```
============================================================
def partition(array, low, high):
    pivot = array[high]
    i = low - 1

    for j in range(low, high):
        if array[j] <= pivot:
            i += 1
            array[i], array[j] = array[j], array[i]

    array[i+1], array[high] = array[high], array[i+1]
    return i+1

def quicksort(array, low=0, high=None):
    if high is None:
        high = len(array) - 1

    if low < high:
        pivot_index = partition(array, low, high)
        quicksort(array, low, pivot_index-1)
        quicksort(array, pivot_index+1, high)

my_array = [64, 34, 25, 12, 22, 11, 90, 5]
quicksort(my_array)
print("Sorted array:", my_array)

#Python


============================================================
```

```
#include <iostream>

using namespace std;


// Function to swap two elements

void swap(int* a, int* b) {

 int t = *a;

 *a = *b;

 *b = t;

}


// Partitioning the array and returning the pivot index
```

```
int partition(int arr[], int low, int high) {

 int pivot = arr[high]; // Choosing the last element as the pivot

int i = (low - 1); // Index of smaller element


 for (int j = low; j <= high - 1; j++) {

 // If the current element is smaller than or equal to the pivot   if

(arr[j] <= pivot) {

 i++; // Increment index of smaller element

swap(&arr[i], &arr[j]);

 }

 }


 swap(&arr[i + 1], &arr[high]);

 return (i + 1);

}


// The main function that implements QuickSort

void quickSort(int arr[], int low, int high) {

 if (low < high) {

 // pi is partitioning index, arr[p] is now at the right place
 int pi = partition(arr, low, high);


 // Separately sort elements before partition and after partition

quickSort(arr, low, pi - 1);

 quickSort(arr, pi + 1, high);

 }

}


// Function to print an array

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

// Driver code
int main() {
 int arr[] = {64, 25, 12, 22, 11};
 int n = sizeof(arr) / sizeof(arr[0]);

 cout << "Original array: " << endl;
 printArray(arr, n);

 quickSort(arr, 0, n - 1);

 cout << "Sorted array: " << endl;
 printArray(arr, n);

 return 0;
}
```