南京信息工程大学

《数据结构》课程设计

题　　目　双层停车场管理

学生姓名 _____

学　　号 _____

学　　院　计算机学院

专　　业　计算机科学与技术

指导教师 _____

二〇二四 年 十二 月 三十 日

目 录

双层停车场管理

南京信息工程大学计算机学院，江苏 南京 210044

1 设计背景

1.1 问题描述

设有一个双层停车场，每层有 3 个空位，汽车进入和离开停车场的位置相互独立，当第一层停车场停满后才允许使用第二层（停车场可用一个二维数组实现，每个数组存放一个车牌号码；每辆车的信息包括车牌号码、层号、车位号、进入时间、离开时间和停车费用）。若场内所有车位停满，则后来的汽车只能在门外的便道上等候，一旦有车辆离开，则排在变道上的第一辆车即可进入停车。每辆停放在车场的车离开时必须按它的停留时间缴纳费用。试编写一个模拟程序进行停车场管理，要求：

（1）假设停车场的初始状态第一层已经停有了四辆车，起始车位号依次为 1-4，进入时间依次为 09:08:20，09:30:23，09:47:45，09:58:20。

（2）停车操作：当一辆车进入停车场时，先输入其车牌号码，进入时间，再为它分配一个层号和一个车位号（分配前先查询车位使用情况）。

（3）收费管理（取车）：当有车离开时，输入其车牌号码，离开时间，按停车时间计算停车费用，输出其各个信息，包括车牌号码，停留时间，停车费用，并将该车对应的车位设置为可使用状态。

（4）显示停车场剩余空位和便道上车辆数。

（5）查找车辆（停车场中）：输入车牌号码，查找对应车辆，输出其层号和车位号。

（6）显示所有车辆信息：按用户需求选择输出停车场中或者便道上全部车辆的信息。

表 1 停车收费标准

停车时间	收费标准
30 分钟（内）	免费
30 分钟-1 小时（含）	5 元
1 小时以上	5 元/小时 24 小时内（含）连续停车 60 元封顶
注：停车 30 分钟以上，不足 1 小时的按 1 小时收费	

1.2 问题分析

为了实现双层停车场的高效管理，本系统采用了三种数据结构：二维数组、循环队列和

哈希表。首先，二维数组用于存储停车场的车位信息，直观地将车牌号按层号和车位号存储，便于快速查询和更新车位状态；其次，循环队列管理便道上等待的车辆，遵循先进先出原则，确保车辆按顺序进入车场，并通过循环结构提高空间利用率；最后，哈希表用于通过车牌号快速查找车辆信息，满足频繁查询的需求。这三种数据结构各司其职，既能高效管理车位和排队车辆，又能快速查找和更新车辆信息，保证了停车场管理系统的高效运行。

2 设计方案

2.1 算法思想

本系统的算法思想围绕高效分配、管理和查询停车位展，结合哈希表、二维数组和循环队列的优势，实现了停车场的动态分配和实时管理。系统采用二维数组表示停车场结构，通过映射常量将层号和车位号映射为数组索引，以快速分配车位。停车操作优先分配第一层车位，若第一层满，则转至第二层；若整个停车场满，则将车辆加入循环队列等待，确保停车场空间利用率最大化。取车操作中，系统通过哈希表快速定位车辆所在的层号和车位号，计算停车时间差以确定费用，释放车位后优先从便道取出车辆补充停车场。车辆查询也利用哈希表，用户只需输入车牌号即可快速获取车辆位置信息。

哈希表在系统中负责高效映射车牌号码与停车位信息。车牌号码因其唯一性成为哈希表的关键字，通过特定的哈希函数生成哈希值，哈希函数为 hash=(hash×31+当前字符的 ASCII 值)%哈希表长度。该哈希函数从车牌号的第二个字符，开始避免处理中文带来的复杂性，结合字符 ASCII 值与乘法因子 31 计算哈希值，以减少冲突，并通过线性探测法解决冲突问题；其中哈希表长度为停车场所有停车位数加 1。哈希地址是停车信息的逻辑位置，而停车位置（层号与车位号）是实际物理位置，哈希地址与停车位置之间的转换关系体现在哈希表中存储的层号和车位号，通过车牌号码快速找到对应的哈希地址，并根据存储的层号和车位号返回实际停车位置，实现逻辑地址与物理位置的高效映射。

便道通过循环队列实现动态管理，用于记录因停车场已满而等待的车辆。循环队列通过 front 和 rear 指针实现首尾相连的逻辑结构，确保队列操作的有序性与空间利用率。车辆离开时，若队列不为空，则系统从队列头取出等待车辆并分配车位，避免资源浪费。此外，便道车辆信息可通过遍历队列快速输出，实时反映系统状态。

此系统还设计了简洁的停车费用计算逻辑，能够处理跨天停车情况。停车时间通过解析进入和离开时间，统一转换为秒数后计算时间差。费用按小时计费，不足 30 分钟不进位，超过 30 分钟则进位至下一小时，超过 12 小时封顶为 60 元。此算法既考虑了用户需求，又确保收费规则的简明高效。

综上，本系统通过二维数组的高效车位管理、哈希表的快速查询和循环队列的有序等待，构建了一套完整的停车管理方案。结合停车费用的精准计算，系统在满足用户需求的同时，也提升了停车场管理的实时性和高效性。

2.2 功能模块分析

根据对双层停车场管理系统需求的分析，本系统主要包括停车操作、取车操作和显示车辆信息功能模块。停车操作模块主要包含车牌号是否存在检查模块；取车操作模块主要由根据车牌号码查询车辆位置信息模块和停车费用计算模块组成；显示车辆信息模块内分为显示停车场车辆信息模块和显示便道车辆信息模块。系统模块划分如图 2-1 系统功能模块图所示。

