Search and Sort in One dimensional array

- **Search:**
  - return the subscript of the array element that match the value that is being searched for
  - return -1 if the value is not there
  - We discuss two methods: linear search vs. binary search (requires the array elements to be sorted)

- **Linear search**

```
int linearSearch(int arr[], int size, int value)
{
        int index = 0;            // Used as a subscript to search the array
        int position = -1;        // To record the position of search value
        bool found = false;       // Flag to indicate if value was found

        while (index < size && !found)
        {
                if (arr[index] == value)        // If the value is found
                {
                        found = true;           // Set the flag
                        position = index;       // Record the value's subscript
                }
                index++;                // Go to the next element
        }
        return position;          // Return the position, or -1
}
```

- **Binary search**

```
int binarySearch(int array[], int size, int value)
{
        int first = 0,          // First array element
        last = size - 1,      // Last array element
        middle,              // Mid point of search
        position = -1;        // Position of search value
        bool found = false;       // Flag

        while (!found && first <= last)
        {
                middle = (first + last) / 2;     // Calculate mid point
                if (array[middle] == value)      // If value is found at mid
                {
                        found = true;
                        position = middle;
                }
                else if (array[middle] > value)  // If value is in lower half
                        last = middle - 1;
                else
                        first = middle + 1;          // If value is in upper half
        }
        return position;
}
```

- **Bubble Sort**

```
void bubbleSort(int array[], int size)
{
  int maxElement;
  int index;

  for (maxElement = size - 1; maxElement > 0; maxElement--)
  {
    for (index = 0; index < maxElement; index++)
    {
      if (array[index] > array[index + 1])
      {
        swap(array[index], array[index + 1]);
      }
    }
  }
}

void swap(int &a, int &b)
{
  int temp = a;
  a = b;
  b = temp;
}
```

- **Selection Sort**

```
void SelectionSort(int array[], int size) {

    int minIndex, minValue;

    // repeat pair-wise comparison across the elements n-1 times
    for (int start = 0; start < (size – 1);  start++) {

         // find the index of the element with the smallest value
         // in the remaining elements
        minIndex = start;
        minValue = array[start];
        for (int index = start + 1; index < size; index++)
        {
            if (arr[index] < minValue)
            {
                  minValue = array[index];
                  minIndex = index;
            }
         }

         Swap(array[minIndex], array[start]);
    }
}
```

```
// This program reads in a number of values from a data file, and stores the values in an array
// It sorts the values in ascending order.
// A search is performed to see if a particular value is in the array or not.

#include <iostream> // Header file for input/output
#include <fstream>
#include <cassert>
using namespace std;

const int MAX_SIZE = 100; // Maximum number of books to be stored

// declare all the functions here
void SelectionSort(int data[], int number);
int LinearSearch(int data[], int numOfValues, int toFind);

int main()
{
   int  toFind, location, count;
   int data[MAX_SIZE];
   ifstream myIn;
   myIn.open("sort.dat");
   assert(myIn);

   // read in the data from data file
   count = 0;
   while (count < MAX_SIZE && myIn >> data[count])  {
      count ++;
   }

   // sort the values into ascending order
   SelectionSort(data, count);

   // display the original data
   cout << "The numbers are : " << endl;
   for (int i=0; i<count; i++)  {
      cout << i << " : " << data[i] << endl;
   }

   // search for a user supplied number using linear search
   cout << "Which value do you look for:" << endl;
   cin >> toFind;
   location = LinearSearch(data, count, toFind);

   if (location >=0)
      cout << "The value is at location: " << location << "." << endl << endl;
   else
      cout << "The value is not in the list." << endl << endl;

   myIn.close();
   return 0;
}
```