

CHAPTER 1

INTRODUCTION

Now a days everything is preferred doing online as it is easier and more convenient. So this Quick Movie Ticket App helps user to book ticket from anywhere at any time you can select any seat of your choice and pay online and you can get your ticket details instantly.

After the completion of my project user can choose between the following functionality:

- i. Edit price: only admin can perform this by giving a correct password and change the ticket price accordingly.
- ii. View reserved tickets: This function also needs password and only admin can access it admin can see all the book tickets using this option.
- iii. Purchase: This a open option for the user that does not requires a password using this option will allow user to purchase tickets from the available list inside this function user are given option to select seat and the user needs to enter details like name, number, and a seat number.
- iv. Cancel: User can delete the reserve ticket using this option user needs to give the booking ID to cancel the ticket.
- v. Exit: This option is used to quit the program.
- vi. Simple Movie Ticket Booking System is based on the concept of booking movie tickets. There's no login system available for this system, the user can freely use its feature. This mini project contains limited features, but the essential one.
- vii. Talking about the features of this Simple system, the user can book movie tickets. For this, the user has to select a movie name, enter customer details such as name and phone number. Then the user has to enter seat number. After this, the booking is done. The user can also change the ticket price and view reservations by entering the admin password. The last feature of this project is about canceling the tickets which can be done by entering the booking id.
- viii. The system does not create an external file to store the user's data permanently. This system is in C Programming Language and different variables, strings have been used for the development of it. Simple Movie Ticket Booking System in C Programming with source code is free to download. Use for educational purposes only! For the project demo, have a look at the video below.

Features:

- Booking tickets
- Cancel tickets
- Change ticket price
- View all booking records

CHAPTER 2

ALGORITHM

A programming algorithm is a computer procedure that is a lot like recipe (called a procedure) and tells your computer precisely what steps to take to solve a problem or reach a goal. The ingredients are called inputs, while the results are called the outputs. Algorithms are the building block of computer programs. They are as important to programming as recipes are to cooking.

STEP 1: Initialize the required global and local variable.

STEP 2: Create a structure for the entry details that user wants to take.

STEP 3: Create a dynamic memory allocation.

STEP 4: Use a switch case within a while to choose the options repeatedly.

STEP 5: Use appropriate loop and if statement for different function like reservation, changed price etc.

CHAPTER 3

DATA STRUCTURE

Pseudo code is a detailed yet readable description of what a computer program or algorithm. Must do, expressed in normal natural language rather than in programming language. Pseudo code is sometimes used as a detailed step in developing a program. It allows us to express the design in a great detailed and provides program a detailed view for the next step of writing code in specific programming language.

3.1 DATA STRUCTURES

Data structure deals with organization, management and storage of data that enables easy. Access and modification of data. Data structure is about collection of different data values, The relationship among the data and the function and operations that can be applied in that particular data.

Data Structures are of two types:

- Primitive data structures: This type of data structures is used in the representation of standard data types of computer language.
- Non primitive data structures: This type of data structures are that complicated data structures which are derived from the primitive data structures.

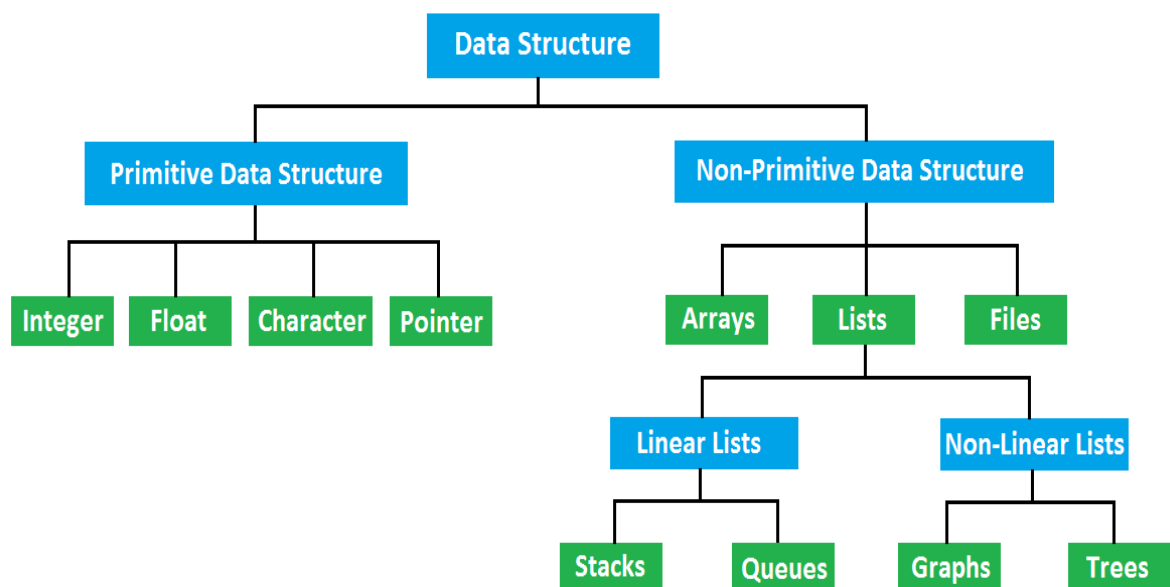


Fig: 3.1.1 Data Structure Tree

3.2 LINKED LIST

Linked lists are a linear collection of data of any type which are called nodes, where each node has its own value and points to the next node in the linked list. The main advantage of linked list is that values can always be efficiently inserted and removed without relocating the rest of the list.

