**数据结构课程设计**

**课题名称： 停车场管理系统**

**专业： 软件工程16级 班级： 4班**

**姓名： 杨赫 学号： 29 成绩：**

**指导教师： 曹梅**

**完成日期： 2017 年 12 月 1 日**

任 务 书

|  |
| --- |
| 题目： |
| 设计内容及要求：  1.课程设计任务内容  设计一个停车场管理系统。  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停车场的最北端）,若停车场内已停满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。   1. 课程设计要求   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、汽车牌照号码以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信息为：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（在便道上停留的时间不收费）。栈以顺序结构实现，队列以链表结构实现。 |
| 开发环境：  DEVC++ |
| 实现目标：   1. 熟悉的运用c语言程序编写代码。 2. 实现数据结构相关算法 3. 能够理清整个程序的运行过程并绘画流程图 4. 了解如何定义局部变量和整体变量； 5. 学会上机调试程序，发现问题，并解决 6. 学习用文档书写程序说明 |

摘 要

**课程设计目的**：

通过课程设计，加深对《数据结构》这一课程所学内容的进一步理解与巩固。

通过课程设计，加深对结构化设计思想的理解，能对系统功能进行分析，并设计合理的模块化结构。

通过课程设计，提高程序开发功能，能运用合理的控制流程编写清晰高效的程序。

通过课程设计，训练C程序调试能力，能将一个中小型各级组织系统联调通过。

通过课程设计，开发一个中小型系统，掌握系统研发全过程。

通话课程设计，培养分析问题、解决实际问题的能力。

**实现方法**：通过建立栈来模拟停车场，当栈未满时即停车场内车辆未满时，就可以停放车辆当停车场满了后车就得停靠在便道上等待。建立一个队列来实现队列的功能，

最后根据车辆在停车场停靠的时间来收费，而在便道上停靠的时间里是不收费的。

**系统特点：**

采用一用户一口令的安全机制，只允许管理员输入正确密码，才能正常登录，提高了系统的安全性。

# 关键词

停车场，栈，队列。

目 录

[1 引 言 6](#_Toc20079)

[2 课题分析 7](#_Toc24331)

[3 总体设计 8](#_Toc22017)

[3.1设计思路 8](#_Toc5228)

[3.2模块划分 8](#_Toc22069)

[3.3重要数据结构和变量说明 1](#_Toc9455)0

[3.4关键算法描述 1](#_Toc6898)1

[3.5程序设计流程图 1](#_Toc30469)4

[4 程序运行结果 1](#_Toc14953)7

[5 软件使用说明 1](#_Toc29589)9

[6 程序优缺点 2](#_Toc24436)2

[7 结论 2](#_Toc24436)3

[参考文献 2](#_Toc12639)3

[附录：源代码 2](#_Toc10233)4

|  |  |  |
| --- | --- | --- |
| 学 号 | 姓 名 | 撰写章节 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

1 引 言

课设题目及目标（1-3句）

* 1. **意义及目的**

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。

**1.2 研究内容**

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。

**1.3 国内外先进算法动态**

1. 课题分析

设计一个停车场管理系统。

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停车场的最北端）,若停车场内已停满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。

**【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、汽车牌照号码以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信息为：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（在便道上停留的时间不收费）。栈以顺序结构实现，队列以链表结构实现。

**【选作内容】**

（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，则他们的占地面积不同，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。

# 3 总体设计

## 3.1设计思路

该问题需要以栈和队列作为基本的存储结构，以顺序栈模拟停车场，以链队列模拟车场外的便道。汽车进入停车场，即是在顺序栈上执行进栈操作，退出停车场即是在顺序栈上执行出栈操作；汽车进入便道，即是在链队列上执行入队操作，退出便道即是在链队列上执行出队操作。

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。设要删除的元素在顺序表St中位置为i，则从i到top之间的全部元素进入到一个临时栈St1中, 其次再删除该元素，然后将临栈St1的元素按照“先进后出”的原则重新回到St中。若链队不空，则使队头进栈St，并以当前时刻开始计费。

程序需要构造两个顺序栈St和St1，其中St用于模拟停车场，St1用作临时栈，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。此外还需要构造一个链队列Qu用于模拟便道。

