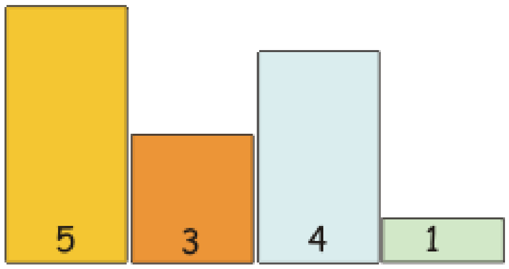
# Designing a Bubble Sort

## Swapping Array Elements

A bubble sort requires that we compare two elements of an array, and then put those two elements in the correct order with respect to one another. So, one of the key elements that we will need to write a bubble sort is a **swap( )** method. You have seen the code to do this before. We will need to pass the elements to be swapped by reference. Let's assume for this exercise, that we are sorting an array of integers. The swap method would look like

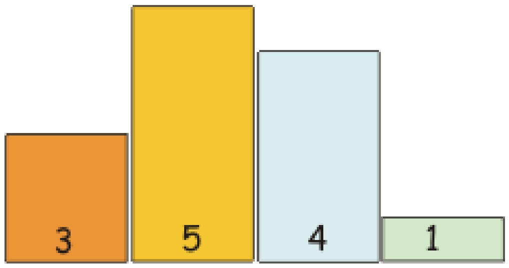
```
void swap(ref int a, ref int b)
{
     int temp = a;
     a = b;
     b = temp;
}
```

## Sorting



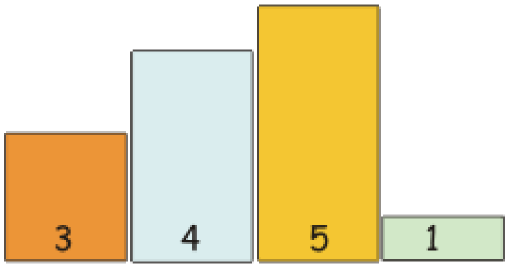
Consider the blocks shown in the figure to the left. Our objective in this example is to put the blocks in order from the smallest to the largest. Let's begin by comparing the first two blocks. Clearly the first block is larger than the second one, so we need to swap the position of these two blocks. The code to do this might look something like

```
if ( block [1] > block [2])
     swap (ref block [1], ref block [2]);
```



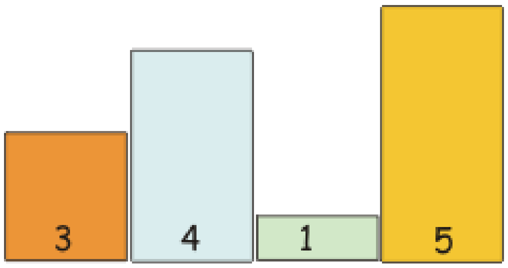
This looks right. Now lets move on and compare the second block with the third. The second is bigger than the first, so we also swap these two. This code will look like

```
if ( block [2] > block [3])
     swap (ref block [2], ref block [3]);
```



Note that after this swap the large black has moved one more place to the right. This is what we want. Now lets move on and compare the third block with the last one. The third is bigger than the fourth, so we also swap these two. This code will look like

```
if ( block [3] > block [4])
     swap (ref block [3], ref block [4]);
```

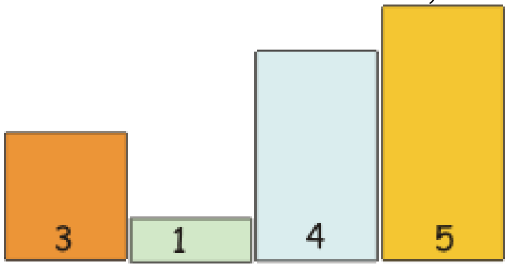


At this point, we have been through the entire array one time. After this last swap, the largest block has moved all of the way to the right side. It has moved one position for each comparison that we made. This "bubbling" of the largest element to where it belongs is a key property of a bubble sort. We can express all of the above steps in a loop of the form

```
for (int i = 0; i < 3; i++)
{
     if ( block [i] > block [i+1])
          swap (ref block [i], ref block [i + 1]);
}
```

Notice the index in the loop. We stop when *i* is less than 3, not 4, as your might expect. This is because once we compare the next to last element of the array (in this case element 3) with the last one, we are done.

We have now moved through the array one time. However, as you can see by looking at the picture, the array is not sorted. What this means is that we have to go through the array a second time. The first comparison will not result in a swap, since the first element is smaller than the second. However, when we compare the second with the third, the second is larger than the third and we will swap them. Finally, the third is smaller than the last, and no swap occurs. The result is



The array is still not sorted, but it is close. The last two blocks are just where they belong. So, we will have to go through the array yet again. To be sure that every element gets swapped, a bubble sort therefore therefore contains two loops that look like

```
for (int j = 0; j < 3; j++)
{
     for (int i = 0; i < 3; i++)
     {
          if ( block [i] > block [i+1])
               swap (ref block [i], ref block [i + 1]);
     }
}
```