Algorithm Library | C++ Magicians STL Algorithm

Difficulty Level : Medium • Last Updated : 17 Oct, 2022

For all those who aspire to excel in competitive programming, only having a knowledge about containers of STL is of less use till one is not aware what all STL has to offer.
STL has an ocean of algorithms, for all < algorithm > library functions : Refer **here**.
Some of the most used algorithms on vectors and most useful one's in Competitive Programming are mentioned as follows :

Non-Manipulating Algorithms

1. **sort(first_iterator, last_iterator)** – To sort the given vector.
2. **sort(first_iterator, last_iterator, greater<int>())** – To sort the given container/vector in descending order
3. **reverse(first_iterator, last_iterator)** – To reverse a vector. (if ascending -> descending OR if descending -> ascending)
4. ***max_element (first_iterator, last_iterator)** – To find the maximum element of a vector.
5. ***min_element (first_iterator, last_iterator)** – To find the minimum element of a vector.
6. **accumulate(first_iterator, last_iterator, initial value of sum)** – Does the summation of vector elements

---

CPP

```
// A C++ program to demonstrate working of sort()
```

```
#include <iostream>
#include <vector>
```



```
int main()
{
    // Initializing vector with array values
    int arr[] = {10, 20, 5, 23 ,42 , 15};
    int n = sizeof(arr)/sizeof(arr[0]);
    vector<int> vect(arr, arr+n);

    cout << "Vector is: ";
    for (int i=0; i<n; i++)
        cout << vect[i] << " ";

    // Sorting the Vector in Ascending order
    sort(vect.begin(), vect.end());


    cout << "\nVector after sorting is: ";
    for (int i=0; i<n; i++)
       cout << vect[i] << " ";


      // Sorting the Vector in Descending order
      sort(vect.begin(),vect.end(), greater<int>());

    cout << "\nVector after sorting in Descending order is: ";
    for (int i=0; i<n; i++)
       cout << vect[i] << " ";


    // Reversing the Vector (descending to ascending , ascending to descending)
    reverse(vect.begin(), vect.end());

    cout << "\nVector after reversing is: ";
    for (int i=0; i<n; i++)
        cout << vect[i] << " ";

    cout << "\nMaximum element of vector is: ";
    cout << *max_element(vect.begin(), vect.end());

    cout << "\nMinimum element of vector is: ";
    cout << *min_element(vect.begin(), vect.end());

    // Starting the summation from 0
```

```
    return 0;
}
```


Output

```
Vector is: 10 20 5 23 42 15
Vector after sorting is: 5 10 15 20 23 42
Vector after sorting in Descending order is: 42 23 20 15 10 5
Vector after reversing is: 5 10 15 20 23 42
Maximum element of vector is: 42
Minimum element of vector is: 5
The summation of vector elements is: 115
```

6.count(first_iterator, last_iterator,x) – To count the occurrences of x in vector.

7. find(first_iterator, last_iterator, x) – Returns an iterator to the first occurrence of x in vector and points to last address of vector ((name_of_vector).end()) if element is not present in vector.

CPP

```
// C++ program to demonstrate working of count()
// and find()
#include <algorithm>
#include <iostream>
#include <vector>
using namespace std;

int main()
{
    // Initializing vector with array values
    int arr[] = {10, 20, 5, 23 ,42, 20, 15};
    int n = sizeof(arr)/sizeof(arr[0]);
    vector<int> vect(arr, arr+n);

    cout << "Occurrences of 20 in vector : ";

    // Counts the occurrences of 20 from 1st to
    // last element
```

```
    // element not present
    find(vect.begin(), vect.end(),5) != vect.end()?
```



```
    return 0;
}
```

Output

```
Occurrences of 20 in vector : 2
Element found
```

8. binary_search(first_iterator, last_iterator, x) – Tests whether x exists in sorted vector or not.

9. lower_bound(first_iterator, last_iterator, x) – returns an iterator pointing to the first element in the range [first,last) which has a value not less than 'x'.

10. upper_bound(first_iterator, last_iterator, x) – returns an iterator pointing to the first element in the range [first,last) which has a value greater than 'x'.

C++

```
// C++ program to demonstrate working of lower_bound()
// and upper_bound().
#include <algorithm>
```

```
int main()
```



```
    int arr[] = {5, 10, 15, 20, 20, 25, 42, 45};
    int n = sizeof(arr)/sizeof(arr[0]);
    vector<int> vect(arr, arr+n);

    // Sort the array to make sure that lower_bound()
    // and upper_bound() work.
    sort(vect.begin(), vect.end());

    // Returns the first occurrence of 20
    auto q = lower_bound(vect.begin(), vect.end(), 20);

    // Returns the last occurrence of 20
    auto p = upper_bound(vect.begin(), vect.end(), 20);

    cout << "The lower bound is at position: ";
    cout << q-vect.begin() << endl;

    cout << "The upper bound is at position: ";
    cout << p-vect.begin() << endl;

    return 0;
}
```

Output

```
The lower bound is at position: 3
The upper bound is at position: 5
```

Some Manipulating Algorithms

1. **arr.erase(position to be deleted)** – This erases selected element in vector and shifts and resizes the vector elements accordingly.
2. **arr.erase(unique(arr.begin(),arr.end()),arr.end())** – This erases the duplicate occurrences in sorted vector in a single line.

