**Sorting**

**Introduction to Sorting:**

Sorting is nothing but arranging the data in ascending or descending order. The term **sorting** came into picture, as humans realised the importance of searching quickly.

There are so many things in our real life that we need to search for, like a particular record in database, roll numbers in merit list, a particular telephone number in telephone directory, a particular page in a book etc. All this would have been a mess if the data was kept unordered and unsorted, but fortunately the concept of **sorting** came into existence, making it easier for everyone to arrange data in an order, hence making it easier to search.

**Sorting** arranges data in a sequence which makes searching easier.

**Sorting can be performed using several techniques or methods, as follows:**  
  
1. Bubble Sort  
2. Insertion Sort  
3. Selection Sort  
4. Quick Sort

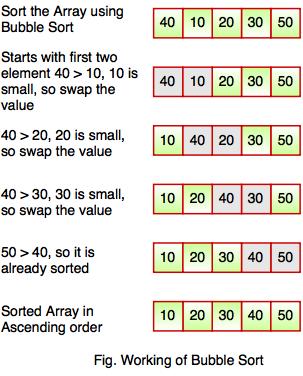
5.Merge Sort  
6. Heap Sort

**Methodology:** At first We open our Code-Blocks Editor ,then File, New , Empty file, save As . Then file Name : Insertion ,save . finally coding start .

At first We Declaration header file ,then define p of print,and s of scanf , then Declaration main method .

**1. Bubble Sort:**Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order.

* Bubble sort is a type of sorting.
* It is used for sorting 'n' (number of items) elements.
* It compares all the elements one by one and sorts them based on their values.



* The above diagram represents how bubble sort actually works. This sort takes O(n2) time. It
* sort starts with first two elements. It compares the element to check which one is starts with the first two elements and sorts them in ascending order.
* Bubble greater.
* In the above diagram, element 40 is greater than 10, so these values must be swapped. This operation continues until the array is sorted in ascending order.

**Discussion:**

**What I learned from this test is given below :**

* I have learn to about Bubble Sort .
* Knowing about use loop and condition in program .
* Learn about sorting element in array program .
* Prepare an Algorithm for this program .
* At last successfully run this program .

It is used in academia to introduce 1st-year computer science students to the concept of a sorting algorithm.

It is comparably simpler to understand than some other sorting algorithms.  You quickly move beyond it after it is introduced.

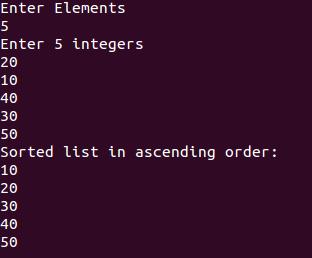
In the industry, I do not know of a single use of it, as there are several others that are significantly more efficient.

*“The only significant advantage that bubble sort has over most other implementations, even quicksort, but not insertion sort, is that the ability to detect that the list is sorted efficiently is built into the algorithm. When the list is already sorted (best-case), the complexity of bubble sort is only O(n). By contrast, most other algorithms, even those with better average-case complexity, perform their entire sorting process on the set and thus are more complex. However, not only does insertion sort have this mechanism too, but it also performs better on a list that is substantially sorted (having a small number of inversions).”*

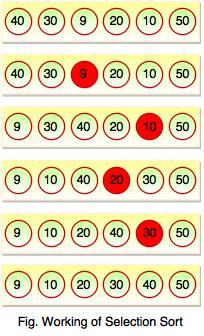
Bubble sort should be avoided in the case of large collections. It will not be efficient in the case of a reverse-ordered collection.

**Source Code:**

#### **Example: Program for Bubble Sort**

#include <stdio.h>  
void bubble\_sort(long [], long);  
  
int main()  
{  
  long array[100], n, c, d, swap;   
  printf("Enter Elements\n");  
  scanf("%ld", &n);   
  printf("Enter %ld integers\n", n);   
  for (c = 0; c < n; c++)  
    scanf("%ld", &array[c]);   
  bubble\_sort(array, n);   
  printf("Sorted list in ascending order:\n");   
  for ( c = 0 ; c < n ; c++ )  
     printf("%ld\n", array[c]);   
  return 0;  
}  
void bubble\_sort(long list[], long n)  
{  
  long c, d, t;   
  for (c = 0 ; c < ( n - 1 ); c++)  
  {  
    for (d = 0 ; d < n - c - 1; d++)  
    {  
      if (list[d] > list[d+1])  
      {  
        /\* Swapping \*/   
        t         = list[d];  
        list[d]   = list[d+1];  
        list[d+1] = t;  
      }  
    }  
  }  
}  
**Output:**  


**2. Selection Sort:** Selection sort is a simple sorting algorithm which finds the smallest element in the array and exchanges it with the element in the first position. Then finds the second smallest element and exchanges it with the element in the second position and continues until the entire array is sorted.



In the above diagram, the smallest element is found in first pass that is 9 and it is placed at the first position. In second pass, smallest element is searched from the rest of the element excluding first element. Selection sort keeps doing this, until the array is sorted.

**Discussion:**

**What I learned from this test is given below :**

* I have learn to about Selection Sort .
* Knowing about use loop and condition in program .
* Learn about sorting element in array program .
* Prepare an Algorithm for this program .
* At last successfully run this program .

Selection sort can also be used on list structures that make add and remove efficient, such as a [linked list](https://en.wikipedia.org/wiki/Linked_list). In this case it is more common to *remove* the minimum element from the remainder of the list, and then *insert* it at the end of the values sorted so far.

**Source Code:**

**Example: Program for Selection Sort**

#include <stdio.h>  
int main()  
{  
   int array[100], n, c, d, position, swap;   
   printf("Enter number of elements\n");  
   scanf("%d", &n);   
   printf("Enter %d integers\n", n);   
   for ( c = 0 ; c < n ; c++ )  
      scanf("%d", &array[c]);   
   for ( c = 0 ; c < ( n - 1 ) ; c++ )  
   {  
      position = c;   
      for ( d = c + 1 ; d < n ; d++ )  
      {  
         if ( array[position] > array[d] )  
            position = d;  
      }  
      if ( position != c )  
      {  
         swap = array[c];  
         array[c] = array[position];  
         array[position] = swap;  
      }  
   }   
   printf("Sorted list in ascending order:\n");   
   for ( c = 0 ; c < n ; c++ )  
      printf("%d\n", array[c]);   
   return 0;  
}

**Output:**  
  
