* **Vectors** : Till Now we have studied about static array in which we explicitly providing the size of an array. `#include <vector>`

- Vectors ~~are~~ is a Data structures.
- Vector is a Dynamic array. whose size can be changed during runtime.

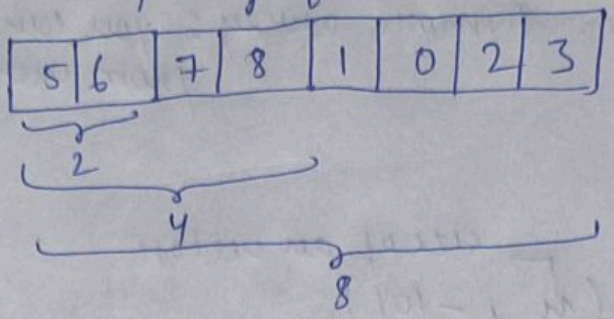
**Declaration**

```
vector<int> arr;
```

- vector → vector keyword
- int → datatype
- arr → variable name

* By default the size of vector is 0.

* The size of vector multiplies to double when the array get completely filled.




* Basic functions of Vector:

- push_back() → to add any elements to an array
- pop_back() → to remove any element from an array
- size() → tells how many elements are there inside an array.
- capacity() → maximum input an an array can take said to be capacity.

* let take an example:

```
vector<int> arr;
arr.push_back(33);
arr.push_back(2);
arr.push_back(1);
arr.push_back(4);
arr.push_back(3);
```
→ to add the items to an array.

```
for(int i=0; i<arr.size(); i++){
    cout<<arr[i]<<" ";
}
```

output: 33 2 1 4 3

```
arr.pop_back(); // → it remove the last element of an array.
for(int i=0; i<arr.size(); i++){
    cout<<arr[i]<<" ";
}
```

output: 33 2 1 4

```
cout<<arr.size() // 4
cout<<arr.capacity //
}
```

* int n;
cin >> n;
arr[n];
Bad practice

array[10] → static array
fix size.

Dynamic array: you can ask size from user.

* ~~vect~~ int n;
cin >> n;
vector<int> arr(n, -101)
↳ size of an vector
↳ initializer

