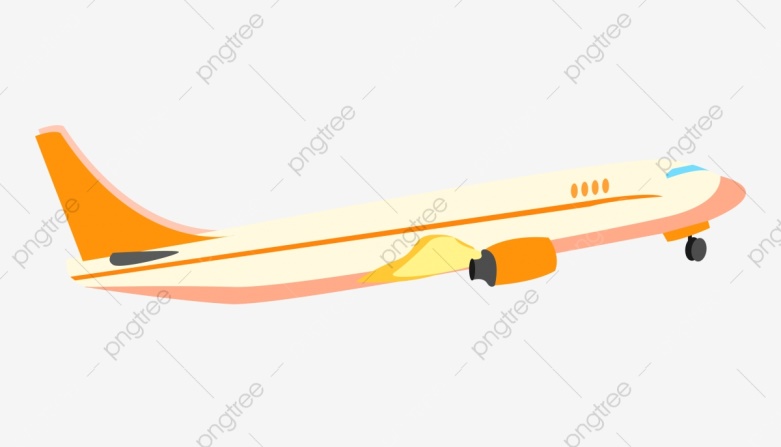


Simulating an airport traffic control operation



|  |  |
| --- | --- |
| 219009389 | Ghaida alarfaj |
| 219010636 | Noura almosened |
| 219022773 | Sara Almudairs |
| 219032482 | Zahra alabbad |

ID:

Name**:**

M e m b e r s

CODE TASKS :

**NAME:** **TASKS:**

|  |  |
| --- | --- |
| The two structs and their function ( isFull, enQueue, , newRecord, etc | Ghaida alarfaj |
| 1)Make function for takeoff list (succTookoff)  2)Make function for generating numbers | **Noura almosened** |
| 1)Print function include the averages  2) Debug the code | **Sara Almudairs** |
| 1)Make function for land list (succLand) | **Zahra alabbad** |

TASKS FOR REPORT:

**NAME:** **TASKS:**

|  |  |
| --- | --- |
| 1)Introduction  2) Data structure used for the solution and why? | Ghaida alarfaj |
| 1)Introduction  2) Data structure used for the solution and why?  1)Design the project report + make(cover page-kfu logo-student name and id)+contents  2- ALGORITHM/PSEUDOCODE | **Noura almosened** |
| 1) Results and discussion  2) Abstract | **Sara Almudairs** |
| 1)Problem statement  2) conclusion | **Zahra alabbad** |

CONTENT:

|  |
| --- |
| ABSTRACT /SUMMARY |
| INTRIDUCTION |
| PROBLEM STATMENT |
| METHODLOGY(DATA STRUCTURE USE,ALGORITHM/PSEUDOCODE,EXECUATBLE PROGRAM CODE) |
| RESULTS AND DISCUSSION |
| CONCLUSION |
| REFRENCES |

introduction

This is project on Simulating An Airport Operation using modern c++ programming. Since airports have become very important nowadays, and airplanes are used on a daily basis in all countries. However, there are too many problems that occur at airports, especially if the airport is small and there is only one runway. Example of problems is, There could be a collision between planes, one plane wanting to land and the another wanting to take off at the same time. Therefore, there will be a huge crowed at the runway. Because of these problems, we decided to design this project. So, the purpose of this project is to control the traffic operation on the airport



Problem statement

The airport has one runway, the plane comes at random instants of time, there is priority for landing planes, The takeoff planes still wait more time until there’re no more landing planes, there is a maximum number of planes that can wait

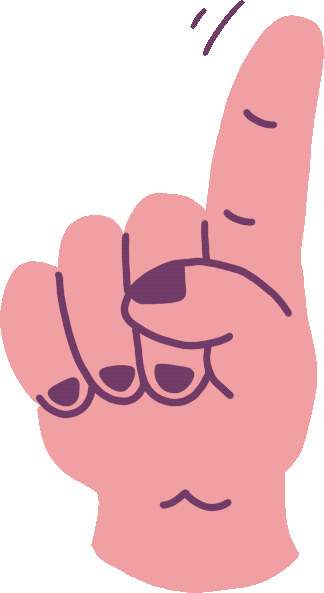


methodology

Set constant value for time to land and take off =30  
CurentTime=1;  
countP=0;//variable just to keep track to check the statue of queue  
arrival time =0;  
for loop starting from current time to the number of execution number of programme  
NewRecord(a,CurrentTime)  
{  
numPalnes++ //total number of plans prosecced  
arrival time=currentTime  
}

