The Bubble Sort

The bubble sort is one of the best known sorting routines. Unfortunately, it is also one of the slowest sorting algorithms. However, the bubble sort is easy to understand and code, and works reasonably well if the number of items to be sorted is less than a thousand or so.

The bubble sort works by making a series of “passes” through the array to be sorted. In making a pass, adjacent elements in the array are compared, and interchanged if they are not in the correct order. After the first pass, the largest element in the array has “bubbled” to the top, or last, location of the array and need no longer be considered. This means that the second pass can be done on the first ArraySize – 1 members of the array, the third pass on the first ArraySize – 2 members and so forth. The passes continue until ArraySize passes have been made, or a pass is completed with no interchange. In either case, the array will be sorted.

There are a number of variations of the bubble sort. A fairly good bubble sort algorithm is described below in psuedocode. The psuedocode assumes that:

1) ArraySize is the number of elements in array to sort.
2) The last subscript in the array is assumed to be ArraySize -1.
3) List is an array [0..ArraySize-1] of some data type.
4) The elements List[0]...List[ArraySize-1] are sorted.
5) Swap(A,B) interchanges the contents of A and B.

Algorithm BubbleSort ( integer array List[], integer ArraySize )

SwapsMade = True
NumberUnsorted = ArraySize - 1

while SwapsMade AND NumberUnsorted >= 1 do

    dec NumberUnsorted
    SwapsMade = False //If this not changed, List is sorted

    for k = 0 to NumberUnsorted
        if List[k] > List[k+1] //Are Adjacent elements out of order?
            swap(List[k], List[k+1]) //If so, swap them
            SwapsMade = True //List not known to be sorted
        end if
    end for
    //k incremented by for loop
end while