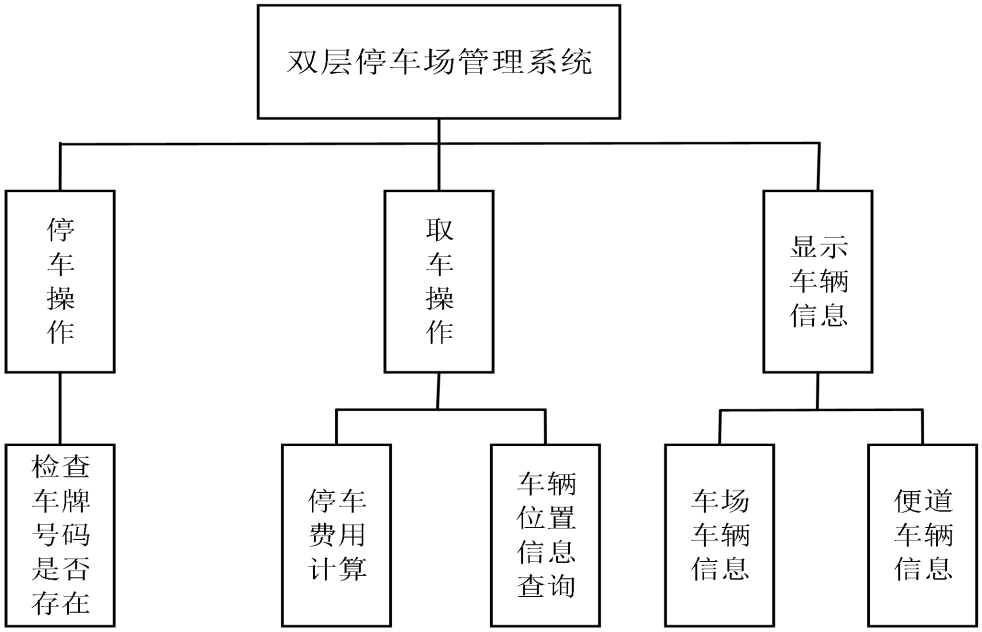

图 2-1 系统功能结构图

（1）检查车牌号码是否存在（IsCarAlreadyExists）

主要任务：判断指定车牌号码是否已经存在于停车场或便道中，以避免重复停车或重复排队。该模块首先通过哈希表快速查找车牌号，判断车辆是否已停放在停车场；如果未找到，则继续遍历便道（循环队列），检查是否有相同车牌号的车辆正在排队等待停车。通过以上两个步骤，该模块能够精准判断车辆是否已经在系统中存在，为停车操作提供逻辑支持，确保车辆管理的唯一性和系统操作的准确性。

（2）停车操作（ParkCar）

主要任务：为新到达的车辆分配停车位，或者将车辆加入便道等待。当车辆到达时，首先检查车牌号码是否已存在于停车场或便道中，若存在则直接返回，避免重复停车或排队。若车牌号不存在，则在停车场的二维数组中按照优先顺序（从第一层开始依次分配）查找空闲车位，找到后将车辆信息存入停车场，并通过哈希表记录其车牌号与停车位的映射关系。若停车场已满，则检查便道是否有空位，将车辆车牌号加入循环队列等待并显示其在便道中的序号。通过高效分配车位或有序安排车辆排队，此模块确保了停车操作的准确性与系统资源的最大化利用。具体如图 2-2 所示。

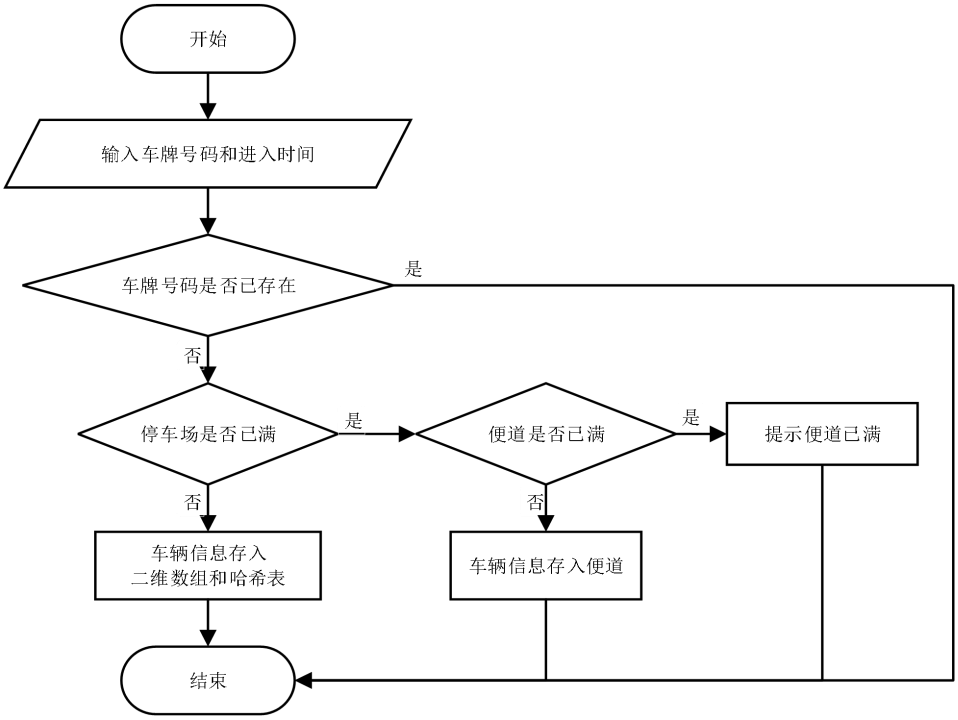

图 2-2 停车操作模块流程图

（3）停车费用计算模块（CalculateFee）

主要任务：根据车辆的进入时间和离开时间计算停车时长，并按照收费规则得出停车费用。此模块首先解析进入时间和离开时间，将其转换为总秒数以便计算时间差，同时处理跨天停车的情况，确保时间差为正值。接着，将停车时长换算为小时、分钟和秒，并依据收费规则计算费用：不足 30 分钟不计入下一小时，超过 30 分钟进位至下一小时，停车时间超过 12 小时按 60 元封顶。

（4）车辆位置信息查询模块（FindCar）

主要任务：通过车牌号码快速查找车辆在停车场中的实际停车位置，并返回其层号和车位号。此模块通过哈希表对车牌号码进行查询，利用哈希地址映射快速定位车辆的存储位置。如果查询成功，则输出车辆的层号和车位号；若未找到，则提示用户该车牌号码对应的车辆不存在。此模块利用哈希表高效的查找特性，实现了停车场车辆信息的快速检索，提升了系统的查询效率和用户体验。具体如图 2-3 所示。

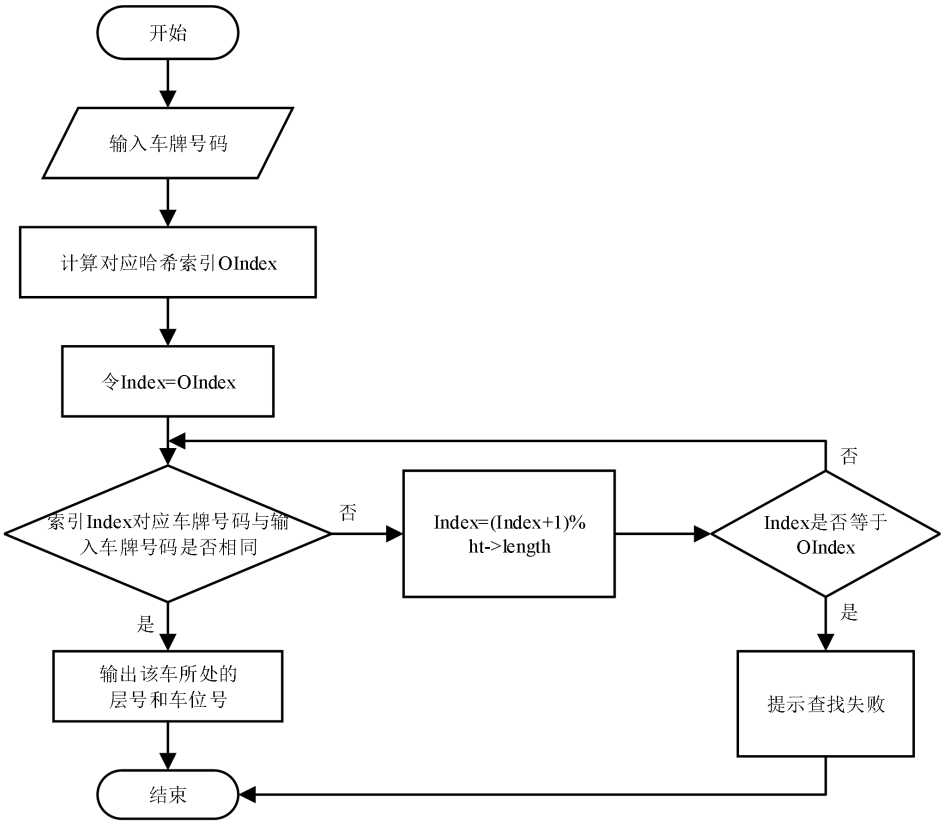

图 2-3 车辆位置信息查询模块流程图

（5）显示车辆信息模块（ShowParkingStatus 和 ShowQueueStatus）

主要任务：实时展示停车场或便道中的车辆状态，帮助用户了解当前车辆分布和排队情况。 其中，ShowParkingStatus 模块通过遍历停车场的二维数组，按层依次显示每个车位的状态，包括已占用车位的车牌号或空闲标记，直观反映停车场的使用情况；ShowQueueStatus 模块通过遍历便道的循环队列，按顺序输出所有等待车辆的车牌号码及其排位号，清晰展示便道中车辆的排队状态。通过实时的车辆信息展示，该模块可以为用户提供了停车场和便道的全局视图，提升了管理的透明性与用户体验。

（6）取车操作模块(LeaveCar)

主要任务：处理车辆离开停车场的操作，包括计算停车费用、释放车位，并将便道中的首辆车分配至空闲车位。模块首先通过哈希表快速查找离开车辆的层号和车位号，若查找成功，则获取车辆的进入时间和离开时间，若计算停车时长并根据收费规则得出停车费用。随后，清空对应车位信息，将其标记为可用，并从哈希表中删除该车辆的记录。若便道中有等待的车辆且停车场尚有空位，则将便道首辆车移入停车场，并将其离开时间作为进入时间记录。反之，若查找失败在，则提示未找到该车辆，结束本次操作。该模块通过高效的取车操作和便道车辆的动态分配，确保停车场资源的最大化利用与车辆管理的流畅性。具体如图 2-4 所示。

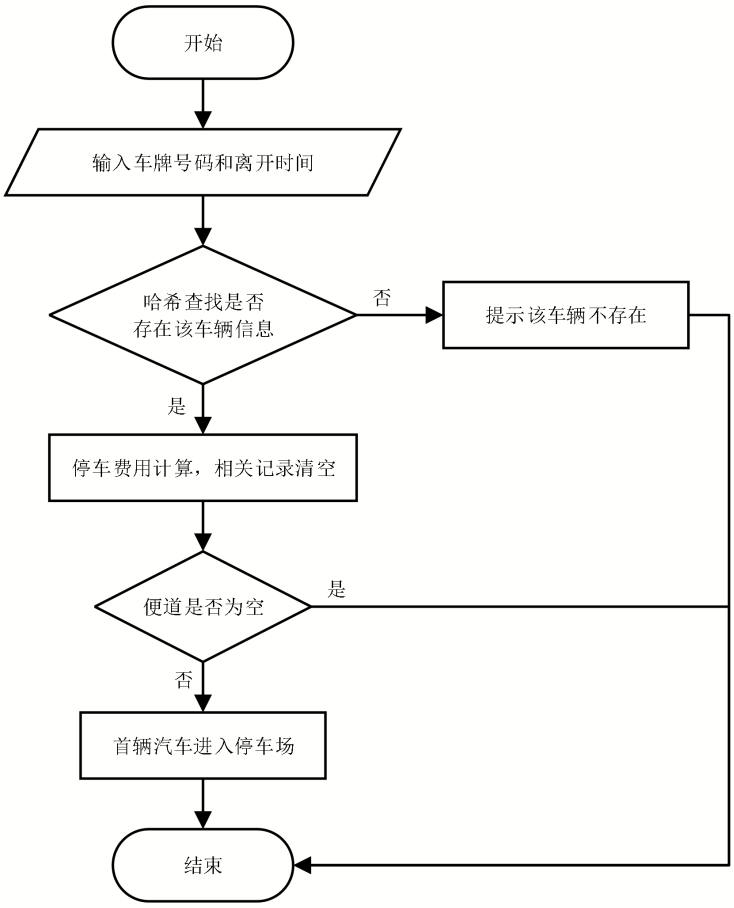

图 2-4 取车操作模块流程图

（7）除了主要功能模块外，该管理系统还包括停车场剩余车位数量查询和便道等待车辆数查询模块，这两个模块通过全局变量实现了简单高效的状态管理。系统中定义了全局变量 parkingCount 和 queueCount，分别用于记录停车场中已占用的车位数量和便道中等待的车辆数量。通过实时更新这两个变量，系统能够快速查询停车场的剩余车位数和便道中的排队车辆数，无需额外的遍历或复杂计算，极大地提升了查询效率和系统响应速度。

