**《算法与数据结构》**

**实 验 指 导 手 册**

# 北京邮电大学软件学院

# 2020-2021学年第1学期实验报告

**课程名称： 算法与数据结构**

1. **实验名称： 实验二 栈、队列与递归算法设计**

**实验完成人：**

**姓名：**\_\_\_王宇涵\_\_\_\_\_**学号：**\_\_\_\_2020211730\_\_\_\_**成绩：**\_\_\_\_\_\_\_\_

**指导教师：**\_\_\_\_\_\_\_**\_\_贾红娓** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**日 期： 2021 年 10 月 8 日**

1. **实验目的**

仅仅认识到栈和队列是两种特殊的线性表是远远不够的，本次实验的目的在于使学生深入了解栈和队列的特征，以便在实际问题背景下灵活运用它们；同时还将巩固这两种结构的构造方法，接触较复杂问题的递归算法设计。

1. **实验内容**

**必做内容**

1. **数制转换问题**

**[问题描述]**

将十进制数N和其它d进制数的转换是计算机实现计算的基本问题，其解决方案很多，

其中最简单方法基于下列原理：即除d取余法。例如：（1348）10=（2504）8

|  |  |  |
| --- | --- | --- |
| **N** | **N div 8** | **N mod 8** |
| 1348 | 168 | 4 |
| 168 | 21 | 0 |
| 21 | 2 | 5 |
| 2 | 0 | 2 |

从表中我们可以看出，最先产生的余数4是转换结果的最低位，这正好符合栈的特性即后进先出的特性。所以可以用顺序栈来模拟这个过程。

**[基本要求]**

对于键盘输入的任意一个非负的十进制整数，打印输出与其等值的八进制数。由于上述

的计算过程是从低位到高位顺序产生的八进制数的各个数位，而打印输出，一般来说应从高位到低位进行，恰好和计算过程相反。因此可以先将计算过程中得到的八进制数的各位进栈，待相对应的八进制数的各位均产生以后，再使其按顺序出栈，并打印输出。即得到了与输入的十进制数相对应的八进制数。

**[测试数据]**

　　由学生依据软件工程的测试技术自己确定。注意测试边界数据。

1. **括号匹配的检验**

**[问题描述]**

假设表达式中允许有两种括号：圆括号和方括号，其嵌套的顺序随意，即(()[])或

[（[ ] [ ]）]等为正确格式，[（ ]）或（（（]均为不正确的格式。检验括号是否匹配的方法可用“***期待的紧迫程度***”这个概念来描述。例如：考虑下列的括号序列：

　　[　(　[　]　[　]　)　]

　　1　2　3　4　5　6　7　8

当计算机接受了第1个括号以后，它期待着与其匹配的第8个括号的出现，然而等来的却是第2个括号，此时第1个括号“[”只能暂时靠边，而迫切等待与第2个括号相匹配的第7个括号“）”的出现，类似的，因只等来了第3个括号“[”，此时，其期待的紧迫程度较第2个括号更紧迫，则第2个括号只能靠边，让位于第3个括号，显然第3个括号的期待紧迫程度高于第2个括号，而第2个括号的期待紧迫程度高于第1个括号；在接受了第4个括号之后，第3个括号的期待得到了满足，消解之后，第2个括号的期待匹配就成了最急迫的任务了，…… ，依次类推。可见这个处理过程正好和栈的特点相吻合。

**[基本要求]**

　　读入圆括号和方括号的任意序列，输出“匹配”或“此串括号匹配不合法”。

**[测试数据]**

　　输入([]())，结果“匹配”

　　输入 [(])，结果“此串括号匹配不合法”

**[实现提示]**

设置一个栈，每读入一个括号，若是左括号，则作为一个新的更急迫的期待压入栈中；

若是右括号，并且与当前栈顶的左括号相匹配，则将当前栈顶的左括号退出，继续读下一个括号，如果读入的右括号与当前栈顶的左括号不匹配，则属于不合法的情况。在初始和结束时，栈应该是空的。

**选作内容**

1. **停车场管理**

**[问题描述]**

设停车场内只有一个可停放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），若车场内已停满n辆汽车，则后来的汽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，在它之后开入的车辆必须先退出车场为它让路，待该辆车开出大门外，其它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交纳费用。试为停车场编制按上述要求进行管理的模拟程序。

**[测试数据]**

设n=2，输入数据为：（‘A’，1，5），（‘A’，2，10），（‘D’，1，15），（‘A’，3， 20），

（‘A’，4，25），（‘A’，5，30），（‘D’，2，35），（‘D’，4，40），（‘E’，0，0）。每一组输入数据包括三个数据项：汽车“到达”或“离去”信息、汽车牌照号码及到达或离去的时刻，其中，‘A’表示到达；‘D’表示离去，‘E’表示输入结束。

**[基本要求]**

以栈模拟停车场，以队列模拟车场外的便道，按照从终端读入的输入数据序列进行模拟

管理。每一组输入数据包括三个数据项：汽车“到达”或“离去”信息、汽车牌照号码及到达或离去的时刻，对每一组输入数据进行操作后的输出数据为：若是车辆到达，则输出汽车在停车场内或便道上的停车位置；若是车离去；则输出汽车在停车场内停留的时间和应交纳的费用（在便道上停留的时间不收费）。栈以顺序结构实现，队列以链表实现。

**[实现提示]**

需另设一个栈，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，也用顺序存

储结构实现。输入数据按到达或离去的时刻有序。栈中每个元素表示一辆汽车，包含两个数据项：汽车的牌照号码和进入停车场的时刻。

**[思考]**

　　（1） 两个栈共享空间，思考应开辟数组的空间是多少？

（2） 汽车可有不同种类，则它们的占地面积不同，收费标准也不同，如1辆客车和

1.5辆小汽车的占地面积相同，1辆十轮卡车占地面积相当于3辆小汽车的占地面积。

（3） 汽车可以直接从便道上开走，此时排在它前面的汽车要先开走让路，然后再依次

排到队尾。

（4） 停放在便道上的汽车也收费，收费标准比停放在停车场的车低，请思考如何修改

结构以满足这种要求。

1. **实验环境**

Dev 或者VS

1. **实验过程和实验结果**

**1.数值转换**

**1.1问题分析**

用键盘输入一个十进制数将其转换为d（8）进制数，计算过程是从低位到高位顺序产生的八进制数的各个数位，而打印输出，一般来说应从高位到低位进行，恰好和计算过程相反，所以我们可以将要输出的数存放到一个栈中，利用栈的LIFO特性

**1.2设计方案**

1. 用结构体先定义栈，包括数据和栈顶指针
2. 在创建栈前要先初始化，并实现一个判断栈空的函数，查栈顶元素函数
3. 入栈函数
4. 出栈函数
5. 转换进制的函数
6. 主函数



**1.3算法**

1）利用模运算和循环转换

void covert(int d)//转换

{

inits(s);

cin>>n;

while(n)

{

int mo=n%d;//将要放入栈的数

pushs(s,mo);

n=n/d;

}

while(!emptys(s))//输出栈

{

int e=0;

pops(s,e);

cout<<e;

}

}

1. 栈的操作

//初始化栈

void inits(sqstack &s)

{

s.top=-1;

}

//判断栈是否为空

bool emptys(sqstack s)

{

if(s.top==-1)

return true;

else

return false;

}

//入栈

bool pushs(sqstack &s,int& x)

{

if(s.top==MAXSIZE-1)//栈满报错

{

return false;

}

s.top++;//指针先加

s.data[s.top]=x;//新元素入栈

return true;

}

//出栈

bool pops(sqstack &s,int& x)

{

if(s.top==-1)//栈空报错

{

return false;

}

x=s.data[s.top];//栈顶元素出栈

s.top--;//指针再减

return true;

}

//查栈顶元素

bool gets(sqstack &s,int& x)

{

if(s.top==-1)//栈空报错

{

return false;

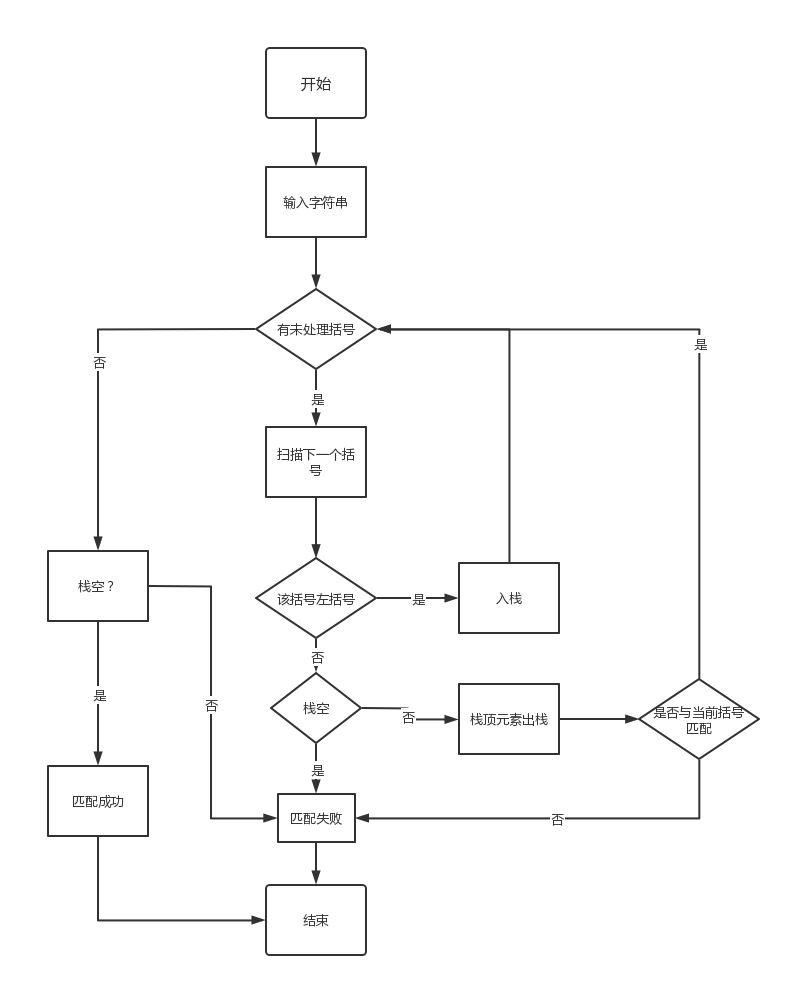
}

x=s.data[s.top];//栈顶元素出栈

return true;

