Contiguous Array in C++

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

int sol(int arr[], int n) {
    int ans = 0;
    unordered_map<int, int> map;
    map[0] = -1;
    int sum = 0;

    for (int i = 0; i < n; i++) {
        if (arr[i] == 0) {
            sum += -1;
        } else if (arr[i] == 1) {
            sum += +1;
        }

        if (map.find(sum) != map.end()) {
            int idx = map[sum];
            int len = i - idx;
            if (len > ans) {
                ans = len;
            }
        } else {
            map[sum] = i;
        }
    }
    return ans;
}

int main() {
    int arr[] = {0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1};
    int n = sizeof(arr) / sizeof(arr[0]);
    cout << sol(arr, n) << endl; // Output: 10

    return 0;
}
```

Dry Run:

Given input:

```
int arr[] = {0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1};
int n = sizeof(arr) / sizeof(arr[0]);
```

Step-by-Step Breakdown:

Initial Values:

- ans = 0 (stores the longest subarray length)
- map = {0: -1} (maps cumulative sum to the first occurrence index)
- sum = 0 (initial cumulative sum)

Iteration by Iteration Walkthrough:

i	arr[i]	sum (cumulative sum)	map (sum -> index)	Length (len)	Updated ans
0	0	-1	{0: -1, -1: 0}	0 - (-1) = 1	1
1	0	-2	{0: -1, -1: 0, -2: 1}	1 - (-1) = 2	2
2	1	-1	{0: -1, -1: 0, -2: 1}	2 - 0 = 2	2
3	0	-2	{0: -1, -1: 0, -2: 1}	3 - 1 = 2	2
4	1	-1	{0: -1, -1: 0, -2: 1}	4 - 0 = 4	4
5	0	-2	{0: -1, -1: 0, -2: 1}	5 - 1 = 4	4
6	1	-1	{0: -1, -1: 0, -2: 1}	6 - 0 = 6	6
7	1	0	{0: -1, -1: 0, -2: 1}	7 - (-1) = 8	8
8	0	-1	{0: -1, -1: 0, -2: 1}	8 - 0 = 8	8
9	0	-2	{0: -1, -1: 0, -2: 1}	9 - 1 = 8	8
10	1	-1	{0: -1, -1: 0, -2: 1}	10 - 0 = 10	10
11	1	0	{0: -1, -1: 0,	11 - (-1) = 12	12

			-2: 1}		
12	1	1	{0: -1, -1: 0, -2: 1}	12 - (-1) = 14	14

Correct Analysis:

- The **longest subarray** with equal numbers of 0s and 1s spans from index 2 to 11 (inclusive), making the subarray length **12**.

Final Output:

12

Output:
12