Multiset in C++ Standard Template Library (STL)

Multisets are a type of associative containers similar to set, with an exception that multiple elements can have same values.

Some Basic Functions associated with multiset:

begin() – Returns an iterator to the first element in the multiset

end() – Returns an iterator to the theoretical element that follows last element in the multiset

size() – Returns the number of elements in the multiset

max_size() – Returns the maximum number of elements that the multiset can hold

empty() – Returns whether the multiset is empty

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

using namespace std;

int main()
{
    // empty multiset container
    multiset <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); // 50 will be added again to the multiset
gquiz1.insert(10);

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

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

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

// remove all elements up to element with value 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 all elements 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 multiset 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 multiset 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 multi-
set gquiz1 is :  60     50    50    40    30    20    10

The multiset gquiz2 after as-
sign from gquiz1 is :  10     20    30    40    50    50    60

gquiz2 after removal of ele-
ments less than 30 :  30    40    50    50    60
gquiz2.erase(50) : 2 removed         30    40    60
gquiz1.lower_bound(40) : 40
gquiz1.upper_bound(40) : 30
gquiz2.lower_bound(40) : 40
gquiz2.upper_bound(40) : 60
```

List of functions of Multiset:

- begin() – Returns an iterator to the first element in the multiset.
- end() – Returns an iterator to the theoretical element that follows last element in the multiset.
- size() – Returns the number of elements in the multiset.
- max_size()– Returns the maximum number of elements that the multiset can hold.
- empty() – Returns whether the multiset is empty.
- pair insert(const g) – Adds a new element ‘g’ to the multiset.
- 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 multiset.
- clear()– Removes all the elements from the multiset.
- key_comp() / value_comp()– Returns the object that determines how the elements in the multiset are ordered (‘<' by default).

- find(const g)– Returns an iterator to the element ‘g’ in the multiset if found, else returns the iterator to end.
- count(const g)– Returns the number of matches to element ‘g’ in the multiset.
- 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 multiset.
- 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 multiset.
- multiset::swap()– This function is used to exchange the contents of two multisets but the sets must be of same type, although sizes may differ.
- multiset::operator=– This operator is used to assign new contents to the container by replacing the existing contents.
- multiset::emplace()– This function is used to insert a new element into the multiset container.
- multiset equal_range()– Returns an iterator of pairs. The pair refers to the range that includes all the elements in the container which have a key equivalent to k.
- multiset::emplace_hint() – Inserts a new element in the multiset.
- multiset::rbegin()– Returns a reverse iterator pointing to the last element in the multiset container.
- multiset::rend()– Returns a reverse iterator pointing to the theoretical element right before the first element in the multiset container.
- multiset::cbegin()– Returns a constant iterator pointing to the first element in the container.
- multiset::cend()– Returns a constant iterator pointing to the position past the last element in the container.
- multiset::crbegin()– Returns a constant reverse iterator pointing to the last element in the container.
- multiset::crend()– Returns a constant reverse iterator pointing to the position just before the first element in the container.
- multiset::get_allocator()– Returns a copy of the allocator object associated with the multiset.

Recent articles on Multiset

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

Recommended Posts:

Set in C++ Standard Template Library (STL)

The C++ Standard Template Library (STL)

Map in C++ Standard Template Library (STL)

Deque in C++ Standard Template Library (STL)

List in C++ Standard Template Library (STL)

Pair in C++ Standard Template Library (STL)

Queue in Standard Template Library (STL)

Multimap in C++ Standard Template Library (STL)

Sort in C++ Standard Template Library (STL)

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

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

Unordered Sets in C++ Standard Template Library

multiset::emplace() in C++ STL

multiset erase() in C++ STL

multiset max_size() in C++ STL

Article Tags : C++ cpp-containers-library cpp-multiset STL

Practice Tags : STL CPP

3

1.7

☐ To-do ☐ Done

Based on **10** 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.