}

**1.4设计图**



**1.5程序**

#include<cstdio>

#include<cstdlib>

#include<cstring>

#include<iostream>

#include<cmath>

#define MAXSIZE 10086

using namespace std;

int n,d;

//d为将要转换的进制

typedef struct

{

char data[MAXSIZE];

int top;//栈顶指针

}sqstack;

sqstack s;

//初始化栈

void inits(sqstack &s)

{

s.top=-1;

}

//判断栈是否为空

bool emptys(sqstack s)

{

if(s.top==-1)

return true;

else

return false;

}

//入栈

bool pushs(sqstack &s,int& x)

{

if(s.top==MAXSIZE-1)//栈满报错

{

return false;

}

s.top++;//指针先加

s.data[s.top]=x;//新元素入栈

return true;

}

//出栈

bool pops(sqstack &s,int& x)

{

if(s.top==-1)//栈空报错

{

return false;

}

x=s.data[s.top];//栈顶元素出栈

s.top--;//指针再减

return true;

}

//查栈顶元素

bool gets(sqstack &s,int& x)

{

if(s.top==-1)//栈空报错

{

return false;

}

x=s.data[s.top];//栈顶元素出栈

return true;

}

void covert(int d)//转换

{

inits(s);

cin>>n;

while(n)

{

int mo=n%d;//将要放入栈的数

pushs(s,mo);

n=n/d;

}

while(!emptys(s))//输出栈

{

int e=0;

pops(s,e);

cout<<e;

}

}

int main()

{

//cin>>d;

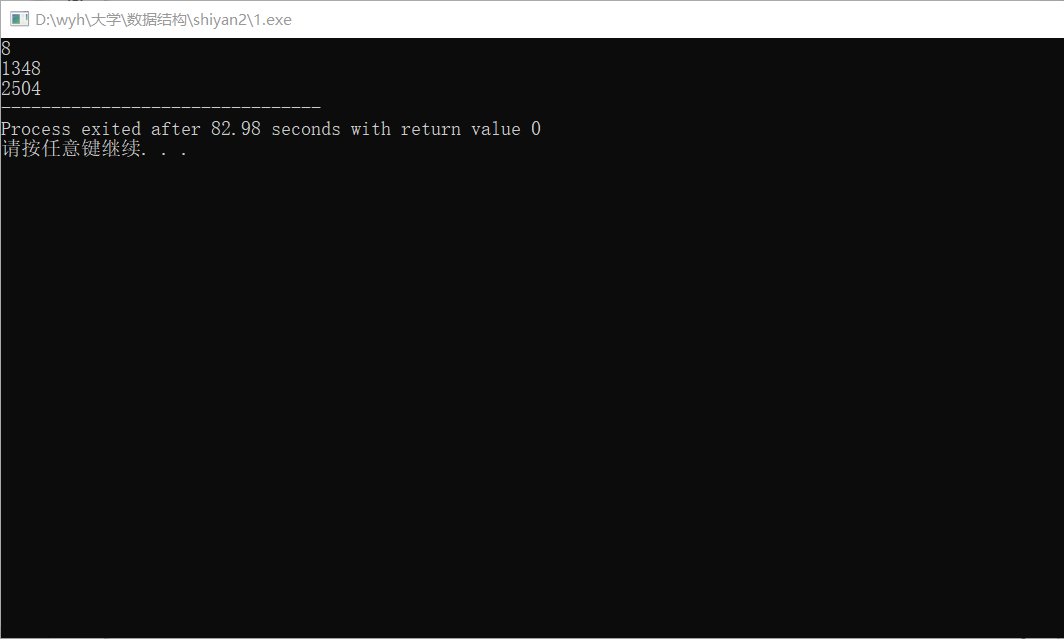
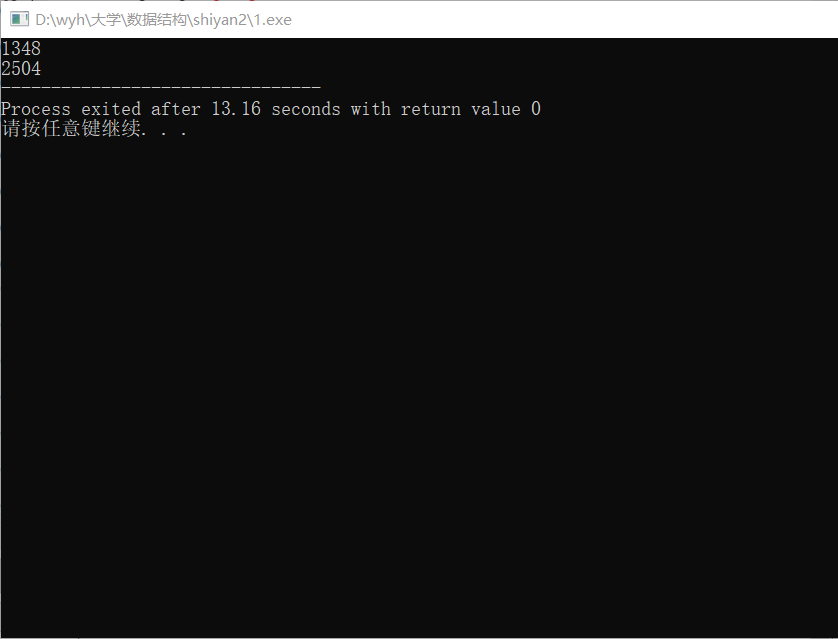
d=8;

covert(d);

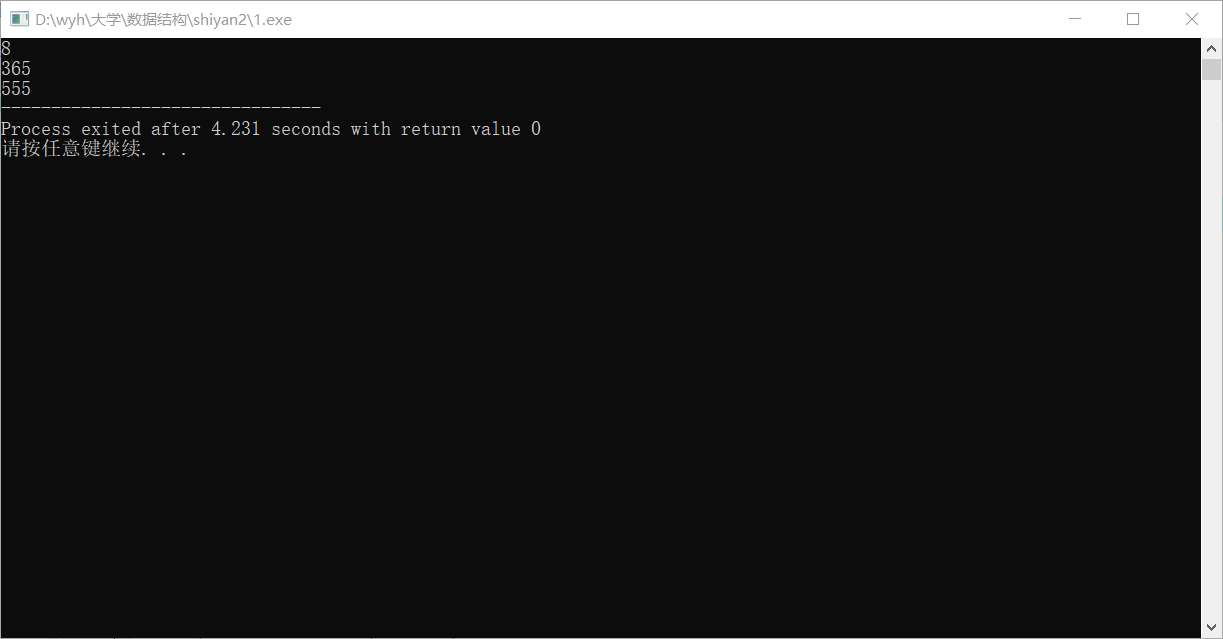
return 0;

