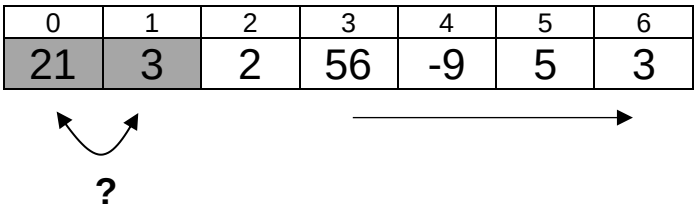
BUBBLE SORT

Bubble sort is a comparative sort, which repeatedly steps through the list to be sorted.

During each pass, adjacent items are compared and swapped if they are out of order. This begins by comparing the first item with the second and swapping if first is greater than second. Second and third place is then compared and so on until the end of the pass.



After each traversal, the largest number will have found its way to its correctly sorted position (if sorting in ascending order).

Key features

- Adjacent values are compared, swapping each time a value is out of order
- At the end of each pass, a boolean variable called *swaps* is checked.
- If a swap has taken place during a pass, will have been set to true.
- If no swap takes place during a pass, swaps will remain false and the loop will terminate.

If no swaps take place during a pass, this indicates that the list must have been sorted by the end of the previous pass.

In order to make sure the list is sorted, the algorithm must always complete this final pass with no swaps taking place before the loop can terminate.

Bubble Sort – The Algorithm

```
PROCEDURE bubble_sort(list)
     DECLARE n INITIALLY length(list)
     DECLARE swapped INITIALLY TRUE
     WHILE swapped AND n >= 0
          SET swapped TO False
          FOR i = 0 to n-2 DO
               IF list[i] > list[i+1] THEN
                    SET temp TO list[i]
                    SET list[i] TO list[i+1]
                    SET list[i+1] TO temp
                    SET swapped TO TRUE
               END IF
          END FOR
          SET n TO n - 1
     END WHILE
END PROCEDURE
```

Bubble Sort Algorithm

Bubble Sort – Pseudocode and Description

Design	Commentary
`start conditional loop`	
`set swapped to false`	Set a flag variable to note if a swap has been made
`fixed loop i = 0 to length of list[] - 2`	Loop from the first element to the penultimate element of the array 'list' Note: -2 is equivalent to the second-last element of the array, as the array indexes from 0
`if list[i] > list[i+1] then`	Compare two adjacent values
`swap the two values`	Swap the two values if they are not in the correct order
`set swapped to true`	Set the flag variable to note that a swap has taken place
`end if`	
`end fixed loop`	
`end conditional loop when swapped is equal to false`	The sort ends when a pass results in no swaps

Bubble Sort – Improvement

Consider an array that stores the following values:

0	1	2	3	4	5	6	7	8
45	23	99	7	3	64	37	63	34

After one pass through the array, the largest value will always 'bubble' up to the end of the array.

23	45	7	3	64	37	63	34	99

After a second pass, the second-largest number is also sorted.

23	7	3	45	37	63	34	64	99

When bubble sorting a list of values, the number of iterations carried out by each nested loop can be reduced by one each pass. This improves the efficiency of the bubble sort algorithm.

Design	Commentary
`n equals the length of an array called list`	The length of the array is stored in a variable
`start conditional loop`	
`  set swapped to false`	
`  fixed loop i = 0 to n - 2`	Loop from the first element to the penultimate array element
`    if list[i] > list[i+1] then`	
`      swap the two values`	
`      set swapped to true`	
`    end if`	
`  end fixed loop`	
`  n = n - 1`	Each fixed loop reduces the iterations by 1, as one more element is sorted correctly at the end of the array
`end conditional loop when swapped is equal to false`	