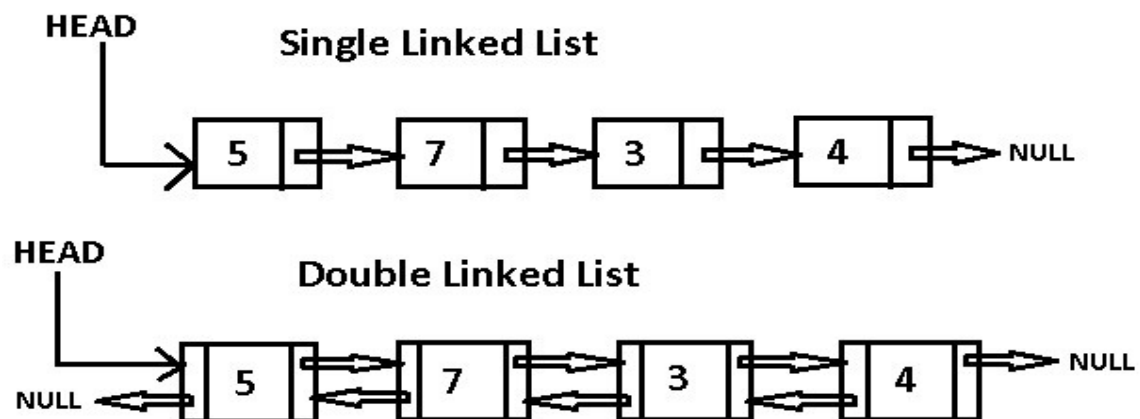


Fig 3.2.1 Structural Representation of linked list

A single linked list is a linked list which consists of a group of nodes where each node has 2 parts. One part is the data stored in the linked list and the other is the address of the next node. The address part of the last node always stores the value NULL. A pointer (Head in above figure) Points to the first node of the linked list and is used to access the linked list.

A double linked list is a linked list which consists of a group of nodes where each node has 3 parts. One part is the data stored in the linked list, the second part is the address of the next node and the third part is the address of the previous node. The next node points to NULL and similarly the previous address part of the first node points to NULL.

- **DOUBLY CIRCULAR LINKED LISTS:**

The last node points to the first node and the first node points to the last.

- Header Linked List (HLL)

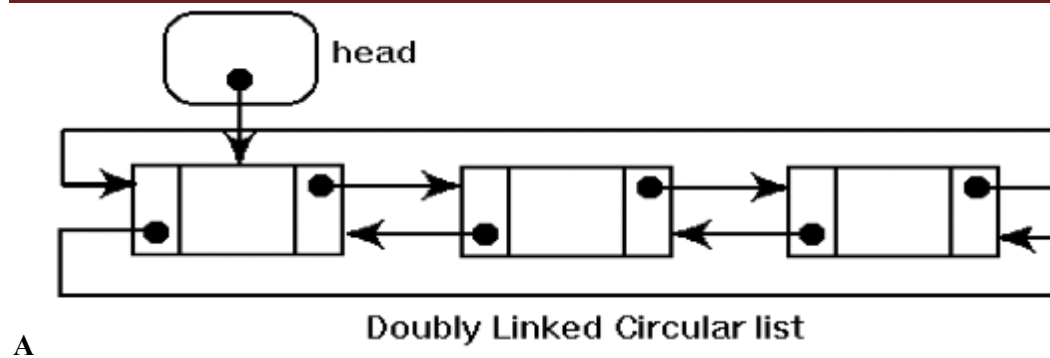


Fig 3.2.2 : Doubly Circular linked list

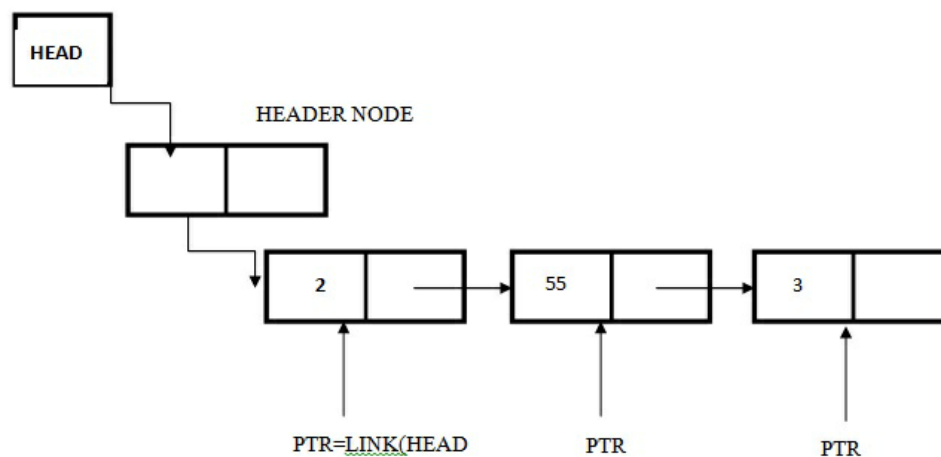


Fig 3.1.3 Header linked list

Header linked list is a linked list in which an extra node called the header node is present between the head node and the first node of the linked list. The header node does not need to store any data. It can be used as a reference node or be left blank. The header node can also contain the sum of all the elements stored in the linked list.

