**Arrays and Functions Lecture – Chapter 5 & 6 – Lab 56**

**Array** – Do you want to write 100 individual declarations to create 100 variables, or should there be included in C++ a feature to declare multiple variables with ONE declarations ?

**Declaring an Static array** – note all elements in an array declaration are of the same type, and of a fixed/**’Static’** size

Syntax: Datatype arrayName [number of elements]

Example: int age[100]; // This declare 100 elements all the same type, integer.

Example: char letter[50]; // This declares 50 elements all of the same type, char.

**Access** – Accessing an array element uses an INDEX. An index is an integer variable.

Access can be of two type. **DIRECT** or **SEQUENTIAL**

Example: age[0] DIRECTLY access the first element. Age[99] accesses element 100.

**Assignment** – To assign an array a value, use the array name and the index.

Example: age[1] = 10; // Assign 10 to the 2nd element of the array.

**Arrays – Behind the scenes**

Arrays are allocated in a contiguous set of bytes in RAM.

The array name is an alias for the address of the first element in memory.

Example: age = 3 is the same as age[0] = 3;

The Index is really an *offset* from the first memory address.

Example: age[0] with index 0, has the address of the first byte.

Example: age[1] with the index of 1, has the address of the first byte plus 1 times 4 types .

4 because there are 4 bytes in an integer.

Example: letter[7 ]with the index of 7, the address of the first byte plus 7 time 1 byte.

1 because there is 1 byte per char.

*In other words, you need three pieces of information to gain access to an array element:*

* The starting address of the array or base address.
* The array element number or array index you wish to access.
* The data type size, in bytes, of the data type stored in the array. The size of the data type, along with the index, will be used to calculate an offset into the array from the address of the first element using the following formula:

address of array element = base address + (data type size \* array index)

**Arrays and loops – Sequential Access**

You can sequentially progress forward or backward through all elements in an array using a loop.

Example: for ( int i = 0, I < 100, i++) { age[i] = 0; } // Initialize all elements to zero.

**Array list Initialization**

When you declare an array, you can at the same time assign values, using a list inside {} braces.

Example: Double Temp[5] = { 0.1, 3.1, 2.7, 3.4, 5.0 };

**Arrays and Functions**

**Passing one element of an array**

Example: given a function fun(int x, char a) { …. Do something },

to pass individual element of an array as function arguments:

fun( age[2), letter[3) )

**Passing the entire array**

Example: Write a function header as follows

fun( const int Array [ ] ).

The [ ] brackets indicate a function is to be passes.

To use the function write – fun(ArrayName) such as fun(age)

Remember the arrayName really is an address. So you are actually passing the starting address of an array.

Problem: The compiler does not know how many elements are in your array.

Solution: Change the fun( const int Array [ ], int arraySIze ). Add an array size variable.

To use the new improved function: fun(arrayName, size) such as fun(age, 100).

**Array Boundary Problem**

Say you declared an array: anArray(5];

To directly access each element of the array, you would do:

anArray[0] // access 1st element

anArray[0] // access 2nd element

anArray[0] // access 3rd element

anArray[0] // access 4th element

anArray[0] // access 5th element

What would happen if you typed anArray[21] ?

Exercise 1. Write an integer array, intArray, of 20 elements.

Use the short list method to assign each element to zero. (Do not use a loop)

Research the internet to find the short cut for the list method.

What is the long method to the list method, what is the short method to the list method ?

Print out the arrays values.

Exercise 2. Write a char array, charArray, of 100 elements.

Use a for loop structure to assign each element to blank.

Assign ‘HELLO’ to the 3rd, 4th, 5th,6th and 7th elements respectively.

Print out the 100 arrays values

Exercise 3. Boundary Problem.

Write a char array, charArray2, of 100 elements.

Use the shortcut list method to set each element to space.

Assign ‘SUNNY’ to the 3rd, 4th, 5th,6th and 7th elements respectively.

Write a loop to print out array elements 0 to 500 (yes 500].

Explain your results.

Exercise 4. Passing an array in Function arguments.

Write program that has a double array, doubleArray, of 5 elements.

Write 4 functions: Enter5Elements, AddElements, MultiplyElements, Print Elements.

Have a prompt that asks the user to enter one number at a time. Check that an inputted number is

between negative 10.00 and positive 10.00, if not correct have the user retry.

Use each function and print out the results

Exercise 5. Explain how passing an array works. Draw a diagram and explain

Exercise 6. (12 Points)Write an program that takes a string of characters and check if the forward and backward version

in a string of characters is the same. This called a Palindrome.