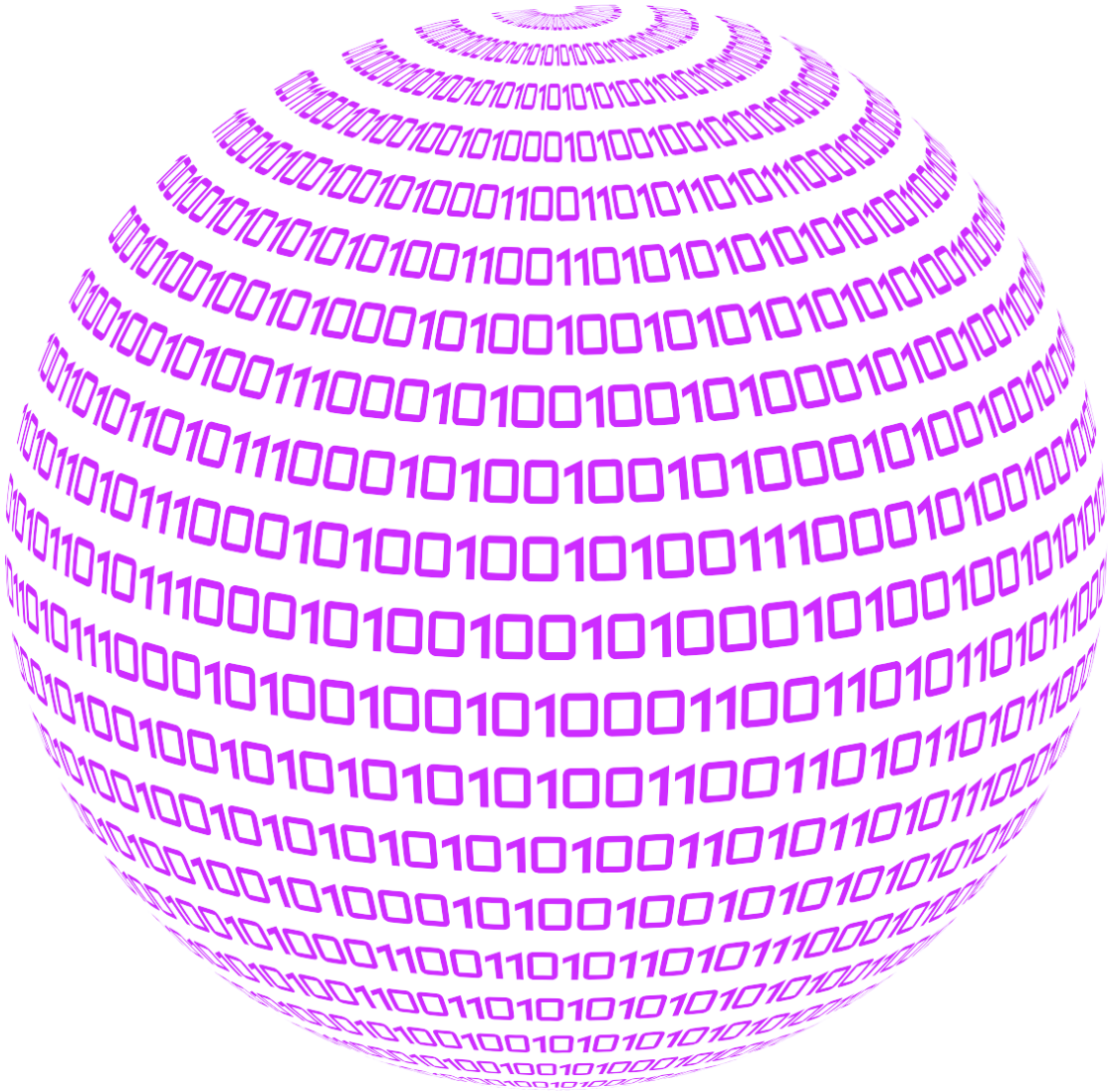
PROGRAMMING FUNDAMENTALS



LAB TASK

Muhammad Luqman
459317

Lab Report: C Structs

Introduction

This lab explored the concept of structures in C programming. Structures are user-defined data types that group variables of potentially different types under a single name. They offer a way to organize related data and improve code readability and maintainability.