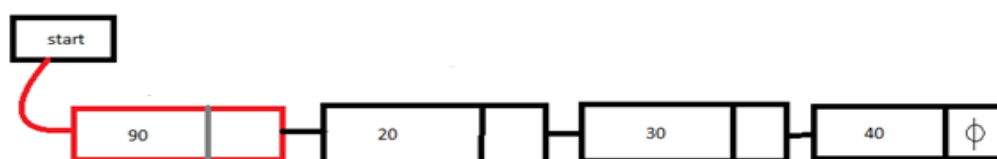


Fig 3.1.4 Header node (in red) contains 90 which is sum of all other elements (20,30 and 40)

• INSERTION

1. Insertion at beginning: A node can be added in the beginning of the List. In this, the head points to the new node and the new node is made to point to the node to which the address previously pointed.
2. Insert at end: In this, we traverse up to the last node and then the node at the end is made to point to the new node and the new node is pointed to NULL.
3. Insertion at kth position: In this we traverse up to the node at k-1 and make this node point to the new node and make the new node point to the node at which the (k-1) the node pointed.
4. Insertion before/after key: In this we traverse up to the node having the key value and according to the condition insert the new node before or after that node.

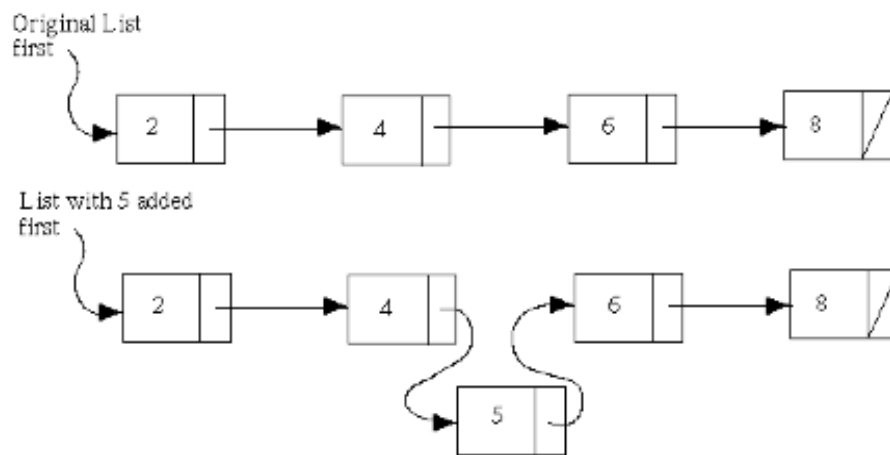


Fig 3.1.5 Insertion operation

• DELETION

1. Deletion at beginning: In this we delete the node to which the head points and head is now made to point the node at which the deleted node pointed.
2. Deletion at end: In this we traverse up to the second last node and make that node point to NULL and delete the last node i.e. the node to which this node points.
3. Deletion at kth position: In this we traverse up to the node at k-1 position and make it point to the node at k+1 position and delete the node to which the (k-1) the node previously pointed.
4. Deletion before/after key: In this we traverse up to the node having the key value and according to the condition delete the node before or after that node.

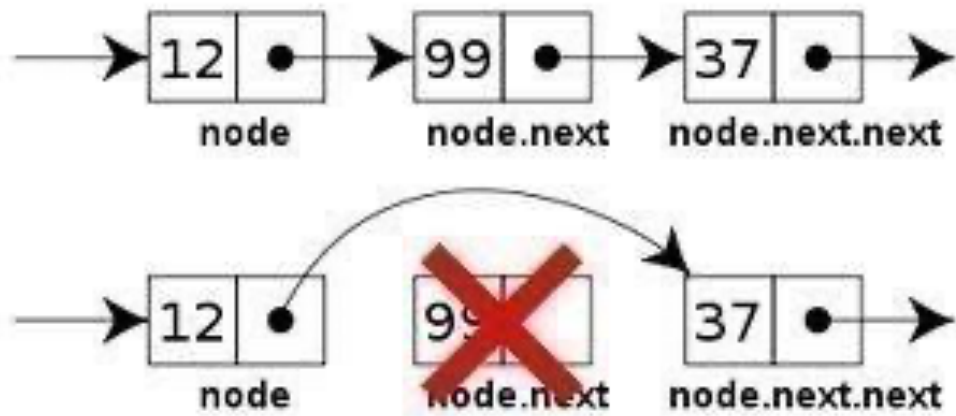


Fig 3.1.6 deletion operation

3.3APPLICATION OF LINKED LIST

- Implementation of stacks and queues.
- Implementation of graphs.
- Dynamic memory allocation.
- Maintaining directory of names.
- Performing arithmetic operations on long integers.
- Representing sparse matrix.
- Manipulation of polynomials.

Linked list is a dynamic data structure and it is also a linear data structure. The linked list consists of node which consists of two parts. The left side consists of the data and the right side of the node have a pointer which consists of the address of the next node. The right side of the last node will be empty or it will be null. If the first node that is the starting node is empty then it indicates that the node is empty.

Linked list provides an efficient way of sorting a related data and the basic operations lie create, insert, delete, traverse, search and update of information.

Why to use linked list over arrays?

When it comes to efficient memory management to make it consume less space in processor during executing the program and also to make it fast in basis of execution time. Arrays will have fixed memory allocation that cannot be changed during execution if the user inputs more than allocated memory, there is no flexibility in the program to expand and also excess variables are considered as overflow where they won't be

assigned any memory space. On the other hand, as linked lists memory is dynamical allocated (i.e. memory allocated during execution time). Which helps to go through the memory required for program to execute linked list will help to use the memory efficiently which is the most important and challenging task for programmer. Linked lists can also be user interactive.

