2️⃣

Vector

What is a Vector?

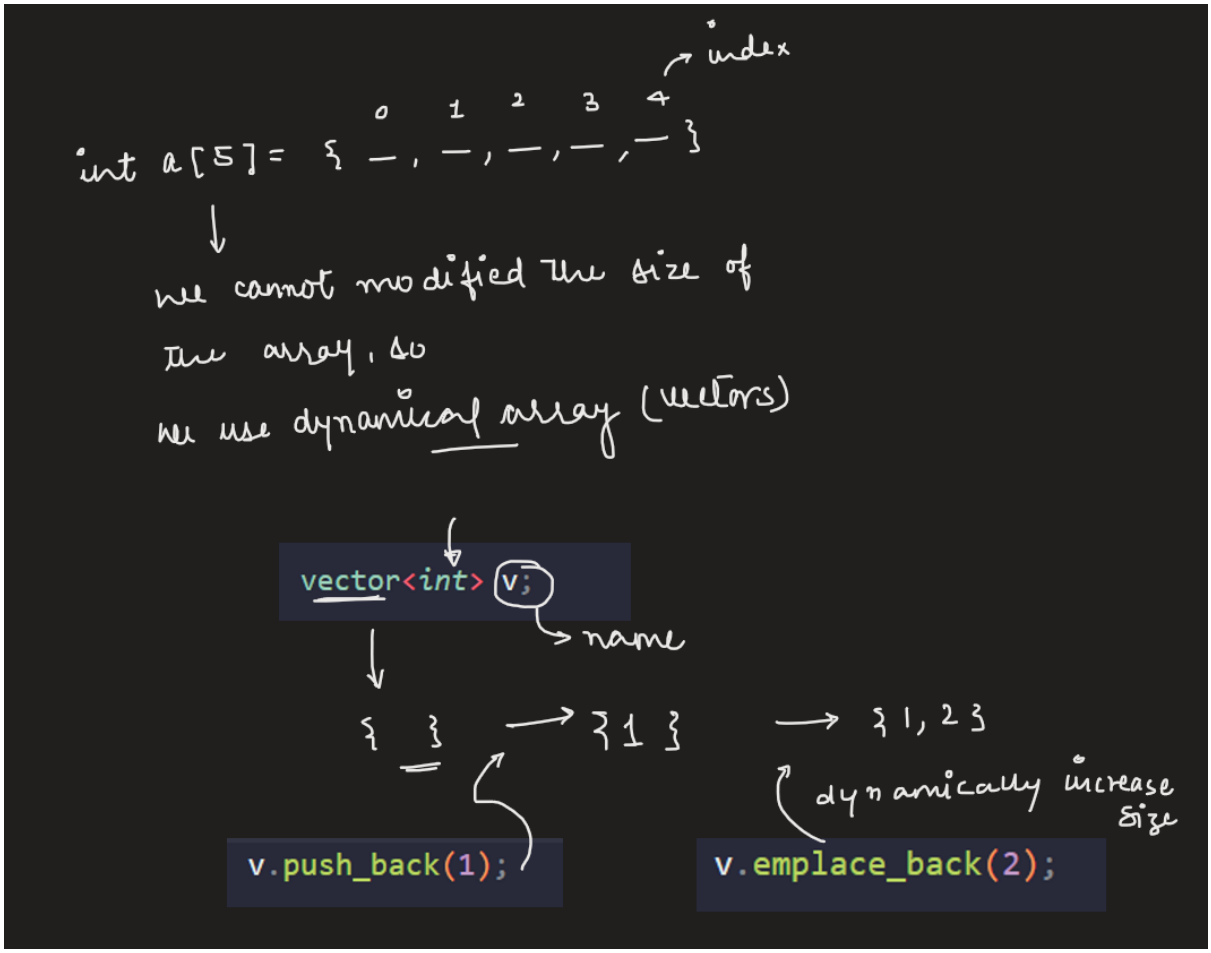
Vectors in STL are basically dynamic arrays that have the ability to change size whenever elements are added or deleted from them. Vector elements can be easily accessed and traversed using iterators. A vector stores elements in contiguous memory locations

