课程设计题目　　**停车场模拟系统**

## Ⅰ课程设计的目的和内容

**设计目的：**模拟一个停车场系统，停车场根据停车的占地面积进行收费。

通过数据结构课程设计的训练，达到以下目的：(1)巩固和加深对数据结构知识的理解，熟练掌握几种重要的数据结构与算法，能够根据具体问题选择合适的数据结构并设计相应算法；(2)初步掌握软件开发过程的需求分析、系统设计、程序测试和编码风格等基本流程和规范；(3)提高综合运用所学的数据结构和算法知识独立分析和解决复杂工程问题的能力；(4)培养学生从事计算机相关专业工作所应具备的科学方法和基本素质。

**设计内容：**

我们假设一个停车场存在大、中、小三种车型的停车位（不同类型的停车位收费标准不同，且上一级别的车不能在下一级别的停车位停车）。当有车辆进入停车场时，登记车的型号，车牌以及时间，并将车停放在停车场内。当取车时，输入车牌号，取出相应的车，并记录现在时间，根据停车时间及车位类型收取费用。当停车场满时，在停车场外的一个长度为N的候车道等待进入停车场停车。

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。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。

## Ⅱ课程设计的创新和特色

以栈模拟停车场，以队列模拟车场外的便道，用链式表模拟队列，用顺序表模拟栈，采用多个数据结构与算法;结构体存储数据，便于维护；格式化读写文件；对时间、车牌号等数据进行充分检验操作；对操作界面进行了优化，界面优美。

## Ⅲ设计进度及完成情况

| 日　　期 | 内　　容 |
| --- | --- |

| | |
|---|---|
| 12 月 12 | 查找相关资料、书籍，分析题目要求 |
| 12 月 13—12 月 14 | 画框架图，，设计功能等模块。 |
| 12 月 15—12 月 18 | 停车模块完成 |
| 12 月 19—12 月 22 | 取模块完成 |
| 12 月 23—12 月 24 | 车库查询模块完成 |
| 12 月 25—12 月 27 | 完成优化代码，调试bug |
| 12 月 28—12 月 30 | 完成课程设计报告的撰写 |

## Ⅳ主要参考文献


[1]《数据结构（C语言版）》，严蔚敏、吴伟民，清华大学出版社，1997.

[2]《大话数据结构》，程杰，清华大学出版社，2011.

[3][C语言教你怎么改变字体颜色]

https://blog.csdn.net/qq_31975227/article/details/51758461

[4][strstr(str1,str2)函数使用时注意事项]

https://blog.csdn.net/ludaoyi88/article/details/52819448


## Ⅴ成绩评定

评语：

指导老师：_____(签字)

年　　月　　日

# 目　录

# 第 1 章 概述

## 1.1 背景与意义

伴随着科技的飞速发展，交通工具的越来越普及。汽车作为人类社会中最主要的交通工具之一，起着重大作用。随着人们生活水平的提高，汽车的数量也与日俱增，于是停车正在成为世界性的问题。以前落后的人力停车管理即将被高科技化的自动停车管理系统所取代，高度自动化的停车场管理系统节省了大量时间和人力物资消耗，大大提高了效率。

## 1.2 主要任务

模拟一个停车场系统。停车场根据停车的占地面积进行收费。我们假设一个停车场存在大、中、小三种车型的停车位（不同类型的停车位收费标准不同，且上一级别的车不能在下一级别的停车位停车）。当有车辆进入停车场时，登记车的型号，车牌以及时间，并将车停放在停车场内。当取车时，输入车牌号，取出相应的车，并记录现在时间，根据停车时间及车位类型收取费用。当停车场满时，在停车场外的一个长度为 N 的候车道等待进入停车场停车。

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。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）。

## 1.3 系统功能

1.车辆进出停车场管理功能
2.停车场车辆查询功能
3.文件储存停车场、便道信息
4.针对大、中、小三种车型收费功能

## 1.3 数据结构与算法

1. 本系统以栈模拟停车场，栈以顺序结构实现；以队列模拟停车场外的便道，队列以链表实现。

2. 以结构体储存时间、汽车、停车场、队列等数据。

3. 停车场和便道中的数据以文件形式存储。

4. 使用了顺序栈的初始化、链队的初始化、顺序栈的入栈、顺序栈的出栈、判断顺序栈是否为空、顺序栈的遍历、链队的入队、链队的出队、文件的读写等数据结构与算法。

## 1.4 创新与特色

1.对时间、车牌号等数据进行充分检验。
2.采用多个数据结构与算法
3.对操作界面进行了优化，界面优美。

# 第 2 章 系统分析

## 2.1 需求分析

普遍使用：随着经济的发展，车辆广泛普，并且停车场也成为公共设施的重要组成部分，该系统对于管理车辆有着很大的实用范围。

操作简单：系统界面简单易懂，只需要操作人员会操作 windows 系统即可，时间计算，费用计算都可以通过系统自主完成，操作人员只需要输入相关变量，输出界面就可以自主输出相关信息。

硬件可行：本系统对计算机配置的条件要求不高，不需要单独的服务器，普通家用计算机就可以完成所有操作。

## 2.2 子任务

初始化停车场、便道、车位（Init_SeqList、Init_Lqueue、Init_Car_Space），汽车进入停车场（Push_SeqList 函数），停车场满的时候车辆进入便道（In_LQueue），汽车离开停车场（Pop_SeqList），停车场有车离开有空位时便道上的车辆按照到达顺序依次离开便道并进入停车场（Out_LQueue），显示停车场内车辆信息（Display），便道第一辆车进入停车场后，便道中后面车辆前移（Traverse_LQueue），获取车辆的型号和进入/离开停车场的信息（Car_Condition），停车时检验停车场内是否有该车辆（Car_if），停车时检查车辆的车牌号格式是否正确（checkNum），输入时间时检查时间格式是否符合要求（checkTime），检查汽车离开出栈的时间是否合法（Check_TimeOut），向便道文件写入车辆数据（LQueue_Input），删除便道文件中离开便道的车辆数据（LQueue_Delete），向停车场写入车辆数据（SeqList_Input），删除停车场车辆数据（SeqList_Delete）