As it is most efficient data structures user can allocate memory according to his need. There will be no restriction for memory allocation. Linked lists form the basis and many other data structures follow the same memory allocation. Linked list memory allocation is not contiguous. Sometimes, you may encounter a situation where one complete block or chunk of memory is not available, where you may have multiple number of chunks of memory. Using array this memory cannot be accessed, but linked list can efficiently be used. Linked list also opens a broad application, linked list are the building blocks for different non-linear data structures like TREES, GRAPHS, etc.

Disadvantages of using linked list: One disadvantage of linked list over an array is that it does not allow direct access to individual elements. If a link of node in between the linked list accidentally deleted/lost then all the elements after that node.

SINGLE LINKED LIST VERSUS DOUBLE LINKED LIST

SINGLE LINKED LIST	DOUBLE LINKED LIST
A linked list that contains nodes which have a data field and a next field which points to the next node in the line of nodes	A linked list that contains the data field, next field that points to the next node and a previous field that points to the previous node in the sequence
Allows traversing in one direction through the elements	Allows traversing in both directions (backward and forward)
Requires less memory as it stores only one address	Requires more memory as it stores two address
Complexity of insertion and deletion at a known position is $O(n)$	Complexity of insertion and deletion at a known position is $O(1)$
	Visit www.PEDIAA.com

3.4 TREES

Tree is non-linear data structure which stores information naturally in the form of hierarchy style. It represents nodes connected by edges. We know that a linked list is a chain of nodes which connect through next pointers, similar is tree but with a slight difference as in tree each node can be connected to multiple nodes.

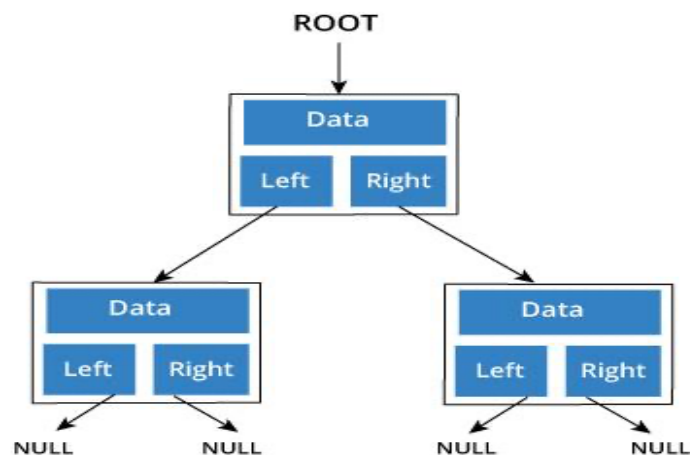


Fig 3.4.1: Binary Tree Representation

There is a special pointer called root which points to the node that is the parent of all the other nodes. The nodes that don't have any children have their right and left pointers point to NULL.

3.5 QUEUES

It is linear type of data structure. It follows the rule "First in First out". It is similar to stack except the removing part. In the queue only two operations are allowed enqueue and dequeue. Enqueue means to insert an item into the back of the queue, dequeue means removing the front item.



Fig 3.5.1 Representation of queue

3.6 STACKS

It is linear data structure type. Stack is a basic data structure that can be logically thought of as a linear structure represented by a real physical stack or pile, a structure where insertion and deletion of items takes place at one end called top of the stack. It follows the “First in last out” rule.

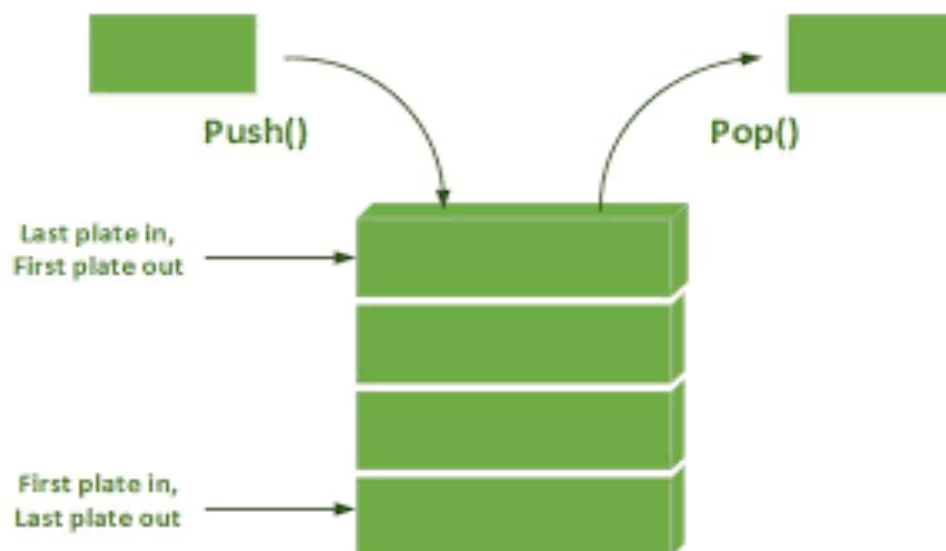


Fig 3.6.1 Representation of stack

3.7 GRAPHS

A graph is a non-linear data structure which consists of nodes and edges. Also represented as $G = (V, E)$ where V is a set of vertices and E is set of edges. Graphs are used to represent networks. There are two ways in which a graph can be represented: -

