实验报告

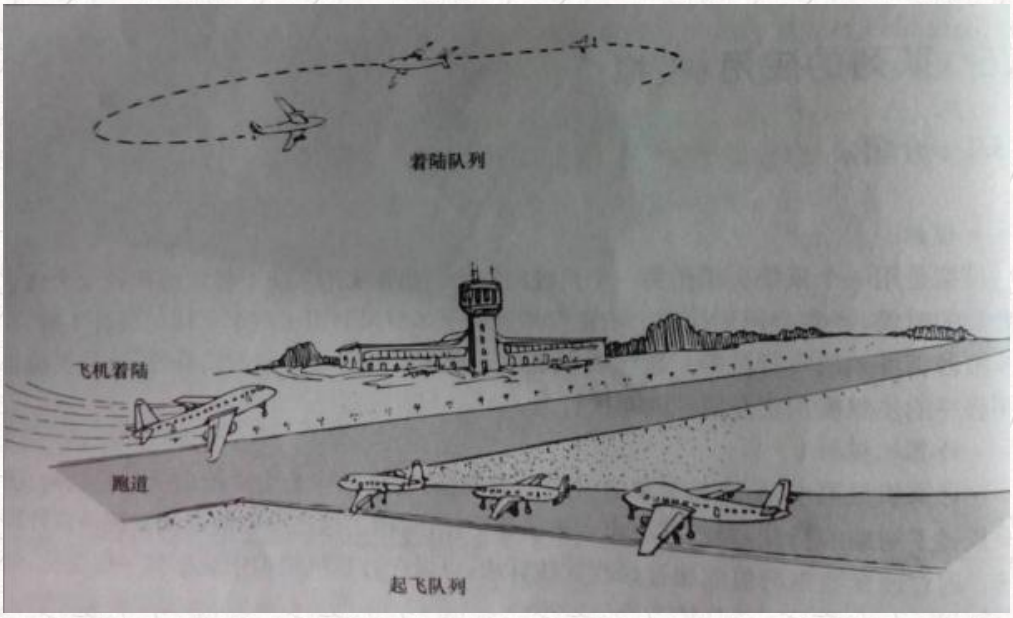
-project2

组长：王迎旭（16340226）
组员：王友坤（16340228）汤万鹏（16340205）
班级：16 级软件工程教务 3 班
日期：2017.10.17

目录

第一部分

问题描述



P1. 将所有用于飞机场模拟的函数和方法组合成一个完整的程序。用飞机场模拟程序做若干次试运行实验，调整准备着陆和起飞的飞机数的期望值，并找出在飞机不会被拒绝服务的条件下这些数字的尽可能大的近似值。如果队列的长度增加或减少，那么这些值将会有什么变化？

P2. 修改模拟程序，使飞机场有两条飞机跑道，其中一条总是用于着陆，另一条总是用于起飞。比较双跑道机场能服务的总飞机数和单条飞机跑道的飞机场的相应数字，前者是否是后者的两倍？

P3. 修改模拟程序，使飞机场有两条飞机跑道，其中一条总是用于着陆，另一条总是用于起飞。如果某个队列是空的，那么两条跑道都能用于其他的队列。如果着陆队列总是满的并且另一架飞机要到达着陆，那么将停止起飞，并将两条跑道都用于清理搁置的着陆飞机。

P4. 修改模拟程序，使飞机场有 3 条飞机跑道，其中各保留一条总用于着陆和起飞，第三条用于着陆，但在着陆队列为空的情况下，第三条亦可用于起飞。

P5. 修改最初的模拟程序（单条跑道），使得当每架飞机到达着陆时，它将有（作为它的数据成员的）一个（随机产生的）油位，以剩余的时间单元度量。如果飞机没有足够的油位在队列中等待，则允许它立即着陆。因此着陆队列里的飞机可能需要再等待附加的单元，因此可能用完自身的燃料。作为着陆函数的一部分检查这一点，并且查明在飞机由于燃料用尽而开始坠毁前机场有多忙。

