|  |  |
| --- | --- |
| 姓名 | 单禹嘉 |
| 学号 | 2023215177 |

|  |  |
| --- | --- |
| 实验成绩 |  |

华中师范大学计算机学院

实验报告书

课程名称：数据结构

主讲教师：沈显君

课程编号：

班级：2306

1. 实验目的

目的：

通过利用线性链表数据结构，模拟并实现两个整数集合的交集、并集和差集运算。通过实验，掌握线性链表在集合运算中的基本应用。实验基本内容

1. 实验要求

1）正确实现栈和队列；

2）完整实现栈和队列的基本操作

1. 实验环境

VSCode

1. 系统描述

• 数据结构设计：

1. 停车场（栈）： 存放已进入停车场的车辆，容量有限。

2. 便道（队列）： 存放等待进入停车场的车辆，容量无限。

• 类设计：

• Car：表示车辆，记录车牌号和进入停车场的时间。

• Stack：实现停车场的栈结构，支持车辆进出。

• Queue：实现便道的队列结构，支持车辆排队等待需求分析

1. 集合表示：

• 使用线性链表表示集合，链表中的每个节点存储一个整数。

2. 基本运算：

• 并集运算： 返回两个集合的并集，包含所有不重复的元素。

• 交集运算： 返回两个集合的交集，包含在两个集合中都存在的元素。

• 差集运算： 返回集合 A 相对集合 B 的差集，即只存在于集合 A 而不在集合 B 中的元素。

1. 附录（源代码）与说明

#include <iostream>

#include <stack>

#include <queue>

#include <string>

#include <iomanip>

using namespace std;

struct Car {

string license; // 车牌号

int arrivalTime; // 到达时间

};

class ParkingLot {

private:

stack<Car> parking; // 停车场栈

queue<Car> waitingQueue; // 等待队列

int capacity; // 停车场容量

int currentTime; // 当前时间

public:

ParkingLot(int cap) : capacity(cap), currentTime(0) {}

// 汽车到达

void arrive(const string& license) {

currentTime++; // 增加当前时间

Car car = {license, currentTime};

if (parking.size() < capacity) {

parking.push(car);

cout << "✅ 汽车 [" << license << "] 已进入停车场。\n";

} else {

waitingQueue.push(car);

cout << "⚠️ 停车场已满，汽车 [" << license << "] 在便道上等待。\n";

}

}

// 汽车离开

void depart(const string& license) {

stack<Car> tempStack;

bool found = false;

// 找到需要离开的车辆，并把它之后的车辆暂存到临时栈

while (!parking.empty()) {

Car car = parking.top();

parking.pop();

if (car.license == license) {

cout << "🚗 汽车 [" << license << "] 已离开停车场。\n";

found = true;

break;

} else {

tempStack.push(car);

}

}

// 若未找到，说明车辆不在停车场

if (!found) {

cout << "❌ 未找到车牌号为 [" << license << "] 的汽车，它不在停车场内。\n";

}

// 将让路的车辆重新入栈

while (!tempStack.empty()) {

parking.push(tempStack.top());

tempStack.pop();

}

// dengdaiduilie

if (!waitingQueue.empty() && parking.size() < capacity) {

Car waitingCar = waitingQueue.front();

waitingQueue.pop();

parking.push(waitingCar);

cout << "🚗 汽车 [" << waitingCar.license << "] 从便道进入停车场。\n";

}

}

// xianshi

void displayStatus() {

cout << "\n🔷 --- 停车场状态 --- 🔷\n";

stack<Car> tempStack = parking;

if (tempStack.empty()) {

cout << "🚫 停车场为空。\n";

} else {

cout << "🚗 停车场内车辆（从出口到入口）：\n";

cout << setw(12) << "车牌号" << setw(12) << "到达时间\n";

cout << "---------------------------------\n";

while (!tempStack.empty()) {

Car car = tempStack.top();

cout << setw(10) << car.license << setw(12) << car.arrivalTime << endl;

tempStack.pop();

}

}

cout << "\n🔷 --- 便道等待区状态 --- 🔷\n";

if (waitingQueue.empty()) {

cout << "🚫 便道上没有等待的车辆。\n";

} else {

cout << "🚗 便道上的车辆（按到达顺序）：\n";

cout << setw(12) << "车牌号" << setw(12) << "到达时间\n";

cout << "---------------------------------\n";

queue<Car> tempQueue = waitingQueue;

while (!tempQueue.empty()) {

Car car = tempQueue.front();

cout << setw(10) << car.license << setw(12) << car.arrivalTime << endl;

tempQueue.pop();

}

}

cout << "---------------------------------\n";

}

};

int main() {

int capacity;

cout << "🚗 欢迎使用智能停车场系统 🚗\n";

cout << "请输入停车场容量: ";

cin >> capacity;

ParkingLot parkingLot(capacity);

while (true) {

cout << "\n========================\n";

cout << "1️⃣ 汽车到达\n";

cout << "2️⃣ 汽车离开\n";

cout << "3️⃣ 显示停车场状态\n";

cout << "4️⃣ 退出程序\n";

cout << "========================\n";

cout << "请输入您的选择: ";

int choice;

cin >> choice;

if (choice == 1) {

string license;

cout << "请输入车牌号: ";

cin >> license;

parkingLot.arrive(license);

} else if (choice == 2) {

string license;

cout << "请输入车牌号: ";

cin >> license;

parkingLot.depart(license);

} else if (choice == 3) {

parkingLot.displayStatus();

} else if (choice == 4) {

cout << "👋 感谢使用停车场系统，程序已退出。\n";

break;

} else {

cout << "❌ 无效的选择，请重新输入。\n";

}

}

return 0;

}