Aim : Write a menu driven Program to Sort an Array Using insertion Sort.

Objective : To implement Sorting algorithm having time complexity of $O(n^2)$ in worst case.

Aim: Write a menu driven Program to sort an Array using Insertion Sort.

Objective: To implement sorting algorithm having time complexity of $O(n^2)$ in worst case.

Theory: Insertion sort is simple & Intutive sorting Algorithm that builds up the sorted array one element at a time. It works by iteratively taking an element from the unsorted part of the array & inserting it into it's correct position within the sorted portion.

The algorithm maintains two sub-arrays within the original way i.e the sorted sub-array that is initially empty and the unsorted sub-array, containing the remaining elements.

The process begins by considering the 1st element of the array as the sorted portion. The Algo. then iterates through the unsorted portion, taking one element at a time & inserting it into it's proper position within the sorted sub-array. Insertion sort sort is an in-place sorting algorithm, ie it requires a constant amount of additional memory for temp vari-ables

Code :

```
#include<stdio.h>

void insertionSort(int arr[], int n) {
  int i, key, j;

for (i=1; i<n; i++){
    key = arr[i];
    j= i-1;
     while (j>=0 && arr[j]>key){
        arr[j+1] = arr[j];
        j=j-1;
     }
arr[j+1] = key;
  }
}

void printArray (int arr[], int size){
    for(int i=0; i<size; i++) {
      printf("%d", arr[i]);
    }

    printf("\n");
}
```

```
int main ( ){
  int n;

  printf (" Enter the number of elements :");
  Scanf [" %d", &n );

  int arr [n];
  printf (" Enter the elements:");

  for (inti= 0 ; i<n ; ; i++)
     {
       printf (" Element %d", i+1);
       scanf (" %d", arr [i] );
     }
  printf (" Original Array is \n:");
  printArray (arr ,n);

  insertionSort (arr,n);

  printf (" Sorted Array is :");
  printArray (arr ,n);

  return 0;
}
```

Output : Enter the number of elements : 5
Enter the elements :
Element 1 : 50
Element 2 : 40
Element 3 : 30
Element 4 : 20
Element 5 : 10
Original Array : 50 40 30 20 10
Sorted Array : 10 20 30 40 50

Alogorithm : i] Input
Take an array of elements as input
ii] Outer loop
Start an outer loop from the second element (index 1) to the last element (index n-1), where 'n' is the number of elements in array.
iii] Select Element :
Select the current element from the unsorted sub-array, called as a key.
iv] Inner loop
Start an inner loop from the current element's index & move towards the begining of the array [index] from i to 0
Compare the key with each element in the sorted sub-array.

Conclusion : Program of insertion sort is implemented & executed successfully using C Programming Langauge.

If the key is smaller than the current key/element, move the current element one position ahead to make space for the key.

v] Insert key

Insert the key into it's correct position in the sorted sub-array.

vi] Repeat

Repeat steps 3-5 for each iteration of the outer Loop.

vii] The output

The array is sorted when outer loop completes.

Conclusion: Program of insertion sort is implemented & executed successfully using C Programming Language.