**Set**

**What:**

std::set is a associative container.

**Why:**

1.It does not allow duplicate elements (only unique elements are allowded). It can contain elements of any specific type.

2. Set are stored in balanced binary tree.

3. Set by default uses the operator < for comparing two elements but if user passes external sorting criteria then it uses that instead of the <.

4. std::set will keep inserted element in sorted order based on the assigned sorting criteria.

Here,

**SetName.size()**

This function returns the number of the entries available in the set.

Iteration through the set is done with the help of the begin() and end().

**To search data in set:**

find() method is used to traverse the iterator and using iterator we can check with if condition whether a data is present or not.

Iterator it = setName.find(data);

if(it != setName.end() )

//statement

else

//statement

**Why std::set::find member is preffered over the std::find ?**

It knows the internal data structure is balanced search tree and designed to operate on it, therefore it takes much less time.

**How std::set verifies whether an insertion has already happened or not?**

By default the std::set uses the operator <

Internally balanced binary search tree is used to insert the data and new elements are used to compare with the existing elements. So new elements of existing value are not inserted.

If < operator is used to compare the two values how come == operation is performed,

Thus if a > b is checkde and b > a is checked if both failed then its obviously its equal.

**How to remove a element in set?**

setName.erase(Data)

**How to use set with user defined class?**

Refer 04\_setUserDefined.cpp program

**To insert to set?**

SetName.insert(Data);

**To delete/erase data in set?**

SetName.erase(Data);

**To iterate a set?**

Forward iteration

std::set< type> :: iterator it;

it = setName.begin()

if(it != setName.end(){

std::cout << \*it << std::endl;

}

Reverse iteration

std::set< type> :: reverse\_iterator it;

it = setName.rbegin()

if(it != setName.rend(){

std::cout << \*it << std::endl;

}

NOTE:

1. We cannot modify a data in set.

2. If modification is needed then the data need to be deleted from the set and modified value can be added back to the set.