**《数据结构》实验报告（第3章）**

**学号：\_09020328\_\_\_\_ 姓名 ：\_ 王亮\_\_\_\_**

**实验题号： P168T2\_ 实验日期：\_2021.10.29\_\_\_**

**实验一**

**1．问题描述：**

模拟机场的着陆与起飞管理模式，飞机场有3条跑道，4种着陆等待模式，前两条各分有两种着陆等待模式，第3条跑道不作正常着陆。到达的飞机进入某个等待队列，分配整数id，另一整数time（降落前在队列中停留时间），每一跑道均有一起飞队列，其中飞机也会分配id值，各队列长度保持大致相同。

每一单位时间内，最多允许3架飞机进入着陆队列，以及最多3架飞机进入起飞队列。每条跑道在一个时间单位内，允许一次起飞或降落。第3条仅用于起飞（除非紧急降落），任何一个着陆队列中剩余时间接近0的飞机必须给予高于其他飞机无论着陆还是起飞的优先级。若多于1加飞机出现这样的情况，其他跑道亦被使用。

设计一个算法模拟上述模式，并进行输出管理信息。

**2．算法思想：**

每回合限定使用跑道数为3，若触发紧急事件，可用跑道数-=本回合紧急事件数目，根据可用跑道数使用跑道(3->1,2,3 2->1,2 1->1)调用跑道功能函数（起飞/降落），每轮循环调用时间事件函数更新时间。

