**Objective**

Create (and update) a program that will manipulate variables via their value and their memory.

**Program 1**

Create an array of 15 ints and place a random number into each slot in the array. Then loop through that array and print out the value and memory address of each item.

**Program 2**

Create a function with parameters for both an int and an int pointer. This function should print out the int and int pointer parameters (do not use any operators on the pointer as we just want to see the memory address). This function should not loop as it is printing only a single element.

Then, similar to Program 1, create another array of 15 ints and place a random number into each slot in the array. Loop through the array and pass each item in the array and its memory address to the function you created.

**Program 3**

**Part 1**

Declare an enum named EnumColorDefinition that contains at least two values (example: Red, Blue). Make sure to declare this at the top of the file because C++ code defines top-down. Then, create a struct named Car that has the following members:  
        Make - char array of size 32  
            Model – char array of size 32  
            Year – int  
            Mileage – int  
            Color - EnumColorDefinition

Declare an array of 3 Car and ask the user for all of the information (all 5 members) of each car in the array by using a loop. Remember to use cin.getline for the make and model input instead of cin >>. You’ll also need to clear and ignore the cin buffer after each numeric input to avoid breaking the next string input. You can ask for the enum input by using the following example:

int colorInput;

cin >> colorInput;

cararray[0].Color = (EnumColorDefinition)colorInput;

Finally, loop through the array of cars and display all information for each car to the screen. *Hint: You’ll need to use a switch on the color of the car to print the associated word manually as the Color member itself is, internally, nothing more than an int.*

Example Output

Car 1 – 2003 Gray Ford Mustang with 4000 miles

Car 2 – 2016 White Ford Fusion with 567 miles

Car 3 - 2019 Silver Tesla Cybertruck with 127204 miles

**Part 2**

Define the following function (remember to declare it at the top of the file, but below the definitions of Car and EnumColorDefinition). It should simply change the Color member of the provided car pointer parameter to the provided color. Remember, the car pointer parameter is not an array, so you should not have any loops inside this function. Implement the function into your main code in whatever way you see fit.

**C++**

|  |
| --- |
| void repaintCar(Car\* ptrCar, EnumColorDefinition newcolor); |

**Part 3**

Let's move our display logic into a separate function. We will make two versions. One will receive a normal Car and the other will receive a Car pointer. Implement both functions and update your printing loop from Part 1 to instead call these functions. Neither of these functions should attempt to loop as they are priting only one car per call. *Hint: You can get away with implementing only one of these functions and simply call (in a correct manner) the completed version from the incomplete version.*

**C++**

|  |
| --- |
| void printCar(Car c); |
| void printCarPointer(Car\* ptrCar); |
|  |

Example Output

Printing cars…  
Car 1 – 2003 Gray Ford Mustang with 4000 miles

Car 2 – 2016 White Ford Fusion with 567 miles

Car 3 - 2019 Silver Tesla Cybertruck with 127204 miles

Printing car pointers…  
Car 1 – 2003 Gray Ford Mustang with 4000 miles

Car 2 – 2016 White Ford Fusion with 567 miles

Car 3 - 2019 Silver Tesla Cybertruck with 127204 miles

**Part 4**

Define the following function. It should increase the Mileage member of the provided car pointer parameter by the provided amount. Implement the function into your main code in whatever way you see fit.