Dealership Inventory

By Draven McConathy

Program Design Document

1. Problem Definition

Auto dealerships need a program to allow them to reliably track their current inventory of vehicles in their possession. Design a program that allows them to keep a running inventory.

1. Program Analysis

2.1 User requirements

While using the program the dealership needs to be able to add cars to the inventory including details about the cars including the year, make, model, color, class (sedan, truck, van), fuel-type, and price as well as the total amount of cars in stock. After entering car data, the user should have an option to display all the cars currently in inventory. The format for displayed cars should look like so:

Dealership Inventory

Year Make model Color Class Fuel-Type Price

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

xxxx xxxxxxx xxxxxxx xxxxxxx xxxxxxx xxxxxxxxxx $99,999

-

-

The user must also be able to search to see if a particular vehicle is in inventory. Once the User is done using the program, the Cars added to the inventory will be saved to a carsFile to be used for later uses of the program.

* 1. Choice of language features, variables, and formulas

The C++ programming language was selected because that is the language we are currently using in this class, and thus is the language I am trying to better understand.

The C++ standard libraries selected are as follows:

|  |  |
| --- | --- |
| **Library** | **Purpose** |
| iostream | Keyboard input and output objects and functions |
| fstream | Disk file read/write objects and functions |
| Iomanip | Functions for formatting output |

**Input:** If the program is being used for the first time the input is purely user keyboard inputs, which will store data into a car structure which will contain the year, make, model, color, fuel-type and price. Each object of the car type will then be stored in an array of car objects. After the first run of the program there will also be **carFile** object of type ifstream used as input.

**Output:** Dealership inventory in a tabular form as described above. We need the **iomanip** library of the C++ programming language for tabular column output.

All of the data for each car will be stored in a car structure data type, which in turn will be stored in a array of car objects.

Here is a summary of the variables required in this program:

|  |  |  |
| --- | --- | --- |
| **Variable/Object** | **Type** | **Purpose** |
| dealershipInventory | AutoDealer | Keeps track of each individual car in the dealership as well as how many cars there are total |
| menuInput | Int | Used to store user menu choice |

* 1. **Data Structure**

The program will be organized into a class called **AutoDealer.** The data members of the class will be the Cars array and the total number of cars in the lot. The functions of the program will make up the methods of the class.

|  |
| --- |
| AutoDealer |
| - carInventory : array of Cars  - totalCars : int |
| + VoteCounter() : constructor  + readCarsFile() : void  + inputCarInfo() : void  + searchCar(tempCar : Car) : Car  + writeCarsFile() : void |

1. **Design**

**Algorithm Design**

The first task to be performed by the program is to read the current cars in the lot from the input file into the array of Cars, while also totaling the total number of cars in the file. From there the user may then see the current cars in inventory, add cars to the inventory, search for a particular car, and save the current cars in stock to the carsFile. These tasks are placed in separate member functions of the class so the main function only calls the member functions.

**Function Main**

Here is a list of the tasks that the program will perform:

1. Read the current cars into the array carInventory

2. Allow user to add cars to the array carInventory

3. Allow user to search for car in carInventory

4. Allow user to print all current cars in the carInventory

5. Save all cars in carInventory array to carsFile

**Main Function Pseudocode**

Declare the AutoDealer object

Call member function readCarsFile

Do the following

Display menu of user options

Get keyboard input of menuInput

If menuInput equals ‘1’ then

Call member function inputCarInfo

Else if menuInput equals ‘2’ then

Call member function searchCar

Else if menuInput equals ‘3’ then

Call member function displayCars

Else if menuInput equals ‘4’ then

Call member function writeCarsFile

Else

Display message, “Invalid menu option selected”

While option is not ‘4’

**Function readCarsFile**

This reads the information of various cars into a Cars array one car at a time. This file is opened in the **readCarsFile** function. This function has no parameters and returns no value.

**readCarsFile Function Pseudocode**

Declare fstream carFile object

Declare index variable and initialize to 0

Open car file

If opening car file fails

Return

Read the first record from the file

While not at the end of the file, store the records in carInventory[index]

Store year at carInventory.year[index]

Store make at carInventory.make[index]

Store model at carInventory.model[index]

Store color at carInventory.color[index]

Store class at carInventory.class[index]

Store fuel-type at carInventory.fuelType[index]

Store price at carInventory.price[index]

Read the next record from the file

Close car file

Set totalCars data member to ending index + 1

**Function inputCarInfo**

This gets user input for a particular car and continues receiving input for additional cars until the user says stop. This function accepts no parameters and returns no value.

**inputCarInfo Function Pseudocode**

Declare index variable and initialize it to member data totalCars - 1

Declare again variable and initialize it to ‘Y’

While again equals ‘Y’

Index++;

Get input for carInventory.year[index]

Get input for carInventory.make[index]

Get input for carInventory.model[index]

Get input for carInventory.color[index]

Get input for carInventory.class[index]

Get input for carInventory.fuelType[index]

Get input for carInventory.price[index]

Get input for again

Make totalCars equal to index + 1

**Function searchCar**

This gets user input for a particular make, model, color and class and prints the first car in the array that matches all the user input. This function accepts no parameters and returns no value.

**seachCar Function Pseudocode**

Declare searchMake, searchModel, searchColor and searchClass c-strings

Get input for searchMake

Get input for searchModel

Get input for searchColor

Get input for searchClass

Declare index and initialize to 0

For each car in carInventory

If searchmake is equal to carInventory.make[index] and searchModel is equal to carInventory.model[index] and searchColor is equal to carInventory.color[index] and searchClass is equal to carInventory.class[index]

Display the matching structure at carInventory[index]

Else

Display that there is no matching car in stock

Store year at carInventory[index]

Store make at carInventory[index]

Store model at carInventory[index]

Store color at carInventory[index]

Store class at carInventory[index]

Store fuel-type at carInventory[index]

Store price at carInventory[index]