

Array-2② Vector

↳ Dynamic array.

declaration → $\begin{matrix} \text{keyword} \\ \downarrow \\ \text{vector} \end{matrix} < \begin{matrix} \text{datatype} \\ \swarrow \\ \text{int} \end{matrix} > \text{arr};$ \swarrow variable name.

default size $\Rightarrow 0$, capacity $\Rightarrow 0$

It doubles its size whenever it is fully filled.

eg,

5	7
---	---

 ← ~~array~~ ^{vector} is full, now it will increase its size.↳

5	7	1	2
---	---	---	---

↳

5	7	1	2	6			
---	---	---	---	---	--	--	--

This memory is wasted.

Initialization-→ $\text{vector} < \text{int} > \text{arr} (10, 20, 30);$ →

10	20	30
----	----	----

→ $\text{vector} < \text{int} > \text{arr} (10, -1);$ →

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
0	1	2	3	4	5	6	7	8	9

→ $\text{int } n;$
 $\text{cin} \gg n;$
 $\text{vector} < \text{int} > \text{arr}(n);$

• how to insert elements → $\text{arr.push_back}(5);$ $\text{arr.push_back}(7);$ • remove the last element - $\text{arr.pop_back}();$ • size - returns the number of elements in vector.
 $\text{arr.size}();$ • empty - $\text{arr.empty}();$ • capacity() → returns the size of the storage space currently allocated for the vector.eg,

1	2	3	4				
---	---	---	---	--	--	--	--

capacity() → 8size() → 4

websites
 ↳ cppreference.com
 ↳ c++.com

$\frac{\text{sizeof(arr)}}{\text{sizeof(int)}} \rightarrow \frac{6}{3} \rightarrow \text{compiler dependent.}$

\Rightarrow
 $\text{int } n;$
 $\text{cin} \gg n;$

\rightarrow
 $\text{vector} < \text{int} > \text{brr}(10);$
 $\text{cout} << \text{"size of brr"} << \text{brr.size()} << \text{endl}; \quad // 10$
 $\text{cout} << \text{"capacity of brr"} << \text{brr.capacity()} << \text{endl}; \quad // 10$
 $// \text{printing the vector.}$
 $\text{for} (\text{int } i = 0; i < \text{brr.size()}; i++) \{$
 $\quad \text{cout} << \text{brr}[i] << \text{" "}; \quad // 0 \rightarrow 0000000000$
 $\}$

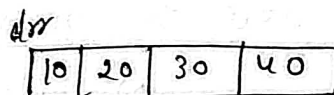
$// \text{Taking input for the vector :-}$

$\text{int } n;$
 $\text{cin} \gg n;$
 $\text{vector} < \text{int} > \text{crr}(n, -10);$



$// \text{create an } n \text{ size}$
 $\text{vector with having}$
 $\text{value of } -10 \text{ each.}$

$\rightarrow \text{vector} < \text{int} > \text{drr} \{ 10, 20, 30, 40 \}; \rightarrow$



① empty or not?

