Set

Sets are part of the C++ STL (Standard Template Library). Sets are the associative containers that stores sorted key, in which each key is unique and it can be inserted or deleted but cannot be altered.

Syntax: template < class T, // set::key_type/value_type

class Compare = less<T>, // set::key_compare/value_compare

class Alloc = allocator<T> // set::allocator_type

> class set;

Parameter

T: Type of element stored in the container set.

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<T>, is the default value.

Alloc: Type of the allocator object which is used to define the storage allocation model.

Member Functions

Constructor/Destructor

Functions	Description
(constructor)	Construct set
(destructor)	Set destructor
operator=	Copy elements of the set to another set.

Iterators

Functions	Description
Begin	Returns an iterator pointing to the first element in the set.
cbegin	Returns a const iterator pointing to the first element in the set.
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	Returns true if set is empty.
Size	Returns the number of elements in the set.
max_size	Returns the maximum size of the set.

Modifiers

Functions	Description
insert	Insert element in the set.
Erase	Erase elements from the set.
Swap	Exchange the content of the set.
Clear	Delete all the elements of the set.
emplace	Construct and insert the new elements into the set.
emplace_hint	Construct and insert new elements into the set 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 set.

Non-Member Overloaded Functions

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

Example: C++ program to demonstrate various functions of STL

```
#include <iostream>
#include <iterator>
#include <set>
using namespace std;

int main()
{
        // empty set container
        set<int, greater<int> > s1;

        // insert elements in random order
        s1.insert(40);
        s1.insert(30);
        s1.insert(60);
```

```
s1.insert(20);
s1.insert(50);

// only one 50 will be added to the set
s1.insert(50);
s1.insert(10);

// printing set s1
set<int, greater<int> >::iterator itr;
cout << "\nThe set s1 is : \n";
for (itr = s1.begin(); itr != s1.end(); itr++) {
        cout << *itr << " ";
}
cout << endl;

// assigning the elements from s1 to s2
set<int> s2(s1.begin(), s1.end());

// print all elements of the set s2
cout << "\nThe set s2 after assign from s1 is : \n";
for (itr = s2.begin(); itr != s2.end(); itr++) {
        cout << *itr << " ";
}
cout << endl;

// remove all elements up to 30 in s2
cout << "\ns2 after removal of elements less than 30 "
                ":\n";
s2.erase(s2.begin(), s2.find(30));
for (itr = s2.begin(); itr != s2.end(); itr++) {
        cout << *itr << " ";
}

// remove element with value 50 in s2
int num;
num = s2.erase(50);
cout << "\ns2.erase(50) : ";
```

```
	cout << num << " removed\n";
	for (itr = s2.begin(); itr != s2.end(); itr++) {
		cout << *itr << " ";
	}

	cout << endl;

	// lower bound and upper bound for set s1
	cout << "s1.lower_bound(40) : "
		<< *s1.lower_bound(40) << endl;
	cout << "s1.upper_bound(40) : "
		<< *s1.upper_bound(40) << endl;

	// lower bound and upper bound for set s2
	cout << "s2.lower_bound(40) : "
		<< *s2.lower_bound(40) << endl;
	cout << "s2.upper_bound(40) : "
		<< *s2.upper_bound(40) << endl;

	return 0;
}
```