Course-CSC 1012 Introduction to Computer Programming

REPORT ON LOGISTICS MANAGEMENT SYSTEM (C PROGRAM)

[Document subtitle]

Name:-Sutharshan Mohanaruby
Index number :-AS20240600

Program Overview

This C program is designed to manage logistics operations such as handling cities, managing distances between cities, viewing vehicle information, and preparing for delivery cost calculations.
It uses **modular programming**, meaning each task is handled by a separate **function**, which makes the code organized, readable, and easy to maintain.

i. cityManagement

ii. distanceManagement

iii. vehicleManagement

iv. requestHandling

v. calculations

vi. performanceReport

How the System Runs

User can choose Manage city or Manage distanace or make deliveries.

During the delivery part

- User select source and destination city
- Select vehicle
- Input weights
- System calculations
    1. Delivery cost
    2. Fuel usage+Fuel cost
    3. Time required
    4. Operatinal cost+Profit+final charge
- User can see the report

Function Usage

1.Main()

```c
int main()
{
    int choice;
do
{
    menu();
    printf("Enter your choice:");
    scanf("%d", &choice);

    switch (choice)
    {
      case 1:
          citymanagement();
          break;
      case 2:
          distancemanagement();
          break;
      case 3:
           vehiclemanagement();
           break;
     case 4:
          requesthandling();
          break;
     case 5:
          calculations();
          break;
     case 6:
          deliveryrecords();
          break;
     case 7:
         performancereport();
         break;
     case 8:
         printf("Exiting program..\n");
         break;
     default:
         printf("Invalid choice!\n");
    }
}
while (choice != 8);
return 0;
```

Purpose :- Acts as the entry point of the program. Displays the menu repeatedly and calls other functions based on user choice.

2.menu()

```c
void menu()
{

    printf("\n\t\t LOGISTICS MANAGEMENT\t\t\n");
    printf("1.Citymanagement\n");
    printf("2.Distancemanagement\n");
    printf("3.Vehiclemanagement\n");
    printf("4.Requesthandling\n");
    printf("5.Calculations\n");
    printf("6.Deliveryrecords\n");
    printf("7.Performancereport\n");
    printf("8.Exit\n");

}
```

Purpose :- Displays the list of available options for the user.It Improves readability by separating display logic from main function.

3.cityManagement()

```c
void citymanagement()
{
    int choice,n,find,i,j;
    char cities[50],name[50],newname[50];
    printf("1.Add city\n");
    printf("2.Rename city\n");
    printf("3.Remove city\n");
    printf("Enter your choice");
    scanf("%d",&choice);
    if(choice==1)
    {
        printf("How many cities do you want to add?");
        scanf("%d",&n);
        if(cityCount+n>MAX_CITIES)
        {
            printf("space not enough");
            citymanagement();
        }
        for (i=0;i<n;n++)
        {
            printf("Enter city name:");
            scanf("%s" ,cities[cityCount]);
            cityCount++;
            printf("City added successfully!\n");
        }
    }

    else if(choice==2)
    {
        printf("Enter city name to rename:");
        scanf("%s",name );
        printf("Enter new name for %s:",name);
        scanf("%s",newname);
```

```
        scanf("%s",newname);
        for (i = 0; i < cityCount; i++)
        {
          if (strcmp(cities[i], name) == 0)
          {
            strcpy(cities[i], newname);
            printf("City renamed successfully!\n");
            return;
          }
          else
          {
             printf("City not found!\n");
          }
        }


    }

    else if(choice==3)
    {
        printf("Enter city index to remove:");
        scanf("%d",&i);
        if (i >= 0 && i < cityCount)
        {
            for (int j = i; j < cityCount - 1; j++)
            {
                strcpy(cities[j], cities[j + 1]);
            }
            cityCount--;
            printf("City removed!\n");
        }
        else
        {
          printf("Invalid index!\n");
        }
        break;
    }
```

Purpose :- Handles operations related to **adding**, **renaming**, and **removing cities**.

- Uses arrays to store city names.
- Allows the user to add new cities up to a maximum limit.
- Supports renaming a selected city and removing an existing one by index.
- Demonstrates use of **loops**, **strings**, and **conditional statements**.

5.distanceManagement()

```
void distancemanagement()
{
    if(cityCount<2)
    {
        printf("Not enough Cities Add min 2 cities");
        return;
    }
    int choice,i,j;
    do
    {
        printf("1. Enter or Edit Distances\n");
        printf("2. View Distance Table\n");
        printf("Enter your choice: ");
        scanf("%d",&choice);

    switch(choice)
    {
      case 1:
           printf("\nEnter distances between cities (in km)\n");
           for(i=0; i<cityCount; i++)
           {
              for(j=i+1; j<cityCount; j++)
              {
                 printf("Distance from %s to %s : ",cities[i],cities[j]);
                 scanf("%d",&distance[i][j]);
                 distance[j][i] = distance[i][j];


              }
           }
           printf("\nAll distances saved\n");
           break;
      case 2:
          printf("\nDistance Table:\n");
          for (i = 0; i < cityCount; i++)
          {
             for (j = 0; j < cityCount; j++)
             {
               printf("%4d ", distance[i][j]);
             }
             printf("\n");
          `
```

```
      case 2:
           printf("\nDistance Table:\n");
           for (i = 0; i < cityCount; i++)
           {
              for (j = 0; j < cityCount; j++)
              {
                printf("%4d ", distance[i][j]);
              }
              printf("\n");
           }
           break;



      default:
        printf("Invalid choice!\n");



    }

    }
    while(choice!=0);
}
```

Purpose :- Stores and views the **distance between cities**

- Uses a **2D array** `distance[i][j]` to store distances.
- Allows the user to **enter or edit distances** between every pair of cities.
- Provides an option to **view the distance table** in a matrix format.

