# Pointers Worksheet

This worksheet aims to cover the basics of pointers and arrays and how the two interact.  
Prerequisites: Read Chapter 3.2.  
LO: learn about pointers and their uses

## Tasks:

1. Create an array of **integer**s
2. Using a for loop fill this array with numbers -100 to 100
3. Create an array of **double**s
4. Using a for loop fill this with numbers -100.0 to 100.0 in increments of 0.1
5. Create an array of **short int**s - -100 to 100
6. Create an array of chars and fill with *lorem ipsum* to around 200 characters.
7. Create a struct (aa) that contains two integers (a and b)
8. Create a second struct (bb) that contains two **aa**s
9. Create an array of 10 **bb**s

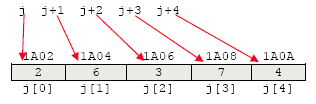
You should now have 5 arrays filled with data ready for use.

We can add to or subtract from pointer variables. However, we must remember that adding a number to a pointer does not necessarily add that many bytes but adds that number of storage units. In this text we will assume that characters are stored in one byte, integers in two and floats in four (you can use the sizeof() function to view the real size of the data type stored in your system). Bytes are used to measure the amount or size of memory.

Ref: https://www.tenouk.com/clabworksheet/labworksheet11.html

When arrays are defined, the array name actually holds the starting address of the array. This string address is fixed because we can’t move the array to some other location in memory that easily once the array is created.

**int j[] = {2, 6, 3, 7, 4};**



You can find the *address of* j, j[0] (both should be the same) and of the third element in the array:

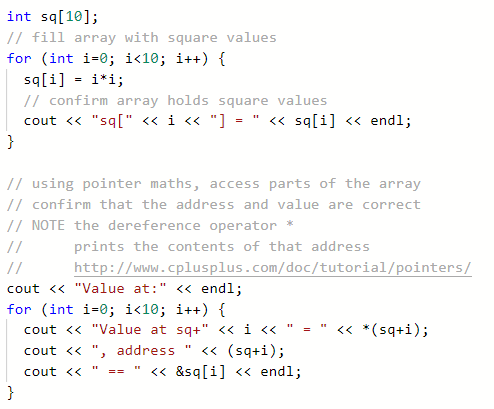
**p = &j;  
q = &j[0];  
r = &j[2];**

## Tasks:

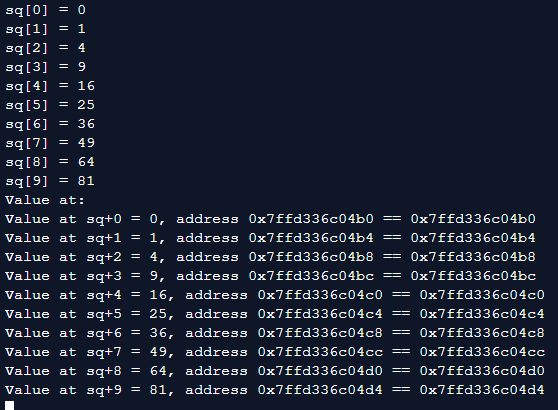
1. Research and calculate the amount of memory used in each of your arrays (maybe using sizeof.)
2. Use pointer arithmetic to display every other item in the **integer** array (task 1) up to 50 items.
3. Use pointer arithmetic to show nth memory address. User selected n.
4. Using a pointer that points at the first item of the array, show what the pointer++ is doing. Calculate how many bits or bytes it is being moved. How does pointer++ differ from integer++.
5. Allow the user to choose to show even or odd values. Allow them to select a start position and count. E.g. I choose the double array, I want to print odd values starting at -51 for 10 values. I expect to see:

* array name
* the value
* the memory location
* the offset from the first memory location in bytes

# Appendix



Outputs:



Struct help:

