请输入需要退票乘客证件号：");

scanf("%s",&id);

customer \*p = cus;

int i = 0;

printf("\n");

for(customer \*c = cus->next; c!=NULL; c = c->next){

if(strcmp(c->name, name)==0 && strcmp(c->cus\_num,id)==0){

printf("\n");

printf("编号 航班号 飞机号 座位号 抵达城市 起飞日期 起飞时间 预计抵达时间 票价\n");

airline \*t;

for(t = air->next;t!=NULL;t= t->next){

if(strcmp(c->air\_num,t->air\_num)==0){

i++;

printf(" %d %s %s %d %s %s %s %s %.2f\n",i, t->air\_num, t->plane\_num, c->seat\_num, t->end\_place, t->start\_data, t->start\_time, t->end\_time, t->price);

}

}

}

}

if(i == 0)

{

printf("该客户没有票务信息！\n");

return 0;

}

int n;

i = 0;

printf("请输入您要删除的机票的编号：");

scanf("%d",&n);

customer \*o = cus;

for(customer \*c = cus->next; c!=NULL; c = c->next){

if(strcmp(c->name, name)==0 && strcmp(c->cus\_num,id)==0){

airline \*t;

for(t = air->next;t!=NULL;t= t->next){//乘客有两张票，一张就遍历所有航班号，寻找对应的航班号

if(strcmp(c->air\_num,t->air\_num)==0){

i++;

if(n==i){

o ->next = c ->next;

free(c);

printf("删除机票成功！\n");

return 0;

}

}

}

}

o = o->next;

}

return 0;

}

int search\_air(airline \*head) /\*查找航班信息操作\*/

{

airline \*p;

p = head->next;

int flag = 0;

printf(" 请输入你所要到达的城市（获取航班信息）：\n");

char end\_place[20];

printf(" ");

scanf("%s",end\_place);

while(p!=NULL)

{

if(strcmp(end\_place,p->end\_place)==0)

{

printf("\n");

printf("航班号 飞机号 起飞日期 起飞时间 到达时间 航班票价 剩余位置\n");

printf("%s\t %s\t %s\t %s\t %s\t %.2f\t %d\n",p->air\_num,p->plane\_num,p->start\_data,p->start\_time,p->end\_time,p->price,p->left);

flag = 1;

}

p = p ->next;

}

if(flag == 0)

{

printf("\n");

printf(" 抱歉,这里没有您所要到达城市的航班信息!\n");

}

return ERROR;

}

int search\_cus(customer \*head) /\*查找顾客信息操作\*/

{

customer \*p;

p = head->next;

int flag = 0;

printf(" 请输入客户证件号：");

char cus\_num[20];

scanf("%s",cus\_num);

while(p!=NULL)

{

if(strcmp(cus\_num,p->cus\_num)==0)

{

printf("\n");

printf("\t\t 姓名 航班号 飞机号 座位号\n");

printf("\t\t %s %s %s %d\n",p->name,p->air\_num,p->plane\_num,p->seat\_num);

flag = 1;

}

p = p ->next;

}

if(flag == 0)

{ printf("\n");

printf(" 你所输入的证件号没有找到对应的客户信息！\n");

}

return ERROR;

}

int creat\_air(airline \*\*l) /\*预先设置航班信息\*/

{

airline \*p = \*l;

int i = 0;

char \*air\_num[3] = {"001af","002af","003af"};

char \*plane\_num[3] = {"plane1","plane2","plane3"};

char \*end\_place[3] = {"Beijing","Beijing","Guangzhou"};

char \*start\_data[3]= {"2016-12-13","2016-12-14","2016-12-15"};

char \*start\_time[3] = {"10:00","11:00","12:00"};

char \*end\_time[3] = {"11:00","12:00","13:00"};

int total[3] = {50,50,50};

double price[3] = {50.00,75.00,65.00};

int left[3] = {45,35,0};

for(i=0;i<3;i++){

insert\_air(&p,air\_num[i],plane\_num[i],end\_place[i],start\_data[i],start\_time[i],end\_time[i],price[i],total[i],left[i]);

}

return OK;

