

Documentation

Koapch Artem

Exercise: Car fleet manager.

To begin with inputs lists. Wiser, easier and more practically if we get two documents from fuel station. In the first document the first important thing that we get the **indexes of the cars**, after that the second important thing that we have **unique number of each car** (Ex. KKK111), third get the mark and the model.

Example of the first list,



```
*Cars.txt – Блокнот
Файл  Правка  Формат  Вид  Справка
1 KKD123 Toyota Avensis
2 TUF543 Kia Rio
3 YTF432 Volkswagen Polo
4 HTE570 Seat Ibiza
5 PFH619 Mercedes-Benz S350
6 YFL831 BMW M5
7 IFZ238 Volkswagen Golf
8 FKG642 Fiat 500C
9 TP0777 Mercedes-Benz G63
10 LCD234 Toyota Camry
11 UTG476 BMW M2
12 OFR249 Porsche Taycan
13 FPR540 Kia Picanto
14 AAS004 Lamborghini Urus
15 IFT583 Seat Leon

...
```

The second list will be only with indexes, dates of refueling (as example yyyy.mm.dd), liters and mileage of on the day of refueling. In this case for us important the indexes of the cars (in the first list it was the number of the car).

And in this case, we are planning to use two dimensional linked list from our two inputs lists.

Some different and important moments for lists and working with them,

1. For us the separator that is space between two piece of date information, that is why it should be only one space
2. In the second list, we don't need and don't have sequence of the date (for ex. 2021.08.21 that is don't guarantee that next date will be 2021.08.22)
3. Our cars refueling, when there is no petrol/gas left in the tank

Example of the second list,

Refueling.txt – Блокнот

Файл	Правка	Формат	Вид	Справка
1	2021.10.12	22	62800	
3	2021.10.12	13	2500	
2	2021.10.13	25	82675	
7	2021.11.11	42	98520	
5	2021.12.13	6	8422	
6	2021.10.20	15	35946	
1	2021.10.18	20	63008	
2	2021.12.19	17	103598	
5	2021.12.15	11	8525	
7	2021.12.28	6	110222	
1	2021.10.22	10	63252	
3	2021.10.14	15	2655	
1	2021.10.23	9	63332	
10	2021.09.29	25	200523	
10	2021.12.15	33	250511	
7	2021.12.29	15	110309	
15	2021.09.03	12	26378	
3	2021.10.17	2823		
7	2021.12.31	26	110410	
...				

Structures and Functions:

- Struct Car
- Struct Date
- Struct Fuel
- Struct FuelNode
- Struct CarNode
- CarNode* AddNodeToCarList(CarNode* hp, Car* newCar)
- void PrintCar(CarNode* car)
- void PrintFuel(Fuel* fuel)
- void PrintCars(CarNode* head) //Function to print our nodes
- void PrintAll(CarNode* head)
- CarNode* FindCarNodeById(CarNode* head, int id)
- void AddFuelNodeToCarNode(CarNode* car, Fuel newFuel)
- void LoadAllFuelsFromFile(CarNode* head, FILE* fp)
- float getAverageConsumptionByCarId(CarNode* head, int id)
- void PrintConsumptionByCar(CarNode* head, CarNode* car)
- void PrintAverageConsumptionForAllCars(CarNode* head)
- int Services(CarNode* head, int id)
- void AddNewReFuel(CarNode* head, int id)
- void SaveToFile(CarNode* head, FuelNode*fn)
- int userInput()
- char SelMenu(int sel, CarNode* CarsHead)
- int main()

Secondly, we are planning implement the following statements:

1. Should load existing database into memory or create a new if we don't have database.
2. Add new re-fueling entry into the database,
3. Save the re-fuelling database (from the memory) into a file.

To complete the tasks we can follow the rules of lists in our text!

4. For do this statement we need minimum 3 strings about refueling of the car, because first mention of refueling the car don't get for us necessary information. Also, if we have more than 3 dates of refueling we should see dates with the smallest interval. After that, we can get the exact value!
5. List all cars that need to go to service in the near future. The machine needs service if the mileage of the car > 20000 km or in the future it predictably will be more than 20000 km in the next 3 months.

In the next step do the block diagram for our program:

