

**Project2实验报告**

机场模拟



2017-10-15

16340213王继武

16340203谭江华

16350027黄俊

16软工教务3班

目录

[题目要求 2](#_Toc496169481)

[题目分析和解答 2](#_Toc496169482)

[P1解答 2](#_Toc496169483)

[P2解答 4](#_Toc496169484)

[P3解答 5](#_Toc496169485)

[P4解答 6](#_Toc496169486)

[P5解答 6](#_Toc496169487)

[P6解答 8](#_Toc496169488)

[项目心得 10](#_Toc496169489)

[分工和自我评价 10](#_Toc496169490)

[实验不足和补充 10](#_Toc496169491)

[附录1：p1代码文件（未随机模拟） 10](#_Toc496169492)

[附录2：p1代码文件（随机模拟） 23](#_Toc496169493)

[附录3：测试数据表格 26](#_Toc496169494)

# 题目要求

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，其中一条总是用来着陆，另一条总是用来起飞。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，前者是否为后者的两倍？

P3、修改模拟程序，使飞机场有两条飞机跑道，其中一条总是用来着陆，另一条总是用来起飞。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，其中各保留一条总是用于着陆和起飞，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要检查这一点，并查明在飞机由于燃料耗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。

# 题目分析和解答

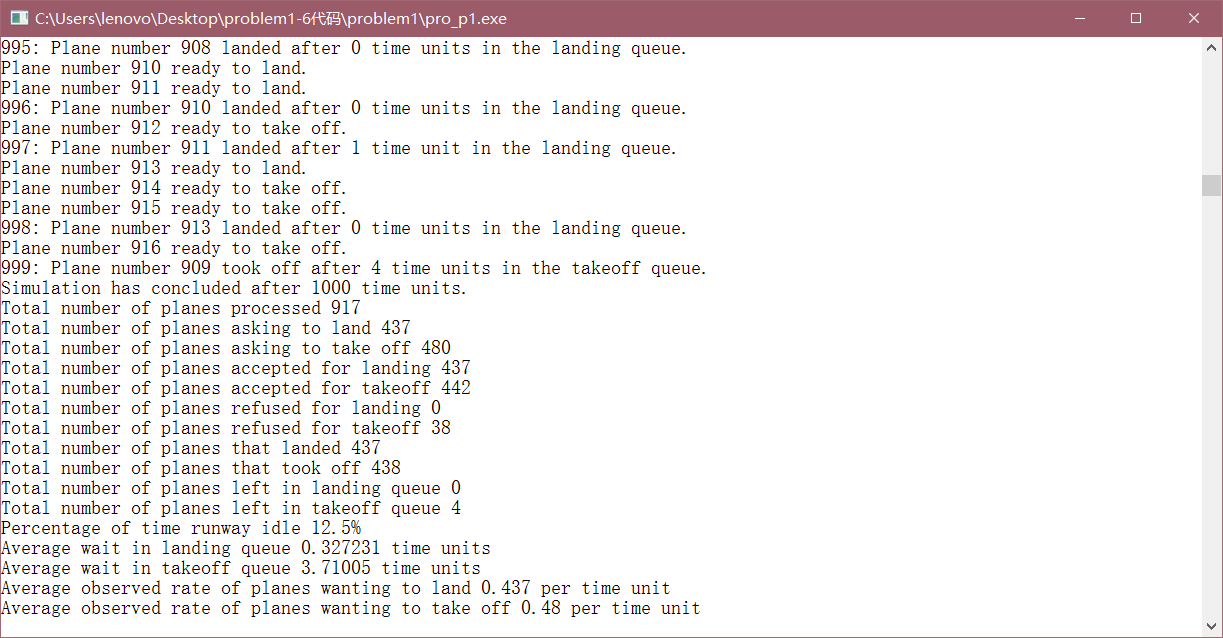
## P1解答

本问主要根据书上所给出的代码，但是要注意的是这里其实用了前面的代码文件比如

**Error\_code，Extended\_queue, Queue,**  所以要补充前面的代码才能运行程序。代码在附录一给出

思路是：建立Runway（跑道类），Plane（飞机类），queue（队列用于储存飞机信息），random（随机类用于给出测试数据），首先在主函数中获取用户希望的着陆率，起飞率，跑道长度以及总时间，然后用循环测试每个时间内飞机是否能起飞，降落等操作，并更新runway对象的信息，直到时间结束，最后输出实验结果。

这里是对p1的测试，和书上给的数据相同，输出结果也相同



**但是模拟次数太少难以找到最大值，所以**

修改main函数用于做大规模模拟，同时把程序中的输出文字全部注释，

这里写了一个初始化函数

void random\_rate\_init(double &ar, double &dr) {

ar = rand() % 1000 / (double)1001;

dr = rand() % 1000 / (double)1001;

}

用于初始化arriving\_rate和departing\_rate，使得每一次模拟的rate都不同，在end\_time = 100,queue\_limit = 5的条件下，模拟1000次，求的arriving\_rate和departing\_rate都最大的值，然后重复100次，得到100组

每1000次模拟中最大的arriving\_rate和departing\_rate。同理对end\_time = 100,queue\_limit = 3的条件和end\_time = 100,queue\_limit = 5的条件下，也各得到100组数据，把这些组数据利用文件输入到txt，导入excel表格中然后处理。

**最终得到了附录2中数据，**

**发现在end\_time = 100,queue\_limit = 3条件下**

**arriving\_rate = 0.507493，departing\_rate = 0.427572最多；**

**发现在end\_time = 100,queue\_limit = 5条件下**

**arriving\_rate = 0.534466，departing\_rate = 0.426573最多；**

**发现在end\_time = 100,queue\_limit = 7条件下**

**arriving\_rate = 0.943057，departing\_rate = 0.157842最多；**

**而且发现当queue\_limit增大时，arriving\_rate和departing\_rate最大值变大**

流程图：



## P2解答

P2仅修改了主函数部分，其他部分不变。

在p2中由于有两个跑道，因此修改主函数建立两跑道对象

Runway arrivals(queue\_limit);

Runway departures(queue\_limit);// set up the two runways.