If(random number<time to land)  
1-Add plane to land’s queue and increase the number of plans processed for landing queue   
Else  
Increase the number of refuse  
End if  
If(random number<time to take off)  
1-Add plane to take off queue and and increase the number of plans processed for take off queue  
Else  
Increase the number of refuse

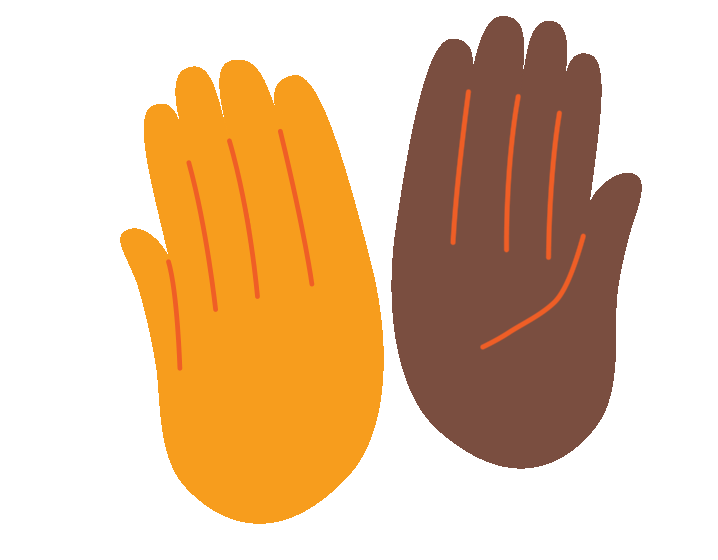
isFullAirport()

If (countP>=size)  
Return true  
Else  
Return false  
End if

isemptyAirport()

If (CountP <=0)

Return true

Else

Return false

End if

airportEnqueu(Rear,Data,currentTime

If(queue for takeoff is not fall)

1-Rear=(Rear+1)%size

2-Data[Rear]=currentTime

3-countP++;

End if

If(queue for landing is not fall)

1-Rear=(Rear+1)%size

2-Data[Rear]=currentTime

3-countP++

End if

If (airport for landing queue is not empy)

1-dequeue landing queue

2-temp =arrivalTime

3-succLand(a,temp,curTime)

Else

If(airport for takeoff queue is not empty)

1-dequeue takeoff queue

2-temp =arrivalTime

succTookoff(a,temp,curtime)

Else

Idle++;

Airportdequeue(front,data,plan)

Int plane;  
1-If(landing queue is not empty)

front=(front+1)%max  
 plane=data[front]

2-countP--  
3-Return plane  
else  
1-If(takeoff queue is not empty)  
front=(front+1)%max  
 plane=data[front]

2-countP--  
3-Return plane  
End if

Int wait=0,tookoff,takeoffwait

succLand(a,temp,currentTime)

1-Wait=currentTime-arriavlTime

2-Tookoff++//Inrease the number of tookoff planes

3-takeoffWait+=wait//to calculate total time for takeoff palnes

Int wait=0,tookoff,takeoffwait

succTookoff(a,temp,currentTime)

1-Wait=currentTime-arriavlTime

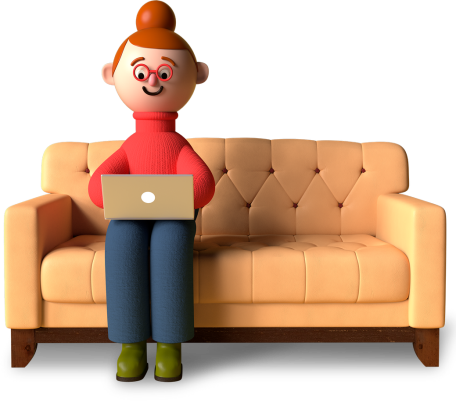
2-Inrease the number of tookoff planes

3-takeoffWait+=wait//to calculate total time for takeoff planes

3-takeoffWait+=wait//to calculate total time for takeoff planes

Data structure used for the solution and why**?**

The main important thing in this program is : the arriving airplanes and departing airplanes must use the same runway. For that reason, it should be allocated to the airplanes that has been waiting the longest amount of time. So, the appropriate structure for this program is queue because Queue use FIFO method, but since we want to use this runway more than once we decided to use ” CURCULAR QUEUE ”.



conclusion

This simulation model program help to avoid congestion issues consist for the landing and takeoff planes, the main idea is to give the plane which has to landing more priority than the take of plane There is also a maximum number of plan that will landing or take off and if it reach this point the airport will refuse the new plane

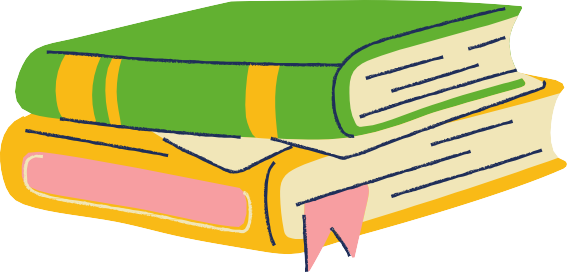
Abstract / Summary  
.

