AutoDealer

By Travis Crumb

Program Design Document

1. Problem definition

CS215 programming assignment 4 part-2 requires that a student write a program that simulates a car dealership inventory program utilizing a structure representing a car, a class defining various functions that can be performed on dealership data, and a main function to implement these features.

2. Problem Analysis

2.1 User Requirements

The auto dealer program shall read vehicle data from a file to add to an array that represents the inventory. The program shall offer a menu of options that deliver the following functionality.

1. User may select to add vehicles to inventory. The user will be asked to give the number of vehicles to be entered, and will be prompted to enter each of the Car structures data members. The user wants the following data member representations as part of the Car struct.
    a. Car make and model
    b. Manufacture year
    c. Vehicle color
    d. Vehicle class
    e. Vehicle fuel type
    f. Vehicle price

2. The user may select to search for a specific vehicle. The user will be prompted to enter the following information.
    a. Vehicle make and model
    b. Vehicle color
    c. Vehicle class

    The vehicle Inventory will be searched until a match is found, or there are no more vehicles to read.

3. The user may select to display the current inventory of vehicles. If this option is selected, an itemized list of each data member of the car structure should be displayed. The

program shall maintain a total of all prices. The format in which the information is displayed shall be as in the example below.

```
                    LEXINGTON, KY AUTO DEALER CAR INVENTORY
MAKE AND MODEL      YEAR      COLOR      CLASS      FUEL TYPE      PRICE
-----------------------------------------------------------------------------------------------------------
Ford Mustang        2005      Red        Coupe      Gasoline       5995.95
xxxxxxxxxxx         xxxx      xxxxx      xxxxx      xxxxxxxxx      xxxxxx

Total of all vehicles: $xxxxx.xx
```

The vehicle information displayed comes from an array of car structures, as such , vehicle information that appears in the inventory list can come from two places. It may come from a file that is read upon opening the program if that file contains any vehicle information, and\or it may come from user input given upon selection of option three.

4. The user may select to exit the program. At this point, current inventory should be written to a file and execution of program should be terminated.

2.2 Choice of language features, variables, and formulas

CS215 is a course that uses the C++ programming language, as such, the C++ programming language will be used to write this program.

The C++ standard libraries that will be used in this program are as follows:

Library	Purpose
iostream	Keyboard input, and output objects and functions
fstream	File read/write objects and functions
iomanip	Output formatting
string	Classes and functions for string objects

Input: File input will come from one file that may or may not contain vehicle inventory information. If the file has vehicle information, this information will be assigned to elements within an array of car structures. The array will be called **carStock**. File variable name will be **carFile** and will be of type **ifstream**. User input will be given in the **inputCarInfo()** function of the **AutoDealer** class, and will be assigned to elements of the **carStock** array.

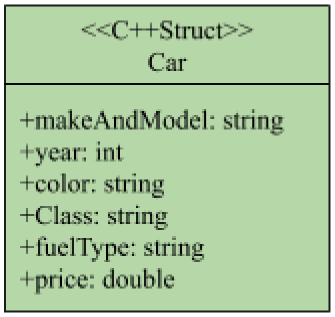
Output: Car inventory output will be given in the format described above if the user selects to display the car inventory. Upon selection to exit the program, any additions to or removals from the car inventory will be written to the **carFile** object of type **ofstream**.

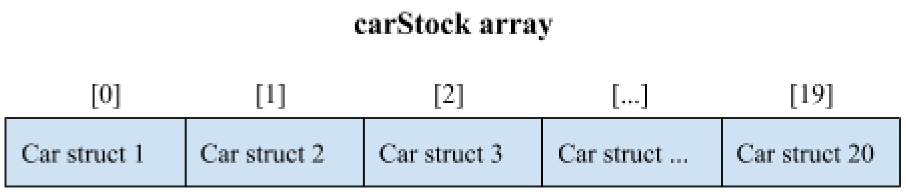
The output must be organized in such a way that when it is written to the **carFile** file object, that it is formatted as in the example below.

```
Ford Taurus
     2018 Red Sedan Gasoline 29295.08
Chevrolet Bolt Premier
     2019 Gray Sedan Electric 41020.95
Toyota Rav 4
     2017 Brown SUV gasoline 25500.63
Ford F150
     2009 Black Truck Diesel 32427.76
Dodge Charger
     2016 Yellow Sports Gasoline 35000.00
Mercedes Benz Metris Passenger
     2015 White Minivan Gasoline 50132.38
```

