目录

实验题目：停车场管理系统

一、要解决的问题

停车场是一条可以停放 n 辆车的狭窄通道，且只有一个大门汽车停放安到达时间的先后依次由北向南排列（大门在最南端，最先到达的第一辆车停在最北端）若停车场已经停满 n 辆车，后来的汽车在便道上等候，一旦有车开走，排在便道上的第一辆车可以开入；当停车场的某辆车要离开时，停在他后面的车要先后退为他让路，等它开出后其他车在按照原次序开入车场，每两停在车场的车要安时间长短缴费。 要求：以栈模拟停车场，以队列车场外的便道，按照从终端输入的数据序列进行模拟管理。每一组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。对每一组数据进行操作后的信息为：若是车辆到达，则输出汽车在停车场的内或便道上的位置：若是车辆离去则输出汽车在停车场内的停留时间和应缴纳的费用（在便道上的停留时间不收费）。栈以顺序结构实现，队列以链表结构实现。

二、基本要求

（1）界面友好，函数功能要划分好

（2）总体设计应画一流程图

（3）程序要加必要的注释

（4）要提供程序测试方案。

三、算法基本思想描述

由于停车场是一个狭窄通道，而且只有一个大门可供汽车进出，问题要求汽车停车场内按车辆到达时间的先后顺序，依次由北向南排列。由此很容易联想到数据结构中的堆栈模型，因此可首先设计一个堆栈，以堆栈来模拟停车场，我设计用顺序存储结构来存储停车场内的车辆信息，并给车辆按进栈顺序编号，当停车场内某辆车要离开时，在他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停车场。这是个一退一进的

过程，而且让道的汽车必须保持原有的先后顺序，因此可再设计一个堆栈，以之来暂时存放为出站汽车暂时让道的汽车。当停车场满后，继续进来的汽车需要停放在停车场旁边的便道上等候，若停车场有汽车开走，则按排队的先后顺序依次进站，最先进入便道的汽车将会最先进入停车场，这完全是一个先进先出模型，因此可设计一个队列来模拟便道，队列中的数据元素设计成汽车的车牌号，并以链表的形式存储。另外，停车场根据汽车在停车场内停放的总时长来收费的，在便道上的时间不计费，因此必须记录车辆进入停车场时的时间和车辆离开停车场时的时间，然后计算、显示费用情况。

四、详细设计

1. 数据结构的设计

（1）车辆信息的表示

车辆可看成是一个节点，设计成一个结构体，车辆信息包括：车牌号码，车辆的进站时间和离开停车的时间，定义如下：

```
typedef struct node{
    char num[10];     //车牌号码
    Time reach;       //到站时间
    Time leave;       //离开时间
}CarNode;
```

（2）时间、栈和队列的定义

时间是由小时和分钟表示的，有两部分数据，所以，类似于复数的表示一样，设计两个变量分别存储小时和分钟。如：

```
    typedef struct time
    {
      int hour;
      int min;
    }Time;
```

停车场内用栈表示：

```
typedef struct NODE
{
    CarNode *stack[MAX+1];     //栈用顺序表示
```

```
    int top;
}SeqStackCar;
便道上的车辆表示：
typedef struct car
{
    CarNode *data;        // 便道上的车用链表表示
    struct car *next;
}QueueNode;
typedef struct Node
{
    QueueNode *head;      //  设置头指针、尾指针。。
    QueueNode *rear;
}LinkQueueCar;
```

2. 算法的设计思想及流程图

（1）主要函数的功能说明

1、void InitStack(SeqStackCar *);　　//车辆节点进栈

当栈未满时，就把到达的车辆进栈。

2、int InitQueue(LinkQueueCar *);　　//车辆节点进队列

当栈满了时，车辆就进入便道上的队列中

3、int Arrival(SeqStackCar *,LinkQueueCar *);　　//车辆到达登记

车辆到达时，先登记车辆车牌号码。然后再判断停车场有没有停满，没停满就进栈，停满了就停在便道上，即进队列。。

4、void Leave(SeqStackCar *,SeqStackCar *,LinkQueueCar *);　//车辆离开处理

通过输入离开车辆的位置处理，然后调用 PRINT(CarNode *p,int room);函数进行收费。。然后再判断便道上有没有车，如果有，就把便道上的车进停车场内。

