**Pseudocode and Flowchart for Sorting Algorithm - Write pseudocode and create a flowchart for a bubble sort algorithm. Provide a brief explanation of how the algorithm works and a simple array of integers to demonstrate a dry run of your algorithm.**

***Bubble Sort Algorithm***

*Bubble sort is a simple sorting algorithm that repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted.*

***Pseudocode:***

***FUNCTION*** *bubble\_sort(data)*

***FOR*** *i = 0 TO len(data) - 2*

*swapped =* ***FALSE***

***FOR*** *j = 0 TO len(data) - i - 2*

***IF*** *data[j] > data[j + 1]*

***SWAP****(data[j], data[j + 1])*

*swapped =* ***TRUE***

***IF NOT*** *swapped*

***BREAK***

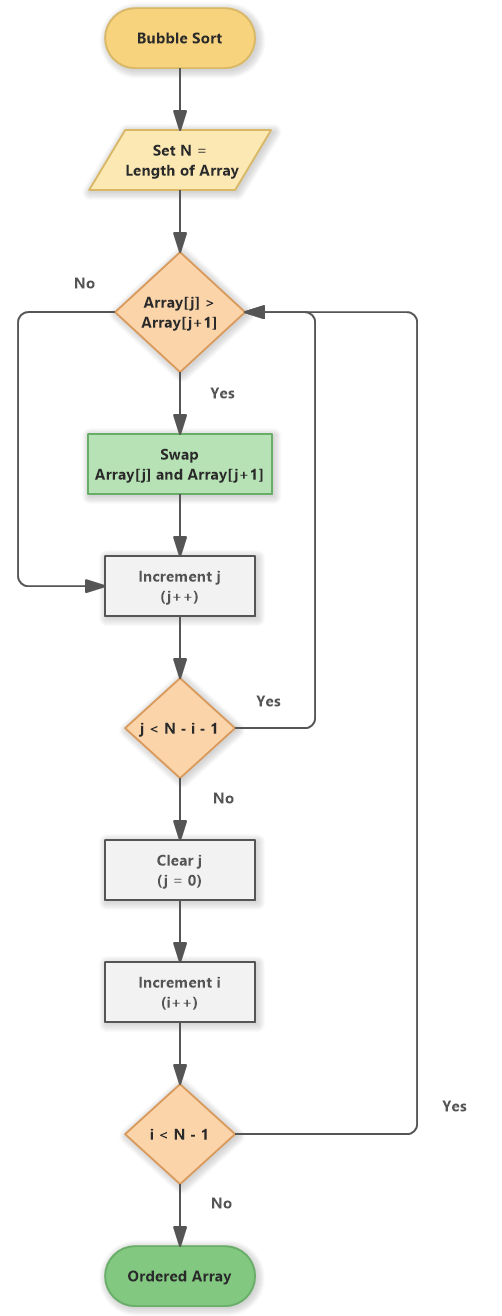
***END FOR***

***END FUNCTION***

***Explanation:***

1. *The bubble\_sort function takes a list of data as input.*
2. *An outer loop (i) iterates through the list from the beginning to the second-last element.*
3. *An inner loop (j) iterates through the list from the beginning to the length of the list minus the current outer loop iteration (i) - 2. This is because the largest element will be "bubbled" to the end in the first iteration, the second largest in the second iteration, and so on.*
4. *Inside the inner loop, the elements at adjacent positions (j and j + 1) are compared.*
5. *If the elements are in the wrong order (i.e., the element at j is greater than the element at j + 1), the SWAP function exchanges their positions.*
6. *A flag swapped is used to track if any swaps occurred in the inner loop.*
7. *If no swaps occurred in the inner loop (swapped remains False), it means the list is already sorted, and the outer loop can be terminated using the BREAK statement.*

***FlowChart:***



***Consider the following list of integers:***

*data = [64, 34, 25, 12, 22, 11, 90]*

*Pass 1*

*i = 0*

*Compared elements: (64, 34) - Swap*

*Compared elements: (34, 25) - Swap*

*Compared elements: (25, 12) - Swap*

*Compared elements: (12, 22) - Swap*

*Compared elements: (22, 11) - Swap*

*swapped is True, so continue looping.*

*After pass 1:*

*data = [11, 12, 22, 25, 34, 64, 90]*

*Pass 2*

*i = 1*

*Compared elements: (12, 22) - Swap*

*Compared elements: (22, 25) - Swap*

*Compared elements: (25, 34) - Swap*

*Compared elements: (34, 64) - Swap*

*swapped is True, so continue looping.*

*After pass 2:*

*data = [11, 12, 22, 25, 34, 64, 90]*

*Pass 3*

*i = 2*

*Compared elements: (12, 22) - Swap*

*Compared elements: (22, 25) - Swap*

*Compared elements: (25, 34) - Swap*

*swapped is True, so continue looping.*

*Pass 4*

*i = 3*

*Compared elements: (22, 25) - Swap*

*Compared elements: (25, 34) - Swap*

*swapped is True, so continue looping.*

*Pass 5*

*i = 4*

*Compared elements: (25, 34) - Swap*

*swapped is False, so the loop breaks.*

*Final Sorted List:*

*data = [11, 12, 22, 25, 34, 64, 90]*

*after each pass, the largest element bubbles up to the end of the list. The algorithm continues iterating until no swaps are needed, indicating that the list is sorted.*