Assignment no. 07. 21118 E-1 (SE-1)

Date 10-Nov-2020

Problem Statement:
The ticket booking system of Cinemax Theatre has to be implemented using C++ program. There are 10 rows & 7 columns. Doubly circular linked list has to be maintained to keep track of the free seats at: rows. Assume some random booking to start with. Use array to store pointers (head pointers) to each row.
On demand
a) The list of available seats is to be displayed.
b) The seats are to be booked.
c) The booking can be cancelled.

Objectives:
a) To understand the concept of OOP paradigm.
b) To understand usage of linked-list in creation of dynamic lists in C++.

Outcome:
a) To implement basic system design functionality using OOP paradigm
b) To write menu driven, modular program in C++ using classes & objects.
c) To implement various dependent & independent classes in C++.

Hardware Requirement:
Manufacturer & Model: Acer Swift-3
Processor: Intel Core i5 - 8th gen (8265U @ 1.8GHz)
Installed Memory: 8GB RAM, 512GB SSD
Architecture: 64 bit

Software Requirement:

Operating System: Ubuntu 20.04 on Oracle Virtual Machine
(Base Memory is 4096 MB & 3 processes are allocated)
Cpp version used: C++14
Compiler: for C++: g++ (Version 10.1.0)
Code editor: Sublime Text (Build 3211)

Theory:

Doubly linked list: (DLL):
A linked list in which every node points to its next node as well as previous node
The adv. of DLL over singly linked list is that it provides previous node access in constant time. But this also require using extra memory for previous pointers.

Circular Linked list: (CLL)
In circular linked list all nodes are connected in such a way they form a circle.
The advantage of CLL is that we can start from any node & we can traverse the whole list. This linked list is very useful in creation of some data structures like queue.
It is also used in implementation of fibonacci Heap.

Pseudo code

Operations of Circular Doubly Linked list:

Algorithm addNode (int data) { // add node at beginning
newnode ← getNewNode (data);
if (head == NULL) head ← newNode.
else { newNode → next ← head
newNode → prev ← head → prev

```
        head → prev → next ← newNode
        head → prev ← newNode;
    }
}
```

```
Algorithm deleteNode(ref data) {
    if (head → data == data) {
        temp = head
        head = head → next
        head → prev = temp → prev
        temp → prev → next = head
    }
    else {
        temp = head
        while (temp → data != data) temp = temp → next
        temp → next → prev = temp → prev
        temp → prev → next = temp → next
    }
    delete temp
}
```

```
Algorithm isPresent(ref data) {
    temp ← head
    do {
        if (temp → data == data) return true
        temp ← temp → next
    } while (temp != head)
    return false
}
```

APT of classes:

The classes used in program are:

```
class Seat {
  private:
    int seat_no;
    bool is_Booked;
    string pers_name;
    Seat *next, *prev;
  public
     Seat (int seat_no, bool isbooked, string pers_name) {
     } // constructor
}
```

```
class TheatreRow {
  private:
    Seat *head;
    int total_seats;
  public:
     TheatreRow() {    // constructor
     {
     void addSeat() {    // add seat in row
     {
      bool updateSeatDet() {    // update seat detail
     }
     void PrintSeatDetail (int seat_no) {    // print detail of seat.
     {
     void PrintBookings() {    // print entire row.
     {
}
```

```
class Theatre {
  private:
    int row, columns;
    TheatreRow* theatre_rows;
```

```
public:
    Theatre() {  // constructor function
    }
    void bookSeat(int r, int c, string passa_name){
    }
    void CancelBooking(int r, int c){
    }
    void PrintSeatData(int r, int c){
    }
    void printBookingStatus(){  // prints booking status of
                                   entire theatre.
    }
}
```

Analysis of Algorithms

① Book seat in a theatre.
The algorithm will take time. $O(m \cdot n)$ in worst case where m = rows & n = columns in the theatre.

② Cancel Booking
The algorithm will take $O(m \cdot n)$ in worst case where m = rows & n = columns in the theatre.

③ The list of available seats in theatre
The algorithm will take $O(m \cdot n)$ time.

Test Cases:

No.	Test Case Description	Expected output	Program o/p
1	Booking seat in 5th row 2nd col.	Seat should be booked successful.	"Seat is booked successfully"
2	Cancelling a seat booked in testcase-1	Seat should be cancelled successfully.	"Booking is cancelled successfully"

3)	printing status of theatre.	Program should print booking status of theatre	Program shows booking status of entire theatre Successfully
4)	Showing details of seat booked in testcase-1	Program should print details of seat	Row:5 Col:2 Name: Michael
5)	Showing details of unbooked seat	There should be error message.	--The seat is not booked--
6)	Booking booked seat in testcase-1	There should be message informing that seat is already booked.	"seat is already occupied. Please check other available options"

Applications:

1) Linked list are used in implementation of other data structures like fibonacci heap, queues.
2) They are also used in implementation of cache memory.

Conclusion:

Linked list provides upper edge in case of memory usage compared to arrays. The assignment focuses on linked list implementation ~~also~~ along with basics of system design using oop concepts which is very useful according to todays industry standards.