Searching and Sorting

As we continue to deal with larger amounts of data in our programs by using files there are two main problems that come up.

Searching the data (in an array) to determine the location of a particular value.

And, sorting the data (in an array) to rearrange the elements in an ordered fashion.

We may want to search a list of scores to see if a student got a particular score. We may want to sort that list in decreasing order by score.

The notes for this section with focus on the process or the algorithm used to accomplish these tasks.

The code becomes relatively minor when the algorithm is understood.

Finding the Smallest Value in an Array

We've seen some sorting done in our cards program where we needed to order the cards given by the random number generator.

But, what about a larger list of numbers?

The steps (algorithm) to search an array for the index of the smallest value are:

```
1) Assume that the first element is the smallest so far and save its
subscript as "the subscript of the smallest element found so far"

2) For each array element after the first one:

    2.1) If the current element < the smallest so far

        2.1.1) Save the subscript of the current element as the
        "subscript of the smallest element found so far"
```

Example

```
int findMin(const int x[],int startIndex,int endIndex)
{
     int minIndex;   // index of the smallest element found
     int i;         // index of the current element

     // Validate subarray bounds
     if ((startIndex < 0) || (startIndex > endIndex))
     {
               cout << "Error in subarray bounds" << endl;
               return -1; // return error indicator
     }

     // Assume the first element of subarray is smallest
     and check the rest.
     // minIndex will contain subscript of smallest
     examined so far.

     minIndex = startIndex;
     for (i = startIndex + 1; i <= endIndex; i++)
               if (x[i] < x[minIndex])
               minIndex = i;

     // All elements are examined and minIndex is
     // the index of the smallest element.

return minIndex;
}
```

Array Search

We can search and array for a particular element by comparing each array element, starting with the first (subscript 0), to the target, with the value we are seeking.

If the match occurs, we have found the target and return the subscript of its location.

If we test all elements without finding a match, return -1 to indicate it was not found.

We choose -1 because no array element has a negative subscript.

Algorithm

```
1) - For each array element

        1.1) - If the current element contains the target

        1.2) - Return the subscript of the current element

2) - Return -1
```

Example

```
int linear_search(int target)
{
     const int items[] = {44, 58, 23, 90, 12, 10, 13, 15};
     int size = 8;
     for (int i = 0; i < size; i++)
             if (items[i] == target)
                     return i; // found, return subscript
     // All elements were tested without success.
     return -1;
}
```

Sorting in Ascending Order

Often times when working with large amounts of data, programs will run more efficiently if the data is first sorted.

We are covering one simple sort algorithm for now but of course there are many others.

Selection Sort

The **selection sort** is a fairly intuitive sorting algorithm.

To perform a selection sort of n elements (subscripts 0 to n-1), we locate the smallest element in the array and then switch the smallest element with the element at subscript 0.

This will place the smallest element found so far in location 0.

We then locate the smallest element remaining in the subarray with subscripts 1 through n-1 and switch it with the element at subscript 1.

This places the 2nd smallest element at subscript 1.

We continue this until all items are placed.

Algorithm

1) - Starting with the first item in the array and ending with the next-to-last item

> 1.1) - Set i equal to the subscript of the first item in the subarray to be processed in the next steps.
>
> 1.2) - Find the subsript of the smallest item in the subarray with the subscripts ranging from i through n-1.
>
> 1.3) - Exchange the smallest item found in step 1.2 with item i.

Example

```
void selection_sort(int items[], int n)
{
int min_subscript;

      for (int i = 0; i < n-1; i++)
      {
            min_subscript = find_min(items, i, n-1);

            //exchange items
            int temp;
            temp = items[min_subscript];
            items[min_subscript] = items[i];
            items[i] = temp;
      }
}
```