底层数据结构图如图 2-5 所示，系统模块关系调用图如图 2-6 所示。

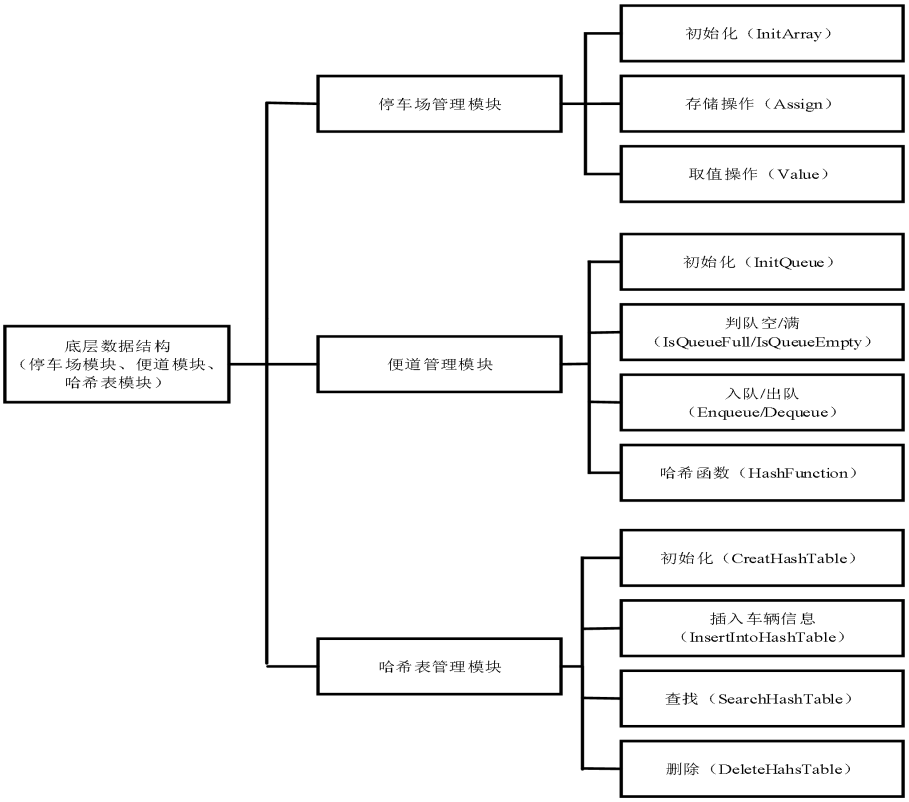

图 2-5 底层数据结构图

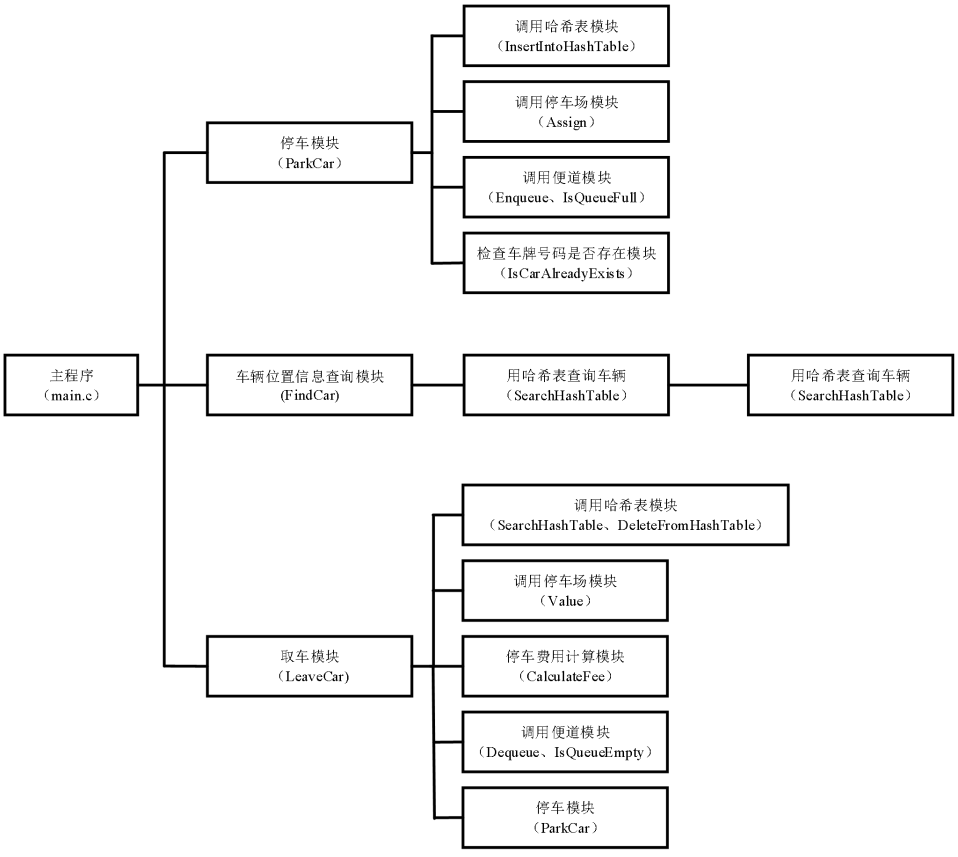

图 2-6 系统模块关系调用图

2.3 功能模块实现

（1）检查车牌号码是否已存在模块（IsCarAlreadyExists）

优点：停车场部分通过哈希表查找，时间复杂度接近 O(1)；便道部分通过循环队列逐一遍历，时间复杂度为 O(n)，其中 n 是便道中排队车辆的数量。由于实际情况中 n 的值不会很大，所以整体效率较高，适合实时检查车辆是否已存在。

```
bool IsCarAlreadyExists(const HashTable *ht,const Queue *waitingQueue,const char *license)
{
    int level, spot;
    // 检查停车场（哈希表）
    if (SearchHashTable(ht, license, &level, &spot)) {
        printf("\n");
        printf("该车辆已停放在停车场中。\n");
        return true;
    }
    // 检查便道（队列）
    int current = waitingQueue->front;
    while (current != waitingQueue->rear) {
        if (strcmp(waitingQueue->elements[current].data, license) == 0) {
            printf("\n");
            printf("该车辆已在便道中等候。\n");
            return true;
        }
        current = (current + 1) % QUEUE_MAX_SIZE;
    }
    return false;   // 停车场和便道中都不存在
}
```

（2）停车操作模块（ParkCar）

优点：此代码可以轻松扩展为支持动态调整车位大小或多便道管理。

```
void ParkCar(Array *parkingLot,HashTable *ht,Queue *waitingQueue,const char *license,const char *entryTime,int *level,int *spot){
    // 如果车牌号已存在，直接返回
    if (IsCarAlreadyExists(ht, waitingQueue, license)) {
        return;
    }
```

```
    Car car;
    strcpy(car.license,license);
    strcpy(car.entryTime,entryTime);
    car.flag=true;
    int i,j;
    for(i=0;i<MAX_LEVELS;i++){
        for(j=0;j<MAX_SPOTS;j++){
            int offset=i*parkingLot->constants[0]+j;
            if(!parkingLot->base[offset].flag){   // 找到空位
                car.level=i+1;
                car.spot=j+1;
                // 存入二维数组
                if(!Assign(parkingLot,i,j,&car)){
                    printf("分配车位失败！\n");
                    return;
                }
                parkingCount++;
                printf("尊敬的 %s 用户，您的停车位置在停车场 %d 层 %d 号车位。
\n",license,i+1,j+1);
                // 存入哈希表
                if(!InsertIntoHashTable(ht,license,i+1,j+1)){
                    printf("车辆存入哈希表失败。\n");
                    return;
                }
                // 返回层号和车位号
                if(level)
                    *level=i+1;
                if(spot)
                    *spot=j+1;
                return;
            }
        }
    }
    // 若停车场已满，将车辆加入便道
    if(IsQueueFull(waitingQueue)){
        printf("尊敬的%s 用户，便道已满。\n");
```

```
    }else{
        Enqueue(waitingQueue,license);  // 加入便道
        queueCount++;
        printf("尊敬的%s用户，停车场已满，您已排队进入便道,序号为%d。
\n",license,queueCount);
    }
}
```

（3）停车费用计算模块（CalculateFee）

优点：考虑了跨天停车和秒级时间精度的情况，保证计算结果准确（24 小时及以内）；代码支持按小时、分钟精确收费，并可以扩展为动态定价或其他复杂计费模式。

```
int CalculateFee(const char *entryTime,const char *exitTime,int *hours,int *minutes,int *seconds){
    int entryHours,entryMinutes,entrySeconds;
    int exitHours,exitMinutes,exitSeconds;
    // 解析时间
    sscanf(entryTime,"%d:%d:%d",&entryHours,&entryMinutes,&entrySeconds);
    sscanf(exitTime,"%d:%d:%d",&exitHours,&exitMinutes,&exitSeconds);
    // 计算总秒数
    int entryTotalSeconds=entryHours*3600+entryMinutes*60+entrySeconds;
    int exitTotalSeconds=exitHours*3600+exitMinutes*60+exitSeconds;
    int parkingSecond=exitTotalSeconds-entryTotalSeconds;
    if(parkingSecond<0) {
        parkingSecond+=86400;  // 处理跨天情况，补偿一天的秒数
    }
    *hours=parkingSecond/3600;
    parkingSecond%=3600;
    *minutes=parkingSecond/60;
    *seconds=parkingSecond%60;
    // 停车费用计算
    if(*hours*3600+*minutes*60+*seconds>=43200){
        return 60;  // 超过或等于 12 小时封顶
    } else if (*minutes<30||*minutes==30&&*seconds==0){
        return 5*(*hours);  // 不超过 30 分钟不进位
    }else if(*minutes==30&&*seconds>0||*minutes>30){
        return 5*(*hours+1);  // 超过 30 分钟小时数加 1
    }
```

```
}
```

（4）车辆位置信息查询模块（FindCar）

优点：使用哈希表存储车辆信息，查询速度接近 O(1)；查询结果直接反馈给用户，输出层号和车位号，反馈直观。

```
void FindCar(const HashTable *ht,const char *license){
    int level,spot;   // 用于存储查找到的层号和车位号
    if(SearchHashTable(ht, license, &level, &spot)){
        printf("车牌号码：%s   层号：%d   车位号：%d\n", license, level, spot);
    } else {
        printf("未找到车牌号码为 %s 的车辆。\n", license);
    }
    printf("\n");
}
```

