Arrays
Session Outline

1. Introduction to Arrays
2. Examples
Arrays

- It is often useful in programs to store many of the same data type together in memory.

- In programming this is known as arrays.

- Arrays can be of any data type.
Why would you need to use arrays?

- Storing a bunch of ints representing counts over time
- Storing a bunch of floats representing grades for students in the class
- Storing a bunch of strings representing the names of people in the class
Simple Array Declaration

```c
int age[5];
```

- In one line this creates 5 different age variables which you can access independently:
  ```
  age[0], age[1], age[2], age[3], age[4]
  ```

- Note the bounds
  - there is an element at index 0
  - there is no element at index 5.
Indecies

- The value inside the square brackets is known as an index

- Expressions can be evaluated within the square brackets to access a particular element

- If you try to access an element out of bounds, you will get a runtime error (segfault)
Array Bounds

- The best practice is to use a constant to specify the array bounds

```c
const int ARRAY_LENGTH = 10;
float marks[ARRAY_LENGTH];
```

  - Makes your code more readable
  - the value is accessible for loops.

- Bounds can be set by a constant, but not a variable.
Array Initialization with Loops

- A typical procedure to initialize an array is to use a for loop

```c
const int ARRAY_LENGTH = 10;
int myArray[ARRAY_LENGTH];
for (int i = 0; i < ARRAY_LENGTH; i++) {
    myArray[i] = 0;
}
```
Array Initialization (quick)

- Shorthand syntax for initializing arrays:

  ```
  int myArray[5] = {3, 5, 6, 2, 1};
  OR
  int myArray[] = {4, 7, 6, 2, 3, 5, 6, 7, 1};
  ```
Arrays in Functions

● Because Arrays are so large, in C++ Arrays are always passed by reference to functions.

● This is fast, but dangerous.

● How would you copy an array if you wanted to be safe?
Remember

- Arrays are just a bunch of the same data types stored together in memory with a convenient way to access individual elements.

- If you want to do anything useful with the array data, you need to access the individual elements. ie. ArrayA is equal to ArrayB if and only if

  (ArrayA[0] == ArrayB[0] &&
Example: Testing Array Equality

• Write a function which takes two arrays, the length of the arrays, and returns whether or not the arrays are equal.

• Assume that the arrays are properly initialized and of the correct length.
Example: Initialize odds and evens

- Write a function which takes an integer array and a length, and initializes all of the odd indexed elements to 1, and all of the even indexed elements to 0.
Example: Array of Fibonacci Numbers

• Create an array of length 10 which is initialized to the first 10 fibonacci numbers.

• The fibonacci sequence in one in which:
  – Element 1 = 1
  – Element 2 = 1
  – Element 3 an onwards = Element 1 + Element 2
  – ie. 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144

• Can you adapt it to the first 50 numbers?