## INSERTION SORT

The insertion sort method sorts a list of elements by inserting each successive element in the previously sorted sub list.

It performs n-1 passes.

For pass=1, through n-1, insertion sort ensures that the elements in position 0 through P are in sorted order.

In pass p, we move the $p^{th}$ element to left until its correct place is found among the first elements.

Example: 34, 8, 64, 51, 32, 21

| | **[0]** | **[1]** | **[2]** | **[3]** | **[4]** | **[5]** | **Passes moved** |
|---|---|---|---|---|---|---|---|
| | 34 | 8 | 64 | 51 | 32 | 21 | |
| After p=1 | 8 | 34 | 64 | 51 | 32 | 21 | 1 |
| After p=2 | 8 | 34 | 64 | 51 | 32 | 21 | 0 |
| After p=3 | 8 | 34 | 51 | 64 | 32 | 21 | 3 |
| After p=4 | 8 | 32 | 34 | 51 | 64 | 21 | 3 |
| After p=5 | 8 | 21 | 32 | 34 | 51 | 64 | 4 |

Sorted list is 8, 21, 32, 34, 51, 64

The element in position p is saved in Tmp and all the larger elements are moved one spot to the right. Then tmp is placed on the correct order.

**Algorithm:**

```
void insertion_sort(int a[], int n)
{
inti,p,tmp;
```

```
for(p=1;p<n;p++)
{
tmp=a[p];
for(j=p;j>0&&a[j-1]>tmp;j--)
a[j]=a[j-1];
a[j]=tmp;
}
}
```

**Advantages:**

- Easy to implement
- Performs well for smaller lists.

**Disadvantages:**

- For larger sized lists($n^2$) is not best efficiency
- No. of elements to be shifted, if no. of elements is high.