5.vehicleManagement()

```
void vehiclemanagement()
{
    printf("Type\tCapacity(kg)\tRate per km(LKR)\tAvg Speed(km/h)\Fuel Efficiency(km/l)\n"),
    printf("Van\t1000\t\t30\t\t60\t\t12\n");
    printf("Truck\t5000\t\t40\t\t50\t\t6\n");
    printf("Lorry\t10000\t\t80\t\t45\t\t4\n");

}
```

Purpose :- Displays details about available **vehicles** used for delivery.

- Lists vehicle types, their capacities, rates, speeds, and fuel efficiencies.
- Helps the user understand which vehicle is suitable for different weights.

6.requestHandling()

```
void requesthandling()
{
    int weight,vehicleType,city1,city2;
    if (cityCount < 2)
    {
       printf("\nAdd at least 2 cities and distances first!\n");
       return;
    }

    printf("\nEnter source city index: ");
    scanf("%d", &city1);
    printf("Enter destination city index: ");
    scanf("%d", &city2);

    if(city1==city2)
    {
        printf("Distance not set between these cities\n");
        return;
    }
    printf("Enter weight (kg):");
    scanf("%d", &weight);
    printf("Select Vehicle:\n");
    printf("1.Van(100kg)\n2.Truck(5000kg)\n3Lorry(10000kg)\n");
    printf("Enter vehicle type(1-3):");
    scanf("%d",&vehicleType);

    if (weight > vehicleCapacity[vehicleType])
    {
      printf("Weight exceeds vehicle capacity!\n");
      return;
    }
}
```

Purpose :- It will be used to take user delivery requests like pickup city, destination, weight, and vehicle type.

7. calculations()

```
void calculations()
{
    int dist,vehicleRate[vehicleType],city1,city2,weight;
    float baseCost,fuelUsed,fuelCost,totalCost,profit,customerCharge,time,vehicleSpeed[
    char vehicleEfficiency[];

    baseCost = dist * vehicleRate[vechileType] * (1 + (float)weight / 10000);
    fuelUsed = (float)dist / vehicleEfficiency[vehicleType];
    fuelCost = fuelUsed * FUEL_PRICE;
    totalCost = baseCost + fuelCost;
    profit = baseCost * 0.25;
    customerCharge = totalCost + profit;
    time = (float)dist / vehicleSpeed[vehicleType];
```

Purpose :- It will calculate total cost, profit, and fuel consumption.

- Distance(D)
- Weight(W)
- Rate per km(R)
- Vehicle speed(S)
- Efficiency (E)
- Fuel price (F)

Calculation	Formula
1.Delivery cost	Cost=D*R*(1+W*1/10000)
2.Estimated Delivery Time(hours)	Time=D/S
3.Fuel consumption	FuelUsed=D/E
4.Fuel cost	FuelCost=FuelUsed*F
5.Total Operatinal Cost	TotalCost=Cost+FuelCost
6.Profit Calculation	Profit=(Cost*0.25)
7.Final Charge	TotalCost+Profit

8. deliveryrecords()

```c
int deliveryrecords(int city1, int city2, int dist, int vehicleType, int weight, float baseCost,
{

    printf("\n========== DELIVERY COST ESTIMATION ==========\n");
    printf("From: %s\nTo: %s\n", cities[city1], cities[city2]);
    printf("Distance: %d km\n", dist);
    printf("Vehicle: %s\n", vehicleNames[vehicleType]);
    printf("Weight: %d kg\n", weight);
    printf("-----------------------------------------\n");
    printf("Base Cost: %.2f LKR\n", baseCost);
    printf("Fuel Used: %.2f L\n", fuelUsed);
    printf("Fuel Cost: %.2f LKR\n", fuelCost);
    printf("Operational Cost: %.2f LKR\n", totalCost);
    printf("Profit (25%%): %.2f LKR\n", profit);
    printf("Customer Charge: %.2f LKR\n", customerCharge);
    printf("Estimated Time: %.2f hours\n", time);
    printf("=========================================\n");


}
```

Purpose :- Show the last bill

9. performanceReport()

```c
void performancereport()
{
   if (totalDeliveries == 0)
   {
      printf("No deliveries completed yet!\n");
      return;
   }
   printf("Total Deliveries Completed : %d\n", totalDeliveries);
   printf("Total Distance Covered     : %.2f km\n", totalDistance);
   printf("Average Distance/Delivery  : %.2f km\n", totalDistance / totalDeliveries);
   printf("Total Revenue              : %.2f LKR\n", totalRevenue);
   printf("Total Profit               : %.2f LKR\n", totalProfit);
}
```

Purpose :- Displays summary statistics about the company's performance.

- In the future, this will show:
  - Total number of deliveries
  - Total revenue and profit
  - Average distance per delivery
- It currently acts as a placeholder showing that the program is expandable

GitHub Repository:[https://github.com/SutharshanMohanaruby/C-project-AS20240600]