

### ① What is Array -

- collection of similar type of elements.
- stored in contiguous memory location.
- linear Data Structure.



can we store 5 in this? Yes

"Hello" → NO

5.5 → NO

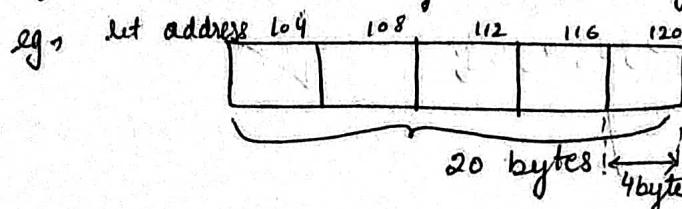
true/false → NO

} because it can only store integer.

### ② Behind the scene (Memory) -

An integer takes 4 bytes of space.

so an array of 5 integers takes a total memory of  $5 \times 4 = 20$  bytes (contiguous memory).



### ③ Why array is needed?

suppose we have to find greatest no. of two numbers. so we will make two variable, compare them and get the result.

Now suppose we want to find greatest element from 30/30000 elements. It's not convenient to create 30000 variables and compare.

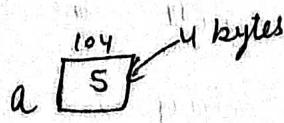
so that's why we use arrays to store element.

eg, int array [30000];

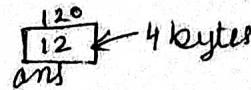
### ④ Creating an Array -

creating variable → int a = 5;

In memory →



int ans = 12;



Symbol table  
is a data structure maintained by compiler

Symbol Table
a → 104
ans → 120

## → declaration of array →

int arr [ 10 ] ; ←  
 datatype      ↓                  size of the  
                   name of         array.  
                   the array

Contiguous space of  
10 integers (40 bytes)  
is allocated.

## • Address of an array-

cout << arr ;  
 cout << &arr ; } both give address of  
the array.

## Questions -

Create an array of

- int → 53 size → int arr[53];
- char → 50 size → char arr[50].
- bool → 23 size → bool arr[23];

## → Initialization -

int arr [ 5 ] = { 1, 2, 3, 4, 5 }; Yes, this size is not mandatory here.

⇒ 

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

int arr [ ] = { 1, 2, 3, 4, 5 }; ✓

int arr [ 10 ] = { 1, 2, 3, 4, 5 }; ✓

Now the question is what is stored in the rest of 5 spaces? ⇒ zero (0).

int arr [ 4 ] = { 1, 2, 3, 4, 5 }; // Error.

Why Error?

Because the size of the array is 4 and we are storing 5 elements in it.

This is STATIC Array. Can we create an dynamic array? ↴ fixed size.

dynamic array is creation of array at the ~~at~~ runtime. (We will discuss it later).

We have one more way →

```
int n;  
cin >> n;
```

`int arr[n];`  
But this is BAD practice and not recommended.

• Questions-

• Create arrays

→ int → 1, 2, 3, 5

int arr[4] = {1, 2, 3, 5};

→ char array → a, b, c, d

char array [4] = {'a', 'b', 'c', 'd'};

① Index and Access in array →

int arr[5] = {10, 20, 30, 40, 50};

	104	108	112	116	120
arr	10	20	30	40	50
Index	0th index	1st index	2nd index	3rd index	4th index.

5 size array → index → [0 - 4]

so n size array → index → [0 to (n-1)]

To access first element →

arr[0] → 10

same way we can access other elements with the help of indexes.

arr[1] → 20, arr[2] → 30, arr[3] → 40, arr[4] → 50

What is the significance of 0?

arr[0]

↓

arr + 0x4

(arr + 0)

value at  
this address

value at (104) address

= 10

arr[1]

value at (arr + 1x4)

value at (104 + 1x4)

value at (104 + 4)

value at 108 = 20

204	10	0	$\text{arr}[0] \rightarrow \text{value at } (\text{arr} + \text{index} * \text{size of data type})$
208	20	1	$= \text{value at } (200 + 0)$
212	30	2	$= \text{value at } (200) = \underline{\underline{10}}$
216	40	3	$\text{arr}[4] \rightarrow \text{value at } (\text{arr} + 4 * 4)$
220	50	4	$\rightarrow \text{value at } 216$ $= \underline{\underline{40}}$

$\text{arr}[10] \rightarrow \text{value at } (\text{arr} + 10 * 4)$   
 $= \text{value at } \underline{\underline{244}}$

↳ (but this memory is not allocated for this array)

so in this case two things can happen.

