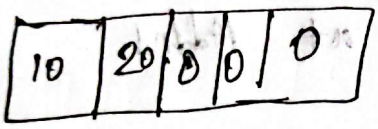
Let's Array

- It's a type of container that store same type of data.

Initialization:

```
int arr[5] = {10, 20}
```



⇝ It will get 0 if you won't give values according the initialization

- if you don't give any value there will be garbage values will be stored —

Indexing:

1 → n
0 → n-1

if array size = n
then index = [0 → n-1].

Starting or Base Address → 104

④ if you try to access something out of the index in a array it will give segmentation fault on it will show garbage result in some compiler it will give list index out of range(Python).

⑤ arr[i] → values Present at [Base Addr.. + sizeof i].

⑥ arr[i] → i[arr].

*(arr+i) *(i+arr) ⑧ → values Present at Addrs.

First Algorithm solving

Linear Search

⑧ In Linear search you will go one by one and check if exists. If exists the true if not the false.

① If you want to find max number you need to initilize the ans → INT_MIN

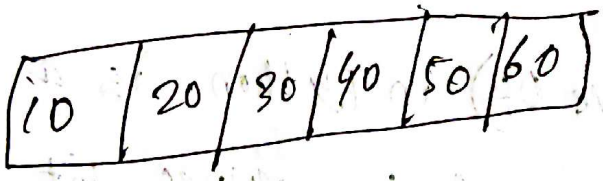
② If you want to find min number you need to initialize the ans → INT_MAX.

Range

$-2^{31} \rightarrow 2^{31}-1$

INT_MIN INT_MAX

2pointer technique!



output → 10 60 20 50 30 40 → this is called extreme Print

i = 0, j = n-1

cout << i
i++
cout << j
j--

if j < i
return 0
} while loop

Homework!

① Swap (swap two numbers) (a → 5, b → 4)

② Reverse array (Reverse an array)

* Array is passed by refenence not value.

XOR Operator!

* XOR (~~exclu~~ exclusive OR) in python is a bitwise operation that compares two values bit by bit. It's represented by the caret symbol (^)

How XOR Works!

① if the corresponding bits are different (one is 0, one is 1), the result is 1.

- If the corresponding bits are different
  same - (both 0 or both 1), the result is 0.

XOR in Different Data types!

- Integers.
- Bool (True is 1 and false is 0)
- Lists (applied element by element)
- Bytes Object.

Why XOR is useful!

- It can toggle (flip) bits! if you XOR any bit with 1, it flips (0->1 or 1->0)
- It's used in cryptography for encryption.
- It is used for error checking and data validation.
- Two XOR operations with the same value cancel out each other out.

Swapping

- Using temporary variable can cause extra memory issue
- Tuple unpacking is the most efficient method for swapping two variable.

Reversing an Array!

Approaches!

- existing reverse method
- using list slicing
- using for loop

- Avoid using list comprehension on loops if is not memory efficient.