

# CHAPTER 1

## INTRODUCTION

The Parking Lot Management System is a user-friendly program written in C programming language that simplifies the management of a parking lot. This system is designed to handle operations such as parking a car, removing a car, and displaying the current status of the parking lot. It is built using fundamental data structure operations and leverages arrays for data storage and management.

### Key Features

1. **Car Parking:** The program allows users to park a car by providing its license plate. It records the time of entry and ensures no car is parked if the lot is full.
2. **Car Removal:** Users can remove a specific car by entering its license plate. The system calculates the parking duration and charges based on the time the car was parked.
3. **Lot Status Display:** The system provides a clear view of all parked cars, along with their license plates and entry times.
4. **Time and Fee Management:** It uses the system clock to track parking durations and calculates charges dynamically based on a defined rate.

### Background

The Parking Lot Management System program is inspired by the need for efficient and structured vehicle parking solutions in modern urban environments. With limited parking spaces and a growing number of vehicles, effective parking management has become a critical issue in both public and private facilities. This code aims to simulate the core functionalities of a parking management system using basic programming and data structure techniques.

## Objective

The primary objective of this program is to provide an efficient and user-friendly solution for managing a parking lot using basic programming and data structure principles. It aims to simulate a real-world parking management system by allowing users to perform essential operations like parking, removing, and tracking vehicles.

### 1. Efficient Vehicle Management:

- Facilitate the addition of vehicles to the parking lot, ensuring no over-parking occurs.
- Maintain an organized structure for tracking parked vehicles.

### 2. Dynamic Time and Fee Calculation:

- Accurately track the duration of a car's stay using the system's clock.
- Calculate parking fees based on a predefined hourly rate.

### 3. User Convenience:

- Provide an intuitive interface for performing operations such as parking a car, removing a car, and viewing the lot's status.
- Ensure users receive clear feedback on operations, such as charges and errors (e.g., when a car is not found).

### 4. Practical Application of Data Structures:

- Use arrays to model the parking lot, demonstrating how fundamental data structures can be applied to solve real-world problems.
- Perform operations such as insertion, deletion, and traversal within the array.

### 5. Error Handling:

- Ensure the system checks for boundary conditions, such as full or empty parking lots, to avoid invalid operations.

### 6. Learning and Prototyping:

- Serve as a prototype for students and developers to learn about basic programming techniques, data structure implementation, and time management in C.
- Provide a foundation for building more advanced systems with additional features such as scalability, automation, and database integration.

## Scope

The scope of this program is limited to providing a simple yet functional simulation of a parking lot management system using basic programming and data structure concepts. While the program is designed for educational and small-scale practical applications, its core functionalities can be extended or scaled up for more complex and real-world scenarios.

## Key Features of the Parking Lot Management System

### 1. Core Functionality:

#### ○ Parking Vehicles:

- Users can park cars in the lot, provided space is available.
- Each car's license plate and entry time are recorded for tracking.

#### ○ Removing Vehicles:

- Users can remove specific cars from the lot using their license plate.
- The program calculates the parking duration and displays the corresponding charges.

### 2. Dynamic Fee Calculation:

- Calculates parking charges based on the time a car spends in the lot using a predefined hourly rate.
- Uses real-time tracking to compute accurate fees.

### 3. Real-Time Tracking:

- Utilizes the time.h library to record and retrieve entry and exit times.
- Displays entry times in a human-readable format.

### 4. Lot Status Display:

- Provides a detailed overview of the parking lot, including:
  - License plates of parked cars.
  - Entry times of each vehicle.

### 5. User-Friendly Messages:

- Offers clear feedback for each operation, such as parking success, full lot notifications, car removal success, or errors when the car is not found.
- Includes emojis for a more engaging user experience.

**6. Error Handling:**

- Prevents invalid operations, such as:
  - Parking a car when the lot is full.
  - Removing a car from an empty lot.
  - Attempting to remove a car that is not in the lot.

