Vectors

04 June 2022 06:44 PM

Vector is a container which is dynamic in natue, you can always increase the size.

$$\boxed{\begin{array}{l} \text{int a[5]} = \{\underset{0}{\_},\underset{1}{\_},\underset{2}{\_},\underset{3}{\_},\underset{4}{\_}\} \\ \quad\quad\uparrow \\ \quad\quad\text{You can't modify this array} \end{array}}$$

```
void explainVector () {

    vector <int> v; // declaration - empty container { }

    v.push_back (1); // => {1}

    v.emplace_back(2); // similary to push back it dynamically
                       increase it size.
                       {1, 2}
```

emplace_back is faster than push back.

```
    vector <pair <int, int >> vec; // vector of pair

     v.push_back ({1,2}); // {1,2}

     v. emplace_back (1,2); // {1,2}

     Container of size 5 of element 100
     vector <int> v(5,100); // {100, 100, 100, 100, 100}
                  ↑             0    1    2    3    4
                 size

     vector <int> v (5); // {0, 0, 0, 0, 0} Garbage
                                             Value

     vector <int> v1 (5, 20); // {20, 20, 20, 20, 20}

      // to copy into another vector
```

```
vector <int> v2 (v1);        // {20, 20, 20, 20, 20}
```

How to access element in a vector?

v → {20, 10, 15, 5, 7}
      0   1   2 3 4

1st way
v[1] ⇒ 10
v[3] ⇒ 5

2nd way → iterator

Syntax:

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

$\underbrace{\text{vector <int> :: iterator}}_{\text{data type}}$ $\underbrace{\text{it}}_{\text{Variable}}$ = $\underbrace{\boxed{\text{v.begin();}}}_{\text{points directly on the memory}}$

v → {20, 10, 15, 5, 7}
      0   1   2 3 4
begin

```
it ++;

cout << *(it) << " "; //10

it = it + 2; // shift it by
                2 position

cout << *(it) << " "; // 5
```

printing the memory address, not the element.

in order to access the element in C++ we use *.

*(v.begin()) = 20

it ++ // it move to next memory

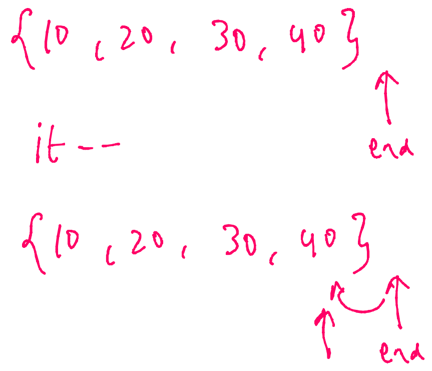
*(v.begin()) = 10

END

```
vector <int> :: iterator it = v.end()
```

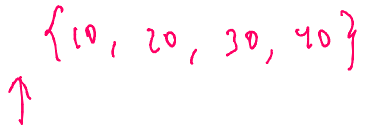
{10, 20, 30, 40}

```
Vector <int> :: iterator it = v.end()
```

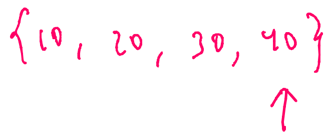


REND : (Reverse END)

```
Vector <int> :: iterator it = v.rend()
```

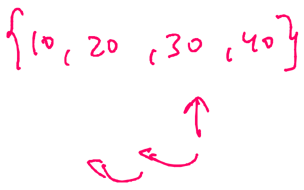


RBEGIN : (Reverse begin)



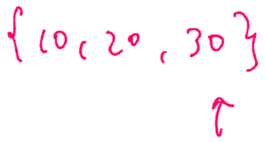
```
Vector <int> :: iterator it = v.rbegin()
```

it++ ⇒

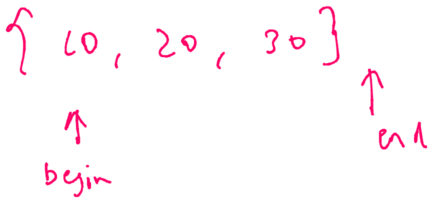


(in reverse way it move.

```
Cout << v.back() << " "; // 30
```



// print using for loop



```
for (vector <int> :: iterator it = v.begin () != v.end (); it++) {
        Cout << *(it) << " ";
}
```

output ⇒ 10 20 30

```
// More short way to use

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

output ⇒ 10 20 30

```
// More More shorted way

    for (auto it : v) {
      cout << it << " ";
    }
```

output ⇒ 10 20 30

Deletion in Vector:

{10, 20, 12, 23}

```
v.erase(v.begin() +1);    // {10, 12, 23}
```

{10, 20, 12, 23, 35}

```
v.erase(v.begin() + 2, v.begin() +4);  // {10, 20, 35} [start, end)
```

↑ not included

Insertion in vector:

```
vector <int> v (2, 100);    // {100, 100}
```

```
V. insert (V. begin (), 300);     // {300, 100, 100}

V. insert ( v. begin () +1 , 2, 10)   // {300, 10, 10, 100, 100}


vector <int> copy (2, 50) ;   // {50, 50}

V. insert (v. begin () , copy. begin () , copy. end ()); // {50, 50, 300, 10,
                                                         10, 100, 100}
{10, 20}

V. size (); // 2 (How many element ther in vector)


{10, 20}

V. pop_back(); // {10}

V1 -> {10, 20}  V2 -> {30, 40}

V1. swap (V2);  V1 -> {30, 40}  V2 -> {10, 20}


V. clear (); // erase the entire vector


cout << V. empty ();   // {1} -> False   { } -> True
```