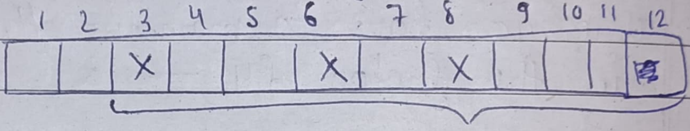
Day-22

Introduction to Array

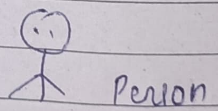
* Let's take an example, if a person wants to store ~~a~~ their things and a person have available space , just like this's —



Now owner gives the person 6 no. room. After some days, the person requires two more room then owner give 3 & 8.

After some days , the person wants 10 room but in continuous manner. So, the owner gives 3 to 12 number rooms to the person.

So,



{3, 12}

Same , in this way, array works. Now, the person easily access their things.

Let's take ~~an~~ a different example —
If you want to add three no. then —
a, b, c → cout << a+b+c;
If you have 10 numbers —
a, b, c, d, e, f, g, h, i, j

Date

Page

⇒ If you have 100 or more than 100 then - it's make a big issue.

⇒ So here, naming makes a big problem.

⇒ So, we can name them as a0, a1, a2, ..., a999.

⇒ Writing this also takes a long time.

⇒ So we want this —

```
for (i=0; i< 1000; i++)
    cin >> a[i];
```

→ This is array

⇒ Array definition:

⇒ It store same type of data type.

⇒ It store at contigeus locations.

Syntax:

→ size of array

int a[1000];

data type → array name

⇒ int a[5] = {6, 5, 4, 3, 2}

	0	1	2	3	4
a	6	5	4	3	2
	a[0]	a[1]	a[2]	a[3]	a[4]

→ Indexing

⇒ int name[] = {3, 8, 2, 6}

	0	1	2	3
name	3	8	2	6

=)
```
int arr[10];
for( int i= 0; i< 10; i++){
        cin >> arr[i];
}
```

=)
```
int a[5] = {8,4}
```

0	1	2	3	4
8	4			

=)
```
int arr[5] = {0};
```

0	1	2	3	4
0	0	0	0	0

It is only valid for '0';

=) To print the elements —

```
int arr[5] = { 1,2,3,4,5};
for (int i= 0; i<5; i++)
      cout << arr[i];
```

=) We require address for everything.

=) we know that,

1 bit = 1 transistor

8 bit = 1 byte

2^{10} byte = 1 kB

=) Our memory is byte addresable memory that means we are giving a address to every byte not every bit.

Date ____________ Page ________

=> int arr[5] ;

	0	1	2	3	4
500 ←	8	7	17	15	18
		504	508	512	516

=> So, if the address of arr is 500 that means arr[0] is at 500 address —

=> Formula to find any element address —

address of index = arr + index + size of data type

=> Why we ~~have~~ use 0-based indexing ?

=> Because it easier to calculate the address when we have 0-based indexing.

=> char name [5];

	0	1	2	3	4
name	'a'	'b'	'c'	'd'	'e'
	100	101	102	103	104

char = 1 Byte

=> 32-bit processor
4GB RAM
⇓
2^{32} byte RAM

64-bit processor
8 GB RAM
and so on
⇓
2^{33} byte RAM

=> All DSA part executed in the RAM.

Page

Date

=> If you want to find min element in the array then ⟶

=> int arr[5] = { 4, 6, 11, 2, 8 };

0	1	2	3	4
4	6	11	2	8

```
int ans = INT_MAX;          for(i=0; i<5; i++)
                            { if (arr[i] < ans)
                                 ans = arr[i]
                            }
```

=> For max element;

int ans = INT_MIN;

(Just change the above syntax with below)

arr[i] > ans;