}

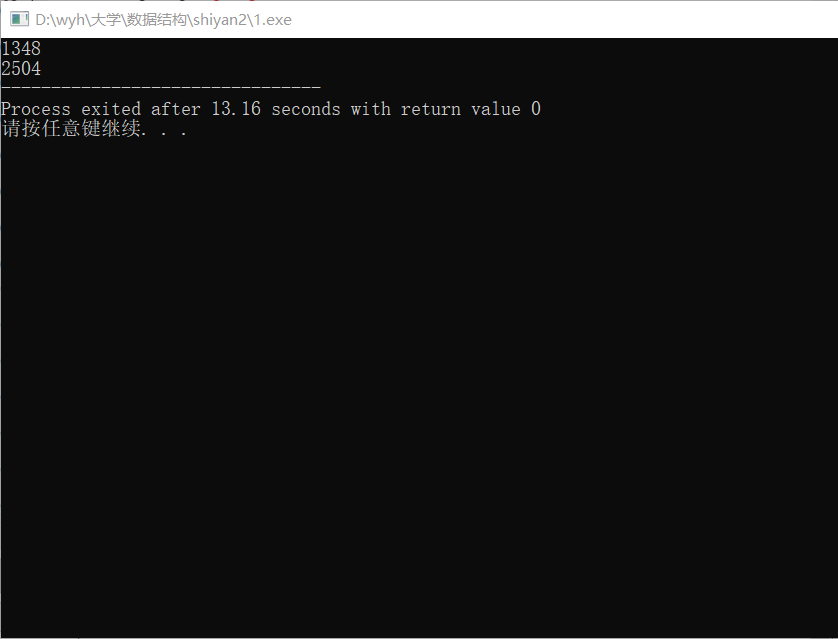
**1.6调试过程截图**

****

**测试d为其他进制时是否成功**

****

**1.7结果截图**

****

**2.括号匹配的检验**

**2.1问题分析**

当计算机接受了第1个括号以后，它期待着与其匹配的第8个括号的出现，然而等来的却是第2个括号，此时第1个括号“[”只能暂时靠边，而迫切等待与第2个括号相匹配的第7个括号“）”的出现，类似的，因只等来了第3个括号“[”，依次类推。这个处理过程正好和栈的特点相吻合。

将符合要求的符号"[“和”(“添加到栈中，然后将每一次输入的”]“和”)"与栈顶元素来进行比较，如果匹配则将栈中压入的元素弹出，否则就失败。当输入结束符“#”时，若栈为空，则匹配成功，否则，匹配失败

**2.2设计方案**

1）用结构体先定义栈，包括数据和栈顶指针

2）在创建栈前要先初始化，并实现一个判断栈空的函数，查栈顶元素函数

3）入栈函数

4）出栈函数

5）括号匹配的函数

6）主函数

**2.3算法**

1. **括号匹配的函数**

//匹配括号函数

bool checks(char str[],int len)

{

sqstack s;

inits(s);

for(int i=0;i<len;i++)

{

if(str[i]=='('||str[i]=='['||str[i]=='{')//扫描到右括号

{

pushs(s,str[i]);//入栈

}

else//扫描到左括号

{

if(emptys(s))//栈中没有元素

{

return false;

}

char tope;//栈顶元素

pops(s,tope);//栈顶元素出栈

if(str[i]==')'&&tope!='(')//判断是否匹配

{

return false;

}

if(str[i]==']'&&tope!='[')

{

return false;

}

if(str[i]=='}'&&tope!='{')

{

return false;

}

}

}

if(emptys(s))//若栈为空则正确

return true;

else

return false;

}

1. **栈操作**

//初始化栈

void inits(sqstack &s)

{

s.top=-1;

}

//判断栈是否为空

bool emptys(sqstack s)

{

if(s.top==-1)

return true;

else

return false;

}

//入栈

bool pushs(sqstack &s,char& x)

{

if(s.top==MAXSIZE-1)//栈满报错

{

return false;

}

s.top++;//指针先加

s.data[s.top]=x;//新元素入栈

return true;

}

//出栈

bool pops(sqstack &s,char& x)

{

if(s.top==-1)//栈空报错

{

return false;

}

x=s.data[s.top];//栈顶元素出栈

s.top--;//指针再减

return true;

}

//查栈顶元素

bool gets(sqstack &s,char& x)

{

if(s.top==-1)//栈空报错

{

return false;

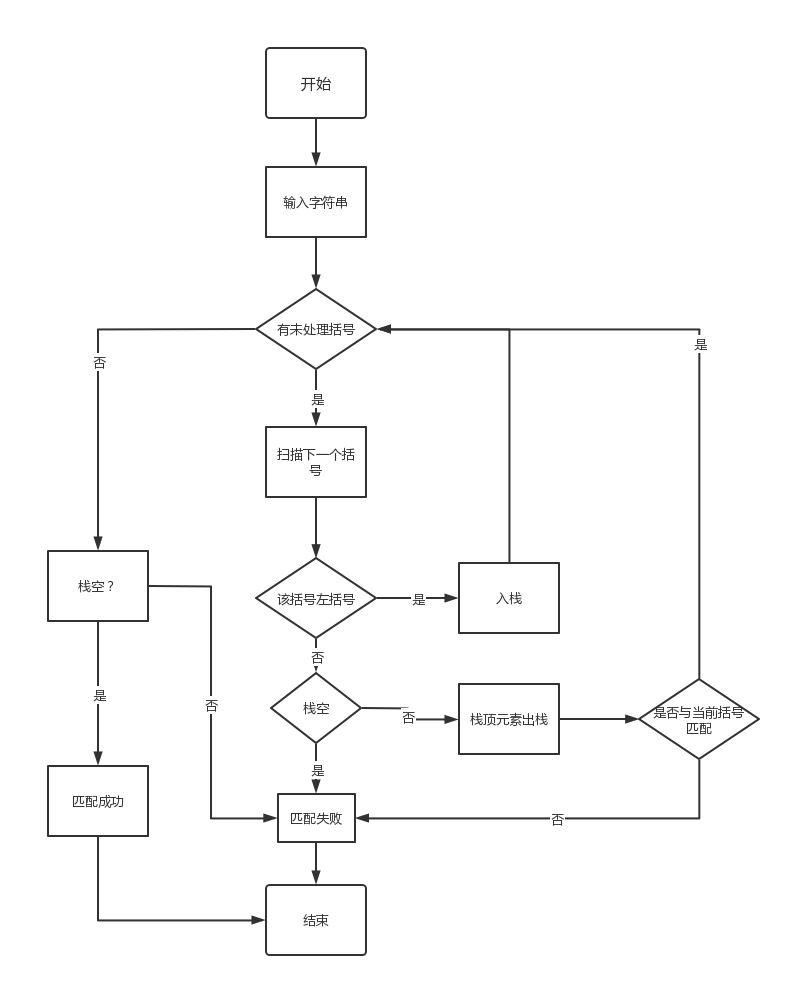
}

x=s.data[s.top];//栈顶元素出栈

return true;