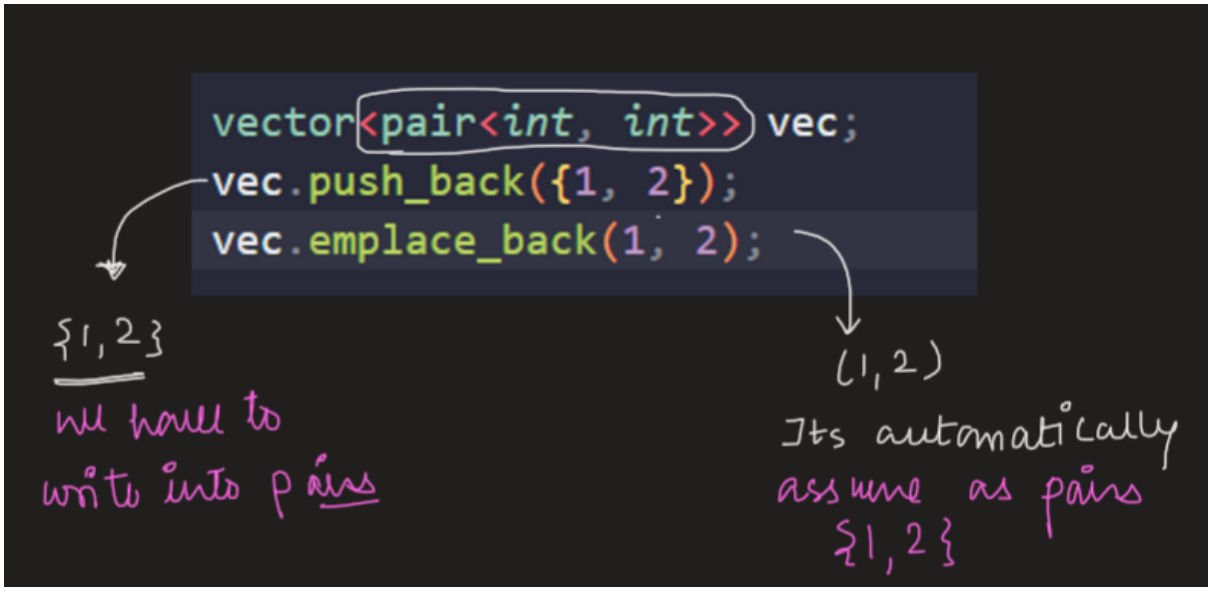
Syntax

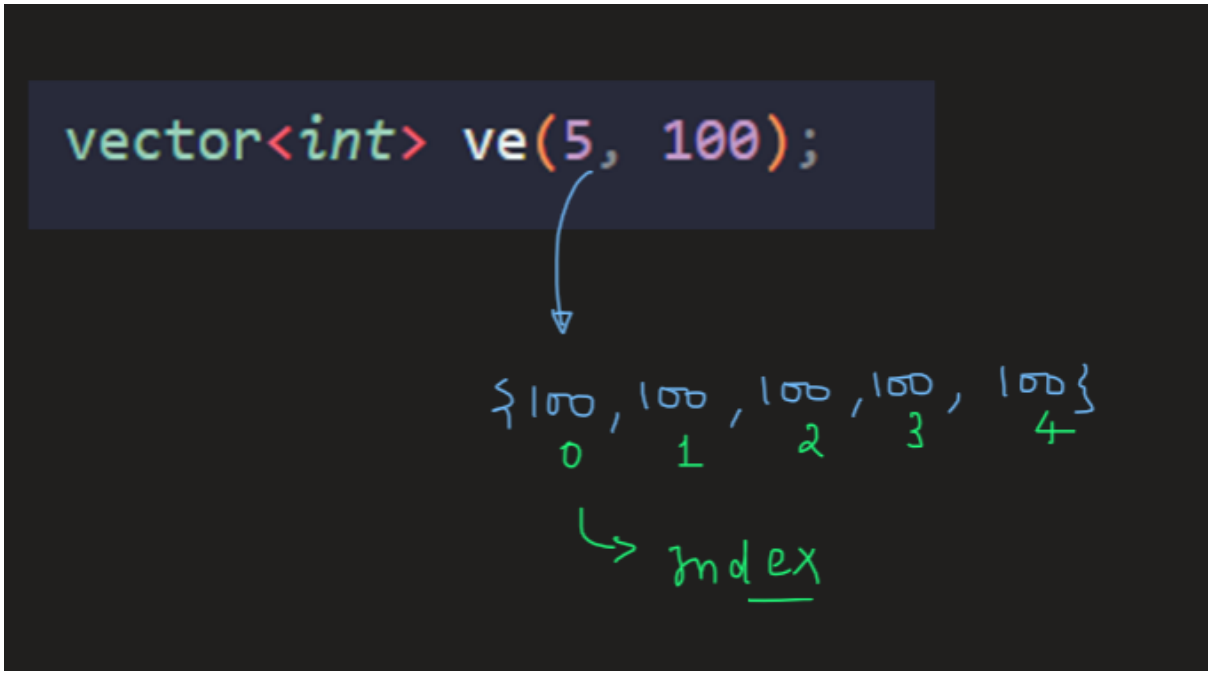
```
vector<object_type> variable_name;
```

