Pointers and Arrays

This exercise is different from our usual exercises. You don't have so much a problem to solve by creating a program but rather some things to understand about the programming language you are learning. To do so, we are going to write little programs which illustrate the relation between what we write and how it executes.

What's in the name (of an array variable)?

We want to write a small program which will display for us the location in memory of the elements of an array of 3 integers. Your program will only contain a main function and declare a local variable *myarray* as an array of three integers. You will initialize this array with integers values { 1,2,3 }. We want then to display on the screen the following information;

- Memory address of element *myarray[0]*, *myarray[1]* and *myarray[2]*. Use a *printf* with a %p (pointer) type converter.
- The value of *myarray* displayed also with %p

Here is a sample of what the output of your program should look like, the <address here> will be of course replaced by the actual address on your system;

```
Displaying basic information on our array;
    &myarray[0]     = <address here>
    &myarray[1]     = <address here>
    &myarray[2]     = <address here>
    myarray         = <address here>
```

Use the box below to take notes in this word file about what can you deduce from this little experience about the way the language understand and evaluates *myarray*?

Pointer arithmetic and subtraction

Have your program display the difference, as an integer, between the memory locations of element index 0 and element index 1

```
printf( "\tFirst address \t = %p\n", &myarray[1] );
printf( "\tFirst address \t = %p\n", &myarray[1] );
printf( "\tDifference \t = %d\n", &myarray[1] - &myarray[0] );
```

Here is a sample of what your output should look like;

```
Let's play with pointers arithmetic (subtract);
    First address [1]          = <address here>
    Second address [0]         = <address here>
    Difference [1] - [0]       = <address here>
```

What is the value that you expected, what is the value that your program printed, why are they different? What do they represent (units)?

Assign to a pointer on *int* named *p* the address of the element *myarray[2]*. You will then display on the screen, as %p, the following values;

- (*p* - 1)
- (*p* - 2)
- (*p* - 3)

How do these values relate to the address of each element of *myarray*?

Are these the same as subtracting 1 from the hexadecimal value contained in *p*? Why so?

Pointer arithmetic and addition

Let's add something to the above mentioned program. We want now to iterate over the array and, for each element of the array, display both the contents (the *int* value) and the address of the variable.

To do so, we will use a *for* loop and a pointer *p*. The pointer will be initialized to the address at which the first element of our array starts (you have two ways to do so, make sure you understand and test both). We will then increment this pointer by using post incrementation and display its value and the value it points to at each iteration.

Here is a sample of what your output should look like;

What about pointer arithmetic with addition;
Iteration #0 p = <address here> *p = 1
Iteration #1 p = <address here> *p = 2
Iteration #2 p = <address here> *p = 3

What is the meaning of adding one unit to a pointer on int?

Modify your code so that it now works on an array of doubles initialized to {1.0, 2.0, 3.0} instead of *int*.
What is now the meaning of adding one unit to a pointer on double?