}

**2.4设计图**

****

**2.5程序**

**#include<cstdio>**

**#include<cstdlib>**

**#include<cstring>**

**#include<iostream>**

**#include<cmath>**

**#define MAXSIZE 10086**

**using namespace std;**

**char str[100086];**

**typedef struct**

**{**

**char data[MAXSIZE];**

**int top;//栈顶指针**

**}sqstack;**

**//初始化栈**

**void inits(sqstack &s)**

**{**

**s.top=-1;**

**}**

**//判断栈是否为空**

**bool emptys(sqstack s)**

**{**

**if(s.top==-1)**

**return true;**

**else**

**return false;**

**}**

**//入栈**

**bool pushs(sqstack &s,char& x)**

**{**

**if(s.top==MAXSIZE-1)//栈满报错**

**{**

**return false;**

**}**

**s.top++;//指针先加**

**s.data[s.top]=x;//新元素入栈**

**return true;**

**}**

**//出栈**

**bool pops(sqstack &s,char& x)**

**{**

**if(s.top==-1)//栈空报错**

**{**

**return false;**

**}**

**x=s.data[s.top];//栈顶元素出栈**

**s.top--;//指针再减**

**return true;**

**}**

**//查栈顶元素**

**bool gets(sqstack &s,char& x)**

**{**

**if(s.top==-1)//栈空报错**

**{**

**return false;**

**}**

**x=s.data[s.top];//栈顶元素出栈**

**return true;**

**}**

**//匹配括号函数**

**bool checks(char str[],int len)**

**{**

**sqstack s;**

**inits(s);**

**for(int i=0;i<len;i++)**

**{**

**if(str[i]=='('||str[i]=='['||str[i]=='{')//扫描到右括号**

**{**

**pushs(s,str[i]);//入栈**

**}**

**else//扫描到左括号**

**{**

**if(emptys(s))//栈中没有元素**

**{**

**return false;**

**}**

**char tope;//栈顶元素**

**pops(s,tope);//栈顶元素出栈**

**if(str[i]==')'&&tope!='(')//判断是否匹配**

**{**

**return false;**

**}**

**if(str[i]==']'&&tope!='[')**

**{**

**return false;**

**}**

**if(str[i]=='}'&&tope!='{')**

**{**

**return false;**

**}**

**}**

**}**

**if(emptys(s))//若栈为空则正确**

**return true;**

**else**

**return false;**

**}**

**int main()**

**{**

**cin>>str;**

**if(checks(str,strlen(str)))**

**{**

**cout<<"匹配"<<endl;**

**}**

**else**

**{**

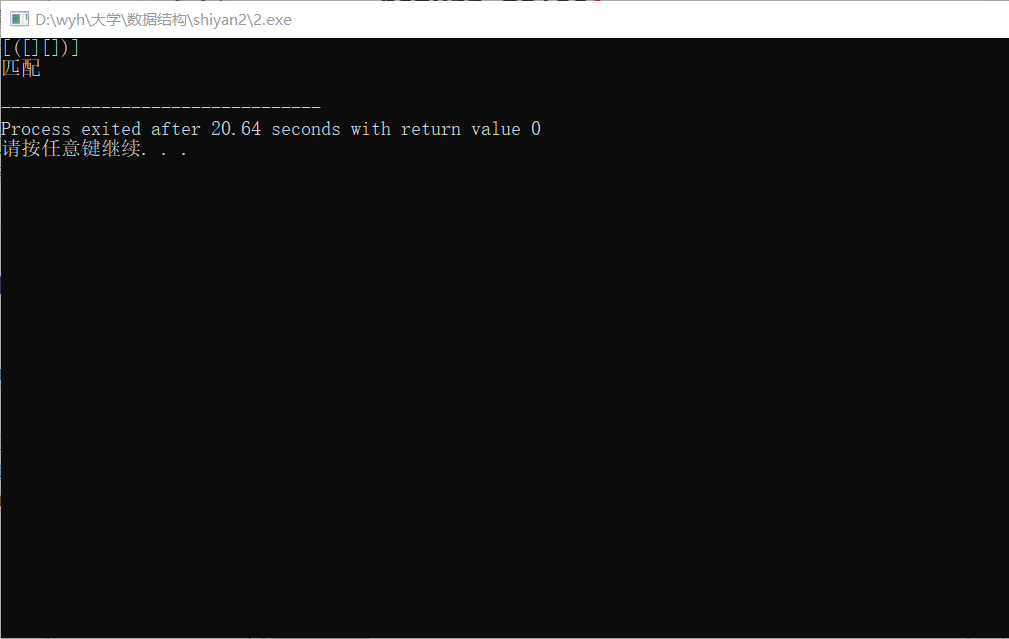
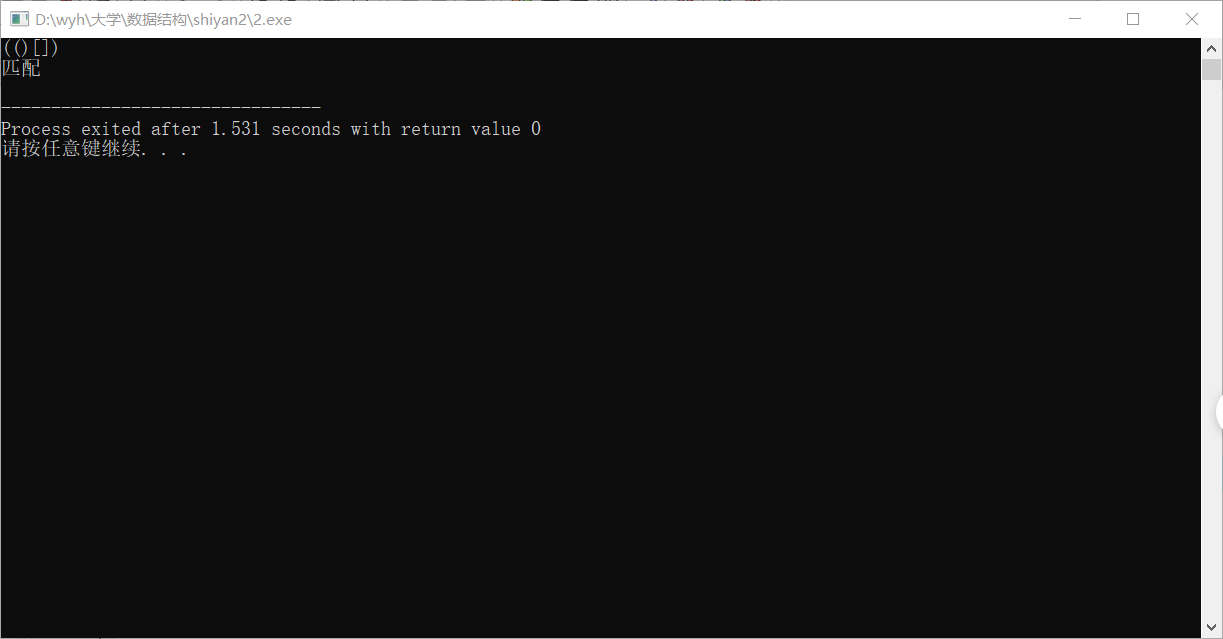
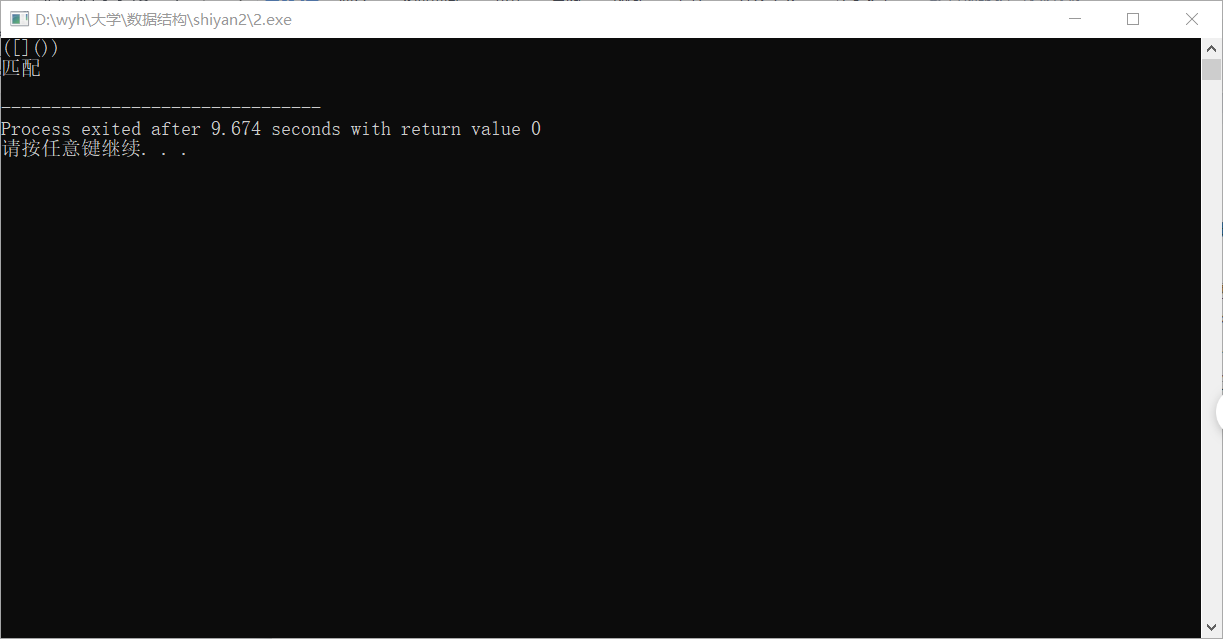
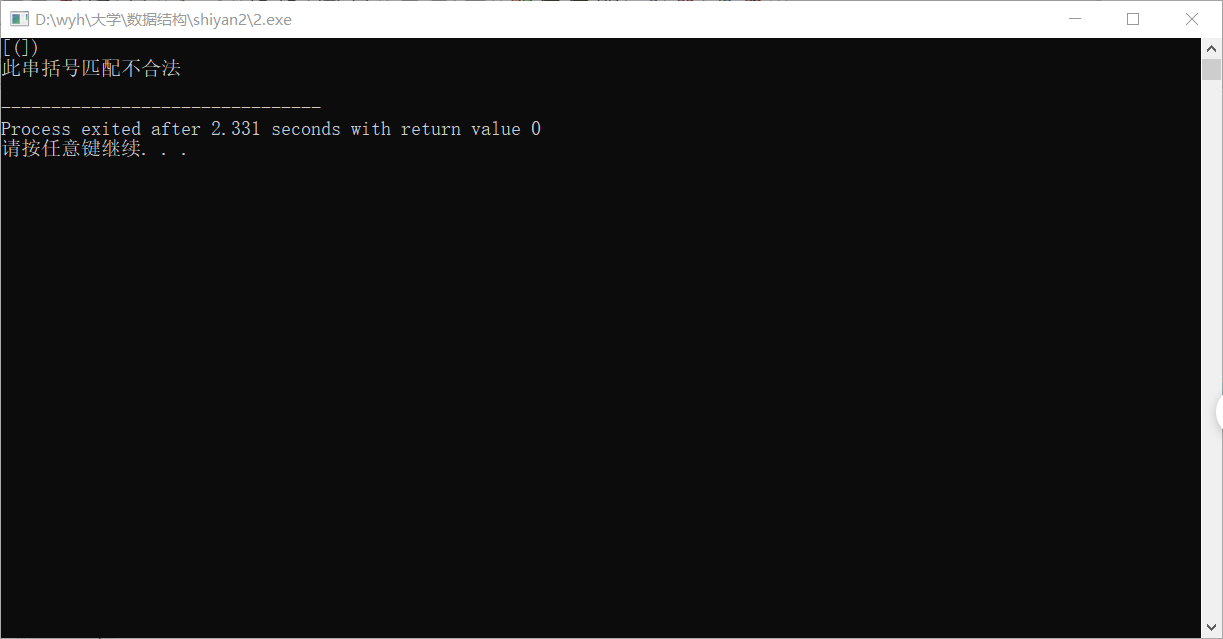
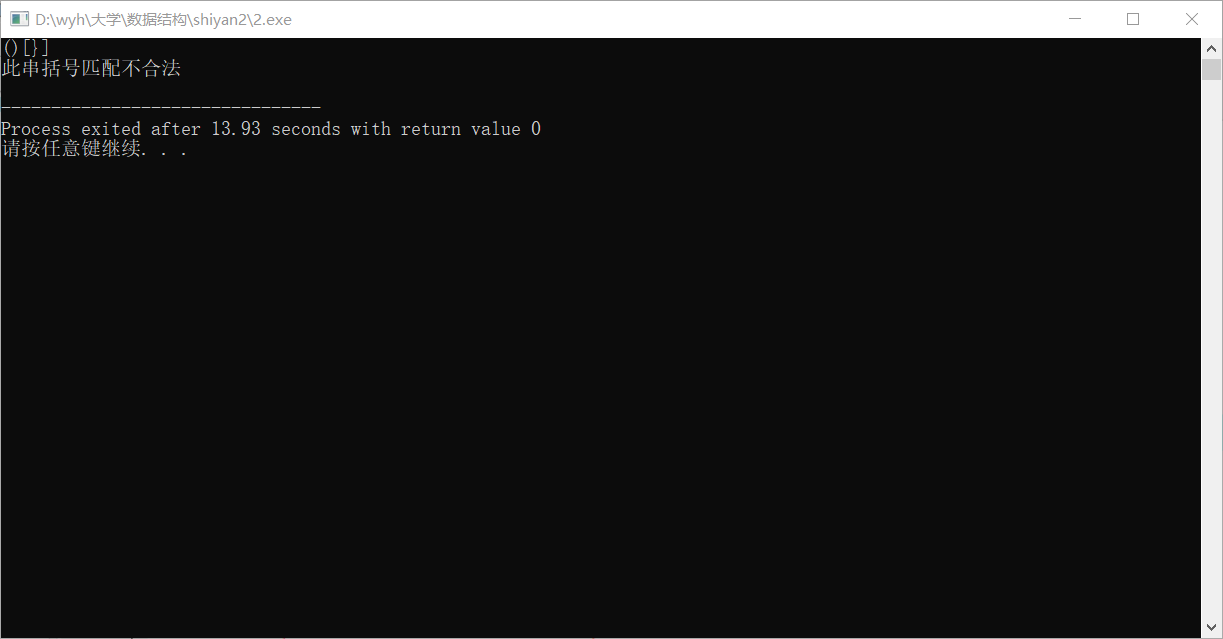
**cout<<"此串括号匹配不合法"<<endl;**

