Hotel Simulator Design Document

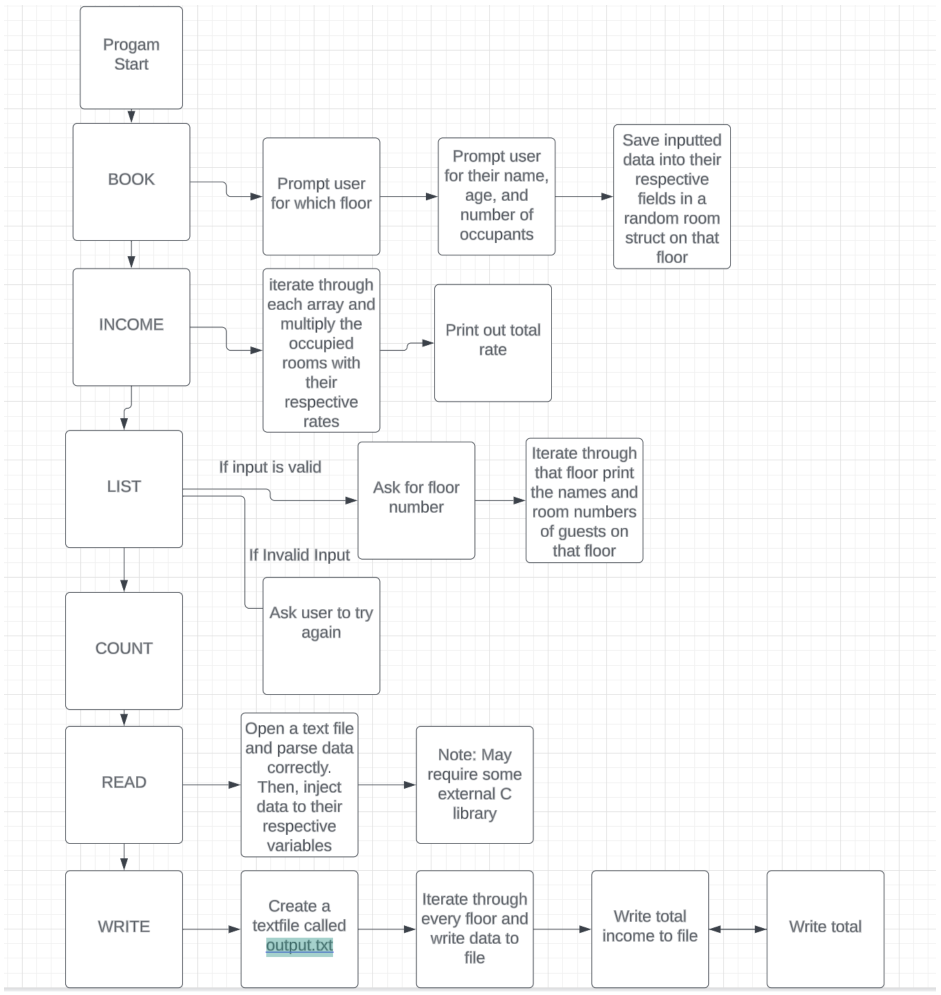
Author: Raed Kabir
GitHub: https://github.com/Reptop

Program Overview

This program is intended to be a simulator for a hotel. This entails that this program keeps track of the number of available rooms on each floor, the price for each room, the nightly income, and the total number of occupants in the hotel. This program will implement the use of structs as a primary method of translating what a room is into C code. The program will have the ability to load and save the hotel simulator data as well. This will require the program to be able to parse data and store the data in their respective variables.

Picture

Below is a flowchart showing how the program will function given a specific command. The flowchart mainly acts as a visual aid when coding the program.



Progam Start
BOOK
Prompt user for which floor
Prompt user for their name, age, and number of occupants
Save inputted data into their respective fields in a random room struct on that floor
INCOME
iterate through each array and multiply the occupied rooms with their respective rates
Print out total rate
LIST
If input is valid
Ask for floor number
Iterate through that floor print the names and room numbers of guests on that floor
If Invalid Input
Ask user to try again
COUNT
READ
Open a text file and parse data correctly. Then, inject data to their respective variables
Note: May require some external C library
WRITE
Create a textfile called output.txt
Iterate through every floor and write data to file
Write total income to file
Write total

Inputs
The inputs of this program will be in the form of commands by user input; the commands: BOOK, INCOME, LIST, COUNT, READ, and WRITE will govern the program's functionality. Through these commands, the user can "BOOK" a new room, for example. After inputting the command "BOOK," the program will then continue to ask the user for more information so a new room can be created. If the program discovers an invalid input, the program will prompt the user to try again. INCOME prints out the total income for a given night; LIST prints out a list of all rooms and their occupants on a given floor. COUNT prints out the total occupants on every floor. Lastly, READ and WRITE either take in data or write data via a text file. Each command essentially progresses to a series of instructions that lead to the command being fulfilled.

Outputs
The outputs of this program will largely be numerical data. For example, numerical data includes the profits information regarding an occupant - most notably age and the number of occupants within a certain age range. Also, information on the room number and the number of occupants are numerical data as well. Other outputs would be an occupant's name and a text file of all information recorded through this program.

Test Plan

Below is a test plan of what the "BOOK" command would look like in this program.

> **INPUT**
> "BOOK"
>
> **OUTPUT**
> "Which floor would like to book a room"
>
> **INPUT**
> "4"
>
> **OUTPUT**
> "There are only 3 floors in total. Try again."
>
> **INPUT**
> "3"
>
> **OUTPUT**
> (This instruction assigns a random index for a room on the third floor)

"You have booked a random room on floor 3. The rate is $100/night; to finish the booking process, enter your first and last name and the number of occupants you will bring."

INPUT

Continue asking the user for information and storing the information in the variables within the room structs

Approach

Below is my outlined approach through pseudo-code.

```
//or any form of infinite loop
for(;;) {

    If command equals BOOK {
        Ask for which floor;
        Randomly assign room on a given floor;
        Prompt user for more information needed to complete room info;
    }

    Else if command equals INCOME {
        Iterate through each floor array of structs;
        Count the occupied rooms;
        Multiply the respective rates by each respective room on a given floor;
        Find the total of all the profits by finding the sum of all the rates;
    }

    Else if command equals LIST
        Iterate through every floor;
        Print out the information of every room on each floor in a formatted manner;

    Else if command equals COUNT {
        Iterate through every floor's room;
        Fetch the number of occupants in each room struct;
        Add each occupant number in every room to a total occupants variable;
        print out the variable value
    }
```

```
        Else if command equals READ {
                Parse data correctly from a text file;
                Insert data into their respective fields in each struct array;
                Close text file;
        }

        Else if command equals WRITE {
                Write all hotel information to a text file;
                Close text file;
        }

        Else if command equals QUIT {
                break out of infinite loop;
                Print out “See ya!”;
        }
}
```