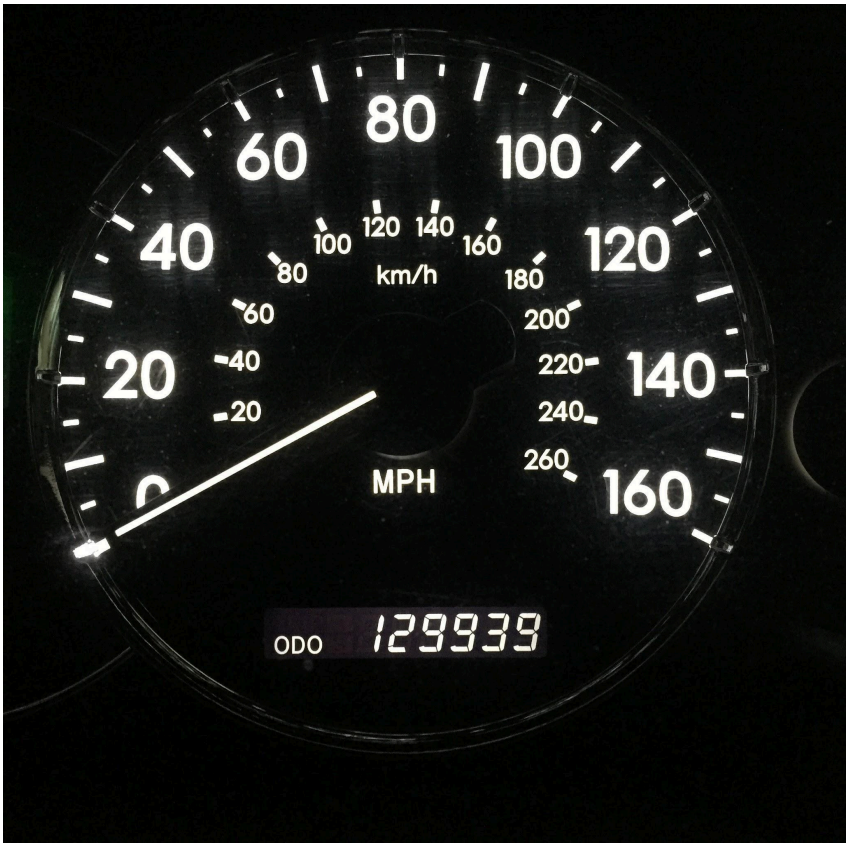
🚗Car Fuel Efficiency Management System🚗

01.Research(Understanding the topic)



Mileage (fuel efficiency) is one of the most important factors people consider while buying a car. It shows how efficiently a car uses fuel — the higher the mileage, the more distance it covers per litre, which directly reduces running costs and pollution.

In this project, the goal is to design a C program that stores and displays car information such as Car ID, Model Name, and Mileage, and sorts them in descending order of mileage.
Using structures allows grouping multiple related data types, and arrays help store multiple car records. Sorting them helps easily identify the most fuel-efficient cars from a dataset — similar to how real-world automobile databases work.

This idea helps in understanding data handling, sorting algorithms, and structure-based programming in C while relating to a real-world automobile context.

Sources:

1. https://www.iea.org/articles/fuel-economy-in-india — International Energy Agency: Fuel economy in India overview
2.https://timesofindia.indiatimes.com/auto/cars/10-most-fuel-efficient-petrol-cars-in-india/articleshow/66235779.cms — Times of India: 10 most fuel-efficient petrol cars in India

02.Analysis(Breaking down the problem)

Each car will have three main details:

1)Car ID (integer): A unique identification number.

2)Model Name (string): The name of the car model.

3)Mileage (float): Distance covered per litre (km/l).

These details will be stored in an array of structures, allowing multiple records to be handled easily.
Once the user enters the data, the program will sort cars in descending order of mileage, i.e., cars with the highest mileage appear first.

The sorting can be achieved using Bubble Sort or Insertion Sort algorithms.
If two cars have the same mileage, the order in which they were entered will remain unchanged (stable sorting).

Finally, the program will display the sorted list, showing each car's ID, model name, and mileage clearly.
This approach combines programming concepts with practical data organization, similar to how auto websites compare cars by efficiency.