**}**

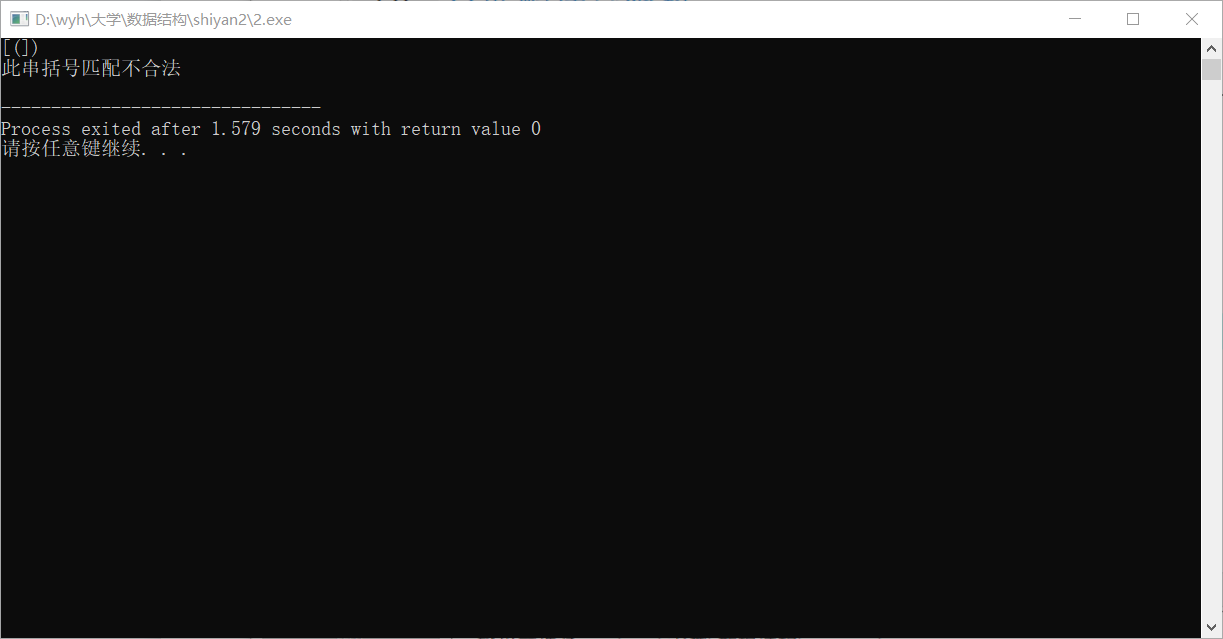
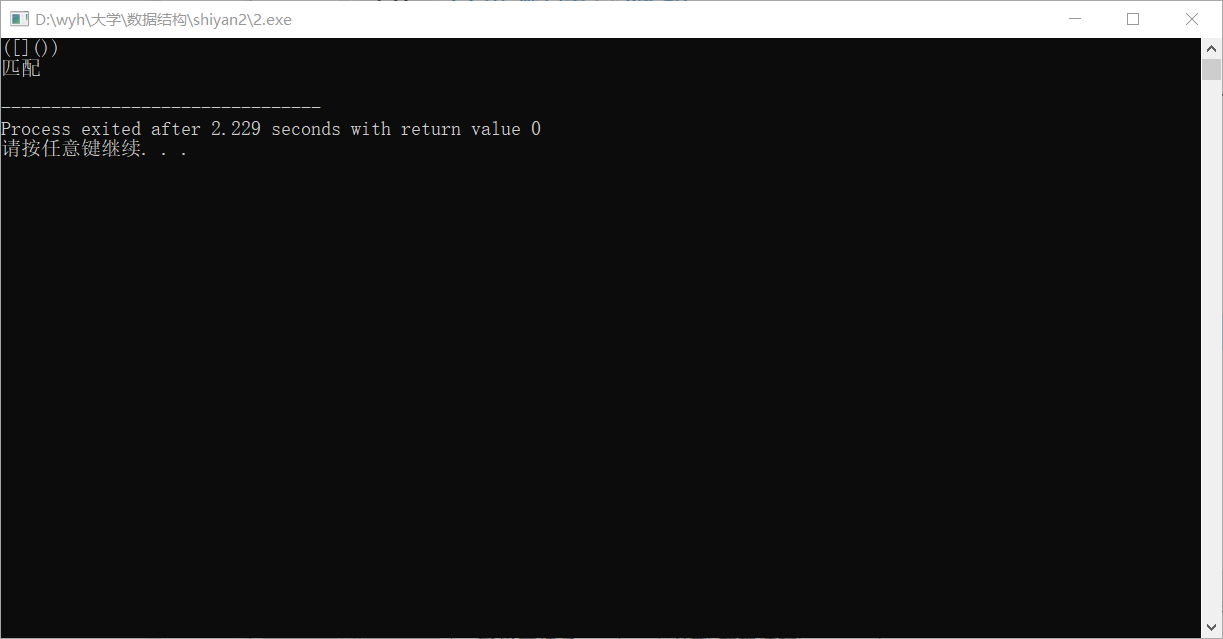
**return 0;**

**}**

**2.6调试过程截图**

****

**2.7运行结果截图**

****

**3选做**

**3.1问题分析**

1. 每一组输入数据包括：汽车“到达”或“离去”信息、汽车牌照号码以及到达或离去的时刻。
2. 输出信息：若是车辆到达，则输出汽车在停车场内或便道上的停车位置；若是车辆离去，则输出汽车在停车场内停留的时间和应交纳的费用。

3.测试数据：n=2，输入数据为：（‘A’,1,5），（‘A’,2,10），（‘D’,1,15），（‘A’,3,20），（‘A’,4,25），（‘A’,5,30），（‘D’,2,35），（‘D’,4,40），（‘E’,0,0）。其中：‘A’表示到达；‘D’表示离去；‘E’表示输入结束**3.2设计方案**