P6. 写一个占位程序来代替随机数函数，这个占位程序既能用于调试程序又允许用户正确地控制每一个时间单元内每个队列到达的飞机数。

第二部分

编译细节的介绍

2.1 运行环境

Windows10 ，Fedora 虚拟机

2.2 编程语言

c++

2.3 开发工具

（1）Windows 环境下的 g++编译器与 Sublime Text3 编辑器

（2）Dev c++（注:由于本地的编写实在 Sublime 中完成，所以在 c++中编译时候会产生注释乱码情况）

2.4 编码的规范

在 Sub 中编辑时候，已经严格按照首行缩进的方式进行编译。但美中不足，在代码进行复制粘贴时候有可能导致代码错位的问题。

第三部分

分析与设计

3.1 数据结构

队列：用于模拟起飞队列和降落队列。一般情况下先进入队列的飞机将会先离开队列完成起飞和降落的操作。每次有飞机请求降落或者请求起飞时，让飞机进入相应的队列

3.2 设计思路

前期考虑：

P1 要求使用书本上提供的代码，于是考虑以 P1 的代码为基础，通过改动实现的细节，而不改动代码整体结构，来完成 P2 到 P6。经过对题目的讨论，我们认为 P2~P4 可以共用一种修改，P5 使用另一种修改，P6 可以独立并融合于 P1~P5。下面将 P1 的结果称作源代码。

（首先应该考虑 P6）考虑到实际情况， 这个部分独立性强，可以先予以实现，方便在完成其他问题时应用于调试。

（P2~P4 可以一起考虑）考虑到实际情况，应该设计成机场拥有起飞降落的两条队列，但可以拥有多条跑道。不妨认为跑道间共用这两条队列。结果是将两条队列作为跑道类的共享成员。也考虑给跑道实例新增两个状态：能用来进行什么活动，闲置能否利用起来，方便完成问题。

（P5 可以作为另一个独立的分支考虑）考虑到实际情况，飞机实例应该新增燃油剩余时间的状态。

整体的思路介绍：

1. 由用户输入运行时间单位数、队列最大长度、单位时间请求降落率、单位时间请求起飞率

2. 生成请求降落的飞机数，并将这些飞机加入降落队列。若一飞机尝试加入时降落队列已满，则拒绝该飞机的降落请求；否则让该飞机加入降落队列等待
3. 生成请求起飞的飞机数，同步骤 2 将这些飞机加入起飞队列
4. 根据两个队列的情况，决定现在跑道是用于哪架飞机的降落或起飞。
    - 若降落队列不为空，则选择降落队列队头的飞机，让它出队降落；
    - 否则，若起飞队列不为空，则选择起飞队列队头的飞机，让它出队起飞；
    - 否则，跑道闲置
5. 进入下一个时间单位。若已经过去的时间单位数还没到达用户指定要运行的时间单位数，则回到步骤 2，继续运行。否则，展示统计数据并结束程序

单步的思路介绍：

P6： 在总体流程的第 2、第 3 步中，将生成这两个数的过程从随机产生换成由用户手动输入。若输入遇到 EOF，即用户不想再输入了，则切换回随机生成。

P2： 总体流程的第 4 步改为：

（1）对降落跑道，若降落队列不为空，则选择降落队列队头的飞机，让它出队降落；否则，跑道闲置。

（2）对起飞跑道，若起飞队列不为空，则选择起飞队列队头的飞机，让它出队起飞；否则，跑道闲置。

P3：

（1）总体流程的第 2 步最后加上一步，若降落队列已满，则起飞跑道设置为只用于降落，直到降落队列里只有 1 架或没有飞机

（2）总体流程的第 4 步改为：

Ⅰ、对降落跑道

①若降落队列不为空，则选择降落队列队头的飞机，让它出队降落；

②否则，若起飞跑道不为空，则选择起飞队列队头的飞机，让它出队起飞；

③否则，跑道闲置

Ⅱ、对起飞跑道（若起飞跑道只用于降落）

①若降落队列不为空，则选择降落队列队头的飞机，让它出队降落；

②否则，跑道闲置

Ⅲ、对起飞跑道（若起飞跑道不只用于降落）

①若起飞队列不为空，则选择起飞队列队头的飞机，让其出队起飞；

②否则，若降落队列不为空，则选择降落队列队头的飞机，让它出队降落；

③否则，跑道闲置

P4:

总体流程的第 4 步改为：

Ⅰ、对于降落跑道

①若降落队列不为空，则选择降落队列对头的飞机，让它出队降落；

②否则，跑道闲置

Ⅱ、对于起飞跑道

①若起飞队列不为空，则选择起飞队列队头的飞机，让它出队起飞；

②否则，跑道闲置

III、对第三跑道

①若降落队列不为空，则选择降落队列队头的飞机，让它出队降落；

②否则，若起飞队列不为空，则选择起飞队列队头的飞机，让它出队起飞；

③否则，跑道闲置

P5：

（1）（总体流程的第 2 步）

飞机在请求降落时增加判断该飞机所带的燃油是否足够，不够的话就拒绝。每次往降落队列添加飞机时，通过燃油剩余数，判断并标记将来最可能需要迫降的飞机，由它占领跑道

（2）（总体流程的第 2 步）

Ⅰ、增加请求紧急迫降飞机数，若有跑道可用，则同意让它紧急迫降；

Ⅱ、否则，飞机惨烈坠毁。若紧急迫降失败，飞机也抱憾坠毁

（3）（总体流程的第 4 步）

Ⅰ、若降落队列不为空，则选择将来最可能需要迫降的飞机，让它出队降落；

Ⅱ、否则，若起飞队列不为空，则选择起飞队列队头的飞机，让它出队起飞；

Ⅲ、否则，跑道闲置

3.3 设计要点

（1）数据结构的申请与构建

（2）各种出队入队条件的设计

（3）操作过程的显示

（4）前端界面的设计

（5）内存空间的检测

（6）非法数据的处理

第四部分

关键代码展示

```
#ifndef Runway_hpp
#define Runway_hpp
#include "Plane.hpp"
#include "Extended_Queue.hpp"
enum Runway_activity {idle, land, takeoff1};
class Runway{
public:
    Runway(int limit);
    Error_code can_land(const Plane &current);
    Error_code can_depart(const Plane &current);
    Runway_activity activity(int time, Plane &moving);
    void shut_down(int time) const;

private:
    Extended_Queue landing;
    Extended_Queue takeoff;
    int queue_limit;
    int num_land_requests; //number of planes asking to land
    int num_takeoff_requests; //number of planes asking to take off
    int num_landings; //number of planes that have landed
    int num_takeoffs; //number of planes that have taken off
    int num_land_accepted; //number of planes queued to land
    int num_takeoff_accepted; //number of planes queued to take off
    int num_land_refused; //number of landing planes refused
    int num_takeoff_refused; //number of departing planes refused
    int land_wait; //total time of planes waiting to land
    int takeoff_wait; //total time of planes waiting to take off
    int idle_time; //total time runway is idle
};

#endif /* Runway_hpp */
```

```
#ifndef Plane_hpp
#define Plane_hpp

enum Plane_status {null, arriving, departing,emergency};
class Plane{
public:
    Plane();

    Plane(int flt, int time, Plane_status status);
    void refuse() const;
    void land(int time) const;
    void fly(int time) const;
    int started() const;
    Plane_status get_status()const;

private:
    int flt_num;
    int clock_start;
    Plane_status state;
    int fuel;
};

#endif /* Plane_hpp */
```

第五部分

实验测试数据截图

（存放在另外一个数据截图 word）

第六部分

项目总结

一.本次项目通过对飞机场的飞机进行调度起飞降落这一实际问题的分析与设计，加深了小组成员对队列这个数据结构的认识与了解：队列是限定仅在表头进行插入和然后在表头进行删除操作的一个线性表。它按照先进先出的原则存储数据，先进入的数据被存入队头，最后的数据在队尾，需要读数据的时候从队头开始弹出数据（先进入的一个数据被第一个弹出来）。

二.解决实际问题的算法要具有以下的特征才能有效的完成设计要求：

（1）有穷性。算法在执行有限步后必须终止。
（2）确定性。算法的每一个步骤必须有确切的定义。
（3）可行性。满足实际问题的需求。

三.在算法设计的整个过程中，小组成员一共提出了两个模型。为排除极端数据对算法的影响，小组成员不断测试程序并修复算法中的不足之处。为适应实际问题的操作，小组成员不断测试程序并改进算法。 在进行实践过程中在做 P5 时候，考虑燃油问题。

例如：将飞机燃油的情况加入考虑其实是队列优先级的一种应用。将飞机的燃油情况加入飞机的标志中，在原来的机场调度的情况下加入了飞机优先级的考虑，通过计算判断其是否处于紧急状况，是的话则将队列中应降落的飞机设为当前的紧急飞机，改变其入队顺序。

四.在小组实践过程中，需要发挥小组各成员的积极性和能动性，同时要发扬团队协作精神。这一次小组组员之间在函数的设计上有各自不同的意见，通过讨论之后，得到了较为满意的统一意见，同时也较好的实现了相应的函数功能，这也是一种客观上的合作进步。

五.Pro 的不足反思

（1）算法较为繁琐不精简，在遇到大数据时候电脑容易卡死，如飞机数目超过 1000
（2）细节处理较为不能让人满意，部分输出常常发生错误，如不能全部输出的情况
（3）时间点的把控，在固定的时间内对任务的完成有所拖拉

但总的来说，通过对 pro 的制作，不管是从心理还是从技术上对个人还是一种很大的客观提升。

第七部分

项目分工与自我评分

姓名	负责部分	自我评分	贡献度
王迎旭	实验报告，异常处理，p1,p2	90	32%
汤万鹏	代码设计，算法改进，p4,p6	93	35%
王友坤	代码设计，前端界面设计，p3,p5	91	33%