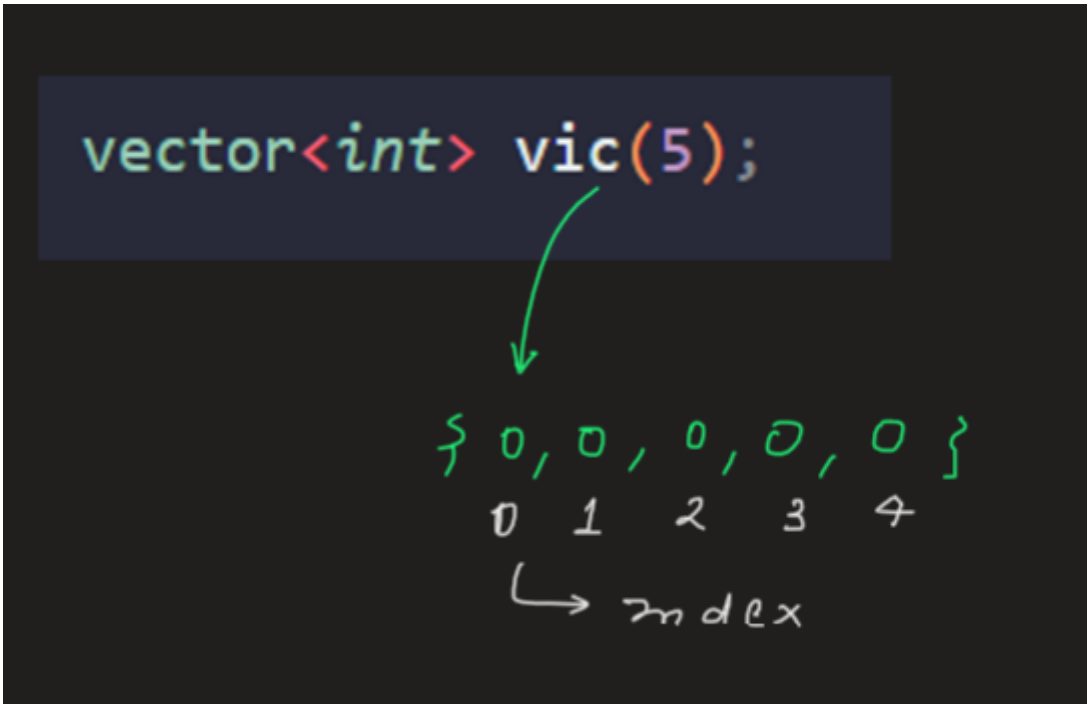
Example: -

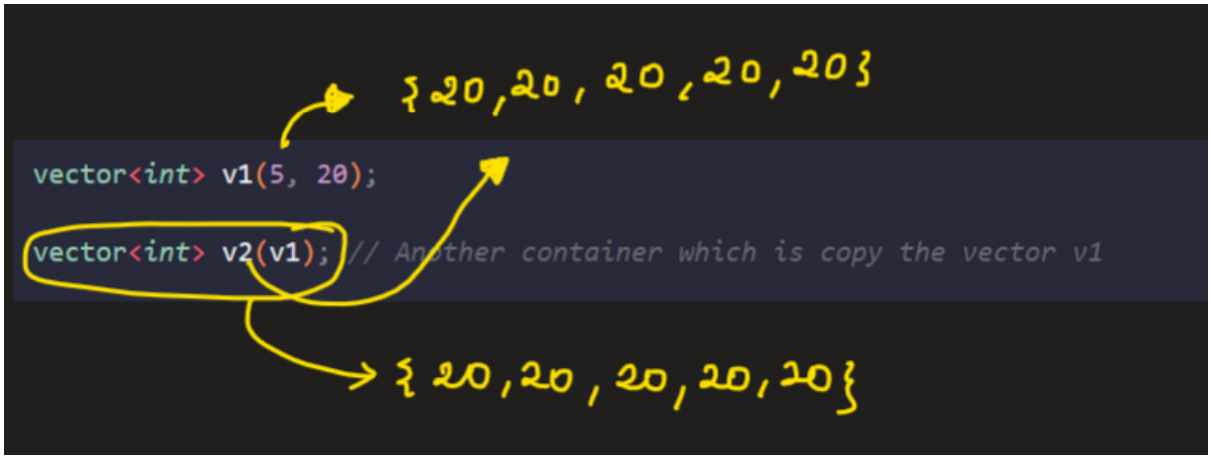
```
vector<int> v1;
vector<char> v2;
vector<string> v3;
```


index
0 1 2 3 4
int a[5] = { _ , _ , _ , _ , _ }
we cannot modified the size of the array, so
we use dynamical array (vectors)
vector<int> v;
name
{ }
{1}
{1, 2}
dynamically increase size
v.push_back(1);
v.emplace_back(2);

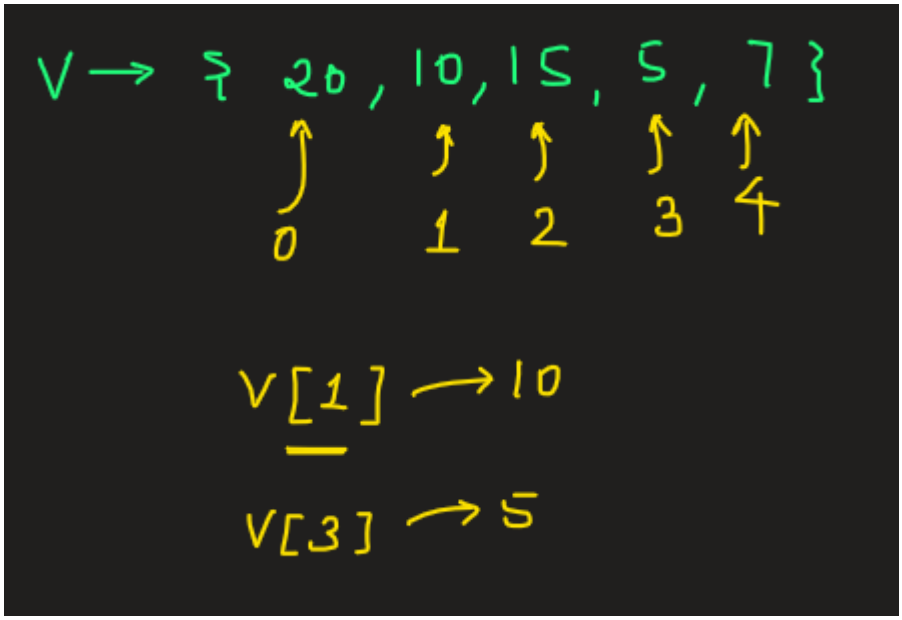

vector<pair<int, int>> vec;
vec.push_back({1, 2});
vec.emplace_back(1, 2);
{1, 2}
we have to write into pairs
(1, 2)
Its automatically assume as pairs
{1, 2}