$\text{cout} << \text{drr.empty()} << \text{endl};$

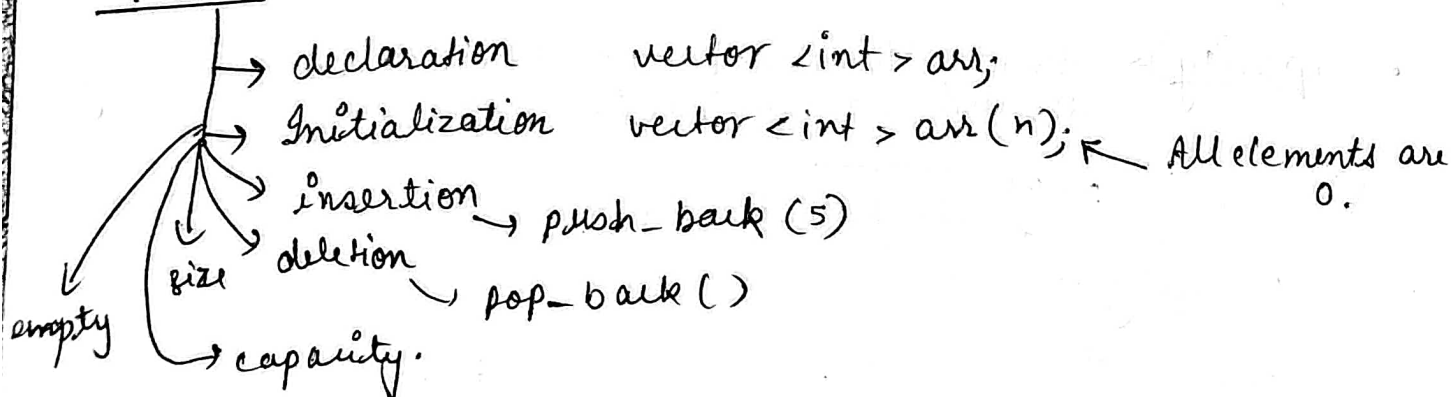
$// 0 \rightarrow \text{False}$
 \Downarrow that means
 it's not empty.

$\rightarrow \text{vector} < \text{int} > \text{err};$

$\text{cout} << \text{err.empty()} << \text{endl};$

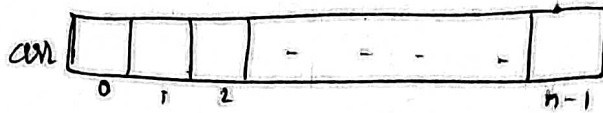
$// 1 \rightarrow \text{true}$
 means it's empty.

• Vector



// how to access n^{th} element.

↳ by indexing. , arr[n-1]



Q- Find Unique element.

i/p → arr

1	2	4	2	1	3	6	5	5	6	4
---	---	---	---	---	---	---	---	---	---	---

every element occurs twice, only 1 element occurs once.

main() { ⇒ XOR

1 ^ 2 ^ 4 ^ 2 ^ 1 ^ 3 ^ 6 ^ 5 ^ 5 ^ 6 ^ 4

= 3 ans

int n;

cin >> n;

vector<int> arr(n);

for (int i = 0; i < n; i++) {

cin >> arr[i];

}

int uniqueElement = findUnique(arr);

return 0;

}

int findUnique (vector<int> arr) {

int ans = 0;

for (int i = 0; i < n; i++) {

ans = arr[i]; // ans = ans ^ arr[i];

}

return ans;

}

Why we initialized ans with 0.

⇒ because →

$$0 \wedge 0 \rightarrow 0$$

$$0 \wedge n \rightarrow \underline{n} \quad \text{eg} \rightarrow 0 \wedge 1 = \underline{1}$$

so it is ineffective to XOR with 0.

① Union of 2 array →

i/o ① $a \rightarrow \{2, 4, 6, 8\}$

② $b \rightarrow \{1, 3, 7\}$

o/b Union $\rightarrow \{2, 4, 6, 8, 1, 3, 7\}$

↑ all elements with
no duplicate

Algo:

→ create an ans array/vector.

→ Put all elements of a into ans array.

→ Put all elements of b into ans array.

→ done. (Assuming there are no duplicates).

```
int n, m;
```

```
vector<int> arr(n);
```

```
vector<int> brr(m);
```

```
for (int i=0; i<n; i++){
```

```
    cin >> arr[i];
```

```
}
```

```
for (int i=0; i<m; i++){
```

```
    cin >> brr[i];
```

```
}
```

} taking
Input

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

```
for (int i=0; i<n; i++){
```

```
    ans.push_back(arr[i]);
```

```
}
```

```
for (int i=0; i<m; i++){
```

```
    ans.push_back(brr[i]);
```

```
}
```

} storing brr into
ans array.

```
// Printing the ans vector.
```

```
for (int i=0; i<n+m; i++){
```

```
    ans  
    cout << ans[i];
```

```
}
```

} printing the array.