**若有飞机等待降落，则让飞机降落；如果没有，则等待降落队列为空。在判断又无飞机要起飞，如果有，则让飞机起飞，如果没有，那么等待起飞队列也是空的**

Plane arriving\_plane;

switch (arrivals.activity(current\_time, arriving\_plane)) {

// Let at most one Plane onto the Runway at current\_time.

case land:

arriving\_plane.land(current\_time);

break;

case take\_off:

cout << "WARNING: Unexpected, catastrophic program failur"<< endl;

break;

case idle:

run\_idle("Arrival", current\_time);

}

Plane departing\_plane;

switch (departures.activity(current\_time, departing\_plane)) {

case land:

cout << "WARNING: Unexpected, catastrophic program failur"<< endl;

break;

case take\_off:

departing\_plane.fly(current\_time);

break;

case idle:

run\_idle("Departure", current\_time);

}

## P3解答

P3是在p2的基础上进行修改，

将等待降落的飞机加入队列，**如果起飞轨道的起飞和降落等待队列均为空，则加入起飞轨道的降落等待队列**，否则判断降落跑道的降落等待队列是否已满，不满则加入，满则拒绝

将等待起飞的飞机加入队列，**如果降落轨道的起飞和降落等待队列均为空，则加入降落轨道的起飞等待队列**，否则判断起飞跑道的起飞等待队列是否已满，不满则加入，满则拒绝

**降落跑道如果返回值是land（降落），则让等待降落的飞机降落。如果降落跑道没有飞机着陆，就判断返回值是否为take\_off（起飞），**如果也不是那么跑道空闲中；如果是，则让飞机起飞。

代码体现：

**for (int i = 0; i < number\_arrivals; i++) {**

**Plane current\_plane(flight\_number++, current\_time, arriving);**

**if (departures.departure\_size() == 0 &&**

**departures.arrival\_size() == 0)**

**departures.can\_land(current\_plane);**

**else if (arrivals.can\_land(current\_plane) != success)**

**if (departures.can\_land(current\_plane) != success)**

**current\_plane.refuse();**

**}**

**int number\_departures= variable.poisson(departure\_rate);**

**for (int j = 0; j < number\_departures; j++) {**

**Plane current\_plane(flight\_number++, current\_time, departing);**

**if (arrivals.departure\_size() == 0 &&**

**arrivals.arrival\_size() == 0)**

**arrivals.can\_depart(current\_plane);**

**else if (departures.can\_depart(current\_plane) != success)**

**current\_plane.refuse();**

**}**

## P4解答

即是在P1基础上增加两条跑道，判断方法与P3基本一致。

代码体现：

**Runway arrivals(queue\_limit); // set up the 3 runways.**

**Runway departures(queue\_limit);**

**Runway overflow(queue\_limit);**

**int number\_arrivals = variable.poisson(arrival\_rate);**

**for (int i = 0; i < number\_arrivals; i++) {**

**Plane current\_plane(flight\_number++, current\_time, arriving);**

**if (arrivals.can\_land(current\_plane) != success)**

**if (overflow.can\_land(current\_plane) != success)**

**current\_plane.refuse();**

**}**

**int number\_departures= variable.poisson(departure\_rate);**

**for (int j = 0; j < number\_departures; j++) {**

**Plane current\_plane(flight\_number++, current\_time, departing);**

**if (departures.can\_depart(current\_plane) != success)**

**if (overflow.arrival\_size() == 0)**

**if (overflow.can\_depart(current\_plane) != success)**

**current\_plane.refuse();**

**}**

## P5解答

在Plane.hpp里添加新属性

int oil\_level=rand();

修改main函数 每次降落的时候，**用check\_oil()函数来判断降落队列中有没有oil\_level=0的飞机，是的话降落这一架飞机**，没有就是正常的运行，但是降落要花时间，一个单位时间，所以，**利用queue\_time\_pp（）函数，其余所有在降落队列中的飞机oil\_level-1,然后判断在这个队列中是否还有飞机是oil\_level小于0的飞机**存在，这些飞机是坠毁的，并且记录机场繁忙的情况。

if(!small\_airport.check\_oil())

{

Plane moving\_plane;

switch (small\_airport.activity(current\_time, moving\_plane)) {

// Let at most one Plane onto the Runway at current\_time.

case land:

moving\_plane.land(current\_time);

break;

case take\_off:

moving\_plane.fly(current\_time);

break;

case idle:

run\_idle(current\_time);

}

}

bool Runway::check\_oil()

{

int flag=0,flight\_num;

for(;!landing.empty();)

{

if(landing.retrieve().oil\_level<=0)

{

if(flag==0)

{

flag=1;

flight\_num=landing.retrieve().flt\_num;

landing.serve();

}

else

{

temp.append(landing.retrieve());

landing.serve();

}

}

else

{

temp.append(landing.retrieve());

landing.serve();

}

}

for(;!temp.empty();)

{

landing.append(temp.retrieve());

temp.serve();

}

if(flag==1)

{

cout<<"plane number "<<flight\_num<<" must get landing right now"<<endl;

return 1;

}

return 0;

}

void Runway::queue\_time\_pp()

{

int num\_landing\_queue=landing.count,num\_takeoff\_queue=takeoff.count;

for(;!landing.empty();)

{

if(landing.retrieve().oil\_level<=0)

{

cout<<"plane number"<<landing.retrieve().flt\_num<<" has crushed"<<endl;

cout<<"this landing\_queue has"<<landing.count<<"plane"<<((num\_landing\_queue>1)?" ":"s")<<endl;

cout<<"this takeoff\_queue has"<<takeoff.count<<"plane"<<((num\_takeoff\_queue)?"":"s")<<endl;

crush\_num++;

landing.serve();

}

else

{

landing.retrieve().oil\_level--;

temp.append(landing.retrieve());

landing.serve();

}

}

for(;!temp.empty();)

{

landing.append(temp.retrieve());

temp.serve();

