# STL (Standard Template Library)

- Array
- Vector (Dynamic Array)
- list (Doubly Link List)
- forward_list (Single Link list)
- Set
- multiset
- Map
- Multimap

} Ordered / Unordered

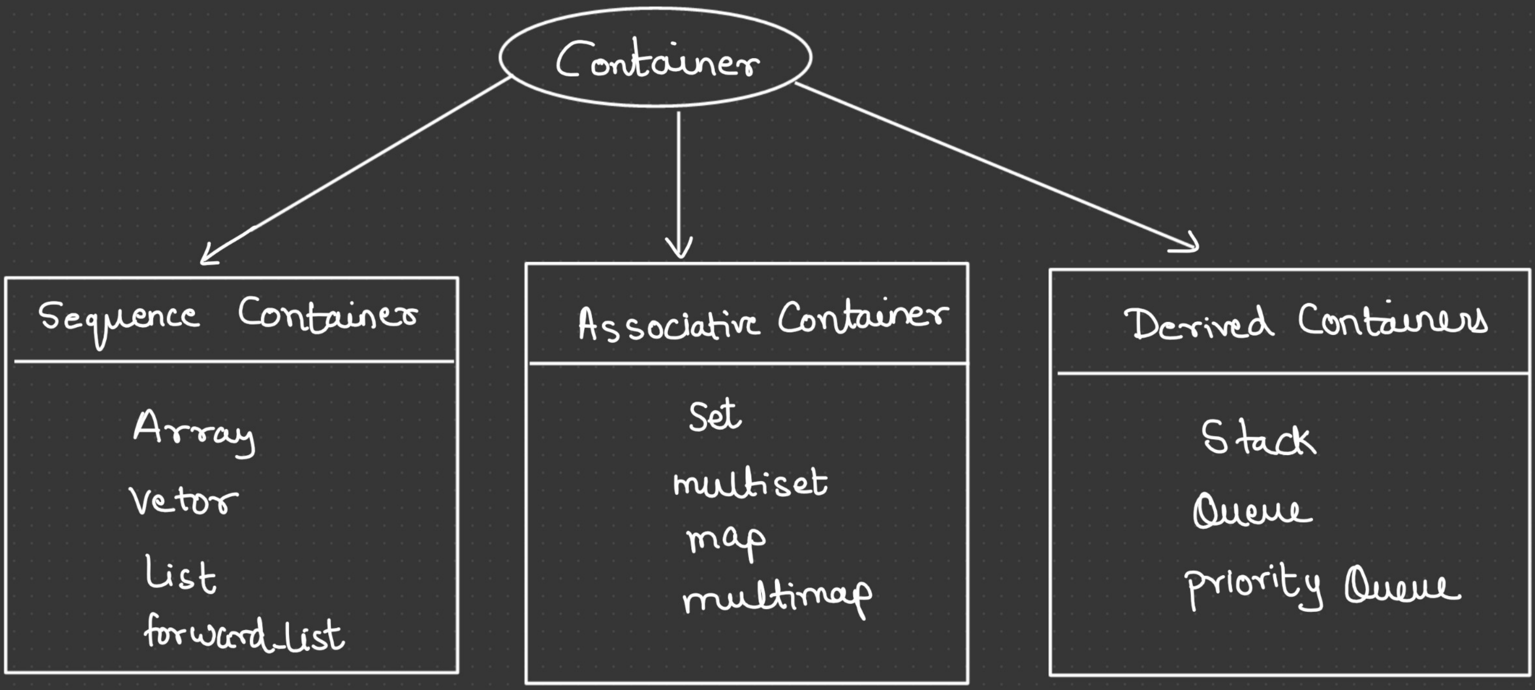
- Stack
- Queue
- Priority Queue.

---

## Components of STL :-

- Containers
- Iterators
- Algorithms.

## Classification of Containers

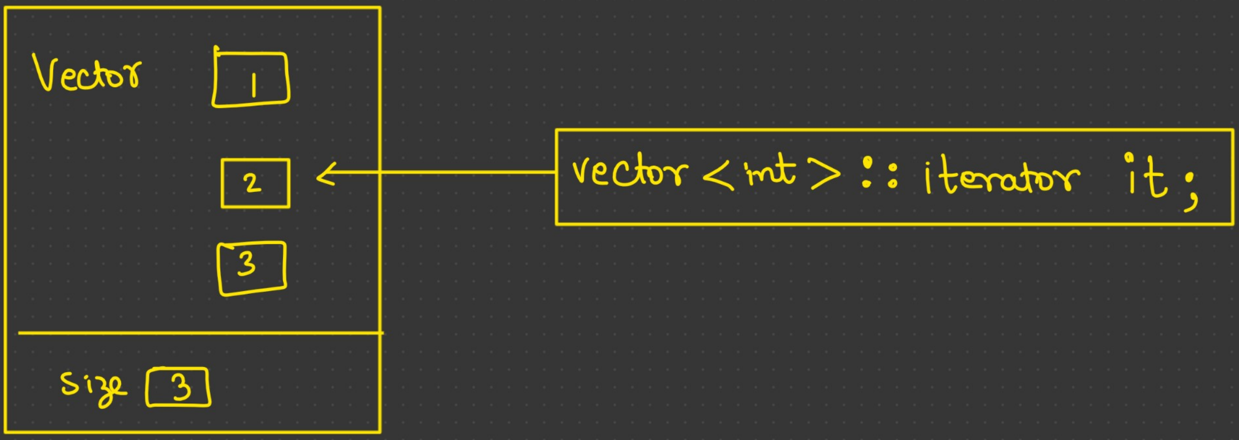


* Each container class contains a set of funtions that can be used to manipulate the contents

# Iterator

* Iterators are pointer - like entities used to access the individual elements in a container.