↳ Some modified questions.

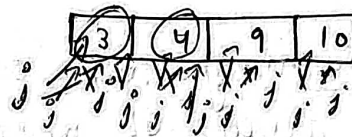
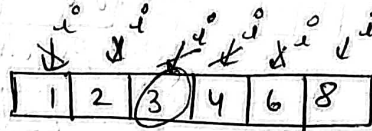
↳ When duplicates exist

↳ Answer should be in \uparrow increasing order.

③ Intersection of two arrays.

i/p $\left\{ \begin{array}{l} a = \{1, 2, 3, 4, 6, 8\} \\ b = \{3, 4, 9, 10\} \end{array} \right.$

o/p $\rightarrow \{3, 4\}$



ans =

3	4
---	---

Algo:

→ pick an element from array a.

→ compare with each element of b array.

→ if they are same then store it in ans array.

⇒ Code -

```
vector<int> ans;  
for (int i=0; i<n; i++){  
    int element = arr[i];  
    for (int j=0; j<m; j++){  
        if (element == arr[j])  
            ans.push_back(element);  
    }  
}
```

// print the ans vector.

~~for (int i=0; i< ans.size(); i++)~~ // for each loop.

```
for (auto value : ans){  
    cout << value << " ";  
}
```

→ What if we have duplicate element in a array.

eg, $a = \{1, 3, 3, 4, 6, 8\}$

$b = \{3, 3, 4\}$

output according to this code $\rightarrow \{3, 3, 3, 3, 4\}$

But it is wrong. o/p should be $\{3, 3, 4\}$.

so we will just add mark to show that the element is already matched with an element of another array.

```

for (i → 0 to n-1) {
    int element = arr[i];
    for (j → 0 to m-1) {
        if (element == brr[j]) {
            brr[j] = -1; // mark
            ans.push-back(arr[i]);
        }
    }
}

```

1	3	3	4	8	9
5	-1	-1	-1	-1	

Here assuming that only positive integer exist in our array. If negative integers also exist then we can mark with `INT_MIN`.

→ now we can easily avoid duplication in union.

```

if (brr[j] != -1)
    ans.push-back(element);
}

```

code in
H.W

⑥ Pair Sum →

i/p $a = \{1, 3, 5, 7, 4, 6\}$, find two elements (a pair) which adds up gives value equal to target.

Brute force →

→ find out all pairs. Check their sum is target or not.

$(1, 3)^4$ $(3, 5)^8$ $(5, 7)^{12}$ $(7, 2)^6$ $(2, 4)^6$ $(4, 6)^{10}$
 $(1, 5)^6$ $(3, 7)^{10}$ $(5, 4)^7$ $(7, 4)^{11}$ $(2, 6)^8$
 $(1, 7)^8$ $(3, 2)^5$ $(5, 4)^9$ $(7, 6)^{13}$
 $(1, 2)^3$ $(3, 4)^7$ $(5, 6)^{11}$
 $(1, 4)^5$ $(3, 6)^9$
 $(1, 6)^7$

target = 40

ans[] = {10, 20, 30, 40}

element = 10

$40 = 10 + 30$
 $40 = 20 + 20$
 $40 = 30 + 10$
 $40 = 40 + 0$

element = 30

outer loop $40 - 30 = 10$

element = 20

inner loop $40 - 20 = 20$

ans = (10, 30)

```

int target = 80;
vector<int> arr {10, 20, 30, 40, 50, 70};
// Print pairs, outer loop will traverse for each element.
for (int i = 0; i < arr.size(); i++) {
    int element = arr[i];
    // Traverse from (i+1)th element.
    for (int j = i+1; j < arr.size(); j++) {
        cout << "(" << element << ", " << arr[j] << ")" << endl;
    }
}
// This will print all pairs.