}

## P6解答

沿用p1代码，**不过不使用随机数来得到每个时间的起飞飞机数和着陆飞机数，而是先让用户初始化这些数字，**

所以**修改main中的void initialize(int &end\_time, int &queue\_limit)函数， 不需要传入arrival\_rate和departure\_rate**

**在初始化阶段就初始好数据，这样相当于一个占位程序**

**void initialize(int &end\_time, int &queue\_limit)**

**{**

**cerr << "This program simulates an airport with only one runway." << endl**

**<< "One plane can land or depart in each unit of time." << endl;**

**cerr << "Up to what number of planes can be waiting to land "**

**<< "or take off at any time? " << flush;**

**cin >> queue\_limit;**

**cerr << "How many units of time will the simulation run?" << flush;**

**cin >> end\_time;**

**while (1) {**

**cout<<"key in the circle you want to customaize about landing"<<endl;**

**for(int i=0;i<end\_time;i++) {**

**int n;**

**cin>>n;**

**if (n < 0) {**

**cerr << "right posstive num" << endl;**

**i--;**

**continue;**

**}**

**arriving\_num.push\_back(n);**

**}**

**cout<<"key in the circle you want to customaize about departure"<<endl;**

**for(int i=0;i<end\_time;i++) {**

**int n;**

**cin>>n;**

**if (n < 0) {**

**cerr << "right posstive num" << endl; //判断输入是否为正数**

**i--;**

**continue;**

**}**

**takeoff\_num.push\_back(n);**

**}**

**if (accumulate(arriving\_num.begin(), arriving\_num.end(), 0)/end\_time +**

**accumulate(takeoff\_num.begin(), takeoff\_num.end(), 0)/end\_time > 1) { //判断是否大于1**

**cerr << "Safety Warning: This airport will become saturated. "<< endl;**

**arriving\_num.clear();**

**takeoff\_num.clear();**

**} else {**

**break;**

**}**

**}**

**}**

# 项目心得

王继武：需要考虑的事情太多，我们需要科学的规划，学会利用搜索引擎，查找相关资料，

谭江华：本次项目类过多，结构相对比较复杂，需要时间也更多，感觉这次没做好时间规划，导致有些仓促，还需要进一步学习和熟悉如github等以提高效率。

黄俊：做第5，6问的时候，有些操之过急，而且有很多地方可能有些问题，思考的不够全面，看来还是需要集体讨论才好

# 分工和自我评价

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 分工 | 比例 | 自我评价分值 |
| 谭江华 | 完成p1 和 p2， p3一部分 | 33.3% | 95 |
| 王继武 | 完成p3 和 p4，实验报告 | 33.4% | 95 |
| 黄俊 | 完成p5 和 p6 | 33.3% | 95 |

# 实验不足和补充

P1可以搞一个文件输入输出，p1模拟的数字很少，可以建立数据库。

# 附录1：p1代码文件（未随机模拟）

文件由main.cpp, Runway.cpp, Runway.hpp, Plane.cpp, Plane.hpp, random.cpp random.hpp组成

**main.cpp：**

#include <utility>

#include <iostream>

#include "Runway.hpp"

#include "Plane.hpp"

#include "random.hpp"

using namespace std;

typedef Plane Queue\_entry;

void initialize(int &end\_time, int &queue\_limit,

double &arrival\_rate, double &departure\_rate)

{

cerr << "This program simulates an airport with only one runway." << endl

<< "One plane can land or depart in each unit of time." << endl;

cerr << "Up to what number of planes can be waiting to land "

<< "or take off at any time? " << flush;

cin >> queue\_limit;

cerr << "How many units of time will the simulation run?" << flush;

cin >> end\_time;

bool acceptable;

do {

cerr << "Expected number of arrivals per unit time?" << flush;

cin >> arrival\_rate;

cerr << "Expected number of departures per unit time?" << flush;

cin >> departure\_rate;

if (arrival\_rate < 0.0 || departure\_rate < 0.0)

cerr << "These rates must be nonnegative." << endl;

else

acceptable = true;

if (acceptable && arrival\_rate + departure\_rate > 1.0)

cerr << "Safety Warning: This airport will become saturated. "

<< endl;

} while (!acceptable);

}

/\*

Pre: The user specifies the number of time units in the simulation,

the maximal queue sizes permitted,

and the expected arrival and departure rates for the airport.

Post: The program prints instructions and initializes the parameters

end\_time, queue\_limit, arrival\_rate, and departure\_rate to

the specified values.

Uses: utility function user\_says\_yes

\*/

void run\_idle(int time)

{

cout << time << ": Runway is idle." << endl; //o

}

/\*

Post: The specified time is printed with a message that the runway is idle.

\*/

int main()

{

int end\_time; // time to run simulation

int queue\_limit; // size of Runway queues

int flight\_number = 0;

double arrival\_rate, departure\_rate;

initialize(end\_time, queue\_limit, arrival\_rate, departure\_rate);

Random variable;

Runway small\_airport(queue\_limit);

for (int current\_time = 0; current\_time < end\_time; current\_time++) {

// loop over time intervals

int number\_arrivals = variable.poisson(arrival\_rate);

// current arrival requests

for (int i = 0; i < number\_arrivals; i++) {

Plane current\_plane(flight\_number++, current\_time, arriving);

if (small\_airport.can\_land(current\_plane) != success)

current\_plane.refuse();

}

int number\_departures= variable.poisson(departure\_rate);

// current departure requests

for (int j = 0; j < number\_departures; j++) {

Plane current\_plane(flight\_number++, current\_time, departing);

if (small\_airport.can\_depart(current\_plane) != success)

current\_plane.refuse();

}

Plane moving\_plane;

switch (small\_airport.activity(current\_time, moving\_plane)) {

// Let at most one Plane onto the Runway at current\_time.

case land:

moving\_plane.land(current\_time);

break;

case take\_off:

moving\_plane.fly(current\_time);

break;

case idle:

run\_idle(current\_time);

}

}

small\_airport.shut\_down(end\_time);

}

// Airport simulation program

/\*

Pre: The user must supply the number of time intervals the simulation

is to run, the expected number of planes arriving, the expected

number of planes departing per time interval, and the maximum

allowed size for runway queues.