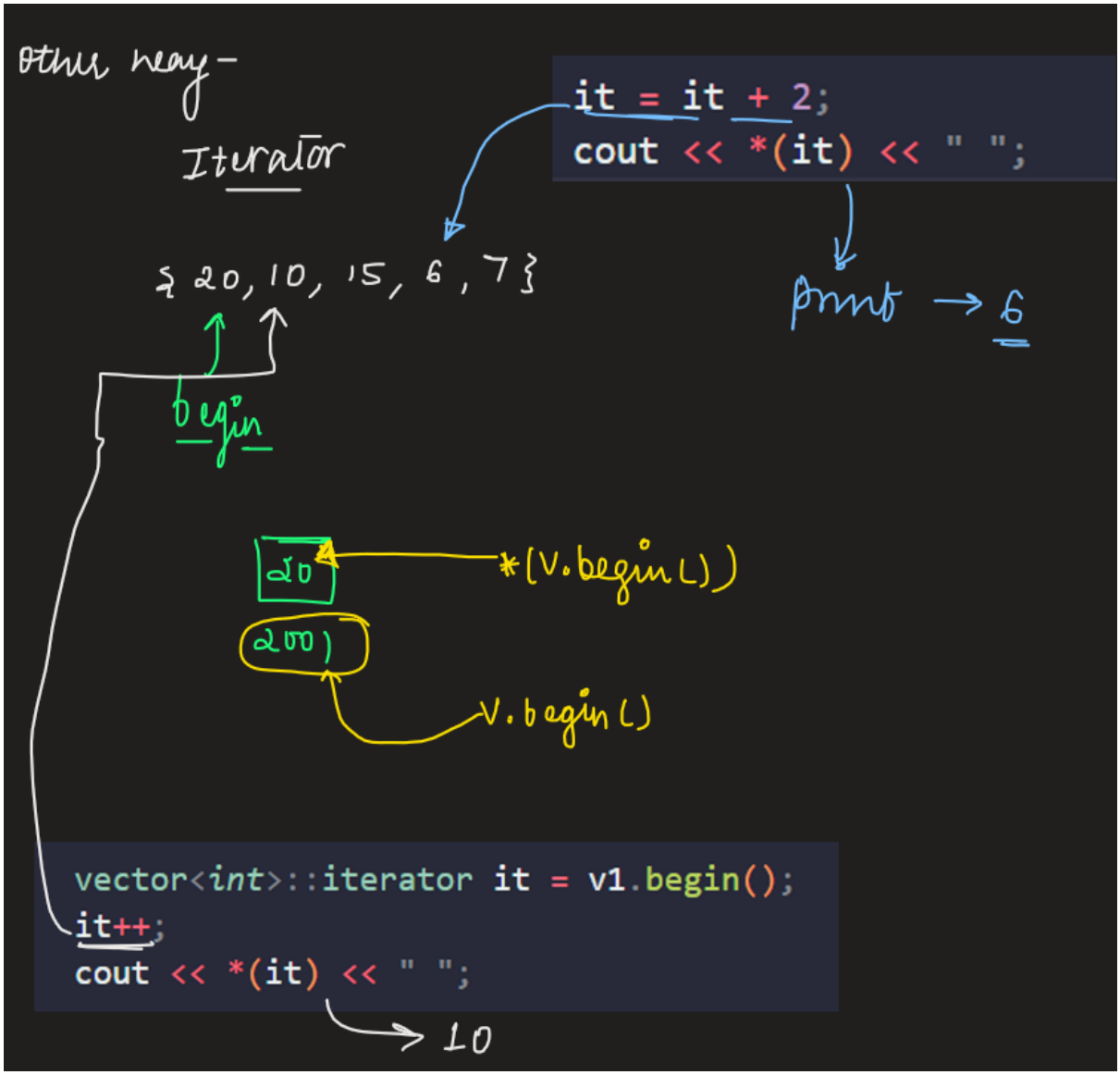

vector<int> ve(5, 100);
{100, 100, 100, 100, 100}
0 1 2 3 4
index

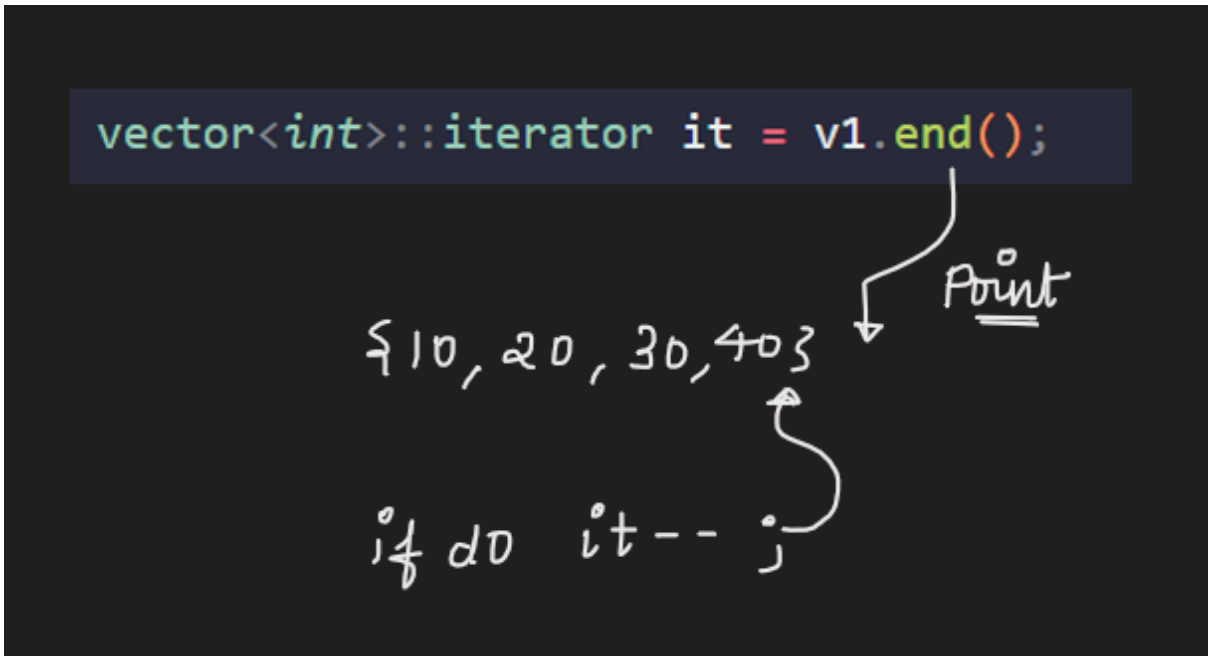

vector<int> vic(5);
{0, 0, 0, 0, 0}
0 1 2 3 4
index

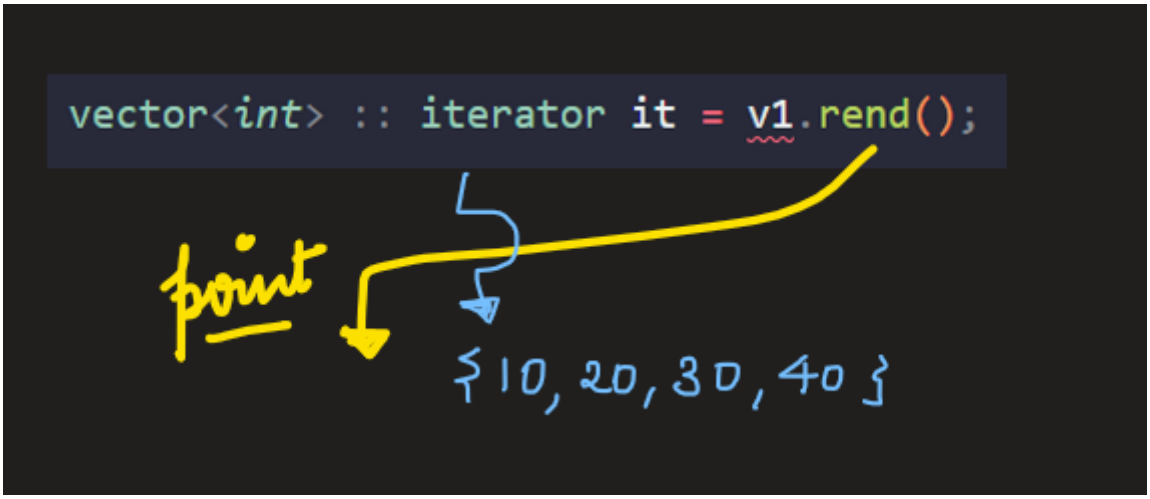


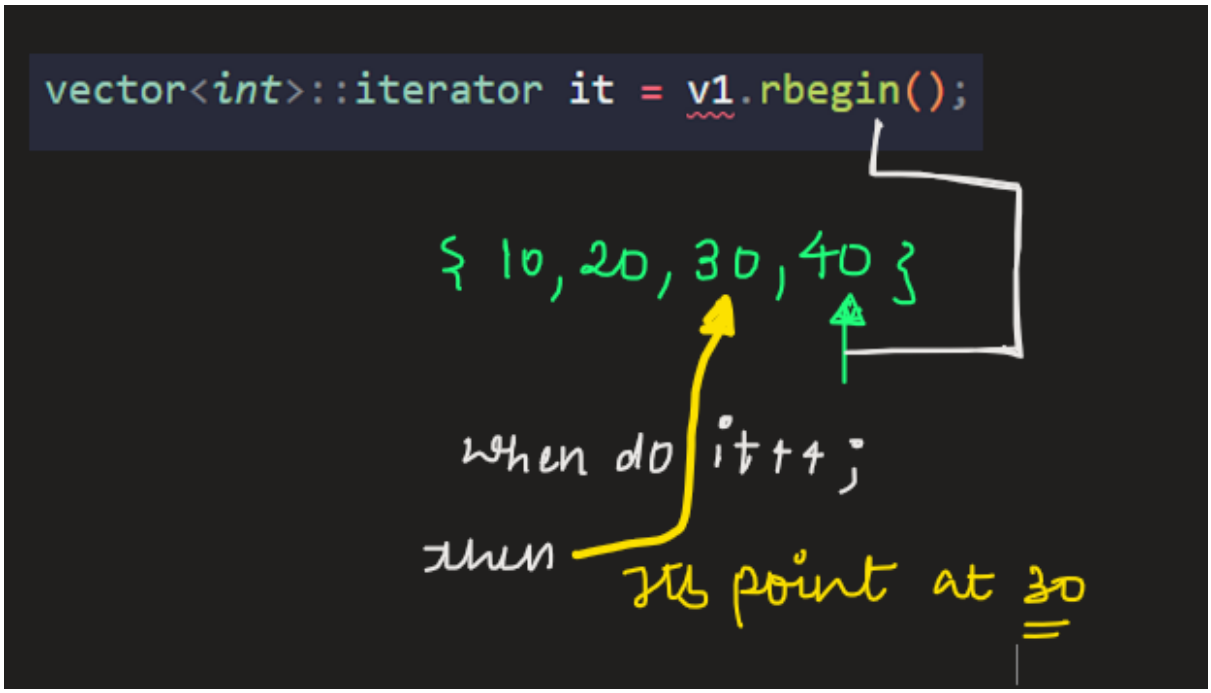