## 2.3 功能模块

### 2.3.1 停车模块

有汽车进入停车场时，登记车的车牌号，型号和到达时间，将车按顺序停到停车场内。若停车场对应车位已满，则让该车在便道等待,若便道也满，则通知其离开。

### 2.3.2 取车模块

当有车离开时，输入车牌号和离开时间，若车牌号正确，该车前面的车为其让路，车辆驶出并输出停车时间和停车费用，若便道有车，则便道的车驶入。

### 2.3.3 查询车库模块

可随时查看停车场内信息，显示车牌号和进图停车场的时间。

### 2.3.4 退出系统模块

可正常退出系统。

# 第 3 章 概要设计

## 3.1 算法分析

（1）模拟停车场的车辆需要输入车辆的信息，比如车辆的车牌号码，车的大小类型，车辆到达的时间等，车辆到达的时间可以分为小时和分钟，所以我们选择定义一个有关汽车信息的结构体，其中包含车牌号，车型和时间。并且时间也是结构体组成，其中包括小时和分钟。完成车辆信息的储存。

（2）模拟停车场需要停车位和便道，我们根据题目要求使用栈来实现停车位，用队列实现便道，根据题目要求，选择用顺序表模拟栈，用链式表模拟队列。需要定义一个模拟停车位的结构体，其中元素包括汽车信息型的变量，主要储存汽车的信息，以及一个控制栈顶的元素；同时需要定义一个实现便道的结构体，实现便道的结构体由两部分组成，首先是储存汽车信息的结构体，另外一个包括前后指针的结构体。

（3）车辆到达时，要输入车辆的信息，并保存在停车位内，每进入一辆车，要判断停车位（顺序栈）是否已停满，若已满，则提示该车要在便道（链式队列）上等待。在停车场未满进入停车场时，要可选择车位，只能选择停车场剩余的车位，进入便道则按进入顺序停车。

（4）汽车离开时，需要输入车牌号，根据车牌号，判断车辆是否在车站中，另外需要另外设计一个辅助栈，当一辆汽车要离开时，在其后的车辆需要给他让道，让路的汽车就暂时停放在这个栈中。车辆离开后，要判断便道上是否有车辆在等待，若有则进行入栈操作，将车辆的信息节点进行入栈操作，且对便道需要实现出队操作，并计算停车时间、费用等。

（5）车辆查询，查询车库时，展现车库的车牌号以及停车时间。

## 3.2 程序流程图

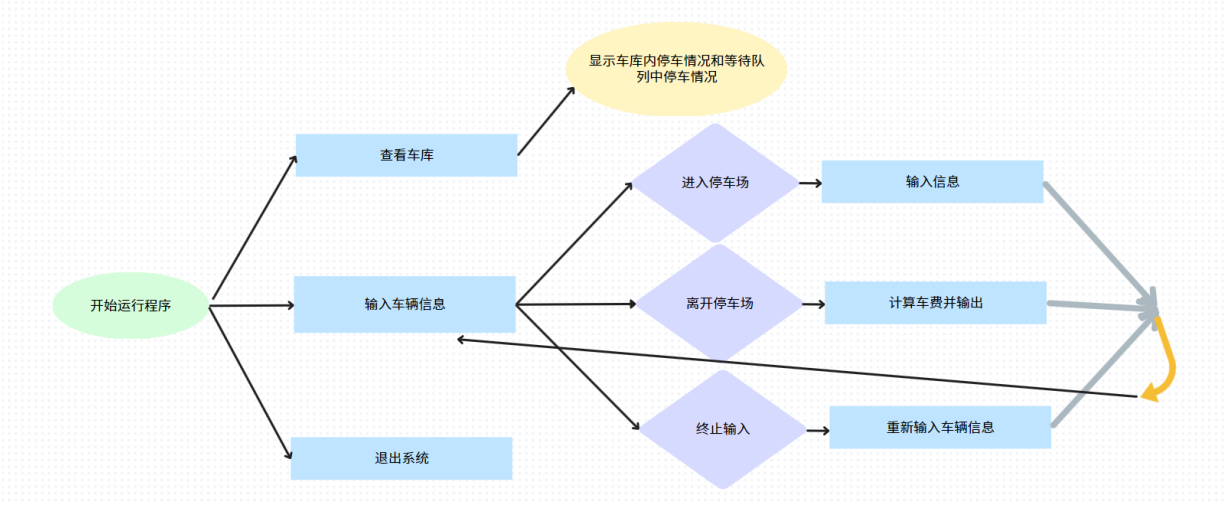


**图 3.1 程序流程图**

## 3.3 停车场模拟图

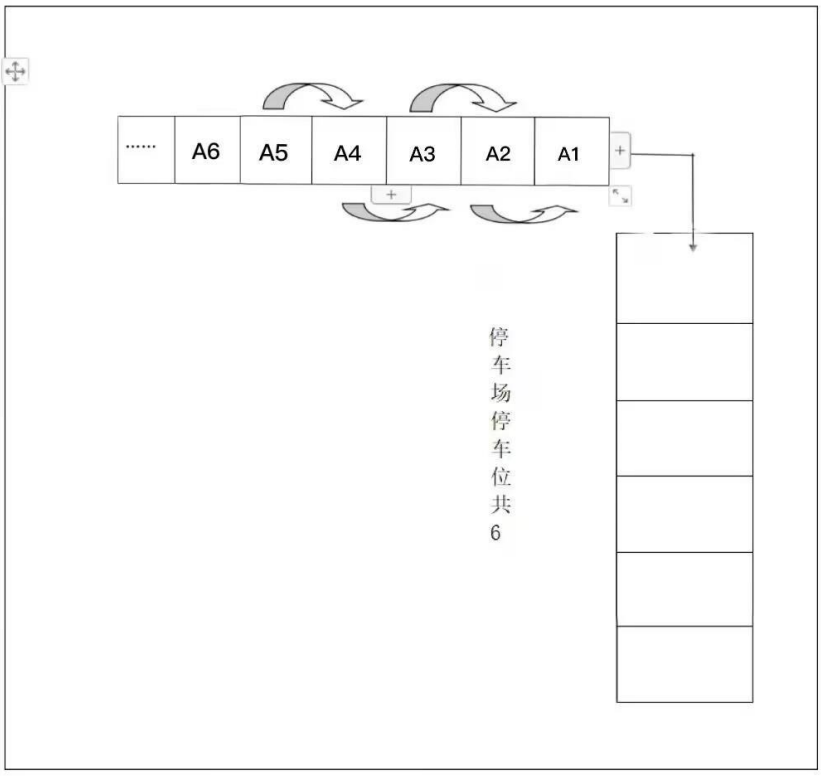


**图 3.2 停车场模拟图**

# 第 4 章 详细设计

## 4.1 全局变量模块

首先，在程序开始之前，我们定义了一些全局变量来使程序编写变得更加方便迅捷

全局常量定义：

```
//下面是每分钟停车费用，我们按分钟收费,分大中小三种车型
const float price_1 = 0.2;   //大
const float price_2 = 0.1;   //中
const float price_3 = 0.05;  //小
```