**3．功能函数：**

tick():飞机对象计时（剩余时间--，等待时间++）

queueUpdate():移除已经起飞/着陆的飞机

enqueue()：入队

dequeue()：出队

first():获取队首元素

findLand()：检查紧急事件，返回此回合必须被优先占用的跑道数

EmergencyHandler()：处理紧急事件

timeEvent()：计时更新

checkForPriority()：封装

enterWaiting()：入等待队列

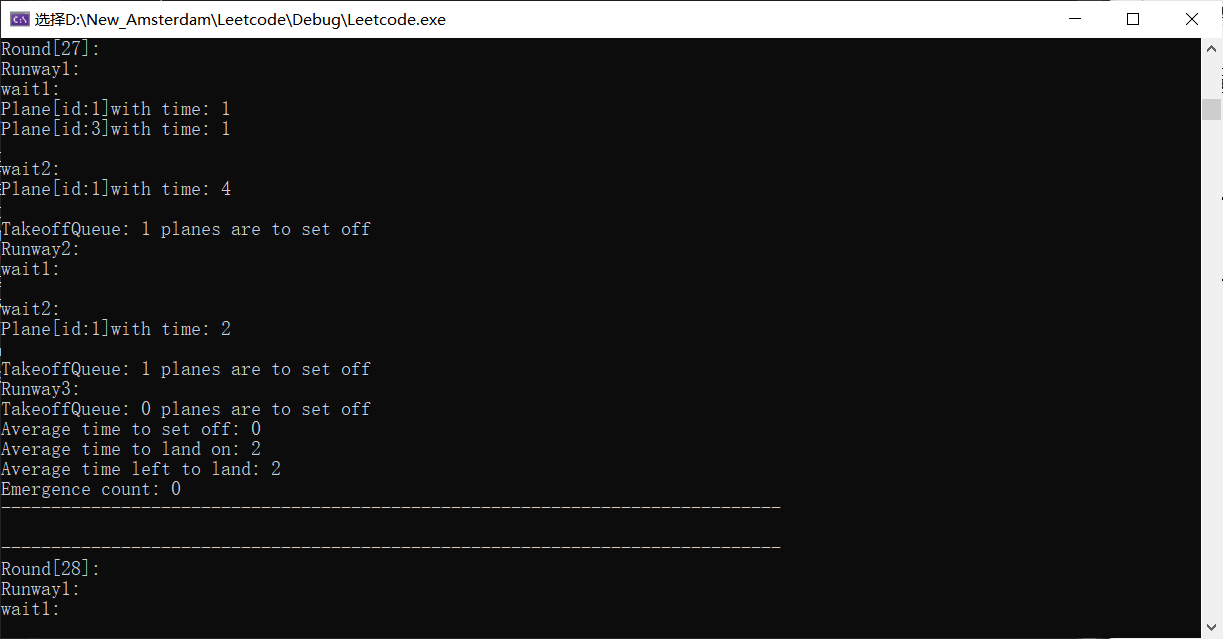
enterTakeoff()：入起飞队列

countWaitSum()：计算等待时间

**4．测试数据：**

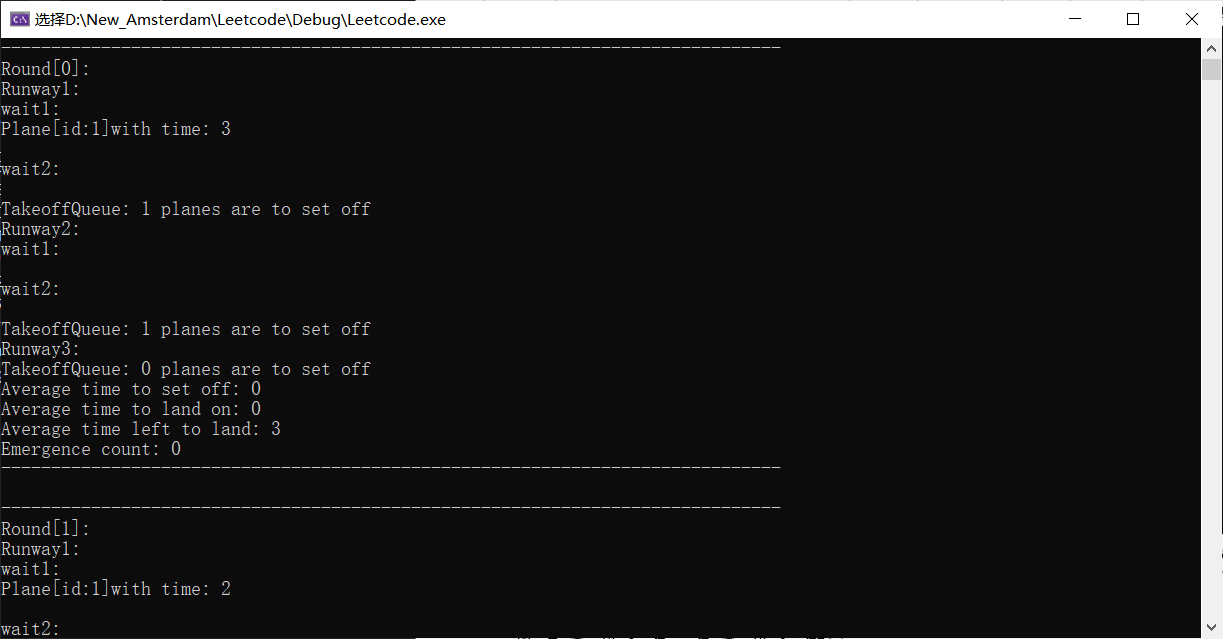
每回合一架飞机进入着陆等待队列（必须降落时间设为随机），两架飞机进入起飞队列（多次测试可以获得所有测试情况）**；**