- ① we get any garbage value.
- ② or we get overflow error.

### ④ Printing all values in a array →

```
int arr[] = {1, 3, 5, 7, 9};  

for (int index = 0; index < 5; index++) {  

    cout << arr[index];  

}
```

11dp → 1 3 5 7 9.

Dry run -

```
for (i = 0; i < 5; i++)  

{  

    cout << arr[i];  

}
```

$i = 0, 0 < 5 \rightarrow T$  print 1  $\text{arr}[0]$   
 $i = 1, 1 < 5 \rightarrow T$  print 3  $\text{arr}[1]$   
 $i = 2, 2 < 5 \rightarrow T$  print 5  $\text{arr}[2]$   
 $i = 3, 3 < 5 \rightarrow T$  print 7  $\text{arr}[3]$   
 $i = 4, 4 < 5 \rightarrow T$  print 9  $\text{arr}[4]$

### ⑤ Taking array's element as an input →

```
int arr[10];  

for (int index = 0; index < 10; index++) {  

    cin >> arr[index];  

}
```

→

```
int n;  

cin >> n;  

arr[0] = n;
```

④ Instead of using int arr[n],

we can use `int arr[10000];`

L is a big number.

even if we are using/storing only 10 elements in it.

Q1) Take 5 element int in array and print their address.

sol. int arr[5];

```
for (int i = 0; i < 5; i++) { // Taking 5 element in  
    cin >> arr[i];           input.
```

31

```
for (int i=0; i<5; i++){
```

`Carve[i] *= 2;`

of court as arr[ing]

printing their doubles.

or simply remove these two lines  
with a single line → cout << arr[i];

Q2. We have an array -

`ans[ ] = {1, 3, 5, 7, 9};`

change it's all elements to 1 like this  $\{1, 1, 1, 1, 1\}$

int arr[ ] = { 1, 3, 5, -7, 9 };

```
for (int i = 0; i < 5; i++) {
```

$\text{ans}[i] = 1;$

H-W → Find out what does memset func do and how to use it.

$$\textcircled{1} \quad \text{int arr[10]} = \{1, 2\};$$

```
for (int i=0; i<10; i++) {
```

cout << arr[i];

3

0/b → 1 2 0 0 0 0 0 0 0

⑥ int arr[10];

```
for (int i=0; i<10; i++) {
```

```
cout << arr[i];
```

3

0/10 → all 10 garbage values.

① To initialize all elements with 0:->

int arr[10] = {0};

① How to initialize all elements with 1?

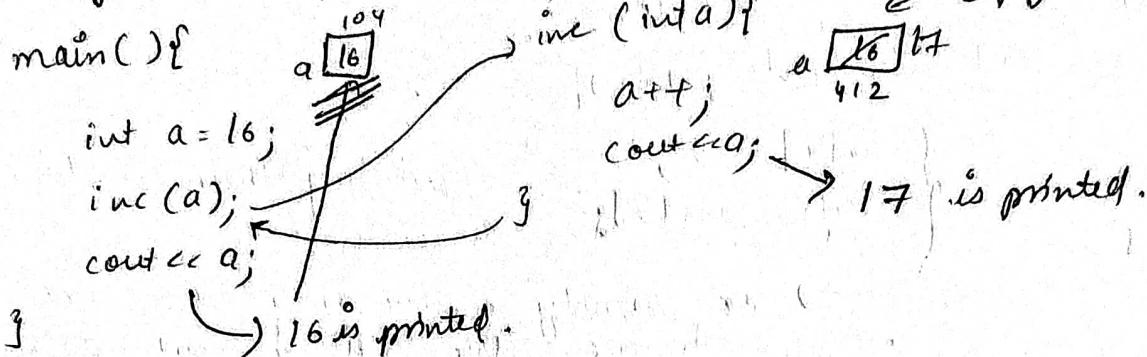
1  $\Rightarrow$  int arr[10];  
for (int i=0; i<10; i++) {  
 arr[i] = 1;

This will not work  
 $arr[10] = \{1\}$ .