C++

```
// C++ program to demonstrate working
```

```
#include <bits/stdc++.h>
#include <iostream>
```



```
int main()
{
    // Initializing vector with array values
    int arr[] = { 5, 10, 15, 20, 20, 23, 42, 45 };
    int n = sizeof(arr) / sizeof(arr[0]);
    vector<int> vect(arr, arr + n);

    cout << "Given Vector is:\n";
    for (int i = 0; i < n; i++)
        cout << vect[i] << " ";

    vect.erase(find(vect.begin(),vect.end(),10));
    cout << "\nVector after erasing element:\n";
    for (int i = 0; i < vect.size(); i++)
        cout << vect[i] << " ";

    vect.erase(unique(vect.begin(), vect.end()),
               vect.end());
    cout << "\nVector after removing duplicates:\n";
    for (int i = 0; i < vect.size(); i++)
        cout << vect[i] << " ";

    return 0;
}
```

Output

```
Given Vector is:
5 10 15 20 20 23 42 45
Vector after erasing element:
5 15 20 20 23 42 45
Vector after removing duplicates:
5 15 20 23 42 45
```

3. next_permutation(first_iterator, last_iterator) – This modified the vector to its next permutation.

4. prev_permutation(first_iterator, last_iterator) – This modified the vector to its

CPP


```cpp
// of next_permutation()
// and prev_permutation()
#include <algorithm>
#include <iostream>
#include <vector>
using namespace std;

int main()
{
    // Initializing vector with array values
    int arr[] = {5, 10, 15, 20, 20, 23, 42, 45};
    int n = sizeof(arr)/sizeof(arr[0]);
    vector<int> vect(arr, arr+n);

    cout << "Given Vector is:\n";
    for (int i=0; i<n; i++)
        cout << vect[i] << " ";

    // modifies vector to its next permutation order
    next_permutation(vect.begin(), vect.end());
    cout << "\nVector after performing next permutation:\n";
    for (int i=0; i<n; i++)
        cout << vect[i] << " ";

    prev_permutation(vect.begin(), vect.end());
    cout << "\nVector after performing prev permutation:\n";
    for (int i=0; i<n; i++)
        cout << vect[i] << " ";

    return 0;
}
```

Output

```
Given Vector is:
5 10 15 20 20 23 42 45
Vector after performing next permutation:
5 10 15 20 20 23 45 42
Vector after performing prev permutation:
```

5. distance(first_iterator,desired_position) – It returns the distance of desired

CPP

```
// C++ program to demonstrate working of distance()
#include <algorithm>
#include <iostream>
#include <vector>
using namespace std;

int main()
{
    // Initializing vector with array values
    int arr[] = {5, 10, 15, 20, 20, 23, 42, 45};
    int n = sizeof(arr)/sizeof(arr[0]);
    vector<int> vect(arr, arr+n);

    // Return distance of first to maximum element
    cout << "Distance between first to max element: ";
    cout << distance(vect.begin(),
                     max_element(vect.begin(), vect.end()));
    return 0;
}
```

Output

```
Distance between first to max element: 7
```

More – STL Articles

This article is contributed by **Manjeet Singh.** If you like GeeksforGeeks and would like to contribute, you can also write an article and mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Previous Next

Related Articles

1. List in C++ Standard Template Library (STL)
2. Multimap in C++ Standard Template Library (STL)
3. library in C++ STL
4. <iterator> library in C++ STL
5. <numeric> library in C++ STL
6. <regex> library in C++ STL
7. Multiset in C++ Standard Template Library (STL)
8. Set in C++ Standard Template Library (STL)
9. The C++ Standard Template Library (STL)
10. Binary Search in C++ Standard Template Library (STL)

Vote for difficulty

Current difficulty : Medium

Easy | Normal | Medium | Hard | Expert

Improved By : shubh_89, devashish_, soumyalahiri, believer411, kunal01, pratikthorat19, sweetyty, saiudaykiranshinagam, triangleofcoding, _MSK_

Article Tags : cpp-algorithm-library, STL, Competitive Programming

Practice Tags : STL

Improve Article | Report Issue

GeeksforGeeks
A-143, 9th Floor, Sovereign Corporate Tower,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org