|  |  |
| --- | --- |
| **Isomorphic Strings in C++** | |
| #include <iostream>  #include <string>  #include <unordered\_map>  using namespace std;  bool iso(string s, string t) {  if (s.length() != t.length()) {  return false;  }  unordered\_map<char, char> map1; // Maps characters from s to t  unordered\_map<char, bool> map2; // Tracks characters used in t  for (int i = 0; i < s.length(); i++) {  char ch1 = s[i];  char ch2 = t[i];  if (map1.count(ch1) > 0) { // If ch1 is already mapped  if (map1[ch1] != ch2) { // Check if mapping is consistent  return false;  }  } else { // ch1 has not been mapped yet  if (map2.count(ch2) > 0) { // If ch2 is already mapped by another character in s  return false;  } else { // Create new mapping  map1[ch1] = ch2;  map2[ch2] = true;  }  }  }  return true;  }  int main() {  string s1 = "abc";  string s2 = "cad";  cout << boolalpha << iso(s1, s2) << endl; // Output: true  return 0;  } | **Step 1: Initialize Variables**   * **Input Strings**: s = "abc", t = "cad" * **Maps Used**:   + map1 → Stores mapping from s to t   + map2 → Tracks characters already mapped in t   **Step 2: Iterating Through s and t**   | **Index (i)** | **s[i]** | **t[i]** | **map1 (s → t)** | **map2 (used t characters)** | **Check for Conflict?** | **Result** | | --- | --- | --- | --- | --- | --- | --- | | 0 | 'a' | 'c' | { a → c } | { c → true } | No | Continue | | 1 | 'b' | 'a' | { a → c, b → a } | { c → true, a → true } | No | Continue | | 2 | 'c' | 'd' | { a → c, b → a, c → d } | { c → true, a → true, d → true } | No | Continue |   **Step 3: Return Result**   * Since no conflicts were found, return true.   **Final Output**  true |
| Output: true | |