小组编号	6

成绩	

华中师范大学计算机科学系
实 验 报 告 书

实验题目：停车场管理系统的实现

课程名称：数据结构实验

主讲教师：刘巍

小组成员姓名：李坤霖，姜高峰，刘镇东

实验时间：2024.11.4

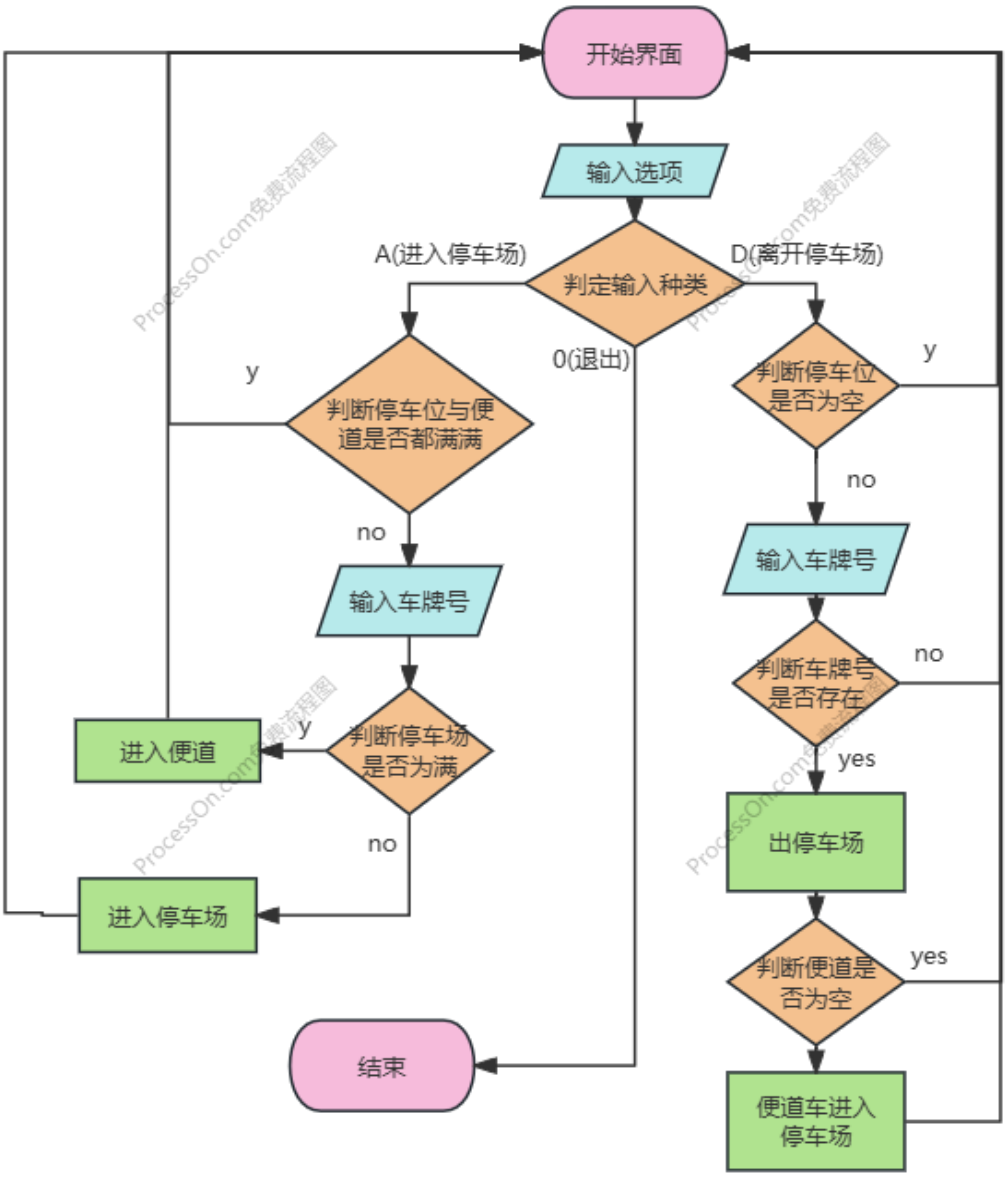
一、实验目的：
编写程序，设计一个停车库管理程序

二、实验内容：
编写程序，利用栈与队列的基本操作来实现停车场管理程序

三、实验环境：
编程语言：c++
编译环境：visual studio code，gcc14.2.0，-std=c++20 -O2 -fPIC -Wall -fno-asm -lm march=native -Wl,-stack=268435456
操作系统：Windows11

四、问题描述
设停车场内只有一个可停放几辆汽车的狭长通道，且只一个大门可供汽车进出。汽车在停车场内按车辆到达时的先后顺序，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），若车场内已停满几辆汽车，则后来的汽车只能在门外的便道上等候，一旦停车场内有车开走，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，由于停车场是狭长的通道，在它之后开入停车场的车辆必须先退出车场为它让路，待该辆车开出大门外后，为它让路的车辆再按原次序进入车场。在这里假设汽车不能从便道上开走。试设计一个停车管理程序

五、主程序流程和函数说明



Menu()

描述： 显示操作提示

用途： 给予用户操作提示

InitStack(Parking)

描述： 初始化栈

用途： 在 main 中负责初始化停车场的栈

InitQueue(Access)

描述： 初始化队列

用途： 在 main 函数中负责初始化便道的队列

StackTraverse(Parking)

描述：显示栈内的元素

用途：显示停车场中的汽车停放状况

QueueTraverse(Access)

描述：显示队列内的元素

用途：显示便道中的汽车停放状况

StackEmpty(Parking)

描述：检查栈是否为空

用途：检查停车场是否已经无车

QueueEmpty(Access)

描述：检查队列是否为空

用途：检查便道是否已经无车

Push(Parking)

描述：将一个元素入栈

用途：main 中如果车库未满，新来的汽车将进入停车场

Enqueue(Access)

描述：将一个元素入队列

用途：main 中如果车库已满，新来的车将进入便道

Pop(Parking,e)

描述：将一个元素出栈

用途：一辆汽车离开停车场

DeQueue(Access,e)

描述：将一个元素出队列

用途：停车场中有空位时，将便道中的车放进停车厂里

DestroyStack(Parking)

描述：销毁栈，并释放内存

用途：在 main 结束时，释放栈中的内存

DestroyQueue(Access,)

描述：销毁队列，并释放内存

用途：在 main 结束时，释放队列中的内存

完整代码

```
 #include <bits/stdc++.h>

#include "CircularQueue_SqList.cpp"
// #include "CircularQueue_stl.cpp"
#include "Stack_Sqlist.cpp"
// #include "Stack_stl.cpp"

using namespace std;

constexpr int PARKING_SIZE = 5; // 停车场大小
constexpr int ACCESS_SIZE = 5;   // 便道大小

char Menu()
{
    char choice;
    for (int i = 0; i <= 30; i++)
        cout << "#\n"[i == 30];
    cout << "#            停车场管理系统            #\n";
    cout << "#          进入停车场(输入 A)        #\n";
    cout << "#          离开停车场(输入 D)        #\n";
    // cout << "#       显示停车场状态(输入 S)     #\n";
    // cout << "#        显示便道状态(输入 Q)       #\n";
    cout << "#            退出系统(输入 0)          #\n";
    for (int i = 0; i <= 30; i++)
        cout << "#\n"[i == 30];
    cout << "请输入指令：";
    cin >> choice;
    return choice;
}

int main()
{
    Stack Parking; // 停车场
    Queue Access;   // 便道
    InitStack(Parking);
    InitQueue(Access);
    char choice, cnt = 0;
    choice = Menu();
    while (choice != '0')
    {
        if (choice == 'A')
        {
            if (cnt > PARKING_SIZE + ACCESS_SIZE)
```

```
        {
            cout << "车位已满，无法进入！" << endl;
        }
        else
        {
            cout << "请输入车号：";
            int car;
            cin >> car;
            if (cnt < PARKING_SIZE)
            {
                Push(Parking, car);
                cnt++;
            }
            else
            {
                EnQueue(Access, car);
                cnt++;
            }
        }
        StackTraverse(Parking);
        QueueTraverse(Access);
    }
    else if (choice == 'D')
    {
        if (cnt == 0)
        {
            cout << "停车场为空！" << endl;
        }
        else
        {
            cout << "请输入车号：";
            int car;
            cin >> car;
            Stack TempParking;
            InitStack(TempParking);
            bool isFound = false;
            while (!StackEmpty(Parking))
            {
                int e;
                Pop(Parking, e);
                if (e == car)
                {
                    isFound = true;
                    break;
```

```
                    }
                    Push(TempParking, e);
                }
                if (isFound)
                {
                    while (!StackEmpty(TempParking))
                    {
                        int e;
                        Pop(TempParking, e);
                        if (e != car)
                            Push(Parking, e);
                    }
                    cnt--;
                    DestroyStack(TempParking);
                    if (!QueueEmpty(Access))
                    {
                        int e;
                        DeQueue(Access, e);
                        Push(Parking, e);
                        cnt++;
                    }
                }
                else
                {
                    cout << "停车场内没有该车辆！" << endl;
                    while (!StackEmpty(TempParking))
                    {
                        int e;
                        Pop(TempParking, e);
                        Push(Parking, e);
                    }
                    DestroyStack(TempParking);
                }
            }
            StackTraverse(Parking);
            QueueTraverse(Access);
        }
        choice = Menu();
    }
    DestroyStack(Parking);
    DestroyQueue(Access);
    return 0;
}
```