vector → keyword, int → datatype, arr → variable name

Output

5 → size of an vector

It will print -101 5 times because -101 is an initializer

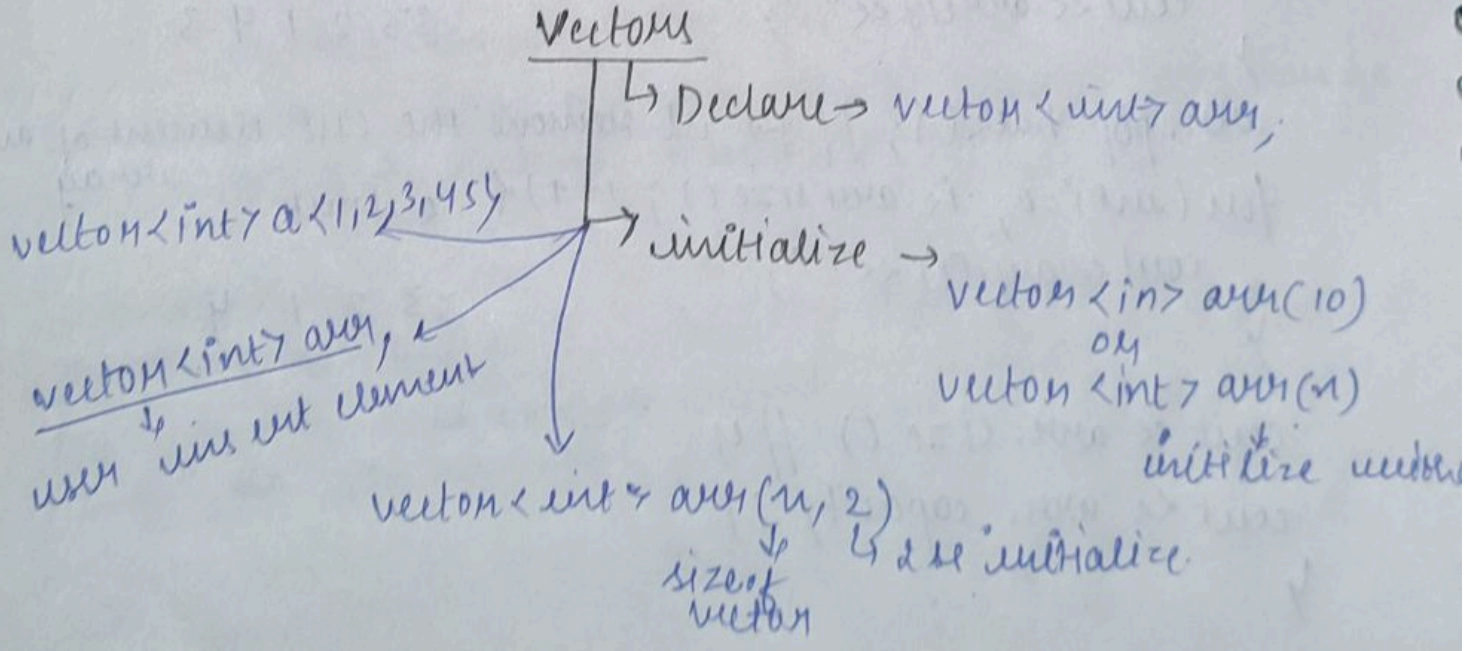
-101 -101 -101 -101 -101

* We can also assign an value in a vector

```
vector<int> p<2,3,4,5,6>;
cout<p.size()<<endl;
cout<<p.capacity()<<endl;
for(int i=0; i<p.size(); i++){
    cout<<p[i]<<" ";
}
```

Output

5
5
2 3 4 5 6



* arr.empty(); to check whether vector is empty or not.

* To take the size of a vector from the user:

```
int n;
cin>>n;
vector<int> a(n);
for(int i=0; i<a.size(); i++){
    cin>>a[i];
}
for(int i=0; i<a.size(); i++){
    cout<<a[i]<<" ";
}
```

Output

```
6 → size of vector is 6
1
2
3
4
5
6
1 2 3 4 5 6
```

① Find the unique element:

We can find unique element with XOR operator.
As 0^0 → 0 (same element will give 0 and
0^1 → 1 different element give 1)

①
```
int num;
cin>>num;
vector<int> arr(num);
for(int i=0; i<arr.size(); i++){
    cin>>arr[i];
}
for(int i=0; i<arr.size(); i++){
    cout<<arr[i];
}
int unique = findUniqueElement(arr);
cout<<unique;
```

```
int findUniqueElement(vector<int> arr){
    int unique = 0;
    for(int i=0; i<arr.size(); i++){
        unique = unique ^ arr[i];
    }
    return unique;
}
```

```
7
1
2
4
2
1
3
3
```

Array

1 2 4 2 1 3 3

4 → unique element

② Union of Two array

```
void findUnion(vector<int> arr, vector<int> brr){
    vector<int> unionVec;
    for(int i=0; i<arr.size(); i++){
        unionVec.push_back(arr[i]);
    }
    for(int i=0; i<brr.size(); i++){
        unionVec.push_back(brr[i]);
    }
    cout<<"The union of vector arr and brr is"
    for(int i=0; i<unionVec.size(); i++){
        cout<<unionVec[i]<<" ";
    }
}
int main(){
    int num1;
    cout<<"Enter the size of a vector :"; cin>>num1;
    vector<int> arr(num1);
    for(int i=0; i<arr.size(); i++){
        cin>>arr[i];
    }
    for(int i=0; i<arr.size(); i++){
        cout<<arr[i];
    }
    endl;
}
```

```
int num2;
cout<<"enter the size of a vector :"; cin>>num2;
vector<int> brr(num2);
for(int i=0; i<brr.size(); i++){
    cin>>brr[i];
}
for(int i=0; i<brr.size(); i++){
    cout<<brr[i];
}
endl;
findunion(arr, brr);
}
```

output

```
enter the size of vector : 3
1
2
3
array 1 -> 1 2 3
enter the size of vector : 3
4
5
6
array 2 -> 4 5 6
union is
1 2 3 4 5 6
```