Post: The program performs a random simulation of the airport, showing

the status of the runway at each time interval, and prints out a

summary of airport operation at the conclusion.

Uses: Classes Runway, Plane, Random and functions run\_idle, initialize.

\*/

**Plane.cpp**

#include "Plane.hpp"

#include <iostream>

using namespace std;

Plane::Plane(int flt, int time, Plane\_status status)

{

flt\_num = flt;

clock\_start = time;

state = status;

cout << "Plane number " << flt << " ready to ";

if (status == arriving)

cout << "land." << endl;

else

cout << "take off." << endl;

}

/\*

Post: The Plane data members flt\_num, clock\_start,

and state are set to the values of the parameters flt,

time and status, respectively.

\*/

Plane::Plane()

{

flt\_num = -1;

clock\_start = -1;

state = null;

}

/\*

Post: The Plane data members flt\_num, clock\_start,

state are set to illegal default values.

\*/

void Plane::refuse() const

{

cout << "Plane number " << flt\_num;

if (state == arriving)

cout << " directed to another airport" << endl;

else

cout << " told to try to takeoff again later" << endl;

}

/\*

Post: Processes a Plane wanting to use Runway, when

the Queue is full.

\*/

void Plane::land(int time) const

{

int wait = time - clock\_start;

cout << time << ": Plane number " << flt\_num << " landed after "

<< wait << " time unit" << ((wait == 1) ? "" : "s")

<< " in the landing queue." << endl;

}

/\*

Post: Processes a Plane that is landing at the specified time.

\*/

void Plane::fly(int time) const

{

int wait = time - clock\_start;

cout << time << ": Plane number " << flt\_num << " took off after "

<< wait << " time unit" << ((wait == 1) ? "" : "s")

<< " in the takeoff queue." << endl;

}

/\*

Post: Process a Plane that is taking off at the specified time.

\*/

int Plane::started() const

{

return clock\_start;

}

/\*

Post: Return the time that the Plane entered the airport system.

\*/

Queue::Queue()

{

count=0;

front=0;

rear=999;

}

bool Queue::empty()const

{

return count==0;

}

Error\_code Queue::append(const Queue\_entry &item)

{

if(count>1000)

return overflow;

count++;

rear=((rear+1)==1000)?0:(rear+1);

entry[rear]=item;

return success;

}

Error\_code Queue::serve()

{

if(count<=0)

return underflow;

count--;

front=((front+1)==1000)?0:(front+1);

return success;

}

Error\_code Queue::retrieve(Queue\_entry &item)const

{

if(count<=0)

return underflow;

item=entry[front];

return success;

}

**Plane.hpp:**

#ifndef PLANE

#define PLANE

enum Plane\_status {null, arriving, departing};

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;

private:

int flt\_num;

int clock\_start;

Plane\_status state;

};

typedef Plane Queue\_entry;

enum Error\_code {success,overflow ,underflow,fail};

class Queue

{

public:

Queue();

bool empty()const;

Error\_code serve();

Error\_code append(const Queue\_entry &item);

Error\_code retrieve(Queue\_entry &item)const;

protected:

int count;

int front,rear;

Queue\_entry entry[1000];

};

class Extended\_queue:public Queue

{

public:

int size()const{

return count;

}

};

#endif

**Runway.cpp:**

#include "Runway.hpp"

#include <iostream>

using namespace std;

Runway::Runway(int limit)

{

queue\_limit = limit;

num\_land\_requests = num\_takeoff\_requests = 0;

num\_landings = num\_takeoffs = 0;

num\_land\_refused = num\_takeoff\_refused = 0;

num\_land\_accepted = num\_takeoff\_accepted = 0;

land\_wait = takeoff\_wait = idle\_time = 0;

}

/\*

Post: The Runway data members are initialized to record no

prior Runway use and to record the limit on queue sizes.

\*/

Error\_code Runway::can\_land(const Plane &current)

{

Error\_code result;

if (landing.size() < queue\_limit)

result = landing.append(current); //

else

result = fail;

num\_land\_requests++;

if (result != success)

num\_land\_refused++;

else

num\_land\_accepted++;

return result;

}

/\*

Post: If possible, the Plane current is added to the

landing Queue; otherwise, an Error\_code of overflow is

returned. The Runway statistics are updated.

Uses: class Extended\_queue.

\*/

Error\_code Runway::can\_depart(const Plane &current)

{

Error\_code result;

if (takeoff.size() < queue\_limit)

result = takeoff.append(current);

else

result = fail;

num\_takeoff\_requests++;

if (result != success)

num\_takeoff\_refused++;

else

num\_takeoff\_accepted++;

return result;

}

/\*

Post: If possible, the Plane current is added to the

takeoff Queue; otherwise, an Error\_code of overflow is

returned. The Runway statistics are updated.

Uses: class Extended\_queue.

\*/

Runway\_activity Runway::activity(int time, Plane &moving)

{

Runway\_activity in\_progress;

if (!landing.empty()) {

landing.retrieve(moving);

land\_wait += time - moving.started();

num\_landings++;

in\_progress = land;

landing.serve();

}

else if (!takeoff.empty()) {

takeoff.retrieve(moving);

takeoff\_wait += time - moving.started();

num\_takeoffs++;

in\_progress = take\_off;

takeoff.serve();

}

else {

idle\_time++;

in\_progress = idle;

}

return in\_progress;

}

/\*

Post: If the landing Queue has entries, its front

Plane is copied to the parameter moving

and a result land is returned. Otherwise,

if the takeoff Queue has entries, its front

Plane is copied to the parameter moving

and a result takeoff is returned. Otherwise,

idle is returned. Runway statistics are updated.

Uses: class Extended\_queue.

\*/

void Runway::shut\_down(int time) const

{

cout << "Simulation has concluded after " << time

<< " time units." << endl

<< "Total number of planes processed "

<< (num\_land\_requests + num\_takeoff\_requests) << endl

<< "Total number of planes asking to land "

<< num\_land\_requests << endl

<< "Total number of planes asking to take off "

<< num\_takeoff\_requests << endl

<< "Total number of planes accepted for landing "

<< num\_land\_accepted << endl

<< "Total number of planes accepted for takeoff "

<< num\_takeoff\_accepted << endl

<< "Total number of planes refused for landing "

<< num\_land\_refused << endl

<< "Total number of planes refused for takeoff "

<< num\_takeoff\_refused << endl

<< "Total number of planes that landed "

<< num\_landings << endl

<< "Total number of planes that took off "

<< num\_takeoffs << endl

<< "Total number of planes left in landing queue "

<< landing.size() << endl

<< "Total number of planes left in takeoff queue "

<< takeoff.size() << endl;

cout << "Percentage of time runway idle "

<< 100.0 \* (( float ) idle\_time) / (( float ) time) << "%" << endl;

cout << "Average wait in landing queue "

<< (( float ) land\_wait) / (( float ) num\_landings)

<< " time units";

cout << endl << "Average wait in takeoff queue "

<< (( float ) takeoff\_wait) / (( float ) num\_takeoffs)

<< " time units" << endl;

cout << "Average observed rate of planes wanting to land "

<< (( float ) num\_land\_requests) / (( float ) time)

<< " per time unit" << endl;

cout << "Average observed rate of planes wanting to take off "

<< (( float ) num\_takeoff\_requests) / (( float ) time)

<< " per time unit" << endl;

}

/\*

Post: Runway usage statistics are summarized and printed.

\*/

**Runway.hpp:**

#ifndef RUNWAY

#define RUNWAY

#include "Plane.hpp"

//#include "queue.hpp"

#include "random.hpp"

enum Runway\_activity {idle, land, take\_off};

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

**random.cpp:**

#include <limits.h>

const int max\_int = INT\_MAX;

#include <math.h>

#include <time.h>

#include "random.hpp"

Random::Random(bool pseudo)

{

if (pseudo) seed = 1;

else seed = time(NULL) % max\_int;

multiplier = 2743;

add\_on = 5923;

}

/\*

Post: The values of seed, add\_on, and multiplier

are initialized. The seed is initialized randomly only if

pseudo == false.

\*/

double Random::random\_real()

{

double max = max\_int + 1.0;

double temp = reseed();

if (temp < 0) temp = temp + max;

return temp / max;

}

/\*

Post: A random real number between 0 and 1 is returned.

\*/

int Random::random\_integer(int low, int high)

{

if (low > high) return random\_integer(high, low);

else return ((int) ((high - low + 1) \* random\_real())) + low;

}

/\*

Post: A random integer between low and high (inclusive)

is returned.

\*/

int Random::poisson(double mean)

{

double limit = exp(-mean);

double product = random\_real();

int count = 0;

while (product > limit) {

count++;

product \*= random\_real();

}

return count;

}

/\*

Post: A random integer, reflecting a Poisson distribution

with parameter mean, is returned.

\*/

int Random::reseed()

{

seed = seed \* multiplier + add\_on;

return seed;

}

/\*

Post: The seed is replaced by a psuedorandom successor.

\*/

**random.hpp:**

#ifndef RANDOM

#define RANDOM

class Random {

public:

Random(bool pseudo = true);

// Declare random-number generation methods here.

int random\_integer(int low, int high);

double random\_real();

int poisson(double mean);

private:

int reseed(); // Re-randomize the seed.

int seed, multiplier, add\_on; // constants for use in arithmetic operations

};

#endif

# 附录2：p1代码文件（随机模拟）

**main.cpp:**

#include <utility>

#include <iostream>

#include <cstdlib>

#include <ctime>

#include <fstream>

#include "Runway.hpp"

#include "Plane.hpp"

#include "random.hpp"

using namespace std;

typedef Plane Queue\_entry;

void run\_idle(int time)

{

//cout << time << ": Runway is idle." << endl;

}

/\*

Post: The specified time is printed with a message that the runway is idle.

\*/

void random\_rate\_init(double &ar, double &dr) {

ar = rand() % 1000 / (double)1001;

dr = rand() % 1000 / (double)1001;

}

int main()

{

int end\_time = 100; // time to run simulation

int queue\_limit = 5; // size of Runway queues

int flight\_number = 0;

srand(time(NULL));

ofstream out("data.txt", ios::out);

//initialize(end\_time, queue\_limit, arrival\_rate, departure\_rate);

for (int n = 0; n < 100; n++) {

double max\_arrival\_rate = 0, max\_departure\_rate = 0; /\* 用于储存最大的rate\*/

double arrival\_rate = 0, departure\_rate = 0;

for (int x = 0; x < 10000; x++) { /\* 实验1000次 在100time和5queue\_limit的情况下，选取随机的rate来模拟\*/

Random variable;

Runway small\_airport(queue\_limit);

random\_rate\_init(arrival\_rate, departure\_rate); //随机函数

bool flag = 0;

for (int current\_time = 0; current\_time < end\_time; current\_time++) {

// loop over time intervals

int number\_arrivals = variable.poisson(arrival\_rate);

// current arrival requests

for (int i = 0; i < number\_arrivals; i++) {

Plane current\_plane(flight\_number++, current\_time, arriving);

if (small\_airport.can\_land(current\_plane) != success) {

current\_plane.refuse();

flag = 1;

}

}

int number\_departures = variable.poisson(departure\_rate);

// current departure requests

for (int j = 0; j < number\_departures; j++) {

Plane current\_plane(flight\_number++, current\_time, departing);

if (small\_airport.can\_depart(current\_plane) != success) {

current\_plane.refuse();

flag = 1;

}

}

Plane moving\_plane;

switch (small\_airport.activity(current\_time, moving\_plane)) {

// let at most one plane onto the runway at current\_time.

case land:

moving\_plane.land(current\_time);

break;

case take\_off:

moving\_plane.fly(current\_time);

break;

case idle:

run\_idle(current\_time);

}

}

small\_airport.shut\_down(end\_time);

if (!small\_airport.if\_ever\_refused() && !flag) {

if (max\_arrival\_rate < arrival\_rate && max\_departure\_rate < departure\_rate) {

max\_arrival\_rate = arrival\_rate;

max\_departure\_rate = departure\_rate;

