Paint Houses in C++

```cpp
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int main() {
    // Input array representing costs to paint each house with three colors
    vector<vector<int>> arr = {{1, 5, 7}, {5, 8, 4}, {3, 2, 9}, {1, 2, 4}};
    int n = arr.size(); // Number of houses

    // Initialize dp array
    vector<vector<long long>> dp(n, vector<long long>(3, 0));

    // Base case: First row initialization
    dp[0][0] = arr[0][0];
    dp[0][1] = arr[0][1];
    dp[0][2] = arr[0][2];

    // Fill dp array from second row onwards
    for (int i = 1; i < n; i++) {
        dp[i][0] = arr[i][0] + min(dp[i - 1][1], dp[i - 1][2]);
        dp[i][1] = arr[i][1] + min(dp[i - 1][0], dp[i - 1][2]);
        dp[i][2] = arr[i][2] + min(dp[i - 1][0], dp[i - 1][1]);
    }

    // Find the minimum cost to paint all houses
    long long ans = min(dp[n - 1][0], min(dp[n - 1][1], dp[n - 1][2]));

    // Output the minimum cost
    cout << ans << endl;

    return 0;
}
```

Input:
arr = {{1, 5, 7}, {5, 8, 4}, {3, 2, 9}, {1, 2, 4}}
n = 4 (number of houses)

Steps:

1. Initialization of dp Array:
   - dp[i][j] will store the minimum cost to paint up to the i-th house, ending with color j.
   - Base case: For the first house (i = 0), we directly take the cost from the input arr.

     dp[0][0] = arr[0][0] = 1
     dp[0][1] = arr[0][1] = 5
     dp[0][2] = arr[0][2] = 7

2. Filling the dp Array (Dynamic Programming):
   For each house i from 1 to n-1, calculate the cost for each color j by considering the minimum cost of the other two colors for the previous house.
   Formula:

   dp[i][0] = arr[i][0] + min(dp[i-1][1], dp[i-1][2])
   dp[i][1] = arr[i][1] + min(dp[i-1][0], dp[i-1][2])
   dp[i][2] = arr[i][2] + min(dp[i-1][0], dp[i-1][1])

3. Extract the Minimum Cost:
   After filling the dp array, the result is the minimum value from the last row (dp[n-1]).

Dry Run Details:

Step 1: Initialization (i = 0)

dp[0][0] = 1
dp[0][1] = 5
dp[0][2] = 7

Step 2: Fill dp for i = 1

```
dp[1][0] = arr[1][0] + min(dp[0][1], dp[0][2])
       = 5 + min(5, 7) = 5 + 5 = 10

dp[1][1] = arr[1][1] + min(dp[0][0], dp[0][2])
       = 8 + min(1, 7) = 8 + 1 = 9

dp[1][2] = arr[1][2] + min(dp[0][0], dp[0][1])
       = 4 + min(1, 5) = 4 + 1 = 5
```

State of dp:

dp[1] = {10, 9, 5}

Step 3: Fill dp for i = 2

```
dp[2][0] = arr[2][0] + min(dp[1][1], dp[1][2])
        = 3 + min(9, 5) = 3 + 5 = 8

dp[2][1] = arr[2][1] + min(dp[1][0], dp[1][2])
        = 2 + min(10, 5) = 2 + 5 = 7

dp[2][2] = arr[2][2] + min(dp[1][0], dp[1][1])
        = 9 + min(10, 9) = 9 + 9 = 18
```

State of dp:

dp[2] = {8, 7, 18}

Step 4: Fill dp for i = 3

```
dp[3][0] = arr[3][0] + min(dp[2][1], dp[2][2])
        = 1 + min(7, 18) = 1 + 7 = 8

dp[3][1] = arr[3][1] + min(dp[2][0], dp[2][2])
        = 2 + min(8, 18) = 2 + 8 = 10

dp[3][2] = arr[3][2] + min(dp[2][0], dp[2][1])
        = 4 + min(8, 7) = 4 + 7 = 11
```

State of dp:

dp[3] = {8, 10, 11}

Step 5: Extract the Result

The minimum cost to paint all houses is the minimum value in the last row of dp:

```
ans = min(dp[3][0], dp[3][1], dp[3][2])
    = min(8, 10, 11)
    = 8
```

Output:-

```
8
```