**Lecture 11 - Chapter 6: Arrays – Wed Sept 27 or Thurs Sept 28**

**Announcements**

Reading:

* Chapter 6

Assignments:

* Due: Assignment #4
* Assign: Assignment #5 - due on **Oct 4** (MW class) or **Oct 5** (TR class) **(no late assignments accepted)**

**Today’s Goals**

1. Arrays

**Today’s Terminology 5 min**

**Terminology**

* Array
  + A data structure used to store a collection of values that are all the same type
* Index
  + Refers to a specific element within an array
  + Must be an integer or integer expression
  + The **first element** in an array is at index 0, the second element is at index 1, etc.
* Indexed Variable
  + Used to reference each element in an array
* Array Initializer
  + A statement where you define and initialize an array
* Out of Bounds Error
  + Attempting to access an element with an index outside the range of the array is a runtime error
* Off by One Error
  + Mistakenly referencing the first element in an array with the index 1, not index 0

**Arrays 20 min**

**Purpose**

* So far, all the types we have discussed (int, float, double, bool, char) store one value

int number = 50;

char littleCharacter = 'a';

double interestRate = 4.0;

* An array is a container that provides a way to store more than one value
* An array is called a "data structure"

**Why Use Arrays**

* Provides ability to store a fixed set of values of the ***same type***
* Can access that set of values through **one** variable name

**Defining Arrays**

* When defining an array, you are defining
  + What **type of elements** the array can store
  + **T**he **variable name** that will be a "reference" to the array
  + The **number of elements** the array will hold
* General form for declaring an array:

**datatype** **arrayNameVariable[n];**

Any type (int, float, double, , etc) Variable used to access the array elements

All elements within the array will be this type It is considered a "***reference***" variable

**Example**

* Define an array that holds 5 integers

// Create an array that can store 5 integers

**int** numberList[5];

0 1 2 3 4

* Each element in the array is an **int**
* The size of the array cannot be changed
* The variable **numberList** is a reference to the entire array

**Accessing Array Elements**

* To access the individual elements within an array use "array index" notation
  + arrayNameVariable [index]

numberList[0]

Array access is

through an index

**for** (size\_t i = 0; i < 5; i++) {

numberList[i] = 0;

}

* Arrays are zero based
  + First element in array - numberList[0]
  + Last element in array - numberList[4]
* For loops generally used with arrays since we know how many times the loop will occur
  + Example: assign the numbers 0 to 4 to numberList
  + Note: The type **size\_t** should be used for an array index (represents an unsigned integral type)

// Assign the numbers 0 to 4 to numberList array

**for** (size\_t i = 0; i < 5; i++) {

numberList[i] = i;

**printf** ("numberList[%u] = %d\n", i, numberList[i]);

}

**Displays**

numberList[0] = 0

numberList[1] = 1

numberList[2] = 2

numberList[3] = 3

numberList[4] = 4

* To manually assign values to our numberList array we could write code like this:

// Manually initialize array

numberList[0] = 3;

numberList[1] = 34;

numberList[2] = 44;

numberList[3] = 81;

numberList[4] = 7;

* Note: if you try to access an element outside the range of an array in C it is not a runtime error but you are causing problems since you are writing over memory!