Data Structures and Algorithms

University of Madeira

Project no 1

Operation of the Hotel EDA

<u>Students Group no 25</u>

- ***Mohapi Matsatsa France:2010917***
- ***Pule Samuel Nhlapo:2010817***
- ***Daniel Czarnozynski:2091617***

Data Specifications

Data specification defines primary data objects, composition of each data object, and attributes of the object, relationships between each object and other objects and between objects and the processes.

Guest

	Types
Names	String
Reservation ID	Int
Nationality	String
Duration of reservation	Int
Family Name	String

Room

	Types
Number	Int
Capacity	Int

Reception

	Types
Reception	Struct

Functionalities

Hotel Logic

The main objective of the operation of the hotel is to implement a system of all the basic activities of a hotel so that the system can be used by external entities.

1.Guest

This function is aimed to display all the guest details such as names, unique reservation ID, duration which is the equivalence of the time they will spend at the hotel, their nationality, family names and the day of the arrival.

2.Room

This function is used to add new guests details, erase entity details and view the details. In that screen, the automatic item is created. It is also used to check out the guest details from database. When the user checks unoccupied room, the same room number will be checked in the database, if the room number is matched in the database, then the guest will be checked-out from the database and transferred the record of the checkout to another table of database so that the Hotel has the record of guests who have checked-out.

In this function, whenever a new entity is required to be added the corresponding forms are opened and the database is manipulated to check whether the data is already existing or not. If it already exists, then it prompts that “THE ROOM IS FULL” and if not than the data is entered with the various validation checks.

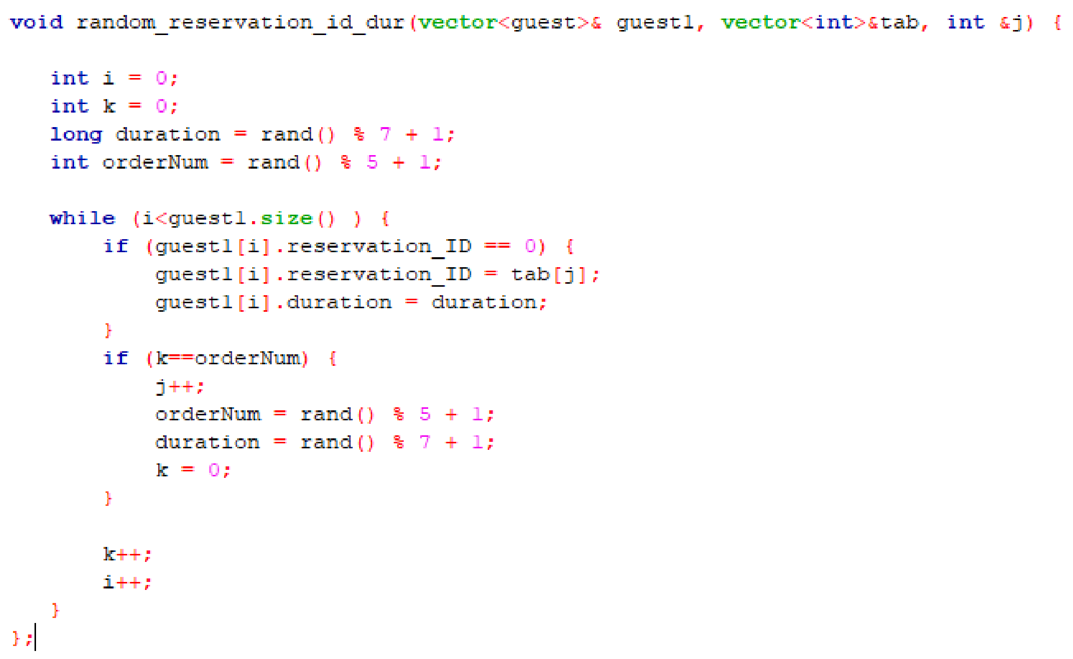
3.Reception

This function is used to display new guest that are put at reception. It is also used to admit a customer in the Hotel after having their details checked. Personal details like Name, Family Name, Reservation ID and Nationality.

Implementation

The hotel consists of a random amount of rooms, between 1 and 15.Each room has a maximum capacity of 5 guests and a minimum of 1. The rooms are created before the first cycle and they are not changed during the execution.