This style of output formatting is critical as this is the same file object that is read upon opening\reopening of the program. Any variation in formatting could lead to unpredictable results, such as infinite loops, or incorrect\unreliable population of the **carStock** array data.

Because the individual vehicles within the data set contain different data types, it is best to represent the vehicles in a C++ struct, to be called **Car**, and store these structs in an array of type **Car**. Below is a UML representation of the **Car** struct, along with a diagram of the **carStock** array, which will have a maximum of 20 elements of type **Car**.





Summary of variables required in this program:

Variable/object	Type	Purpose
cars	AutoDealer	Object containing vehicle information and member Functions for inventory processing
user_choice	int	Used to store user menu choice

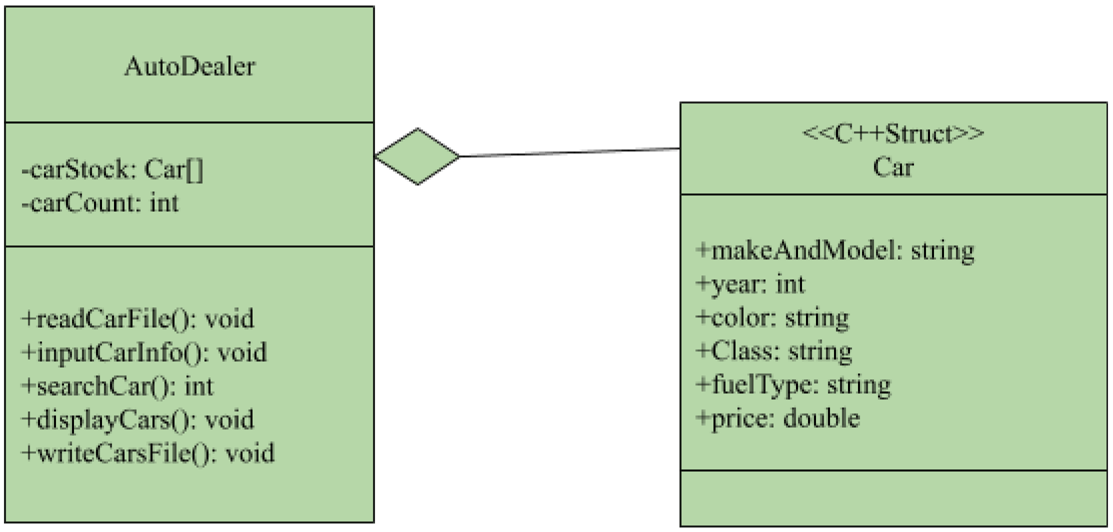
Formulas

Total price of all vehicles will be calculated in the **displayCars()** member function of the **AutoDealer** class using the following formula:

```
total_of_prices += carStock[index].price;
```

2.3 Data Structure

The program will be organized into a class called **AutoDealer.** The data members of the class will be an array of 20 **Car** structs called **carStock**, and a counter called **carCount** that keeps a total of the vehicles in inventory. In addition, the class will contain five member functions to perform various processes on the inventory data. Below is a class diagram.



3. Design

Algorithm Design

Upon opening the program, an AutoDealer object will be created. The objects **readCarFile()** member function will be called to populate the **carStock** array with any vehicle data that that file object contains. The user then be prompted with a menu that provides options to enter a new vehicle, search for a vehicle, display current inventory, or exit the program. Once the user in done with their processing of the inventory data and selects to exit the program, the old car file will be overwritten with the new data the user provides and execution of the program will be terminated.

