|  |  |
| --- | --- |
| **Range Sum in C++** | |
| #include <iostream>  #include <vector>  using namespace std;  vector<int> prefixSum;  void NumArray(vector<int>& nums) {  prefixSum.resize(nums.size());  prefixSum[0] = nums[0];  for (int i = 1; i < nums.size(); i++) {  prefixSum[i] = prefixSum[i - 1] + nums[i];  }  }  int sumRange(int i, int j) {  if (i == 0) {  return prefixSum[j];  }  return prefixSum[j] - prefixSum[i - 1];  }  int main() {  vector<int> arr = {1, 2, 3, 4};  NumArray(arr);  int res = sumRange(1, 2);  cout << res << endl; // Output should be 5  return 0;  } | Prefix Sum Table Construction in NumArray(arr) Let’s build prefixSum[] based on the input arr = {1, 2, 3, 4}.   | **Index i** | **nums[i]** | **prefixSum[i] = prefixSum[i - 1] + nums[i]** | **prefixSum array** | | --- | --- | --- | --- | | 0 | 1 | 1 | [1] | | 1 | 2 | 1 + 2 = 3 | [1, 3] | | 2 | 3 | 3 + 3 = 6 | [1, 3, 6] | | 3 | 4 | 6 + 4 = 10 | [1, 3, 6, 10] |   Final prefixSum = [1, 3, 6, 10] 📦 sumRange(1, 2) Execution We want to find sum from index 1 to 2 in original array (2 + 3 = 5).  Since i != 0, it uses:  prefixSum[2] - prefixSum[0] = 6 - 1 = 5   | **Expression** | **Value** | | --- | --- | | prefixSum[2] | 6 | | prefixSum[0] | 1 | | Result | 5 |   ✅ Output printed: **5** |
| 5 | |