5、void List(SeqStackCar,LinkQueueCar);　　//显示车场内和便道上的车辆情况

用个 switch();函数选择显示车场内或是便道上的车辆情况。

包括对下面两个子函数的调用： void List1(SeqStackCar *S);

　　void List2(LinkQueueCar *W);　　//分别为显示车场和便道上的车辆情况

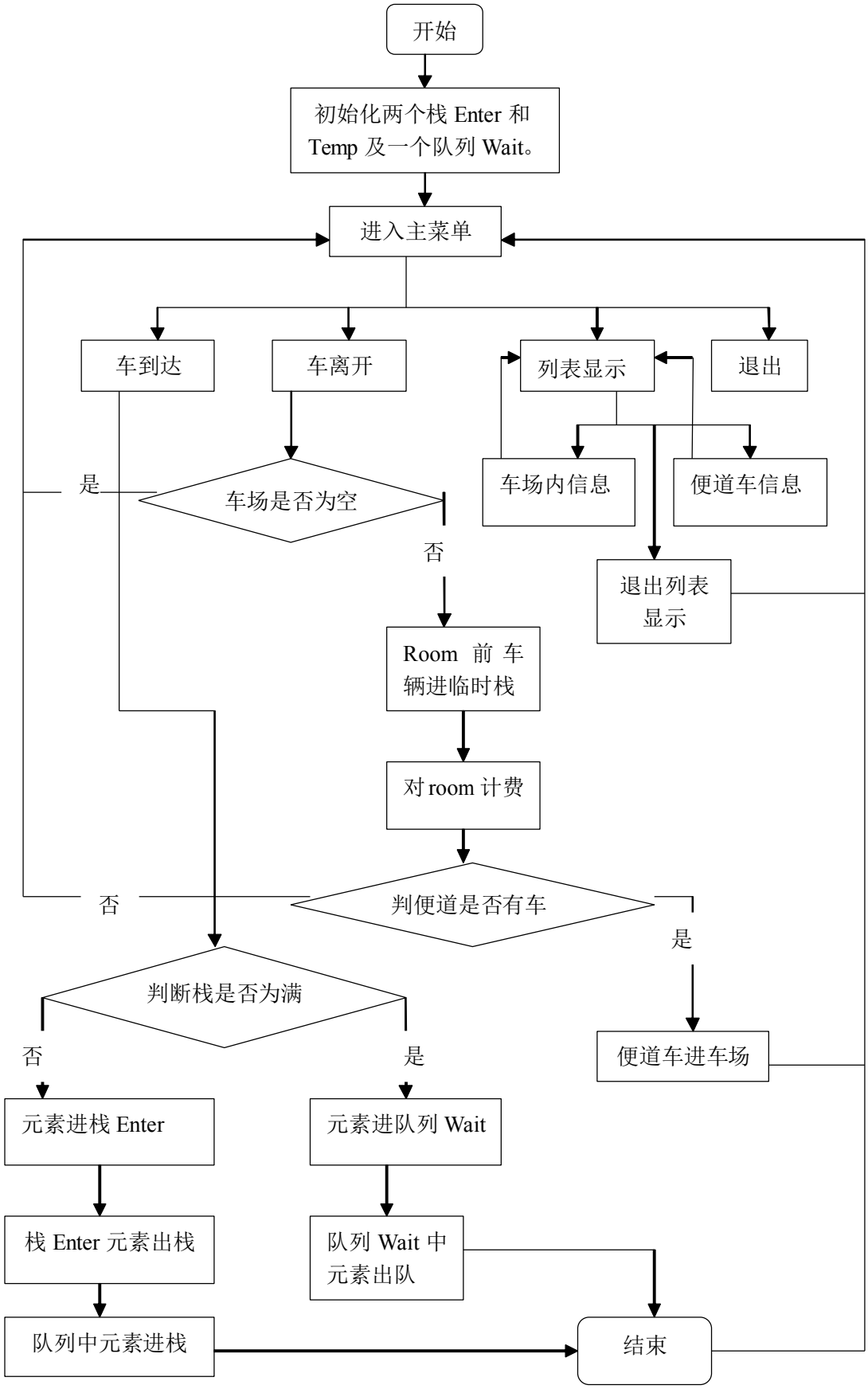
6、void PRINT(CarNode *p,int room);　//　车辆离开是的收费

