**A Report on Open ended problem titled**

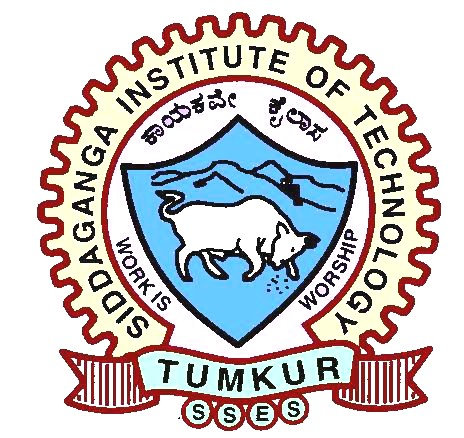
“AIR TRAFFIC CONTROLLER”

*Submitted for partial fulfillment of III semester Data Structures Laboratory*

by

**Team Members:**

1. **Name: MRIGANK SRIVASTAVA USN: 1SI17CS061**
2. **Name: MRINAL RAJ USN:1SI17CS062**
3. **Name: NITIKIA SINGH USN:1SI17CS070**
4. **Name: NIKHIL SAH USN:1SI17CS067**



**Department of Computer Science & Engineering**

**Siddaganga Institute of Technology, Tumakuru-3**

(An Autonomous Institute affiliated to VTU, Accredited by NBA)

**2018-19**

TABLE OF CONTENTS

SL.NO CONTENT PAGE.NO

1. ABSTRACT 2
2. INTRODUCTION 3
3. IMPLEMENTATION 5
4. CONCLUSION 22

**AIR TRAFFIC CONTROLLER**

A program in C that will allow the employee of the air traffic controller to update the information of the flight and also give them permission to takeoff and landing.

**ABSTRACT:**

In the Linked List there will be N such nodes which are for takeoff and landing The terminal window will be open with a welcome and asking the airline employee to register the flight details. Each node in the list will be holding flight ID, arrival time, departure time.

Once the flight details has been entered the **ATC**can access to the flight details and give permission to the pilots weather they need to takeoff or land from the runway.

**DATA STRUCTURE USED**

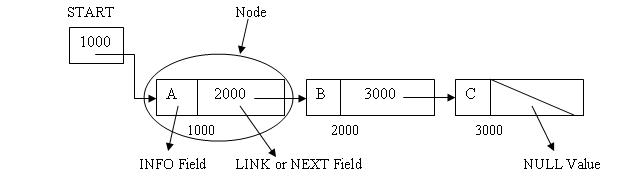
* Linked list
* Structures

INTRODUCTION

The Data structures used in the programs are:

1. LINKED LIST:

Linked List is a linear data structure and it is very common data structure which consists of group of nodes in a sequence which is divided in two parts. Each node consists of its own data and the address of the next node and forms a chain. Linked Lists are used to create trees and graphs.



#### Advantages of Linked Lists

* They are a dynamic in nature which allocates the memory when required.
* Insertion and deletion operations can be easily implemented.
* Stacks and queues can be easily executed.
* Linked List reduces the access time.

#### Disadvantages of Linked Lists

* The memory is wasted as pointers require extra memory for storage.
* No element can be accessed randomly; it has to access each node sequentially.
* Reverse Traversing is difficult in linked list.

#### Applications of Linked Lists

* Linked lists are used to implement stacks, queues, graphs, etc.
* Linked lists let you insert elements at the beginning and end of the list.
* In Linked Lists we don't need to know the size in advance.

1. STRUCTURES:

Structure is a collection of different data types under a single name.

**For example:** You want to store some information about a person: his/her name, citizenship number and salary. You can easily create different variables name cityno to store these information separately.

However, in the future, you would want to store information about multiple persons. Now, you'd need to create different variables for each information per person: name1, citNo1, salary1, name2, citNo2, salary2

You can easily visualize how big and messy the code would look. Also, since no relation between the variables (information) would exist, it's going to be a daunting task.

A better approach will be to have a collection of all related information under a single name Person, and use it for every person. Now, the code looks much cleaner, readable and efficient as well.

This collection of all related information under a single name Person is a structure.