## 3.2模块划分

1. 对于子函数模块，则调用顺序栈的基本操作和链队列的基本操作。

如下：

/\*顺序栈的基本运算算法\*/

void InitStack(SqStack \*&s)

int StackEmpty(SqStack \*s)

int StackFull(SqStack \*s)

/\*S中的插入新元素\*/

int Push(SqStack \*&s,int e1,int e2)

/\*删除S的栈顶元素，并用e1,e2返回其值\*/

int Pop(SqStack \*&s,int &e1,int &e2)

void DispStack(SqStack \*s)

/\*以下为链队列的基本运算算法\*/

void InitQueue(LiQueue \*&q)

int QueueLength(LiQueue \*q)

int QueueEmpty(LiQueue \*q)

void enQueue(LiQueue \*&q, int e,int w)

int deQueue(LiQueue \*&q,int &e,int &w)

void DisplayQueue(LiQueue \*q)

2.主程序模块

void main()

{

初始化；

do {

接受命令；

处理命令；

} while(命令!=”退出”)；

}

1. 局部模块设计

1.停车场车离开倒车到临时栈St1中，将e1,e2插入到临时栈中

for (j=i;j<=St->top;j++)

{

Pop(St,e1,e2); /\*e1,e2用来返回被删元素的车牌号和停车时刻\*/

Push(St1,e1,e2); /\*倒车到临时栈St1中，将e1,e2插入到临时栈中\*/

}

Pop(St,e1,e2);

1. 对小汽车而言：当前时刻 减去 该车当时停车的时刻，再乘以价格就是费用,而对于客车和卡车而言，就要乘以kind倍小汽车的价格

printf("\n车牌号为%d的汽车停车时间为：%d。停车费用为:%d\n",no,time-e2,(time-e2)\*Price\*kind);

3.. 将临时栈St1重新回到St中 /\*该汽车离开\*/

while (!StackEmpty(St1)) /\*将临时栈St1重新回到St中\*/

{

Pop(St1,e1,e2);

Push(St,e1,e2);

｝

4.输出停车场车辆

void DispStack(SqStack \*s)

{

int i;

for (i=0;i<=s->top;i++)

{

printf(" %d\t %d\t\t\t %d\t",i+1,s->CarNo[i],s->CarTime[i]);

printf("\n");

}

}

5.输出队列车辆显示

void DisplayQueue(LiQueue \*q)

{

int i=1;

QNode \*p=q->front;

while (p!=NULL)

{

printf(" %d\t %d\t\t\t %d\t",i,p->CarNo,p->CarTime);

printf("\n");

p=p->next ;

i++;

}

}

## 3.3重要数据结构和变量说明

/\*定义顺序栈类型\*/

typedef struct

{

int CarNo[N]; /\*车牌号\*/

int CarTime[N]; /\*进场时间\*/

int top; /\*栈指针\*/

} SqStack; /\*定义顺序栈类型\*/

/\*定义链队类型\*/

typedef struct qnode

{

int CarNo; /\*车牌号\*/

struct qnode \*next;

} QNode;

typedef struct

{

QNode \*front; /\*队首和队尾指针\*/

QNode \*rear;

} LiQueue;

## 3.4关键算法描述

1.停车场车入栈

int Push(SqStack \*&s,int e1,int e2) //入栈

{

if (s->top==N-1)

return 0;

s->top++;

s->CarNo[s->top]=e1;//车牌号入栈

s->CarTime[s->top]=e2;//时间入栈

return 1;

}

2.出栈删除S的栈顶元素，并用e1,e2返回其值

int Pop(SqStack \*&s,int &e1,int &e2)

{

if (s->top==-1)

return 0;

e1=s->CarNo[s->top];

e2=s->CarTime[s->top];

s->top--;

return 1;

}

3.输出停车场使用状况

void DispStack(SqStack \*s)

{

int i;

for (i=0;i<=s->top;i++)

{

printf(" %d\t %d\t\t\t %d\t",i+1,s->CarNo[i],s->CarTime[i]);

printf("\n");

}

}

1. 进队列（汽车进入便道）

void enQueue(LiQueue \*&q, int e,int w)

