Rajshahi University of Engineering & Technology

Department of Mechatronics Engineering

Project Report

Course No : CSE 1288
Course Title : Computer Fundamentals & Programming Sessional.
Project Name : Development of a Parking Management System Using C Programming.

Submitted By: Al Nimra Kabir
Roll: 2008042
1st Year, Even Semester
Department of Mechatronics Engineering
Rajshahi University of Engineering & Technology, Rajshahi.

Submitted To:

Md. Manirul Islam
Assistant Professor
Department of Mechatronics
Rajshahi University of Engineering & Technology, Rajshahi.

Md. Mehedi Hasan
Lecturer
Department of Mechatronics
Rajshahi University of Engineering & Technology, Rajshahi.

Date Of Submission: 31 December 2022

Title:

Development of a Parking Management System Using C Programming.

Introduction:

The project titled " Development of a Parking Management System" is a program designed to manage the parking of vehicles in a parking lot. The program allows users to park a car, show the current status of the parking lot, remove a car, and delete all data. The program has the capability to store and display the number of buses, cars, and bikes parked in the lot, as well as the total number of vehicles and the total earnings. The program is written in C language and utilizes various libraries such as stdio.h, conio.h, and stdlib.h. The program consists of various functions including the main function, menu function, and functions for parking and removing vehicles. The program also has the ability to store data in an array for the spaces in the parking lot, keeping track of which spaces are occupied and which are available. Overall, the " Development of a Parking Management System Using C Programming." program is a useful tool for efficiently managing and organizing the parking of vehicles in a parking lot.

How the project works:

After the execution of the Code, the C program is a parking management system that allows a user to park, remove, and view the status of vehicles in a parking lot. It has a menu-driven interface that allows the user to select different options. The options include:

1. Parking a vehicle.
2. Viewing the status of the parking lot.
3. Removing a vehicle.
4. Deleting all data.
5. Exiting the program.

The program keeps track of the number of buses, rickshaws, cycles, and the total number of vehicles in the parking lot. It also keeps track of the total earnings and the number of occupied spaces in the parking lot.

- When a user selects the option to park a vehicle, they are asked to enter the type of vehicle they want to park, which can be a bus, car, or bike. The program then adds the vehicle to the parking lot and updates the relevant data, such as the number of vehicles, the total earnings, and the number of occupied spaces.
- When the user selects the option to view the status of the parking lot, the program displays the occupied spaces, the number of buses, cars, bikes, and the total number of vehicles in the parking lot, as well as the total earnings.
- When the user selects the option to remove a vehicle, they are asked to enter the space number of the vehicle they want to remove. The program then removes the vehicle from the parking lot and updates the relevant data.
- When the user selects the option to delete all data, the program resets all data, such as the number of vehicles, the total earnings, and the number of occupied spaces, to zero.

Algorithm:

Step 1: Start.

Step 2: Initialize constant NUM_SPACES and variables:

- noc: number of cars
- nob: number of buses
- nok: number of bikes
- amount: total earnings
- count: total number of vehicles

- num_occupied: number of occupied parking spaces
- spaces: an array of size NUM_SPACES to keep track of occupied parking spaces.

Step 3: Display the main menu and prompt the user for a choice.

Step 4: Based on the user's choice, perform the following actions:

1. If the choice is 1, call the park_a_vehicle() function.

   And go to Step 5.
2. If the choice is 2, call the show_data() function.

   And go to Step 6.
3. If the choice is 3, call the remove_a_vehicle() function.

   And go to Step 7.
4. If the choice is 4, call the Delete() function.

   And go to Step 8.
5. If the choice is 5, exit the program.

   And go to Step 12.

Step 5: In the park_a_vehicle() function:

- If all parking spaces are occupied, display a message saying that the parking lot is full.
- If there are available parking spaces, prompt the user to choose a type of vehicle to park (car, bike, or bus).
- Based on the user's choice, call the appropriate function

  If the choice is 1; call the bus() function and go to Step 9.

  If the choice is 2; call car() function and go to Step 10.

  If the choice is 3; call bike() function and go to Step 11.

Step 6: In the show_data() function:

- Display the list of parking spaces and their occupied status.
- Display the number of cars, buses, bikes, and the total number of vehicles.
- Display the total earnings.