3.3.1 Reservation ID (6 digit integer) implementation.

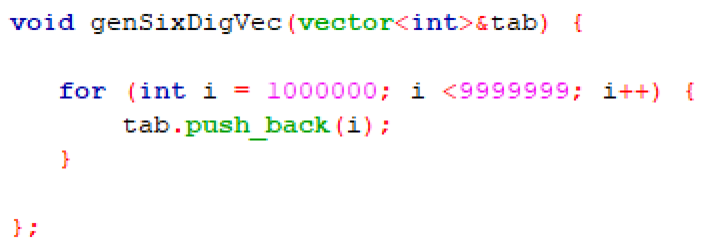


```
void random_reservation_id_dur(vector<guest>& guestl, vector<int>&tab, int &j) {

   int i = 0;
   int k = 0;
   long duration = rand() % 7 + 1;
   int orderNum = rand() % 5 + 1;

   while (i<guestl.size() ) {
       if (guestl[i].reservation_ID == 0) {
           guestl[i].reservation_ID = tab[j];
           guestl[i].duration = duration;
       }
       if (k==orderNum) {
           j++;
           orderNum = rand() % 5 + 1;
           duration = rand() % 7 + 1;
           k = 0;
       }

       k++;
       i++;
   }
};
```



```
void genSixDigVec(vector<int>&tab) {

    for (int i = 1000000; i <9999999; i++) {
        tab.push_back(i);
    }

};
```

3.3.1 Duration of the reservation implementation

Duration of the reservation comprises of a random value between 1 and 10.

```
void random_duration(vector<guest>& guestl) {

   int i = 0;
   long duration;

   while (i < guestl.size()) {
       duration = rand() % 7 + 1;
       if (guestl[i].duration == 0) {
           guestl[i].duration = duration;
           i++;
       }
   }
}
```

3.3.3 Data ID and Duration

In this implementation data id and duration are shared by 1 to 8 guests.

```
void random_reservation_id_dur(vector<guest>& guestl, vector<int>&tab, int &j) {

    int i = 0;
    int k = 0;
    long duration = rand() % 7 + 1;
    int orderNum = rand() % 5 + 1;

    while (i<guestl.size() ) {
        if (guestl[i].reservation_ID == 0) {
            guestl[i].reservation_ID = tab[j];
            guestl[i].duration = duration;
        }
        if (k==orderNum) {
            j++;
            orderNum = rand() % 5 + 1;
            duration = rand() % 7 + 1;
            k = 0;
        }

        k++;
        i++;
    }
};
```

3.3.4 Creation of guests

The system uses three external files named:

name.txt
nationality.txt
family.txt

Which contain names, nationalities, and family names separated by "enters". The system select data randomly from these files and create a guest.

(a) Name implementation

```
void random_names(ifstream &FileNam, vector<guest> &guestl, int size) {

    int lineNum;
    long random;
    int i = 0;
    string line;

    while (i< size) {


        random = rand() % 4000 + 1;

        lineNum = 0;

        FileNam.open("names.txt");
        while (getline(FileNam, line)) {

            if (lineNum == random &&  guestl[i].name == "") {
                guestl[i].name = line;
            }

            lineNum++;
        }
        FileNam.close();

        i++;
    }
};
```

(b) Nationality Implementation

```
void random_nationalities(ifstream &FileNat, vector<guest> &guestl, int size) {

   long random;
   int i = 0;
   int lineNum;
   string line;

   while (i<size) {
       random = rand() % 200 + 1;

       lineNum = 0;

       FileNat.open("nationalities.txt");

       while (getline(FileNat, line)) {

           if (lineNum == random && guestl[i].nationality == "") {
               guestl[i].nationality = line;

           }

           lineNum++;

       }
       FileNat.close();

       i++;
   }
};
```

(c) Family Names implementation

```
void random_fullnames(ifstream &FileFam, vector<guest>& guestl, int size) {

   long random;
   int lineNum;
   int i = 0;
   string line;

   while (i<size) {

       random = rand() % 505 + 1;

       lineNum = 0;

       FileFam.open("family.txt");
       while (getline(FileFam, line)) {

           if (lineNum == random && guestl[i].familyName == "") {
               guestl[i].familyName = line;
           }

           lineNum++;

       }
       FileFam.close();

       i++;
   }
};
```

(d) This implementation presents guests that are always to the reception and where they await for a new cycle so that they can be placed in the rooms depending on the availability of rooms.

```
void putGuestReception(vector<guest> &guestl, reception &reception) {

    for (int i = 0; i < reception.family.size(); i++) {
        reception.family[i] = guestl[i].familyName;
    }
};
```

(e) This function presents implementations of rooms that are created in each cycle when the number of rooms are divided by 2.

```
int make_rooms(room rooms[]) {

    int sumOfRooms = 0;

    for (int i = 0; i<15; i++) {
        rooms[i].number = i;
        rooms[i].capacity = rand() % 5 + 1;
    }

    for (int i = 0; i<15; i++) {
        sumOfRooms = sumOfRooms + rooms[i].capacity;
    }

    return sumOfRooms;
};
```