- using DFS
- vector :: assign() in C++ STL
- Pre-increment and Post-increment in C/C++
- static_cast in C++ | Type Casting operators
- map count() function in C++ STL
- Sum of array Elements without using loops and recursion
- set insert() function in C++ STL
- How to return multiple values from a function in C or C++?
- std::any Class in C++

vector rbegin() and rend() function in C++ STL

Applications of Pointers in C/C++

map erase() function in C++ STL

Memory leak in C++ and How to avoid it?

STL Priority Queue for Structure or Class

Loader in C/C++

vector emplace() function in C++ STL

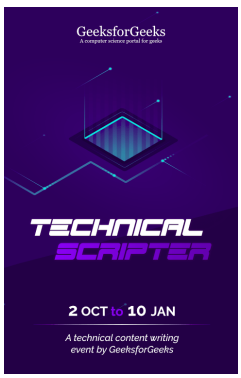
list erase() function in C++ STL

Types of Operator Overloading in C++

Check if X
can give
change to
every person
in the Queue

set
lower_bound()
function in
C++ STL

multimap
insert() in
C++ STL



Map in C++ Standard Template Library (STL)

Maps are associative containers that store elements in a mapped fashion. Each element has a key value and a mapped value. No two mapped values can have same key values.

Some basic functions associated with Map:

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

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

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

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

empty() – Returns whether the map is empty

pair insert(keyvalue, mapvalue) – Adds a new element to the map

erase(iterator position) – Removes the element at the position pointed by the iterator

erase(const g)– Removes the key value ‘g’ from the map

clear() – Removes all the elements from the map

```
#include <iostream>
#include <iterator>
#include <map>

using namespace std;

int main()
{

    // empty map container
    map<int, int> gquiz1;

    // insert elements in random order
    gquiz1.insert(pair<int, int>(1, 40));
    gquiz1.insert(pair<int, int>(2, 30));
    gquiz1.insert(pair<int, int>(3, 60));
    gquiz1.insert(pair<int, int>(4, 20));
    gquiz1.insert(pair<int, int>(5, 50));
    gquiz1.insert(pair<int, int>(6, 50));
    gquiz1.insert(pair<int, int>(7, 10));

    // printing map gquiz1
    map<int, int>::iterator itr;
    cout << "\nThe map gquiz1 is : \n";
    cout << "\tKEY\tELEMENT\n";
    for (itr = gquiz1.begin(); itr != gquiz1.end(); ++itr) {
        cout << '\t' << itr->first
             << '\t' << itr->second << '\n';
    }
    cout << endl;

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

    // print all elements of the map gquiz2
    cout << "\nThe map gquiz2 after"
         << " assign from gquiz1 is : \n";
    cout << "\tKEY\tELEMENT\n";
    for (itr = gquiz2.begin(); itr != gquiz2.end(); ++itr) {
        cout << '\t' << itr->first
             << '\t' << itr->second << '\n';
    }
    cout << endl;

    // remove all elements up to
    // element with key=3 in gquiz2
    cout << "\ngquiz2 after removal of"
            " elements less than key=3 : \n";
    cout << "\tKEY\tELEMENT\n";
    gquiz2.erase(gquiz2.begin(), gquiz2.find(3));
    for (itr = gquiz2.begin(); itr != gquiz2.end(); ++itr)
        cout << '\t' << itr->first
             << '\t' << itr->second << '\n';
}
```

Output:

```
The map gquiz1 is :
        KEY     ELEMENT
        1       40
        2       30
        3       60
        4       20
        5       50
        6       50
        7       10


The map gquiz2 after assign from gquiz1 is :
        KEY     ELEMENT
        1       40
        2       30
        3       60
        4       20
        5       50
        6       50
        7       10


gquiz2 after removal of elements less than key=3 :
        KEY     ELEMENT
        3       60
        4       20
        5       50
        6       50
        7       10

gquiz2.erase(4) : 1 removed
        KEY     ELEMENT
        3       60
        5       50
        6       50
        7       10
```

```
gquiz1.lower_bound(5) :         KEY = 5         ELEMENT = 50
gquiz1.upper_bound(5) :         KEY = 6         ELEMENT = 50
```