}

}

}

/\* cout << max\_arrival\_rate << " " << max\_departure\_rate << endl;\*/

out << max\_arrival\_rate << " " << max\_departure\_rate << endl;

}

out.close();

cout << "finished" << endl;

getchar();

return 0;

}

# 附录3：测试数据表格

|  |  |  |
| --- | --- | --- |
| end\_time = 100,queue\_limit = 3 | | |
| arrival\_rate | departure\_rate | sum |
| 0.507493 | 0.427572 | 0.935065 |
| 0.507493 | 0.425574 | 0.933067 |
| 0.504496 | 0.427572 | 0.932068 |
| 0.504496 | 0.426573 | 0.931069 |
| 0.507493 | 0.421578 | 0.929071 |
| 0.502498 | 0.425574 | 0.928072 |
| 0.502498 | 0.425574 | 0.928072 |
| 0.506494 | 0.421578 | 0.928072 |
| 0.506494 | 0.420579 | 0.927073 |
| 0.503497 | 0.423576 | 0.927073 |
| 0.507493 | 0.41958 | 0.927073 |
| 0.5005 | 0.425574 | 0.926074 |
| 0.507493 | 0.418581 | 0.926074 |
| 0.505495 | 0.41958 | 0.925075 |
| 0.5005 | 0.424575 | 0.925075 |
| 0.5005 | 0.424575 | 0.925075 |
| 0.502498 | 0.421578 | 0.924076 |
| 0.507493 | 0.415584 | 0.923077 |
| 0.4995 | 0.422577 | 0.922077 |
| 0.495504 | 0.425574 | 0.921078 |
| 0.493506 | 0.427572 | 0.921078 |
| 0.502498 | 0.417582 | 0.92008 |
| 0.497502 | 0.422577 | 0.920079 |
| 0.495504 | 0.422577 | 0.918081 |
| 0.507493 | 0.40959 | 0.917083 |
| 0.504496 | 0.411588 | 0.916084 |
| 0.491508 | 0.424575 | 0.916083 |
| 0.504496 | 0.410589 | 0.915085 |
| 0.5005 | 0.414585 | 0.915085 |
| 0.501499 | 0.412587 | 0.914086 |
| 0.505495 | 0.408591 | 0.914086 |
| 0.502498 | 0.410589 | 0.913087 |
| 0.507493 | 0.405594 | 0.913087 |
| 0.485514 | 0.426573 | 0.912087 |
| 0.477522 | 0.433566 | 0.911088 |
| 0.482517 | 0.427572 | 0.910089 |
| 0.486513 | 0.422577 | 0.90909 |
| 0.477522 | 0.430569 | 0.908091 |
| 0.505495 | 0.401598 | 0.907093 |
| 0.477522 | 0.428571 | 0.906093 |
| 0.463536 | 0.432567 | 0.896103 |
| 0.461538 | 0.433566 | 0.895104 |
| 0.471528 | 0.421578 | 0.893106 |
| 0.502498 | 0.38961 | 0.892108 |
| 0.463536 | 0.428571 | 0.892107 |
| 0.456543 | 0.433566 | 0.890109 |
| 0.457542 | 0.42957 | 0.887112 |
| 0.452547 | 0.430569 | 0.883116 |
| 0.43956 | 0.435564 | 0.875124 |
| 0.507493 | 0.365634 | 0.873127 |
| 0.441558 | 0.431568 | 0.873126 |
| 0.414585 | 0.446553 | 0.861138 |
| 0.414585 | 0.446553 | 0.861138 |
| 0.413586 | 0.446553 | 0.860139 |
| 0.412587 | 0.444555 | 0.857142 |
| 0.415584 | 0.436563 | 0.852147 |
| 0.404595 | 0.446553 | 0.851148 |
| 0.406593 | 0.442557 | 0.84915 |
| 0.413586 | 0.434565 | 0.848151 |
| 0.400599 | 0.443556 | 0.844155 |
| 0.3996 | 0.443556 | 0.843156 |
| 0.385614 | 0.444555 | 0.830169 |
| 0.380619 | 0.447552 | 0.828171 |
| 0.373626 | 0.447552 | 0.821178 |
| 0.515485 | 0.283716 | 0.799201 |
| 0.346653 | 0.448551 | 0.795204 |
| 0.604396 | 0.184815 | 0.789211 |
| 0.595405 | 0.185814 | 0.781219 |
| 0.626374 | 0.154845 | 0.781219 |
| 0.621379 | 0.155844 | 0.777223 |
| 0.604396 | 0.170829 | 0.775225 |
| 0.588412 | 0.185814 | 0.774226 |
| 0.601399 | 0.172827 | 0.774226 |
| 0.589411 | 0.182817 | 0.772228 |
| 0.543457 | 0.228771 | 0.772228 |
| 0.623377 | 0.148851 | 0.772228 |
| 0.542458 | 0.228771 | 0.771229 |
| 0.602398 | 0.165834 | 0.768232 |
| 0.609391 | 0.155844 | 0.765235 |
| 0.608392 | 0.154845 | 0.763237 |
| 0.595405 | 0.163836 | 0.759241 |
| 0.238761 | 0.519481 | 0.758242 |
| 0.234765 | 0.522478 | 0.757243 |
| 0.234765 | 0.518482 | 0.753247 |
| 0.22977 | 0.521479 | 0.751249 |
| 0.233766 | 0.516484 | 0.75025 |
| 0.230769 | 0.518482 | 0.749251 |
| 0.236763 | 0.511489 | 0.748252 |
| 0.230769 | 0.516484 | 0.747253 |
| 0.225774 | 0.521479 | 0.747253 |
| 0.237762 | 0.505495 | 0.743257 |
| 0.222777 | 0.519481 | 0.742258 |
| 0.231768 | 0.503497 | 0.735265 |
| 0.281718 | 0.448551 | 0.730269 |
| 0.207792 | 0.52048 | 0.728272 |
| 0.238761 | 0.487512 | 0.726273 |
| 0.207792 | 0.512488 | 0.72028 |
| 0.238761 | 0.45954 | 0.698301 |
| 0.0579421 | 0.556444 | 0.6143861 |
| 0.033966 | 0.557443 | 0.591409 |

