std::partition in C++ STL

C++ has a class in its STL algorithms library which allows us easy partition algorithms using certain inbuilt functions. Partition refers to act of dividing elements of containers depending upon a given condition.

Partition operations :

1. partition(beg, end, condition) :- This function is used to **partition the elements** on **basis of condition** mentioned in its arguments.

2. is_partitioned(beg, end, condition) :- This function returns boolean **true if container is partitioned** else returns false.

```
// C++ code to demonstrate the working of
// partition() and is_partitioned()
#include<iostream>
#include<algorithm> // for partition algorithm
#include<vector> // for vector
using namespace std;
int main()
{
    // Initializing vector
    vector<int> vect = { 2, 1, 5, 6, 8, 7 };

    // Checking if vector is partitioned
    // using is_partitioned()
    is_partitioned(vect.begin(), vect.end(), [](int x)
    {
        return x%2==0;

    })?

    cout << "Vector is partitioned":
    cout << "Vector is not partitioned";
    cout << endl;

    // partitioning vector using partition()
    partition(vect.begin(), vect.end(), [](int x)
    {
        return x%2==0;

    });

    // Checking if vector is partitioned
    // using is_partitioned()
    is_partitioned(vect.begin(), vect.end(), [](int x)
    {
        return x%2==0;

    })?

    cout << "Now, vector is partitioned after partition operation
    cout << "Vector is still not partitioned after partition oper
    cout << endl;

    // Displaying partitioned Vector
    cout << "The partitioned vector is : ";
    for (int &x : vect) cout << x << " ";

    return 0;

}
```

Output:

```
Vector is not partitioned
Now, vector is partitioned after partition operation
The partitioned vector is : 2 8 6 5 1 7
```

In the above code, partition function partitions the vector depending on whether an element is even or odd, even elements are partitioned from odd elements in no particular order.

3. stable_partition(beg, end, condition) :- This function is used to **partition the elements** on **basis of condition** mentioned in its arguments in **such a way that the relative order of the elements is preserved.**.

4. partition_point(beg, end, condition) :- This function **returns an iterator pointing to the partition point** of container i.e. the first element in the partitioned range [beg,end) for which condition is not true. The container should already be partitioned for this function to work.

```
// C++ code to demonstrate the working of
// stable_partition() and partition_point()
#include<iostream>
#include<algorithm> // for partition algorithm
#include<vector> // for vector
using namespace std;
int main()
{
    // Initializing vector
    vector<int> vect = { 2, 1, 5, 6, 8, 7 };

    // partitioning vector using stable_partition()
    // in sorted order
    stable_partition(vect.begin(), vect.end(), [](int x)
    {
        return x%2 == 0;
    });

    // Displaying partitioned Vector
    cout << "The partitioned vector is : ";
    for (int &x : vect) cout << x << " ";
    cout << endl;

    // Declaring iterator
    vector<int>::iterator it1;

    // using partition_point() to get ending position of partitio
    auto it = partition_point(vect.begin(), vect.end(), [](int x)
    {
        return x%2==0;
    });

    // Displaying partitioned Vector
    cout << "The vector elements returning true for condition are
    for ( it1= vect.begin(); it1!=it; it1++)
    cout << *it1 << " ";
    cout << endl;

    return 0;

}
```

Output:

```
The partitioned vector is : 2 6 8 1 5 7
The vector elements returning true for condition are : 2 6 8
```

In the above code, even and odd elements are partitioned and in the increasing order (sorted)

5. partition_copy(beg, end, beg1, beg2, condition) :- This function **copies the partitioned elements** in the differenet containers mentioned in its arguments. It takes 5 arguments. **Beginning and ending position of container, beginning position of new container where elements have to be copied (elements returning true for condition), beginning position of new container where other elements have to be copied (elements returning false for condition) and the condition**. **Resizing** new containers **is necessary** for this function.

```
// C++ code to demonstrate the working of
// partition_copy()
#include<iostream>
#include<algorithm> // for partition algorithm
#include<vector> // for vector
using namespace std;
int main()
{
    // Initializing vector
    vector<int> vect = { 2, 1, 5, 6, 8, 7 };

    // Declaring vector1
    vector<int> vect1;

    // Declaring vector1
    vector<int> vect2;

    // Resizing vectors to suitable size using count_if() and res
    int n = count_if (vect.begin(), vect.end(), [](int x)
    {
        return x%2==0;

    } );
    vect1.resize(n);
    vect2.resize(vect.size()-n);

    // Using partition_copy() to copy partitions
    partition_copy(vect.begin(), vect.end(), vect1.begin(),
                                     vect2.begin(), [](int x)
    {
        return x%2==0;
    });


    // Displaying partitioned Vector
    cout << "The elements that return true for condition are : ";
    for (int &x : vect1)
            cout << x << " ";
    cout << endl;

    // Displaying partitioned Vector
    cout << "The elements that return false for condition are : "
    for (int &x : vect2)
            cout << x << " ";
    cout << endl;

    return 0;
}
```

Output:

```
The elements that return true for condition are : 2 6 8
The elements that return false for condition are : 1 5 7
```

This article is contributed by **Manjeet Singh** .If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@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.

Recommended Posts:

fill in C++ STL

Conditional or Ternary Operator (?:) in C/C++

forward_list insert_after() function in C++ STL

Count of distinct remainders when N is divided by all the numbers from the range [1, N]

C++ | asm declaration

Pointers and References in C++

Strings in C++ and How to Create them?

Enum Classes in C++ and Their Advantage over Enum DataType

Deque vs Vector in C++ STL

C++ Programming Basics

Introduction to C++ Programming Language

ios manipulators noshowpos() function in C++

C program to Insert an element in an Array

ios rdstate() function in C++ with Examples

ios operator !() function in C++ with Examples

Improved By : MaheshJagtap

Article Tags : C C++ cpp-algorithm-library STL

Practice Tags : STL C CPP

👍
4

1.8

□ To-do □ Done

Based on **11** vote(s)

Feedback/ Suggest Improvement   Add Notes   Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

GeeksforGeeks
A computer science portal for geeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org

COMPANY
About Us
Careers
Privacy Policy
Contact Us

LEARN
Algorithms
Data Structures
Languages
CS Subjects
Video Tutorials

PRACTICE
Courses
Company-wise
Topic-wise
How to begin?

CONTRIBUTE
Write an Article
Write Interview Experience
Internships
Videos