Multi Map

Multimaps are part of the C++ STL (Standard Template Library). Multimaps are the associative containers like map that stores sorted key-value pair, but unlike maps which store only unique keys, multimap can have duplicate keys. By-default it uses < operator to compare the keys.

Syntax

```
template < class Key,                                // multimap::key_type
        class T,                                     // multimap::mapped_type
        class Compare = less<Key>,                   // multimap::key_compare
        class Alloc = allocator<pair<const Key,T> >  // multimap::allocator_type
        > class multimap;
```

Parameter

key: The key data type to be stored in the multimap.

type: The data type of value to be stored in the multimap.

compare: A comparison class that takes two arguments of the same type bool and returns a value. This argument is optional and the binary predicate less<"key"> is the default value.

alloc: Type of the allocator object. This argument is optional and the default value is allocator.

Creating a multimap

Multimaps can easily be created using the following statement:

typedef pair<const Key, T> value_type;

The above form will use to create a multimap with key of type Key_type and value of type value_type. One important thing is that key of a multimap and corresponding values are always inserted as a pair, you cannot insert only key or just a value in a multimap.

Member Functions

Constructor/Destructor

Functions	Description
constructor	Construct multimap
destructor	Multimap destructor
operator=	Copy elements of the multimap to another multimap.

Iterators

Functions	Description
begin	Returns an iterator pointing to the first element in the multimap.
cbegin	Returns a const_iterator pointing to the first element in the multimap.
end	Returns an iterator pointing to the past-end.
cend	Returns a constant iterator pointing to the past-end.
rbegin	Returns a reverse iterator pointing to the end.
rend	Returns a reverse iterator pointing to the beginning.
crbegin	Returns a constant reverse iterator pointing to the end.
crend	Returns a constant reverse iterator pointing to the beginning.

Capacity

Functions	Description
empty	Return true if multimap is empty.
size	Returns the number of elements in the multimap.
max_size	Returns the maximum size of the multimap.

Modifiers

Functions	Description
insert	Insert element in the multimap.
erase	Erase elements from the multimap.
swap	Exchange the content of the multimap.
clear	Delete all the elements of the multimap.
emplace	Construct and insert the new elements into the multimap.
emplace_hint	Construct and insert new elements into the multimap by hint.

Observers

Functions	Description
key_comp	Return a copy of key comparison object.
value_comp	Return a copy of value comparison object.

Operations

Functions	Description
find	Search for an element with given key.
count	Gets the number of elements matching with given key.
lower_bound	Returns an iterator to lower bound.
upper_bound	Returns an iterator to upper bound.
equal_range()	Returns the range of elements matches with given key.

Allocator

Functions	Description
get_allocator	Returns an allocator object that is used to construct the multimap.

Non-Member Overloaded Functions

Functions	Description
operator==	Checks whether the two multimaps are equal or not.
operator!=	Checks whether the two multimaps are equal or not.
operator<	Checks whether the first multimap is less than other or not.
operator<=	Checks whether the first multimap is less than or equal to other or not.
operator>	Checks whether the first multimap is greater than other or not.
operator>=	Checks whether the first multimap is greater than equal to other or not.
swap()	Exchanges the element of two multimaps.

Example:

```
#include <iostream>
#include <map>
#include <string>

using namespace std;

int main()
{
   multimap<string, string> m = {
         {"India","New Delhi"},
         {"India", "Hyderabad"},
         {"United Kingdom", "London"},
         {"United States", "Washington D.C"}
   };
```

```
    cout << "Size of map m: " << m.size() <<endl;
    cout << "Elements in m: " << endl;

    for (multimap<string, string>::iterator it = m.begin(); it != m.end(); ++it)
    {
      cout << "  [" << (*it).first << ", " << (*it).second << "]" << endl;
    }

    return 0;
}
```