# Difference between set, multiset, unordered_set, unordered_multiset

1. **Set**

(i) Stores the values in sorted order.

(ii) Stores only unique values.

(iii) Elements can only be inserted or deleted but cannot be modified.

(iv) We can erase more than 1 element by giving start iterator and end iterator position.

(v) Traversal using iterators.

(vi) Sets are implemented as Binary Search Tree.

```
// CPP program to demonstrate insert and
// delete in set
#include <bits/stdc++.h>
using namespace std;
int main()
{
    // set declare
    set<int> s;

    // Elements added to set
    s.insert(12);
    s.insert(10);
    s.insert(2);
    s.insert(10); // duplicate added
    s.insert(90);
    s.insert(85);
    s.insert(45);

    // Iterator declared to traverse
    // set elements
    set<int>::iterator it, it1, it2;
    cout << "Set elements after sort and "
            "removing duplicates:\n";
    for (it = s.begin(); it != s.end(); it++)
        cout << *it << ' ';
    cout << '\n';

    it1 = s.find(10);
    it2 = s.find(90);

    // elements from 10 to elements before
    // 90 erased
    s.erase(it1, it2);
    cout << "Set Elements after erase:\n";
    for (it = s.begin(); it != s.end(); it++)
        cout << *it << ' ';

    return 0;
}
```

```
Set elements after sort and removing duplicates:
2 10 12 45 85 90
Set Elements after erase:
2 90
```

2. **Multiset**

(i) Stores elements in sorted order.

(ii) It allows storage of multiple elements.

(iii) We can erase more than 1 element by giving start iterator and end iterator.

Note:- All other properties similar to set.

```
// CPP program to demonstrate insert and
// delete in set
#include <bits/stdc++.h>
using namespace std;
int main()
{
    // multiset declare
    multiset<int> s;

    // Elements added to set
    s.insert(12);
    s.insert(10);
    s.insert(2);
    s.insert(10); // duplicate added
    s.insert(90);
    s.insert(85);
    s.insert(45);

    // Iterator declared to traverse
    // set elements
    multiset<int>::iterator it, it1, it2;
    cout << "Multiset elements after sort\n";
    for (it = s.begin(); it != s.end(); it++)
        cout << *it << ' ';
    cout << '\n';

    it1 = s.find(10);
    it2 = s.find(90);

    // elements from 10 to elements before 90
    // erased
    s.erase(it1, it2);

    cout << "Multiset Elements after erase:\n";
    for (it = s.begin(); it != s.end(); it++)
        cout << *it << ' ';

    return 0;
}
```



```
OUTPUT:
Multiset elements after sort
2 10 10 12 45 85 90
Multiset Elements after erase:
2 90
```

3. **Unordered_set**

(i) Elements can be stored in any order. ( no sorted order )

(iv) We can erase only the element for which iterator position is given.

Note:- All other properties similar to set.

```
// CPP program to demonstrate insert and
// delete in unordered_set
#include <bits/stdc++.h>
using namespace std;
int main()
{
    // unordered_set declare
    unordered_set<int> s;

    // Elements added to set
    s.insert(12);
    s.insert(10);
    s.insert(2);
    s.insert(10); // duplicate added
    s.insert(90);
    s.insert(85);
    s.insert(45);
    s.insert(12);
    s.insert(70);

    // Iterator declared to traverse
    // set elements
    unordered_set<int>::iterator it, it1;
    cout << "Unordered_set elements after sort:\n";
    for (it = s.begin(); it != s.end(); it++)
        cout << *it << ' ';
    cout << '\n';

    it1 = s.find(10);

    // element 10 erased
    s.erase(it1);
    cout << "Unoredered_set Elements after erase:\n";
    for (it = s.begin(); it != s.end(); it++)
         cout << *it << ' ';

    return 0;
}
```

```
OUTPUT:
Unordered_set elements after sort:
70 85 45 12 10 2 90
Unoredered_set Elements after erase:
70 85 45 12 2 90
```

4. **Unordered_multiset**

(i) Elements can be stored in any order.

(ii) Duplicate elements can be stored.

(iii) Hash-table used to store elements.

(iv) We can erase only the element for which iterator position is given.

Note:- All other properties similar to set.

```
// CPP program to demonstrate insert and
// delete in unordered_multiset
#include <bits/stdc++.h>
using namespace std;
```

```
    // Elements added to set
    s.insert(12);
    s.insert(10);
    s.insert(2);
    s.insert(10); // duplicate added
    s.insert(90);
    s.insert(85);
    s.insert(45);

    // Iterator declared to traverse
    // set elements
    unordered_multiset<int>::iterator it, it1;
    cout << "Unordered-Multiset elements after sort:\n";
    for (it = s.begin(); it != s.end(); it++)
        cout << *it << ' ';
    cout << '\n';

    it1 = s.find(10);

    // element 10 trained
    s.erase(it1);

    cout << "Unordered-Multiset Elements after "
            "erase:\n";
    for (it = s.begin(); it != s.end(); it++)
        cout << *it << ' ';

    return 0;
}
```

```
OUTPUT:
Unordered-Multiset elements after sort:
85 45 12 90 2 10 10
Unordered-Multiset Elements after erase:
85 45 12 90 2 10
```

**Conclusion :**

In simple words, **set** is a container that stores ***sorted and unique*** elements. If ***unordered*** is added means elements are ***not sorted***. If ***multiset*** is added means ***duplicate elements*** storage is allowed.

This article is contributed by **SHAURYA UPPAL**. 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.

## Recommended Posts:

multiset::emplace() in C++ STL

multiset::swap() in C++ STL

multiset max_size() in C++ STL

multiset erase() in C++ STL

multiset::operator= in C++ STL

multiset value_comp() method in C++ STL

multiset upper_bound() in C++ STL with Examples

multiset get_allocator() function in C++ STL

multiset key_comp() function in C++ STL

multiset count() function in C++ STL

multiset emplace_hint() function in C++ STL

multiset insert() function in C++ STL

multiset empty() function in C++ STL

**Improved By :** Rajeev_Verma

**Article Tags :** C++ Difference Between cpp-unordered_set STL

**Practice Tags :** STL CPP


3

To-do Done

2

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.

Load Comments Share this post!

A computer science portal for geeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org