Programming Example 10: Sorting and Searching

Introduction

These three programs illustrate

1. A Linear Search.
2. A Binary Search.
3. A Bubble Sort.

As you will see from the code, a linear search is easy to write. All you need is one loop where you look at each element of the array in turn to see if it contains the value that you are looking for. However, a linear search is not very efficient. If an array has n elements in it, you will have to access the array, on average, $n/2$ times to locate the value that you are looking for. So, for example, if an array has 10,000 elements in it, you would have to access the array, on average, 5,000 times.

The source code for the linear search is here.

The executable for the linear search is here.

It takes a bit more code to do a binary search, and the logic is a little more complex. However, a binary search is very efficient when compared to a linear search. This is because you eliminate one half of the search range each time that you access the array. On average, you will have to access an array of n elements $\log_2(n)$ - 1 times to locate the value that you are looking for. For an array with 10,000 elements, you would have to access the array, on average, about 12 times. That's pretty fast compared with our linear search! Of course, there is a downside. Before you can use a binary search on an array, it must be sorted first.

The source code for the binary search is here.

The executable for the binary search is here.

The easiest sorting algorithm for an array is the bubble sort. The bubble sort is quite inefficient, but the logic is straightforward and it is easy to write. When you look at the code for the bubble sort, note the nested for loops and the swapping of values when they are out of order.

The source code for the bubble sort is here.

The executable for the bubble sort is here.