SEARCHING AND SORTING IN ARRAYS

Linear Search

```
int LinearSearch (const int a[], int aSize, int toFind)
{
   // Look through all items, starting at the front.
   for (int i = 0; i < aSize; i++)
                  if (a[i] == toFind)
                              return i;

   // You've gone through the whole list without success.
   return -1;
}
```

Search with Array of structs

```
int FindMovieInDatabase(const MovieType movies[],
                                int numOfMovies, string movieTitle)
{
    int index=-1;
    for (int i=0; i<numOfMovies; i++)
    {
        if (movieTitle == movies[i].title)
        {
            index=i;
            break;
        }
    }
    return index;
}
```

Sorting -- bubble sort

(11) (34) | 26 90 37 58 10 47 36

34 (11) (26) | 90 37 58 10 47 36

34 26 (11) (90) | 37 58 10 47 36

34 26 90 (11) (37) | 58 10 47 36

34 26 90 37 (11) (58) | 10 47 36

34 26 90 37 58 (11) (10) | 47 36

34 26 90 37 58 11 (10) (47) | 36

34 26 90 37 58 11 47 (10) (36)

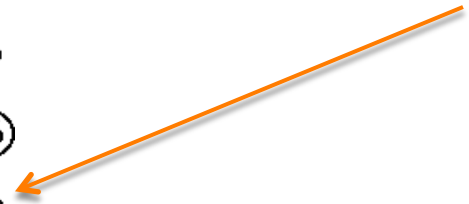
34 26 90 37 58 11 47 36 | 10

34 90 37 58 26 47 36 | 11 10

34 90 37 58 26 47 36 | 11 10

90 37 58 34 47 36 | 26 11 10

90 58 37 47 36 | 34 26 11 10



At the end of the first iteration,
The smallest value is placed at the last position

Sorting to descending order

Sorting -- bubble sort

(11) (34) | 26 90 37 58 10 47 36

34 (11) (26) | 90 37 58 10 47 36

34 26 (11) (90) | 37 58 10 47 36

34 26 90 (11) (37) | 58 10 47 36

34 26 90 37 (11) (58) | 10 47 36

34 26 90 37 58 (11) (10) | 47 36

34 26 90 37 58 11 (10) (47) | 36

34 26 90 37 58 11 47 (10) (36)

34 26 90 37 58 11 47 36 | 10

34 90 37 58 26 47 36 | 11 10

34 90 37 58 26 47 36 | 11 10

90 37 58 34 47 36 | 26 11 10

90 58 37 47 36 | 34 26 11 10

At the end of the second iteration,
The two smallest values are
placed in their final position

Sorting to descending order

Bubble Sort – descending order

```
void BubbleSort (int list[], int listSize)
{
     bool sorted= false;       //is the list sorted?
     int last = listSize - 1;    //start last at the last array element
     int i;                    //used as a loop index

    while ( !sorted )
    {
          sorted = true; //assume the list is in order
          for (i = 0; i < last; i++)
          {
                if (list[i] < list[i+1])
                {          Swap (list[i], list[i+1]);        //swap two elements
                           sorted = false; //the list wasn't already sorted
                }
          }
          last--;
    }
}
```

Bubble Sort

```
void Swap (int  &value1,  int  &value2)
{
        int    tmp;

        tmp = value1;
        value1 = value2;
        value2 = tmp;

        return;
}
```

```
void SortMovies(MovieType movies[], int numOfMovies)
{
   bool sorted=false;    // indicates whether additional comparison passes are needed
   int  last=numOfMovies-1; // the index of the last item in the remaining part of the array
   MovieType tmp;
   while (!sorted)
   {
      sorted=true; // assuming the remaining array is sorted.
      for (int i=0; i<last; i++)
      {
         if (movies[i].title > movies[i+1].title)
         {
            tmp = movies[i];    // swap the two records
            movies[i] = movies[i+1];
            movies[i+1] = tmp;

            // the remaining array is not sorted, need at least another pass of comparison
            sorted = false;
         }
      }
      last--;
   }
   return;
}
```

Sorting with Array of structs