全局变量定义

```
string Car_Type; //汽车的型号
int i;//汽车在便道中的位置
int number=0; //计数器：记录停车时的总输入量
int number_SeqList=0; //计数器：记录当前停车场已有的车辆数
int number_LQueue=0;  //计数器：记录当前便道上已有的停车数
```

上述全局变量中，我们采用 number_和 Car_加具有实际意义的名词，来表示不同的变量，car 是一个储存车辆信息的结构体型的变量，它其中包括的元素是车的型号，车的

车牌号，车的离开时间，车在停车场位置。主要的作用是 将这几份元素封装，调用的同时不会造成信息混乱，在停车和取车的时候，可以根据一个元素匹配，调出来所有的元素。number_SeqList 存取的是停车场上已有的车辆数，number_LQueue 则是存取的便道上已有的停车数。因为数据的存储，存的数据为汽车型的变量，所以需要用到结构体，我们采用顺序表的形式存储栈，用链式表的形式存储队列。

## 4.2 算法设计模块

### 4.2.1 定义停车场以及停车便道的结构体储存不同数据

```
//停车场
typedef struct {
Car data[MAXSIZE];
int top;
}SeqList;

// 停车场车位
typedef struct{
    int space;
    int flag;
}Car_space;
Car_space Car_Space[MAXSIZE];

//便道
typedef struct qnode {
Car data;
struct qnode* next;
}QNode;
typedef struct lqueue {
QNode* front, * rear;
}LQueue;
```

根据题目的要求，我们选择用顺序表模拟栈（停车场），用链式表模拟队列（停车便道）。因此，需要定义结构体来储存车辆的信息、栈顶元素、队列的前后指针。上述结构体中 top 表示栈顶元素所在的位置，即停车场此时存的车辆，top 等于零的时候表示停车场内没有车辆。data 型数组的表示 carData 型的数据，主要的用途是储存汽车的信息，而把它定义成数组型的数据主要是顺序表的存储需要定义成数组型的，方便存取。CarData 型的数据表示当取车的时候可以用 carData 型的临时变量临时储存从停车场或者是停车便道中的车。当停车的时候可以定义一个 carData 型的变

量或者是定义三个不同的变量通过输入数据然后将数据传输给 StopCar 的结构体中的 data。WaitCar 结构体中 next 表示一个指向下一个元素的指针，通过 next 可以将数据穿成一条链子，而 front 和 rear 表示便道的头和尾。主要作用是初始化和定位。

### 4.2.2 初始化操作

```
SeqList* Init_SeqList()
{
    SeqList* s;
    s = new SeqList;
    s->top = -1;
    return s;
}
```

图 4.1 初始化停车场

```
//初始化停车场车位
void Init_Car_Space()
{
    for(int j=0;j<MAXSIZE;j++)
    {
        Car_Space[j].space = j+1;
        Car_Space[j].flag = 1;
    }
}
```

图 4.2 初始化停车场车位

```
LQueue* Init_LQueue()
{
    LQueue* q;
    QNode* p;
    q = new lqueue;
    p = new qnode;
    p->next = NULL;
    q->front = p;
    q->rear = p;
    return q;
}
```

图 4.3 初始化便道

### 4.2.3 停车函数

```
void Push_SeqList(SeqList* s, Car x)
{
    s->top++;
    s->data[s->top] = x;
}
```

图 4.4 汽车进入停车场

```
void In_LQueue(LQueue* q, Car x)
{
    QNode* p;
    p = new qnode;
    p->data = x;
    p->next = NULL;
    q->rear->next = p;
    q->rear = p;
}
```

**图 4.5 停车场满的时候进入便道**

下面是停车的主函数操作，停车时需要判断车牌号是否重复，车牌号的格式、时间格式是否正确，并根据停车场是否满栈判断是否进入停车场。

```
if (p->top == MAXSIZE - 1)  //当停车场满时
{
    //进入便道
    c.p2 = ++i;
    color(4);
    cout << "生意火爆，停车场已满，进入便道等待，位置为：00" << c.p2 << endl;
    number_LQueue++;    //便道上等待的车辆+1
    In_LQueue(q, c);
    LQueue_Input(Car_Type,c);   //向便道文件写入车辆数据
    Sleep(2000);
    system("cls");
}
else {
    //进入停车场
    int flag = 1;   //标记输入车位是否错误，输入正确置为0
    int arr[MAXSIZE];   //临时记录停车场的空位
    color(4);
    cout << "停车场未满，正在进入停车场~" <<endl;
    cout << "当前空闲车位有:";
    for(int j=0;j<MAXSIZE;j++)
```

**图 4.6 判断是否进入停车场**

```
while(!checkNum(num))  //检验车牌号的格式,如果输入错误则需要重新输入
{
    color(4);
    cout << "-------------------------------------------------------------------------------------------------------" << endl;
    cout << "| 注意：车牌号格式为普通车牌号格式【第一位为省份，第二位为发证机关A~H|J~N|P~Z，第三~七位为号码0~9|A~H|J~N|P~Z】 |" << endl;
    cout << "-------------------------------------------------------------------------------------------------------" << endl;
    cout << "请重新输入汽车车牌号:\n";
    cin >> num;
}
c.num = num;
if(p->top!=-1)
{
    while(Car_if(p,c))      //判断车牌号是否重复
    {
        color(4);
        cout<<"你输入的车牌号已经在车库当中了，请检查你的车牌号是否正确：\n";
        color(7);
        cin >> c.num;
    }
}
color(4);
cout << "请输入汽车的到达时间（格式：小时+空格+分钟）:\n";
color(7);
cin >> c.t1.h >> c.t1.m;
while(!checkTime(c.t1.h,c.t1.m))     //判断输入的到达时间是否符合要求
{
    color(4);
    cout << "-----------------------------------------------------------" << endl;
    cout << "| 注意：我们的时间要求是24小时的标准时间,注意中间的空格 |" << endl;
    cout << "-----------------------------------------------------------" << endl;
    cout << "请重新输入汽车的到达时间（格式：小时+空格+分钟）:\n";
    color(7);
    cin >> c.t1.h >> c.t1.m;
}
```

