Different ways to delete elements in std::map (erase() and clear())

This article deals with the deletion part of Maps.

1. **Using erase()** : erase() is used to **erase** the pair in map mentioned in argument, either its position, its value or a range of number.
   - **erase(key)** : Erases the **key-value pair** using key mentioned in its argument. reorders the map after deletion. It returns the **number of entries** deleted. If non-existing keys is deleted, 0 is returned.
     Time complexity : **log(n)** (n is size of map)
   - **erase(iter)** : Erases the pair at the position **pointed by the iterator** mentioned in its argument.
     Time complexity : **log(n)** (n is size of map)
   - **erase(strt_iter, end_iter)** : Erases the **range of pairs** starting from “strt_iter” to the “end_iter”.
     Time complexity : **O(k)** (k is size of map)

```
// C++ code to demonstrate the working of erase()

#include<iostream>
#include<map> // for map operations
using namespace std;

int main()
{
    // declaring map
    // of char and int
    map< char, int > mp;

    // declaring iterators
    map<char, int>::iterator it ;
    map<char, int>::iterator it1;
    map<char, int>::iterator it2;

    // inserting values
    mp['a']=5;
    mp['b']=10;
    mp['c']=15;
    mp['d']=20;
    mp['e']=30;

    // printing initial map elements
    cout << "The initial map elements are : \n";

    for (it1 = mp.begin(); it1!=mp.end(); ++it1)
        cout << it1->first << "->" << it1->second << endl;

    it = mp.begin();

    cout << endl;

    // erasing element using iterator
    // erases 2nd element
    // 'b'
    ++it;
    mp.erase(it);
```



```
    for (it1 = mp.begin(); it1!=mp.end(); ++it1)
```

```
        cout << it1->first << "->" << it1->second << endl;

    cout << endl;

    // erasing element using value
    int c = mp.erase('c');

    // printing map elements after deletion
    cout << "The map elements after 2nd deletion are : \n";

    for (it1 = mp.begin(); it1!=mp.end(); ++it1)
        cout << it1->first << "->" << it1->second << endl;

    cout << "The number of elements deleted in 2nd deletion are : ";
    cout << c << endl;

    cout << endl;

    // erasing element using value
    // key not present
    int d = mp.erase('w');

    // printing map elements after deletion
    cout << "The map elements after 3rd deletion are : \n";

    for (it1 = mp.begin(); it1!=mp.end(); ++it1)
        cout << it1->first << "->" << it1->second << endl;

    cout << "The number of elements deleted in 3rd deletion are : ";
    cout << d << endl;

    cout << endl;


    ++it;
    ++it;

    // erasing element using range iterator
    // deletes "d" and "e" keys
    mp.erase(it, mp.end());

    // printing map elements 4th deletion
    cout << "The map elements after 4th deletion are : \n";

    for (it1 = mp.begin(); it1!=mp.end(); ++it1)
        cout << it1->first << "->" << it1->second << endl;

    cout << endl;

}
```

Run on IDE

Output:

```
The initial map elements are :
a->5
b->10
c->15
d->20
e->30

The map elements after 1st deletion are :
a->5
c->15
d->20
e->30

The map elements after 2nd deletion are :
a->5
d->20
e->30
```



```
a->5
```

```
d->20
e->30
The number of elements deleted in 3rd deletion are : 0

The map elements after 4th deletion are :
a->5
```

2. **Using clear()** : This function **clears** all the elements present in the map. After this function is called, the size of map becomes 0.

```
// C++ code to demonstrate the working of clear()

#include<iostream>
#include<map> // for map operations
using namespace std;

int main()
{
    // declaring map
    // of char and int
    map< char, int > mp;

    // declaring iterator
    map<char, int>::iterator it ;

    // inserting values
     mp['a']=5;
     mp['b']=10;
     mp['c']=15;
     mp['d']=20;
     mp['e']=30;

    // printing initial map elements
    cout << "The initial map elements are : \n";
    for (it1 = mp.begin(); it1!=mp.end();  ++it1)
    cout << it1->first << "->" << it1->second << endl;

    // using clear() to erase all elements in map
    mp.clear();

    // printing map elements after deletion
    cout << "The map elements after clearing all elements are : \n";
    for (it1 = mp.begin(); it1!=mp.end();  ++it1)
    cout << it1->first << "->" << it1->second << endl;

}
```

Run on IDE

Output:

```
The initial map elements are :
a->5
b->10
c->15
d->20
e->30
The map elements after clearing all elements are :
```

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.

Practice Tags : STL CPP

Article Tags : C++ cpp-map STL

Login to Improve thisArticle

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

Recommended Posts:

How to transform Vector into String?
Map in C++ Standard Template Library (STL)
Multimap in C++ Standard Template Library (STL)
How to store Data Triplet in a Vector in C++?
Pointer to an Array | Array Pointer
list max_size() function in C++ STL
asinh() function in C++ STL
atanh() function in C++ STL
multimap equal_range() in C++ STL
Check if X can give change to every person in the Queue

(Login to Rate)

0 Average Difficulty : **0/5.0**
No votes yet.

Basic Easy Medium Hard Expert

Add to TODO List
Mark as DONE

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

Load Comments Share this post!