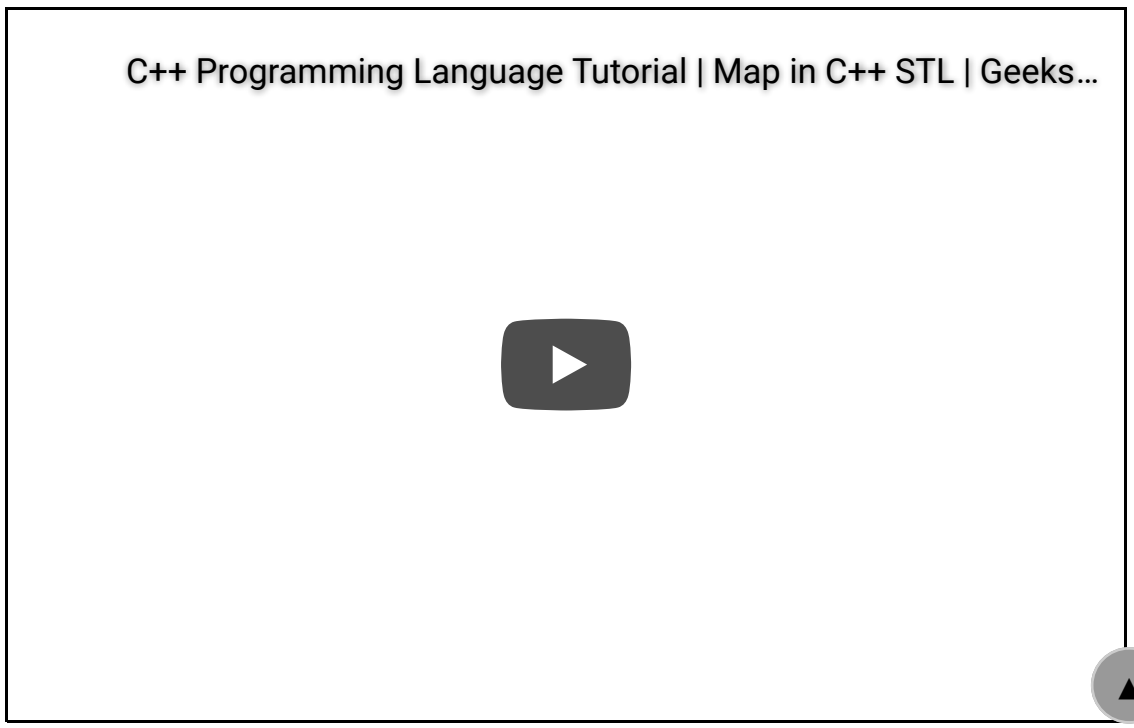
List of all functions of Map:

- map insert() in C++ STL– Insert elements with a particular key in the map container. .
- map count() function in C++ STL– Returns the number of matches to element with key value 'g' in the map.
- map equal_range() in C++ STL– Returns an iterator of pairs. The pair refers to the bounds of a range that includes all the elements in the container which have a key equivalent to k.
- map erase() function in C++ STL– Used to erase element from the container.
- map rend() function in C++ STL– Returns a reverse iterator pointing to the theoretical element right before the first key-value pair in the map(which is considered its reverse end).
- map rbegin() function in C++ STL– Returns a reverse iterator which points to the last element of the map.
- map find() function in C++ STL– Returns an iterator to the element with key value 'g' in the map if found, else returns the iterator to end.
- map crbegin() and crend() function in C++ STL– **crbegin()** returns a constant reverse iterator referring to the last element in the map container. **crend()** returns a constant reverse iterator pointing to the theoretical element before the first element in the map.
- map cbegin() and cend() function in C++ STL– **cbegin()** returns a constant iterator referring to the first element in the map container. **cend()** returns a constant iterator pointing to the theoretical element that follows last element in the multimap.
- map emplace() in C++ STL– Inserts the key and its element in the map container.
- map max_size() in C++ STL– Returns the maximum number of elements a map container can hold.
- map upper_bound() function in C++ STL– Returns an iterator to the first element that is equivalent to mapped value with key value 'g' or definitely will go after the element with key value 'g' in the map
- map operator= in C++ STL– Assigns contents of a container to a different container, replacing its current content.
- map lower_bound() function in C++ STL– Returns an iterator to the first element that is equivalent to mapped value with key value 'g' or definitely will not go before

the element with key value ‘g’ in the map.

- map emplace_hint() function in C++ STL– Inserts the key and its element in the map container with a given hint.
- map value_comp() in C++ STL– Returns the object that determines how the elements in the map are ordered (‘<' by default).
- map key_comp() function in C++ STL– Returns the object that determines how the elements in the map are ordered (‘<' by default).
- map::size() in C++ STL– Returns the number of elements in the map.
- map::empty() in C++ STL– Returns whether the map is empty.
- map::begin() and end() in C++ STL– **begin()** returns an iterator to the first element in the map. **end()** returns an iterator to the theoretical element that follows last element in the map
- map::operator[] in C++ STL– This operator is used to reference the element present at position given inside the operator.
- map::clear() in C++ STL– Removes all the elements from the map.
- map::at() and map::swap() in C++ STL– **at()** function is used to return the reference to the element associated with the key k. **swap()** function is used to exchange the contents of two maps but the maps must be of same type, although sizes may differ.

Recent articles on Map



Please write comments if you find anything incorrect, or you want to share more infor-

mation about the topic discussed above

Recommended Posts:

The C++ Standard Template Library (STL)

Set in C++ Standard Template Library (STL)

Multimap in C++ Standard Template Library (STL)

List in C++ Standard Template Library (STL)

Sort in C++ Standard Template Library (STL)

Multiset in C++ Standard Template Library (STL)

Pair in C++ Standard Template Library (STL)

Deque in C++ Standard Template Library (STL)

Queue in Standard Template Library (STL)

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

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

<algorithms> library in C++ STL

snprintf() in C library

<strings> library in C++ STL

<numeric> library in C++ STL

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

Practice Tags : STL CPP

3

2.8

To-do Done

Based on **42** vote(s)

Feedback 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 Share this post!