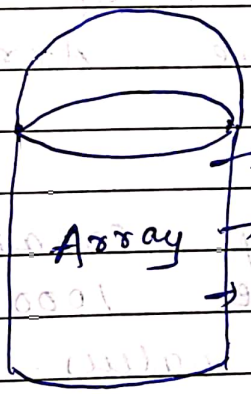
Arrays Level - 1

① What is an Array ?

- List of similar items.
- Collection of elements.
- Data structure.
- continuous memory block.

मम्मी ने बोला लाने के लिये शब्जी

- only Aloo or
- only onion or
- only mattar.



- only int - 1, 2, 3, 4, 5, 6. ✓
- only char - 'a', 'b', 'c' ✓
- only Boolean - True, false. ✓

1, 2, 3, 'a', True. ✗

Here we are demonstrating that- in a array we can only store similar type of data elements. / types.

Why Arrays?

Let's assume we have to add two no. So we create two variables and just add it.

```
int a = 5;
int b = 9;
int c = a+b;
```

assume adding 10 nos we also can do so for taking 10 variables.

But what if we have to store the sum of 1000 values, so here Array comes in picture.

We just simply create an Array of size 1000 and store the values.

```
int arr [1000];
// Array declaration
```

- it simply holds 1000 values of Integer type.

Understanding Arrays.

How normal variables allocates in memory.

```
int a = 5;
```

type 'int'

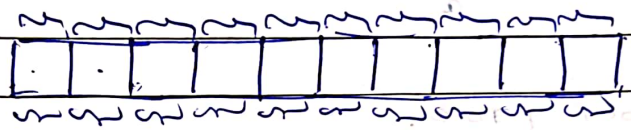
Array Declaration.

```
int arr[10];
```

↳ created to continuous memory blocks.

datatype.

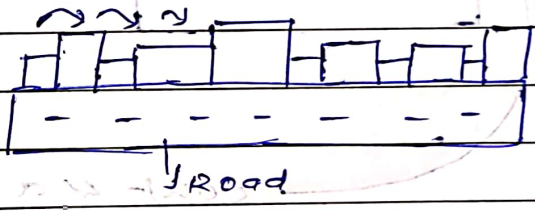
(array)



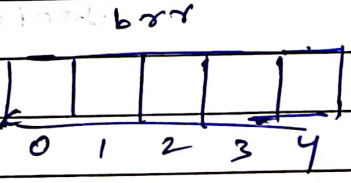
HL → 0 1 2 3 4 5 6 7 8 9

(Memory locations)

→ placed in contiguous memory.




```
int brr[5]
```



Base Address.

Static Array

(size is fixed to 5)

Memory Allocated.

↳ int brr[5]

↳ 4 bytes × 5

So, 5 × 4

it holds 20 bytes.

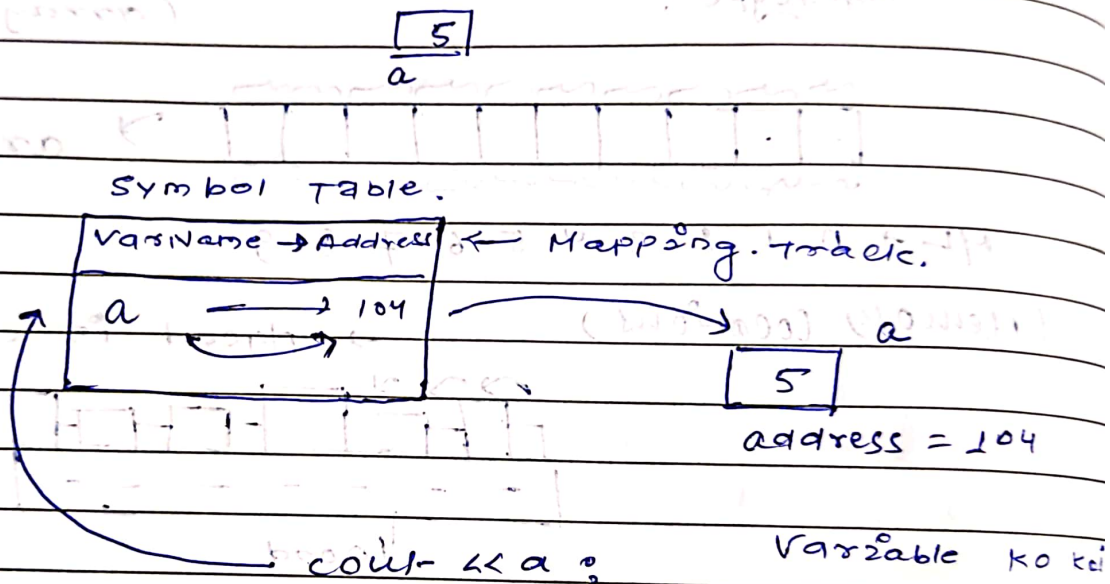
To check.

(i) address
↳ address of Operator — &

(ii). size
↳ size of function

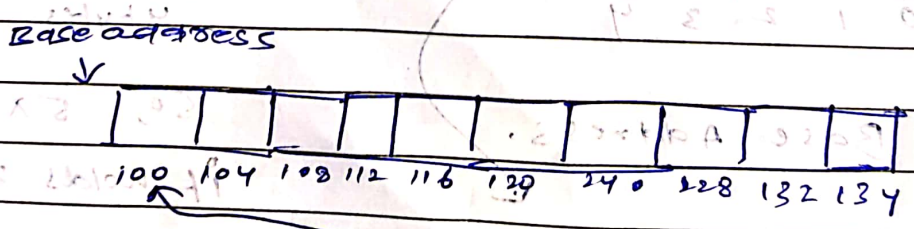
Address Mapping with variable name

int a = 5;



Variable ko to Nahi jaanta, Sb address ko jaante hai..!

int marks [10]



Array initialisation.

```
int arr[] = {1, 3, 2, 6, 8}
int brr[5] = {1, 2, 9, 5, 6}
```

what if.

```
int crr[5] = {1, 2}
```

1	2	0	0	0

→ assigned 0 if no value is given.

```
int drr[2] = {2, 4, 4, 6, 4}
```

2	4

→ error

getting input -->

```
int n;
cin >> n; // user input.
int arr[n];
```

operating system allocates memory for every program.