这个函数由车辆离开的函数调用，以分钟计时算费，但只能计算当天之内的费用，如果第

二天的话会导致计费为负或减少。即只能当天停，当天开走。

（2）模块结构及流程图

下图为程序的主流程图，比较清晰的显示了程序的整个运行过程。如：图 1.


开始
初始化两个栈 Enter 和 Temp 及一个队列 Wait。
进入主菜单
车到达
车离开
列表显示
退出
车场内信息
便道车信息
是
车场是否为空
否
退出列表显示
Room 前车辆进临时栈
对 room 计费
否
判便道是否有车
是
判断栈是否为满
否
是
便道车进车场
元素进栈 Enter
元素进队列 Wait
栈 Enter 元素出栈
队列 Wait 中元素出队
队列中元素进栈
结束

（3）主要模块算法描述

本程序最主要的算法就是车辆到达登记的和车辆离开的。

①车辆到达：int Arrival(SeqStackCar *Enter,LinkQueueCar *W)

首先定义一个栈和队列的结构体指针为：*p， *t 。然后申请一个车辆信息的内存空间，并把它赋给栈指针。车辆到达时就输入车牌号，并通过 if(Enter->top<MAX)来判断该车是进车场内还是进便道上，如果是进车场内就把 top 加 1，显示在车场内的位置，还要输入进车场的时间，然后把该节点进栈。如果是 else 就显示该车要停在便道上，并进行进队列的操作。

②车辆离开： void Leave(SeqStackCar *Enter,SeqStackCar *Temp,LinkQueueCar *W)

定义一个整型变量 room 记录要离开车辆的位置，定义两个栈指针和一个队列指针，用个 if(Enter->top>0) 确保栈不空，然后用个 while(1) 确保输入的车辆离开位置的合法性。如果不和法，显示输入有误，要重新输入。通过 while(Enter->top>room) 判断离开车辆的位置，如果是中间位置，就要再用一个栈前面临时开出来的车，等要开出的车开出后，再把临时栈的车看进 车场内，并要调用 PRINT(p,room); 这个函数计算显示费用。然后还要用 if((W->head!=W->rear)&&Enter->top<MAX) 语句判断便道上有没有车，如果有车就要显示进车场的车的车牌号，并登记进入时间。并要进行相应的出队列和进栈操作。

