**Insertion Sort**

Insertion sort is a sorting algorithm that places an unsorted element at its suitable place in each iteration.

**Insertion Sort Pseudocode:**

procedure insertionSort(A : array of items )

int holePosition

int valueInsert

for i = 1 to length(A) inclusive do:

valueInsert = A[i]

holePosition = I - 1

while holePosition>0 and A[holePosition] > valueInsert do:

A[holePosition + 1] = A[holePosition]

holePosition = holePosition -1

end while

A[holePosition + 1] = valueInsert

end for

end procedure

**Complexities:** Time Complexity: Best – O(n), Average – O(n2), Worst – O(n2)

Space Complexity: O(1)

Stability: Yes

**Applications:** The insertion sort is used when

* the array has a small number of elements
* there are only a few elements left to be sorted

**Source Code:**

using System;

namespace InsertionSort

{

class Program

{

static void Main(String[] args)

{

Input();

}

static void Input()

{

Console.Write("Enter Number of Items: ");

int noOfItems = Convert.ToInt32(Console.ReadLine());

int[] itemsList = new int[noOfItems];

Console.Write("Enter Items: ");

for (int i = 0; i < noOfItems; i++)

{

itemsList[i] = Convert.ToInt32(Console.ReadLine());

}

Console.Write("For ascending write 'a' or descending write 'd': ");

char order = Convert.ToChar(Console.ReadLine());

InsertionSort(itemsList, order);

}

static void InsertionSort(int[] itemsList, char order)

{

for (int i = 1; i < itemsList.Length; i++)

{

int item = itemsList[i];

int j = i - 1;

if (order == 'a')

{

while (j >= 0 && itemsList[j] > item)

{

itemsList[j + 1] = itemsList[j];

j--;

}

}

else if (order == 'd')

{

while (j >= 0 && itemsList[j] < item)

{

itemsList[j + 1] = itemsList[j];

j--;

}

}

itemsList[j+1] = item;

}

Output(itemsList);

}

static void Output(int[] itemList)

{

Console.Write("After sorting: ");

for (int i = 0; i < itemList.Length; i++)

{

Console.Write($"{itemList[i]}\t");

}

}

}

}