First Missing Positive in C++

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

int firstMissingPositive(vector<int>& nums) {
    int n = nums.size();

    int i = 0;
    while (i < n) {
        if (nums[i] == i + 1) {
            i++;
            continue;
        }

        if (nums[i] <= 0 || nums[i] > n) {
            i++;
            continue;
        }

        int idx1 = i;
        int idx2 = nums[i] - 1;

        if (nums[idx1] == nums[idx2]) {
            i++;
            continue;
        }

        int temp = nums[idx1];
        nums[idx1] = nums[idx2];
        nums[idx2] = temp;
    }

    for (int j = 0; j < n; j++) {
        if (nums[j] != j + 1) {
            return j + 1;
        }
    }

    return n + 1;
}

int main() {
    vector<int> nums = {3, 4, -1, 1};
    int result = firstMissingPositive(nums);
    cout << "First missing positive: " << result << endl;
    return 0;
}
```

Input:

vector<int> nums = {3, 4, -1, 1};

💡 Goal:

Find the **smallest positive integer** that is **missing** from the array.

🌀 Algorithm Insight:

You're trying to **place each positive integer x (1 ≤ x ≤ n)** at index x - 1 using cyclic swaps.

🔍 Dry Run Table:

🔁 While loop swaps

Step	i	nums[i]	Action	nums after
1	0	3	swap nums[0] with nums[2] (index 2 = 3 - 1)	{-1, 4, 3, 1}
2	0	-1	invalid (<= 0), move to i = 1	{-1, 4, 3, 1}
3	1	4	swap nums[1] with nums[3] (index 3 = 4 - 1)	{-1, 1, 3, 4}
4	1	1	swap nums[1] with nums[0] (index 0 = 1 - 1)	{1, -1, 3, 4}
5	1	-1	invalid, move to i = 2	{1, -1, 3, 4}
6	2	3	already at correct index (2 = 3 - 1)	no change
7	3	4	already at correct index (3 = 4 - 1)	no change

📌 Final nums array after placements:

{1, -1, 3, 4}

✅ Final Check:

Go through the array to find first j where nums[j] ! = j + 1:

j	nums[j]	j + 1	Match?
0	1	1	✅
1	-1	2	❌ → return 2

	📄 **Output:** First missing positive: 2
First missing positive: 2	