五、源程序清单

"parking.h" //*****头文件*****//

```
typedef struct time
{    //  定义时间结构体
    int hour;
    int min;
}Time;

typedef struct node    //  定义车辆信息结构体
{
    char num[10];
    Time reach;
    Time leave;
}CarNode;

typedef struct NODE     //栈用顺序表示
```

```
{
    CarNode *stack[MAX+1];
    int top;
}SeqStackCar;

typedef struct car     // 便道上的车用链表表示
{
    CarNode *data;
    struct car *next;
}QueueNode;

typedef struct Node  //  设置头指针、尾指针
{
    QueueNode *head;
    QueueNode *rear;
}LinkQueueCar;

// 自定义函数
void InitStack(SeqStackCar *s)
{   // 栈的初始化
     int i;
     s->top=0;
     for(i=0;i<=MAX;i++)
         s->stack[s->top]=NULL;
}
int InitQueue(LinkQueueCar *Q)
{     // 队列的初始化
    Q->head=(QueueNode *)malloc(sizeof(QueueNode));
    if(Q->head!=NULL)
    {
        Q->head->next=NULL;
        Q->rear=Q->head;
        return(1);
    }
    else return(0);
}
void PRINT(CarNode *p,int room)
{     // 车辆收费
    int A1,A2,B1,B2;
    printf("\n车辆离开的时间:");
    scanf("%d:%d",&(p->leave.hour),&(p->leave.min));
    printf("\n离开车辆的车牌号为:");
    puts(p->num);
    printf("\n其到达时间为: %d:%d",p->reach.hour,p->reach.min);
    printf("\n离开时间为: %d:%d",p->leave.hour,p->leave.min);
    A1=p->reach.hour;
```

```
    A2=p->reach.min;
    B1=p->leave.hour;
    B2=p->leave.min;
    printf("\n 应交费用为: %2.1f 元",((B1-A1)*60+(B2-A2))*price);
    free(p);
}

    // 车辆的到达登记
int Arrival(SeqStackCar *Enter,LinkQueueCar *W)
{
    CarNode *p;
    QueueNode *t;
    p=(CarNode *)malloc(sizeof(CarNode));
    flushall();
    printf("\n 请输入车牌号(例:鄂 B1234):");
    gets(p->num);
    if(Enter->top<MAX)
    {
        Enter->top++;
        printf("\n 车辆在车场第%d 位置.",Enter->top);
        printf("\n 车辆到达时间:");
        scanf("%d:%d",&(p->reach.hour),&(p->reach.min));
        Enter->stack[Enter->top]=p;
        return(1);
    }
    else
    {
         printf("\n 该车须在便道等待!有车位时进入车场");
        t=(QueueNode *)malloc(sizeof(QueueNode));
        t->data=p;
        t->next=NULL;
        W->rear->next=t;
        W->rear=t;
        return(1);
    }
}

void Leave(SeqStackCar *Enter,SeqStackCar *Temp,LinkQueueCar *W)
{                                                    //车辆的离开
    int room;
    CarNode *p,*t;
    QueueNode *q;

    if(Enter->top>0)          //  判断车场是否为空
    {
        while(1)
```

```
        {
            printf("\n 请输入车在车场的位置/1--%d/: ",Enter->top);
            scanf("%d",&room);
            if(room>=1&&room<=Enter->top) break;
            else printf("\n 输入有误,请重输: ");

        }
        while(Enter->top>room)   //  把要删除的车辆的前面的车开出来，进临时栈。
        {
              Temp->top++;
            Temp->stack[Temp->top]=Enter->stack[Enter->top];
            Enter->stack[Enter->top]=NULL;
            Enter->top--;
        }
        p=Enter->stack[Enter->top];  // 把要删除的车辆节点赋给 p。
        Enter->stack[Enter->top]=NULL;
        Enter->top--;
        while(Temp->top>=1)     //  再把临时栈里德车辆进停车场
        {
            Enter->top++;
            Enter->stack[Enter->top]=Temp->stack[Temp->top];
            Temp->stack[Temp->top]=NULL;
            Temp->top--;
        }
        PRINT(p,room);      //   调用计费函数计费。
     if((W->head!=W->rear)&&Enter->top<MAX) //如果便道上有车，则再开进停车场。
        {
            q=W->head->next;
            t=q->data;
            Enter->top++;
            printf("\n 便道的%s 号车进入车场第%d 位置.",t->num,Enter->top);
            printf("\n 请输入%s 号车进入车场的时间:",t->num);
            scanf("%d:%d",&(t->reach.hour),&(t->reach.min));
            W->head->next=q->next;
            if(q==W->rear) W->rear=W->head;
            Enter->stack[Enter->top]=t;
            free(q);
        }
        else printf("\n 便道里没有车.\n");
    }
    else printf("\n 车场里没有车.");
}

void List1(SeqStackCar *S)   //显示车场里的车辆情况
{
    int i;
```

```
    if(S->top>0)
    {
        printf("\n车场:");
        printf("\n 位置  到达时间    车牌号\n");
        for(i=1;i<=S->top;i++)
        {
            printf("  %d   ",i);
            printf(" %d:%d   ",S->stack[i]->reach.hour,S->stack[i]->reach.min);
            puts(S->stack[i]->num);
        }
    }
    else printf("\n车场里没有车");
}

void List2(LinkQueueCar *W)     //显示便道上的车辆情况
{
    QueueNode *p;
    int i;
    p=W->head->next;
    if(W->head!=W->rear)
    {
        printf("\n等待车辆的号码为:");
        for(i=1; (p!=NULL); i++)
        {
            printf("\n第 %d 车辆.",i);
            puts(p->data->num);
            p=p->next ;
        }
    }
    else printf("\n便道里没有车.");

    printf("\n");
}

void List(SeqStackCar S,LinkQueueCar W)  //显示,遍历
{
    int flag,tag;
    flag=1;
    while(flag)
    {
        printf("   查看车辆列表显示:  ");
       printf("\n 1.车场列表\n 2.便道列表\n 3.返回主菜单\n");
       printf("\n请选择 1~3:");
        while(1)
        {
          scanf("%d",&tag);
```