(1)将停车场建立为一个栈

(2)将停车场的车将要开出时再建立一个栈

(3)停车场满时建立一个队列为便道

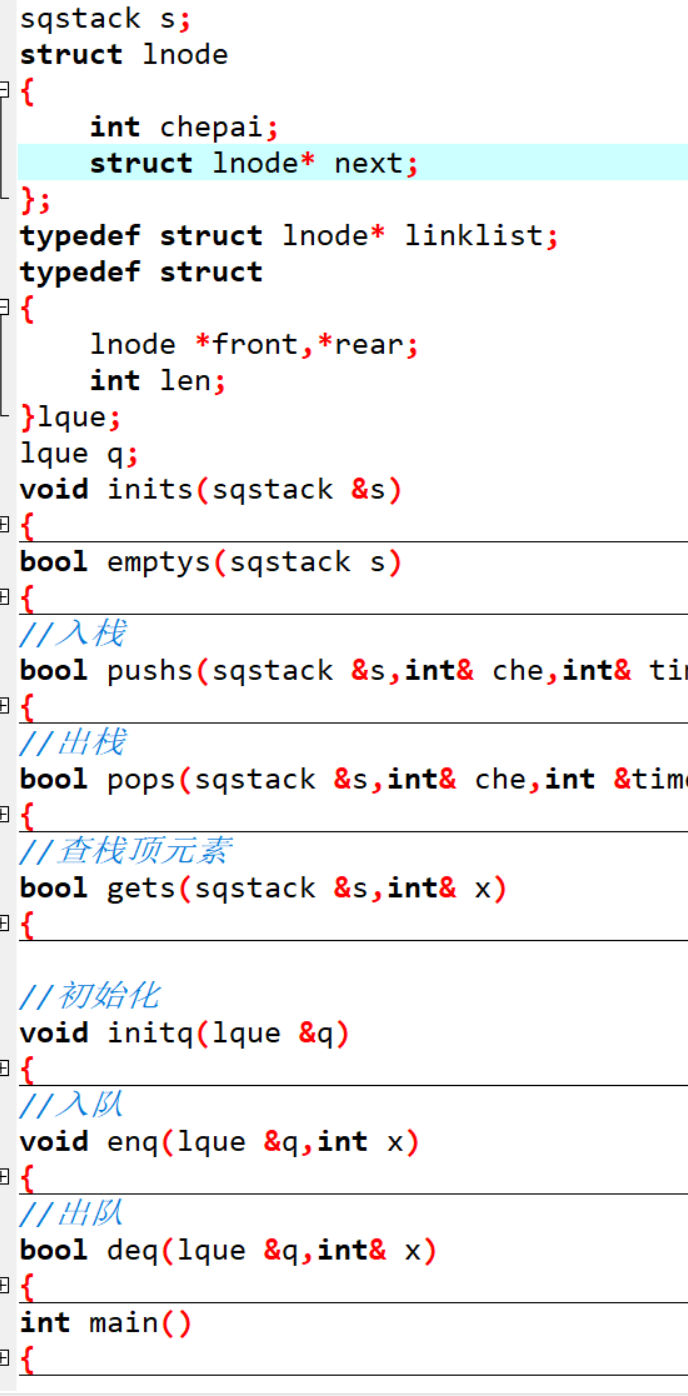
(4)用结构体先定义栈，包括数据和栈顶指针

(5)在创建栈前要先初始化，并实现一个判断栈空的函数，查栈顶元素函数

(6)入栈函数,出栈函数

(7)用结构体先定义链表，以此定义队列，出对入队，初始化

(8)主函数



**3.3算法**

1. 队列

//初始化

void initq(lque &q)

{

q.front=q.rear=(lnode\*)malloc(sizeof(lnode));

q.front->next=NULL;

}

//入队

void enq(lque &q,int x)

{

lnode\* s=(linklist)malloc(sizeof(lnode));

s->chepai=x;

s->next=NULL;

q.rear->next=s;

q.rear=s;

q.len++;

cout<<x<<"停在便道"<<q.len<<endl;

}

//出队

bool deq(lque &q,int& x)

{

if(q.front==q.rear)

{

return false;

}

lnode \*p=q.front->next;

x=p->chepai;//返回头结点元素

q.front->next=p->next;//修改头结点的next

if(q.rear==p)//特殊处理最后一个结点

{

q.rear=q.front;//修改rear

}

free(p);

return true;

}

1. 栈

//初始化

void inits(sqstack &s)

{

s.top=-1;

}

//栈空

bool emptys(sqstack s)

{

if(s.top==-1)

return true;

else

return false;

}

//入栈

bool pushs(sqstack &s,int& che,int& time)

{

s.top++;//指针先加

s.chepai[s.top]=che;//新元素入栈

s.shijian[s.top]=time;

cout<<che<<"停在"<<s.top+1<<endl;

//cout<<"che"<<s.chepai[s.top]<<endl;

//cout<<s.shijian[s.top]<<endl;

return true;

}

//出栈

bool pops(sqstack &s,int& che,int &time)

{

if(s.top==-1)//栈空报错

{

return false;

}

time=s.shijian[s.top];//栈顶元素出栈

che=s.chepai[s.top];

s.top--;//指针再减

return true;

}

//查栈顶元素

bool gets(sqstack &s,int& x)

{

if(s.top==-1)//栈空报错

{

return false;

}

// x=s.data[s.top];//栈顶元素出栈

return true;

}

1. 主函数

int main()

{

cout<<"输入停车场大小"<<endl;

cin>>n;

inits(s);

initq(q);

while(1)

{

cout<<"输入信息："<<endl;

cin>>info;

if(info=='E')

{

break;

}

else

{

if(info=='A')//有车停入

{

cout<<"输入车牌和时间"<<endl;

int che=0,time=0;

cin>> che>>time;

if(s.top==n-1)//栈满进入便道

{

cout<<"停车场已满,请暂时停在便道"<<endl;

enq(q,che);

}

else

{

pushs(s,che,time);

}

}

else if(info=='D')//有车开出

{

int flag=1;

cout<<"输入车牌和时间"<<endl;

int che=0,letime=0,money=0;

cin>>che>>letime;

sqstack s1;

inits(s1);

int che1,time1;

//车出停车场

while(s.top!=-1)

{

pops(s,che1,time1);

if(che1==che)

{

flag=0;

money=(letime-time1)\*2;

cout<<che1<<"需要付的钱"<<money<<endl;

break;

}

pushs(s1,che1,time1);

}

while(s1.top!=-1)

{

pops(s1,che1,time1);

pushs(s,che1,time1);

}

if(flag==0)

{

if(q.len!=0)

{

int che2;//入栈车牌

deq(q,che2);

pushs(s,che2,letime);

}

}

else

{

cout<<"停车场不存在该车"<<endl;

}

}

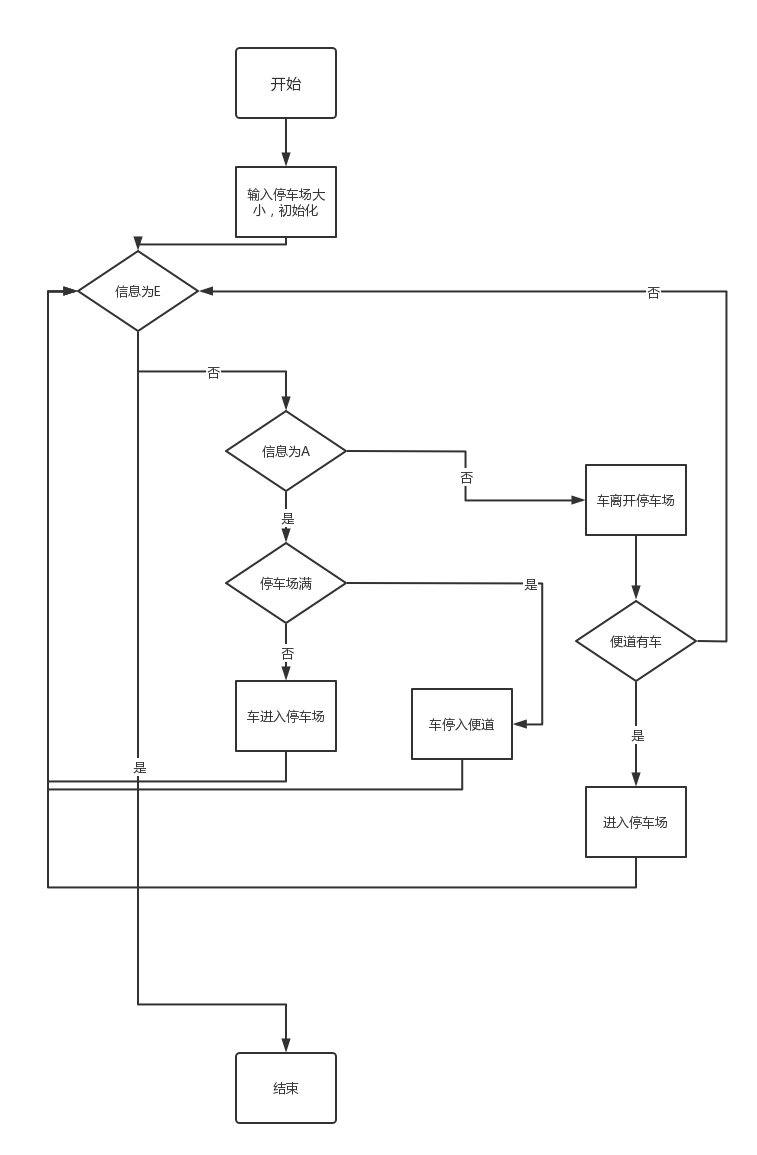
}

}

return 0;