How to access the vector—

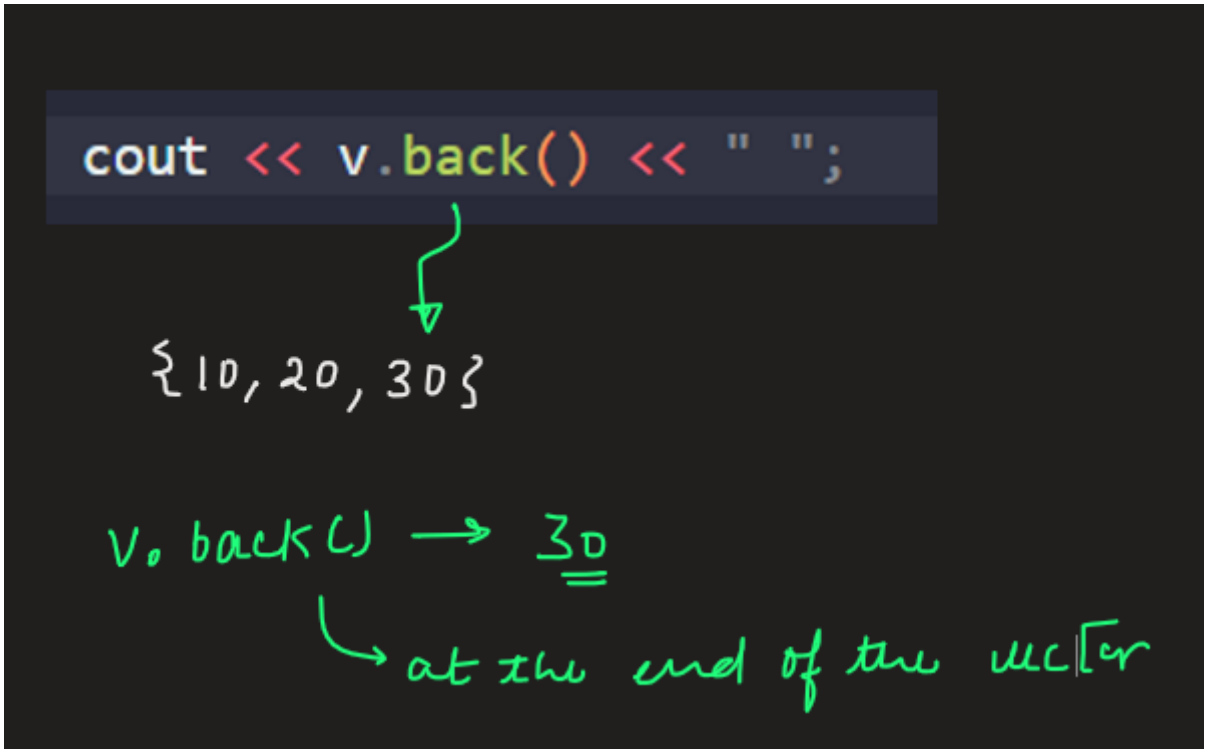



Other way -
Iterator
it = it + 2;
cout << *(it) << " ";
{ 20, 10, 15, 6, 7 }
Print → 6
begin
20
200
*(V.begin())
V.begin()
vector<int>::iterator it = v1.begin();
it++;
cout << *(it) << " ";
10


vector<int>::iterator it = v1.end();
