# 停车场管理模拟程序设计报告

## 项目简介

本项目旨在模拟有限容量停车场的管理过程。停车场为狭长通道，仅有一个大门供车辆进出。车辆按照到达时间先后顺序依次停放，最先到达的车辆停在停车场最北端（距离出口最远）。超过容量限制的车辆在门外便道上等待。一旦有车辆离开停车场，需要将其后面的车辆依次移出让路，待该车辆离开后，其他车辆按原序入场；若便道中有车辆等待，则第一个等待的车辆进入停车场。此外，车辆需按停留时间支付费用。本程序按照上述规则，通过栈和队列实现停车场管理逻辑，并输出详细的停车操作日志。

## 设计思路

停车场管理的核心基于以下操作逻辑：

1. **车辆入场**：
   * 检查停车场是否已满：
     + 若未满，车辆进入停车场，记录入场时间；
     + 若已满，车辆进入门外便道，等待入场。
2. **车辆离场**：
   * 将在目标车辆后方的车辆移动至辅助栈以腾出通道；
   * 目标车辆离开停车场，根据停留时间和收费比率计算费用并输出相应日志；
   * 原先在辅助栈中的车辆依次回到停车场；
   * 若便道上有等待车辆，首位车辆进入停车场并记录入场时间。
3. **便道管理**：
   * 使用队列实现先进先出（FIFO）的逻辑，确保便道车辆按原顺序入场。

为了实现上述流程，该停车场管理模拟程序通过使用各类C++ 特性实现了高度抽象化和灵活性。

## 数据结构

停车场管理程序采用了以下数据结构以满足场景需求：

1. **停车场：栈**
   * 使用栈来存储停车场内的车辆，符合 “后进先出” 的特点。
   * 当目标车辆离场时，需要挪动目标车辆后方的车辆至辅助栈。
2. **辅助栈**
   * 在目标车辆离场时，所有之后停放的车辆暂时转移至此栈，待目标车辆离场后再依序返回停车场。
3. **便道：队列**
   * 使用队列存储超出停车场容量的车辆，符合 “先进先出” 的特点。
   * 停车场中有空余位置时，便道首位车辆优先入场。
4. **时间记录**
   * 使用 std::map 存储车辆的入场时间，时间戳由 std::chrono::steady\_clock::time\_point 表示。
   * 出场时根据当前时间和入场时间计算停留时间，并基于停车比率计算收费。

此外，本项目引入了自定义的**容量限定双端队列**（LimitedDeque\_），在保证栈行为的同时提供高度可控的容量限制。此结构基于 STL 容器 std::deque 实现，通过覆盖插入操作，确保停车场容量符合设计。

值得一提的是，本项目设计和实现中大量使用了 C++ 的现代特性（其实是想借机会练习一下，设计可能不一定完全合理）：

1. **模板编程与单例模式**：
   * 使用模板参数为停车场设定名称（const char\*）和容量（const int），保证了类的通用性并避免冗余代码，同时匹配现实生活中停车场作为建筑在设计容量等方面的稳定性。
   * 采用单例模式避免多实例冲突，静态 getInstance() 方法通过懒加载确保唯一实例。
2. **标准库时间类型 (std::chrono)**：
   * 使用 std::chrono::steady\_clock 和 std::chrono::system\_clock 实现高精度时间测量与人类可读时间戳打印。
   * 停车时间的单位灵活性（如秒、毫秒等）通过 std::chrono 的内置接口方便实现。
3. **友元函数与运算符重载**：
   * 定义了运算符重载 << 和 >>（分别对应车辆入场和离场逻辑），提供简洁的语法糖，提升代码可读性与简洁性。
4. **现代 STL 容器与流操作**：
   * 使用 std::stack 作为停车场核心数据结构，std::queue 模拟便道行为，std::map 跟踪车辆时间数据。
   * 利用 std::thread 和 this\_thread::sleep\_for 实现即时延迟，模拟现实停车场管理中的物理操作时间。
5. **自定义抽象数据结构：LimitedDeque\_**
   * 基于继承的容量限制双端队列，更为优雅地实现了严格控制停车场的最大容量并提供溢出检查。
6. **日志工具**：
   * 输出日志时，支持多信息展示（停车场状态、时间戳、操作详情等），提高调试和用户体验。

**程序运行流程**

1. **启动与停车场信息初始化**：
   * 通过模板实例化停车场单例，设定停车场名称、容量与收费比率。
2. **车辆入场测试**：
   * 新车辆尝试进入停车场，受限时进入便道等待，所有操作打印详细日志。
3. **车辆离场测试**：
   * 按指定逻辑模拟让路和收费过程，计算离场车辆停留时间与费用，最终恢复停车场的完整性。

通过这样的逻辑，程序可动态管理车辆进出与收费操作，满足题目要求的完整功能。

## 运行结果

程序在windows11上使用CLion 2024.3.2 编写，使用CMake3.30（最低设置为3.8）构建系统+ninja链接器，编译器为mingw64\_x86\_64-14.2.0-release-posix-seh-ucrt-rt\_v12-rev1，使用特性至少涉及到C++17。提交前已在上述条件下跑通程序。