}

**3.4流程图**



**3.5程序**

**#include<cstdio>**

**#include<cstdlib>**

**#include<cstring>**

**#include<iostream>**

**#include<cmath>**

**#define MAXSIZE 10086**

**/\***

**\* 栈 与 队列 两个栈 一个充当停车场 另一个保存出栈的元素顺序 队列用于存放排队的车**

**\* 另外每一组输入数据包括三个数据项：汽车“到达”或“离去”信息、汽车牌照号码及到达或离去的时刻**

**\*/**

**using namespace std;**

**char str[100086];**

**char info;**

**int n;**

**typedef struct**

**{**

**int chepai[MAXSIZE];**

**int shijian[MAXSIZE];**

**int top;//栈顶指针**

**}sqstack;**

**sqstack s;**

**struct lnode**

**{**

**int chepai;**

**struct lnode\* next;**

**};**

**typedef struct lnode\* linklist;**

**typedef struct**

**{**

**lnode \*front,\*rear;**

**int len;**

**}lque;**

**lque q;**

**//初始化**

**void inits(sqstack &s)**

**{**

**s.top=-1;**

**}**

**//栈空**

**bool emptys(sqstack s)**

**{**

**if(s.top==-1)**

**return true;**

**else**

**return false;**

**}**

**//入栈**

**bool pushs(sqstack &s,int& che,int& time)**

**{**

**s.top++;//指针先加**

**s.chepai[s.top]=che;//新元素入栈**

**s.shijian[s.top]=time;**

**cout<<che<<"停在"<<s.top+1<<endl;**

**//cout<<"che"<<s.chepai[s.top]<<endl;**

**//cout<<s.shijian[s.top]<<endl;**

**return true;**

**}**

**//出栈**

**bool pops(sqstack &s,int& che,int &time)**

**{**

**if(s.top==-1)//栈空报错**

**{**

**return false;**

**}**

**time=s.shijian[s.top];//栈顶元素出栈**

**che=s.chepai[s.top];**

**s.top--;//指针再减**

**return true;**

**}**

**//查栈顶元素**

**bool gets(sqstack &s,int& x)**

**{**

**if(s.top==-1)//栈空报错**

**{**

**return false;**

**}**

**// x=s.data[s.top];//栈顶元素出栈**

**return true;**

**}**

**//初始化**

**void initq(lque &q)**

**{**

**q.front=q.rear=(lnode\*)malloc(sizeof(lnode));**

**q.front->next=NULL;**

**}**

**//入队**

**void enq(lque &q,int x)**

**{**

**lnode\* s=(linklist)malloc(sizeof(lnode));**

**s->chepai=x;**

**s->next=NULL;**

**q.rear->next=s;**

**q.rear=s;**

**q.len++;**

**cout<<x<<"停在便道"<<q.len<<endl;**

**}**

**//出队**

**bool deq(lque &q,int& x)**

**{**

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

**{**

**return false;**

**}**

**lnode \*p=q.front->next;**

**x=p->chepai;//返回头结点元素**

**q.front->next=p->next;//修改头结点的next**

**if(q.rear==p)//特殊处理最后一个结点**

**{**

**q.rear=q.front;//修改rear**

**}**

**free(p);**

**return true;**

**}**

**int main()**

**{**

**cout<<"输入停车场大小"<<endl;**

**cin>>n;**

**inits(s);**

**initq(q);**

**while(1)**

**{**

**cout<<"输入信息："<<endl;**

**cin>>info;**

**if(info=='E')**

**{**

**break;**

**}**

**else**

**{**

**if(info=='A')//有车停入**

**{**

**cout<<"输入车牌和时间"<<endl;**

**int che=0,time=0;**

**cin>> che>>time;**

**if(s.top==n-1)//栈满进入便道**

**{**

**cout<<"停车场已满,请暂时停在便道"<<endl;**

**enq(q,che);**

**}**

**else**

**{**

**pushs(s,che,time);**

**}**

**}**

**else if(info=='D')//有车开出**

**{**

**int flag=1;**

**cout<<"输入车牌和时间"<<endl;**

**int che=0,letime=0,money=0;**

**cin>>che>>letime;**

**sqstack s1;**

**inits(s1);**

**int che1,time1;**

**//车出停车场**

**while(s.top!=-1)**

**{**

**pops(s,che1,time1);**

**if(che1==che)**

**{**

**flag=0;**

**money=(letime-time1)\*2;**

**cout<<che1<<"需要付的钱"<<money<<endl;**

**break;**

**}**

**pushs(s1,che1,time1);**

**}**

**while(s1.top!=-1)**

**{**

**pops(s1,che1,time1);**

**pushs(s,che1,time1);**

**}**

**if(flag==0)**

**{**

**if(q.len!=0)**

**{**

**int che2;//入栈车牌**

**deq(q,che2);**

**pushs(s,che2,letime);**

**}**

**}**

**else**

**{**

**cout<<"停车场不存在该车"<<endl;**

**}**

**}**

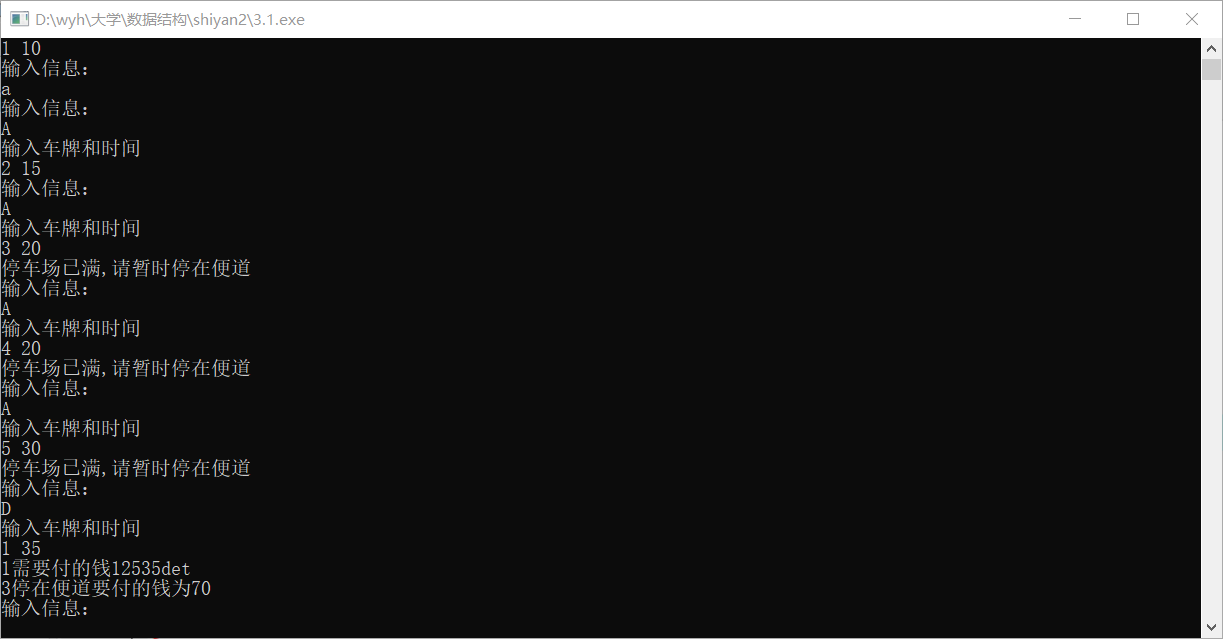
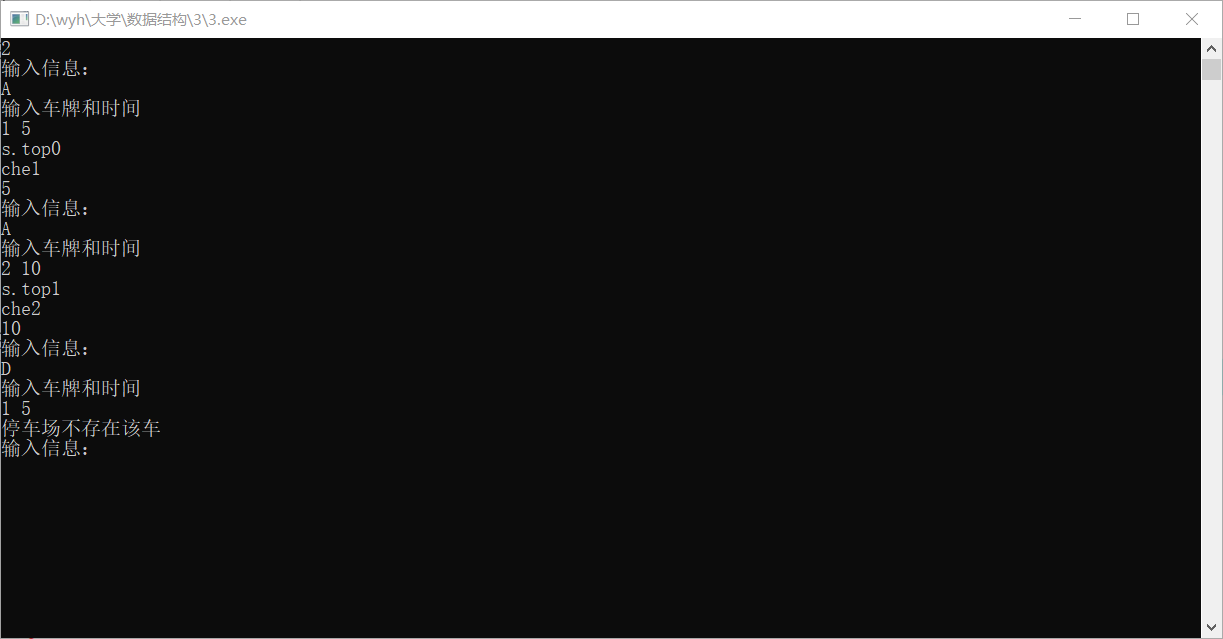
**}**