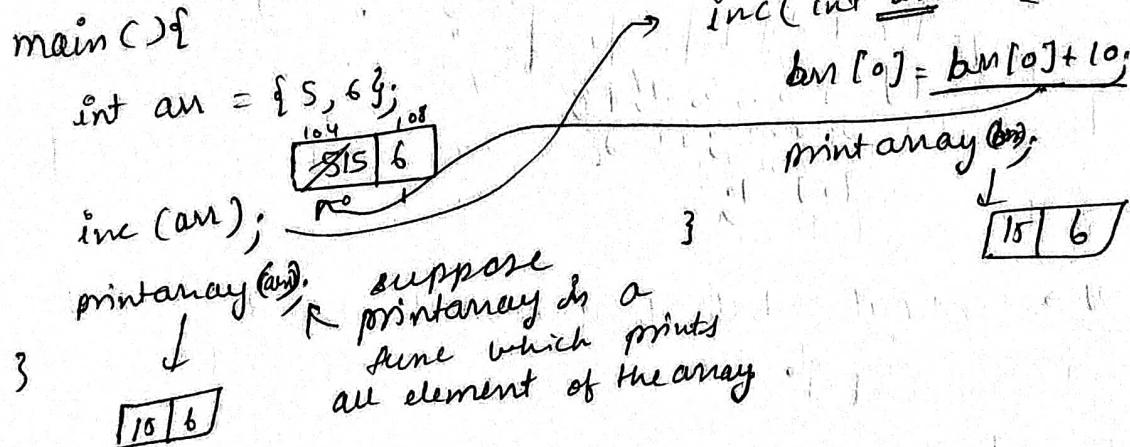
1	0	1	0	1	0	1	0	1	0	1	0
---	---	---	---	---	---	---	---	---	---	---	---

2  $\Rightarrow$  memset, explore it.

② Arrays & functions

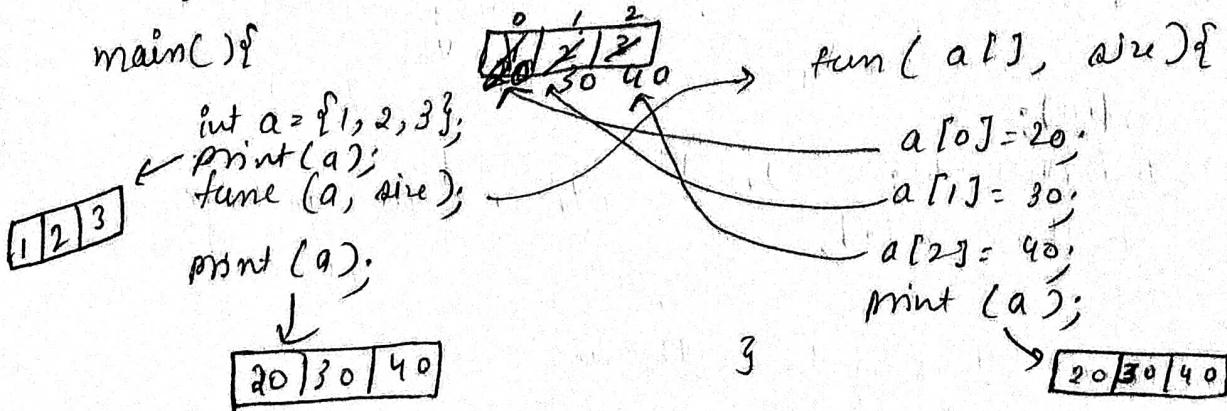


In array  $\Rightarrow$



so this is pass by reference.

Whenever an array is passed in a function it is always passed by reference.



- sizeof → gives the size of an array.  
But what happens when only some elements are there in array (not all).

int arr[10] = { 1, 2 };

in this case sizeof operator does not give us the value filled with element but it gives the size allocated. (In this case 10).

## ⑥ Linear Search in Array

arr [ 2 | 9 | 6 | 7 | 4 | 12 | 15 ]

→ 6 is present in this array or not?  
⇒ Yes, it's present.

How do we know?

→ Go to each element and compare.  
arr[i] == 6 ?

