Perfect Square In C++

```
#include <iostream>
#include <vector>
#include <climits>
#include <cmath>
using namespace std;

int main() {
    vector<int> arr = {0, 1, 2, 3, 1, 2, 3, 4, 2, 1, 2, 3};
    int n = arr.size();
    vector<int> dp(n + 1, INT_MAX); // dp array where dp[i] represents the minimum number of perfect squares summing up to i
    //int dp[n+1]={INT_MAX};
    dp[0] = 0; // Base case: 0 requires 0 squares
    dp[1] = 1; // 1 requires 1 square (1)

    for (int i = 2; i <= n; i++) {
        for (int j = 1; j * j <= i; j++) {
            dp[i] = min(dp[i], dp[i - j * j] + 1);
        }
    }

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

    return 0;
}
```

Dry Run with Table

We compute dp[i] for i = 0 to 12 using the given transition formula.

i	Perfect Squares (≤ i)	dp[i] Computation	dp[i]
0	-	dp[0] = 0	**0**
1	1	dp[1] = min(dp[1 - 1] + 1) = 1	**1**
2	1	dp[2] = min(dp[2 - 1] + 1) = 2	**2**
3	1	dp[3] = min(dp[3 - 1] + 1) = 3	**3**
4	1, 4	dp[4] = min(dp[4 - 1] + 1, dp[4 - 4] + 1) = min(4, 1) = 1	**1**
5	1, 4	dp[5] = min(dp[5 - 1] + 1, dp[5 - 4] + 1) = min(2, 2) = 2	**2**
6	1, 4	dp[6] = min(dp[6 - 1] + 1, dp[6 - 4] + 1) = min(3, 3) = 3	**3**
7	1, 4	dp[7] = min(dp[7 - 1] + 1, dp[7 - 4] + 1) = min(4, 4) = 4	**4**
8	1, 4	dp[8] = min(dp[8 - 1] + 1, dp[8 - 4] + 1) = min(5, 2) = 2	**2**
9	1, 4, 9	dp[9] = min(dp[9 - 1] + 1, dp[9 - 4] + 1, dp[9 - 9] + 1) = min(3, 3, 1) = 1	**1**
10	1, 4, 9	dp[10] = min(dp[10 - 1] + 1, dp[10 - 4] + 1, dp[10 - 9] + 1) = min(2, 4, 2) = 2	**2**
11	1, 4, 9	dp[11] = min(dp[11 - 1] + 1, dp[11 - 4] + 1, dp[11 - 9] + 1) = min(3, 5, 3) = 3	**3**
12	1, 4, 9	dp[12] = min(dp[12 - 1] + 1, dp[12 - 4] + 1, dp[12 - 9] + 1) = min(4, 3, 4)	**3**

<table>
<tr>
<td></td>
<td>
<table>
<tr><td></td><td></td><td>= 3</td><td></td></tr>
</table>

Final Output (dp Array)

The DP array will be:

0 1 2 3 1 2 3 4 2 1 2 3 3
</td>
</tr>
<tr>
<td colspan="2">Output:-
0 1 2 3 1 2 3 4 2 1 2 3 3</td>
</tr>
</table>