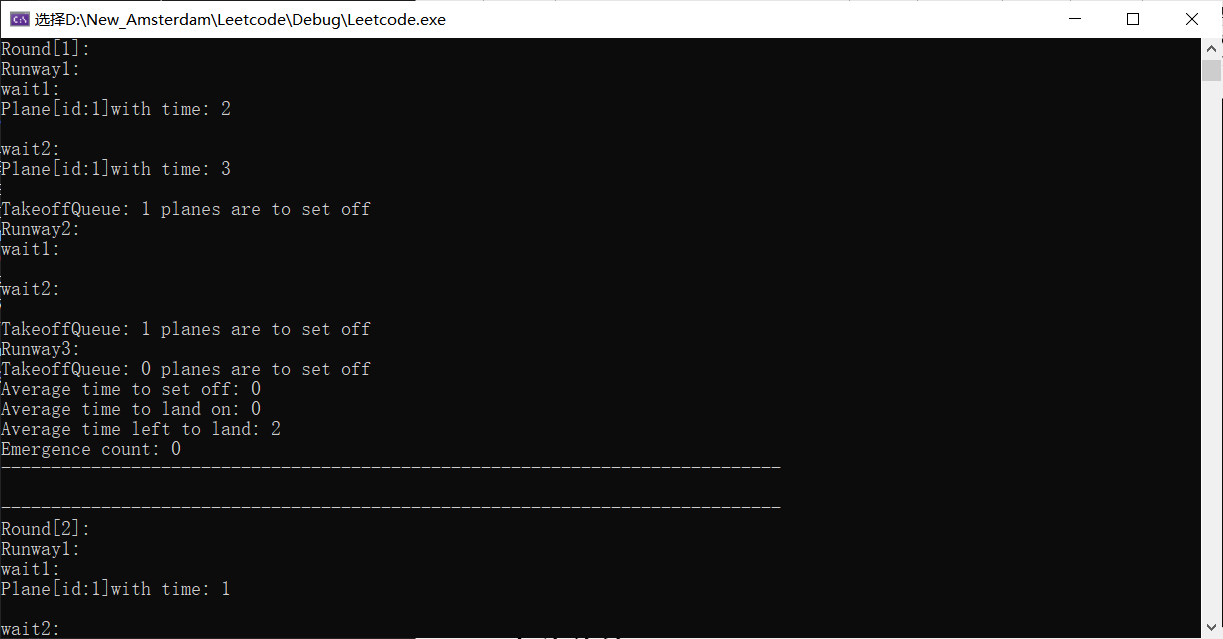
**5．测试情况：**

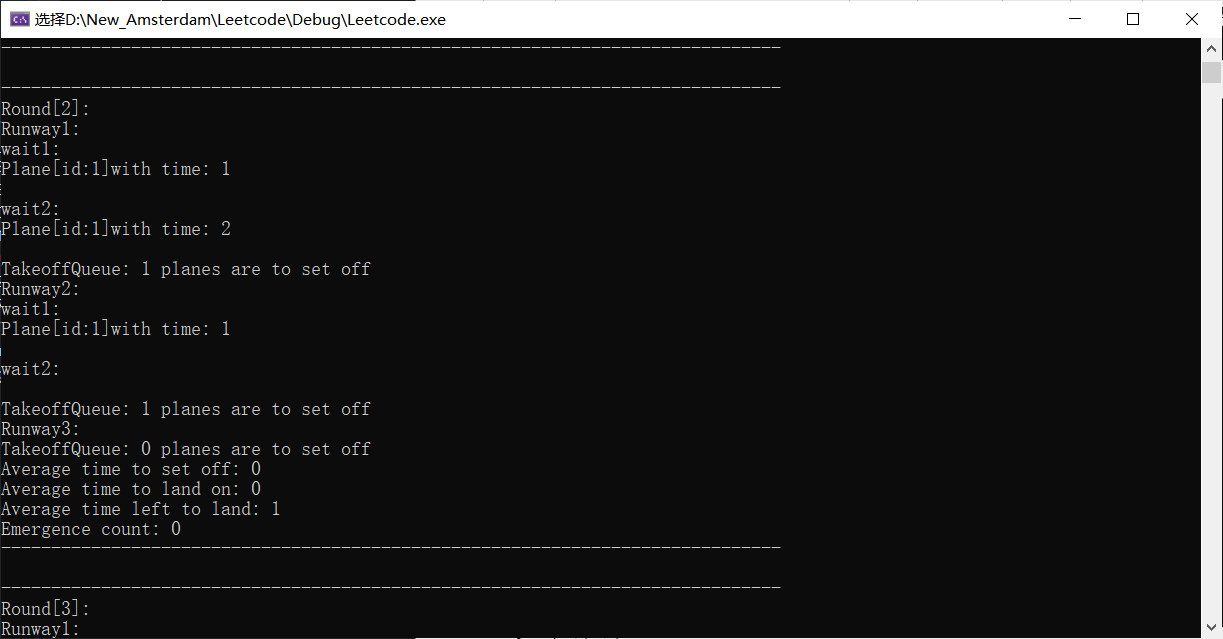




**//Round[27]有两架飞机均为1回合后降落，所以触发一次紧急事件**



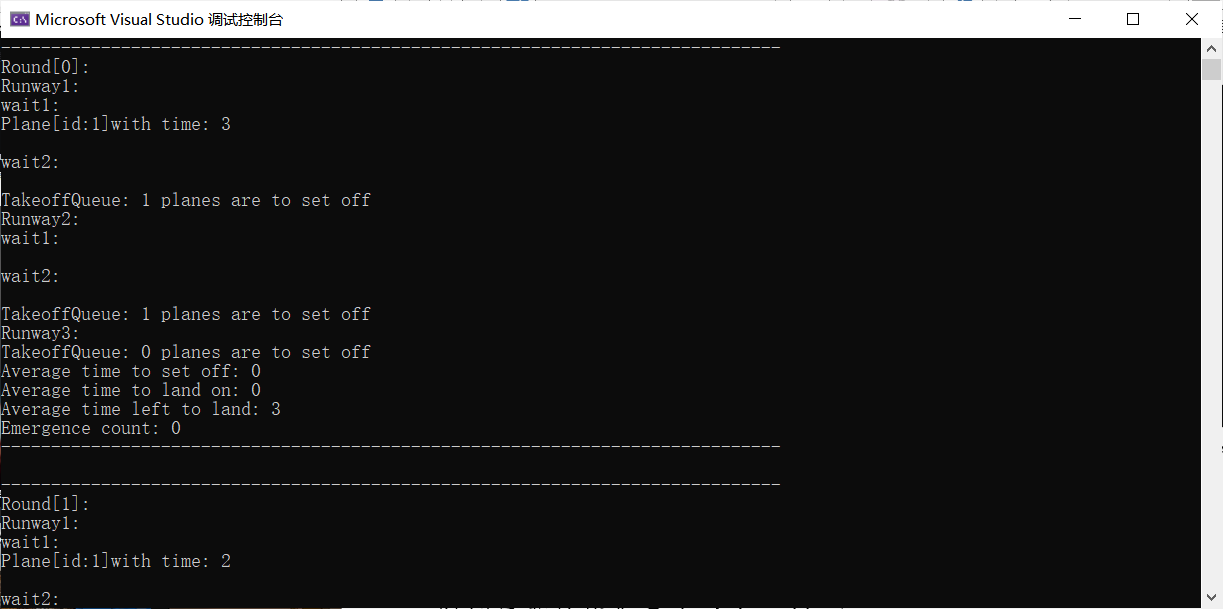


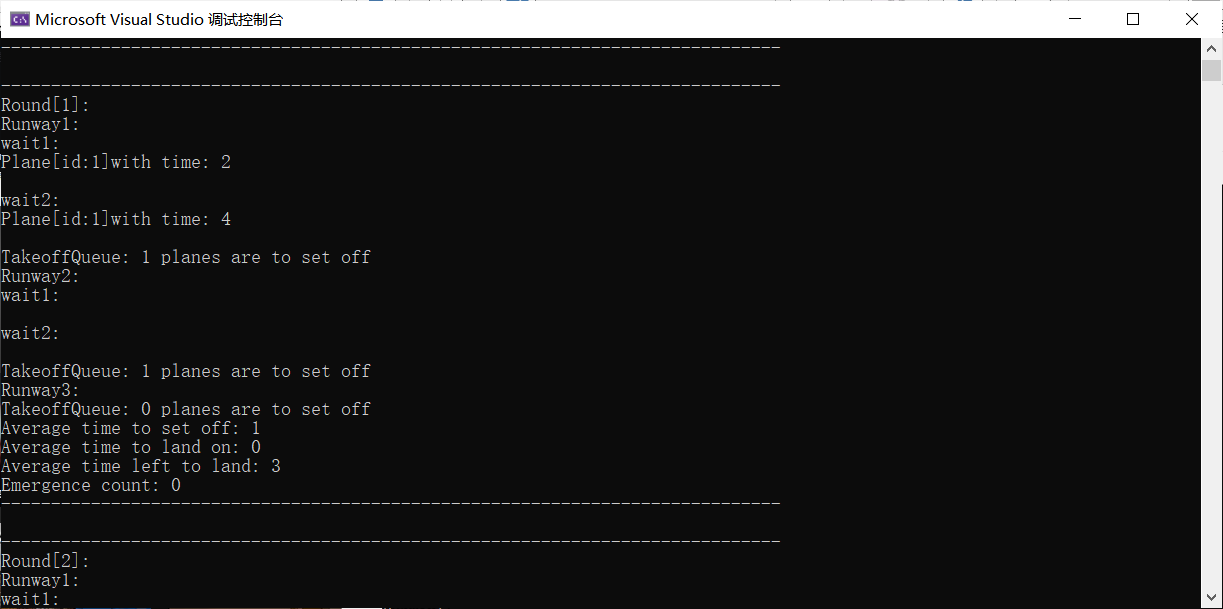




//Round[0-3]中，总计已着陆飞机等待时间为3+1，着陆2架，故为平均着陆时间2

//剩余平均着陆时间：（1+2）/2 = 1（int）





//初始平均起飞时间为0，两架飞机入队后起飞，等待1回合，平均为2/2=1

