M T W T F S SU Date

$$\begin{tikzcd}
 & \boxed{\text{C++ STL}} \arrow[dl] \arrow[dr] & \\
\boxed{\text{algorithms}} & & \boxed{\text{Containers}}
\end{tikzcd}$$

1) Array

- `array<int, 4> a = {1,2,3,4};` (create)
- `int size = a.size()` (size of array)
- (print)

```
for(int i=0; i<size; i++){
    cout<<a[i]<<endl;
}
```

- `a.at(2)` $\longrightarrow$ element at 2nd Index
- `a.empty()` $\rightarrow$ Empty or not
- `a.front()` $\rightarrow$ first element
- `a.back()` $\rightarrow$ last element

2) Vector (Dynamic array)

-> double its size when vector is full. by copying old vector to doubled new vector & deleting old one.

-> #include <vector> -> library.

-> vector <int> v; -> create a vector.
-> v.capacity(); -> size. (double after full)
-> v.size();
-> v.push_back(1) -> add 1 in vector as element.

-> v.at(2) -> Element at 2nd Index.

-> front() & back().
-> v.pop-back(); -> remove last inserted element

-> v.clear(); -> remove all.

-> vector<int> a(5, 1); -> 5 element in vector with 1 as value for all.

3) Deque

→ #include <deque>

→ deque<int> d; → create

→ d.push_back(1); → add at end
d.push_front(2); → add at front.

→ d.pop_back();
d.pop_front();

→ d.at(1); → 1st index element.
→ front() , back()

→ d.erase (d.begin(), d.begin()+1);
↳ remove 1st element.

4) List

→ #include <list>

→ list<int> l; → create
→ l.push_back(1);
l.push_~~back~~ front(2);

5) Stack (last In first Out)

→ #include <stack>

→ stack<string> s;

```
s.push("Yash");
s.push("Sri");
s.push("Vastav");
```

→ s.top(); ⟶ o/p → $\boxed{\text{vastav}}$

→ s.pop(); ⟶ remove top.

⟶ o/p → $\boxed{\text{Sri}}$

→ s.size();

6) Queue (First In First Out)

→ #include <queue>

→ queue<string> q;

```
→ q.push("Yash");
  q.push("Srivastav");
```

→ q.front(); ⟶ o/p → $\boxed{\text{Yash}}$

q.pop(); ⟶ remove

q.front(); ⟶ o/p → $\boxed{\text{Srivastav}}$

7) Priority Queue

→ #include <queue>

→ priority_queue <int> p;   //max heap

→ priority_queue <int, vector<int>, greater<int>> mini;   //min heap;

```
→ p.push(1);
  p.push(2);
  p.push(3);

→ int n = p.size();
→ for(int i=0; i<n; i++){
        cout<<p.top();
        p.pop();
  }
```

O/P
```
3 2 1
```

M T W T F S SU Date

8) <u>Set</u> (store unique element only)

→ #include<set>

→
```
set<int> s;
s.insert(5);
s.insert(1);
s.insert(6);
s.insert(0);
```

→
```
for (auto i:s){
    cout<<i<<endl;
}
```

```
O/P:-
 0
 1
 5
 6
```

sorted order

→ s.erase(s.begin());

> s.count(5); → 5 present or not

→
```
set<int>::iterator itr = s.find(5);
cout<<*itr;
```
→ O/P → 1

M T W T F S SU   Date

9) Map

```
> #include<map>

→ map<int, string> m;

    m[1] = "Yash";             |  m.insert({5, "bheem"});
    m[12] = "Sri";             |
    m[3] = "Vastav";           |

  for(auto i:m){          ]   O/P
     cout<<i.first<<endl;  ]    1   ]
  }                        ]    2   ]  sorted
                                13  ]
```

10) Binary Search

```
> #include <algorithm>

>   vector<int> v;
    v.push_back(1);
    v.push_back(3);
    v.push_back(6);
    v.push_back(7);
> cout<<binary_search(v.begin(), v.end(), 6)
```

return true because it is present

> $\underbrace{\text{lower\_bound}}$(v.begin(), v.end(), 6) - v.begin();
  └→ o/p → $\boxed{2}$

$\underbrace{\text{upper\_bound}}$
  └→ o/p → $\boxed{3}$

> a=3, b=5;
  max(a,b); → 5
  min(a,b); → 3
  swap(a,b); → 5, 3

> String abcd = "abcd"
  reverse(abcd.begin(), abcd.end());
  └→ o/p → $\boxed{\text{dcba}}$

> rotate(v.begin(), v.begin()+1, v.end());
  └→ i/p → 1 3 6 7
     o/p → 3 6 7 1

> sort(v.begin(), v.end());
  └→ using Intro Sort → $\underbrace{\text{Quick, heap, Insertion}}_{\text{combination}}$