③ Intersection of 2 array:

```
vector<int> findintersect(vector<int> arr, vector<int> brr){
  vector<int> intersectVec;
  for(int i=0; i<arr.size(); i++){
    for(int j=0; j<brr.size(); j++){
      if(arr[i]==brr[j]){
        intersectVec.pushback(arr[i]);
      }
    }
  }
return intersectVec;
}
int main(){
start same as union of 2 arrays.
calling of func:
vector<int> intersectVec = findintersect(arr, brr);
cout<<"The intersection of arr & brr is"<<endl;
for(int i=0; i<intersectVec.size(); i++){
    cout<<intersectVec[i]<<" ";
}
}
```

Output

enter the size of vector: 5
1
2
3
4
5
1 2 3 4 5
enter the size of vector: 5
3
4
5
6
7
3 4 5 6 7

Intersection of arr1 & arr2 is
3 4 5

(4)

Pair sum:

a[] = {1, 2, 3, 4, 5, 6}

sum = 6 ~~(3,3)=6~~ (1,5) = 6

(2,4) = 6

i.e (2,4) = 6 and (1,5) = 6

```
void findpairsum(vector<int>arr, int sum){
  for(int i=0; i<arr.size(); i++){
    for(int j=i+1; j<arr.size(); j++){
      if(arr[i]+arr[j]==sum){
        cout<<"Pair Found"<<arr[i]<<"and"<<arr[j]<<"equals"
        <<sum<<endl;
      }
    }
  }
}
int main(){
  // same as union input & printing of an array
  calling of function:
  int sum;
  cout<<" enter the number to check the sum:"; cin>>sum;
  findpairsum(arr, sum);
}
```

output

Enter the size of vector : 6

1
2
3
4
5
6

1 2 3 4 5 6

Enter the number to check the sum : 6

"Pair Found" 1 and 5 equals 6

" " 2 and 4 " 6

(5) Triplet sum :

a[] = {10, 20, 30, 40, 50, 60, 70, 80}

sum = 80

(10, 20, 50) = 80

(10, 30, 40) = 80

```
void findtripletsum (vector<int> arr, int sum){
  for(int i=0; i<arr.size(); i++){
    for(int j=i+1; j<arr.size(); j++){
      for(int k=j+1; k<arr.size(); k++){
        if(arr[i]+arr[j]+arr[k]==sum){
          cout<<"Pair Found"<<arr[i]<<"and"<<arr[j]<<"and"
              <<arr[i]<<arr[k]<<"equals"<<sum<<endl;
        }
      }
    }
  }
}
int main(){
same as union input & printing of an array
calling of function:
int sum;
cout<<"Enter the number to check sum :"; cin>>sum;
findtripletsum(arr, sum);
}
```

output

```
enter the size of the vector : 8
10 20 30 40 50 60 70 80
Pair found  10 and 20 and 50. equals 80
            10 and 30 and 40 equals 80
```

⑥ sort 0's and 1

a[] = | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |

output = | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

sorted array 0 on left side & 1 on right side.

using while loop

```
void sortingOand1(vector<int> arr) {
    int start = 0;
    int end = arr.size()-1
    int i=0;
    while (i != end) {
        if(arr[i] == 0) {
            swap(arr[i], arr[start]);
            start++;
            i++;
        }
        if( arr[i] == 1) {
            swap ( arr[end], arr[i])
            end--;
        }
    for( auto arr : arr) {  // means (i=0 ; i< arr.size ; i++)
        cout << arr;                    auto
    }
int main() {
    same as union input & output of an array.
    sortingOand1(arr);
}
```

Output

Enter the number/size of a vector: 9

0 0 1 0 1 0 1 0 0

The sorted array is:

0 0 0 0 0 0 1 1 1

// using for loop

everything will be same.

```
vector<int> sorting0and1 (vector<int>arr){
int start=0;
for(int i=0; i<arr.size(); i++){
    if(arr[i]==0){
            swap(arr[i], arr[start]);
            start++;
    }
}
for(auto val: arr){
    cout<<val<<" ";
}
```