飞机场的模拟：

考虑一个小型繁忙的飞机场，该飞机场只有一条飞机跑道。在每个单位时间内，只有一架飞机可以着陆或者只有一架飞机可以起飞，不允许同时着陆和起飞。飞机的到达和起飞时是随机的，因此在任何时刻，该机场跑道可能是空闲的，或者有一架飞机正在着陆或起飞，也可能有若干架飞机在等待着陆或等待起飞。

我们需要创建的类有：

飞机类：

这个类描述的对象为单个飞机。

首先枚举飞机的状态通过以下语句：

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

然后描述两个类分别如下，

**class** Plane{

**public:**

Plane();

Plane(**int** flt, **int** time, Plane\_status status);

**void** refuse() **const**;

**void** land(int time) **cons**t;

**void** fly(int time) **const**;

**int** started() **const**;

**private:**

**int** flt\_num;

**int** clock\_start;

Plane\_status state;

};

跑道类：

enum Runway\_activity {idle, land, takeoff}

**class** Runway{

**public:**

Runway(int limit);

Error\_code can\_land(**const** Plane& current);

Error\_code can\_land(**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;//队列的最大长度

**int** num\_land\_requests; //要求降落的飞机数目

**int** num\_takeoff\_requests;//要求起飞的飞机数目

**int** num\_landing;//已降落的飞机数目

**int** num\_takeoffs;//已起飞的飞机数目

**int** num\_land\_accepted;//在降落队列里的飞机数目

**int** num\_takeoff\_accepted;//在起飞队列里的飞机数目

**int** num\_land\_refused;//被拒绝的要降落飞机数目

**int** num\_takeoff\_refused;//被拒绝的要起飞飞机数目

**int** land\_wait; //飞机等待降落的总时间

**int** takeoff\_wait;//飞机等待起飞的总时间

**int** idle\_time;//机场处于空闲状态的总时间

};

其中

关闭函数定义如下

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

**//跑道数据在这里总结计算并打印出来**

{

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 taking 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\_takeoff\_accepted<<endl

<<``Total number of planes refused for takeoff’’

<< num\_takeoff\_accepted<<endl

<<``Total number of planes that landed’’

<< num\_takeoff\_accepted<<endl

<<``Total number of planes that took offf’’

<< num\_takeoff\_accepted<<endl

<<``Total number of planes left 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;

}

主函数

void main()

{

int end\_time;

int queue\_limit;

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++){

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

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);

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)){

case land:

moving\_plane.land(current\_time);

break;

case land:

moving\_plane.fly(current\_time);

break;

case idle;

run\_idle(current\_time);

}

}

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

}

其中初始函数如下：

**void** initialize(int &end\_time, int &queue\_limit, double &arrival\_rate, double&departure\_rate)

**//用户在此声明仿真所需的time units（时间单位）的数量，队列的最大长度和机场的平均到达率和离开率**。

{

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

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

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

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

cin >> queue\_limit;

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

cin >> end\_time;

bool acceptable;

do {

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

cin >> arrival\_rate;

cout << "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);

}

main函数跑完后窗口出现的结果如下：