Point
{10, 20, 30, 40}
if do it--;


vector<int> :: iterator it = v1.rend();
point
{10, 20, 30, 40}


vector<int>::iterator it = v1.rbegin();
{ 10, 20, 30, 40 }
when do it++;
then
it point at 30



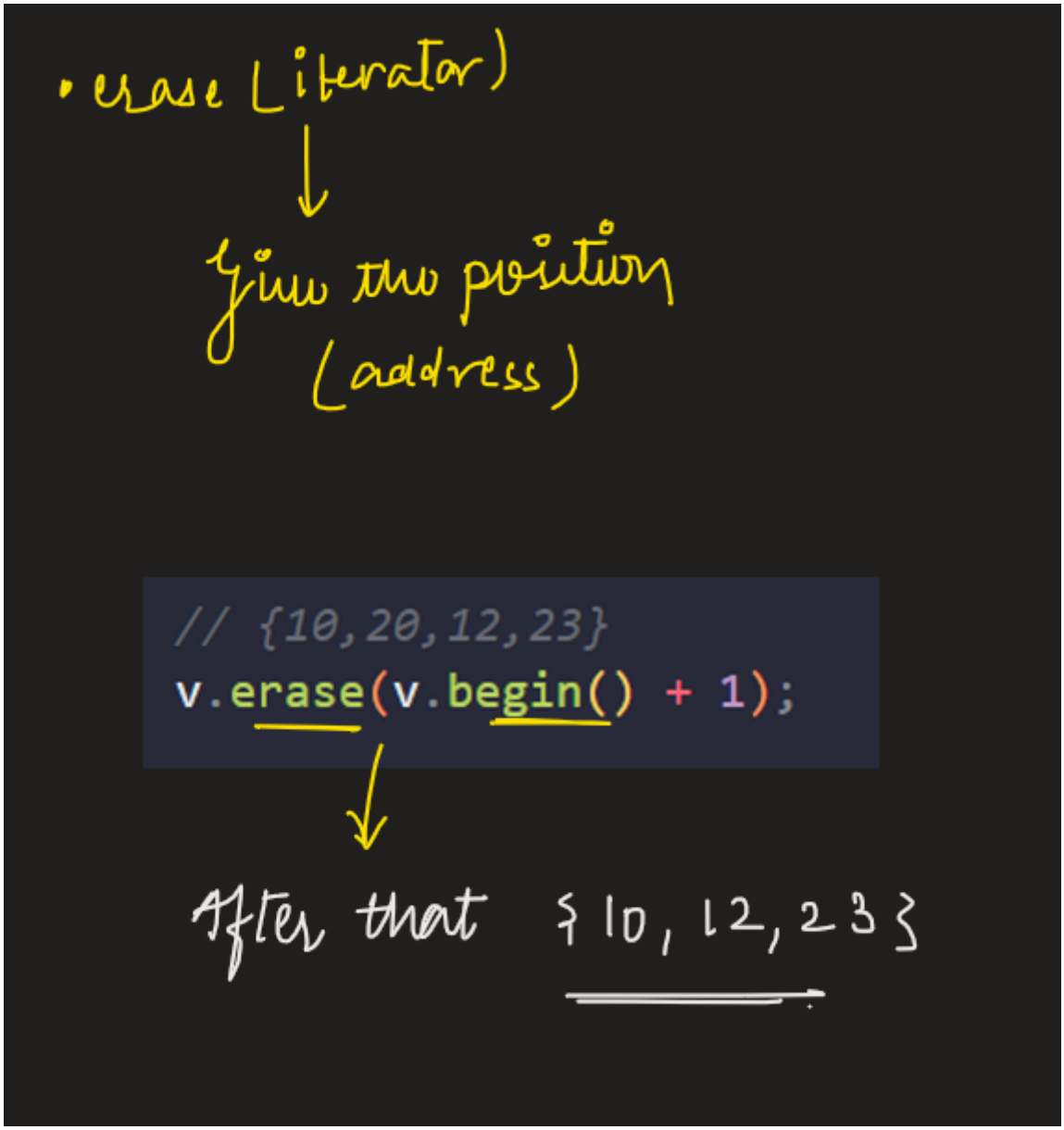
How to print the vector array using for loop