Function main tasklist

1. Create *AutoDealer* object
2. Read car information into *carStock* array.
3. Prompt user with menu options
4. Perform user selected option
5. Repeat 3-4 until user selects to exit
6. Write new data to file
7. Terminate program

Main Pseudocode

```
1. Create AutoDealer Object
2. Call readCarfile member function to populate carStock array
3. Enter do-while loop until user selects to exit program
4. If user selects "Enter a new vehicle"
    a. Call inputCarInfo member function
5. If user selects "Search for a vehicle"
    a. Call searchCar function
6. If user selects "Display current inventory"
    a. Call displayCars member function
7. If user selects "Exit Program"
    a. Call writeCarsFile member function
8. Exit do-while loop
9. Display Thank you message to user
10. Terminate execution of program
```

Function get_user_choice () - contained in source.cpp

This function will present the user with a menu of four options to choose from. The user's selection will be validated to be an option that appears on the menu, then the appropriate value will be assigned to the reference variable passed in that represents the users choice.

get_user_choice pseudocode

```
1. Display menu to user
2. Prompt user to select from menu
3. Validate user choice selection
4. Reprompt if needed
5. Assign value to choice reference variable
```

Function readCarFile()- member of AutoDealer class

This function will read data from a file that will be used to populate the *carStock* array data member.

readCarFile pseudocode

```
1. Open input file object
2. Set count to 0
3. While carCount data member is <20 and the end of the file hasn't been reached
    a. Process file and assign data to carStock[count].dataMember
    b. Increment carCount by one
    c. Increment count by one
4. Close input file object
```

Function inputCarInfo() - member of AutoDealer class

This function will prompt the user to enter vehicle data and add that date to the *carStock* array

inputCarInfo pseudocode

```
1. Prompt user for quantity of vehicles to enter
        i. If there is space in carStock array
            1. Prompt user for vehicle information
            2. Add information to carStock
        ii. Otherwise let user know that no more vehicles can be entered
```

Function searchCar - member of AutoDealer class

This function will prompt the user for information of vehicle being searched for. If the vehicle is found in the *carStock* array, it's index position will be returned, otherwise -1 will be returned.

searchCar pseudocode

```
1. Prompt user to enter vehicle information
2. Search the carStock array for a match
3. If a match is found
    a. Return index position
4. Otherwise return
    a. Return -1
```

Function displayCars() - member of AutoDealer class

This function will process the *carStock* array and display an itemized list of vehicles currently in stock. It will let the user know if there are no vehicles to display if the *carStock* array is empty.

displayCars pseudocode

```
1. If there are no cars in inventory
    a. Let user know there are no cars to display

2. Otherwise
    a. Initialize total_of_prices to 0
    b. User for loop to calculate total_of_prices of current inventory
    c. Use for loop to display current inventory
```

Function writeCarsFile() - member of AutoDealer class

This function will determine if there are any cars in inventory to be written to file. If there are, and output file object will be opened and the *carStock* array will be processed to write current inventory to a file.

writeCarsFile pseudocode

```
1. If there are no cars in inventory
    a. Let user know there are no cars to write to file
2. Otherwise
    a. Open output file object
    b. Use for loop to process current inventory and write to file
    c. Close output file object
```

4. Implementation

This program will be developed on Microsoft visual studio community using a PC running Windows 7 pro 64 bit, and intel i5-4690k processor, and 16gb of ram. Test data will be written into a file called *cars.txt*. The file will be formatted with test data as follows:

```
Ford Mustang
        2005 red coupe gasoline 5395.95
Pontiac Firebird
        1996 Black Coupe Gasoline 8727.32
Dodge Viper
        2016 Blue Coupe Gasoline 119732.63
Ford Focus
```

2015 Silver Sedan Gasoline 8995.47
Tesla Model 3
2018 Red Sedan Electric 49934.29