Insertion Sort

The **insertion sort**, unlike the other sorts, **passes through the array only once.** The insertion sort is commonly compared to organizing a handful of playing cards. You pick up the random cards one at a time. As you pick up each card, you insert it into its correct position in your hand of organized cards.

The insertion sort splits an array into two sub-arrays. The first sub-array (like the cards in your hand) is always sorted and increases in size as the sort continues. The second sub-array (like the cards to be picked up) is unsorted, contains all the elements yet to be inserted into the first sub-array, and decreases in size as the sort continues.

Let's look at our same example using the insertion sort for descending order.

Array at beginning:	84	69	76	86	94	91
☐ = 1st sub-array	84	69	76	86	94	91
☐ = 2nd sub-array	84	69	76	86	94	91
	84	76	69	86	94	91
	86	84	76	69	94	91
	94	86	84	76	69	91
2nd sub-array empty	94	91	86	84	76	69

The insertion sort maintains the two sub-arrays within the same array. At the beginning of the sort, the first element of the first sub-array is considered the "sorted array". With each pass through the loop, the next element in the unsorted second sub-array is placed into its proper position in the first sorted sub-array.

The insertion sort can be very fast and efficient when used with smaller arrays. Unfortunately, it loses this efficiency when dealing with large amounts of data

// Insertion Sort Method for Descending Order

```
public static void InsertionSort( int [ ] num)
{
   int j;                    // the number of items sorted so far
   int key;                  // the item to be inserted
   int i;

   for (j = 1; j < num.length; j++)     // Start with 1 (not 0)
  {
```

```
               key = num[ j ];
               for(i = j - 1; (i >= 0) && (num[ i ] < key); i--)   // Smaller values are moving up
              {
                    num[ i+1 ] = num[ i ];
              }
             num[ i+1 ] = key;    // Put the key in its proper location
         }
     }
```