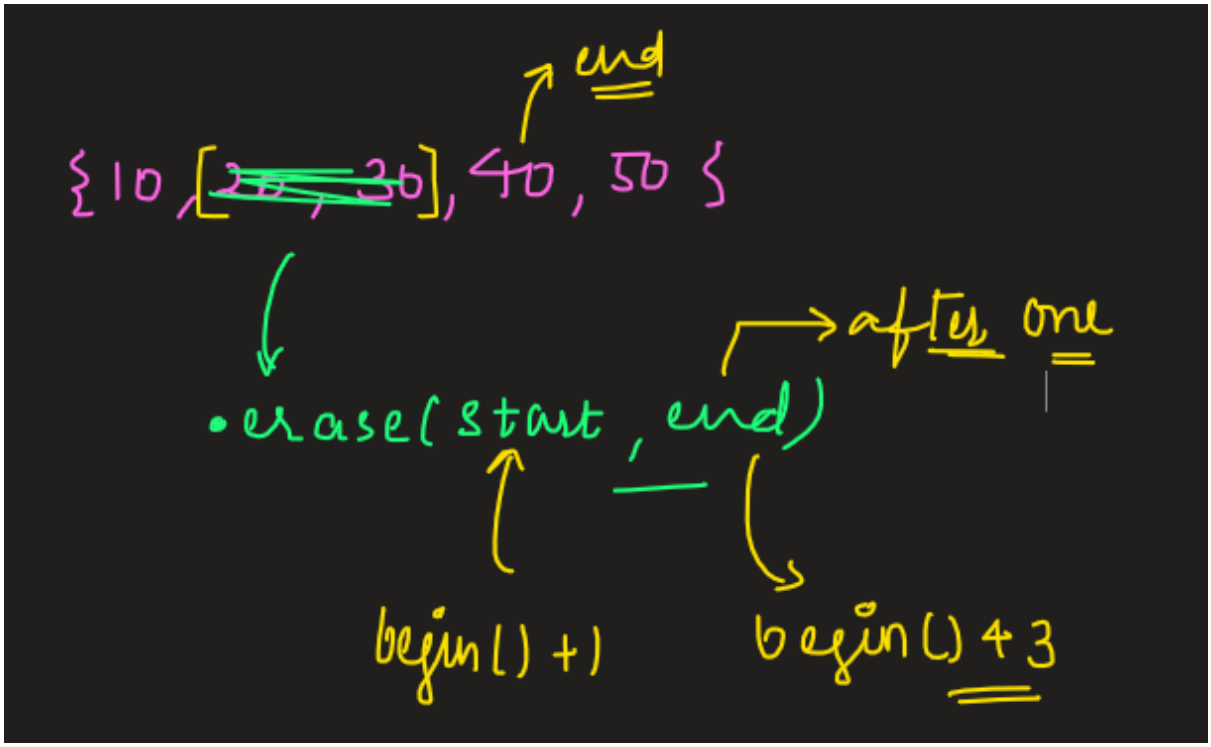
```
for (vector<int>::iterator it = v.begin(); it != v.end();it++)
{
      cout << *(it) << " ";
}

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

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

Deletion in a vector


• erase (iterator)
Give the position
(address)
// {10,20,12,23}
v.erase(v.begin() + 1);
After that {10, 12, 23}



Insert Function

```
// Insert Function --
      vector<int> vv(2, 100); //{100,100};
      vv.insert(vv.begin(), 300); // {300,100,100}
      vv.insert(vv.begin() + 1, 2, 10); // {300,10,10,100,100} where 2 is number of el
ement

      vector<int> copy(2, 50); // {50,50};
      vv.insert(vv.begin(), copy.begin(), copy.end()); // {50,50,300,10,10,100,100}
```

The Whole Code