**图 4.7 在进入停车场之前判断车票好以及停车时间**

```
for(int j=0;j<MAXSIZE;j++)
{
    if(Car_Space[j].flag == 1)
    {
        color(9);
        cout << " "<< Car_Space[j].space;      //输出停车场的空位
        arr[j] = Car_Space[j].space;     //将空位信息存到arr数组中，方便后续检查
    }
}
```

图 4.8 选择车位

### 4.2.4 取车函数

```
Car Pop_SeqList(SeqList* s)
{
    Car x = s->data[s->top];
    s->top--;
    return x;
}
```

图 4.9 汽车离开停车场

```
Car Out_LQueue(LQueue* q)
{
    Car x;
    QNode* p;
    p = q->front->next;
    q->front->next = p->next;
    x = p->data;
    delete p;
    if (q->front->next == NULL) //只有一个元素时，出队后队空，此时还要修改队尾指针
    {
        q->front->next = NULL;
        q->rear = q->front;
    }
    return x;
}
```

图 4.10 便道中的车进入停车场

```
Car Traverse_SeqList(SeqList *p,int top,string num)
{
    while(num!=p->data[top].num)
    {
        top--;
    }
    return p->data[top];
}
```

图 4.11 寻找出栈的车辆

下面是汽车离开停车场的主函数，汽车离开时，我们需要判断输入的车牌号是否合法，并且离开的时间要大于汽车到达停车场的时间，时间的格式也要做出检验，停车场中的车辆离开后，需要输出汽车停车的时间和收取的费用，并且需要判断便道中是否有车，有车则需要进入停车场，车位继承上一辆车的车位，并且需要更新汽车到达停车场的时间。

```
while(!Check_TimeOut(car.t1.h,c.t2.h,car.t1.m,c.t2.m))      //判断目标车辆的离开时间是否合法
{
    color(4);
    //不合法需要重新输入
    cout << "时间不合法，请重新输入:\n";
    color(7);
    cin >> c.t2.h >> c.t2.m;
}
```

图 4.12 判断汽车离开时间是否合法

```
//判断停车场内是否有该车辆
if(!Car_if(p,c))
{
     color(4);
     cout << "该停车场内没有该车辆~" << endl;
     Car_Condition(p,q); //再进行下一次的输入判断
     return;
}
```

图 4.13 判断停车场是否存在车辆

```
if (p->top == MAXSIZE - 1)
{
    //当停车库满了的时候判断便道是否有车
    if (q->front == q->rear) return;
    else {
        first = Out_LQueue(q);  //便道内第一辆车
        flag = 1;
    }
}
```

图 4.14 判断便道中是否有车

```
if (flag)
{
    first.p1 = x1.p1;     //将离开的车位给便道上开入的汽车
    Car_Space[x1.p1-1].flag = 0;     //便道的上车开入，停车场的车位被占用，flag置为0
    first.t1.h = x1.t2.h;     //更新便道上开入的汽车的开始停车的时间【便道上停车的时间不收费】
    first.t1.m = x1.t2.m;
    Push_SeqList(p, first);
    color(9);
    cout << "车牌号为" << first.num << "的车开入" << endl;
    i--;     //对便道上的车位标签-1
    Traverse_LQueue(q);      //便道第一辆车进入停车场后，对后面车辆前移
    number_SeqList++;        //便道上的车开入停车场，停车场的车辆数目+1
    number_LQueue--;         //便道上的车开入停车场，便道上的车辆数目-1
    LQueue_Delete(first);    //删除便道文件的第一辆车
}
```

图 4.15 便道车进入停车场

```
int t = (x1.t2.h - x1.t1.h) * 60 + x1.t2.m - x1.t1.m; //总停留分钟
int t1, t2;
if (x1.t2.m >= x1.t1.m)
{
    t2 = x1.t2.m - x1.t1.m;
    t1 = x1.t2.h - x1.t1.h;
}
else {
    t2 = x1.t2.m + 60 - x1.t1.m;
    t1 = x1.t2.h - x1.t1.h - 1;
}
```

**图 4.16 停车时间、车费计算**

### 4.2.4 查询车库函数

对车停车场车库进行查询，展示停车场中车辆的信息，包括车牌号、车型、停车车位以及停车时间

```
void Display(SeqList* p)
{
    Car x;
    SeqList* temp = Init_SeqList();
    if (p->top == -1)
    {
        color(4);
        cout << "当前停车场无车，生意惨淡TvT~" << endl;
        Sleep(2000);
        system("cls");
        return;
    }
    else
    {
        while (p->top != -1)
        {
            x = Pop_SeqList(p);
            Push_SeqList(temp, x);
        }
        color(4);
        cout << "当前停放的车辆信息:" << endl;
        while (temp->top != -1)
        {
            x = Pop_SeqList(temp);
            color(9);
            cout << "停车位置为：00" << x.p1 << "  车牌号码:" << x.num << "  停车时间：" << x.t1.h << "点" << x.t1.m << "分" << endl;
            Push_SeqList(p, x);
        }
    }
    Sleep(4000);
    system("cls");
}
```

**图 4.17 查询车库函数**

### 4.2.5 检验函数

由于停车场系统数据严谨，我们需要对汽车车牌号、停车时间等做出检验

```
bool Car_if(SeqList* p, Car c)
{
    int t=p->top;
    while (!Str_strcmp(c.num,p->data[t].num))
    {
        t--;
        if(t==-1) break;
    }
    if (t == -1) return 0;
    else return 1;
}
```

