Casting operators

Sum of array Elements without using loops and recursion

“static const” vs “#define” vs “enum”

How to find Segmentation Error in C & C++ ? (Using GDB)

Why strcpy and strncpy are not safe to use?

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

Loader in C/C++

Type Conversion in C++

GDB (Step by Step Introduction)

Modulus

- function in C++ STL
- Some useful C++ tricks for beginners in Competitive Programming
- Interesting Facts about C++
- Passing Reference to a Pointer in C++
- Difference between Call by Value and Call by Reference
- Difference between fundamental data types and derived data types
- Types of Operator Overloading in C++
- list insert() in C++ STL
- std::any

Class in C++

C++:
Methods of
code
shortening in
competitive
programming

Split a string
in equal
parts such
that all parts
are
palindromes

Amadeus
Labs R & D |
On Campus
(freshers) |
Full
time+Internship

#pragma
Directive in
C/C++

Measure
execution
time with
high
precision in
C/C++

Multimap in C++ Standard Template Library (STL)

Multimap is similar to map with an addition that multiple elements can have same keys. Rather than each element being unique, the key value and mapped value pair has to be unique in this case.

Some Basic Functions associated with multimap:

- **begin()** – Returns an iterator to the first element in the multimap
- **end()** – Returns an iterator to the theoretical element that follows last element in the multimap
- **size()** – Returns the number of elements in the multimap
- **max_size()** – Returns the maximum number of elements that the multimap can hold
- **empty()** – Returns whether the multimap is empty
- **pair<int,int> insert(keyvalue,multimapvalue)** – Adds a new element to the multimap

C++ implementation to illustrate above functions

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

using namespace std;

int main()
{
    multimap <int, int> gquiz1;        // empty multimap contair

    // 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> (6, 10));

    // printing multimap gquiz1
    multimap <int, int> :: iterator itr;
    cout << "\nThe multimap 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
    multimap <int, int> gquiz2(gquiz1.begin(),gquiz1.end());

    // print all elements of the multimap gquiz2
    cout << "\nThe multimap gquiz2 after assign from gquiz1 is :
    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 value 30 in gquiz2
    cout << "\ngquiz2 after removal of elements less than key=3
    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';
    }

    // remove all elements with key = 4
```

Output:

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


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


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

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

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

List of Functions of Multimap:

- multimap::operator= in C++ STL– It is used to assign new contents to the container by replacing the existing contents.
- multimap::crbegin() and multimap::crend() in C++ STL– **crbegin()** returns a constant reverse iterator referring to the last element in the multimap container. **crend()** returns a constant reverse iterator pointing to the theoretical element before the first element in the multimap.
- multimap::emplace_hint() in C++ STL– Inserts the key and its element in the multimap container with a given hint.
- multimap clear() function in C++ STL– Removes all the elements from the multimap.
- multimap empty() function in C++ STL– Returns whether the multimap is empty.
- multimap maxsize() in C++ STL– Returns the maximum number of elements a multimap container can hold.
- multimap value_comp() function in C++ STL– Returns the object that determines how the elements in the multimap are ordered (‘<‘ by default)
- multimap rend in C++ STL– Returns a reverse iterator pointing to the theoretical element preceding to the first element of the multimap container.
- multimap::cbegin() and multimap::cend() in C++ STL– **cbegin()** returns a constant iterator referring to the first element in the multimap container. **cend()** returns a constant iterator pointing to the theoretical element that follows last element in the multimap.
- multimap::swap() in C++ STL– Swap the contents of one multimap with another multimap of same type and size.
- multimap rbegin in C++ STL– Returns an iterator pointing to the last element of the container.
- multimap size() function in C++ STL– Returns the number of elements in the multimap container.
- multimap::emplace() in C++ STL– Inserts the key and its element in the multimap container.
- multimap::begin() and multimap::end() in C++ STL– **begin()** returns an iterator referring to the first element in the multimap container. **end()** returns an iterator to the theoretical element that follows last element in the multimap.
- multimap upper_bound() function in C++ STL– Returns an iterator to the first element that is equivalent to multimapped value with key value ‘g’ or definitely will go after the element with key value ‘g’ in the multimap.
- multimap::count() in C++ STL– Returns the number of matches to element with key value ‘g’ in the multimap.

- multimap::erase() in C++ STL– Removes the key value from the multimap.
- multimap::find() in C++ STL– Returns an iterator to the element with key value 'g' in the multimap if found, else returns the iterator to end.
- multimap 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.
- multimap insert() in C++ STL– Used to insert elements in the multimap container.
- multimap lower_bound() function in C++ STL– Returns an iterator to the first element that is equivalent to multimapped value with key value 'g' or definitely will not go before the element with key value 'g' in the multimap.
- multimap key_comp() in C++ STL– Returns the object that determines how the elements in the multimap are ordered ('<' by default).

Recent articles on Multimap

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)

List in C++ Standard Template Library (STL)

Sort in C++ Standard Template Library (STL)

Multiset in C++ Standard Template Library (STL)

Queue in Standard Template Library (STL)

Deque in C++ Standard Template Library (STL)

Pair in C++ Standard Template Library (STL)

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

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

multimap::crbegin() and multimap::crend() in C++ STL

multimap::cbegin() and multimap::cend() in C++ STL

multimap::begin() and multimap::end() in C++ STL

multimap insert() in C++ STL

Article Tags : C++ cpp-containers-library cpp-multimap cpp-multimap-functions STL

Practice Tags : STL CPP

Be the First to upvote.

To-do Done

2.2

Based on **10** 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!