- Sequential or Array representation
- Linked Representation

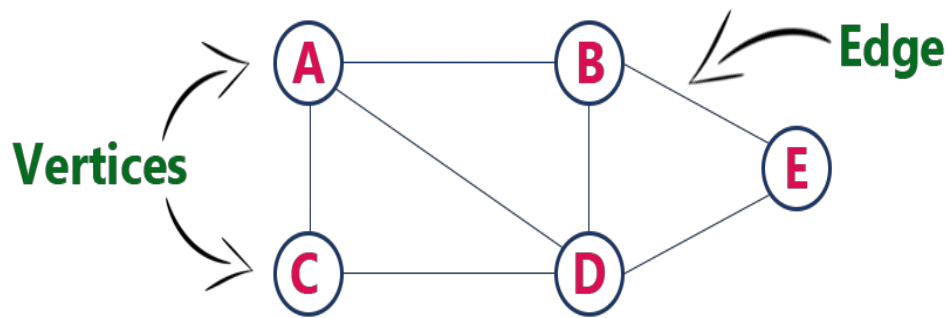


Fig 3.7.1 : Representation of graphs

CHAPTER 4

COMPLETE PROGRAM

```
#include <stdio.h>
#include<stdlib.h>
#include<strings.h>
struct moviedetails{
    char name[25];
    char phone[15];
    int seat;
    int id;
};
struct moviedetails person[300];
int count=0;
int id2=1000;
int changeorize(int);
void reservation(int *,int,int );
int choice1(void);
void cancel(int *);
void ticket1(int choice,char name[10],int id2,int price);
void ticket2(int choice,char name[10],int id2,int price);
void ticket3(int choice,char name[10],int id2,int price);
int cmovie(void);
int movie(void);
void details(void);
int main()
{
    int **seat,choice,price=500,slection,i;
    seat=(int **)calloc(101,sizeof(int *));
```

Quick Movie Tickets

```
for (i=0;i<3;i++)

    *(seat+i)=(int *)calloc(101,sizeof(int ));

int x;
while(x!=5)
{
    choice=choice1();
    switch(choice)
    {
        case 1:
            price=changeprize(price);
            break;
        case 2:
            details();
            break;
        case 3:
            slection=movie();
            reservation(seat[slection-1],price,slection);
            count++;
            break;
        case 4:
            slection=cmovie();
            cancel(seat[slection-1]);
            break;
        case 5:
            x=5;
            break;
        default:
            printf("Choice not available\n");
            break;
    }
}
```

Quick Movie Tickets

```
}

int changeprize(int prize)
{
    char pass[10],pak[10]="pass";
    printf("Enter the password to change price of ticket: ");
    scanf("%s",&pass);
    if (strcmp(pass,pak)==0)
    {
        printf("Please enter new price: ");
        scanf("%d",&prize);
        system("PAUSE");
        system("CLS");
    }
    else
        printf("The entered password is wrong! ");
    return prize;
}

void reservation(int *array,int price,int slection)
{
    int i,j;
    printf("\n          SCREEN\n\n\n");
    for (i=1;i<=100;i++)
    {
        if (array[i]==0)
            printf("%d\t",i);
        else
            printf("*\t",i);
        if(i%10==0)
            printf("\n\n");
    }
    printf("Please enter your name: ");
```


Quick Movie Tickets

```
scanf("%19[^\n]%*^[^\n]", &person[count].name);
printf("Please enter your phone number: ");
scanf("%u", &person[count].phone);
printf("Which seat number you want? ");
scanf("%d", &j);
if (j>100 || j<1)
{
    printf("seat1 number is unavailable in this theater\n");
    printf("Please re-enter seat number: ");
    scanf("%d", &j);
}
if (array[j]==1)
{
    printf("Sorry, this ticket is already booked! Please choose
another seat.\n");
    scanf("%d", &j);
}
else
    array[j]=1;
person[count].seat=j;
if (slection==1)
    ticket1(j, person[count].name, id2, price);
else if (slection==2)
    ticket2(j, person[count].name, id2, price);
else
    ticket3(j, person[count].name, id2, price);
id2++;

}

int choice1(void)
{
```

Quick Movie Tickets

```
int choice;

printf("      Simple Movie Ticket Booking System\n");

printf("
=====\\n");

printf("||      1- To edit price of ticket (only admin):      ||\\n");
printf("||      2- To view reserved tickets (only admin):      ||\\n");
printf("||      3- To purchase ticket:                          ||\\n");
printf("||      4- To cancel the seat:                            ||\\n");
printf("||      5- Exit system:                                    ||\\n");
printf("||=====\\n");
||\\n");

printf(" Enter your choice: ");
scanf("%d",&choice);

return choice;
}

void cancel(int *array)
{
    int Cseat,i,stop;

    printf("Please enter ID number of ticket: ");
    scanf("%d",&Cseat);
    for (i=0;i<300;i++)
    {
        if(Cseat==person[i].id)
        {
            stop=5;
            system("cls");

            printf("%s your seat is %d
cancelled",person[i].name,person[i].seat);

            array[person[i].seat]=0;

            i=300;
        }
    }
}
```

Dept of CSE, NHCE	20
-------------------	----


Quick Movie Tickets

```
    person[count].id=id2;
    printf("\t=====\\n");
    return;
}
```

CHAPTER 5

RESULTS

- Choice in the menu

 C:\Users\MY PC\Desktop\atanumini1.exe

```
Simple Movie Ticket Booking System
=====
| 1- To edit price of ticket (only admin): |
| 2- To view reserved tickets (only admin): |
| 3- To purchase ticket: |
| 4- To cancel the seat: |
| 5- Exit system: |
=====
Enter your choice:
```

