**Map in C++ Standard Template Library (STL)**

Maps are associative containers that store elements in a mapped fashion. Each element has a key value and a mapped.

key must be unique.

**Some basic functions associated with Map:**  
[begin()](https://www.geeksforgeeks.org/mapbegin-end-c-stl/) – Returns an iterator to the first element in the map  
[end()](https://www.geeksforgeeks.org/mapbegin-end-c-stl/) – Returns an iterator to the last element in the map  
[size()](https://www.geeksforgeeks.org/mapsize-c-stl/) – Returns the number of elements in the map  
[max\_size()](https://www.geeksforgeeks.org/map-max_size-in-c-stl/) – Returns the maximum number of elements that the map can hold  
[empty()](https://www.geeksforgeeks.org/mapempty-c-stl/) – Returns whether the map is empty  
[pair insert(keyvalue, mapvalue)](https://www.geeksforgeeks.org/map-insert-in-c-stl/) – Adds a new element to the map  
[erase(iterator position)](https://www.geeksforgeeks.org/map-erase-function-in-c-stl/) – Removes the element at the position pointed by the iterator  
[clear()](https://www.geeksforgeeks.org/mapclear-c-stl/) – Removes all the elements from the map

#include <iostream>

#include <iterator>

#include <map>

using namespace std;

int main()

{

// empty map container

**map< int, int >** **mp**;

// insert elements in random order

**mp**.**insert(** **pair< int, int >(** 1, 40 **)** **)**;

**mp**.**insert(** **pair< int, int >(** 2, 30 **)** **)**;

**mp**.**insert(** **pair< int, int >(** 3, 60 **)** **)**;

**mp**.**insert(** **pair< int, int >(** 4, 20 **)** **)**;

**mp**.**insert(** **pair< int, int >(** 5, 50 **)** **)**;

**mp**.**insert(** **pair< int, int >(** 6, 500 **)** **)**;

**mp**.**insert(** **pair< int, int >(** 7, 10 **)** **)**;

// printing map mp

**map< int, int > ::** **iterator** **itr** ;

cout << " The map mp is :";

cout << " KEY ELEMENT :";

for (**itr** = **mp**. **begin()**; **itr** != **mp**. **end();** **++itr**)

{

cout << **itr -> first**

<< **itr -> second** << '\n';

}

// assigning the elements from mp to mp1

**map< int, int >** **mp1** **(** **mp**. **begin()**, **mp**. **end() )**;

// print all elements of the map mp1

cout << "The map mp1 after assign from mp is : ";

cout << "KEY ELEMENT:";

for (**itr** = **mp1**. **begin()**; **itr** != **mp1**. **end();** **++itr**)

{

cout << **itr -> first**

<< **itr -> second** << '\n';

}

// remove all elements with key=3 in mp1

cout << "mp1 after removal of elements less than key=3 : ";

cout << " KEY ELEMENT:";

**mp1**.**erase( mp1**. **begin()** , **mp1**.**find(3) )**;

for (**itr** = **mp1**. **begin()**; **itr** != **mp1**. **end();** **++itr**)

{

cout << **itr -> first**

<< **itr -> second** << '\n';

}

// remove all elements with key = 4

int num;

num = **mp1**.**erase(4);**

cout << "mp1.erase(4) : ";

cout << num << " removed:";

cout << " KEY ELEMENT:";

for (**itr** = **mp1**. **begin()**; **itr** != **mp1**. **end();** **++itr**)

{

cout << **itr -> first**

<< **itr -> second** << '\n';

}

// lower bound and upper bound for map mp key = 5

cout << "mp.lower\_bound(5) : KEY = "

<<**mp.lower\_bound(5)->first** << endl;

     cout << " ELEMENT = "

         << **mp.lower\_bound(5)->second** << endl;

     cout << "mp.upper\_bound(5) : KEY = "

    << **mp.upper\_bound(5)->first** <<endl;

     cout << " ELEMENT = "

         << **mp.upper\_bound(5)->second** << endl;

return 0;

}

**Output:**

The map mp is :

KEY ELEMENT

1 40

2 30

3 60

4 20

5 50

6 50

7 10

The map mp1 after assign from mp is :

KEY ELEMENT

1 40

2 30

3 60

4 20

5 50

6 50

7 10

Mp1 after removal of elements less than key=3 :

KEY ELEMENT

3 60

4 20

5 50

6 50

7 10

Mp1.erase(4) : 1 removed

KEY ELEMENT

3 60

5 50

6 50

7 10

mp.lower\_bound(5) : KEY = 5 ELEMENT = 50

mp.upper\_bound(5) : KEY = 6 ELEMENT = 50