IMPLEMENTATION

WORKING CODE:

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include<time.h>

#include<string.h>

#define BAYFULL 10

typedef struct

{

int day,month,year,hour,min;

}TIME\_T;

struct node

{

char flight\_id[11];

TIME\_T arvltm,dprttm;

struct node \* link;

};

typedef struct node \* NODEPTR;

NODEPTR fnRegister();

bool rnwyoccupied(int \*);

void fnDelete(NODEPTR,NODEPTR);

NODEPTR fnInsertArrivalNodeOrder(NODEPTR ,NODEPTR );

NODEPTR fnInsertDepartureNodeOrder(NODEPTR ,NODEPTR );

bool fnArrivalConditionLessThan(NODEPTR ,NODEPTR );

bool fnDepartureConditionLessThan(NODEPTR ,NODEPTR );

NODEPTR fnGetNode();

NODEPTR fnGetArrivalTime(NODEPTR);

NODEPTR fnGetDepartureTime(NODEPTR);

NODEPTR fnSearchId(NODEPTR,char[]);

void fnDisplayTime(TIME\_T);

bool FnBayFull(int);

void fnGetTime(int \*,int \*);

void fnDisplayInfo(NODEPTR);

bool fnInvalidDates(TIME\_T);

int main()

{

NODEPTR temp,takeoff=NULL,land=NULL;

int ch,rnwy1,rnwy2,bay=0;

char id[11];

for(;;)

{

printf("\n1.Registration\t2.Take off\t3.Landing\t4.Display list and exit\n");

printf("\nEnter your choice:");

scanf("%d",&ch);

switch(ch)

{

case 1: temp=fnRegister();

land=fnInsertArrivalNodeOrder(temp,land);

takeoff=fnInsertDepartureNodeOrder(temp,takeoff);

break;

case 2: if(!rnwyoccupied(&rnwy1))

{

printf("\nEnter your unique id\n");

getchar();

scanf("%s",id);

temp=fnSearchId(takeoff,id);

if(temp==NULL)

break;

printf("\nYou are permitted to take off from runway 1\n");

fnDelete(temp,takeoff);

bay--;

}

else if(!rnwyoccupied(&rnwy2))

{

printf("\nEnter your unique id\n");

getchar();

scanf("%s",id);

temp=fnSearchId(takeoff,id);

if(temp==NULL)

break;

printf("\nYou are permitted to take off from runway 2\n");

fnDelete(temp,takeoff);

bay--;

}

else

printf("\nSorry,but you will have to wait for ten minutes so that we can arrange for you\n");

break;

case 3: if(!rnwyoccupied(&rnwy1)&&!FnBayFull(bay))

{

printf("\nEnter your unique id\n");

getchar();

scanf("%s",id);

temp=fnSearchId(takeoff,id);

if(temp==NULL)

break;

printf("\nYou are permitted to use runway 1 for landing\n");

fnDelete(temp,land);

bay++;

}

else if(!rnwyoccupied(&rnwy2)&&!FnBayFull(bay))

{

printf("\nEnter your unique id\n");

getchar();

scanf("%s",id);

temp=fnSearchId(takeoff,id);

if(temp==NULL)

break;

printf("\nYou are permitted to use runway 2 for landing\n");

fnDelete(temp,land);

bay++;

}

else

printf("\nSorry,but you will have to wait for ten minutes so that we can arrange for you\n");

break;

case 4: fnDisplayInfo(land);

printf("\n\nExiting...\n\n");

exit(0);

break;

default:printf("\nInvalid choice,re enter the choice\n");

}

}

return 0;

}

void fnDisplayInfo(NODEPTR land)

{

printf("\nThe flights that have been registered are:\n");

if(land==NULL)

{

printf("\nList currently empty,nothing to display\n");

return;

}

NODEPTR cur;

cur=land;

while(cur!=NULL)

{

printf("\nFlight id: %s\n",cur->flight\_id);

printf("\nArrival time:");

fnDisplayTime(cur->arvltm);

printf("\nDeparture time:");

fnDisplayTime(cur->dprttm);

cur=cur->link;

}

}

bool FnBayFull(int bay)

{

if(bay==BAYFULL)

return true;

else

return false;

}

NODEPTR fnRegister()

{

NODEPTR flight;

flight=fnGetNode();

flight->link=NULL;

printf("\nEnter your unique id\n");

getchar();

scanf("%s",flight->flight\_id);

flight=fnGetArrivalTime(flight);

while(fnInvalidDates(flight->arvltm))

{

printf("\nInvalid arrival time,please re enter\n");

flight=fnGetArrivalTime(flight);

}

flight=fnGetDepartureTime(flight);

while(fnInvalidDates(flight->dprttm))

{

printf("\nInvalid departure time,please re enter\n");

flight=fnGetDepartureTime(flight);

}

printf("\nYou have successfully registered.\nYou will be considered and we will try our best to make arrangements for it\n");

return flight;

}

NODEPTR fnInsertArrivalNodeOrder(NODEPTR temp,NODEPTR land)

{

NODEPTR prev,cur;

if(land ==NULL)

{

return temp;

}

if(fnArrivalConditionLessThan(temp,land))

{

temp->link=land;

return temp;

}

prev=NULL;

cur=land;

while(cur!=NULL && !fnArrivalConditionLessThan(temp,cur))

{

prev=cur;

cur=cur->link;