{

QNode \*s;

s=(QNode \*)malloc(sizeof(QNode));

s->CarNo=e;

s->CarTime=w;

s->next=NULL;

if (q->rear==NULL) /\*若链队为空,则新结点是队首结点又是队尾结点\*/

q->front=q->rear=s;

else

{

q->rear->next=s; /\*将\*s结点链到队尾,rear指向它\*/

q->rear=s;

}

}

1. 出队列汽车出便道

int deQueue(LiQueue \*&q,int &e,int &w)

{

QNode \*t;

if (q->rear==NULL) /\*队列为空\*/

return 0;

if (q->front==q->rear) /\*队列中只有一个结点时\*/

{

t=q->front;

q->front=q->rear=NULL;

}

else /\*队列中有多个结点时\*/

{

t=q->front;

q->front=q->front->next;

}

e=t->CarNo;

w=t->CarTime;

free(t);

return 1;

}

1. 便道使用情况

void DisplayQueue(LiQueue \*q)

{

int i=1;

QNode \*p=q->front;

while (p!=NULL)

{

printf(" %d\t %d\t\t\t %d\t",i,p->CarNo,p->CarTime);

printf("\n");

p=p->next ;

i++;

}

}

1. 判断栈与队列是否为空和满

int StackEmpty(SqStack \*s)//判断栈是否为空

{

return(s->top==-1);

}

int StackFull(SqStack \*s)//判断栈是否已满（停车场是否已满）

{

return(s->top==N-1);

}

int QueueLength(LiQueue \*q)

{

int n=0;

QNode \*p=q->front;

while (p!=NULL)

{

n++;

p=p->next;

}

return(n);

}

//判断队列是否为空

int QueueEmpty(LiQueue \*q)

{

if (q->rear==NULL)

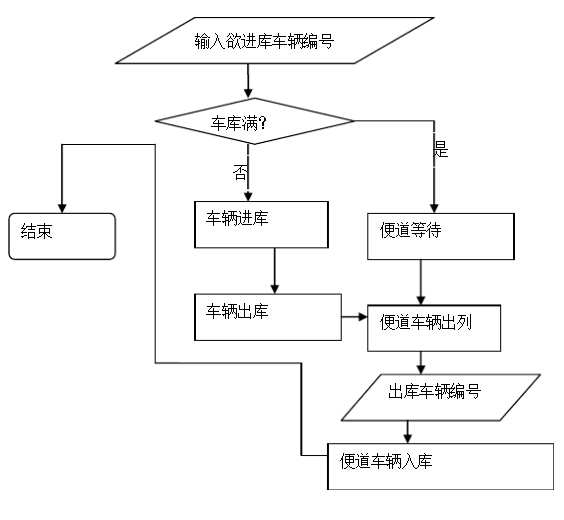
return 1;

else

return 0;

