Experiment 3
Implementation of Insertion Sort Algorithm

Date: 06-08-2020

Aim: To sort a given set of elements using the insertion sort algorithm

Data Structures Used: Arrays
Operations Used: Comparison and Swapping

Algorithm:

Input : An Unsorted Array A[L...U] //L and U are the lower and upper bounds respectively

Output: Sorted Array

Steps:
Step 1 : Start
Step 2 : i ← L+1
Step 3 : while i<=U
    Step 1 : temp ← A[i]
    Step 2 : j ← i-1
    Step 3 : while j>=0
        Step 1 : if A[j]<temp
            Step 1 : End While
        Step 2 : End if
        Step 3 : A[j+1] ← A[j]
        Step 4 : j ← j-1
    Step 4 : End while
    Step 5 : A[j+1] ← temp
    Step 6 : I ← i+1
Step 4 : End while
Step 5 : Stop

Description of the Algorithm:
The insertion sort as the name suggests uses the insertion algorithm for an element in an array to sort the array. Initially the algorithm inserts the second element in the array in the proper position by moving elements before it to the right, just like the insertion algorithm for a sorted array, then we get the first two elements are sorted and the rest of the array is unsorted. Then the outer for loop moves on to the third element and the inner for loop inserts the element in the proper position of the sorted sub-array. This continuous till the last element resulting in the array being completely sorted.
The worst case complexity is $O(n^2)$ and the best case complexity is O(n).

Result : The Program was successfully complied and the required output was obtained.

Program:

```
#include<stdio.h>

void printarr(int *a, int n){
    for(int i = 0;i<n;i++){
        printf("%d ",*(a+i));
    }
}

void enterValues(int *a, int n){
    printf("Enter the elements of the array: ");
    for(int i =0; i<n;i++){
        scanf("%d%*c",a+i);
    }
}

void main(){
    int i,j;
    int n,comp=0,swaps=0;
    int arr[100];
    int temp;
    char c;

    printf("Enter the number of elements in the array: ");
    scanf("%d%*c",&n);
    enterValues(arr,n);
    printf("\n\n");

    for(i=1;i<n;i++){
        temp=arr[i];
        for(j=i-1;j>=0;j--){
            comp++;
            if(arr[j]<temp)break;
            arr[j+1]=arr[j];
            swaps++;
        }
        if(j!=i-1){
            swaps++;
            arr[j+1]=temp;
        }
        printf("The array after %d step is : ",i);
        printarr(arr,n);printf("\n");

    }

    printf("\nThe sorted array is -> ");
    printarr(arr,n);printf("\n\n");
    printf("The total number of comparisons = %d\nThe total number of swaps = %d\
n",comp,swaps);
}
```

Sample Input 1:
```
5
98 45 34 32 12
```

Sample Output 1:
```
Enter the number of elements in the array: 5
Enter the elements of the array: 98 45 34 32 12

The array after 1 step is : 45 98 34 32 12
The array after 2 step is : 34 45 98 32 12
The array after 3 step is : 32 34 45 98 12
The array after 4 step is : 12 32 34 45 98

The sorted array is -> 12 32 34 45 98

The total number of comparisons = 10
The total number of swaps = 14
```

Sample Input 2:
```
5
12 32 34 45 98
```

Sample Output 2:
```
Enter the number of elements in the array: 5
Enter the elements of the array: 12 32 34 45 98

The array after 1 step is : 12 32 34 45 98
The array after 2 step is : 12 32 34 45 98
The array after 3 step is : 12 32 34 45 98
The array after 4 step is : 12 32 34 45 98

The sorted array is -> 12 32 34 45 98

The total number of comparisons = 4
The total number of swaps = 0
```

Sample Input 3:
```
5
45 32 12 98 34
```

Sample Output 3:
```
Enter the number of elements in the array: 5
Enter the elements of the array: 45 32 12 98 34

The array after 1 step is : 32 45 12 98 34
The array after 2 step is : 12 32 45 98 34
The array after 3 step is : 12 32 45 98 34
The array after 4 step is : 12 32 34 45 98

The sorted array is -> 12 32 34 45 98

The total number of comparisons = 7
The total number of swaps = 8
```