**7. Capacity Management:**

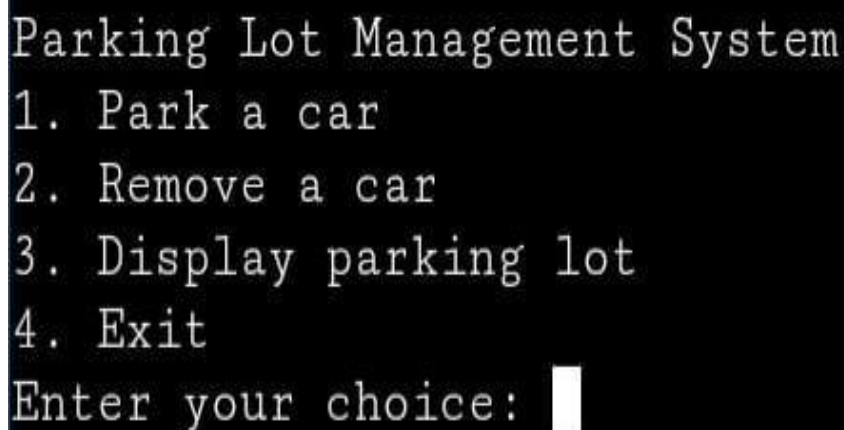
- Supports a maximum capacity (MAX) for the parking lot.
- Ensures all operations respect the capacity constraint.

**8. Educational Design:**

- Provides a modular structure with clear function definitions for each operation.
- Demonstrates the application of arrays for sequential data management.

**9. Cross-Platform Console Interface:** ○ A simple text-based interface that runs on any platform with a C compiler.**10. Code Readability and Modularity:**

- Each function is designed to perform a specific task, making the code easy to read, debug, and extend.

**Sample of service offered:**

```
Parking Lot Management System
1. Park a car
2. Remove a car
3. Display parking lot
4. Exit
Enter your choice: █
```

Fig.1.1 Samples offered

## CHAPTER 2

### PROBLEM STATEMENT

The objective of this project is to design and implement a Parking Lot Management System using C programming. The system should simulate the operation of a parking lot where cars can be parked, removed, and parking charges can be calculated based on the time spent in the lot.

#### 1. Park a Car:

- The system should allow the user to park a car by entering its license plate number.
- The car is assigned an entry time, which will be used to calculate parking duration and charges.
- The parking lot can hold a maximum of 10 cars (this number can be adjusted).

#### 2. Remove a Car:

- The system should allow a specific car to be removed from the parking lot.
- The user should provide the car's license plate number for removal.
- Upon removal, the system will calculate:
  - The parking duration (from entry time to the time of removal).
  - The charges based on the duration (calculated as ₹10 per hour).
- The system should print the parking duration and the amount to be paid.

#### 3. Display Parking Lot Status:

- The system should display all the cars currently parked in the lot, along with their respective entry times.
- The system should show a message if the lot is empty.

#### 4. Full Lot Handling:

- If the parking lot is full, no new cars should be allowed to park, and the system should display a message saying "Parking lot is full."

**5. Error Handling:**

- If a car is not found in the lot for removal, the system should display an appropriate message.
- If the parking lot is empty and the user tries to remove a car, the system should indicate that no cars are available.

**6. Exit the System:**

- The system should have an option to exit, with a friendly goodbye message.

**Non-Functional Requirements:****1. User Interface:**

- The system should provide a simple text-based interface that allows the user to easily select options for parking, removing, and viewing cars in the lot.
- The interface should guide the user through available options with clear instructions.

**2. Data Persistence:**

- The system should maintain the state of the parking lot only during runtime. Upon program termination, the parking lot data should be lost (for this project, we won't implement file handling or database storage).

**3. Scalability:**

- The system should be easily scalable if the parking lot size (number of spaces) needs to be increased.

**Example Scenarios:**

- 1. Parking a Car:** The user enters a license plate, and the system parks the car, recording the Every time
  - If the parking lot is full, the user is notified that the parking lot is full.
- 2. Removing a Car:** The user enters the license plate of the car they want to remove.
  - The system calculates the parking duration and the charges based on the time the car was parked.
  - If the car is not found, the system will notify the user.
- 3. Viewing Parking Lot Status:** The user can see all cars currently parked in the lot, including the entry times.