Fig5.1 : Menu for the application

- Reserving of the ticket

C:\Users\MY PC\Desktop\atanumini1.exe

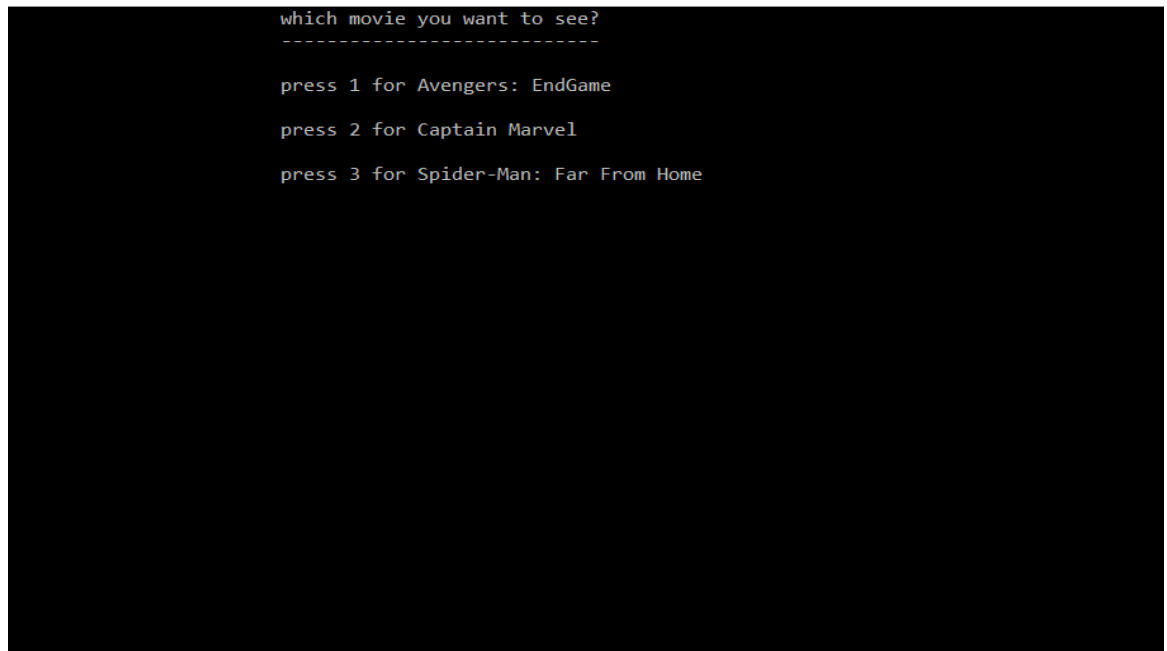


Fig 5.2: List of movies

C:\Users\MY PC\Desktop\atanumini1.exe

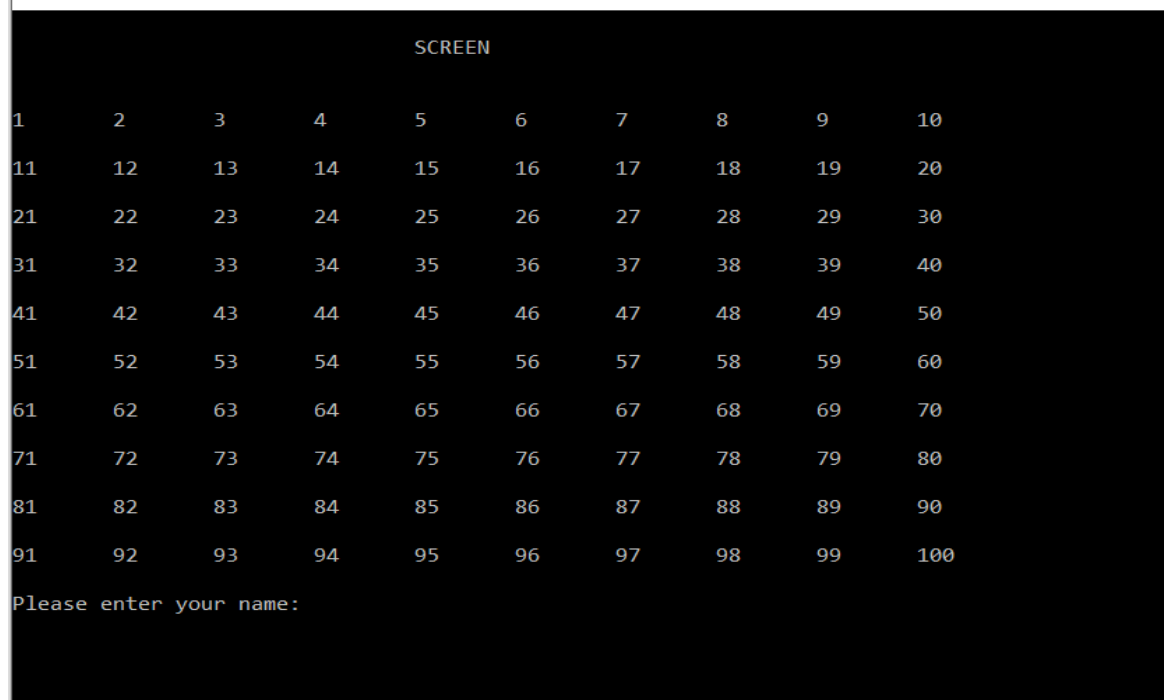


Fig 5.3: Seats


```
C:\Users\MY PC\Desktop\atanumini1.exe

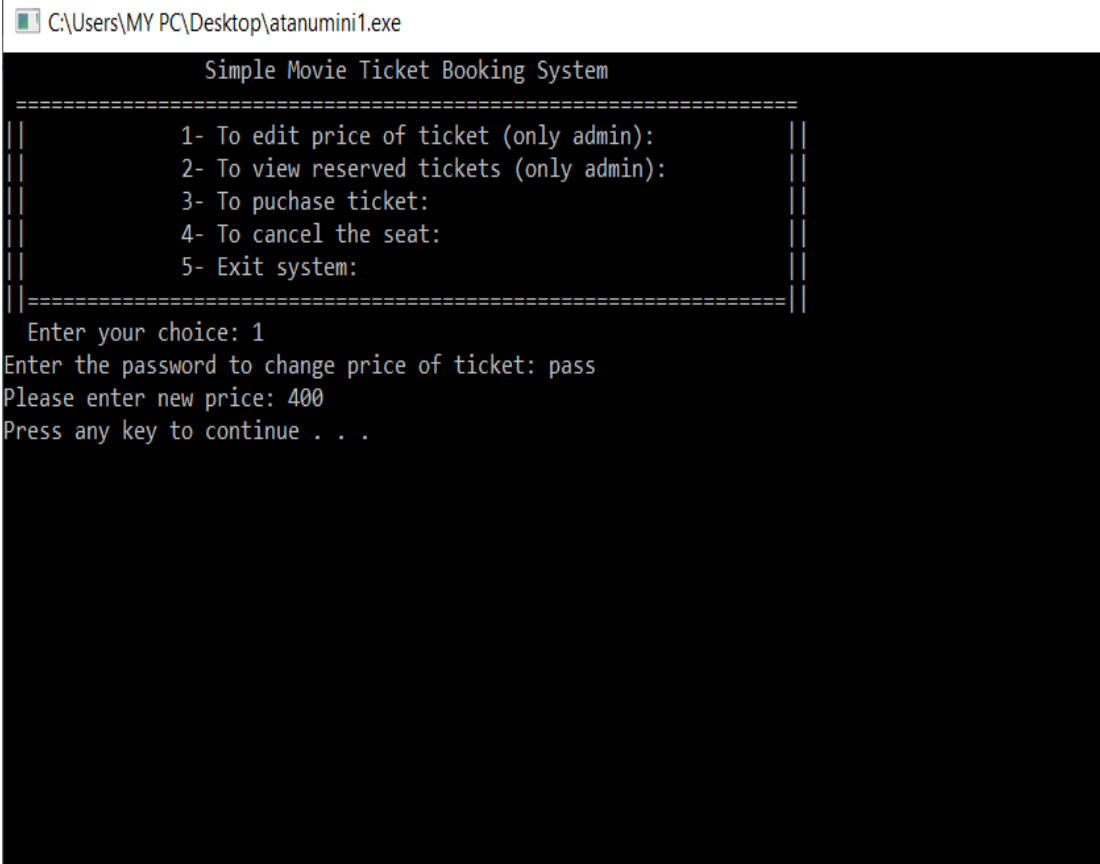
-----THEATER BOOKING TICKET-----
=====
Booking ID : 1000                      Show Name : Avengers: EndGame
Customer  : Atanu

                                   Date      : 29-04-2019
                                   Time       : 08:00pm
                                   Hall       : 02
                                   seats No.  : 45
                                   price .   : 500

=====
Simple Movie Ticket Booking System
=====