In this project, we created a program to simulate a busy airport with only one runway, and only one landing and taking off planes arriving each hour.

The program helps in dealing with planes requests for landing or taking off and controlling airplane traffic in the runway easily. Before accepting new requests, the program will ensure that the arriving time for the planes (1-60) won’t be more than the time needed for landing/ taking off. After accepting the requests, planes will enter the waiting list and will be processed according to order, the planes that requested first will be processed first.

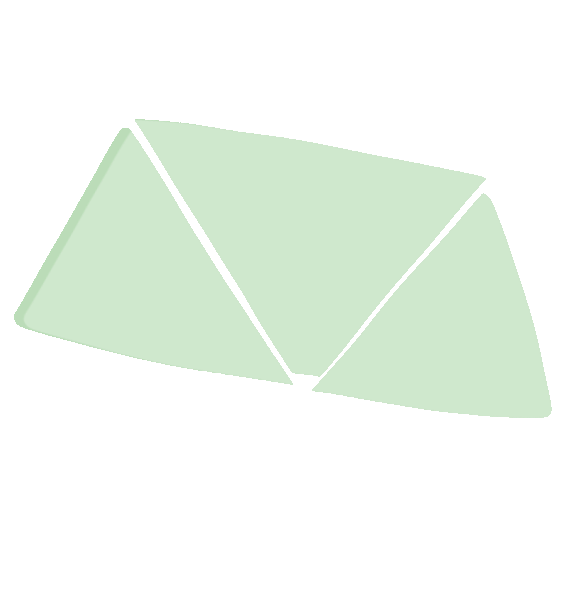
The program will seamlessly deal with the problems that the airport might face, such as planes requesting to land and fly at the same time, or airport reaching its capacity for the waiting planes, while maintaining the safety of the airplanes.

The program will print important statistics at each unit time to keep the user informed, including, total number of planes processed – total number planes accepted for landing/ taking off – total number of planes refused - total number of planes successfully landed/ took off – average waiting time for landing/ taking off – total number of times runway was idle.



Results and discussion

|  |  |
| --- | --- |
| Looking at the final output for the average waiting time, we notice that the waiting time for the takeoff is times more than the landing, which’s due always giving the priority to landing. |  |
|  | From the total planes processed, planes will only get accepted or refused. Then the accepted will eventually land/ take off successfully after waiting enough time in the queue. Meaning the airport will 100% process all the airplanes. |
| Despite having one runway, the program process only one plane at a time, so there’s no collision between the airplanes |  |



Output:

صورة تحتوي على نص

تم إنشاء الوصف تلقائياً

صورة تحتوي على نص

تم إنشاء الوصف تلقائياً

|  |  |
| --- | --- |
| https://www.programiz.com/dsa/circular-queue | Ghaida alarfaj |
| [https://www.bitdegree.org/learn/random-number-generator-cpp#:~:text=You%20can%20create%20a%20random,seed)%20and%20the%20maximum%20value.](https://www.bitdegree.org/learn/random-number-generator-cpp) | **Noura almosened** |
|  | **Sara Almudairs** |
| https://www.chegg.com/homework-help/questions-and-answers/airport-simulation-program-java-write-simulation-program-small-airport-one-runway-queue-pl-q9477193 | **Zahra alabbad** |

references



CODE :

/\*The project is on simulating an airport traffic control operation

authors : Ghaida , Noura , Sara , Zahra \*/

#include <iostream>

#include<ctime>//for rand() method, it return a postive numbers within the rasnge from 0.0 to RAND\_MAX

#include <cstdlib>//for time to making the random numbers different after every execution

**using** **namespace** std;

#define MAXP 5//the number of plans (array size)