|  |  |  |
| --- | --- | --- |
| end\_time = 100,queue\_limit = 5 | | |
| arrival\_rate | departure\_rate | sum |
| 0.534466 | 0.426573 | 0.961039 |
| 0.529471 | 0.427572 | 0.957043 |
| 0.529471 | 0.426573 | 0.956044 |
| 0.531469 | 0.424575 | 0.956044 |
| 0.532468 | 0.422577 | 0.955045 |
| 0.527473 | 0.426573 | 0.954046 |
| 0.445554 | 0.507493 | 0.953047 |
| 0.506494 | 0.446553 | 0.953047 |
| 0.504496 | 0.447552 | 0.952048 |
| 0.448551 | 0.503497 | 0.952048 |
| 0.532468 | 0.418581 | 0.951049 |
| 0.445554 | 0.505495 | 0.951049 |
| 0.532468 | 0.417582 | 0.95005 |
| 0.507493 | 0.442557 | 0.95005 |
| 0.527473 | 0.422577 | 0.95005 |
| 0.53047 | 0.418581 | 0.949051 |
| 0.448551 | 0.5005 | 0.949051 |
| 0.440559 | 0.507493 | 0.948052 |
| 0.501499 | 0.444555 | 0.946054 |
| 0.444555 | 0.501499 | 0.946054 |
| 0.518482 | 0.426573 | 0.945055 |
| 0.438561 | 0.506494 | 0.945055 |
| 0.526474 | 0.417582 | 0.944056 |
| 0.498501 | 0.445554 | 0.944055 |
| 0.501499 | 0.438561 | 0.94006 |
| 0.536464 | 0.401598 | 0.938062 |
| 0.48951 | 0.447552 | 0.937062 |
| 0.431568 | 0.503497 | 0.935065 |
| 0.434565 | 0.5005 | 0.935065 |
| 0.43956 | 0.494505 | 0.934065 |
| 0.0609391 | 0.872128 | 0.9330671 |
| 0.871129 | 0.0609391 | 0.9320681 |
| 0.423576 | 0.507493 | 0.931069 |
| 0.0599401 | 0.869131 | 0.9290711 |
| 0.447552 | 0.481518 | 0.92907 |
| 0.867133 | 0.0609391 | 0.9280721 |
| 0.869131 | 0.0579421 | 0.9270731 |
| 0.874126 | 0.0529471 | 0.9270731 |
| 0.430569 | 0.495504 | 0.926073 |
| 0.869131 | 0.0539461 | 0.9230771 |
| 0.519481 | 0.403596 | 0.923077 |
| 0.861139 | 0.0609391 | 0.9220781 |
| 0.0579421 | 0.863137 | 0.9210791 |
| 0.558442 | 0.355644 | 0.914086 |
| 0.407592 | 0.506494 | 0.914086 |
| 0.556444 | 0.355644 | 0.912088 |
| 0.403596 | 0.503497 | 0.907093 |
| 0.553447 | 0.352647 | 0.906094 |
| 0.547453 | 0.357642 | 0.905095 |
| 0.395604 | 0.507493 | 0.903097 |
| 0.558442 | 0.343656 | 0.902098 |
| 0.543457 | 0.354645 | 0.898102 |
| 0.04995 | 0.843157 | 0.893107 |
| 0.547453 | 0.345654 | 0.893107 |
| 0.357642 | 0.535465 | 0.893107 |
| 0.0609391 | 0.831169 | 0.8921081 |
| 0.355644 | 0.535465 | 0.891109 |
| 0.353646 | 0.536464 | 0.89011 |
| 0.559441 | 0.330669 | 0.89011 |
| 0.659341 | 0.22977 | 0.889111 |
| 0.659341 | 0.226773 | 0.886114 |
| 0.355644 | 0.528472 | 0.884116 |
| 0.644356 | 0.238761 | 0.883117 |
| 0.656344 | 0.225774 | 0.882118 |
| 0.657343 | 0.224775 | 0.882118 |
| 0.647353 | 0.234765 | 0.882118 |
| 0.654346 | 0.227772 | 0.882118 |
| 0.648352 | 0.230769 | 0.879121 |
| 0.351648 | 0.527473 | 0.879121 |
| 0.558442 | 0.320679 | 0.879121 |
| 0.221778 | 0.657343 | 0.879121 |
| 0.817183 | 0.0609391 | 0.8781221 |
| 0.21978 | 0.656344 | 0.876124 |
| 0.34965 | 0.525475 | 0.875125 |
| 0.215784 | 0.658342 | 0.874126 |
| 0.214785 | 0.659341 | 0.874126 |
| 0.342657 | 0.529471 | 0.872128 |
| 0.335664 | 0.536464 | 0.872128 |
| 0.21978 | 0.652348 | 0.872128 |
| 0.658342 | 0.213786 | 0.872128 |
| 0.216783 | 0.653347 | 0.87013 |
| 0.213786 | 0.653347 | 0.867133 |
| 0.237762 | 0.627373 | 0.865135 |
| 0.336663 | 0.527473 | 0.864136 |
| 0.625375 | 0.237762 | 0.863137 |
| 0.326673 | 0.535465 | 0.862138 |
| 0.238761 | 0.618382 | 0.857143 |
| 0.232767 | 0.624376 | 0.857143 |
| 0.324675 | 0.531469 | 0.856144 |
| 0.231768 | 0.621379 | 0.853147 |
| 0.233766 | 0.619381 | 0.853147 |
| 0.237762 | 0.614386 | 0.852148 |
| 0.237762 | 0.604396 | 0.842158 |
| 0.601399 | 0.235764 | 0.837163 |
| 0.773227 | 0.0609391 | 0.8341661 |
| 0.236763 | 0.597403 | 0.834166 |
| 0.226773 | 0.603397 | 0.83017 |
| 0.111888 | 0.659341 | 0.771229 |
| 0.118812 | 0.647001 | 0.765813 |

|  |  |  |
| --- | --- | --- |
| end\_time = 100,queue\_limit = 7 | | |
| arrival\_rate | departure\_rate | sum |
| 0.943057 | 0.157842 | 1.100899 |
| 0.94006 | 0.154845 | 1.094905 |
| 0.942058 | 0.152847 | 1.094905 |
| 0.938062 | 0.153846 | 1.091908 |
| 0.934066 | 0.157842 | 1.091908 |
| 0.941059 | 0.14985 | 1.090909 |
| 0.53047 | 0.559441 | 1.089911 |
| 0.529471 | 0.559441 | 1.088912 |
| 0.933067 | 0.155844 | 1.088911 |
| 0.934066 | 0.153846 | 1.087912 |
| 0.527473 | 0.559441 | 1.086914 |
| 0.932068 | 0.154845 | 1.086913 |
| 0.527473 | 0.557443 | 1.084916 |
| 0.529471 | 0.555445 | 1.084916 |
| 0.529471 | 0.555445 | 1.084916 |
| 0.53047 | 0.553447 | 1.083917 |
| 0.526474 | 0.556444 | 1.082918 |
| 0.53047 | 0.551449 | 1.081919 |
| 0.529471 | 0.552448 | 1.081919 |
| 0.94006 | 0.141858 | 1.081918 |
| 0.53047 | 0.549451 | 1.079921 |
| 0.523477 | 0.556444 | 1.079921 |
| 0.521479 | 0.556444 | 1.077923 |
| 0.527473 | 0.55045 | 1.077923 |
| 0.531469 | 0.546454 | 1.077923 |
| 0.526474 | 0.551449 | 1.077923 |
| 0.525475 | 0.55045 | 1.075925 |
| 0.531469 | 0.544456 | 1.075925 |
| 0.525475 | 0.55045 | 1.075925 |
| 0.522478 | 0.552448 | 1.074926 |
| 0.94006 | 0.134865 | 1.074925 |
| 0.52048 | 0.553447 | 1.073927 |
| 0.516484 | 0.557443 | 1.073927 |
| 0.513487 | 0.559441 | 1.072928 |
| 0.525475 | 0.547453 | 1.072928 |
| 0.918082 | 0.154845 | 1.072927 |
| 0.514486 | 0.557443 | 1.071929 |
| 0.526474 | 0.544456 | 1.07093 |
| 0.913087 | 0.157842 | 1.070929 |
| 0.51049 | 0.559441 | 1.069931 |
| 0.515485 | 0.554446 | 1.069931 |
| 0.91009 | 0.157842 | 1.067932 |
| 0.524476 | 0.542458 | 1.066934 |
| 0.91009 | 0.153846 | 1.063936 |
| 0.505495 | 0.557443 | 1.062938 |
| 0.503497 | 0.558442 | 1.061939 |
| 0.502498 | 0.559441 | 1.061939 |
| 0.526474 | 0.533467 | 1.059941 |
| 0.506494 | 0.553447 | 1.059941 |
| 0.5005 | 0.559441 | 1.059941 |
| 0.4995 | 0.558442 | 1.057942 |
| 0.495504 | 0.559441 | 1.054945 |
| 0.496503 | 0.557443 | 1.053946 |
| 0.527473 | 0.525475 | 1.052948 |
| 0.533467 | 0.518482 | 1.051949 |
| 0.492507 | 0.557443 | 1.04995 |
| 0.536464 | 0.509491 | 1.045955 |
| 0.988012 | 0.0579421 | 1.0459541 |
| 0.524476 | 0.52048 | 1.044956 |
| 0.527473 | 0.515485 | 1.042958 |
| 0.882118 | 0.154845 | 1.036963 |
| 0.535465 | 0.5005 | 1.035965 |
| 0.475524 | 0.559441 | 1.034965 |
| 0.475524 | 0.558442 | 1.033966 |
| 0.474525 | 0.559441 | 1.033966 |
| 0.941059 | 0.0919081 | 1.0329671 |
| 0.537463 | 0.494505 | 1.031968 |
| 0.472527 | 0.558442 | 1.030969 |
| 0.533467 | 0.476523 | 1.00999 |
| 0.536464 | 0.46953 | 1.005994 |
| 0.431568 | 0.559441 | 0.991009 |
| 0.555445 | 0.414585 | 0.97003 |
| 0.556444 | 0.412587 | 0.969031 |
| 0.549451 | 0.411588 | 0.961039 |
| 0.558442 | 0.397602 | 0.956044 |
| 0.557443 | 0.395604 | 0.953047 |
| 0.383616 | 0.559441 | 0.943057 |
| 0.706294 | 0.233766 | 0.94006 |
| 0.557443 | 0.371628 | 0.929071 |
| 0.69031 | 0.237762 | 0.928072 |
| 0.713287 | 0.208791 | 0.922078 |
| 0.682318 | 0.238761 | 0.921079 |
| 0.04995 | 0.87013 | 0.92008 |
| 0.721279 | 0.196803 | 0.918082 |
| 0.718282 | 0.197802 | 0.916084 |
| 0.0599401 | 0.855145 | 0.9150851 |
| 0.723277 | 0.191808 | 0.915085 |
| 0.0519481 | 0.861139 | 0.9130871 |
| 0.03996 | 0.871129 | 0.911089 |
| 0.676324 | 0.230769 | 0.907093 |
| 0.238761 | 0.655345 | 0.894106 |
| 0.647353 | 0.242757 | 0.89011 |
| 0.647353 | 0.240759 | 0.888112 |
| 0.235764 | 0.649351 | 0.885115 |
| 0.236763 | 0.647353 | 0.884116 |
| 0.233766 | 0.639361 | 0.873127 |
| 0.237762 | 0.635365 | 0.873127 |
| 0.228771 | 0.643357 | 0.872128 |
| 0.242757 | 0.625375 | 0.868132 |
| 0.23976 | 0.623377 | 0.863137 |