|| 1- To edit price of ticket (only admin): ||
|| 2- To view reserved tickets (only admin): ||
|| 3- To purchase ticket: ||
|| 4- To cancel the seat: ||
|| 5- Exit system: ||
||=====||
Enter your choice:
```

Fig 5.4: Reserved

- Changing of price

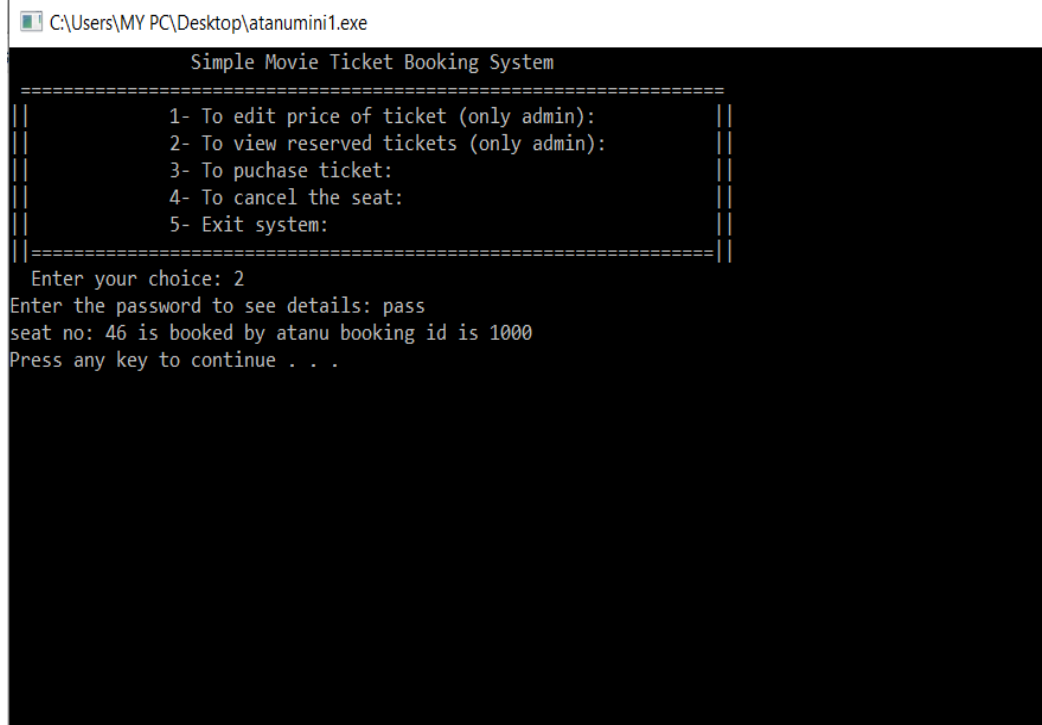


```
C:\Users\MY PC\Desktop>atanumini1.exe

Simple Movie Ticket Booking System
=====
||      1- To edit price of ticket (only admin):      ||
||      2- To view reserved tickets (only admin):     ||
||      3- To purchase ticket:                       ||
||      4- To cancel the seat:                       ||
||      5- Exit system:                               ||
||=====||
Enter your choice: 1
Enter the password to change price of ticket: pass
Please enter new price: 400
Press any key to continue . . .
```

Fig 5.5: Successfully Changed

- Viewing the reserved tickets

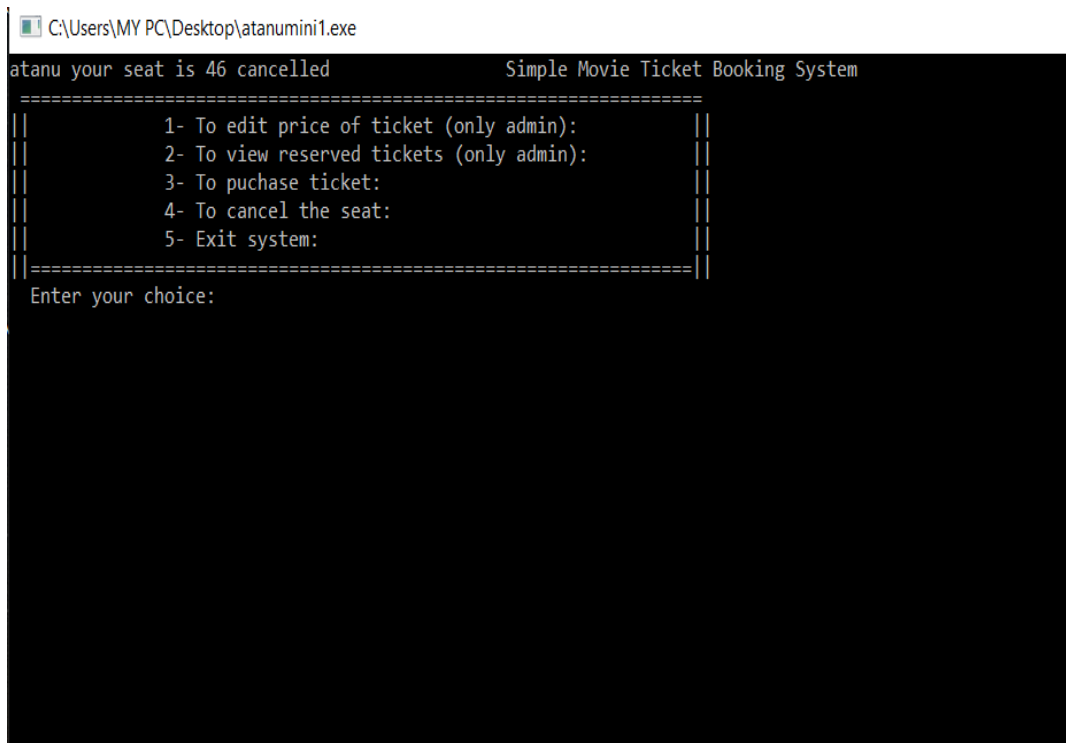


```
C:\Users\MY PC\Desktop\atanumini1.exe

Simple Movie Ticket Booking System
=====
||      1- To edit price of ticket (only admin):      ||
||      2- To view reserved tickets (only admin):     ||
||      3- To purchase ticket:                       ||
||      4- To cancel the seat:                       ||
||      5- Exit system:                               ||
||=====||
Enter your choice: 2
Enter the password to see details: pass
seat no: 46 is booked by atanu booking id is 1000
Press any key to continue . . .
```

Fig 5.6: List of tickets booked

- Cancellation of ticket



```
C:\Users\MY PC\Desktop\atanumini1.exe
atanu your seat is 46 cancelled          Simple Movie Ticket Booking System
=====
||      1- To edit price of ticket (only admin):      ||
||      2- To view reserved tickets (only admin):    ||
||      3- To purchase ticket:                       ||
||      4- To cancel the seat:                       ||
||      5- Exit system:                              ||
||=====||
Enter your choice:
```

Fig 5.7: Successfully cancelled

CHAPTER 6

CONCLUSION

We can conclude by saying that we can meet the objectives of our mini project by using the data structures in which we can access and store data. By doing mini project on this topic I have learnt a lot about data structures in C and also improved my coding knowledge but data structure concept plays a core role in implementing the logics the functionality of the program is data base is necessary to effectively store the data and access it in less lag time.

As the present technology asks for faster accessing of data in both storing as well as retracing back, there is high scope for data structure and also for new efficient logics.

From this mini project I am able to understand the need of data structure in the modern day, where we need to find a logic and code a program for real time problems. It also helped to demonstrate the a part of data structure as well as few concepts of C Programming Language. Being a Computer Science Engineer it is just to know the concepts of data structures, but it is to understand the depth of the program coded and also to understand the background logics and crack down the new advanced logic or technique which perform same functionality with additional advanced outcome step by step logic of the code with necessary result flowing to next level of code has to be understood to find the best of all.

Reference

- Geeks for Geeks • Quora
- Wikipedia
- Stack flow