}

int creat\_cus(customer \*\*l) /\*预先设置已订票的顾客信息\*/

{

customer \*p = \*l;

int i = 0;

char \*name[3] = {"Jack","Jack"};

char \*air\_num[3] = {"001af","002af"};

char \*plane\_num[3] = {"plane1","plane2"};

char \*cus\_num[3] = {"123456","123456"};

int seat\_num[3] = {2,10};

for(i = 0;i < 2;i++)

insert\_cus(p,name[i],air\_num[i],plane\_num[i],cus\_num[i],seat\_num[i]);

return OK;

}

int main() /\*主菜单\*/

{

int t = 1,x;

customer \*cus = start\_cus();

cus->next = NULL;

airline \*air = start\_air();

char name[8],cus\_num[20],end\_place[20],ch,start\_time[20],start\_data[20];

creat\_air(&air);

creat\_cus(&cus);

while(t == 1)

{ printf("\n");

printf("\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\* \*\n");

printf("\t\t\* 欢迎使用航空订票系统 \*\n");

printf("\t\t\* 查询服务----------1 \*\n");

printf("\t\t\* 订票业务----------2 \*\n");

printf("\t\t\* 退票业务----------3 \*\n");

printf("\t\t\* 退出--------------0 \*\n");

printf("\t\t\* \*\n");

printf("\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

fflush(stdin);

printf("\t\t请输入您的选择：");

scanf("%c",&ch);

getchar();

switch(ch)

{

case '1':

system("cls");

printf("\t ---------------------------------------------------");

printf("\n");

printf("\n");

printf(" 请输入您要选择的服务：\n");

printf("\n");

printf(" 查询航线信息-------1\n");

printf("\n");

printf(" 查询客户预订信息---2\n");

printf("\n");

printf(" ");

scanf("%d",&x);

printf("\n");

if(x==1){

search\_air(air);

system("pause");

system("cls");

break;}

else if(x==2){

search\_cus(cus);

system("pause");

system("cls");

break;}

else

break;

case '2':

system("cls");

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

printf("\t \n");

printf("\t 欢迎来到航空客运订票窗口 \n");

printf("\t \n");

printf("\t 请输入您的名字：");

scanf("%s",name);

printf("\n");

printf("\t 请输入您的证件号：");

scanf("%s",cus\_num);

printf("\n");

printf("\t 请输入您要到达的城市：");

scanf("%s",end\_place);

printf("\n");

printf("\t 请输入您要出发的日期(例:2016-12-13)：");

scanf("%s",start\_data);

printf("\t ");

printf("\n");

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

printf("\n");

book(air,end\_place,cus,name,cus\_num,start\_data);

system("pause");

system("cls");

break;

case '3':

system("cls");

delete\_cus(air, cus);

system("pause");

system("cls");

break;

case '0':

t=0;

break;

}

}

return 0;

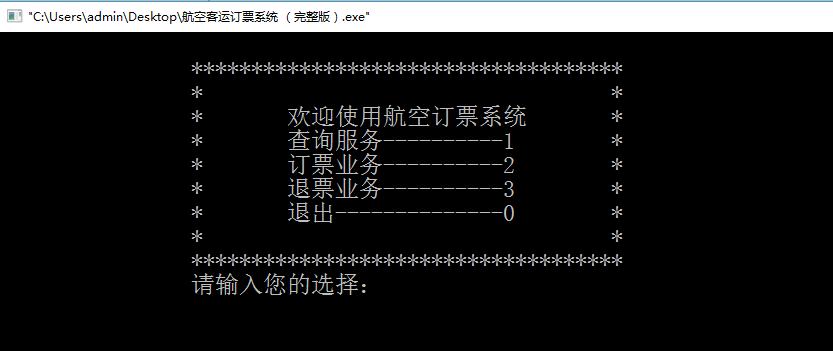
}

**4.6、调试分析**

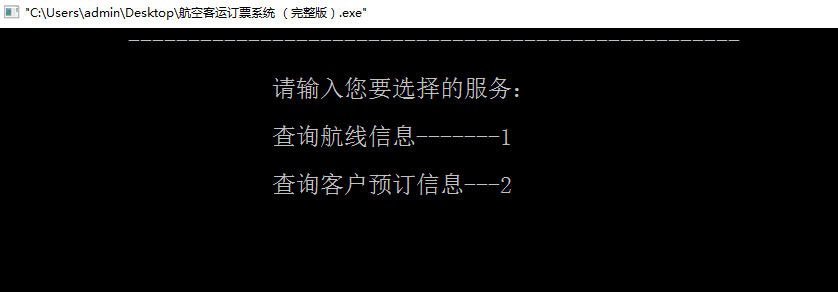
一开始调试课本里的代码的时候，可以运行，但是功能上和逻辑上不够健全，于是在网上找了部分代码进行参考借鉴，在调试增加某航班的顾客信息的函数的时候，发现用课本的代码无法将顾客的信息用链表的形式一个一个连接起来，所以做了部分的修改。

