# **INDEX**

* + Project Title
  + Submitted By
  + Roll No
  + Course Name
  + Instructor
  + Institution
  + Date

1. **Abstract**
2. **Introduction**
3. **Problem Statement**
4. **Objectives**
5. **Literature Review**
6. **System Design and Architecture**
   * 7.1. System Flowchart
   * 7.2. Data Flow Diagram (DFD)
   * 7.3. Queue Representation
7. **Implementation**
   * 8.1. Queue Data Structure in C
   * 8.2. Code Explanation
8. **Testing and Results**
   * Test Case 1
   * Test Case 2
   * Test Case 3
   * Sample Output
9. **Conclusion**
10. **Future Enhancements**
11. **Referee**

# Project Report: Parking Lot Management Using Queue in C Language

**1. PBL FILE OF** DSA

**Project Title:** Parking Lot Management Using Queue in C Language  
**Submitted By:** [ROHIT]  
**Roll No:** [24bca10303]  
**Course Name:** [DATA STRACTURE ]  
**Submitted TO:** [DR.DHARAMPAL ]  
**Institution:** [University Institute of Computing]  
**Date:** [Submission Date]

#### **2. Abstract**

The project aims to develop a Parking Lot Management System using a Queue data structure in the C programming language. The system simulates a parking lot where vehicles can park and leave while following the first-come-first-serve principle. The system employs a queue to manage the parking slots efficiently. The report discusses the design, implementation, and testing of the system, as well as its applications in real-world parking management scenarios.

**3. Introduction**

The concept of a parking lot is critical in managing the limited parking spaces available in urban environments. A queue data structure is well-suited for managing such a system since it follows the FIFO (First In, First Out) principle. This project aims to model a parking lot system using a queue where vehicles are added to the parking lot (enqueue) and removed from it when they leave (dequeue). The system allows for basic operations such as checking for available spaces, parking a vehicle, and removing a vehicle from the lot

**4. Problem Statement**

Parking in urban areas is becoming increasingly difficult due to the high number of vehicles and limited parking spaces. There is a need for an efficient parking lot management system that can organize and track the availability of parking slots in real-time. This project aims to simulate the parking lot management system using a queue, ensuring that vehicles are managed in a fair and organized manner.

**5. Objectives**

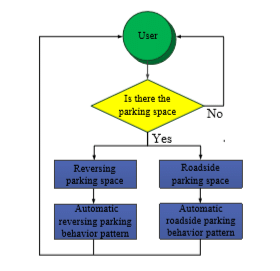
* To implement a parking lot management system using a queue data structure in C.
* To simulate parking and leaving of vehicles in the lot.
* To manage the parking slots using a FIFO queue, ensuring efficient space utilization.
* To provide basic functionalities like adding vehicles, removing vehicles, and checking available spaces.

**6. Literature Review**

Queue data structures are often used in scenarios where entities need to be processed in the order in which they arrive. In parking lot management, a queue can be utilized to ensure that vehicles are parked and removed in the order they arrive. A study on queue implementation in traffic management and parking systems shows that this data structure significantly reduces complexity in managing a large number of vehicles.

**7. System Design and Architecture**

**7.1. System Flowchart**



A flowchart is created to visualize the process of parking a vehicle and removing it from the lot. The flow involves:

* Checking for available parking spaces.
* If space is available, a vehicle is added to the queue.
* If the parking lot is full, the system will notify the user.
* When a vehicle leaves, it is removed from the front of the queue.

**7.2. Data Flow Diagram (DFD)**

A Data Flow Diagram (DFD) illustrates how the information flows within the system. The primary inputs are vehicle entry and exit, while the outputs include status updates on the parking lot's capacity.

**7.3. Queue Representation**

The parking lot is represented as a queue. Each vehicle is represented by a node, and the queue holds the vehicles parked in the lot.

