|  |  |
| --- | --- |
| **Two Sum in C++** | |
| #include <iostream>  #include <unordered\_map>  #include <vector>  using namespace std;  vector<int> twoSum(vector<int>& nums, int target) {  unordered\_map<int, int> map; // Hash map to store number and its index  vector<int> result;  for (int i = 0; i < nums.size(); i++) {  int complement = target - nums[i];  if (map.find(complement) != map.end()) {  result.push\_back(map[complement]);  result.push\_back(i);  return result;  }  map[nums[i]] = i;  }  throw invalid\_argument("No two sum solution");  }  int main() {  vector<int> nums1 = {2, 7, 11, 15};  int target1 = 9;  vector<int> nums2 = {3, 2, 4};  int target2 = 6;  vector<int> result1 = twoSum(nums1, target1);  vector<int> result2 = twoSum(nums2, target2);  cout << "Output for nums1: [" << result1[0] << ", " << result1[1] << "]" << endl;  cout << "Output for nums2: [" << result2[0] << ", " << result2[1] << "]" << endl;  return 0;  } | **Test Case 1**  vector<int> nums1 = {2, 7, 11, 15};  int target1 = 9;   * We need to find two indices i, j such that nums1[i] + nums1[j] = 9.  | **Step** | **i** | **nums1[i]** | **Complement (target - nums1[i])** | **map (stored indices)** | **Match Found?** | | --- | --- | --- | --- | --- | --- | | 1 | 0 | 2 | 7 | {2:0} | ❌ No | | 2 | 1 | 7 | 2 | {2:0, 7:1} | ✅ Yes (2 found at index 0) |   ✅ **Output:** [0, 1] (because nums1[0] + nums1[1] = 2 + 7 = 9)  **Test Case 2**  vector<int> nums2 = {3, 2, 4};  int target2 = 6;   | **Step** | **i** | **nums2[i]** | **Complement (target - nums2[i])** | **map (stored indices)** | **Match Found?** | | --- | --- | --- | --- | --- | --- | | 1 | 0 | 3 | 3 | {3:0} | ❌ No | | 2 | 1 | 2 | 4 | {3:0, 2:1} | ❌ No | | 3 | 2 | 4 | 2 | {3:0, 2:1, 4:2} | ✅ Yes (2 found at index 1) |   ✅ **Output:** [1, 2] (because nums2[1] + nums2[2] = 2 + 4 = 6) |
| Output:- Output for nums1: [0, 1]  Output for nums2: [1, 2] | |