Count Of Subarrays With Equal 0 and 1 in C++

```
#include <iostream>
#include <unordered_map>
#include <vector>

using namespace std;

int solution(vector<int>& arr) {
    unordered_map<int, int> map;
    int ans = 0;
    map[0] = 1; // Initialize with sum 0 having
count 1
    int sum = 0;

    for (int val : arr) {
        // Treat 0 as -1 for sum calculation
        if (val == 0) {
            sum += -1;
        } else {
            sum += 1;
        }

        if (map.find(sum) != map.end()) {
            ans += map[sum];
            map[sum]++;
        } else {
            map[sum] = 1;
        }
    }

    return ans;
}

int main() {
    vector<int> arr = {0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1,
1, 1};
    cout << solution(arr) << endl; // Output the
result

    return 0;
}
```

Dry Run for Input:

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

Initial Values:

- ans = 0
- map = {0: 1}
- sum = 0

Iteration Breakdown:

i	arr[i]	sum (cumulative sum)	map[sum]	ans (after update)	map (updated)
0	0	-1	map[-1] = 0	0	{0: 1, -1: 1}
1	0	-2	map[-2] = 0	0	{0: 1, -1: 1, -2: 1}
2	1	-1	map[-1] = 1	1	{0: 1, -1: 2, -2: 1}
3	0	-2	map[-2] = 1	1	{0: 1, -1: 2, -2: 2}
4	1	-1	map[-1] = 2	3	{0: 1, -1: 3, -2: 2}
5	0	-2	map[-2] = 2	3	{0: 1, -1: 3, -2: 3}
6	1	-1	map[-1] = 3	6	{0: 1, -1: 4, -2: 3}
7	1	0	map[0] = 1	7	{0: 2, -1: 4, -2: 3}
8	0	-1	map[-1] = 4	11	{0: 2, -1: 5, -2: 3}
9	0	-2	map[-2] = 3	14	{0: 2, -1: 5, -2: 4}
10	1	-1	map[-1] = 5	19	{0: 2, -1: 6, -2: 4}
11	1	0	map[0] = 2	21	{0: 3, -1: 6, -2: 4}
12	1	1	map[1] = 0	24	{0: 3, -1: 6, -2: 4, 1: 1}

Explanation of Each Iteration:

- **At i = 0**:
  - arr[0] = 0
  - Treat 0 as -1.
  - sum = -1.
  - map[sum] = map[-1] = 0, so we add 0 to ans.
  - We update map[-1] = 1.
- **At i = 1**:
  - arr[1] = 0
  - Treat 0 as -1.
  - sum = -2.
  - map[sum] = map[-2] = 0, so we add 0 to ans.

	○ We update map[-2] = 1. • **At i = 2**: ○ arr[2] = 1 ○ sum = -1. ○ map[sum] = map[-1] = 1, so we add 1 to ans. ○ We update map[-1] = 2. • **At i = 3**: ○ arr[3] = 0 ○ Treat 0 as -1. ○ sum = -2. ○ map[sum] = map[-2] = 1, so we add 1 to ans. ○ We update map[-2] = 2. • **At i = 4**: ○ arr[4] = 1 ○ sum = -1. ○ map[sum] = map[-1] = 2, so we add 2 to ans. ○ We update map[-1] = 3. • **At i = 5**: ○ arr[5] = 0 ○ Treat 0 as -1. ○ sum = -2. ○ map[sum] = map[-2] = 2, so we add 2 to ans. ○ We update map[-2] = 3. • **At i = 6**: ○ arr[6] = 1 ○ sum = -1. ○ map[sum] = map[-1] = 3, so we add 3 to ans. ○ We update map[-1] = 4. • **At i = 7**: ○ arr[7] = 1 ○ sum = 0. ○ map[sum] = map[0] = 2, so we add 2 to ans. ○ We update map[0] = 3. • **At i = 8**: ○ arr[8] = 0 ○ Treat 0 as -1. ○ sum = -1. ○ map[sum] = map[-1] = 4, so we add 4 to ans. ○ We update map[-1] = 5. • **At i = 9**: ○ arr[9] = 0 ○ Treat 0 as -1. ○ sum = -2. ○ map[sum] = map[-2] = 3, so we add 3 to ans. ○ We update map[-2] = 4. • **At i = 10**: ○ arr[10] = 1 ○ sum = -1. ○ map[sum] = map[-1] = 5, so we add 5 to ans. ○ We update map[-1] = 6.

	• **At i = 11**:
	  ○ arr[11] = 1
	  ○ sum = 0.
	  ○ map[sum] = map[0] = 3, so we add 3 to ans.
	  ○ We update map[0] = 4.
	• **At i = 12**:
	  ○ arr[12] = 1
	  ○ sum = 1.
	  ○ map[sum] = map[1] = 0, so we add 0 to ans.
	  ○ We update map[1] = 1.
	
	Final Result:
	
	• The total count of subarrays whose sum is 0 is **24**.

Output: 24	