So if the user entered very large no. which extends the memory

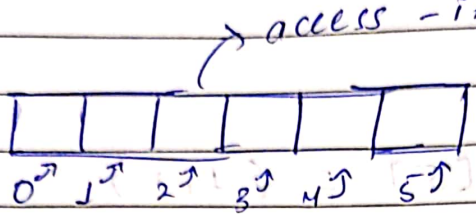
→ So, this is then the program a bad practice. can be crash

To prevent this - we use dynamic arrays. b!!

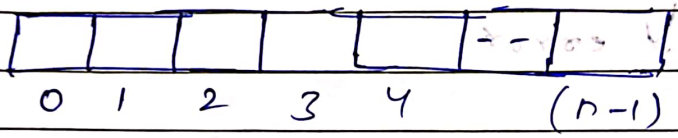
Indexing in Array.

0-based indexing

→ way of accessing any element.

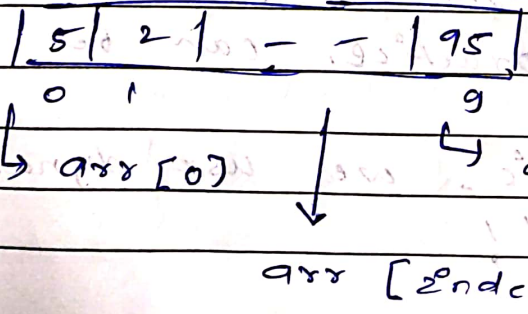


Array — n size.



1-based indexing.	*	*	*	*	↘ n.
	↳ start.				
0-based indexing	0	1	2	3	
	↳ start.			↘ n-1	

int arr [10]



int arr [5] = {3, 5, 8, 9, 12}

0	3	→ arr [0] = 3
1	5	→ arr [1] = 5
2	8	→ arr [2] = 8
3	9	→ arr [3] = 9
4	12	→ arr [4] = 12.

Program via loops. (printing)

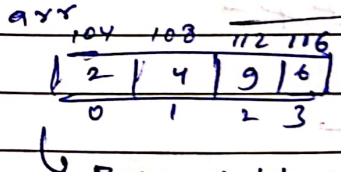
```
int arr [5] = {1,2,4,6,8}        OR  1 2 4 6 8
int n = 5
for (int i = 0; i < n; i++) {
        cout << arr [i];
}
```

Taking input in array.

```
int arr [5];
for (int i = 0; i < 5; i++){
        cin >> arr [0]
}
```

Formula for finding index in any Array:

arr[i] ⇒ Value at (Base Address + (datatype size * index))



↳ Base Address → 104

arr [2] = 6

arr [2] → Value at (104 + (4 * 2))

= 112

Arrays and Functions

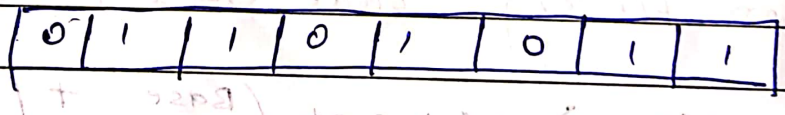
→ syntax.

```
Solve ( int arr[], int size){

    // code.

}

int main (){
        int arr [5] ;
        int size = 5 ;
        Solve (arr, size) ;
```

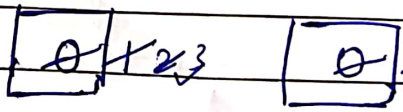
We can't use sizeof function. cuz...

size → we want only this not full array size.

→ count 0's and 1's in an Array.



or



zero count → 3

one count → 5

→ Min no. in array

Best practice

Header file
↳ limits.h
int maximum value = INT_MA
minimum value = INT_MIN - x

142 no. → min Ans
↓
INT_MAX

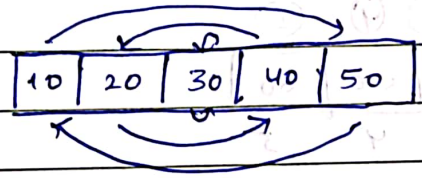
Max no → max ans
↓
INT_MAX

↳ here we apply the condition.

```
if(arr[i] < minAns)
    minAns = arr[i]
```

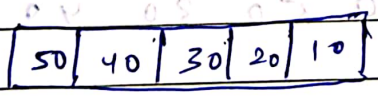
→ Reverse an array.

I/P



:: Builtin function
swap(a,b)

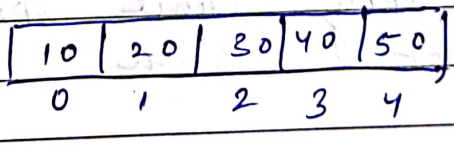
O/P



Dry Run

Left = 0

Right = ~~5~~ 4




```
swap(left element, right element)
left ++;
Right --;
```

while loop content

```
initialization
while (condⁿ)
{
    ____
    ____
    ____     } body / logic.
    ____

    // updation
}
```

→ Extreme print in an array.

①	③	⑤	⑥	④	②
10	20	30	40	50	60
0	1	2	3	4	5

O/P 10 60 20 50 30 40

→ here we apply same approach and print both elements.

But for condition
left == Right
we only print arr[left] or arr[Right]