Set in C++ Standard Template Library (STL)

Sets are a type of associative containers in which each element has to be unique, because the value of the element identifies it. The value of the element cannot be modified once it is added to the set, though it is possible to remove and add the modified value of that element.

Some basic functions associated with Set:

- begin() – Returns an iterator to the first element in the set.
- end() – Returns an iterator to the theoretical element that follows last element in the set.
- size() – Returns the number of elements in the set.
- max_size() – Returns the maximum number of elements that the set can hold.
- empty() – Returns whether the set is empty.

```
#include <iostream>
#include <set>
#include <iterator>

using namespace std;

int main()
{
    // empty set container
    set <int, greater <int> > gquiz1;

    // insert elements in random order
    gquiz1.insert(40);
    gquiz1.insert(30);
    gquiz1.insert(60);
    gquiz1.insert(20);
    gquiz1.insert(50);
    gquiz1.insert(50); // only one 50 will be added to the set
    gquiz1.insert(10);

    // printing set gquiz1
    set <int, greater <int> > :: iterator itr;
    cout << "\nThe set gquiz1 is : ";
    for (itr = gquiz1.begin(); itr != gquiz1.end(); ++itr)
    {
        cout << '\t' << *itr;
    }
    cout << endl;

    // assigning the elements from gquiz1 to gquiz2
    set <int> gquiz2(gquiz1.begin(), gquiz1.end());

    // print all elements of the set gquiz2
    cout << "\nThe set gquiz2 after assign from gquiz1 is : ";
    for (itr = gquiz2.begin(); itr != gquiz2.end(); ++itr)
    {
        cout << '\t' << *itr;
    }
    cout << endl;

    // remove all elements up to 30 in gquiz2
    cout << "\ngquiz2 after removal of elements less than 30 : ";
    gquiz2.erase(gquiz2.begin(), gquiz2.find(30));
    for (itr = gquiz2.begin(); itr != gquiz2.end(); ++itr)
    {
        cout << '\t' << *itr;
    }

    // remove element with value 50 in gquiz2
    int num;
    num = gquiz2.erase (50);
    cout << "\ngquiz2.erase(50) : ";
    cout << num << " removed \t" ;
    for (itr = gquiz2.begin(); itr != gquiz2.end(); ++itr)
    {
        cout << '\t' << *itr;
    }

    cout << endl;

    //lower bound and upper bound for set gquiz1
    cout << "gquiz1.lower_bound(40) : "
         << *gquiz1.lower_bound(40) << endl;
    cout << "gquiz1.upper_bound(40) : "
         << *gquiz1.upper_bound(40) << endl;

    //lower bound and upper bound for set gquiz2
    cout << "gquiz2.lower_bound(40) : "
         << *gquiz2.lower_bound(40) << endl;
    cout << "gquiz2.upper_bound(40) : "
         << *gquiz2.upper_bound(40) << endl;

    return 0;

}
```

The output of the above program is :

```
The set gquiz1 is :      60    50    40    30    20    10

The set gquiz2 after assign from gquiz1 is : 10      20    30    40    50    60
```

```
gquiz2 after removal of elements less than 30 : 30    40    50    60
gquiz2.erase(50) : 1 removed     30    40    60
gquiz1.lower_bound(40) : 40
gquiz1.upper_bound(40) : 30
gquiz2.lower_bound(40) : 40
gquiz2.upper_bound(40) : 60
```

Methods of set:

- begin() – Returns an iterator to the first element in the set.
- end() – Returns an iterator to the theoretical element that follows last element in the set.
- rbegin()– Returns a reverse iterator pointing to the last element in the container.
- rend()– Returns a reverse iterator pointing to the theoretical element right before the first element in the set container.
- crbegin()– Returns a constant iterator pointing to the last element in the container.
- crend() – Returns a constant iterator pointing to the position just before the first element in the container.
- cbegin()– Returns a constant iterator pointing to the first element in the container.
- cend() – Returns a constant iterator pointing to the position past the last element in the container.
- size() – Returns the number of elements in the set.
- max_size() – Returns the maximum number of elements that the set can hold.
- empty() – Returns whether the set is empty.
- insert(const g) – Adds a new element 'g' to the set.
- iterator insert (iterator position, const g) – Adds a new element 'g' at the position pointed by iterator.
- erase(iterator position) – Removes the element at the position pointed by the iterator.
- erase(const g)– Removes the value 'g' from the set.
- clear() – Removes all the elements from the set.
- key_comp() / value_comp() – Returns the object that determines how the elements in the set are ordered ('<' by default).
- find(const g) – Returns an iterator to the element 'g' in the set if found, else returns the iterator to end.
- count(const g) – Returns 1 or 0 based on the element 'g' is present in the set or not.
- lower_bound(const g) – Returns an iterator to the first element that is equivalent to 'g' or definitely will not go before the element 'g' in the set.
- upper_bound(const g) – Returns an iterator to the first element that is equivalent to 'g' or definitely will go after the element 'g' in the set.
- equal_range()– The function returns an iterator of pairs. (key_comp). The pair refers to the range that includes all the elements in the container which have a key equivalent to k.
- emplace()– This function is used to insert a new element into the set container, only if the element to be inserted is unique and does not already exists in the set.
- emplace_hint()– Returns an iterator pointing to the position where the insertion is done. If the element passed in the parameter already exists, then it returns an iterator pointing to the position where the existing element is.
- swap()– This function is used to exchange the contents of two sets but the sets must be of same type, although sizes may differ.
- operator= – The '=' is an operator in C++ STL which copies (or moves) a set to another set and set::operator= is the corresponding operator function.
- get_allocator()– Returns the copy of the allocator object associated with the set.

Recent Articles on set

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

Recommended Posts:

The C++ Standard Template Library (STL)

Map in C++ Standard Template Library (STL)

Deque in C++ Standard Template Library (STL)

Sort in C++ Standard Template Library (STL)

Multimap in C++ Standard Template Library (STL)

Multiset in C++ Standard Template Library (STL)

Pair in C++ Standard Template Library (STL)

List in C++ Standard Template Library (STL)

Queue in Standard Template Library (STL)

Unordered Sets in C++ Standard Template Library

Binary Search in C++ Standard Template Library (STL)

Priority Queue in C++ Standard Template Library (STL)

Program to implement standard error of mean

Variance and standard-deviation of a matrix

Given N and Standard Deviation, find N elements

Improved By : BabisSarantoglou

Article Tags : C++ Mathematical cpp-containers-library cpp-set STL

Practice Tags : Mathematical STL CPP

28

2.1

Based on **34** vote(s)

To-do Done

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