## CHAPTER 3

### IMPLEMENTATION

Below is the complete implementation of the Parking Lot Management System in C, following the specifications outlined in the problem statement.

**Code:**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#define MAX 10 // Maximum capacity of the parking lot
#define RATE_PER_HOUR 10.0 // Define parking rate per hour

// Structure to represent a car
typedef struct {
    char license_plate[20];
    time_t entry_time; // Store entry time
} Car;

// Structure for the parking lot
typedef struct {
    Car cars[MAX];
    int size;
} ParkingLot;

// Initialize the parking lot
void initialize(ParkingLot *lot) {
    lot->size = 0;
}

// Check if the parking lot is full
int isFull(ParkingLot *lot) {
    return lot->size == MAX;
}

// Check if the parking lot is empty
int isEmpty(ParkingLot *lot) {
    return lot->size == 0;
}

// Add a car to the parking lot
void parkCar(ParkingLot *lot, char *license_plate) {
    if (isFull(lot)) {
        printf("Parking lot is full! Cannot park the car: %s\n", license_plate);
        return;
    }
}
```

```
}
strcpy(lot->cars[lot->size].license_plate, license_plate);
lot->cars[lot->size].entry_time = time(NULL); // Record entry time
lot->size++;
printf("Car parked: %s \U0001F697 \U0001F17F\n", license_plate);
}

// Remove a specific car from the parking lot
void leaveCar(ParkingLot *lot, char *license_plate) {
    if (isEmpty(lot)) {
        printf("Parking lot is empty! No cars to remove.\n");
        return;
    }

    // Find the car
    int found = -1;
    for (int i = 0; i < lot->size; i++) {
        if (strcmp(lot->cars[i].license_plate, license_plate) == 0) {
            found = i;
            break;
        }
    }

    if (found == -1) {
        printf("Car with license plate %s not found in the parking lot.\n", license_plate);
        return;
    }

    // Calculate parking duration
    time_t exit_time = time(NULL);
    double duration = difftime(exit_time, lot->cars[found].entry_time) / 3600; // Duration
```



```
// Calculate charges
double charges = duration * RATE_PER_HOUR;
printf("Car leaving: %s\n", lot->cars[found].license_plate);
printf("Parking duration: %.2f hours\n", duration);
printf("Parking charges: ₹%.2f\n", charges);

// Shift cars to fill the gap
for (int i = found; i < lot->size - 1; i++) {
    lot->cars[i] = lot->cars[i + 1];
}
lot->size--;
}

// Display the status of the parking lot
void displayParkingLot(ParkingLot *lot) {
    if (isEmpty(lot)) {
        printf("Parking lot is empty.\n");
        return;
    }
    printf("Cars in the parking lot:\n");
    for (int i = 0; i < lot->size; i++) {
        printf("%d. %s (Parked at: %s)\n", i + 1, lot->cars[i].license_plate,
            ctime(&lot->cars[i].entry_time)); // Convert time to readable format
    }
}

// Main function
int main() {
    ParkingLot lot;
    initialize(&lot);
```

```
ParkingLot lot;
initialize(&lot);

int choice;
char license_plate[20];

do {
    printf("\nParking Lot Management System\n");
    printf("1. Park a car\n");
    printf("2. Remove a car\n");
    printf("3. Display parking lot\n");
    printf("4. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            printf("Enter license plate number: ");
            scanf("%s", license_plate);
            parkCar(&lot, license_plate);
            break;
        case 2:
            printf("Enter license plate number to remove: ");
            scanf("%s", license_plate);
            leaveCar(&lot, license_plate);
            printf("THANKYOU \U0001F642 for visting \U0001F64F");
            break;
        case 3:
            displayParkingLot(&lot);
            break;
        case 4:
            printf("Exiting system. Have a NICE DAY bye\U0001F44B!\n");
            break;
        default:
            printf("Invalid choice! Please try again.\n");
    }
} while (choice != 4);

return 0;
}
```

## CHAPTER 4

## RESULTS

### OUTPUT(screenshots)

#### Step1:

```
Parking Lot Management System
1. Park a car
2. Remove a car
3. Display parking lot
4. Exit
Enter your choice: 
```

Fig.4.1 starting choice

#### Step2/choice1