**图 4.18 检验停车场是否存在当前车辆**

```
bool checkNum(string S)
{
    char s[S.length()];
    for(int j=0;j<S.length();j++)
    {
        s[j] = S[j];
    }
    int n = S.length(); //车牌的长度
    char *ch; //用来放车牌号的汉字
    strncpy(ch,s,2);     //截取字符串的前两个字符，汉字占两个字节
    if(n!=8) return false;
    if((strcmp(ch,"京")!=0)&&(strcmp(ch,"津")!=0)&&(strcmp(ch,"晋")!=0)&&(strcmp(ch,"冀")!=0)&&(strcmp(ch,"蒙")!=0)&&(strcmp(ch,"辽")!=0)&&
       (strcmp(ch,"吉")!=0)&&(strcmp(ch,"黑")!=0)&&(strcmp(ch,"沪")!=0)&&(strcmp(ch,"苏")!=0)&&(strcmp(ch,"浙")!=0)&&(strcmp(ch,"皖")!=0)&&
       (strcmp(ch,"闽")!=0)&&(strcmp(ch,"赣")!=0)&&(strcmp(ch,"鲁")!=0)&&(strcmp(ch,"豫")!=0)&&(strcmp(ch,"鄂")!=0)&&(strcmp(ch,"湘")!=0)&&
       (strcmp(ch,"粤")!=0)&&(strcmp(ch,"桂")!=0)&&(strcmp(ch,"琼")!=0)&&(strcmp(ch,"渝")!=0)&&(strcmp(ch,"川")!=0)&&(strcmp(ch,"贵")!=0)&&
       (strcmp(ch,"云")!=0)&&(strcmp(ch,"藏")!=0)&&(strcmp(ch,"陕")!=0)&&(strcmp(ch,"甘")!=0)&&(strcmp(ch,"青")!=0)&&(strcmp(ch,"宁")!=0)&&
       (strcmp(ch,"新")!=0)) return false;
    if(s[2]<'A' || s[2]>'Z' || s[2]=='I' || s[2]=='O') return false;
    if(!(s[3]>='0' && s[3]<='9') && !(s[3]>='A' && s[3]<='H') && !(s[3]>='J' && s[3]<='N') && !(s[3]>='P' && s[3]<='Z')) return false;
    if(!(s[4]>='0' && s[4]<='9') && !(s[4]>='A' && s[4]<='H') && !(s[4]>='J' && s[4]<='N') && !(s[4]>='P' && s[4]<='Z')) return false;
    if(!(s[5]>='0' && s[5]<='9') && !(s[5]>='A' && s[5]<='H') && !(s[5]>='J' && s[5]<='N') && !(s[5]>='P' && s[5]<='Z')) return false;
    if(!(s[6]>='0' && s[6]<='9') && !(s[6]>='A' && s[6]<='H') && !(s[6]>='J' && s[6]<='N') && !(s[6]>='P' && s[6]<='Z')) return false;
    return true;
}
```

图 4.19 检验车牌号格式

```
bool checkTime(int h,int m)
{
    if(0<=h&&h<24 && 0<=m&&m<60)
        return true;
    else return false;
}
```

图 4. 20 检验时间格式

```
bool Check_TimeOut(int h1,int h2,int m1,int m2)
{
    if(((h2*60+m2)-(h1*60+m1))>0) return true;
    else return false;
}
```

图 4.21 检验出栈时间合法性

### 4.2.6 文件操作函数

```
void LQueue_Input(string Car_Type, Car car)
{
    FILE* fp;
    if ((fp = fopen("LQueue.txt", "at+")) == NULL)
    {
        color(4);
        cout<<"不能打开文件"<<endl;
        return;
    }
    string st1 = "车牌号: "+ car.num + "; 型号: "+car.type + "; ";
    int h = car.t1.h;
    int m = car.t1.m;
    int locate = car.p2;
    char st2[st1.length()];
    for(int j=0;j<st1.length();j++)
    {
        st2[j] = st1[j];
    }
    fprintf(fp,"%s便道车位00%d; 停入便道时间: %d时%d分。\n",st2,locate,h,m);
    //fputs(st2,fp);
    fclose(fp);
}
```

图 4.22 向便道文件写入车辆信息

```
void SeqList_Input(string Car_Type, Car car)
{
    FILE* fp;
    if ((fp = fopen("SeqList.txt", "at+")) == NULL)
    {
        color(4);
        cout << "不能打开文件" << endl;
        return;
    }
    string st1 = "车牌号: "+ car.num + "; 型号: "+car.type + "; ";
    int h = car.t1.h;
    int m = car.t1.m;
    int locate = car.p1;
    char st2[st1.length()];
    for(int j=0;j<st1.length();j++)
    {
        st2[j] = st1[j];
    }
    fprintf(fp,"%s停车场车位00%d; 停车时间: %d时%d分. \n",st2,locate,h,m);
    //fputs(st2, fp);
    fclose(fp);
}
```

图 4.23 向停车场文件写入车辆信息

```
char st2[st1.length()];
for(int j=0;j<st1.length();j++)
{
    st2[j] = st1[j];
}

while (fscanf(fp1,"%[^\n]",strLine)!=EOF)            //循环读取每一行，直到文件尾
{
    fgetc(fp1);
    if(strstr(strLine,st2)==NULL)
    {
        fprintf(fp2,"%s\n",strLine);
    }
}
fclose(fp1);
fclose(fp2);

if ((fp1 = fopen("temp.txt", "rt")) == NULL)
{
    color(4);
    cout << "不能打开文件" << endl;
    return;
}
if ((fp2 = fopen("LQueue.txt", "wt")) == NULL)
{
    color(4);
    cout << "不能打开文件" << endl;
    return;
}
while (fscanf(fp1,"%[^\n]",strLine)!=EOF)    //循环读取每一行，直到文件尾
{
    fgetc(fp1);
    fprintf(fp2,"%s\n",strLine);
}
```

图 4.24 删除便道文件进入停车场的车辆数据

```
string st1 = car.num;
char st2[st1.length()];
for(int j=0;j<st1.length();j++)
{
    st2[j] = st1[j];
}

while (fscanf(fp1,"%[^\n]",strLine)!=EOF)    //循环读取每一行，直到文件尾
{
    fgetc(fp1);                    //将fp所指向的文件一行内容读到strLine缓冲区
    if(strstr(strLine,st2)==NULL)
    {
        fprintf(fp2,"%s\n",strLine);
    }
}
fclose(fp1);
fclose(fp2);

if ((fp1 = fopen("temp.txt", "rt")) == NULL)
{
    color(4);
    cout << "不能打开文件" << endl;
    return;
}
if ((fp2 = fopen("SeqList.txt", "wt")) == NULL)
{
    color(4);
    cout << "不能打开文件" << endl;
    return;
}
while (fscanf(fp1,"%[^\n]",strLine)!=EOF)    //循环读取每一行，直到文件尾
{
    fgetc(fp1);                //将fp所指向的文件一行内容读到strLine缓冲区
    fprintf(fp2,"%s\n",strLine);
```