**6．实验总结：**

有另外一种想法。本次实验中需要频繁的对多个队列中元素进行遍历统计（平均着陆剩余时间），需要查询剩余时间为1的飞机数目（考虑紧急事件时），对于每个等待队列都需要逐一进行更新（时间为0飞机出队，但是由于本实验中存在紧急事件，实际上一定程度“违反”了队列特性，并非“先进先出”，而是存在“优先级”）。

不妨考虑用一个以等待时间为Key一一映射的哈希表，使用拉链法处理同一位置对应的飞机。

其优势如下：

1.这样对于时间为1的飞机查找会从O(n){遍历队列}降低为O(1)；

2.给每架飞机加一个标识位，标记其所在队列(如R1Q1一跑道一等待队列)，每次取时间最短的出队，或者先起飞飞机，等最短时间降至一定范围再作起飞，这会减少紧急事件的发生，每次优先遍历1对应链表，对四个队列（如果有的话）标记的飞机各出一架（增加标识位代表四个队列是否已经降落过飞机（若没有相应标记的飞机记为已降落））。 若遍历完最短时间所在链表，对应跑道已经降落过飞机，那么触发紧急事件；

3.采用上述方式不再需要队列存储，全部放在一张哈希表上，若时间范围有限，且较小，可以减少空间使用；

4.更新不需要遍历四个队列，进行数据迁移，只需更新一张哈希表，遍历一次即可，因为同一时间的都会-1，只需将原先1对于链表清空，然后将剩余key对应的链表依次向前移动一格即可（或看作循环队列，映射为front+时间，这样就不需要移动其他数据，只需要清空当前front链表并进行front后移）。

上述操作对于起飞队列同理。感觉上比较对应到实际生活机场里面的航班出发表。

时间复杂度:除了入队O(1)，其他均为O(n)（操作均需遍历队列，没用堆）

空间复杂度:O(1)（操作不需要修改原队列，除去队列扩容）

**7. 源代码：**

实验源程序清单

(1) As-Lab333.cpp