Objectives

1. ***Understand the concept of structures in C programming.***
2. ***Define structures and access their members.***
3. ***Create structure variables and arrays.***
4. ***Utilize functions with structures (passing by value and reference).***
5. ***Implement dynamic memory allocation for structures.***

Task 5: Car Details using Structures

Instead of a simple app to show car details, `I made a car rental application.` While a basic car details program could demonstrate structures, the car rental app adds user interaction and functionality. This allows us to explore how structures can be used to manage and display car data in a more practical and engaging way. ***TLDR: It's more fun this way.***

Objectives

- Define a structure named `Car` to store car information.
- Create a function `displayCarDetails()` to print car details.

- Implement user interaction to get car details and display them.

Implementation

First thing I did was to define a `struct` to store car details. The `Car` structure is defined with the following members:

- `name` (string): Stores the full name of the car model (e.g., "Toyota Corolla").
- `model` (string): Stores the car model (e.g., "Corolla").
- `year` (integer): Stores the year the car was manufactured.
- `price` (float): Stores the car's price.
- `mileage` (integer): Stores the car's mileage.
- `available` (boolean): Indicates if the car is available for rent (true) or not (false).

Next up, I made, `displayCarDetails(),` a function that takes a `Car` struct as input and prints its details in a user-friendly format, including:

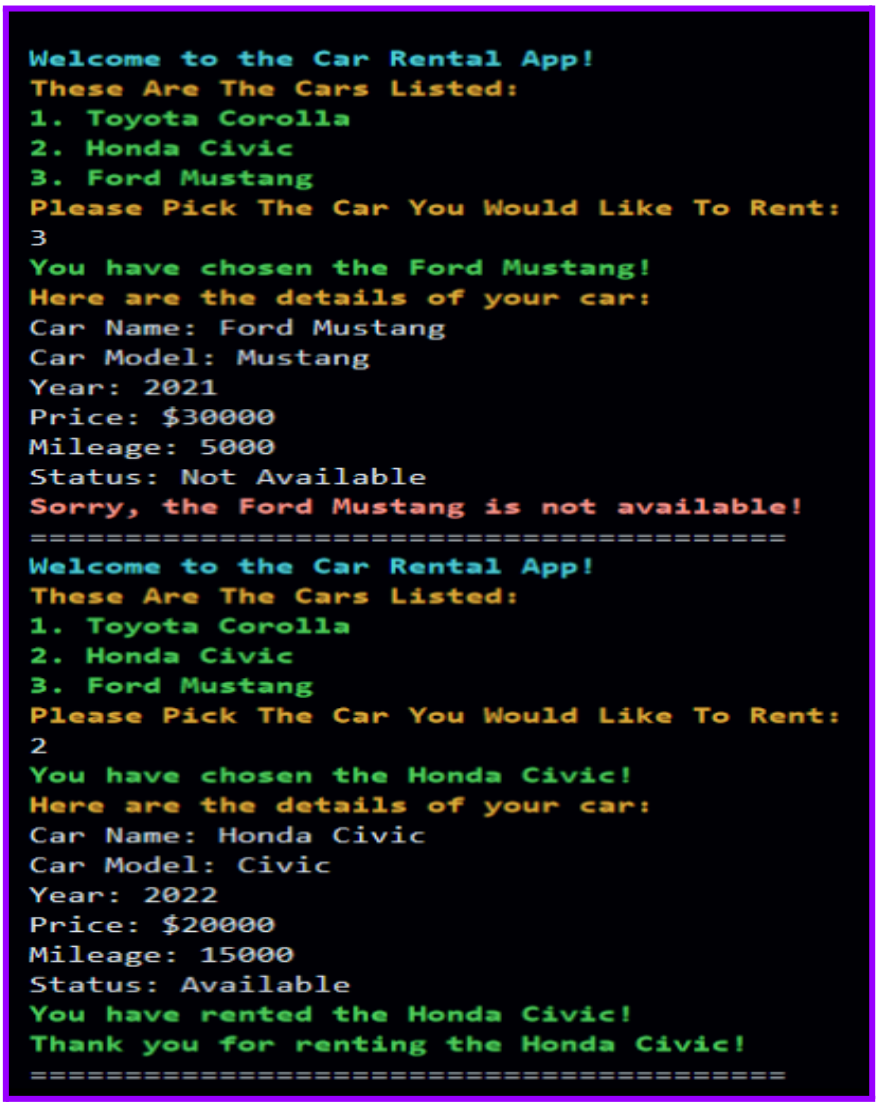
- ***Car Name***
- ***Car Model***
- ***Year***
- ***Price***
- ***Mileage***
- ***Availability Status ("Available" or "Not Available")***

