<Vector>

→ it is a dynamic array.

→ declaration :

vector <int> v; (creates a vector of size = 0).

vector <int> v1 (5); (creates a vector of size = 5)

vector <char> v2(5, 'a'); (creates a vector of size = 5 and all elements = 'a').

vector <int> v3{1,2,3,4,5}; (v3 = {1,2,3,4,5})
vector <int> v4(v3); (v4 = v3)

→ iterators

- begin() (on ++ goes towards end).
- end() (on -- goes towards begin).
- rbegin() (on ++ goes towards rend).
- rend() (on -- goes towards rend).

→ memory functions :

- size() (returns size)
- capacity() (returns capacity)

- empty() (return bool whether empty)

- resize(n) (resizes to store n elements)

(if size > n: add default value). (if size < n : then delete extra element).

- shrink-to-fit() (capacity = size).

→ 2-D vector

vector < vector <int> > v;

(vector of vector).

rows = v.size();

columns = v[0].size();

→ using iterator

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

element.

or vector<int>:: iterator it = v.begin();

easy loop :

```
for ( int /auto  i : v)
{
   cout << i;
}
```

(can also be done in conventional way v[i]).

→ element access

→ v [i]

→ v . at (i)

→ v . front ( ) (first ele)

→ v . back ( ) (last ele)

→ imp functions

→ . pop - back ( ) (removes last ele)

→ . push - back ( ) (adds element to the end)

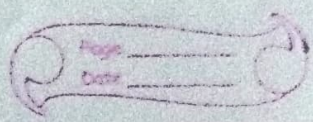
→ . insert (v. begin ( ) + i, value) (insert value at i)

→ . erase (v. begin ( ) + i) (remove ele at i)

End pointer (optional)

→ . clear ( ) (deletes all the ele)

<algorithms>



these change the container itself

→ sort (v.begin( ) , v.end( )) // asc sort

→ sort (v.begin( ), v.end( ), greater<int> ( ))
// desc sort

→ reverse (v.begin( ), v.end( )) // reverse

→ max_element (v.begin( ), v.end( )) // max

→ min_element (v.begin( ), v.end( )) // min

→ accumulate (v.begin( ), v.end( ), initial sum)
// sum
↓
put = 0.

→ returns

For arrays use arr for v.begin() and arr+n for v.end()