Step 7: In the remove_a_vehicle() function:

- If there are no vehicles in the parking lot, display a message saying that the parking lot is empty.
- If there are vehicles in the parking lot, prompt the user to enter the space number of the vehicle they want to remove.
- If the space number is invalid or the space is already empty, display an error message.
- If the space number is valid and the space is occupied, mark the space as empty and decrement the num_occupied variable.

Step 8: In the Delete() function:

Set all variables to 0 (noc, nob, nok, amount, count, num_occupied).

Step 9: In the bus() function:

- Increment nob by 1.
- Increment amount by 100 (the fee for bus parking).
- Increment count by 1.
- If all parking spaces are occupied, display a message saying that the parking lot is full.
- If there are available parking spaces, prompt the user to enter the space number where they

want to park their bus.

- If the space number is invalid, display an error message.
- If the space number is valid and the space is occupied, display an error message.
- If the space number is valid and the space is not occupied, mark the space as occupied and increment the num_occupied variable.

Step 10: In the car() function:

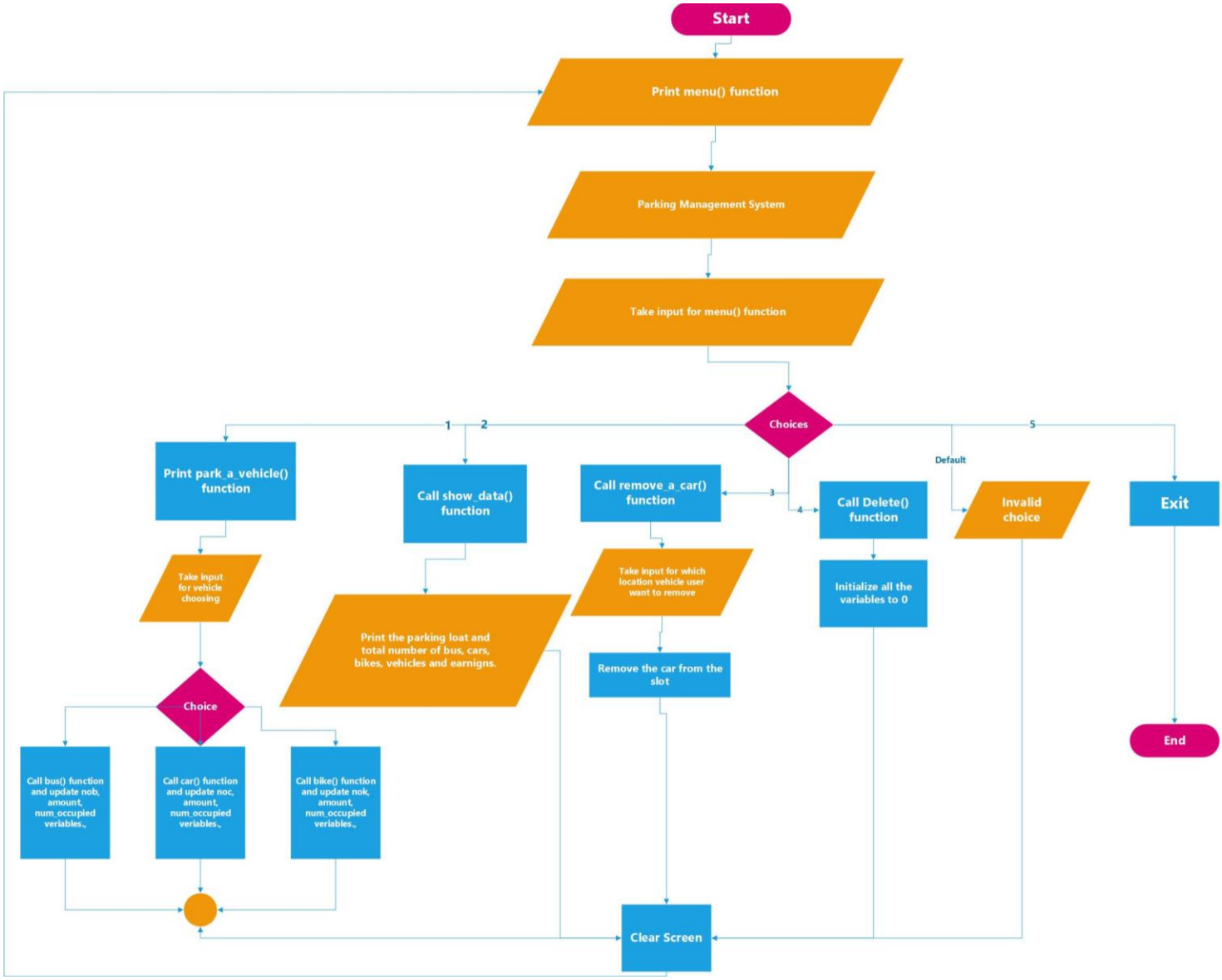
- Increment noc by 1.
- Increment amount by 50 (the fee for car parking).
- Increment count by 1.
- If all parking spaces are occupied, display a message saying that the parking lot is full.
- If there are available parking spaces, prompt the user to enter the space number where they want to park their car.
- If the space number is invalid, display an error message.
- If the space number is valid and the space is occupied, display an error message.
- If the space number is valid and the space is not occupied, mark the space as occupied and increment the num_occupied variable.