```
// Vector --
#include <iostream>
#include <bits/stdc++.h>
using namespace std;
void explainVector()
{
      vector<int> v;
      v.push_back(1);
      v.emplace_back(2);
```

```
	// emplace_back is faster than push_back

	vector<pair<int, int>> vec;
	vec.push_back({1, 2});
	vec.emplace_back(1, 2);

	vector<int> ve(5, 100);

	vector<int> vic(5);

	vector<int> v1(5, 20);

	vector<int> v2(v1); // Another container which is copy the vector v1

	vector<int>::iterator it = v1.begin();
	it++;
	cout << *(it) << " ";

	it = it + 2;
	cout << *(it) << " ";

	vector<int>::iterator it = v1.end();

	// Never use this
	// vector<int> :: iterator it = v1.rend();

	// vector<int>::iterator it = v1.rbegin();

	cout << v[0] << " " << v.at(0);

	cout << v.back() << " ";

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

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

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

	// {10,20,12,23}
	v.erase(v.begin() + 1);

	// {10,20,30,40,50}
	v.erase(v.begin() + 2, v.begin() + 4); // {10,20,50}

	// Insert Function --
	vector<int> vv(2, 100); //{100,100};
	vv.insert(vv.begin(), 300); // {300,100,100}
	vv.insert(vv.begin() + 1, 2, 10); // {300,10,10,100,100} where 2 is number of el
ement
```

```
	vector<int> copy(2, 50); // {50,50};
	vv.insert(vv.begin(), copy.begin(), copy.end()); // {50,50,300,10,10,100,100}

	//v-->{10,20}
	cout << v.size(); // 2

	// v--> {10,20}
	v.pop_back(); // {10} it pop the last element in the vector

	// v1 --> {10,20}
	// v2 --> {30,40}
	v1.swap(v2); // v1--> {30,40} , v2--> {10,20}

	v1.clear(); // erases the entire vector

	cout << v.empty(); // Give true when vector is empty otherwise false
}
int main()
{
	explainVector();
	return 0;
}
```

Most used functions in Vector:

begin() – it returns an iterator pointing to the first element of the vector.

```
auto iterator = itr;

itr = v1.begin();
```

end() – it returns an iterator pointing to the element theoretically after the last element of the vector.

```
auto iterator = itr;

itr = v1.end();
```

push_back() – it accepts a parameter and insert the element passed in the parameter in the vectors, the element is inserted at the end.

```
vector<int> v1;

v1.push_back(1);
v1.push_back(2);
```

insert() – it is used to insert an element at a specified position.

```
auto it= v1.begin();
v1.insert(it,5); //inserting 5 at the beginning
```

erase() – it is used to delete a specific element

```
vector<int> v1;
auto it= v1.begin();
v1.erase(it);// erasing the first element
```

pop_back() – it deletes the last element and returns it to the calling function.

```
v1.pop_back();
```

front() – it returns a reference to the first element of the vector.

```
v1.front();
```

back() – it returns a reference to the last element of the vector.

```
v1.back();
```

clear() – deletes all the elements from the vector.

```
v1.clear();
```

empty() – to check if the vector is empty or not.

```
v1.empty();
```

size() – returns the size of the vector

```
v1.size();
```

Striver Code for Vector

```
#include<bits/stdc++.h>

using namespace std;

int main() {
  vector < int > v;

  for (int i = 0; i < 10; i++) {
    v.push_back(i); //inserting elements in the vector
  }

  cout << "the elements in the vector: ";
  for (auto it = v.begin(); it != v.end(); it++)
    cout << * it << " ";

  cout << "\\nThe front element of the vector: " << v.front();
  cout << "\\nThe last element of the vector: " << v.back();
  cout << "\\nThe size of the vector: " << v.size();
  cout << "\\nDeleting element from the end: " << v[v.size() - 1];
  v.pop_back();

  cout << "\\nPrinting the vector after removing the last element:" << endl;
  for (int i = 0; i < v.size(); i++)
    cout << v[i] << " ";

  cout << "\\nInserting 5 at the beginning:" << endl;
  v.insert(v.begin(), 5);
  cout << "The first element is: " << v[0] << endl;
  cout << "Erasing the first element" << endl;
  v.erase(v.begin());
```

```
  cout << "Now the first element is: " << v[0] << endl;

  if (v.empty())
    cout << "\\nvector is empty";
  else
    cout << "\\nvector is not empty" << endl;

  v.clear();
  cout << "Size of the vector after clearing the vector: " << v.size();

}

Output:

the elements in the vector: 0 1 2 3 4 5 6 7 8 9
The front element of the vector: 0
The last element of the vector: 9
The size of the vector: 10
Deleting element from the end: 9
Printing the vector after removing the last element:
0 1 2 3 4 5 6 7 8
Inserting 5 at the beginning:
The first element is: 5
Erasing the first element
Now the first element is: 0
vector is not empty
Size of the vector after clearing the vector: 0
```

Other Functions:

- **cbegin()** – it refers to the first element of the vector.
- **cend()** – it refers to the theoretical element after the last element of the vector.
- **rbegin()** – it points to the last element of the vector.
- **rend()** – it points to the theoretical element before the first element of the vector.
- **crbegin()** – it refers to the last element of the vector.
- **crend()** – it refers to the theoretical element before the first element of the vector.
- **max_size()** – returns the maximum size the vector can hold.
- **capacity()** – it returns the current capacity of the vector.