**图 4.25 删除停车场文件离开的汽车的数据**

## 4.3 时空复杂度分析

| 函数名 | 描述 | 时间复杂度 | 空间复杂度 |
|---|---|---|---|
| Push_SeqList | 顺序栈的入栈 | O(1) | O(1) |
| Pop_SeqList | 顺序栈的出栈 | O(1) | O(1) |
| In_LQueue | 链队的入队 | O(1) | O(1) |
| Out_LQueue | 链队的出队 | O(1) | O(1) |
| Traverse_SeqList | 顺序栈的遍历 | O(n) | O(1) |
| Traverse_LQueue | 链队的遍历 | O(1) | O(1) |

# 第 5 章 测试与运行

## 5.1 测试与运行

1.在 12：00 存入车牌为鲁 H12345 的 small 型车，车辆位置为 001，界面提示已经输入了几组数据，现在正在进行第几次数据输入，以及停车场内已经停了多少辆车和当前便道上等待的车辆，再根据所提示的信息输入需要进行的操作。如图 5-1 所示：

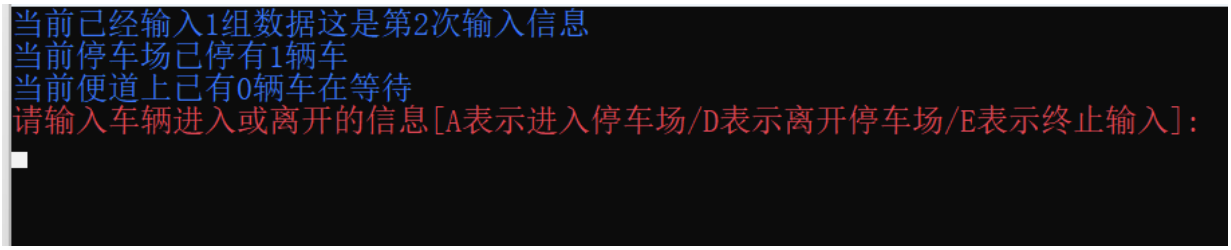


图 5.1　停车场停车

2.停车场已满，12：00 在便道存入车牌为鲁 D12345 的 small 型车，当停车场内停满车辆时，车辆将停到便道中，并且无法选择车位,如图 5-2

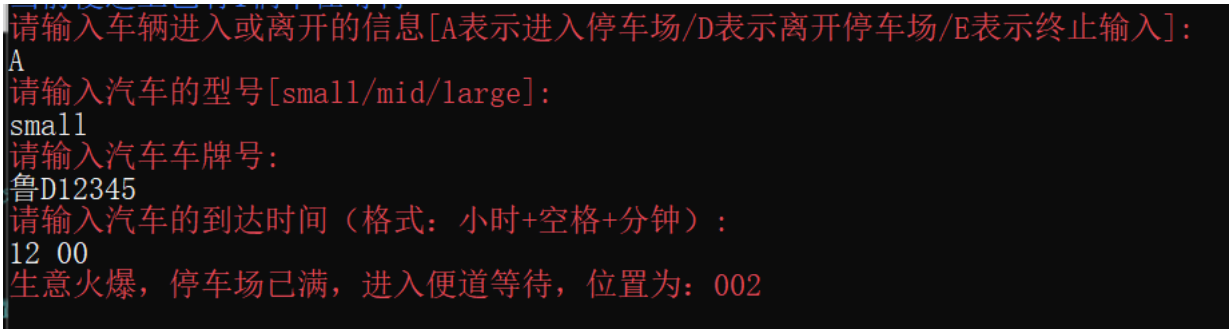


图 5.2　在便道停车

3.查询车库，在完成任意一个操作后，都可以输入‘E’退出输入操作，返回主界面，在主界面可选择进行车库查询功能，界面显示停车场内车辆停放信息。如图 5-3

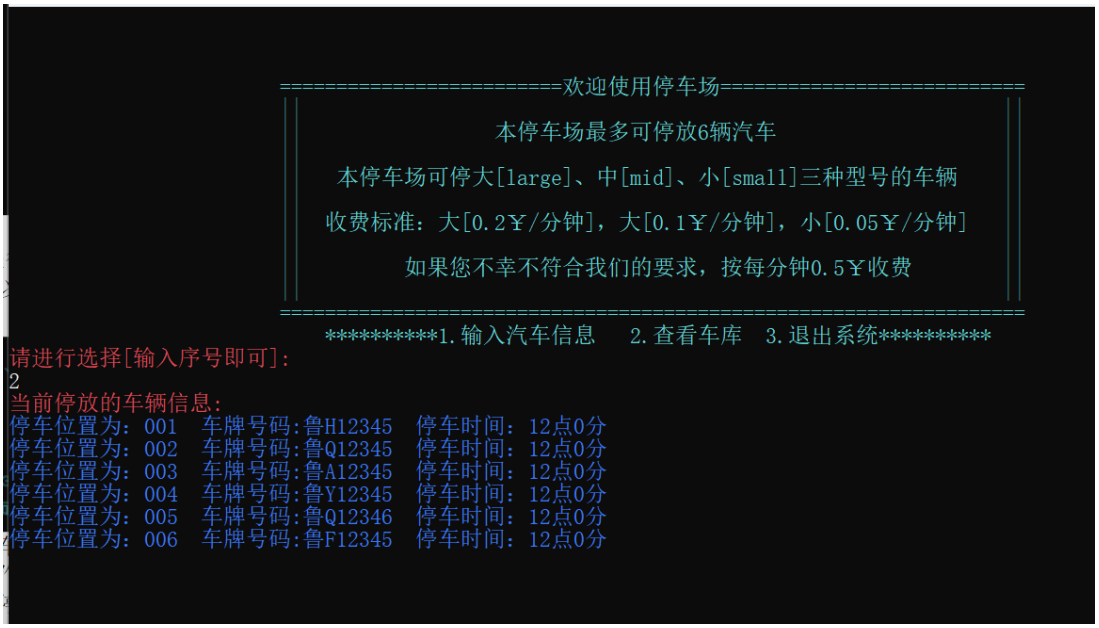


图 5.3　车库查询

4.便道上无车时，进行取车操作，在主界面输入‘1’后再输入‘D’进行取车操作，根据界面提示，依次输入车牌号，以及离开时间，此时在被取车的前面的车会进入临时停车场等待，待车被取走后，临时停车场中的车依次进入停车场。界面就会显示停放时间以及收费标准和停放费用。如图 5-4

```
当前已经输入1组数据这是第2次输入信息
当前停车场已停有1辆车
当前便道上已有0辆车在等待
请输入车辆进入或离开的信息[A表示进入停车场/D表示离开停车场/E表示终止输入]:
D
请输入汽车车牌号:
鲁H12345
请输入汽车的离开时间（格式：小时+空格+分钟）：
12 00
时间不合法，请重新输入:
12 30
车牌号为鲁H12345的车开出
您的车辆型号符合大车型，我们对你进行0.05￥/分钟的收费标准进行收费
停留时间：0小时30分钟      收费：1.5元
欢迎下次光临~
当前已经输入2组数据这是第3次输入信息
当前停车场已停有0辆车
当前便道上已有0辆车在等待
请输入车辆进入或离开的信息[A表示进入停车场/D表示离开停车场/E表示终止输入]:
```

**图 5.4　正常取车**

5. 便道中有车时，停车场中有车离开，正常取车操作后，便道有车，便道中的车进入停车场，由于车位有限，此时停在停车场中的车无法选择车位，将会停在上一辆离开的车的车，并且此时停入的车的停车时间将会更新为上一辆车离开的时间，以便计算车费。如图 5-5

```
D
请输入汽车的离开时间（格式：小时+空格+分钟）：
12 10
车牌号为鲁F12345的车让路
车牌号为鲁E12345的车让路
车牌号为鲁D12345的车让路
车牌号为鲁C12345的车让路
车牌号为鲁B12345的车让路
车牌号为鲁A12345的车开出
车牌号为鲁B12345的车开入
车牌号为鲁C12345的车开入
车牌号为鲁D12345的车开入
车牌号为鲁E12345的车开入
车牌号为鲁F12345的车开入
车牌号为鲁G12345的车开入
您的车辆型号符合大车型，我们对你进行0.05￥/分钟的收费标准进行收费
停留时间：0小时10分钟      收费：0.5元
欢迎下次光临~
当前已经输入8组数据这是第9次输入信息
当前停车场已停有6辆车
当前便道上已有0辆车在等待
请输入车辆进入或离开的信息[A表示进入停车场/D表示离开停车场/E表示终止输入]:
```

