|  |  |
| --- | --- |
| **IsSorted in C++** | |
| #include <iostream>  using namespace std;  bool isSortedEff(int arr[], int n) {  for (int i = 1; i < n; i++) {  if (arr[i] < arr[i - 1]) {  return false;  }  }  return true;  }  bool isSorted(int arr[], int n) {  for (int i = 0; i < n; i++) {  for (int j = i + 1; j < n; j++) {  if (arr[j] < arr[i]) {  return false;  }  }  }  return true;  }  int main() {  int arr1[] = {1, 2, 3, 4, 5, 6};  int arr2[] = {11, 2, 3, 4, 5, 6};  int n1 = sizeof(arr1) / sizeof(arr1[0]);  int n2 = sizeof(arr2) / sizeof(arr2[0]);  cout << boolalpha; // Print boolean values as true/false  cout << isSortedEff(arr1, n1) << endl;  cout << isSortedEff(arr2, n2) << endl;  cout << isSorted(arr1, n1) << endl;  cout << isSorted(arr2, n2) << endl;  return 0;  } | Check if an array is **sorted in non-decreasing order** (each element is ≤ the next). 🔍 Difference between isSortedEff and isSorted:  | **Function** | **Approach** | **Time Complexity** | | --- | --- | --- | | isSortedEff | Linear scan (compare adjacent) | **O(n)** | | isSorted | Brute force (nested loops) | **O(n²)** |  ✅ Dry Run with Sample ArraysArray 1: {1, 2, 3, 4, 5, 6} (Sorted) **isSortedEff(arr1, n1)**:   | **i** | **arr[i-1]** | **arr[i]** | **Comparison** | **Result** | | --- | --- | --- | --- | --- | | 1 | 1 | 2 | 2 ≥ 1 | ✅ | | 2 | 2 | 3 | 3 ≥ 2 | ✅ | | 3 | 3 | 4 | 4 ≥ 3 | ✅ | | 4 | 4 | 5 | 5 ≥ 4 | ✅ | | 5 | 5 | 6 | 6 ≥ 5 | ✅ | | → All passed → **Returns: true** |  |  |  |  |   **isSorted(arr1, n1)**: Checks every pair (i, j) where j > i:   * For every arr[i] <= arr[j] → all OK → **Returns: true**  Array 2: {11, 2, 3, 4, 5, 6} (Not sorted) **isSortedEff(arr2, n2)**:   | **i** | **arr[i-1]** | **arr[i]** | **Comparison** | **Result** | | --- | --- | --- | --- | --- | | 1 | 11 | 2 | 2 < 11 ❌ | 🔴 | | → Early exit → **Returns: false** |  |  |  |  |   **isSorted(arr2, n2)**:   * (0,1) → 2 < 11 → ❌ → **Returns: false**  🖨 Output: true  false  true  false |
| true  false  true  false | |