}

## 3.5程序设计流程图

****

(1) 功能模块说明：停车场管理系统含有三个模块，即：车辆到达、离开、列表显示



（2）以模块为单位分析算法

1、“到达”模块：到达时有两种情况，即车场是否满，未满则直接进入停车场；满时，到便道等待。如图2。



2.“离开”模块：离开时，当车库为空时，提示没有车，结束；否则车辆离开。如

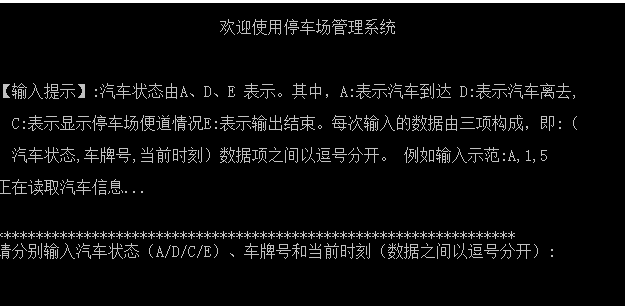


3. “显示”模块：显示模块有两个显示选项，即：车场与便道。如图4。

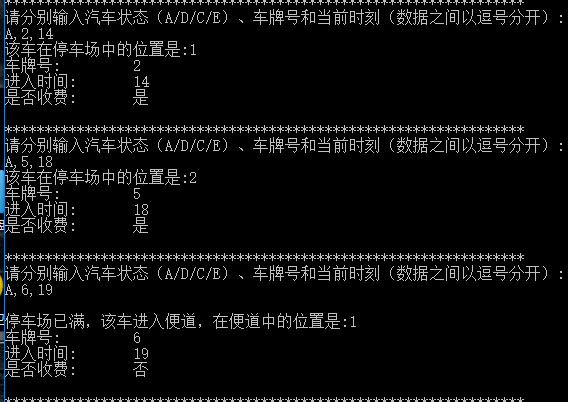


# 程序运行结果

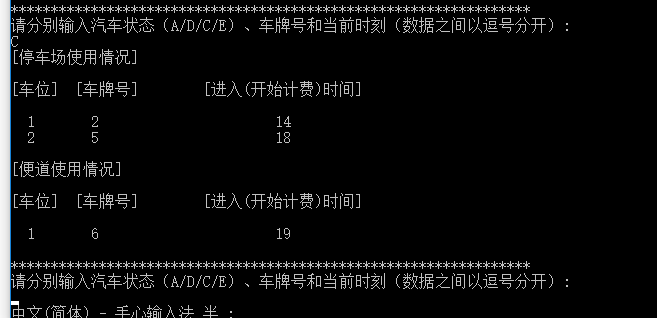
1.主程序菜单页面

****

1. 停车场输入页面

****

1. **停车场便道使用情况页面**

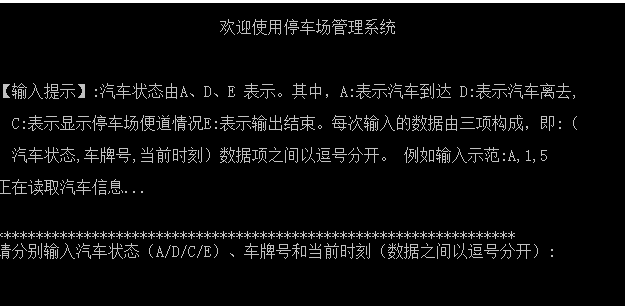
****

1. **计费离开页面**

****

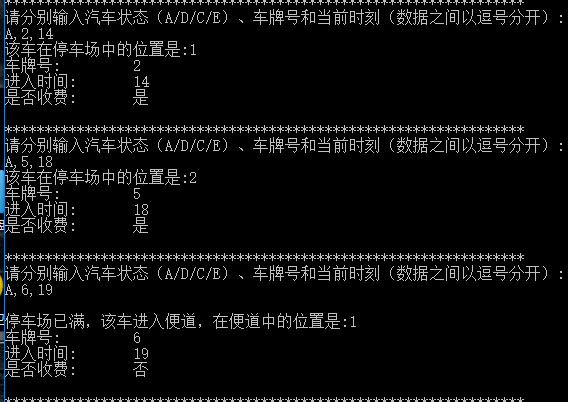
# 5 软件使用说明

1. **开始是显示页面如图一**

****

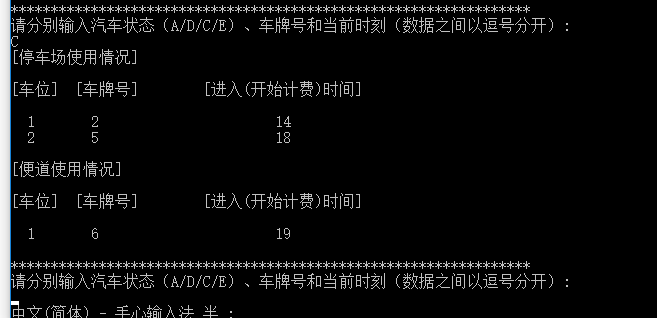
**图一**

1. **进入三辆车车牌号于是否收费如图2由于设定停车场只能停两辆车所以第三辆进入便道并不收费。**

****

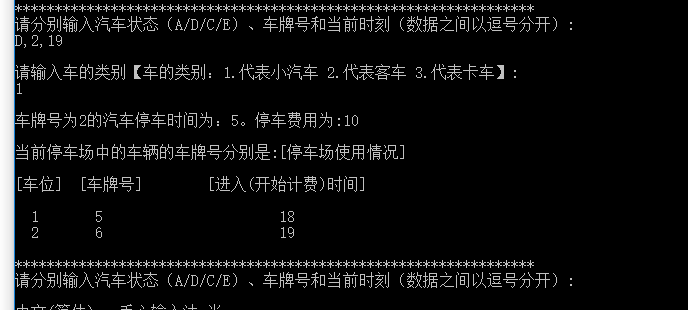
**图二**

**3.显示停车场和便道的使用情况如图三**

****

**图三**

1. **汽车开出停车场并且选择汽车类型计算费用便道中的车开入停车场开始计费如图四**

****

**图四**

**5.输入E退出系统**

****

**图五**

**6.程序优缺点分析及优化**

**1.程序的优点**

在程序中设置了kind变量，用于保存车的类别，便于计算不同类别车的停车费用，如下程序段：

printf("\n请输入车的类别【车的类别：1.代表小汽车 2.代表客车 3.代表卡车】:\n");

scanf("%d",&kind);

其中kind可取值1,2,3；若kind取2，则表示一辆客车单位时间内的停车费用是一辆小汽车的2倍，若kind取3，则表示一辆卡车单位时间内的停车费用是一辆小汽车的3倍。

当然printf("\n请输入车的类别【车的类别：1.代表小汽车 2.代表客车 3.代表卡车】:\n");中的1.2.3也可以根据实际情况改变。比如，若实际中，一辆小汽车单位时间内的停车费用

是一辆客车的2倍，一辆卡车单位时间内的停车费用是一辆客车的4倍，则可以改成：

printf("\n请输入车的类别【车的类别：1.代表客车2.代表小汽车 4.代表卡车】:\n");

