Sort 012 in C++

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

class Sort012 {
public:
    void sort012(vector<int>& arr) {
        int i = 0, j = 0, k = arr.size() - 1;
        while (j <= k) {
            if (arr[j] == 0) {
                swap(arr[i], arr[j]);
                i++;
                j++;
            } else if (arr[j] == 1) {
                j++;
            } else {
                swap(arr[j], arr[k]);
                k--;
            }
        }
    }

    void swap(int& a, int& b) {
        int temp = a;
        a = b;
        b = temp;
    }
};

int main() {
    // Hardcoded input vector
    vector<int> arr = {0, 1, 2, 0, 1, 2, 1, 0, 2, 1};

    // Print the original array
    cout << "Original array: ";
    for (int num : arr) {
        cout << num << " ";
    }
    cout << endl;

    // Create an instance of Sort012 class
    Sort012 solution;

    // Call sort012 to sort the array
    solution.sort012(arr);

    // Print the sorted array
    cout << "Sorted array: ";
    for (int num : arr) {
        cout << num << " ";
    }
    cout << endl;

    return 0;
}
```

Input Array:

{0, 1, 2, 0, 1, 2, 1, 0, 2, 1}

🧠 Three-pointer strategy:

- i: points to the position where the next 0 should go.
- j: current index being processed.
- k: points to the position where the next 2 should go.

📋 Dry Run Table:

Step	i	j	k	arr[j]	Action	Array State
1	0	0	9	0	swap(i,j), ++i,++j	0 1 2 0 1 2 1 0 2 1
2	1	1	9	1	j++	0 1 2 0 1 2 1 0 2 1
3	1	2	9	2	swap(j,k), k--	0 1 1 0 1 2 1 0 2 2
4	1	2	8	1	j++	0 1 1 0 1 2 1 0 2 2
5	1	3	8	0	swap(i,j), ++i,++j	0 0 1 1 1 2 1 0 2 2
6	2	4	8	1	j++	0 0 1 1 1 2 1 0 2 2
7	2	5	8	2	swap(j,k), k--	0 0 1 1 1 2 1 0 2 2
8	2	5	7	2	swap(j,k), k--	0 0 1 1 1 0 1 2 2 2
9	2	5	6	0	swap(i,j), ++i,++j	0 0 0 1 1 1 1 2 2 2
10	3	6	6	1	j++	0 0 0 1 1 1 1 2 2 2

✅ Final Output:

Sorted array: 0 0 0 1 1 1 1 2 2 2

```
Original array: 0 1 2 0 1 2 1 0 2 1

Sorted array: 0 0 0 1 1 1 1 2 2 2
```

Sort Colors in C++

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

class SortColors {
public:
    void sortColors(vector<int>& nums) {
        int n = nums.size();
        int i = 0, j = 0, k = n - 1;
        while (j <= k) {
            if (nums[j] == 0) {
                swap(nums[i], nums[j]);
                i++;
                j++;
            } else if (nums[j] == 1) {
                j++;
            } else {
                swap(nums[j], nums[k]);
                k--;
            }
        }
    }

    void swap(int& a, int& b) {
        int temp = a;
        a = b;
        b = temp;
    }
};

int main() {
    // Hardcoded input vector
    vector<int> arr = {0, 1, 2, 0, 1, 2, 1, 0, 2, 1};

    // Print the original array
    cout << "Original array: ";
    for (int num : arr) {
        cout << num << " ";
    }
    cout << endl;

    // Create an instance of SortColors class
    SortColors solution;

    // Call sortColors to sort the array
    solution.sortColors(arr);

    // Print the sorted array
    cout << "Sorted array: ";
    for (int num : arr) {
        cout << num << " ";
    }
    cout << endl;

    return 0;
}
```

Input

vector<int> arr = {0, 1, 2, 0, 1, 2, 1, 0, 2, 1};

📌 Initial Setup

- i = 0 (position to place next 0)
- j = 0 (current index)
- k = 9 (position to place next 2)
- Size n = 10

🔍 Dry Run Table

Step	i	j	k	nums[j]	Action	Resulting Array
1	0	0	9	0	swap(i,j), ++i, ++j	0 1 2 0 1 2 1 0 2 1
2	1	1	9	1	++j	0 1 2 0 1 2 1 0 2 1
3	1	2	9	2	swap(j,k), --k	0 1 1 0 1 2 1 0 2 2
4	1	2	8	1	++j	0 1 1 0 1 2 1 0 2 2
5	1	3	8	0	swap(i,j), ++i, ++j	0 0 1 1 1 2 1 0 2 2
6	2	4	8	1	++j	0 0 1 1 1 2 1 0 2 2
7	2	5	8	2	swap(j,k), --k	0 0 1 1 1 2 1 0 2 2
8	2	5	7	2	swap(j,k), --k	0 0 1 1 1 0 1 2 2 2
9	2	5	6	0	swap(i,j), ++i, ++j	0 0 0 1 1 1 1 2 2 2
10	3	6	6	1	++j	0 0 0 1 1 1 1 2 2 2

🎉 Final Sorted Output

Sorted array: 0 0 0 1 1 1 1 2 2 2

```
Original array: 0 1 2 0 1 2 1 0 2 1

Sorted array: 0 0 0 1 1 1 1 2 2 2
```