Arrays (1)

Basic array usage
Goals

By the end of this lesson you will be able to:

- Understand the basics of arrays
- Use one-dimensional arrays in a variety of ways
- Use arrays with functions
What are arrays?

- Arrays are simply a collection of multiple variables of the same datatype under one identifier.
- Arrays are very useful when you need many storage spaces but declaring many individual variables would be cumbersome.
Basic array usage

- Arrays are declared as follows:
  ```
  datatype identifier[number];
  ```
- `number` indicates how many of that specific datatype you need to store
- `number` must be hardcoded before you compile – you cannot use a variable, but may use a constant expression like `5 * 6`
- Access array elements using
  ```
  identifier[0] through identifier[number - 1];
  ```
Some vocabulary

- The brackets [] are sometimes called the **array index operator** or **subscript operator**
- The value inside [] is called the **index**
- These first arrays we will examine are often called **one-dimensional arrays**
Looping through an array

• Given the declaration:
  
  ```
  datatype identifier[number];
  ```

• Access all elements of the array using a loop:
  
  ```
  for (int i = 0; i < number; i++)
      // some code with identifier[i] here
  ```
Unknown amounts of data

• If you do not know how much data you will have but have an upper bound, make the array that size and keep track of how many elements you are using in the array as well

• Example:
  – Up to 100 lines in a file to read & process
  – Make your array size 100
  – Every time you read a line into a different array element, increment a counter
  – Later when processing the entire array increment from 0 to the counter
What restrictions are there?

- You cannot return an array from a function
- You cannot set two arrays equal to each other
  - You can set two array elements equal to each other
- You cannot access an element outside the bounds of the array (doing so may cause a **segmentation fault**)

Arrays can be initialized when declared as such:

```plaintext
datatype identifier[number] = { value1, value2, ... };
```

You do not have to initialize all elements and, with numerical types, the remaining values are 0.

For string arrays the remaining values will be the empty string.
Arrays and functions

- As noted before, arrays cannot be returned from functions, but they can be passed to functions.

- The syntax for doing this is as such:

  ```c
  void fcn(int []); // prototype
  void fcn(int a[]) // implementation
  {
    ...
  }
  ```

- Importantly, all arrays are passed by reference automatically.

- If you are not changing the array in the function, pass it by constant reference instead.
How are arrays stored?

- Arrays are stored contiguously in memory and the actual “value” referred to by the array identifier is the memory address of the location of the first element in the array.

- Example:
  ```
  int a[2] = { 5, 8 };  // allocates spots for 2 integers starting at memory address 0x1000
  - 5 is stored at memory address 0x1000
  - 8 is stored at memory address 0x1004 (because integers are 4 bytes)
  - a contains the value 0x1000
  - a[0] contains the value 5
  - a[1] contains the value 8
  ```
Exploration

- Write a program that will generate 1,000,000 random numbers and place them in an array, then count how many of those values are even and how many are odd
- Write a program that will read all lines of a file (up to 100 lines) and then output those lines from bottom to top
- Write a program that will read all numbers in a file (up to 100 numbers separated by spaces) and then calculate the average and standard deviation of those values