按照main函数中的测试用例，结果应当类似于：  
- [SDUPark#01] 停车场名称: SDUPark#01, 设计容量:4, 收费比率(元每秒): 1

- [SDUPark#01] 已修改收费比率: 0.01

- [SDUPark#01(1/4): 03-02 15:39:58] car#100001 从外界进入停车场

- [SDUPark#01(1/4): 03-02 15:39:59] 在停车场寻找 car#1

- [SDUPark#01(0/4): 03-02 15:39:59] car#100001 从停车场暂离至中转区

- [SDUPark#01(0/4): 03-02 15:39:59] 停车场内未找到 car#1

- [SDUPark#01(1/4): 03-02 15:39:59] car#100001 从中转区回到停车场

- [SDUPark#01(2/4): 03-02 15:39:59] car#100002 从外界进入停车场

- [SDUPark#01(3/4): 03-02 15:39:59] car#100003 从外界进入停车场

- [SDUPark#01(4/4): 03-02 15:40:00] car#100004 从外界进入停车场

- [SDUPark#01(4/4): 03-02 15:40:00] 停车场已满, car#100005 进入过道等候

- [SDUPark#01(4/4): 03-02 15:40:00] 停车场已满, car#100006 进入过道等候

- [SDUPark#01(4/4): 03-02 15:40:00] 停车场已满, car#100007 进入过道等候

- [SDUPark#01(4/4): 03-02 15:40:00] 在停车场寻找 car#100001

- [SDUPark#01(3/4): 03-02 15:40:01] car#100004 从停车场暂离至中转区

- [SDUPark#01(2/4): 03-02 15:40:01] car#100003 从停车场暂离至中转区

- [SDUPark#01(1/4): 03-02 15:40:01] car#100002 从停车场暂离至中转区

- [SDUPark#01(1/4): 03-02 15:40:01] car#100001 离开停车场，停车时间: 2.228732秒, 收费: 0.022287元

- [SDUPark#01(1/4): 03-02 15:40:01] car#100002 从中转区回到停车场

- [SDUPark#01(2/4): 03-02 15:40:01] car#100003 从中转区回到停车场

- [SDUPark#01(3/4): 03-02 15:40:01] car#100004 从中转区回到停车场

- [SDUPark#01(4/4): 03-02 15:40:01] car#100005 从过道进入停车场

- [SDUPark#01(4/4): 03-02 15:40:02] 在停车场寻找 car#100002

- [SDUPark#01(3/4): 03-02 15:40:02] car#100005 从停车场暂离至中转区

- [SDUPark#01(2/4): 03-02 15:40:02] car#100004 从停车场暂离至中转区

- [SDUPark#01(1/4): 03-02 15:40:02] car#100003 从停车场暂离至中转区

- [SDUPark#01(1/4): 03-02 15:40:02] car#100002 离开停车场，停车时间: 2.779189秒, 收费: 0.027792元

- [SDUPark#01(1/4): 03-02 15:40:02] car#100003 从中转区回到停车场

- [SDUPark#01(2/4): 03-02 15:40:02] car#100004 从中转区回到停车场

- [SDUPark#01(3/4): 03-02 15:40:03] car#100005 从中转区回到停车场

- [SDUPark#01(4/4): 03-02 15:40:03] car#100006 从过道进入停车场

- [SDUPark#01(4/4): 03-02 15:40:03] 在停车场寻找 car#100003

- [SDUPark#01(3/4): 03-02 15:40:03] car#100006 从停车场暂离至中转区

- [SDUPark#01(2/4): 03-02 15:40:03] car#100005 从停车场暂离至中转区

- [SDUPark#01(1/4): 03-02 15:40:03] car#100004 从停车场暂离至中转区

- [SDUPark#01(1/4): 03-02 15:40:03] car#100003 离开停车场，停车时间: 3.778777秒, 收费: 0.037788元

- [SDUPark#01(1/4): 03-02 15:40:04] car#100004 从中转区回到停车场

- [SDUPark#01(2/4): 03-02 15:40:04] car#100005 从中转区回到停车场

- [SDUPark#01(3/4): 03-02 15:40:04] car#100006 从中转区回到停车场

- [SDUPark#01(4/4): 03-02 15:40:04] car#100007 从过道进入停车场

- [SDUPark#01(4/4): 03-02 15:40:04] 在停车场寻找 car#100004

- [SDUPark#01(3/4): 03-02 15:40:04] car#100007 从停车场暂离至中转区

- [SDUPark#01(2/4): 03-02 15:40:04] car#100006 从停车场暂离至中转区

- [SDUPark#01(1/4): 03-02 15:40:04] car#100005 从停车场暂离至中转区

- [SDUPark#01(1/4): 03-02 15:40:04] car#100004 离开停车场，停车时间: 4.794984秒, 收费: 0.047950元

- [SDUPark#01(1/4): 03-02 15:40:05] car#100005 从中转区回到停车场

- [SDUPark#01(2/4): 03-02 15:40:05] car#100006 从中转区回到停车场

- [SDUPark#01(3/4): 03-02 15:40:05] car#100007 从中转区回到停车场

## 总结

本项目利用 C++ 的模板机制和标准库特性实现了一个功能完备、高效灵活的停车场管理程序，充分体现了 C++ 语言在多样性与复杂性处理上的优势，如模板编程、现代 STL 容器、时间处理接口以及友元函数等。当前实现不仅满足了题目要求，还提供了扩展性强、功能抽象明确的基础设计。