**void** display(){

cout << "\n\t \*\*\*\*\*\*\* ";

cout << "\n\t \* \* ";

cout << "\n\t \* \* ";

cout << "\n\t \*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ";

cout << "\n\t \* \*\*\*\* ";

cout << "\n\t \* \*\*\*\*\* ";

cout << "\n\t \* \* ";

cout << "\n\t \* \* ";

cout << "\n\t \*\*\* \*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\* ";

cout << "\n\t \* \* \* \* ";

cout << "\n\t \* \* \* \* ";

cout << "\n\t \*\*\*\*\*\*\*\*\*\* \* \* ";

cout << "\n\t \*\*\*\*\*\*\* ";

cout << "\n \n";

}

**struct** plane{

**int** timeArrive;//time of arrival in queue

}Plane;

**struct** circQueue{

**int** countP;//number of airplanes in the queue

**int** front;//front of the queue

**int** rear;//rear of the queue

plane data[MAXP];//the number of plane, ex: plane num 1 has this id .....

}Queue;

**void** initialQue(circQueue \*que){ // to initialize queue at main

que -> countP = 0;

que -> front = -1;

que -> rear = -1;

}

**bool** isEmpty(circQueue \*que){ // to check whether the queue is empty or not

**if** (que -> countP <= 0)

**return** **true**;

**else**

**return** **false**;

}

**bool** isFull(circQueue \*que){ // to check whether the queue is full or not

**if** (que -> countP >= MAXP)

**return** **true**;

**else**

**return** **false**;

}

**void** enQueue(circQueue \*que , plane p){ // insert plane from the rear of the queue

que -> countP ++;

que -> rear = (que -> rear + 1) % MAXP;//ex : (-1 + 1) % 5 = 0

que -> data[que -> rear] = p;

}

plane deQueue(circQueue \*que){ // delete plane from the front of the queue

que -> countP --;

Plane = que -> data[que -> front];

que -> front = ( que -> front + 1 ) % MAXP;//(0 + 1) % 5 = 1

**return** Plane;

}

**struct** mangeAirport{

circQueue landing; // variable to represent landing queue

circQueue takeoff; // variable to represent taking off queue

circQueue \*planeLanding; // pointer for landing

circQueue \*planeTakeoff; // pointer for taking off

**float** LandWait, TakeoffWait; // to calculate the total waiting time

**int** numLand, numTakeoff; // number of requests that were accepted for landing or taking off

**int** landed, tookoff, numPlanes, numRefuse;//number of planes landed, number of planes took off, number of planes processed, number of planes refused to use the runway

**int** idle; // when there is no landing or takeoff on the Runway

**int** curTime;//time for process(landing or take off )

plane p;

};

**void** initialAirport(mangeAirport \*a){

initialQue(&(a -> landing));//landing now has countP, front and rear

initialQue(&(a -> takeoff));//takeoff now has countP, front, and rear

a -> planeLanding = &(a -> landing);//as a counter for the plane that landing

a -> planeTakeoff = &(a -> takeoff);//as a counter for the plane that takeoff

a -> LandWait = 0.0;

a -> TakeoffWait = 0.0;

a -> numLand = 0;

a -> numTakeoff = 0;

a -> landed = 0;

a -> tookoff = 0;

a -> numPlanes = 0;

a -> numRefuse = 0;

a -> idle = 0;

}

**void** newRecord(mangeAirport \*a, **int** curTime){//make a new record for a plane

a -> numPlanes ++;

a -> p.timeArrive = curTime;

}

**void** succLand(mangeAirport \*a, plane pl, **int** curtime ) // bring planes to successfully land

{

**int** wait; // variable to calculate waiting time

wait = curtime - pl.timeArrive; // Calculates the wait time by subtracting the current time from the time it entered the queue

a -> landed ++;

a -> LandWait += wait; // to calculate the total waiting time for landing planes

}

**void** succTookoff(mangeAirport \*a, plane pl, **int** curtime ) // bring planes to take off

{

**int** wait; // variable to calculate waiting time

wait = curtime - pl.timeArrive; // Calculates the wait time by subtracting the current time from the time it entered the queue

a -> tookoff ++;

a -> TakeoffWait += wait; // to calculate the total waiting time for taking off planes

}

**int** airportIsEmpty(mangeAirport \*a, **int** kind){//kind : 1 for land, 2 for takeoff

**if**(kind==1)

{

**return** (isEmpty(a -> planeLanding));

}

**else**

**return** (isEmpty(a -> planeTakeoff));

}

**int** airportIsFull(mangeAirport \*a, **int** kind){//kind : 1 for land, 2 for takeoff

**if**(kind==1)

{

**return** (isFull(a -> planeLanding));

}

**else**

{

**return** (isFull(a -> planeTakeoff));

}

}

**void** airportEnQueue(mangeAirport \*a, **int** kind){//kind : 1 for land, 2 for takeoff

**if**(kind==1)

{

enQueue(a -> planeLanding, a -> p);

a -> numLand ++;