则kind可取值1,2,4； kind取1时对应的是客车，表示计算停车费用时以一辆客车单位时间内的停车费用为基数，若kind取2，则表示一辆小汽车单位时间内的停车费用是一辆客车的2倍，若kind取4，则表示一辆卡车单位时间内的停车费用是一辆客车的4倍。

1. **程序的缺点**

typedef struct qnode

{

int CarNo; /\*车牌号\*/

int CarTime; /\*进场时间\*/

struct qnode \*next;

} QNode;

由于字符型数据定义结构体总出错，所以运用整型来定义了车牌号与时间从而无法像真正的停车管理系统一样能够真实的记录准确的时间，和车牌号码，只能实现功能，实际运用时略差一些。

# 7.结 论

1. 该实验涉及到顺序栈的建立、插入、删除等操作，涉及到了链队列的建立、插入、删除等操作。 做这个实验，加深了我对以上知识点的认识和理解。

(2) 提高了 C 语言编程的能力。在程序设计过程中，需要经过反复地编写，调试，运行，发现问题并解决问题，在这次实验的设计中，我加深对程序的了解，提高自己的实际动手能力和独立思考的能力同时我也学会了综合以前学到的基本知识来解决较大问题的方法。

（3）我也深刻地认识到了《数据结构》这门课程的重要性。“数据结构”在计算机科学中是一门综合性的专业基础课。数据结构的研究不仅涉及到计算机硬件的研究，而且和计算机软件的研究有着更密切的关系，无论是编译程序还是操作系统，都涉及到数据元素在存储器中的分配问题。在研究信息检索时也必须 考虑如何组织数据，以便使查找和存取数据元素更为方便。可以认为数据结构是介于数学、计算机硬件和计算机软件三者之间的一个核心内容，是从事计算机科学研究及其应 用的科技工作者必须掌握的重要内容

# 参考文献

1. 谭浩强.《C程序设计（第三版）》，清华大学出版社，2005.7

[2] 郭翠英.《C语言课程设计案例精编》, [中国水利水电出版社](http://www.youlu.net/publisher/5084" \t "_blank),2004.3

[3] 朱振元.《数据结构（面向对象语言描述）》，清华大学出版社，2004.2

[4] 张翔.《C语言函数大全》，电子工业出版社，2002.4

[5]徐金梧.《TURBO C实用大全》，机械工业出版社，1996.5

# 附录：源代码

#include <stdio.h>

#include <malloc.h>

#define N 2 /\*停车场内最多的停车数\*/

#define Price 2 /\*每单位停车费用\*/

typedef struct

{

int CarNo[N]; /\*车牌号\*/

int CarTime[N]; /\*进场时间\*/

int top; /\*栈指针\*/

} SqStack; /\*定义顺序栈类型\*/

/\*定义链队类型\*/

typedef struct qnode

{

int CarNo; /\*车牌号\*/

int CarTime; /\*进场时间\*/

struct qnode \*next;