**图 5-5　便道中有车时取车**

6. 在对停车场进行操作时，用户会出现手误出现输入错误的情况，我们对此进行提示并再次输入。

在系统主界面时，规定输入 1—3 的数字，输入错误将会提示并重新输入：

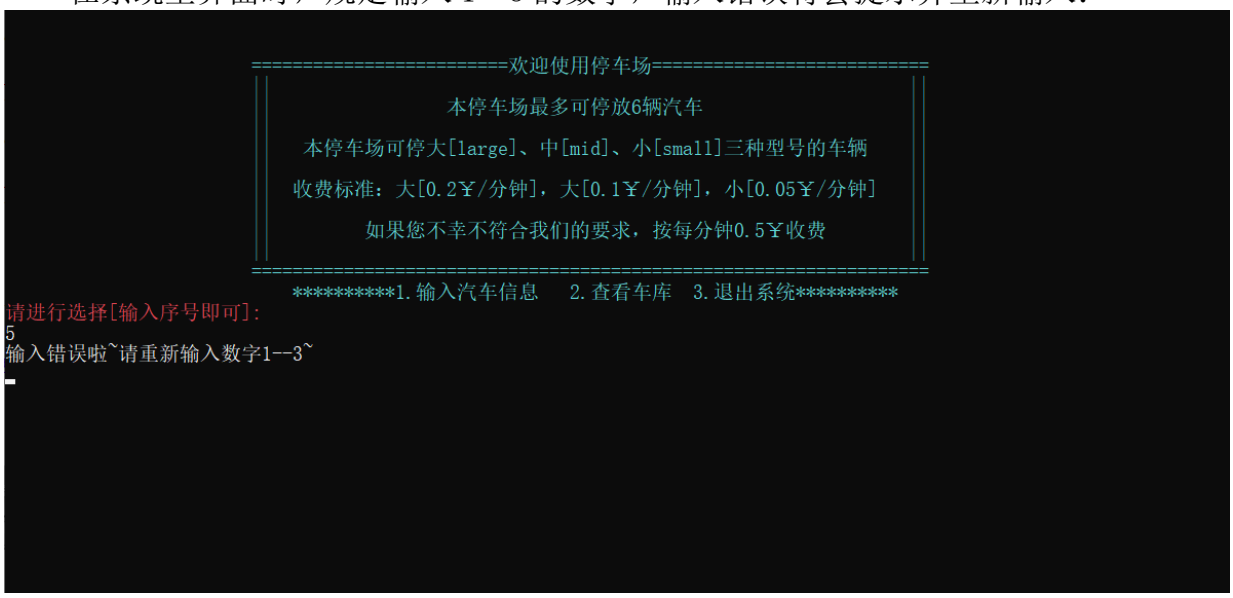


**图 5.6 系统选择输入错误**

在输入操作信息时，规定三个输入 A/D/E，输入错误将提十四并重新输入

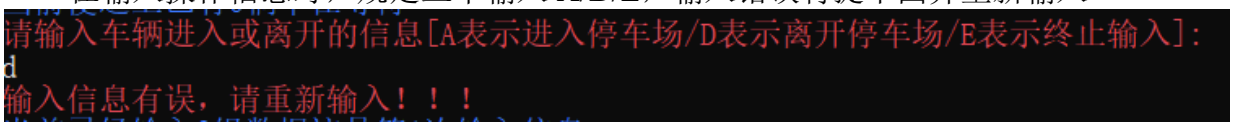


**图 5.7 操作信息输入错误**

在输入车牌号时，我们规定车牌号格式为省份+发证机关+5 位号码，输入格式错误将会提示并重新输入：


**图 5.8 车牌信息输入错误**

在输入汽车到达时间时，我们规定输入时间格式位 24 小时制，输入错误将胡提示并重新输入：

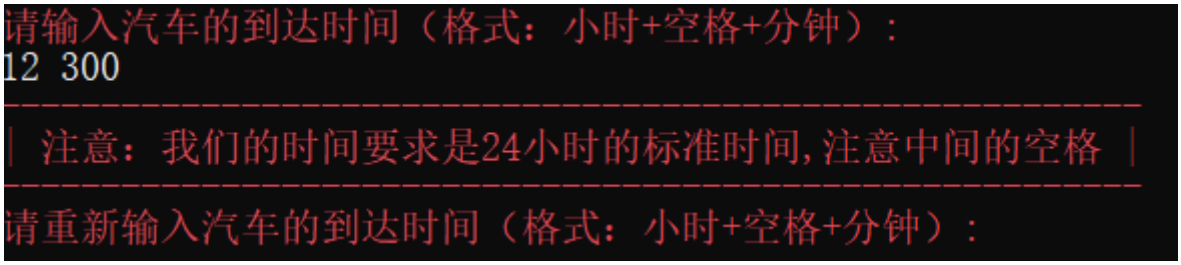


**图 5.9 停车时间输入错误**

在停车并选择车位时，输入停车场已占有的车位或者停车场没有的车位会提示并重新输入：

```
停车场未满，正在进入停车场~
当前空闲车位有：1 2 3 4 5 6
请选择车位~
8
输入有误，请重新选择车位~
```

**图 5.10 车位信息输入错误**

停车时，输入的车牌号在停车场中已经存在，会提示并重新输入：

```
请输入汽车车牌号:
鲁L11111
0
你输入的车牌号已经在车库当中了，请检查你的车牌号是否正确：
```

**图 5.11 车牌号重复**

取不在停车场中的车，输入一个没有停在停车场中的车，因为不在停车场中，所以，无法取出，界面会显示“该停车场内没有该车辆”如图 5-6

```
请输入车辆进入或离开的信息[A表示进入停车场/D表示离开停车场/E表示终止输入]:
D
请输入汽车车牌号:
鲁G12345
该停车场内没有该车辆~
当前已经输入7组数据这是第8次输入信息
当前停车场已停有6辆车
当前便道上已有1辆车在等待
请输入车辆进入或离开的信息[A表示进入停车场/D表示离开停车场/E表示终止输入]:
```

**图 5. 12 停车场不存在车辆**

当取车时，输入的取车时间小于停车时间会提示输入的时间不合法并重新输入：

```
请输入汽车的离开时间（格式：小时+空格+分钟）：
12 1
时间不合法，请重新输入:
```

**图 5.13 取车时间不合法**

7.文件操作

本系统共有两个文件：SeqList.txt（存储停车场的数据）和 LQueue.txt（存储便道中汽车的数据）。


**图 5.14 停车时 SeqList.txt 文件**


**图 5.15 停车场满时 LQueue.txt 便道车位**

## 5.2 系统使用说明书

系统主界面选择 1—3 进行相应的操作，1 为输入汽车信息，2 为查看车库，3 为退出系统。

输入汽车信息界面选择 A/D/E，A 为汽车进入停车场，D 为汽车离开停车场，E 为终止操作输入，退出输入。