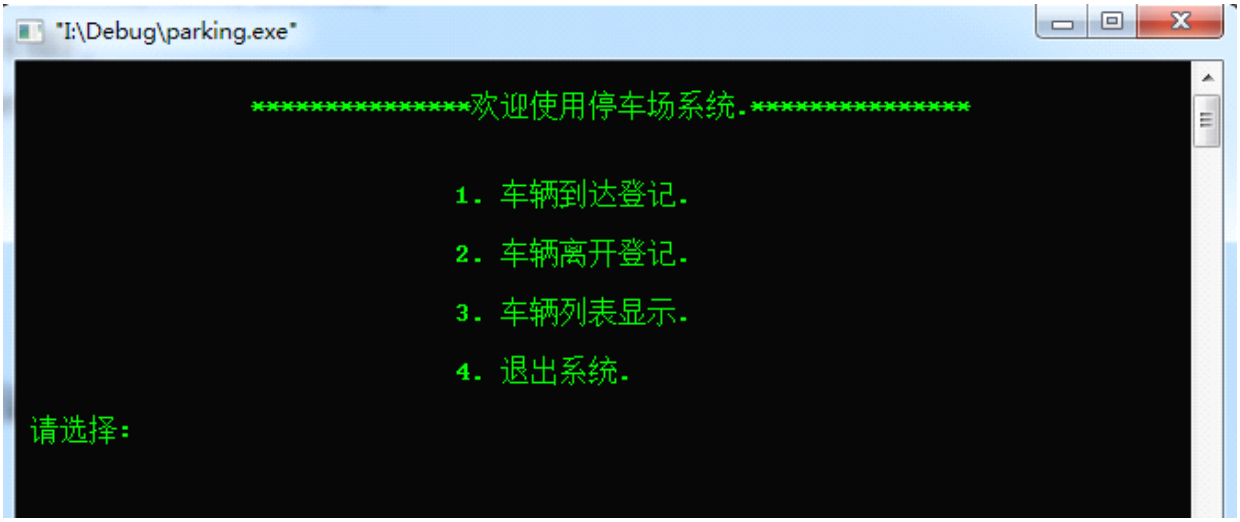
```
          if(tag>=1 && tag<=3) break;
          else printf("\n 输入有误,请重新选择 1~3:");
        }
        switch(tag)
        {
         case 1:List1(&S);break;
        case 2:List2(&W);break;
        case 3:flag=0;  system("cls");  break;
        default: break;
        }
    }
}
```