```

```

// now pair sum.
for (int i = 0; i < arr.size(); i++) {
    for (int j = i+1; j < arr.size(); j++) {
        if (arr[i] + arr[j] == target)
            cout << arr[i] << " " << arr[j] << endl;
    }
}

```

② Triplet Sum -

Dry Run -

arr[] → {10, 20, 30, 40} target = 40

(10, y)

10, 20 → 10+20=30

10, 30 → 40

10, 40 → 50

{20, 20, 30, 40}

(20, y)

↳ (20, 30) → 50

↳ (20, 40) → 60

{10, 20, 30, 40}

(30, y)

↳ (30, 40) → 70

{10, 20, 30, 40}

(40, y)

③ Triplet Sum -

a[] → {10, 20, 30, 40}, sum = 80.

(10, 20, 30)

(20, 30, 40)

(10, 20, 40) → 70

(10, 30, 40) → 80 ✓ ans

{10, 20, 30, 40}

loop i = 0 → n-1

(x, y, z) → loop k = . → n-1

loop j = i+1 → n-1

vector <int> arr {10, 20, 30, 40};

```
for (int i = 0; i < arr.size(); i++) {
    int element1 = arr[i];
    for (int j = i+1; j < arr.size(); j++) {
        int element2 = arr[j];
        for (int k = j+1; k < arr.size(); k++) {
            if (element1 + element2 + arr[k] == sum)
                cout << element1 << " " << element2
                    << " " << arr[k] << endl;
        }
    }
}
```

① Find 4 nums that adds up to sum.

Algo vector <int> arr {10, 5, 15, 20, 30};

pseudo code.

```
int sum = 20;
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++) {
            for (int l = k+1; l < arr.size(); l++) {
                if (arr[i] + arr[j] + arr[k] + arr[l] == sum) {
                    cout << arr[i] << arr[j]
                        << arr[k] << arr[l];
                }
            }
        }
    }
}
```

① Sort 0's & 1's →

i/p → arr →

0	1	1	0	1	0	1	0	0
---	---	---	---	---	---	---	---	---

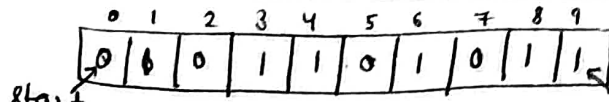
o/p →

0	0	0	0	0	1	1	1	1
---	---	---	---	---	---	---	---	---

Two pointers approach →

put 0 on start & 1 on end.

0	1	1	0	1	0	1	0	0
---	---	---	---	---	---	---	---	---



size = 10

start shows a position where we can insert next 0.

end shows a position where we can insert next 1.

if (arr[i] == 0)

swap(arr[i], arr[start])

i++;

start++;

}

if (arr[i] == 1)

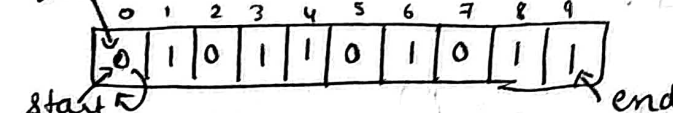
swap(arr[i], arr[end])

end--;