Let's code this -

```
main() {
    int arr[5] = { 1, 3, 5, 7, 8 };
    int size = 5;
    cout << "Enter the key to find";
    int key;
    cin >> key;

    if ( linearSearch (arr, size, key) ) {
        cout << "Found" << endl;
    }
    else {
        cout << "Not Found" << endl;
    }
}

bool linearSearch ( int arr[], int size, int key ) {
    for ( int i = 0; i < size ; i++ ) {
        if ( arr[i] == key )
            return true;
    }
    return false;
}
```

## \* Alternate way - C/o function

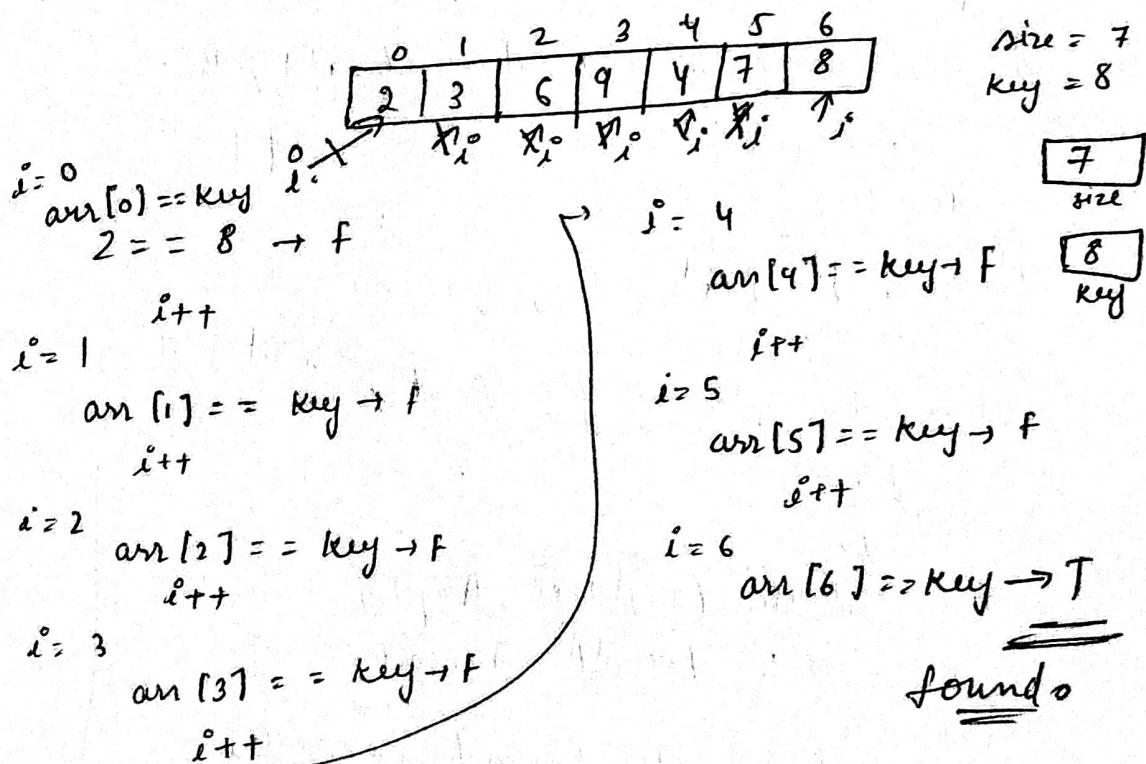
main() {

```
int arr[] = { 1, 2, 3, 4, 5 };
int size = 5;
int key = 3;
bool flag = 0;
for (int i = 0; i < size; i++) {
    if (arr[i] == key) {
        flag = 1;
        break;
    }
}
```

```
if (flag)
    cout << "Present" << endl;
else
    cout << "Not present" << endl;
```

}

flag → 1	(found the element)
flag → 0	(Not found)



① Count 0's and 1's in an array.

```
int arr[] = { 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0 };
```

```
int size = 13;
```

```
int numZero = 0;
```

```
int numOne = 0;
```

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

```

        if (arr[i] == 0)
            numZero++;
        if (arr[i] == 1)
            numOne++;
    }
}

We could have used else here but
in case of array contains other than
0 and 1 than using else is a problem.

cout << "Number of Zeros : " << numZero << endl;
cout << "Number of Ones : " << numOne << endl;

```

### ⑥ Count no. of 2's →

1	2	3	2	4	6	2	9	9	4	1	2
---	---	---	---	---	---	---	---	---	---	---	---

```

int cnt = 0;
for (int i = 0; i < size; i++) {
    if (arr[i] == 2)
        cnt++;
}

