DATE : __/__/____
PAGE : _____

Dequeue

→ Deque or Doubly ended queue are sequence containers which can be used as queue as well as stack. STL, Dequeue has some additional features which are as follows :-

(i) Perform Random Access (ii) Perform arbitary insertation in constant time.

Syntax ⇒ dequeue < Type of Elements > deque name;

Traversal ⇒ for (auto x : dq)

cout << x << " ";

Set

→ Set is a associative container which stores only unique and sorted elements it. It work as Red Black Tree / Self Balancing trees. It does not duplicate values.

Syntax :- Set < int > S; ⊛ Elements in Increasing order

Set < int, greater <int> > S; ⊛ Element in Decreasing order

Traversal :- for (int x : S)

cout << x << " ";

Good Write

DATE: __/__/____
PAGE: _____

(*) Some function Perform on Set

1. insert() ⇒ S.insert(5);
2. erase() ⇒ S.erase(x);
3. find() ⇒ if(S.find(u) != S.end());
4. empty ⇒ S.empty();

(**) Lower and Upper Bound function()

Lower Bound	Upper Bound
① auto it = S.lower_bound(x);	① auto it = S.upper_bound(x);
Case I → If x is Present	Case I → If x is Present
→ It is will return the x.	→ It will return just greater element
Case II → If x is Not Present	Case II → If x is Not Present
→ It will return just greater element.	→ It will return just greater element
Case III → If x is greater than greatest	Case III → If x is greater than greatest.
→ It will return S.end();	→ It will return S.end();

(**) Application of Set

- Used as Self-Balancing Tree
- Sorted Stream Array
- Doubly Ended Priority Queue

Good Write

DATE: __/__/____
PAGE: _____

Multi-Set

→ All the features are retained by the Set except it allows the duplicate values it. It's internally working as R-B Tree

Syntax :- multiset <int> m;

Traversal :- for(auto x : m)
cout << x << " ";

* Some function like erase, count, lower bound, upper bound behaves slightly different from the Set.

1. erase() ⇒ It is erases all the occurences of the element.
2. count() ⇒ It will count all occurence as if they repeated as well.
3. lower_bound() ⇒ It will return the first occurence if multiple allows
4. upper_bound() ⇒ It will return greatest element of first occurance.

Good Write

DATE: __/__/____
PAGE: _____

#1 MAP

→ It is used to store key-value pair in ordered fashion.

The order of elements in the map is in increasing order of key values.

Syntax → map<Type key, Type value> map_name;

#2 Multi-MAP

→ It is similar container like map in addition to that multi-map can have multiple key-value pair.

* It is also implemented as Red-Black Trees, hence the basic operation like search, insert, delete works in O(LogN) time.

Syntax ⇒ multiMap<int, int> mp;

Traversal ⇒ for(auto x : mp)

cout << x.first << x.second << endl;

** [] operator is used to access & insert the element in map, But multiMap does not allow use of member access operator.

* count() in MAP vs MultiMap

MAP

→ return either 1 or 0 for key exist or not.

MultiMap

→ return the no. of ouccurences of key in multimap passed it as parameter.

Good Write

DATE: __/__/____
PAGE: _____

Unordered Set in STL (VVI).

A unordered Set is implemented using hash table where keys are hashed into indices of a hash table so that the insertion is always randomized. All the operation on the unordered Set take constant time O(1).

(*) Set is ordered sequence of unique key whereas unordered Set is a Set in which key can be stored in any order.

(**) We can traverse unordered Set just like iterators, but the elements will be printed in any random order every time.

Unordered map in STL

It is used to store elements formed by a combination of key-value pair. Internally unordered map is implemented using hash table.

Syntax ⇒ unordered_map < String, int > umap.

(*) The operations such as begin(), end(), Size(), empty() works in O(1) time in worst case.

Good Write