Step 11: In the bike() function:

- Increment nok by 1.
- Increment amount by 20 (the fee for bike parking).
- Increment count by 1.
- If all parking spaces are occupied, display a message saying that the parking lot is full.
- If there are available parking spaces, prompt the user to enter the space number where they want to park their bike.
- If the space number is invalid, display an error message.
- If the space number is valid and the space is occupied, display an error message.
- If the space number is valid and the space is not occupied, mark the space as occupied and increment the num_occupied variable.

Step 12: End.

Flowchart:


Start
Print menu() function
Parking Management System
Take input for menu() function
Choices
1
2
3
4
5
Default
Print park_a_vehicle() function
Call show_data() function
Call remove_a_car() function
Call Delete() function
Invalid choice
Exit
Take input for vehicle choosing
Print the parking loat and total number of bus, cars, bikes, vehicles and earnigns.
Take input for which location vehicle user want to remove
Initialize all the variables to 0
Remove the car from the slot
Choice
Call bus() function and update nob, amount, num_occupied veriables.,
Call car() function and update noc, amount, num_occupied veriables.,
Call bike() function and update nok, amount, num_occupied veriables.,
End
Clear Screen

Code:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define NUM_SPACES 10
int  noc=0,nob=0,nok=0,ammount=0,count=0,num_occupied = 0;
int spaces[NUM_SPACES] = {0};
int main()
{
   while(1)
   {
   printf("\nParking Management System\n");
   printf("_____\n");
   switch(menu())
   {
      case 1:
         park_a_vehicle();
         break;
      case 2:
         show_data();
         break;
      case 3:
         remove_a_vehicle();
         break;
      case 4:
         Delete();
         break;
      case 5:
         exit(0);
      default :
         printf("\nInvalid Choice");
   }
   getch();
   system("cls");
   }
}
int menu()
{
   int choice;
   printf("\n1. Park a car");
   printf("\n2. Show Status.");
   printf("\n3. Remove a car.");
   printf("\n4. Delete All Data.");
   printf("\n5. Exit.");
   printf("\n\nEnter your choice:");
   scanf("%d",&choice);
   return choice;
}
void park_a_vehicle()
{
   if (num_occupied == NUM_SPACES)
      {
        printf("Sorry, the parking lot is full.\n");
      }
```

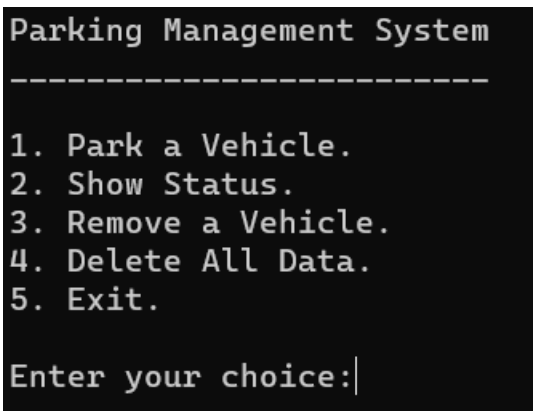
```
    else
    {
    int cho;
    printf("1.  Enter Bus.\n2.  Enter Car.\n3.  Enter Bike.\n");
    printf("\nEnter your choice:");

    scanf("%d", &cho);

        switch(cho)
        {
        case 1:
            bus();
            break;
        case 2:
            car();
            break;
        case 3:
            bike();
            break;
        default:
            printf("\nInvalid Choice");
        }
    }
}
void show_data()
{
    printf("Space\tOccupied\n");
        int i;

        for (i = 0; i < NUM_SPACES; i++)
        {
          printf("%d\t%s\n", i + 1, spaces[i] ? "Yes" : "No");