（5）显示车辆信息模块（ShowParkingStatus 和 ShowQueueStatus）

优点：格式化输出停车场中每一层的车位状态或便道中车辆状态，便于管理员和用户查看，信息清晰直观。

ShowParkingStatus 模块：

```
void ShowParkingStatus(const Array *parkingLot){
    printf("停车场车辆信息：\n");
    int i,j;
    for(i=0;i<MAX_LEVELS;i++){
        printf("第%d 层：",i+1);
        for(j=0;j<MAX_SPOTS;j++){
            Car car;
            if(Value(parkingLot,i,j,&car)){
                if(car.flag&&car.license[0]!='\0'){
                    printf("[%8s] ",car.license);
                }else{
                    printf("[ 未   知 ] ");
                }
            }else{
                printf("[ 空   闲 ] ");
            }
        }
```

```
        printf("\n");
    }
    printf("\n");
}
```

ShowQueueStatus 模块：

```
void ShowQueueStatus(const Queue *waitingQueue){
    if(IsQueueEmpty(waitingQueue)){
        printf("便道中无车辆。\n");
        printf("\n");
        return;
    }else{
        printf("便道车辆信息：\n");
        int current=waitingQueue->front;
        int position=1;
        while(current!=waitingQueue->rear){
            printf("排序号：%d 车牌号码：%s\n",position,waitingQueue->elements[current].data);
            current=(current+1)%QUEUE_MAX_SIZE;
            position++;
        }
        printf("\n");
    }
}
```

（6）取车操作模块（LeaveCar）

优点：便道和停车场结合管理，取车后自动检查便道，尽量减少空车位；停车费用试试计算，支持秒级时间精度的停车费用计算，逻辑清晰，易于调整收费策略。

```
void LeaveCar(Array *parkingLot,HashTable *ht,Queue *waitingQueue,const char *license, const char *exitTime){
    int level,spot;
    printf("\n");
    if(SearchHashTable(ht,license,&level,&spot)){
        int offset=(level-1)*parkingLot->constants[0]+(spot-1);   // 计算车辆的索引
        // 获取车辆信息
        Car car;
        Value(parkingLot,level-1,spot-1,&car);
        // 清空车位
```

```
        parkingLot->base[offset].flag=false;
        // 计算停车费用
        int hours,minutes,seconds;
        int fee=CalculateFee(car.entryTime,exitTime,&hours,&minutes,&seconds);
        printf("尊敬的 %s 用户，您的停车时长为 %d 时%d 分%d 秒 ，停车费为 %d 元，祝您一路顺风!\n",license,hours,minutes,seconds,fee);
        parkingCount--;
        // 删除哈希表中对应车辆信息
        DeleteFromHashTable(ht, license);
        // 检查车道是否为空，若不为空则进入停车，进入时间与离开汽车的离开时间相同，不考虑时间损失
        if(!IsQueueEmpty(waitingQueue)&&parkingCount<MAX_LEVELS*MAX_SPOTS){
            char waitingLicense[STRING_MAX];
            int newlevel,newspot;
            if(Dequeue(waitingQueue,waitingLicense)){   // 从便道取出第一辆车
        ParkCar(parkingLot,ht,waitingQueue,waitingLicense,exitTime,&newlevel,&newspot);
                queueCount--;
            }
        }
    }else{
        printf("未找到车牌号码为 %s 的车辆。\n",license);
    }
}
```

（7）在系统中，一些核心代码的实现对于提升停车场管理的效率和准确性至关重要。例如 HashFunction 和 SearchHashTable 函数，是哈希表模块的核心组成部分，用于快速映射和查找车牌号与停车位信息的对应关系。

HashFunction 函数：

```
int HashFunction(const char *license,int length){
    unsigned int hash=0;
    const char *start=license+1;   // 从第二个字符开始计算,因为第一个字符为汉字
    while(*start){
        hash=(hash*31+*start)%length;
        start++;
    }
    return hash;
```

```
    }
```

SerachHashTable 函数：

```
int SearchHashTable(const HashTable *ht,const char *license,int *level,int *spot){
    int index=HashFunction(license,ht->length);   // 计算哈希索引
    int originalIndex=index;
    while(ht->slots[index].flag){
        if(strcmp(ht->slots[index].license,license)==0){
            *level=ht->slots[index].level;
            *spot=ht->slots[index].spot;
            return 1;
        }
        index=(index+1)%ht->length;
        if(index==originalIndex){
            break;   // 已循环一圈
        }
    }
    return 0;
}
```

3 设计结果

3.1 预期结果

开始使用系统，初始化四辆汽车，接着显示对应操作编号。

车辆到达时，选择编号 1，输入该车辆的车牌号码和到达时间，若停车场和便道中无此车辆信息，则进行后序操作，反之显示停车场或便道已存在该辆汽车。若停车场未满，则显示该车当前处于停车场的层号和车位号，若停车场已满，则显示该车当前处于便道的位置。

车辆离开时，选择编号 2，输入该车辆的车牌号码和离开时间，若停车场不存在该车辆信息，则显示未找到该车；反之输出该车辆的停车时间和停车费用，接着判断便道是否为空，若不为空则拍第一的车辆进入停车场停车。

查询车辆位置信息（停车场中）时.选择编号 3，输入查询的车牌号码，若停车场中不存在该汽车，则显示未找到该车牌号码的汽车，反之显示该车牌号码对应汽车在停车场中的层号和车位号。

查看停车场所有车辆信息时，选择编号 4，返回每一层每一个车位号对应车辆的车牌号码，若无则显示空闲；查看便道所有车辆信息时，选择编号 5，按照排序号返回便道中的车辆信息。