在汽车进入停车场时，输入汽车的型号、汽车的车牌号、汽车停入停车场的时间，并选择车位。

在汽车离开停车场时，输入汽车的车牌号以及汽车离开的时间。

汽车型号我们有 small、mid、large 以及其他四种车型，收费标准依次为 0.05，0.1，0.2，0.5。

车票号我们遵循普通车牌号格式，省份简称+发证机关+五位号码。

# 第 6 章 总结与心得

通过这一段时间的课程设计，让我们对《数据结构》这一课程中所学到的内容加深了理解，学习到了如何将自己所学到的知识结合到实践当中；同时，通过对停车场模拟管理系统的开发，让我们加深了对团队合作的理解与感悟，懂得了交流的重要性。我们发现，在平时的学习中，仅仅的将代码背过，那么距离能够写出代码是远远不够的，在课程设计中，你不仅要有理解学习代码的能力，你还要能对用户的需求进行系统分析，完成用户提出的功能。这就要求我们不仅要有写出代码的本事，你还要能分析问题的能力，并且在分析完这些需求后，还要能够用计算机语言表示出来。而且，这每一件需求都不是孤立的，这和我们平时机测做的题是有很大的不同的，平时的我们只需要能够独立的完成这一道题，能够让代码运行正确就可以，但是，在课程设计中，一旦某个环节处理不好，在别的地方对代码进行调用的时候，就会出 bug，这极大的锻炼了我们融会贯通的能力。可以肯定的是，在我们解决完一个个问题的时候，我们的内心对编码的热爱就更进一分。

此次项目是我们独立完成的第一个中小型系统，我们切实的体会到了程序功能实现的喜悦，也感受到了遇到问题的迷茫，解决问题时候的执着。在这场程序设计中，使我们很好地了解了开发过程中，坚持的重要性。当我们终于成功的完成所有功能运行的时候，我们收获到了极大的成就感与喜悦感，但是我们并没有只沉浸在代码实现的喜悦中我们决定对这个代码进行一些改造升级，让这个代码更加的具有美感，于是我们加入了界面等待页面，使得代码更加的灵动，同时在输入车辆格式出错的时候我们加入了正确的输入格式的提示，是得其更加具有人性。

在此次课程实际中，我们小组开发的停车场模拟系统，基本上可以完成每一项功能。汽车进入停车场的信息、离开停车场的信息以及通道上的信息都可以在程序上一一实现。在此次课程设计当中，我们用到了课程中学到的栈和队列的知识，以及顺序表链表模拟栈和队列的操作，顺序栈模拟停车场，链式队列模拟便道。通过这次实训，切实的提高了我们的能力，至少我们掌握了顺序表实现栈，链式表实现队列，以及在网上查到的一些函数用法。我们相信学无止境，在未来的路上我们会继续的努力向前！

# 参考文献

[1]《数据结构（C语言版）》，严蔚敏、吴伟民，清华大学出版社，1997.
[2]《大话数据结构》，程杰，清华大学出版社，2011.
[3][C语言教你怎么改变字体颜色]
https://blog.csdn.net/qq_31975227/article/details/51758461
[4][strstr(str1,str2)函数使用时注意事项]
https://blog.csdn.net/ludaoyi88/article/details/52819448