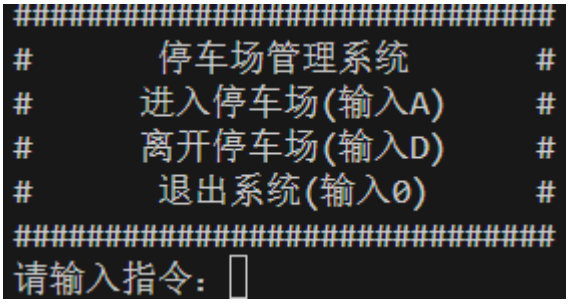
六、实验调试、测试样例与结果分析（问题的发现、分析、解决方案与创新）

测试用例

1. 车辆 1、2、3、4、5、6 依次进入
2. 车辆 3 驶出
3. 结束程序

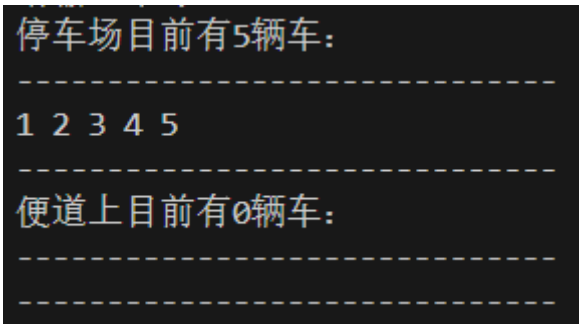
测试结果

初始界面



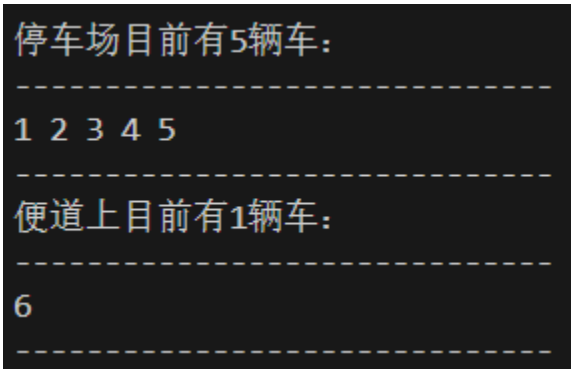
让车辆 1、2、3、4、5 进入

停车场已满，便道为空



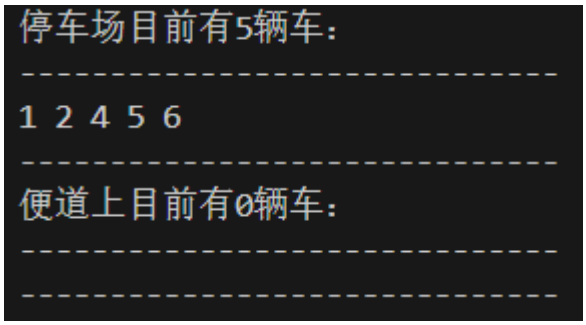
让车辆 6 进入

车辆 6 停在便道



车辆 3 驶出

车辆 6 进入了停车场



七、小组成员任务分配

李坤霖：栈的基本操作的实现与测试

姜高峰：队的基本操作的实现与测试

刘镇东：其余部分代码以及整体代码的合并

八、实验改进意见与建议

1.采用面向对象编程加强规范性

2.考虑利用数据库实现数据的持久化存储

3.假如错误信息提示

4.提供图形化交互界面

5.对程序进行完整测试包括单元测试及集成测试并记录测试样例

6.使停车场与便道的大小可动态调整

7.完善注释

九、附录与说明

无