**}**

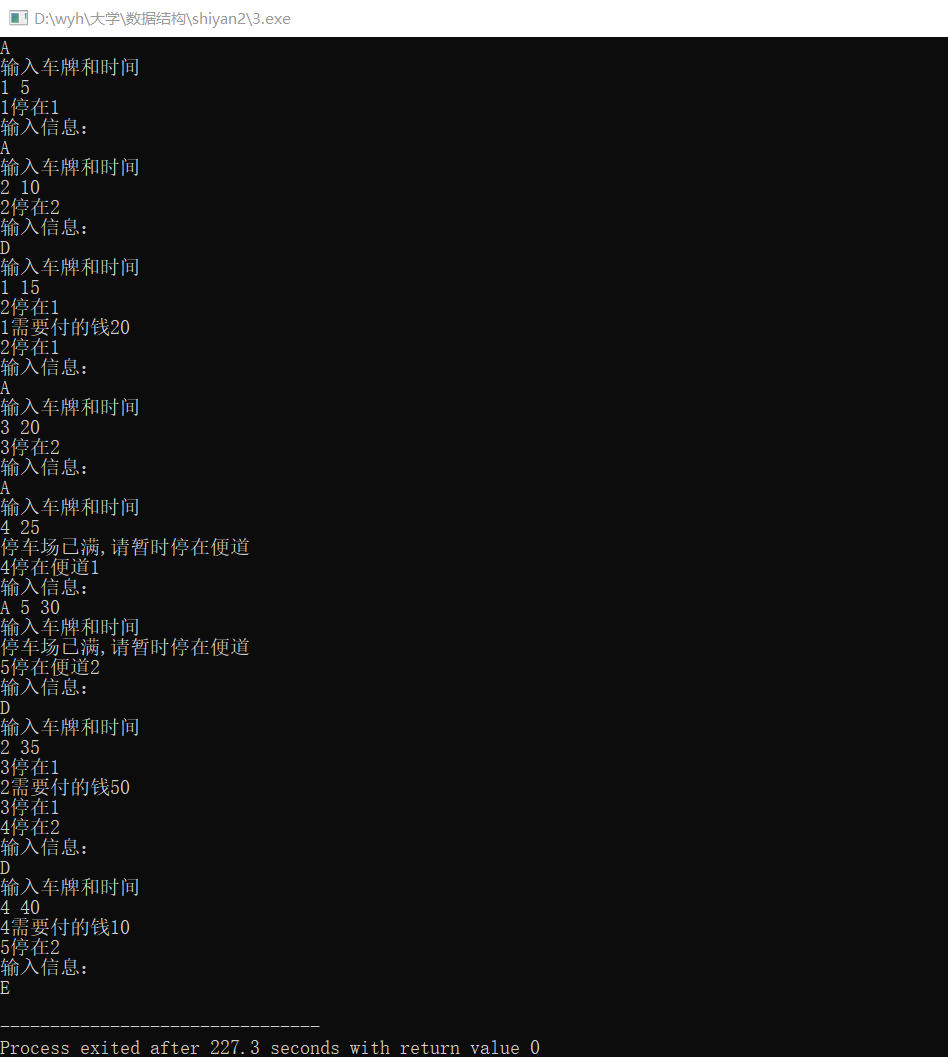
**return 0;**

**}**

**3.6调试过程截图**

****

**3.7运行结果截图**

****

**3.8思考**

**（1）两个栈共享空间，思考应开辟数组的空间是多少？**2\*n-1

**（2） 汽车可有不同种类，则它们的占地面积不同，收费标准也不同，如1辆客车和1.5辆小汽车的占地面积相同，1辆十轮卡车占地面积相当于3辆小汽车的占地面积。**

用车牌判断或者新设置一个输入和变量

**（3） 汽车可以直接从便道上开走，此时排在它前面的汽车要先开走让路，然后再依次排到队尾。**

增加一个判断车是否在便道，车离开后将所有车往后一位

1. **停放在便道上的汽车也收费，收费标准比停放在停车场的车低，请思考如何修改结构以满足这种要求。**

从便道出来时候新加一个判断输出

**#include<cstdio>**

**#include<cstdlib>**

**#include<cstring>**

**#include<iostream>**

**#include<cmath>**

**#define MAXSIZE 10086**

**/\***

**\* 栈 与 队列 两个栈 一个充当停车场 另一个保存出栈的元素顺序 队列用于存放排队的车**

**\* 另外每一组输入数据包括三个数据项：汽车“到达”或“离去”信息、汽车牌照号码及到达或离去的时刻**

**\*/**

**using namespace std;**

**char str[100086];**

**char info;**

**int n;**

**typedef struct**

**{**

**int chepai[MAXSIZE];**

**int shijian[MAXSIZE];**

**int top;//栈顶指针**

**}sqstack;**

**sqstack s;**

**struct lnode**

**{**

**int chepai,time;**

**struct lnode\* next;**

**};**

**typedef struct lnode\* linklist;**

**typedef struct**

**{**

**lnode \*front,\*rear;**

**int len;**

**}lque;**

**lque q;**

**void inits(sqstack &s)**

**{**

**s.top=-1;**

**}**

**bool emptys(sqstack s)**

**{**

**if(s.top==-1)**

**return true;**

**else**

**return false;**

**}**

**//入栈**

**bool pushs(sqstack &s,int& che,int& time)**

**{**

**s.top++;//指针先加**

**s.chepai[s.top]=che;//新元素入栈**

**s.shijian[s.top]=time;**

**cout<<che<<"停在"<<s.top+1<<endl;**

**//cout<<"s.top"<<s.top<<endl;**

**//cout<<"che"<<s.chepai[s.top]<<endl;**

**//cout<<s.shijian[s.top]<<endl;**

**return true;**

**}**

**//出栈**

**bool pops(sqstack &s,int& che,int &time)**

**{**

**if(s.top==-1)//栈空报错**

**{**

**return false;**

**}**

**time=s.shijian[s.top];//栈顶元素出栈**

**che=s.chepai[s.top];**

**s.top--;//指针再减**

**return true;**

**}**

**//查栈顶元素**

**bool gets(sqstack &s,int& x)**

**{**

**if(s.top==-1)//栈空报错**

**{**

**return false;**

**}**

**// x=s.data[s.top];//栈顶元素出栈**

**return true;**

**}**

**void initq(lque &q)**

**{**

**q.front=q.rear=(lnode\*)malloc(sizeof(lnode));**

**q.front->next=NULL;**

**}**

**void enq(lque &q,int x,int time1)**

**{**

**lnode\* s=(linklist)malloc(sizeof(lnode));**

**s->chepai=x;**

**s->time=time1;**

**s->next=NULL;**

**q.rear->next=s;**

**q.rear=s;**

**q.len++;**

**cout<<x<<"停在便道"<<q.len<<endl;**

**}**

**bool deq(lque &q,int& x,int& leti)**

**{**

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

**{**

**return false;**

**}**

**lnode \*p=q.front->next;**

**x=p->chepai;//返回头结点元素**

**int det=leti-p->time;**

**//cout<<det<<"det"<<endl;**

**int qian=2\*det;**

**cout<<x<<"停在便道要付的钱为"<<qian<<endl;**

**q.front->next=p->next;//修改头结点的next**

**if(q.rear==p)//特殊处理最后一个结点**

**{**

**q.rear=q.front;//修改rear**

**}**

**free(p);**

**return true;**

**}**

**void checkq(lque &q,int&che,int &letime)**

**{**

**lnode \*pr=q.front;**

**//x=p->chepai;**

**//int det=leti-p->time;**

**while(pr->next->chepai!=che)**

**{**

**pr=pr->next;**

**}**

**lnode \*p=pr->next;**

**cout<<p->chepai<<"钱为"<<2\*(letime-p->time)<<endl;**

**che=p->chepai;**

**letime=p->time;**

**if(p->next!=NULL)**

**{**

**pr->next=p->next;**

**free(p);**

**}**

**else**

**{**

**pr->next=NULL;**

**free(p);**

**}**

**}**

**int main()**

**{**

**cout<<"输入停车场大小"<<endl;**

**cin>>n;**

**inits(s);**

**initq(q);**

**while(1)**

**{**

**cout<<"输入信息："<<endl;**

**cin>>info;**

**if(info=='E')**

**{**

**break;**

**}**

**else**

**{**

**if(info=='A')**

**{**

**cout<<"输入车牌和时间"<<endl;**

**int che=0,time=0;**

**cin>> che>>time;**

**//che为1，3客车 ，2，4卡车**

**if(s.top==n-1)//栈满进入便道**

**{**

**cout<<"停车场已满,请暂时停在便道"<<endl;**

**enq(q,che,time);**

**}**

**else**

**{**

**pushs(s,che,time);**

**}**

**}**

**else if(info=='D')**

**{**

**int flag=1;**

**cout<<"输入车牌和时间"<<endl;**

**int che=0,letime=0,money=0;**

**cin>>che>>letime;**

**sqstack s1;**

**inits(s1);**

**int che1,time1;**

**//车出停车场**

**while(s.top!=-1)**

**{**

**pops(s,che1,time1);**

**if(che1==che)**

**{**

**flag=0;**

**if(che==1||che==3)**

**{**

**money=(letime-time1)\*5;**

**cout<<che1<<"客车需要付的钱"<<money<<endl;**

**break;**

**}**

**else if(che==2||che==4)**

**{**

**money=(letime-time1)\*7.5;**

**cout<<che1<<"卡车需要付的钱"<<money<<endl;**

**break;**

**}**

**}**

**pushs(s1,che1,time1);**

**}**

**while(s1.top!=-1)**

**{**

**pops(s1,che1,time1);**

**pushs(s,che1,time1);**

**}**

**if(flag==0)**

**{**

**if(q.len!=0)**

**{**

**int che2;//入栈车牌**

**deq(q,che2,letime);**

**pushs(s,che2,letime);**

**}**

**}**

**else**

**{**

**cout<<"车在便道"<<endl;**

**checkq(q,che,letime) ;**

**}**

**}**

**}**

**}**

**return 0;**

**}**

1. **实验心得**

通过这次实验，我对数据结构的应用有了一定了解，加强了对栈与队列的理解，也明白了实际应用中顺序结构和链表栈以及队列的差异，明白这两种各有优缺点。栈是被限制了的链表，对于链表的运用熟悉了，栈的运用也就轻松了很多。在做必做题时只要想清楚基本上都没有遇到什么困难。而在做选做题最大的难题是栈和队列的指针移动的条件需细心考虑，其在循环中的移动情况要考虑充分，以及地址在多个函数调用时的变化，经独立仔细观察和调试修改，在多次修改下才完成。说明对于指针以及链表这块还不是很熟悉，所以要多动手编程，才能熟练灵活的掌握数据结构知识，才能更好的理解掌握数据结构的精髓。从而避免基础语法错误，让代码变得更简洁高效。如此才能准确高效的解决问题。在今后的编程过程中要更注重代码整体设计的提纲记录，用更多的注释，让自己的思路更清晰。