        }
    printf("\nNumber of Bus=%d", nob);
    printf("\nNumber of Car=%d", noc);
    printf("\nNumber of Bike=%d", nok);
    printf("\nTotal Number of Vehicle=%d", count);
    printf("\nTotal Earnings=%d",ammount);
}
void remove_a_vehicle()
{
     if (num_occupied == 0) {
          printf("The parking lot is empty.\n");
        } else {
          int space_num;
          printf("Enter the space number: ");
          scanf("%d", &space_num);
          if (space_num < 1 || space_num > NUM_SPACES) {
            printf("Invalid space number.\n");
          } else if (!spaces[space_num - 1]) {
            printf("This space is already empty.\n");
          } else {
            spaces[space_num - 1] = 0;
            num_occupied--;
```

```
          printf("Your car has been removed from space %d.\n", space_num);
        }
      }
}
void Delete()
{
   noc=0;
   nob=0;
   nok=0;
   ammount=0;
   count=0;
   num_occupied=0;
}
void car()
{
 noc++;
 ammount=ammount+50;  //fee for Car parking is 50
 count++;
 if (num_occupied == NUM_SPACES)
      {
       printf("Sorry, the parking lot is full.\n");
      }
 else
      {
       int i;
       for (i = 0; i < NUM_SPACES; i++)
       {
         if (!spaces[i])
         {
           spaces[i] = 1;
           num_occupied++;
           printf("Your car has been parked in space %d.\n", i + 1);
           break;
         }
       }
      }
}
void bike()
{
 nok++;
 ammount=ammount+20;  //fee for Bike parking is 20
 count++;
 if (num_occupied == NUM_SPACES)
      {
       printf("Sorry, the parking lot is full.\n");
      }
 else
      {
       int i;
       for (i = 0; i < NUM_SPACES; i++)
       {
         if (!spaces[i])
         {
           spaces[i] = 1;
           num_occupied++;
```

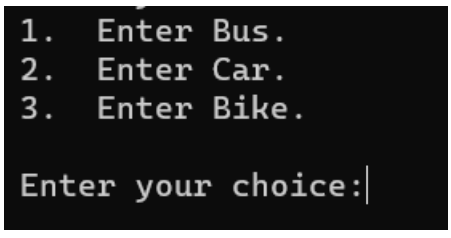
```
            printf("Your car has been parked in space %d.\n", i + 1);
            break;
          }
        }
      }
}

void bus()
{
  nob++;
  ammount=ammount+100;  //fee for Bus parking is 100
  count++;
  if (num_occupied == NUM_SPACES)
      {
        printf("Sorry, the parking lot is full.\n");
      }
  else
      {
        int i;
        for (i = 0; i < NUM_SPACES; i++)
        {
          if (!spaces[i])
          {
            spaces[i] = 1;
            num_occupied++;
            printf("Your car has been parked in space %d.\n", i + 1);
            break;
          }
        }
      }
}
```

Outputs:

Main menu



Park a vehicle



Show Status

```
Space   Occupied
1       Yes
2       No
3       No
4       No
5       No
6       No
7       No
8       No
9       No
10      No

Number of Bus=1
Number of Car=0
Number of Bike=0
Total Number of Vehicle=1
Total Earnings=100
```

Remove a vehicle

```
Enter your choice:3
Enter the space number: 1
Your car has been removed from space 1.
```

Conclusion:

The Parking Management System is a useful tool for managing a parking lot. It allows users to park, remove, and view the status of vehicles in the parking lot, as well as reset all data. The program can track the number of cars, buses, and bikes in the parking lot, as well as the total number of vehicles and the total earnings. The program also has error checking in place to ensure that users cannot park a vehicle in an already occupied space or remove a vehicle from an empty space. Overall, the program is easy to use and helps streamline the process of managing a parking lot.