"parking.c" //*****主函数*****//

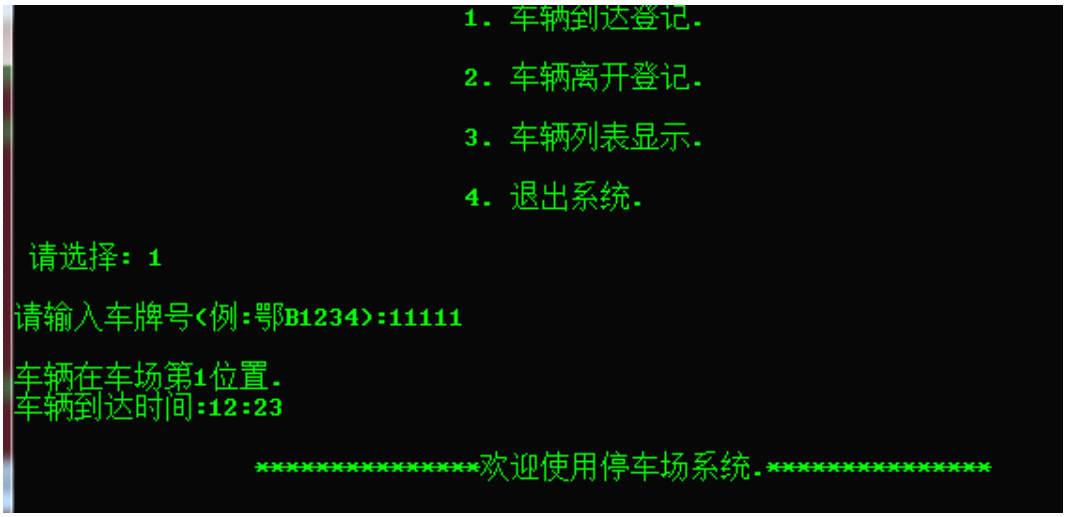
```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define MAX 3       //  停车场最大容量为 3 辆，便于观察
#define price 0.05    //   定义价格为每分钟 0.05 元
#include "parking.h"

void InitStack(SeqStackCar *);
int InitQueue(LinkQueueCar *);
int Arrival(SeqStackCar *,LinkQueueCar *);
void Leave(SeqStackCar *,SeqStackCar *,LinkQueueCar *);
void List(SeqStackCar,LinkQueueCar);
void main()
{
    SeqStackCar Enter,Temp;
    LinkQueueCar Wait;
    int ch;
    InitStack(&Enter);
    InitStack(&Temp);
    InitQueue(&Wait);
    while(1)
    {
        printf("\n\t                    ********** 欢 迎 使 用 停 车 场 系
统.**********\n\n");
        printf("\n\t                      1. 车辆到达登记.\t\n");
        printf("\n\t                      2. 车辆离开登记.\t\n");
        printf("\n\t                      3. 车辆列表显示.\t\n");
        printf("\n\t                      4. 退出系统.\t\n\n");
        while(1)
        {
            printf(" 请选择: ");
```