```
2. Remove a car
3. Display parking lot
4. Exit
Enter your choice: 1
Enter license plate number: ka06f1718
Car parked: ka06f1718 🚗

Parking Lot Management System
1. Park a car
2. Remove a car
3. Display parking lot
4. Exit
Enter your choice: 1
Enter license plate number: ka51d7859
Car parked: ka51d7859 🚗

Parking Lot Management System
1. Park a car
2. Remove a car
3. Display parking lot
4. Exit
Enter your choice: ka15j9621
Enter license plate number: Car parked: ka15j9621 🚗

Parking Lot Management System
1. Park a car
2. Remove a car
3. Display parking lot
4. Exit
Enter your choice: 
```

Fig.4.2 choice 1 of the project

### Step3:choice3,choice2

```
1. Park a car
2. Remove a car
3. Display parking lot
4. Exit
Enter your choice: 3
Cars in the parking lot:
1. ka06f1718 (Parked at: Sun Dec 1 17:56:10 2024
)
2. ka51d7859 (Parked at: Sun Dec 1 17:56:23 2024
)
3. ka15j9621 (Parked at: Sun Dec 1 17:56:35 2024
)

Parking Lot Management System
1. Park a car
2. Remove a car
3. Display parking lot
4. Exit
Enter your choice: 2
Enter license plate number to remove: ka06f1718
Car leaving: ka06f1718
Parking duration: 0.09 hours
Parking charges: ₹0.88
THANKYOU 😊 for visting 🙏
Parking Lot Management System
1. Park a car
2. Remove a car
3. Display parking lot
4. Exit
Enter your choice: []
```

Fig.4.3 Choice 3, Choice 2 of the project

### Step4:choice2/choice4

```
3. Display parking lot
4. Exit
Enter your choice: 2
Enter license plate number to remove: ka06f1718
Car leaving: ka06f1718
Parking duration: 0.02 hours
Parking charges: ₹0.20
THANKYOU 😊 for visting 🙏
Parking Lot Management System
1. Park a car
2. Remove a car
3. Display parking lot
4. Exit
Enter your choice: 2
Enter license plate number to remove: ka51d6985
Car leaving: ka51d6985
Parking duration: 0.02 hours
Parking charges: ₹0.21
THANKYOU 😊 for visting 🙏
Parking Lot Management System
1. Park a car
2. Remove a car
3. Display parking lot
4. Exit
Enter your choice: 4
Exiting system. Have a Niceday bye 🙏
```

Fig.4.4 Choice 2/ Choice 4 of the project

## CHAPTER 5

### CONCLUSION

The Parking Lot Management System is a well-designed and efficient solution for managing parking operations. By leveraging fundamental programming concepts and data structures, this system allows users to perform key operations such as parking cars, removing them, and displaying the current parking status with ease.

#### Key Takeaways:

##### 1. Functionality:

- The system provides a user-friendly interface to manage a limited-capacity parking lot.
- Essential features like duration-based billing and real-time status display ensure smooth operation.

##### 2. Scalability:

- The program is designed to handle basic parking management needs and can be easily extended for more complex functionalities, such as online reservations, multi-level parking, or dynamic pricing.

##### 3. Practical Application:

- The system simulates a real-world parking scenario, showcasing its practical utility in small-scale parking lots like offices, residential complexes, or private events.

##### 4. Learning Aspects:

- This implementation demonstrates the use of data structures (arrays) and concepts like modular programming, time handling, and input validation in solving real-life problems.

#### Future Enhancements:

- Integrating a database for persistent storage of records.
- Adding features for advanced billing (e.g., flat rates for specific durations).
- Enabling online access or mobile app integration for remote operations.

In conclusion, this program achieves its objective of efficiently managing parking operations while being simple to use and implement. It serves as a foundation for more sophisticated parking management solutions.

**Outcome:** The Parking Lot Management System successfully achieves its objectives by providing an efficient and user-friendly platform for parking operations.

## REFERENCES

- <https://www.geeksforgeeks.org>
- <https://chat.openai.com>
- <https://pages.github.com>
- <https://sites.google.com>