查询停车场剩余停车位和便道车辆数时，选择编号 6，返回对应信息。

退出系统选择编号 7，若输入错误编号，系统会提示重新输入正确的编号。

3.2 运行结果

初始化四辆汽车，接着系统提示不同操作对应编号。具体运行如图 3-1 所示。

```
尊敬的 苏K5665L 用户，您的停车位置在停车场 1 层 1 号车位。
尊敬的 苏AABCD1 用户，您的停车位置在停车场 1 层 2 号车位。
尊敬的 苏AABCD2 用户，您的停车位置在停车场 1 层 3 号车位。
尊敬的 苏AABCD3 用户，您的停车位置在停车场 2 层 1 号车位。

欢迎使用车辆管理系统：
    1：车辆到达
    2：车辆离开
    3：车辆位置信息查询
    4：查看停车场所有车辆信息
    5：查看便道所有车辆信息
    6：查询停车场剩余停车位和便道车辆数
    7：退出系统
您好！请输入操作编号：
```

图 3-1 停车场初始化车辆状态图

接着选择编号 1，依次到达的四辆汽车的车牌号码分别为苏 AABCD4、苏 AABCD5、苏 AABCD6，苏 AABCD7,由于苏 AABCD5 进入停车场后，停车场已满，所以车辆苏 AABCD6、苏 AABCD7 进入便道,显示其在便道中的序号。具体运行如图 3-2 所示。

```
欢迎使用车辆管理系统：
    1：车辆到达
    2：车辆离开
    3：车辆位置信息查询
    4：查看停车场所有车辆信息
    5：查看便道所有车辆信息
    6：查询停车场剩余停车位和便道车辆数
    7：退出系统
您好！请输入操作编号：1

请输入车牌号码：苏AABCD5
请输入车辆到达时间（HH:MM:SS）：12:30:00
尊敬的 苏AABCD5 用户，您的停车位置在停车场 2 层 3 号车位。

欢迎使用车辆管理系统：
    1：车辆到达
    2：车辆离开
    3：车辆位置信息查询
    4：查看停车场所有车辆信息
    5：查看便道所有车辆信息
    6：查询停车场剩余停车位和便道车辆数
    7：退出系统
您好！请输入操作编号：1

请输入车牌号码：苏AABCD6
请输入车辆到达时间（HH:MM:SS）：13:00:00
尊敬的苏AABCD6用户，停车场已满，您已排队进入便道,序号为1。

欢迎使用车辆管理系统：
    1：车辆到达
    2：车辆离开
    3：车辆位置信息查询
    4：查看停车场所有车辆信息
    5：查看便道所有车辆信息
    6：查询停车场剩余停车位和便道车辆数
    7：退出系统
您好！请输入操作编号：1

请输入车牌号码：苏AABCD7
请输入车辆到达时间（HH:MM:SS）：14:00:01
尊敬的苏AABCD7用户，停车场已满，您已排队进入便道,序号为2。
```

图 3-2 车辆成功进入运行图

若继续选择编号 1，待停车辆的车牌号为苏 AABCD6,则会显示该车辆已存在。具体运行如图 3-3 所示。

```
欢迎使用车辆管理系统:
    1: 车辆到达
    2: 车辆离开
    3: 车辆位置信息查询
    4: 查看停车场所有车辆信息
    5: 查看便道所有车辆信息
    6: 查询停车场剩余停车位和便道车辆数
    7: 退出系统
您好! 请输入操作编号: 1

请输入车牌号码: 苏AABCD6
请输入车辆到达时间 (HH:MM:SS) : 10:10:10

该车辆已停放在停车场中。
```

图 3-3 车辆失败进入运行图

选择编号 2，输入要离开车辆的车牌号码苏 AABCD5、离开时间，输出停车时间和停车费用。由于便道不为空，所以序号为 1 的汽车苏 AABCD6 进入停车场停车，输出其停车位置的层号和车位号。具体运行如图 3-4 所示。

```
欢迎使用车辆管理系统:
    1: 车辆到达
    2: 车辆离开
    3: 车辆位置信息查询
    4: 查看停车场所有车辆信息
    5: 查看便道所有车辆信息
    6: 查询停车场剩余停车位和便道车辆数
    7: 退出系统
您好! 请输入操作编号: 2

请输入车牌号码: 苏AABCD5
请输入车辆离开时间 (HH:MM:SS) : 14:00:01

尊敬的 苏AABCD5 用户, 您的停车时长为 1时30分1秒 , 停车费为 10 元, 祝您一路顺风!
尊敬的 苏AABCD6 用户, 您的停车位置在停车场 2 层 3 号车位。
```

图 3-4 车辆成功离开运行图

若继续选择编号 2，离开车辆的车牌号码为苏 AABCD5，则会显示未找到车牌号为此的车。具体运行如图 3-5 所示。

```
欢迎使用车辆管理系统:
    1: 车辆到达
    2: 车辆离开
    3: 车辆位置信息查询
    4: 查看停车场所有车辆信息
    5: 查看便道所有车辆信息
    6: 查询停车场剩余停车位和便道车辆数
    7: 退出系统
您好! 请输入操作编号: 2

请输入车牌号码: 苏AABCD5
请输入车辆离开时间 (HH:MM:SS) : 20:20:20

未找到车牌号码为 苏AABCD5 的车辆。
```

图 3-5 车辆失败离开运行图

选择编号 3，分别查询车牌号码为苏 K5665L 和苏 AABCD5 的车辆。若查询成功则显示其所在层号和车位号；若不存在则显示未找到。具体运行如图 3-6 所示。

```
欢迎使用车辆管理系统:
    1: 车辆到达
    2: 车辆离开
    3: 车辆位置信息查询
    4: 查看停车场所有车辆信息
    5: 查看便道所有车辆信息
    6: 查询停车场剩余停车位和便道车辆数
    7: 退出系统
您好！请输入操作编号: 3

请输入车牌号码: 苏K5665L
车牌号码: 苏K5665L  层号: 1  车位号: 1

欢迎使用车辆管理系统:
    1: 车辆到达
    2: 车辆离开
    3: 车辆位置信息查询
    4: 查看停车场所有车辆信息
    5: 查看便道所有车辆信息
    6: 查询停车场剩余停车位和便道车辆数
    7: 退出系统
您好！请输入操作编号: 3

请输入车牌号码: 苏AABCD5
未找到车牌号码为 苏AABCD5 的车辆。
```