Sources:

1. https://www.carwale.com/compare-cars/ — CarWale: Compare cars by specifications and mileage
2. https://www.v3cars.com/mileage-calculator — V3Cars: Car Mileage Comparison Tool

03.Ideate(Planning our solution)

Step 1: Define a structure

```
struct Car {
    int car_id;
    char model[50];
    float mileage;
};
```

Step 2: Input Section
Ask the user how many cars they want to enter, then take each car's details — ID, model name, and mileage.

Step 3: Sorting Section
Use Bubble Sort to arrange cars by mileage in descending order. If mileage values are equal, keep the same order (no unnecessary swapping).

Step 4: Output Section

After sorting, display all car records in an easy-to-read format showing ID, model name, and mileage.

This system enhances understanding of C programming fundamentals (structures, arrays, sorting) and demonstrates a real-world data application relevant to the automobile industry.

Algorithm

Algorithm

This is the algorithm for the program. We take the algorithm for users' understanding.

Step 1: Start

Step 2:
Declare a structure named Car having following members:
id = integer
brand = character array
name = character array
mileage = float

Step 3:

Declare variables:
n, i, j → integers
temp → structure variable of type Car
car[n] → array of structure Car

Step 4:
Read the number of cars n.

Step 5:
For each car i from 0 to n–1, do the following:
a) Read ID of the car
b) Read Brand name of the car
c) Read Model name of the car
d) Read Mileage of the car

Step 6:
Sort the car records in descending order of mileage using Bubble Sort:
For i = 0 to n–2
    For j = 0 to n–i–2
        If car[j].mileage < car[j+1].mileage then
Swap car[j] and car[j+1]

Step 7:
Display the heading:
"Car Details Sorted by Mileage (Highest to Lowest)"

Step 8:

Display the details of all cars after sorting:
ID
Brand
Name
Mileage

Step 9:
Stop

04.Build(Understanding the program)

```
#include <stdio.h>
#include <string.h>
struct Car {
```

```
    int id;
    char name[50], brand[50];
    float mileage;
};
int main() {
    int n, i, j;
    struct Car temp;

    printf("Enter the number of cars: ");
    scanf("%d", &n);

    struct Car car[n];

    for (i = 0; i < n; i++) {
        printf("\nEnter details for Car %d:\n", i + 1);
        printf("Car ID: ");
        scanf("%d", &car[i].id);
        printf("Brand: ");
        scanf("%s", car[i].brand);
        printf("Model Name: ");
        scanf("%s", car[i].name);
        printf("Mileage (in km/l): ");
        scanf("%f", &car[i].mileage);
    }

    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - i - 1; j++) {
            if (car[j].mileage < car[j + 1].mileage) {
                temp = car[j];
                car[j] = car[j + 1];
                car[j + 1] = temp;
            }
        }
    }
```

```
    printf("\n---------- CAR FUEL EFFICIENCY MANAGEMENT
SYSTEM ----------\n");
    printf("Car Details Sorted by Mileage (Highest to
Lowest):\n\n");
    printf("ID\tBrand\t\tModel\t\tMileage (km/l)\n");

    for (i = 0; i < n; i++) {
        printf("%d\t%-10s\t%-10s\t%.2f\n", car[i].id, car[i].brand,
car[i].name, car[i].mileage);
    }

    return 0;
}
```

05.Testing

By filling required inputs, The output will be displayed as follows:

```
Enter the number of cars: 3

Enter details for Car 1
Car ID: 101
Brand: Toyota
Model Name: Fortuner
Mileage (in km/l): 13

Enter details for Car 2
Car ID: 102
Brand: Honda
Model Name: City
Mileage (in km/l): 17

Enter details for Car 3
Car ID: 103
Brand: Tata
```

Model Name: Nexon
Mileage (in km/l): 16

CAR FUEL EFFICIENCY MANAGEMENT SYSTEM (SORTED LIST)

ID	Brand	Model	Mileage (km/l)
102	Honda	City	17.00
103	Tata	Nexon	16.00
101	Toyota	Fortuner	13.00

06.Conclusion

This program successfully manages and compares the fuel efficiency of different cars using structures and arrays in C.
It reads each car's details such as ID, brand, model, and mileage, and displays them in descending order of mileage.
The implementation helps in analyzing which car gives the best performance in terms of fuel economy.
By using a simple sorting logic, the program efficiently organizes data for quick comparison.
It demonstrates the practical use of arrays, structures, and sorting techniques in real-life applications.
Hence, the program fulfills the objective of managing car data in an organized and efficient way.

07.Implementation