C++ Week 2 Algorithms

Introduction

These exercises provide a demonstration of the efficiencies of various sorting and searching algorithms.

The swap function defined below takes the address of two integer variables and swaps their values. Note that the formal parameters are pointers. This ensures that any changes made to their value inside the Sort function are passed back to the arguments in the calling code.
Create a new Visual Studio project and add the Swap function above main.

```
void Swap(int * a, int * b){
        int temp = *b;
        *b = *a;
        *a = temp;
}
```

Exercise 1

Implement the SelectionSort algorithm (see presentation) as a stand alone function located below Swap.
Within main create an array (named data) of integers with the following values 4,5,6,1,3,9,4,8,2,7 and pass the array to the SelectionSort function to be sorted. Then within main display the array within the console.

Exercise 2

Let's create a bigger data set to test our algorithms. Create a dynamic array of integers (named bigData) that contains 1000 random numbers between 0 and 100 (inclusive). Test the SelectionSort on this new dataset.

Exercise 3

Display the number of seconds it takes to sort the array. If the sort completes within less than 1 second increase the array size by a factor of 10 until it takes longer than a second. On my 3.7GHZ laptop it takes 14 seconds to sort a 100,000 element array.

Exercise 4

Comment out the previous invocation of SelectionSort and implement the bubble sort algorithms as a standalone function and test it out on the same bigData array. Once more display the processing time.

C++ Week 2 Algorithms

Exercise 5

Comment out the previous invocation of BubbleSort and implement the Enhanced Bubble Sort and time it.

Exercise 6

Implement the Sequential Search function and test it out within main.

Exercise 7

Implement the Binary Search function and test it out within main.