}

**else**

{

enQueue(a -> planeTakeoff, a -> p);

a -> numTakeoff ++;

}

}

plane airportDeQueue(mangeAirport \*a, **int** kind){//kind : 1 for land, 2 for takeoff

**if**(kind==1)

{

Plane = deQueue(a -> planeLanding);

}

**else**

{

Plane = deQueue(a -> planeTakeoff);

}

**return** Plane;

}

**void** print(mangeAirport \*a, **int** fRandom, **int** sRandom){ // function to print statistics

cout << "Total number of planes processed " << a -> numPlanes << endl;

cout << "Total number of planes accepted for landing " << a -> numLand << endl;

cout << "Total number of planes accepted for taking off " << a -> numTakeoff << endl;

cout << "Total number of planes refused " << a -> numRefuse << endl;

cout << "Total number of planes successfully landed " << a -> landed << endl;

cout << "Total number of planes successfully took off " << a -> tookoff << endl;

cout << "Total number of times runway was idle " << a -> idle << endl;

**if** (a -> landed > 0) // calculating the average waiting time when we have planes successfully landed

cout << "The average wait time to land : " << (a -> LandWait / a -> landed) << endl;

**else** // if no planes landed, it will print the number as it is without calculating (zero)

cout << "The average wait time to land : " << a -> landed << endl;

**if** (a -> tookoff > 0) // calculating the average waiting time when we have planes successfully tookoff

cout << "The average wait time to takeoff : " << (a -> TakeoffWait / a -> tookoff) << endl;

**else** // if no planes tookoff, it will print the number as it is without calculating (zero)

cout << "The average wait time to takeoff: " << a -> tookoff << endl;

cout << "-------------------------------------------" <<endl;

}

**int** main(){

mangeAirport a; // object of struct airport

**int** curTime; // to keep track of the loop iterations, that will represent time

**int** timetoland = 30; // time needed for planes to land (minutes)

**int** timetotakeoff = 30; // time needed for planes to take off (minutes)

plane temp; // object of struct plane

display(); // to display the plane shape!

initialAirport(&a); // initializing queue variables

srand((**unsigned**)time(0));//function to genrate random numbers

cout<<"\n- The program will run for 75 time units -" << endl << endl;

**for** (curTime = 1; curTime <= 75; curTime++){// for loop from 1 to 75 (time units) to perform the simulation

**int** fRandom = (rand() % 60)+1; //genrate random number between 1-60 for landing plane

**if**(fRandom < timetoland){// random needs to less than the time needed to land in order for it to accepted

newRecord(&a, curTime);// creates new plane for landing

**if** (airportIsFull(&a, 1)) // to check whether there's enough space in the landing queue

a.numRefuse++; // true (it's full), the planes will be refused

**else**

airportEnQueue(&a, 1); // false (not full), the planes will be added to the landing queue

}

**int** sRandom = (rand() % 60)+1; //genrate random number between 1-60 for taking off plane

**if** (sRandom < timetotakeoff) { // random needs to less than the time needed to takeff in order for it to accepted

newRecord(&a, curTime);//creates new plane for taking off

**if** (airportIsFull(&a, 2)) // to check whether there's enough space in the takeoff queue

a.numRefuse++; // true (it's full), the planes will be refused

**else**

airportEnQueue(&a, 2); // false (not full), the planes will be added to the takeoff queue

}

**if** (!(airportIsEmpty(&a, 1))){// processeing landing planes first when there since they have priority

temp = airportDeQueue(&a, 1); // store the plane that will pop in temp

succLand( &a, temp, curTime ); // take action on the plane that was dequeued

}

**else** **if** (!(airportIsEmpty(&a, 2))){ // only when landing is empty, taking off planes will be processed

temp = airportDeQueue(&a, 2); // store the plane that will pop in temp

succTookoff( &a, temp, curTime ) ; // take action on the plane that was dequeued

}

**else** // both queues are empty

a.idle ++;

cout<<"\n\tInformation for: " << curTime << " proccess ....\t" << endl << endl;

print(&a, fRandom, sRandom); // prints statistics each time unit

}

**return** 0;

}