Hotel Management System

Complex Computing Project (CT-175)

Team: Innovators

Marium Faheem (CT-168)

Memoona (CT-166)

Muhammad Bakhtiyar (CT-189)

Instructor: Sir Muhammad Abdullah

Presentation Agenda

A quick overview of our project, from the problem to the solution and its future.

Our Roadmap

- **The Big Picture:** Introduction & Problem Statement
- **The Blueprint:** System Architecture
- **The Core:** Data Structure (`struct Room`)
- **The Engine:** Code Deep Dive (Our Main Focus)
- **The Proof:** Testing & Results
- **The Future:** Conclusion & Future Work

Introduction

What is it?

A simple and efficient **console application in C** designed for small-scale hotels and hostels.

Why did we build it?

To replace inefficient, slow, and error-prone manual record-keeping.

What's the goal?

To streamline and automate key hotel operations:

- Room Bookings
- Guest Check-in & Check-out

The Problem with Manual Systems

Data Duplication

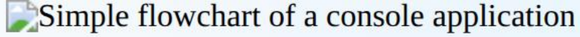
Paper-based records are prone to duplication, leading to confusion and double-bookings.

Billing Errors

Manual calculations for bills and rates can lead to frequent and costly mistakes.

Slow Service

Staff can't check room availability in real-time, making check-in a slow process.



System Architecture

The Core Data

The entire system runs on a **50-element array** of `struct Room`. This array acts as our simple, in-memory database.

Functional Modules

The program is built on four key functions that interact with this array:

- `main()`: Initializes the data and runs the main menu loop.
- `displayRoomDetails()`: **Reads** from the array to show room status.
- `bookRoom()`: **Writes** to the array to mark a room as "Occupied".
- `checkOut()`: **Writes** to the array to mark a room as "Available".

Code Deep Dive

Let's walk through the C code, function by function.

Code: The Core Data Structure

```
struct Room {
    int room_number;
    char room_type[50];
    int capacity;
    float rate;
    char status[50];
    char guest_name[50];
};
```

Explanation

- This `struct` is the **heart of our system.** It's the blueprint for all our data.
- An array of 50 of these structs holds all hotel data.
- `status`: The most critical field. It tracks if a room is **"Available"** or **"Occupied"**.
- `guest_name`: Stores the current guest's name. It's empty if the room is available.

Code: Initialization (in `main()`)

```
int main() {
    int TOTAL_ROOMS = 50;
    struct Room rooms[TOTAL_ROOMS];

    // Initialize rooms with default values
    for (int i = 0; i < TOTAL_ROOMS; i++) {
        rooms[i].room_number = i + 1;

        if (i <= 9) {
            strcpy(rooms[i].room_type, "Standard");
            rooms[i].rate = 800.0;
            rooms[i].capacity = 2;
        } else if (i <= 19) {
            // ... "Deluxe" ...
        }

        strcpy(rooms[i].status, "Available");
    }
    // ... main menu loop follows ...
}
```

Explanation

- The `main()` function creates the 50-room array.
- A `for` loop iterates from 0 to 49 to initialize each room's data.
- `if-else if` statements are used to set different `room_type` and `rate`.
- **Crucially**, `strcpy(rooms[i].status, "Available");` sets every room to be free at the start.

Code: The Main Menu (In `main()`)

```
int choice;
while (1) {
    displayMenu();
    printf("Enter Your Choice: ");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            displayRoomDetails(rooms, TOTAL_ROOMS);
            break;
        case 2:
            bookRoom(rooms, TOTAL_ROOMS);
            break;
        case 3:
            checkOut(rooms, TOTAL_ROOMS);
            break;
        case 4:
            printf("...Thanks for visiting...");
            return 0; // Exits the program
        default:
            printf("Invalid Choice!\n");
            break;
    }
}
```

Explanation

- An **infinite `while(1)` loop** keeps the program running.
- The `displayMenu()` function prints the options.
- A `switch` statement reads the user's `choice`.
- Each `case` calls the correct function (e.g., `bookRoom`).
- The **`rooms` array is passed** to each function so they can read/modify the data.