**4.7、运行结果及分析**

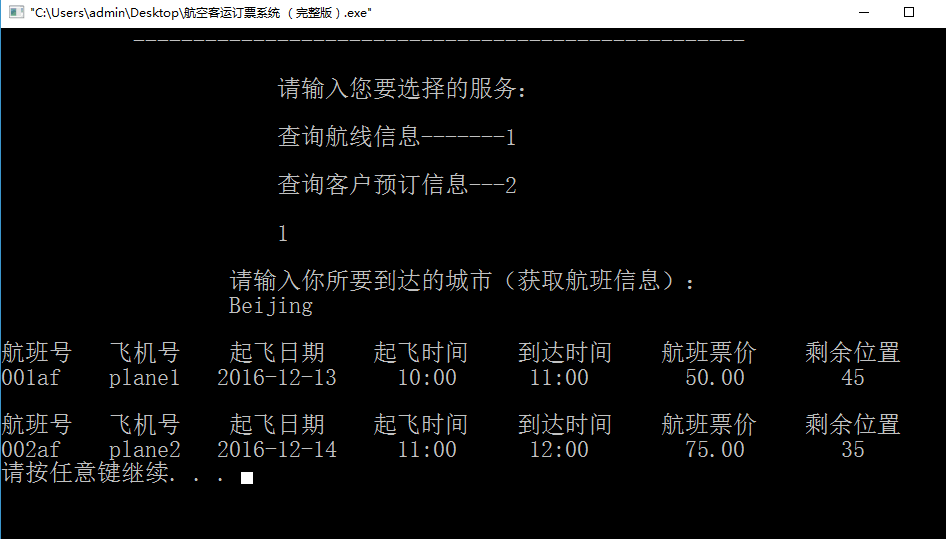
主菜单：



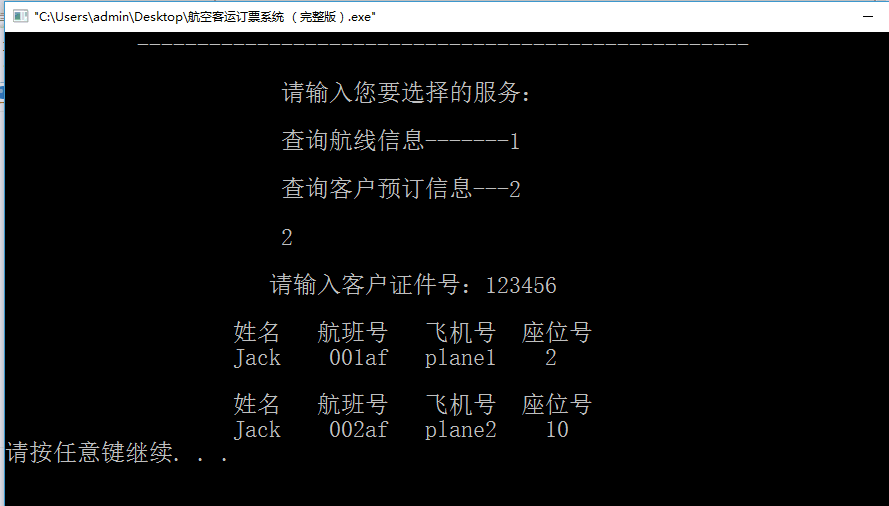
查询服务：

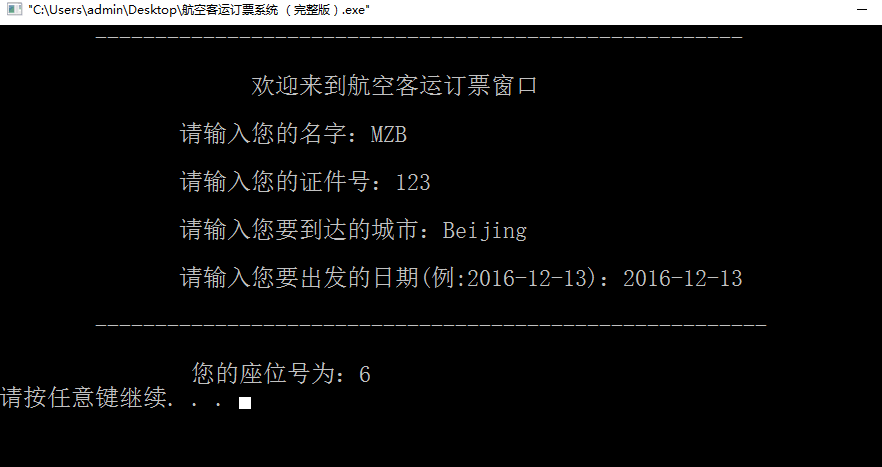


查询航线信息：

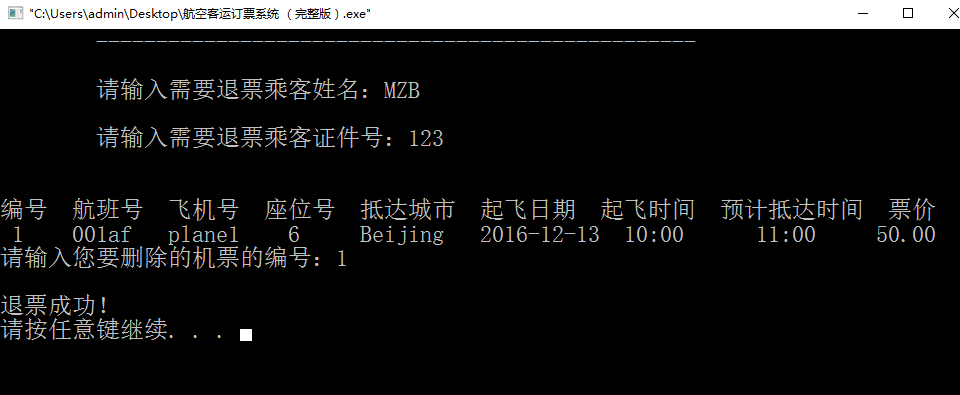


查询客户预订信息：

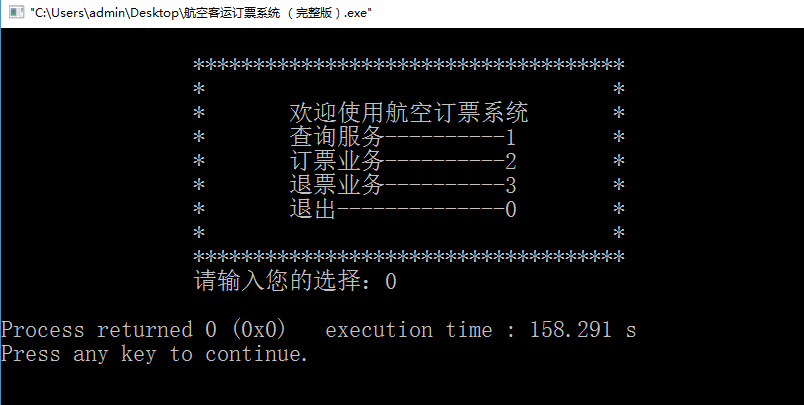
 订票业务：



退票业务：



退出：



符合预期的结果。

分析：首先打开查询功能，根据自己的需求点击按航线信息查询，输入自己想要的抵达的城市，然后得到相应的航班信息，再点击订票业务，输入相应的姓名，证件号码，想要抵达的城市，以及相关航班的起飞日期，得到相应的座位号码，想确定自己是否订票成功，则可以继续点击查询服务，再点击按客户预订信息查询，输入证件号，查询自己的预订信息，若想退票，则可点击退票业务，输入姓名和证件号码，接着会显示出你所预订的所有航班信息，选择想要删除的机票编号，输入相关编号则退票成功，最后直接点击退出即可退出程序。

《数据结构应用》评分表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 评分项目 | 高志劲 | 黄葆宁 | 梁智翔 | 马圳彬 |
| 完成情况（40%） |  |  |  |  |
| 答辩情况（20%） |  |  |  |  |
| 课程设计报告（20%） |  |  |  |  |
| 考勤与表现（20%） |  |  |  |  |
| 综合评分 |  |  |  |  |

指导教师签名：

日 期：