SORTING

- The term sorting means **arranging the elements of the array so that they are placed in some relevant order which may either be ascending order or descending order.**
- For example, if we have an array that is declared and initialized as,

  int A[] = {21, 34, 11, 9, 1, 0, 22};
- Then the sorted array (ascending order) can be given as,

  A[] = {0, 1, 9, 11, 21, 22, 34}

SORTING

- There are two types of sorting:

- ***Internal sorting*** which deals with sorting the data stored in computer’s memory

- ***External sorting*** which deals with sorting the data stored in files. External sorting is applied when there is voluminous data that cannot be stored in computer’s memory.

Bubble Sort

- Bubble sort is a very simple method that sorts the array elements by repeatedly moving the largest element to the highest index position of the array (in case of arranging elements in ascending order).

- In bubble sorting, **consecutive adjacent pairs of elements in the array are compared with each other. If the element at lower index is greater than the element at the higher index, the two elements are interchanged so that the smaller element is placed before the bigger one. This process is continued till the list of unsorted elements exhaust.**

Bubble Sort Example

A[] = {30, 52, 29, 87, 63, 27, 19, 54}

Pass 1:

30, 52, 29, 87, 63, 27, 19, 54 - Compare 30 and 52. Since30<52, No Swap

30, **52, 29**, 87, 63, 27, 19, 54 - Compare 52 and 29. Since 52>29, Swap

30, **29, 52**, 87, 63, 27, 19, 54

30, 29, **52, 87**, 63, 27, 19, 54 - Compare 52 and 87. Since 52<87, No Swap

30, 29, 52, **87, 63**, 27, 19, 54 - Compare 87 and 63. Since 87>63, Swap

30, 29, 52, **63, 87**, 27, 19, 54

30, 29, 52, 63, **87, 27**, 19, 54 - Compare 87 and 27. Since 87>27, Swap

30, 29, 52, 63, **27, 87**, 19, 54

30, 29, 52, 63, 27, **87, 19**, 54 - Compare 87 and 19. Since 87>19, Swap

30, 29, 52, 63, 27, **19, 87**, 54

30, 29, 52, 63, 27, 19, **87, 54** - Compare 87 and 54. Since 87>54, Swap

30, 29, 52, 63, 27, 19, **54, 87**

30, 29, 52, 63, 27, 19, 54**, 87**

- After the end of the first pass, the largest element is placed at the highest index of the array.
- All the other elements are still unsorted.

Bubble Sort Example

Pass 2:

30, 29, 52, 63, 27, 19, 54, **87** - Compare 30 and 29. Since 30>29, Swap

29, 30, 52, 63, 27, 19, 54, **87**

29, **30, 52**, 63, 27, 19, 54 , **87** - Compare 30 and 52. Since 30<52, No Swap

29, 30, **52, 63**, 27, 19, 54 , **87** - Compare 52 and 63. Since 52<63, No Swap

29, 30, 52, **63, 27**, 19, 54 , **87** - Compare 63 and 27. Since 63>27, Swap

29, 30, 52, **27, 63**, 19, 54 , **87**

29, 30, 52, 27, **63, 19**, 54 , **87** - Compare 63 and 19. Since 63>19, Swap

29, 30, 52, 27, **19, 63**, 54 , **87**

29, 30, 52, 27, 19, **63, 54** , **87** - Compare 63 and 54. Since 63>54, Swap

29, 30, 52, 27, 19, **54, 63** , **87**

29, 30, 52, 27, 19, 54, **63** , **87**

- After the end of the second pass, the second largest element is placed at the second highest index of the array.
- All the other elements are still unsorted.

Bubble Sort Example

After end of Pass N-1, all the elements of the list will be in sorted order

BUBBLE_SORT(A, N)

```
Step 1: Repeat steps 2 For I = 0 to N-1
Step 2:     Repeat For J = 0 to N-I
Step 3:    If A[J] > A[J + 1], then
           SWAP A[J] and A[J+1]
           [End of Inner Loop]
      [End of Outer Loop]
Step 4: EXIT
```

Selection Sort

- Consider an array ARR with N elements. The selection sort takes N-1 passes to sort the entire array and works as follows.
- **First find the smallest value in the array and place it in the first position. Then find the second smallest value in the array and place it in the second position. Repeat this procedure until the entire array is sorted.** Therefore,
- In Pass 1, find the position POS of the smallest value in the array and then swap ARR[POS] and ARR[0]. Thus, ARR[0] is sorted.
- In pass 2, find the position POS of the smallest value in sub-array of N-1 elements. Swap ARR[POS] with ARR[1]. Now, A[0] and A[1] is sorted
- In pass 3, find the position POS of the smallest value in sub-array of N-2 elements. Swap ARR[POS] with ARR[2]. Now, ARR[0], ARR[1] and ARR[2] is sorted

In pass N-1, find the position POS of the smaller of the elements

ARR[N-2] and ARR[N-1}. Swap ARR[POS] and ARR[N-2] so that

ARR[0], ARR[1], … , ARR[N-1] is sorted.

Selection Sort

Example: Sort the array given below using selection sort

39	9	81	45	90	27	72	18

PASS	LOC	ARR[0]	ARR[1]	ARR[2]	ARR[3]	ARR[4]	ARR[5]	ARR[6]	ARR[7]
1	1	9	39	81	45	90	27	72	18
2	7	9	18	81	45	90	27	72	39
3	5	9	18	27	45	90	81	72	39
4	7	9	18	27	39	90	81	72	45
5	7	9	18	27	39	45	81	72	90
6	6	9	18	27	39	45	72	81	90

Selection Sort

SMALLEST (ARR, K, N)

```
Step 1: [Initialize] SET SMALL = ARR[K]
Step 2: [Initialize] SET POS = K
Step 3: Repeat for J = K+1 to N
            IF SMALL > ARR[J], then
                   SET SMALL = ARR[J]
                   SET POS = J
            [END OF IF]
       [END OF LOOP]
Step 4: Return POS
```

SELECTION_SORT (ARR, N)

```
Step 1: Repeat Steps 2 and 3 for K =0 to N-1
Step 2:    POS = SMALLEST(ARR, K, N)
Step 3:    SWAP ARR[K] with ARR[POS]
       [END OF LOOP]
Step 4: Exit
```