Insertion Sort in C++

```
#include <iostream>
using namespace std;

class InsertionSort {
public:
    // Function to perform insertion sort on array arr of size n
    void insertionSort(int arr[], int n) {
        for (int i = 1; i < n; i++) {
            insert(arr, i);
        }
    }

private:
    // Helper function to insert arr[i] into the sorted sub-array arr[0...i-1]
    void insert(int arr[], int i) {
        int key = arr[i]; // Element to be inserted
        int j = i - 1;    // Start comparing with the previous element

        // Move elements of arr[0..i-1], that are greater than key, to one position ahead of their current position
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key; // Place key at its correct position
    }
};

int main() {
    InsertionSort solution;

    // Hardcoded input array
    int arr[] = {5, 2, 9, 1, 5, 6};
    int n = sizeof(arr) / sizeof(arr[0]);

    // Sorting the array using insertion sort
    solution.insertionSort(arr, n);

    // Printing the sorted array
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;

    return 0;
}
```

Let's dry run your **Insertion Sort** code step by step with the input:

int arr[] = {5, 2, 9, 1, 5, 6};

🔄 Insertion Sort Dry Run Table

i	Key	Initial Array State	Comparison Index (j)	Action Taken	Updated Array
1	2	[5, 2, 9, 1, 5, 6]	j = 0 (5 > 2)	Shift 5 to index 1	[5, 5, 9, 1, 5, 6]
			j = -1	Insert 2 at index 0	[2, 5, 9, 1, 5, 6]
2	9	[2, 5, 9, 1, 5, 6]	j = 1 (5 < 9)	No shifting, insert 9 at index 2	[2, 5, 9, 1, 5, 6]
3	1	[2, 5, 9, 1, 5, 6]	j = 2 (9 > 1)	Shift 9 to index 3	[2, 5, 9, 9, 5, 6]
			j = 1 (5 > 1)	Shift 5 to index 2	[2, 5, 5, 9, 5, 6]
			j = 0 (2 > 1)	Shift 2 to index 1	[2, 2, 5, 9, 5, 6]
			j = -1	Insert 1 at index 0	[1, 2, 5, 9, 5, 6]
4	5	[1, 2, 5, 9, 5, 6]	j = 3 (9 > 5)	Shift 9 to index 4	[1, 2, 5, 9, 9, 6]
			j = 2 (5 == 5)	No shifting (stable), insert 5 at index 3	[1, 2, 5, 5, 9, 6]
5	6	[1, 2, 5, 5, 9, 6]	j = 4 (9 > 6)	Shift 9 to index 5	[1, 2, 5, 5, 9, 9]
			j = 3 (5 < 6)	Insert 6 at index 4	[1, 2, 5, 5, 6, 9]

✅ Final Sorted Array:

	[1, 2, 5, 5, 6, 9]
1 2 5 5 6 9	