```

cout << "No. of 2's : " << cnt << endl;

### ⑦ Maximum no. in an array →

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

maxNum = ~~INT-MIN~~ INT-MIN;

Include a header int size = 7;  
 file, limits.h int maxi = INT-MIN;  
 for (int i = 0; i < size; i++)  
 if (arr[i] > maxi)  
 maxi = arr[i];

cout << "Maximum no. is " << maxi << endl;

Best practice (when you  
you have to find  
 max no., use  
 INT-MIN to  
 initialize and when  
we have you have  
 to find min, use  
 INT-MAX as  
 initialization.)

### ⑧ Minimum no. in an array →

```

int arr[] = {2, 4, 6, 1, 3, 7, 9, 11, 56, 43, 213};
int size = 11;
int mini = INT-MAX;
for (int i = 0; i < size; i++) {
    if (arr[i] < mini)
        mini = arr[i];
}

```

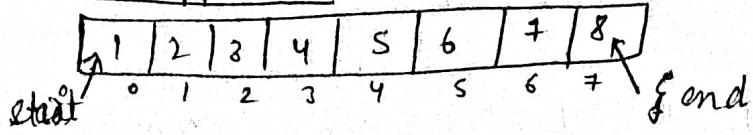
cout << "Minimum no. is : " << mini << endl;

## ① Extreme print in Array -

i/p → {1, 2, 3, 4, 5, 6, 7, 8};

o/p → 1, 8, 2, 7, 3, 6, 4, 5

### Two pointer's approach -



start → starting index → 0

end → last index → (size-1)

→ print arr[start]  
 print arr[end]

Start ++      ← increment start  
 end --      ← decrement end.

loop.

### Rukna kab hai?

→ jaise hi start end se aage nikal payega.

int arr[] = {10, 20, 30, 40, 50, 60, 70, 80};

int size = 8;

int start = 0, end = size - 1;

~~while (true)~~

while (!true) {

    if (start > end)

        break;

    cout << arr[start] << " ";

    cout << arr[end] << " ";

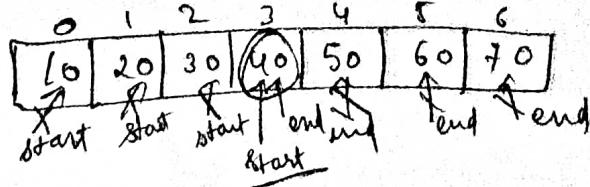
    start++;

    end--;

}

o/p → 10 80 20 70 30 60 40 50

In case of odd element one element will be printed 2 times.



0/p  $\rightarrow$  10 20 30 40 50 60 40 40

because start and end pointing at the same element.

~~so will~~ so we will first add an simple condition.

if (start == end)

cout << arr[start] << " ";

→ or here we can also write

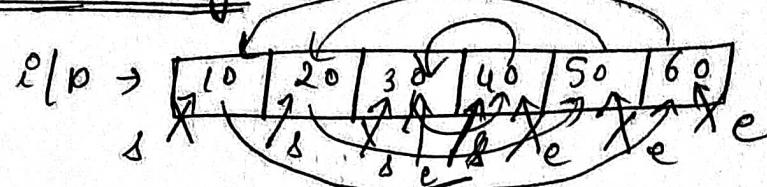
arr[end].

(either of them).

Now while condition

↳ while (start <= end)

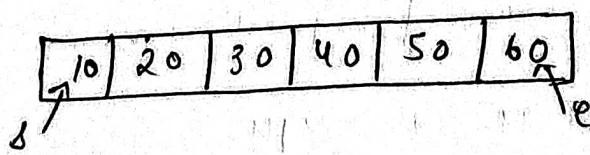
① Reverse an array :-



0/p  $\rightarrow$  [60 | 50 | 40 | 30 | 20 | 10]

⇒ we only swapped extreme points.

we have a predefined function to swap.



s = 0  
e = n - 1

let's code this →

int arr[] = {10, 20, 30, 40, 50, 60};

int size = 6;

int start = 0, end = size - 1;

while (start <= end) {

    swap(arr[start], arr[end]);

    start ++;

    end --;

loop :-  
    ① swap(arr[s], arr[e])  
    ② s++;  
    ③ e--;