} QNode;

typedef struct

{

QNode \*front; /\*队首和队尾指针\*/

QNode \*rear;

} LiQueue;

/\*顺序栈的基本运算算法\*/

void InitStack(SqStack \*&s)

{

s=(SqStack \*)malloc(sizeof(SqStack));

s->top=-1;

}

int StackEmpty(SqStack \*s)//判断栈是否为空

{

return(s->top==-1);

}

int StackFull(SqStack \*s)//判断栈是否已满（停车场是否已满）

{

return(s->top==N-1);

}

/\*S中的插入新元素\*/

int Push(SqStack \*&s,int e1,int e2) //入栈

{

if (s->top==N-1)

return 0;

s->top++;

s->CarNo[s->top]=e1;//车牌号入栈

s->CarTime[s->top]=e2;//时间入栈

return 1;

}

/\*删除S的栈顶元素，并用e1,e2返回其值\*/

int Pop(SqStack \*&s,int &e1,int &e2)

{

if (s->top==-1)

return 0;

e1=s->CarNo[s->top];

e2=s->CarTime[s->top];

s->top--;

return 1;

}

/\*输出停车场使用状况\*/

void DispStack(SqStack \*s)

{

int i;

for (i=0;i<=s->top;i++)

{

printf(" %d\t %d\t\t\t %d\t",i+1,s->CarNo[i],s->CarTime[i]);

printf("\n");

}

}

/\*以下为链队列的基本运算算法\*/

void InitQueue(LiQueue \*&q)

{

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

q->front=q->rear=NULL;

}

int QueueLength(LiQueue \*q)

{

int n=0;

QNode \*p=q->front;

while (p!=NULL)

{

n++;

p=p->next;

}

return(n);

}

//判断队列是否为空

int QueueEmpty(LiQueue \*q)

{

if (q->rear==NULL)

return 1;

else

return 0;

}

void enQueue(LiQueue \*&q, int e,int w)

{

QNode \*s;

s=(QNode \*)malloc(sizeof(QNode));

s->CarNo=e;

s->CarTime=w;

s->next=NULL;

if (q->rear==NULL) /\*若链队为空,则新结点是队首结点又是队尾结点\*/

q->front=q->rear=s;

else

{

q->rear->next=s; /\*将\*s结点链到队尾,rear指向它\*/

q->rear=s;

}

}

int deQueue(LiQueue \*&q,int &e,int &w)

{

QNode \*t;

if (q->rear==NULL) /\*队列为空\*/

return 0;

if (q->front==q->rear) /\*队列中只有一个结点时\*/

{

t=q->front;

q->front=q->rear=NULL;

}

else /\*队列中有多个结点时\*/

{

t=q->front;

q->front=q->front->next;

}

e=t->CarNo;

w=t->CarTime;

free(t);

return 1;

}

/\*输出便道使用情况\*/

void DisplayQueue(LiQueue \*q)

{

int i=1;

QNode \*p=q->front;

while (p!=NULL)

{

printf(" %d\t %d\t\t\t %d\t",i,p->CarNo,p->CarTime);

printf("\n");

p=p->next ;

i++;

}

}

int main()

{

char choose; /\*用于选择命令\*/

int no,e1,time,e2,kind; /\*用于存放车牌号、当前停车时刻\*/

int i,j;

SqStack \*St,\*St1; /\*临时栈St1,当停车场中间的车要推出去时，用于倒车\*/

LiQueue \*Qu;

InitStack(St);

InitStack(St1);