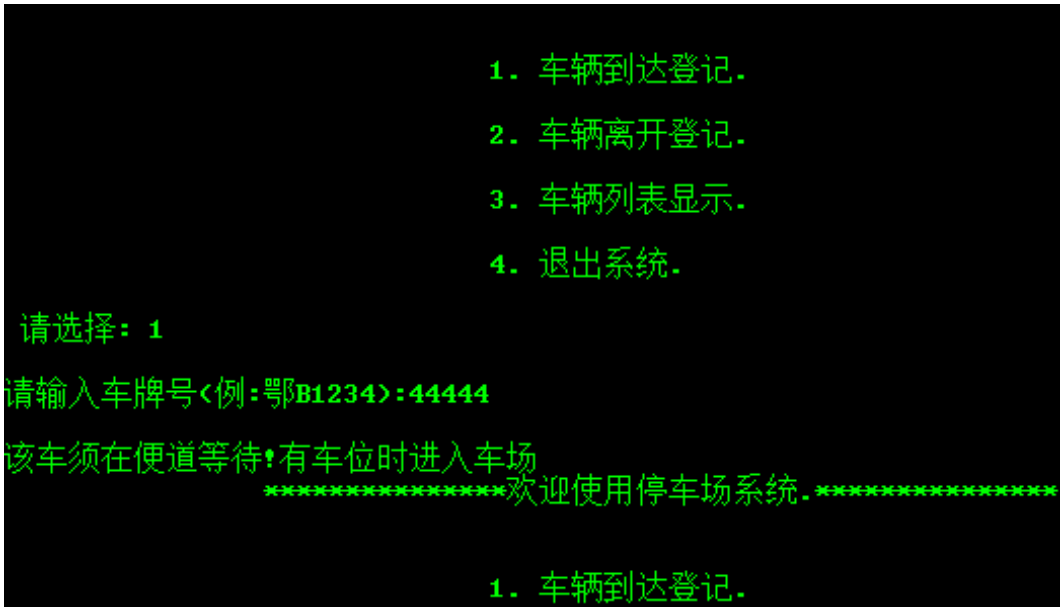
```
            scanf("%d",&ch);
            if(ch>=1&&ch<=4)break;
            else printf("\n 输入有误，请重新选择： 1~4: ");
        }
        switch(ch)
        {
         case 1:Arrival(&Enter,&Wait);  break;
        case 2:Leave(&Enter,&Temp,&Wait);break;
        case 3:List(Enter,Wait);break;
        case 4:exit(0);
        default: break;
        }
    }
}
```

