```
// Michael Steele
// Asks for the number or value for and array than has you fill the array. After, you will enter a value to search the array for.
#include <iostream>
using namespace std;

int binarySearch(int[], int, int);
void displayArray(int[], int);
void selectionSortArray(int[], int);

int main(){
    int numValue,enter,searchValue,result;
    cout << "Enter the number of values to have inside the list :   ";
    cin >> numValue;
    cout << endl;
    int values[numValue];

    for(int i = 0; i < numValue; i++)
    {
        cout << "Enter a value to hold in position " << i+1 << " :   ";
        cin >> enter;
        values[i]=enter;
        cout << endl;
    }
    cout << "   >   Array before sorting" << endl;
    displayArray(values,numValue);

    selectionSortArray(values,numValue);
    cout << "   >   Array after sorting" << endl;
    displayArray(values,numValue);
    cout << "Enter a value to search the array for :   ";
    cin >> searchValue;
    cout << endl;
    result = binarySearch(values,numValue,searchValue);
    if (result == -1)
                        cout << "The value " << searchValue << " is not in the list" << endl;
            else
                        cout << "The value " << searchValue << " is in position number "
                            << result + 1 << " of the list" << endl;


return 0;
}
```

```
int binarySearch(int array[], int numElems, int value)           // function heading
{
        int first = 0;                            // First element of list
        int last = numElems - 1;         // last element of the list
        int middle;                                       // variable containing the current
                                                                  // middle value of the list

        while (first <= last)
        {
                middle = first + (last - first) / 2;

                if (array[middle] == value)
                        return middle;            // if value is in the middle, we are done

                else if (array[middle]<value)
                        last = middle - 1;        // toss out the second remaining half of

                else
                        first = middle + 1;       // toss out the first remaining half of
                                                                  // the array and search the second
        }

        return -1;      // indicates that value is not in the array
}

void selectionSortArray(int array[], int elems)
{
        int seek;                // array position currently being put in order
        int minCount;  // location of smallest value found
        int minValue;  // holds the smallest value found

        for (seek = 0; seek < (elems - 1); seek++)        // outer loop performs the swap
                                                                                                   // and
then increments seek
        {
                minCount = seek;
                minValue = array[seek];

                for (int index = seek + 1; index < elems; index++)
                {
                        // inner loop searches through array
                        // starting at array[seek] searching
                        // for the smallest value. When the
                        // value is found, the subscript is
                        // stored in minCount. The value is
                        // stored in minValue.

                        if (array[index] > minValue)
                        {
                                minValue = array[index];
                                minCount = index;



                        }

                }

                // the following two statements exchange the value of the
                // element currently needing the smallest value found in the
                // pass(indicated by seek) with the smallest value found
                // (located in minValue)

                array[minCount] = array[seek];
                array[seek] = minValue;



        }
}

void displayArray(int array[], int elems)       // function heading
{
        // displays the array
        for (int count = 0; count < elems; count++)
                cout << array[count] << "       " << endl;
}
```

```
Enter the number of values to have inside the list :   5

Enter a value to hold in position 1 :   11

Enter a value to hold in position 2 :   22

Enter a value to hold in position 3 :   33
Enter a value to hold in position 4 :   44

Enter a value to hold in position 5 :   55

   >   Array before sorting
11
22
33
44
55
   >   Array after sorting
55
44
33
22
11
Enter a value to search the array for :   22

The value 22 is in position number 4 of the list

Process returned 0 (0x0)   execution time : 10.440 s
Press any key to continue.
```