InitQueue(Qu);

puts("\n 欢迎使用停车场管理系统 ");

puts("\n ");

puts("\n【输入提示】:汽车状态由A、D、E 表示。其中，A:表示汽车到达 D:表示汽车离去, ");

puts("\n C:表示显示停车场便道情况E:表示输出结束。每次输入的数据由三项构成，即:（ ");

puts("\n 汽车状态,车牌号,当前时刻）数据项之间以逗号分开。 例如输入示范:A,1,5 ");

puts("\n正在读取汽车信息...\n");

do

{

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

printf("\n请分别输入汽车状态（A/D/C/E）、车牌号和当前时刻（数据之间以逗号分开）:\n");

scanf(" %c,%d,%d",&choose,&no,&time);

switch(choose)

{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

汽车到达

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

case 'A':

case 'a':

if (!StackFull(St)) /\*停车场不满\*/

{

Push(St,no,time);

printf("该车在停车场中的位置是:%d\n",St->top+1);

printf( "车牌号:\t\t%d\n", no );

printf( "进入时间:\t%d\n",time );

puts( "是否收费:\t是" );

}

else /\*停车场满\*/

{ enQueue(Qu,no,time);

printf("\n停车场已满，该车进入便道，在便道中的位置是:%d\n",QueueLength(Qu));

printf( "车牌号:\t\t%d\n", no );

printf( "进入时间:\t%d\n",time );

puts( "是否收费:\t否" );

}

break;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 汽车离开 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

case 'D':

case 'd':

printf("\n请输入车的类别【车的类别：1.代表小汽车 2.代表客车 3.代表卡车】:\n");

scanf("%d",&kind);

for (i=0;i<=St->top && St->CarNo[i]!=no;i++);

if (i>St->top) /\*要离开的汽车在便道上\*/

{ /\*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\*/

while (Qu->front->CarNo!=no )

{

enQueue(Qu,Qu->front->CarNo,Qu->front->CarTime );

// deQueue(Qu,Qu->front->CarNo );

Qu->front = Qu->front->next ;

}

deQueue(Qu,no,time);

printf("\n便道上车牌号为%d的汽车已离开!\n",no);

puts( "[便道使用情况]\n" );

puts( "[车位]\t[车牌号]\t[进入(开始计费)时间]\n");

DisplayQueue(Qu);

printf("\n");

}

else /\*要离开的汽车在停车场中\*/

{

for (j=i;j<=St->top;j++)

{

Pop(St,e1,e2); /\*e1,e2用来返回被删元素的车牌号和停车时刻\*/

Push(St1,e1,e2); /\*倒车到临时栈St1中，将e1,e2插入到临时栈中\*/

}

Pop(St,e1,e2); /\*该汽车离开\*/

printf("\n车牌号为%d的汽车停车时间为：%d。停车费用为:%d\n",no,time-e2,(time-e2)\*Price\*kind);

/\*对小汽车而言：当前时刻 减去 该车当时停车的时刻，再乘以价格就是费用,而对于客车和卡车而言，就要乘以kind倍小汽车的价格\*/

while (!StackEmpty(St1)) /\*将临时栈St1重新回到St中\*/

{Pop(St1,e1,e2);

Push(St,e1,e2);

}

if (!QueueEmpty(Qu)) /\*队不空时,将队头进栈St\*/

{

deQueue(Qu,e1,time);

Push(St,e1,time); /\*以当前时间开始计费\*/

}

printf("\n当前停车场中的车辆的车牌号分别是:");

puts( "[停车场使用情况]\n" );

puts( "[车位]\t[车牌号]\t[进入(开始计费)时间]\n"); //输出停车场中的车辆

DispStack(St);

}

break;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 查看停车场 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

case 'C':

case 'c':

if (!StackEmpty(St)) //显示停车场情况

{

puts( "[停车场使用情况]\n" );

puts( "[车位]\t[车牌号]\t[进入(开始计费)时间]\n"); //输出停车场中的车辆

DispStack(St);

printf("\n");

}

else printf("\n当前停车场中无车辆\n\n");

if (!QueueEmpty(Qu))

{

puts( "[便道使用情况]\n" );

puts( "[车位]\t[车牌号]\t[进入(开始计费)时间]\n");

DisplayQueue(Qu);

}

else printf("\n当前便车道中无车辆\n\n");

break;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 结束 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

case 'E':

case 'e':

printf("\n正在退出系统...\n");

break;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 结束 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

default: /\*其他情况\*/

printf("输入的命令错误!\n");

break;}

} while(choose!='E'&&choose!='e');

}