* Iterators are moved sequentially from one element to another element. This process is known as iterating through a container.



```
vector<int>::iterator it;
```

# Array Class

```
template <class T>
class array
{

};
```

+

- at()
- get()
- operator[]
- front()
- back()
- size()
- max_size()
- swap()

# Vector Class

① #include <vector>

② vector <int> v;

③ vector <int> :: iterator it;

④
```
for( it = v.begin() ; it != v.end() ; it++)
{
    cout << *it << "  ";
}
```

size = 0 (size of array) , capacity = 0 (No.. of element present in array)

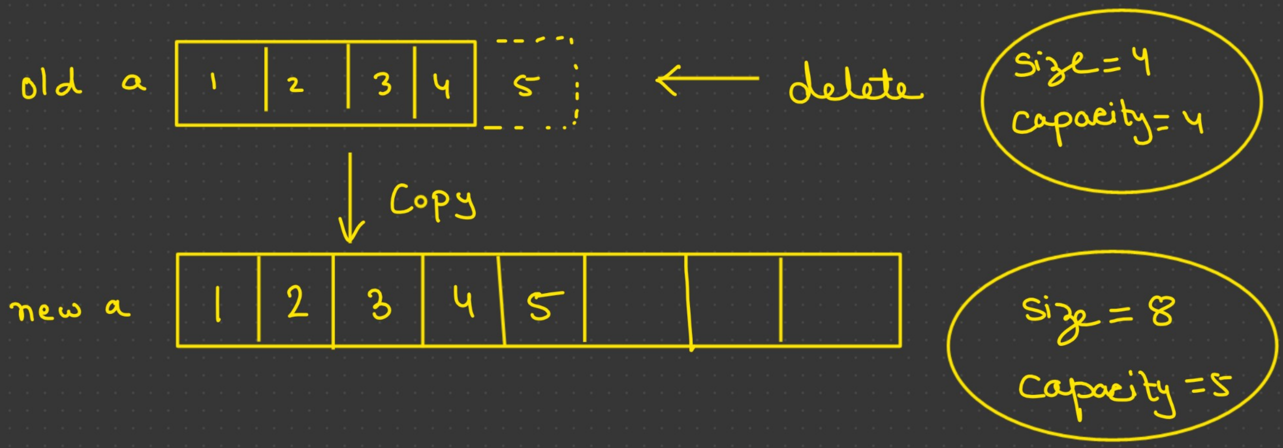
size = 1 , capacity = 1

size = 2 , capacity = 2

size = 4 , capacity = 3, 4

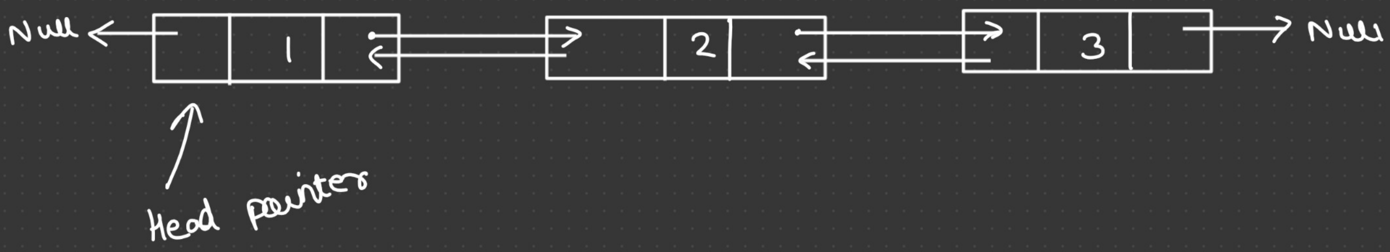
size = 8 , capacity = 5, 6, 7, 8

size = 16 , capacity = 9, 10, 11, . . . . . 16


old a
1
2
3
4
5
delete
size = 4
capacity = 4
Copy
new a
1
2
3
4
5
size = 8
capacity = 5

# List

- It is non-contiguous memory allocation.
- It is same as doubly linked list.
- It is slow in traversal as compared to vector.



- Insertion & Deletion is fast as compared to vector.

① `#include<list>`

② `list <int> l1;`

③ `l1.push_back(5);`

④ `list<int> :: iterator x;`

⑤
```
for ( x = l1.begin() ; x != l1.end(); x++)
{
    cout<<*x <<endl;
}
```