图 3-6 车辆位置信息查找运行图

接下来分别选择编号 4 和编号 5，输出停车场所有车辆信息和便道所有车辆信息，具体运行如图 3-7 所示。

```
欢迎使用车辆管理系统:
    1: 车辆到达
    2: 车辆离开
    3: 车辆位置信息查询
    4: 查看停车场所有车辆信息
    5: 查看便道所有车辆信息
    6: 查询停车场剩余停车位和便道车辆数
    7: 退出系统
您好！请输入操作编号: 4

停车场车辆信息:
第1层: [苏K5665L] [苏AABCD1] [苏AABCD2]
第2层: [苏AABCD3] [苏AABCD4] [苏AABCD6]

欢迎使用车辆管理系统:
    1: 车辆到达
    2: 车辆离开
    3: 车辆位置信息查询
    4: 查看停车场所有车辆信息
    5: 查看便道所有车辆信息
    6: 查询停车场剩余停车位和便道车辆数
    7: 退出系统
您好！请输入操作编号: 5

便道车辆信息:
排序号: 1    车牌号码: 苏AABCD7
```

图 3-7 停车场和便道车辆信息输出图

输入错误编号 8，系统会提示重新输入正确的编号；最后选择编号 7，退出系统。具体运行如图 3-8 所示。

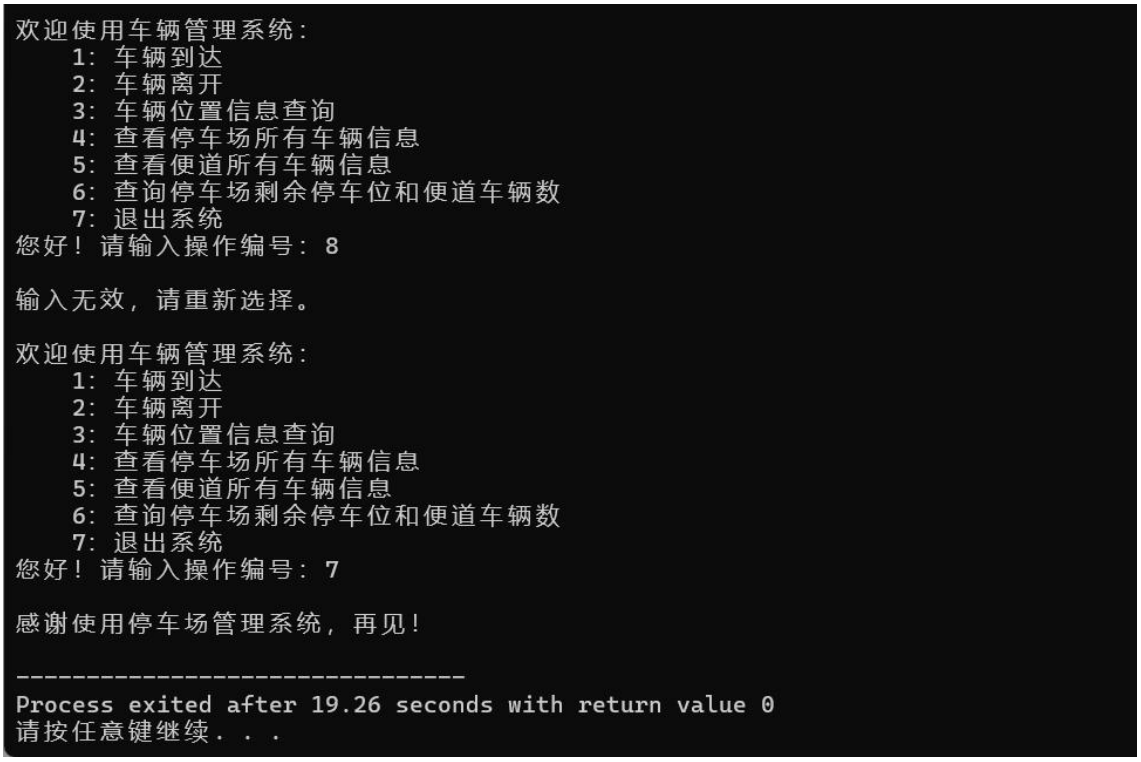

图 3-8 输入错误编号提示与系统退出运行图

4 总结

在系统开发之前，我对停车场管理的需求进行了详细分析，设计了一个双层停车场，每层具有固定数量的车位。当停车场满员时，车辆可以进入便道队列等待。我们将系统功能分解为多个核心模块，包括停车操作、车辆离开、车牌号查询、停车状态显示、便道队列管理等，为整个系统的开发提供了明确的方向和结构。

在功能实现阶段，我充分考虑了功能的高效性与合理性，设计并采用了以下数据结构：二维数组用于管理停车场的车位状态，存储每个车位的车辆信息；哈希表用于快速查找车辆的停车位置，保证查询操作高效且准确；循环队列用于管理便道中的等待车辆，保证先进先出（FIFO）的顺序性。基于这些设计，系统成功实现了核心功能，包括停车操作、车辆离开操作、车辆查询与状态显示以及用户交互界面。

尽管系统已实现主要功能，但在实际应用中仍存在一些需要改进的地方。例如，当前程序默认车辆停留时间在 24 小时及以内，如要处理车辆停留时间超过 24 小时的情况，则系统需进一步优化。此外，为了便于程序测试，我将停车场每层的最大停车容量设置为 3 个车位。在实际应用中，可以根据实际需求修改停车场容量的宏定义（MAX_SPOTS），同样也可以调整停车场层数的宏定义（MAX_LEVELS）以适配不同场景需求。通过以上设计和改进建议，系统在功能实现和适应性方面能够更加完善。