}

prev->link=temp;

temp->link=cur;

return land;

}

NODEPTR fnGetArrivalTime(NODEPTR flight)

{

printf("\nEnter your arrival time\n");

printf("\nDay:");

scanf("%d",&flight->arvltm.day);

printf("\nMonth:");

scanf("%d",&flight->arvltm.month);

printf("\nYear:");

scanf("%d",&flight->arvltm.year);

printf("\nHour:");

scanf("%d",&flight->arvltm.hour);

printf("\nMinute:");

scanf("%d",&flight->arvltm.min);

return flight;

}

NODEPTR fnGetDepartureTime(NODEPTR flight)

{

printf("\nEnter your departure time\n");

printf("\nDay:");

scanf("%d",&flight->dprttm.day);

printf("\nMonth:");

scanf("%d",&flight->dprttm.month);

printf("\nYear:");

scanf("%d",&flight->dprttm.year);

printf("\nHour:");

scanf("%d",&flight->dprttm.hour);

printf("\nMinute:");

scanf("%d",&flight->dprttm.min);

return flight;

}

NODEPTR fnInsertDepartureNodeOrder(NODEPTR temp,NODEPTR takeoff)

{

NODEPTR prev,cur;

if(takeoff ==NULL)

{

return temp;

}

if(fnDepartureConditionLessThan(temp,takeoff))

{

temp->link=takeoff;

return temp;

}

prev=NULL;

cur=takeoff;

while(cur!=NULL && !fnDepartureConditionLessThan(temp,cur))

{

prev=cur;

cur=cur->link;

}

prev->link=temp;

temp->link=cur;

return takeoff;

}

bool rnwyoccupied(int\* r)

{

printf("The runway is occupied? Enter 0 or 1\n");

scanf("%d",r);

if(\*r!=0 &&\*r!=1)

{

printf("\nInvalid Input. Please Re-enter If ");

rnwyoccupied(r);

}

if(\*r==0)

return false;

else

return true;

}

bool fnArrivalConditionLessThan(NODEPTR temp,NODEPTR cur)

{

if((temp->arvltm.year <= cur->arvltm.year)&&(temp->arvltm.month <= cur->arvltm.month)&&(temp->arvltm.day <= cur->arvltm.day))

{

if((temp->arvltm.hour <= cur->arvltm.hour)&&(temp->arvltm.min < cur->arvltm.min))

{

return true;

}

}

return false;

}

bool fnDepartureConditionLessThan(NODEPTR temp,NODEPTR cur)

{

if((temp->dprttm.year <= cur->dprttm.year)&&(temp->dprttm.month <= cur->dprttm.month)&&(temp->dprttm.day <= cur->dprttm.day))

{

if((temp->dprttm.hour <= cur->dprttm.hour)&&(temp->dprttm.min <= cur->dprttm.min))

{

return true;

}

}

return false;

}

bool fnInvalidDates(TIME\_T t)

{

if(t.day>31||t.day<1||t.month<1||t.month>12||t.hour<0||t.hour>23||t.min<0||t.min>60)

return true;

return false;

}

NODEPTR fnGetNode()

{

NODEPTR temp;

temp=(NODEPTR)malloc(sizeof(struct node));

if(temp==NULL)

{

printf("\nFailed to allocate memory\n");

return NULL;

}

return temp;

}

NODEPTR fnSearchId(NODEPTR first,char id[])

{

if(first==NULL)

{

printf("\nList empty,cannot search\n");

return NULL;

}

if(strcmp(first->flight\_id,id)==0)

return first;

NODEPTR prev,cur;

prev=NULL;

cur=first;

while(cur!=NULL&&strcmp(cur->flight\_id,id)!=0)

{

prev=cur;

cur=cur->link;

}

if(cur==NULL)

{

printf("\nFlight not found in the list,please re enter your id\n");

return NULL;

}

return cur;

}

void fnDelete(NODEPTR temp,NODEPTR first)

{

NODEPTR prev,cur;

if(temp==first)

{

cur=first->link;

first=cur;

free(cur);

}

if(temp==first->link)

{

prev=first->link;

cur=prev->link;

first->link=cur;

free(cur);

}

}

void fnDisplayTime(TIME\_T time)

{

printf("\n%2d/%2d/%d\t%2d:%2d\n",time.day,time.month,time.year,time.hour,time.min);

return;