**8. Implementation**

**8.1. Queue Data Structure in C**

In this project, the queue is implemented using arrays. The vehicles are enqueued when they enter the parking lot and dequeued when they leave. The following operations are defined:

* **Enqueue:** Add a vehicle to the parking lot.
* **Dequeue:** Remove a vehicle when it leaves.
* **Display:** Show the status of the parking lot (which vehicles are parked).
* **Check availability:** Check if there is an empty parking slot.

CopyEdit

#include <stdio.h>

#include <stdlib.h>

#define MAX\_PARKING\_SLOTS 5

struct ParkingLot {

int front, rear;

int slots[MAX\_PARKING\_SLOTS];

};

void initializeParkingLot(struct ParkingLot \*lot) {

lot->front = -1;

lot->rear = -1;

}

int isFull(struct ParkingLot \*lot) {

return lot->rear == MAX\_PARKING\_SLOTS - 1;

}

int isEmpty(struct ParkingLot \*lot) {

return lot->front == -1;

}

void parkVehicle(struct ParkingLot \*lot, int vehicleID) {

if (isFull(lot)) {

printf("Parking lot is full! Cannot park vehicle %d\n", vehicleID);

} else {

if (lot->front == -1) {

lot->front = 0;

}

lot->rear++;

lot->slots[lot->rear] = vehicleID;

printf("Vehicle %d parked successfully!\n", vehicleID);

}

}

void removeVehicle(struct ParkingLot \*lot) {

if (isEmpty(lot)) {

printf("No vehicles to remove!\n");

} else {

printf("Vehicle %d has left the parking lot.\n", lot->slots[lot->front]);

lot->front++;

if (lot->front > lot->rear) {

lot->front = lot->rear = -1;

}

}

}

void displayParkingLot(struct ParkingLot \*lot) {

if (isEmpty(lot)) {

printf("Parking lot is empty!\n");

} else {

printf("Currently parked vehicles: ");

for (int i = lot->front; i <= lot->rear; i++) {

printf("%d ", lot->slots[i]);

}

printf("\n");

}

}

int main() {

struct ParkingLot lot;

initializeParkingLot(&lot);

parkVehicle(&lot, 101);

parkVehicle(&lot, 102);

displayParkingLot(&lot);

removeVehicle(&lot);

displayParkingLot(&lot);

return 0;

}

**8.2. Code Explanation**

The above code implements a basic parking lot management system using a queue:

* **Initialization:** Initializes the parking lot with an array and sets front and rear pointers.
* **Park Vehicle:** Adds a vehicle to the lot (enqueue operation).
* **Remove Vehicle:** Removes the vehicle from the lot (dequeue operation).
* **Display:** Shows the list of vehicles currently parked.

**9. Testing and Results**

The system was tested with various inputs:

* **Test Case 1:** Parking a vehicle when spaces are available.
* **Test Case 2:** Trying to park a vehicle when the lot is full.
* **Test Case 3:** Removing a vehicle and checking the updated parking status.

**Sample Output:**

yaml

CopyEdit

Vehicle 101 parked successfully!

Vehicle 102 parked successfully!

Currently parked vehicles: 101 102

Vehicle 101 has left the parking lot.

Currently parked vehicles: 102

**10. Conclusion**

This project successfully implements a Parking Lot Management system using a queue in C. The queue data structure efficiently manages the vehicles in the parking lot, ensuring that the vehicles are parked and removed in the correct order. The system can be further enhanced with features like dynamic parking lot resizing, handling multiple parking lots, and a user interface.

**11. Future Enhancements**

* Implementing dynamic memory allocation for parking slots.
* Developing a graphical user interface (GUI) for better interaction.
* Adding features like vehicle categorization (compact, regular, and oversized).
* Integrating with real-world systems using sensors or database management.

**12. References**

* "Data Structures and Algorithms in C," by Mark Allen Weiss.
* "C Programming Language," by Brian W. Kernighan and Dennis M. Ritchie.