3. main() Function:

The `main()` function demonstrates the usage of the `Car` structure and `displayCarDetails()` function. Here's a breakdown:

- A `bool` variable `rented` is initialized to `false`, indicating no car is currently rented.

- An array of `Car` structs is created to hold information for multiple cars. Each element is initialized with sample car details (name, model, year, price, mileage, and availability).
- A `while` loop keeps the program running until a car is rented using a simple condition `while(!rented)`:
    - A menu displays the available cars with their names.
    - User input is obtained to choose a car.
    - If the choice is valid, the chosen car's details are displayed using `displayCarDetails()`.
    - The `RentCar()` function is called to handle the rental process.
- The `RentCar()` function would update the car's availability status utilizing pointers to the `Car` structure.



Remaining Tasks

Task 1: Book Management

This task involved creating a program to manage and display book details using structures.

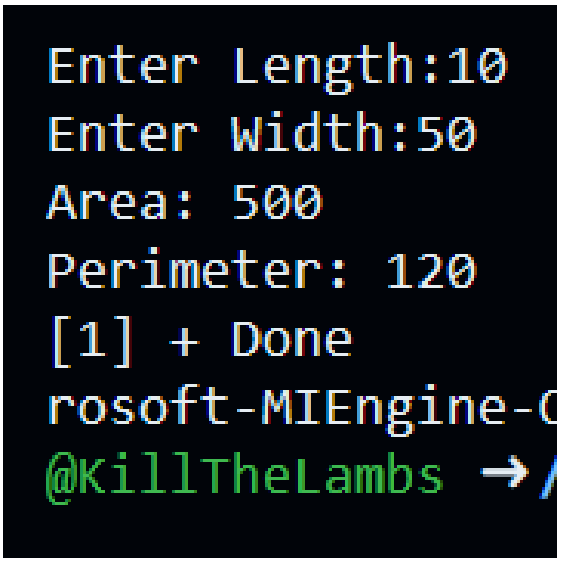
OUTPUT

```
Enter the title of the book: love is a dog from hell
Enter the author of the book: charles bukwowski
Enter the price of the book: 12

--------------------------------------------------

Title: love is a dog from hell
Author: charles bukwowski
Price: 12.00

--------------------------------------------------
```

Task 2: Rectangle Calculations

This task focused on creating a program to handle rectangle calculations using structures.

OUTPUT



Task 3: Update Person's Age

This task involved creating a program to update a person's age using structures and pointers.

OUTPUT

```
Enter the name of the first person: loki
Enter the age of the first person: 45
Enter the address of the first person: jk
Enter the new age of the person: 19
The new age of the person is: 19
Name: loki
Age: 19
Address: jk
```

Task 4: Student Details

Create a program that dynamically allocates memory for an array of student structures.

OUTPUT

```
Enter details for student 1
ID: 544354
Name: loki
Marks: 3.5
Enter details for student 2
ID: 64576
Name: pentium 5
Marks: 3.5
Enter details for student 3
ID: 45687
Name: emad
Marks: 3.2

Student Details:
Student 1
ID: 544354
Name: loki
Marks: 3.50
Student 2
ID: 64576
Name: pentium 5
Marks: 3.50
Student 3
ID: 45687
Name: emad
Marks: 3.20
```

Conclusion

This lab provided a hands-on exploration of structures in C programming. By completing the tasks, we gained a solid understanding of:

1. ***Defining structures and their member variables.***
2. ***Creating structure variables and arrays.***
3. ***Accessing members using the dot (.) and arrow (->) operators.***
4. ***Passing structures to functions by value and reference.***
5. ***Implementing dynamic memory allocation for structures.***
