**Set ..** Artful

Functions associated with Set:  
begin() – Returns an iterator to the first element in the set  
end() – Returns an iterator to the theoretical element that follows last element in the set  
size() – Returns the number of elements in the set  
max\_size() – Returns the maximum number of elements that the set can hold  
empty() – Returns whether the set is empty  
pair <iterator, bool> insert(const g) – Adds a new element ‘g’ to the set  
iterator insert (iterator position, const g) – Adds a new element ‘g’ at the position pointed by iterator  
erase(iterator position) – Removes the element at the position pointed by the iterator  
erase(const g) - Removes the value ‘g’ from the set  
clear() – Removes all the elements from the set  
key\_comp() / value\_comp() – Returns the object that determines how the elements in the set are ordered (‘<‘ by default)  
find(const g) – Returns an iterator to the element ‘g’ in the set if found, else returns the iterator to end  
count(const g) – Returns 1 or 0 based on the element ‘g’ is present in the set or not.  
lower\_bound(const g) – Returns an iterator to the first element that is equivalent to ‘g’ or definitely will not go before the element ‘g’ in the set  
upper\_bound(const g) – Returns an iterator to the first element that is equivalent to ‘g’ or definitely will go after the element ‘g’ in the set.

Swap() -- first.swap(second);

struct classcomp

{

bool operator() (const int& lhs, const int& rhs) const

{

return lhs<rhs;

}

};

int main()

{

set<int> first; // empty set of ints

int myints[]= {10,20,30,40,50};

set<int> second (myints,myints+5); // range

set<int> third (second); // a copy of second

set<int> fourth (second.begin(), second.end()); // iterator .

set<int,classcomp> fifth; // class as Compare.

set <int, greater <int> > gquiz1;

gquiz1.insert(40);

gquiz1.insert(30);

gquiz1.insert(60);

gquiz1.insert(20);

gquiz1.insert(50);

gquiz1.insert(50);

gquiz1.insert(10);

set <int, greater <int> > :: iterator itr;

cout << "\nThe set gquiz1 is : ";

for (itr = gquiz1.begin(); itr != gquiz1.end(); ++itr)

{

cout <<" "<< \*itr;

}

cout << endl;

set <int> gquiz2(gquiz1.begin(), gquiz1.end());// assigning the elements from gquiz1 to gquiz2

cout << "\nThe set gquiz2 after assign from gquiz1 is : ";// print all elements of the set gquiz2

for (itr = gquiz2.begin(); itr != gquiz2.end(); ++itr)

{

cout << " " << \*itr;

}

cout << endl;

cout << "\ngquiz2 after removal of elements less than 30 : ";// remove all elements up to 30 in gquiz2

gquiz2.erase(gquiz2.begin(), gquiz2.find(30));

for (itr = gquiz2.begin(); itr != gquiz2.end(); ++itr)

cout << " " << \*itr;

int num;

num = gquiz2.erase (50); // remove element with value 50 in gquiz2

cout << "\ngquiz2.erase(50) : ";

cout << num << " removed \t" ;

for (itr = gquiz2.begin(); itr != gquiz2.end(); ++itr)

{

cout << " " << \*itr;

}

cout << endl;

//lower bound and upper bound for set gquiz1

cout << "gquiz1.lower\_bound(40) : "

<< \*gquiz1.lower\_bound(40) << endl;

cout << "gquiz1.upper\_bound(40) : "

<< \*gquiz1.upper\_bound(40) << endl;

//lower bound and upper bound for set gquiz2

cout << "gquiz2.lower\_bound(40) : "

<< \*gquiz2.lower\_bound(40) << endl;

cout << "gquiz2.upper\_bound(40) : "

<< \*gquiz2.upper\_bound(40) << endl;

return 0;

}

Output

The set gquiz1 is : 60 50 40 30 20 10

The set gquiz2 after assign from gquiz1 is : 10 20 30 40 50 60

gquiz2 after removal of elements less than 30 : 30 40 50 60

gquiz2.erase(50) : 1 removed 30 40 60

gquiz1.lower\_bound(40) : 40

gquiz1.upper\_bound(40) : 30

gquiz2.lower\_bound(40) : 40

gquiz2.upper\_bound(40) : 60