Code: bookRoom() Logic

```
void bookRoom(struct Room rooms[], int totalRooms) {
    // ... (scanf to get roomNumber, capacity, guestName) ...

    for (int i = 0; i < totalRooms; i++) {

        // This is the core booking logic
        if (rooms[i].room_number == roomNumber &&
            strcmp(rooms[i].status, "Available") == 0 &&
            rooms[i].capacity >= capacity)
        {
            // ... (Confirm booking) ...

            strcpy(rooms[i].status, "Occupied");
            strcpy(rooms[i].guest_name, guestName);

            printf("Room %d Booked Successfully!\n", roomNumber);
            return; // Exit function early
        }
    }
    printf("Room Not Available or Insufficient Capacity!\n");
}
```

Explanation

- The `if` statement is the most important part. It checks 3 conditions:
    1. Does the `room_number` match?
    2. Is the `status` "Available"? (using `strcmp`)
    3. Is the room's `capacity` sufficient?
- If all 3 are true, `strcpy` is used to **update the data**: `status` becomes "Occupied" and the `guest_name` is saved.

Code: checkOut() Logic

```
void checkOut(struct Room rooms[], int totalRooms) {
    int roomNumber;
    printf("Enter Room Number: ");
    scanf("%d", &roomNumber);

    for (int i = 0; i < totalRooms; i++) {

        // Core check-out logic
        if (rooms[i].room_number == roomNumber &&
            strcmp(rooms[i].status, "Occupied") == 0)
        {
            // ... (Confirm check out) ...

            strcpy(rooms[i].status, "Available");
            rooms[i].guest_name[0] = '\0'; // Clear guest name

            printf("Check Out Successful!\n");
            return;
        }
    }
    printf("Room Not Occupied!\n");
}
```

Explanation

- This is the reverse of `bookRoom`.
- It checks if the `room_number` matches AND the `status` is "Occupied".
- If true, it **resets the `status`** back to "Available".
- It clears the `guest_name` by setting its first character to `\0` (the null terminator), making it an empty string.

Code: `displayRoomDetails()` Logic

```
void displayRoomDetails(struct Room rooms[], int totalRooms) {
    char current_type[50];
    int count = 0;
    int start_room = 1;

    // Outer loop finds the start of a new type
    for (int i = 0; i < totalRooms; ) {
        strcpy(current_type, rooms[i].room_type);
        count = 0;

        // Inner loop counts all rooms of that type
        for (int j = i; j < totalRooms; j++) {
            if (strcmp(rooms[j].room_type, current_type) == 0) {
                count++;
                i++; // Increment outer loop index!
            } else {
                break; // Stop when type changes
            }
        }
        printf("%-20s\t%d (%d-%d)...\n",
               current_type, count, start_room, start_room + count - 1);
        start_room += count;
    }
}
```

Explanation

- This function uses a **nested loop** to group rooms by type.
- The **outer loop** finds the start of a new type (e.g., "Standard").
- The **inner loop** counts how many rooms in a row are of that same type.
- It increments the outer loop's index (`i`) from *within* the inner loop.
- This lets us print a clean summary line like: **"Standard 10 (1-10)"**.

Testing & Results

Test Case	Input	Expected Output	Actual Output	Result
Invalid Room Number	Book room 99	Error message	"Room Not Available..."	Pass
Successful Check-out	Check-out room 5 (Occupied)	Room marked as available	"Check Out Successful!"	Pass
Book Occupied Room	Book room 5 (Occupied)	Error message	"Room Not Available..."	Pass
Check-out Available Room	Check-out room 10 (Available)	Error message	"Room Not Occupied!"	Pass

Conclusion & Future Work

Conclusion: The system successfully automates core hotel operations using fundamental C concepts like structures, arrays, and functions.

Future Work:

- Implement **File Handling** for persistent data storage.
- Add a proper **database (MySQL/SQLite)**.
- Create a **Graphical User Interface (GUI)**.
- Build a full **Billing Module**.

Thank You

Questions?

Image Sources

https://upload.wikimedia.org/wikipedia/en/thumb/8/8b/NEDUET_logo.svg/200px-NEDUET_logo.svg.png

Source: en.wikipedia.org

https://static.vecteezy.com/system/resources/thumbnails/012/702/809/small_2x/impression-management-promote-your-business-entrepreneurship-personal-brand-strategy-social-interaction-and-influence-abstract-metaphor-set-flat-modern-illustration-vector.jpg

Source: www.vecteezy.com

https://svg.template.creately.com/i0mut4673

Source: creately.com