六、测试数据及测试结果

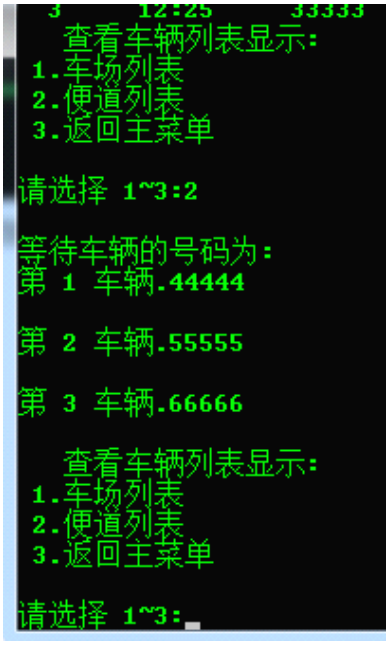


1、停车场管理系统的主界面

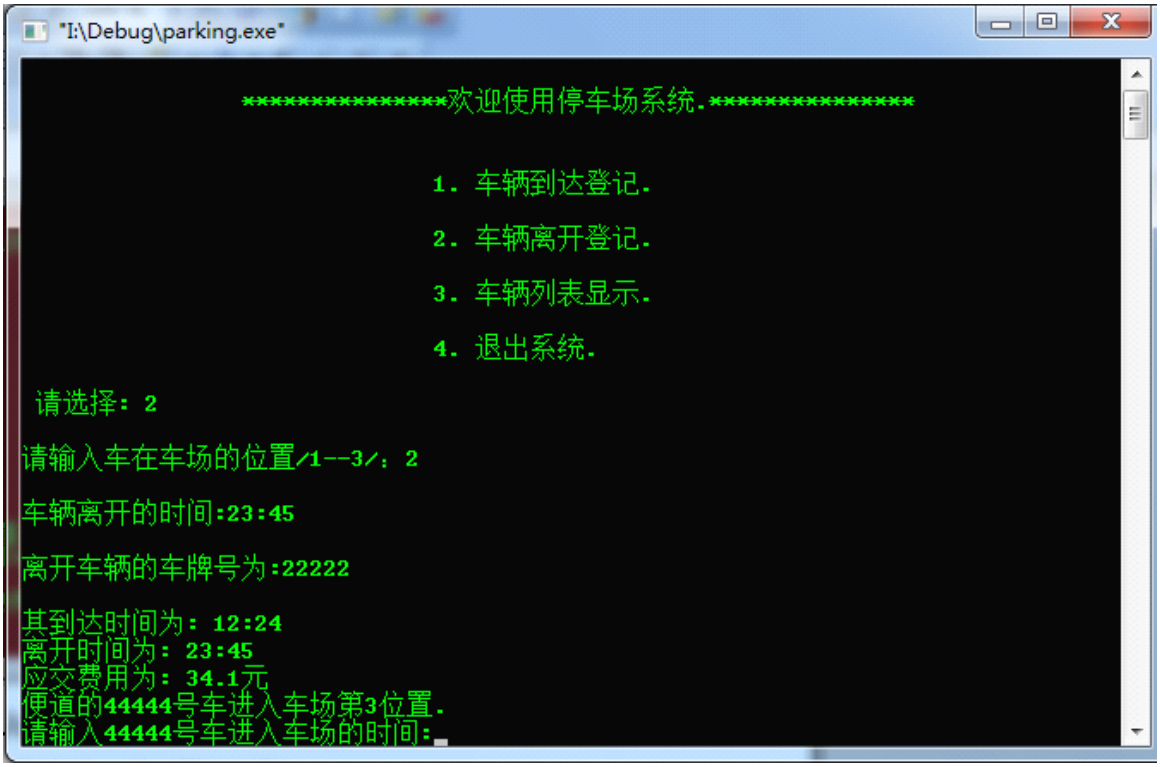




2、车辆到达等级信息，为了显示的方便，指明停车场内只能停 3 辆车；当停车场内停满时，到达的车辆只登记车牌，进入便道，此时即进入队列中。



3、分别显示停车场内和便道内的车辆的信息



4、**车场内车辆离开时，输入离开时间，然后计算、显示费用，如果便道上有车，则显示要进入车场内的车牌号码，同时登记时间。**

七、课程设计总结及心得体会

我这次的课程设计题目是关于停车场问题的，总体来说，这个题目还是比较简单的，主要是运用了栈和队列的一些知识和操作。也没有用到其他太多的数据结构知识。程序基本上还是能够运行，结果也正确，能够实现那些基本的车辆到达、离开、收费、遍历显示等主要功能。

但我觉得这个程序还有很多小的地方是可以完善的，比如：在输入登记车辆到达时间的时候，没有相关的小时、分钟数字的限制范围——（小时 0~23，分钟 0~60）；这就使程序不那么健壮了，还有，在计算收费时如果离开时间是到了第二天了，这样就可能会导致收费时负的或减少很多。就是说，还应该算天数，要不就要规定只能当天停，当天开走。我改了几次改不好，又鉴于程序要求中也没有提及，所以我也就没去该了。只要能实现主要的功能就好了。

另外，在题目中我还发现了一些不太切合实际的地方，此题中的停车过程不够人性化。若在有车辆开出的时候，其他车辆的车主不在时此题中的模拟过程将比较难于实现；此外，即使其他车辆的车主都在，这样的停车过程太过于复杂，加重了人们停车时的负担。

当然，这次的课程设计、编程实践还是大有收获的。

通过实习我的收获如下

1、我知道了怎样去简化程序，减少他的时间复杂度和空间复杂度。还知道了怎样去完善程序，使其更具健壮性。

2、巩固和加深了对数据结构的理解，提高综合运用本课程所学知识的能力。

3、培养了我选用参考书，查阅手册及文献资料的能力。培养独立思考，深入研究，分析问题、解决问题的能力。

4、通过实际编译系统的分析设计、编程调试，掌握应用软件的分析方法。

5、通过课程设计，培养了我严肃认真的工作作风，逐步建立正确的生产观念、经济观念和全局观念。

根据我在实习中遇到得问题，我将在以后的学习过程中注意以下几点：

1、认真上好专业实验课，多在实践中锻炼自己。更让我懂得实践是检验和掌握真理的最好办法。

2、写程序的过程中要考虑周到，严密。

3、在做设计的时候要有信心，有耐心，切勿浮躁。

4、认真的学习课本知识，掌握课本中的知识点，并在此基础上学会灵活运用。

5、在课余时间里多写程序，熟练掌握在调试程序的过程中所遇到的常见错误，以便能节省调试程序的时间。