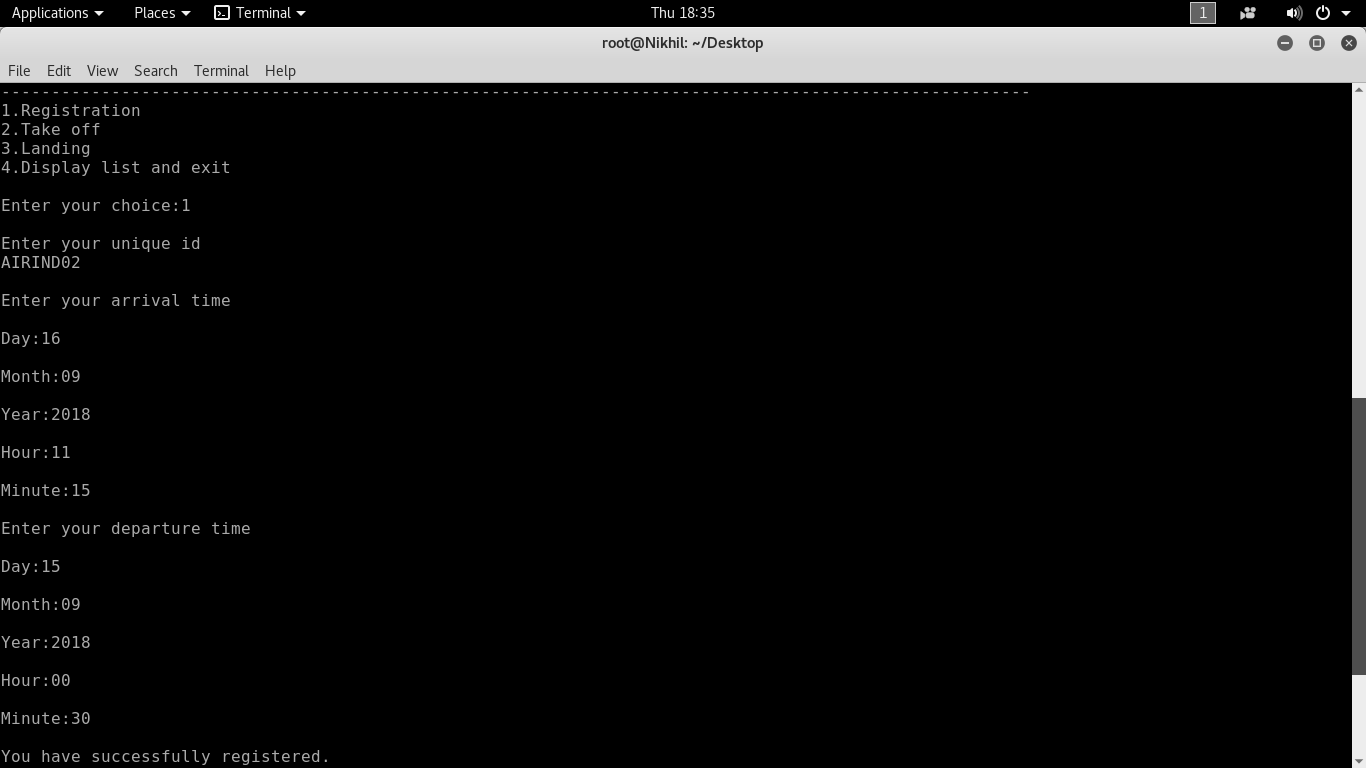
}



Ss: Opening a terminal



Ss: Choice Menu



Ss: Data Entered



Ss: Asking for take off



Ss: Asking for landing

CONCLUSION

On conclusion firstly we would like to thank our faculty for this project Mr. PRABODH CP for giving us such a wonderful opportunity to peek out from our course syllabus and realized some real time application of our study material.

The program is all about maintaining the flight traffic considering their information,such as landing time, departure time. There are some program strengths as well as weakness.

**STRENGTHS**

* As linked list is used it is more flexible to add or delete information.
* It provide a good interface for communicating between ATC and Airline

**WEAKNESS**

* As it uses linked list it is less efficient compared to data base system.

**Siddaganga Institute of Technology, Tumakuru-3**

**Department of Computer Science and Engineering**

**Assessment for III semester Data structures lab Open Ended problem**

**Title: AIR TRAFFIC CONTROLLER**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Sl. No.** | **Name** | **USN** | **Evaluation Criteria** | | | | **Total**  **(25)** | | **Signature** | |
| **Complexity of problem chosen**  **(5)** | **Implementation**  **(10)** | **Coding Standards followed**  **(5)** | **Report**  **(5)** | |  | |  |
| **1** | **MRIGANK SRIVASTAVA** | **1SI17CS061** |  |  |  |  | |  | |  |
| **2** | **MRINAL RAJ** | **1SI17CS062** |  |  |  |  | |  | |  |
| **3** | **NITIKIA SINGH** | **1SI17CS070** |  |  |  |  | |  | |  |
| **4** | **NIKHIL SAH** | **1SI17CS067** |  |  |  |  | |  | |  |

**Signature of Faculty**