}

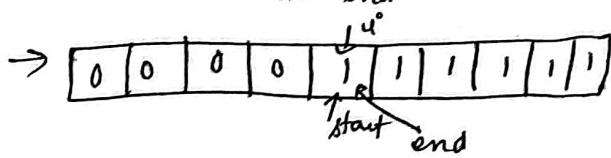
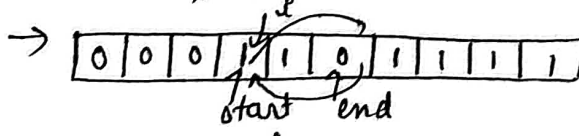
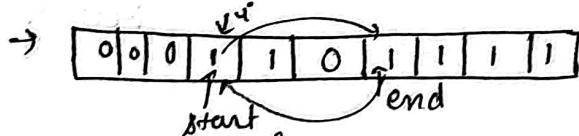
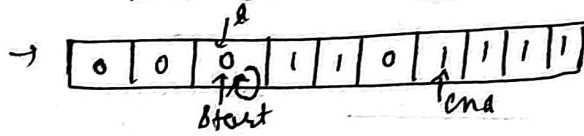
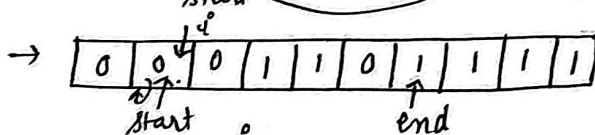
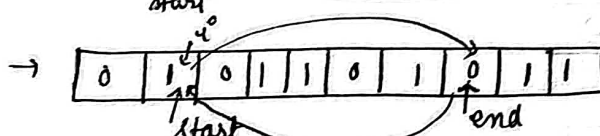
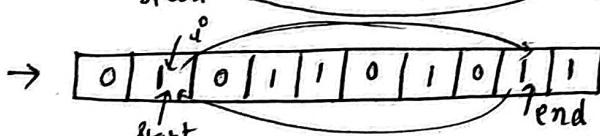
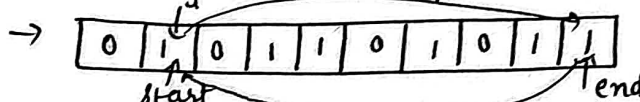
dry run -

i is used to traverse the array.



arr[start] = 0

arr[i] = 0 swap(arr[i], arr[start]) i++, start++;



i == end → stop

code -

vector <int> arr {0, 1, 0, 1, 1, 0, 1, 0, 1, 1};

int start = 0;

int end = arr.size() - 1;

int i = 0;

while (i != end) {

if (arr[i] == 0)

swap(arr[i], arr[start]);

i++;

start++;

if (arr[i] == ~~0~~ 1)

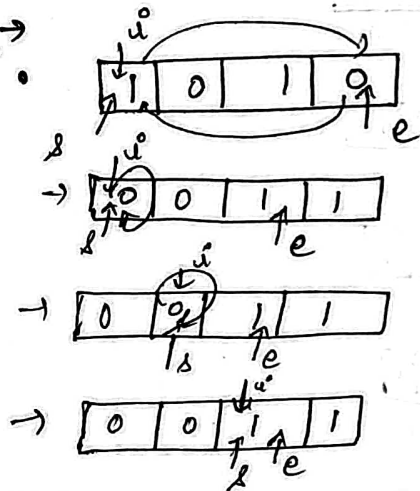
swap(arr[i], arr[end]);

end--;

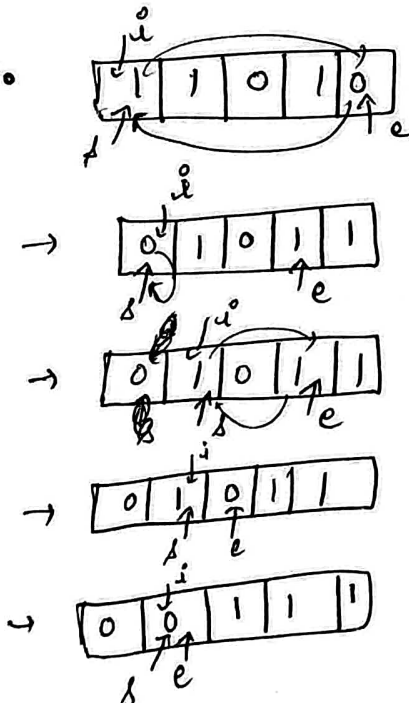
}

}

dry run →



i == end → array is sorted.
(out of loop).



i == e → stop the loop.

① for - each loop - keyword.
for (auto i : arr)
{
 cout << i;
}

By using auto keyword we don't need to tell the datatype explicitly.

Questions -

- ① left rotate an array by 1 element.
- ② majority element in an array.
- ③ Buy & Sell Stock → Level-1