**EXPERIMENT -18**

**AIM:** Write a program to implement insertion sort on array.

**ALGORITHM:**

**SOURCE CODE:**

#include <bits/stdc++.h>

using namespace std;

void insertion\_sort (int arr[], int n) {

for (int i=1; i<n; i++) {

int temp = arr[i];

for (int j=i-1; j>=0; j--) {

if (temp < arr[j]) {

arr[j+1] = arr[j];

arr[j] = temp;

}

}

}

}

int main() {

int size;

cout << "Enter the size of the array : ";

cin >> size;

int arr[size];

cout << "Enter your elements : ";

for (int i=0; i<size; i++) {

cin >> arr[i];

}

insertion\_sort(arr, size);

cout << "Sorted array: ";

for (int i=0; i<size; i++) {

cout << arr[i] << " ";

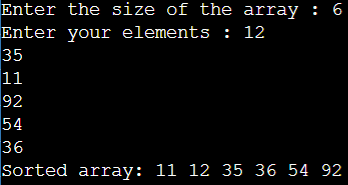
}

}

**EXPERIMENT -18**

**AIM:** Write a program to implement insertion sort on array.

**OUTPUT:**

****

**EXPERIMENT -19**

**AIM:** Write a program to implement quick sort on array.

**ALGORITHM:**

**SOURCE CODE:**

#include <iostream>

using namespace std;

int partition(int arr[], int start, int end){

int pivot = arr[start];

int count = 0;

for (int i = start + 1; i <= end; i++) {

if (arr[i] <= pivot)

count++;

}

int pivotIndex = start + count;

swap(arr[pivotIndex], arr[start]);

int i = start, j = end;

while (i < pivotIndex && j > pivotIndex) {

while (arr[i] <= pivot) {

i++;

}

while (arr[j] > pivot) {

j--;

}

if (i < pivotIndex && j > pivotIndex) {

swap(arr[i++], arr[j--]);

}

}

return pivotIndex;

}

void quickSort(int arr[], int start, int end){

if (start >= end)

return;

int p = partition(arr, start, end);

quickSort(arr, start, p - 1);

quickSort(arr, p + 1, end);

}

int main(){

int size;

cout << "Enter the size of the array : ";

cin >> size;

int arr[size];

cout << "Enter your elements : ";

for (int i=0; i<size; i++) {

cin >> arr[i];

}

quickSort(arr, 0, size - 1);

cout<<"Sorted array:"<<endl;

for (int i = 0; i < size; i++) {

cout << arr[i] << " ";

}

return 0;}

**EXPERIMENT -19**

**AIM:** Write a program to implement quick sort on array.

**OUTPUT:**

**Text

Description automatically generated**