

## C++

[Information](#)  
[Tutorials](#)  
[Reference](#)  
[Articles](#)  
[Forum](#)

## Reference

## C library:

## Containers:

[<array>](#)[<deque>](#)[<forward\\_list>](#)[<list>](#)[<map>](#)[<queue>](#)[<set>](#)[<stack>](#)[<unordered\\_map>](#)[<unordered\\_set>](#)[<vector>](#)

## Input/Output:

## Multi-threading:

## Other:

## &lt;set&gt;

[multiset](#)  
[set](#)

## multiset

[multiset::multiset](#)[multiset::~~multiset](#)

## member functions:

[multiset::begin](#)[multiset::cbegin](#)[multiset::cend](#)[multiset::clear](#)[multiset::count](#)[multiset::crbegin](#)[multiset::crend](#)[multiset::emplace](#)[multiset::emplace\\_hint](#)[multiset::empty](#)[multiset::end](#)[multiset::equal\\_range](#)[multiset::erase](#)[multiset::find](#)[multiset::get\\_allocator](#)[multiset::insert](#)[multiset::key\\_comp](#)[multiset::lower\\_bound](#)[multiset::max\\_size](#)[multiset::operator=](#)[multiset::rbegin](#)[multiset::rend](#)[multiset::size](#)[multiset::swap](#)[multiset::upper\\_bound](#)[multiset::value\\_comp](#)

## non-member overloads:

[relational operators \(multiset\)](#)[swap \(multiset\)](#)

public member function

**std::multiset::count**

&lt;set&gt;

`size_type count (const value_type& val) const;`**Count elements with a specific key**Searches the container for elements equivalent to *val* and returns the number of matches.Two elements of a [multiset](#) are considered equivalent if the container's [comparison](#) object returns false reflexively (i.e., no matter the order in which the elements are passed as arguments).**Parameters***val*

Value to search for.

Member type `value_type` is the type of the elements in the container, defined in [multiset](#) as an alias of its first template parameter (`T`).**Return value**The number of elements in the container that are equivalent to *val*.Member type `size_type` is an unsigned integral type.**Example**

```
1 // multiset::count
2 #include <iostream>
3 #include <set>
4
5 int main ()
6 {
7     int myints[]={10,73,12,22,73,73,12};
8     std::multiset<int> mymultiset (myints,myints+7);
9
10    std::cout << "73 appears " << mymultiset.count(73) << " times in mymultiset.\n";
11
12    return 0;
13 }
```

Output:

`73 appears 3 times in mymultiset.`**Complexity**Logarithmic in [size](#) and linear in the number of matches.**Iterator validity**

No changes.

**Data races**

The container is accessed.

Concurrently accessing the elements of a [multiset](#) is safe.**Exception safety****Strong guarantee:** if an exception is thrown, there are no changes in the container.**See also**

<a href="#">multiset::find</a>	Get iterator to element ( <a href="#">public member function</a> )
<a href="#">multiset::equal_range</a>	Get range of equal elements ( <a href="#">public member function</a> )
<a href="#">multiset::size</a>	Return container size ( <a href="#">public member function</a> )
<a href="#">multiset::lower_bound</a>	Return iterator to lower bound ( <a href="#">public member function</a> )
<a href="#">multiset::upper_bound</a>	Return iterator to upper bound ( <a href="#">public member function</a> )