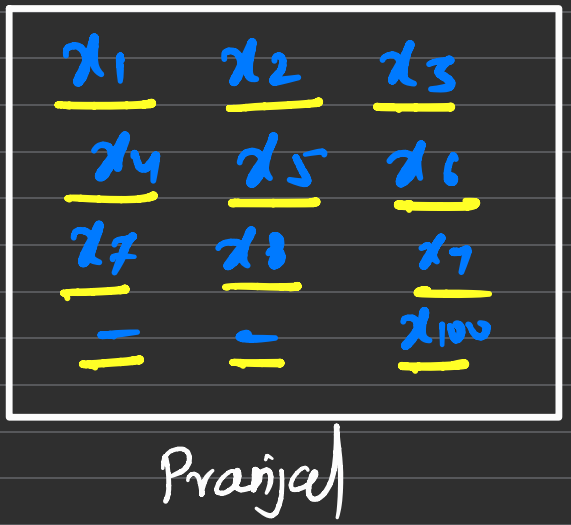
"खुल जाएंगे सभी रास्ते, तू रुकावटों से लड़ तो सही,
सब होगा हासिल, तू अपनी ज़िद पर अड़ तो सही।"

Why we need Arrays



5

$x_1 = 10$

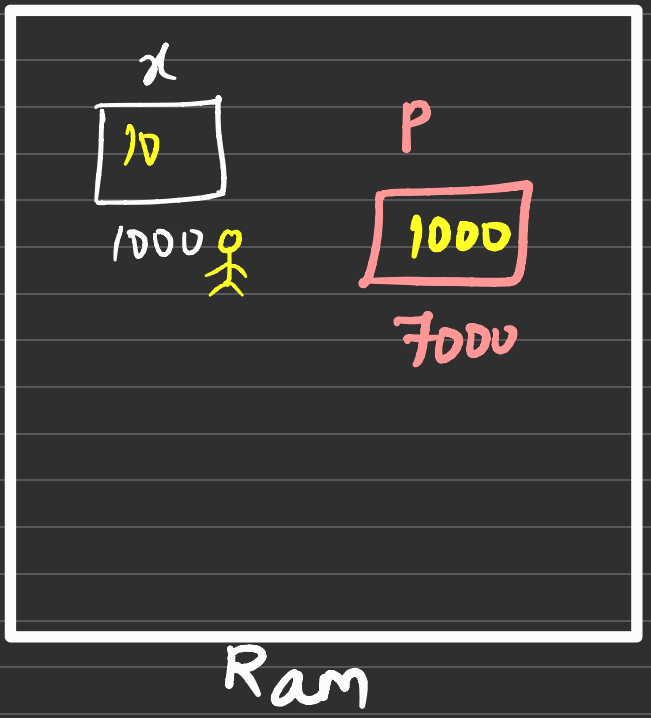
$x_2 = 70$

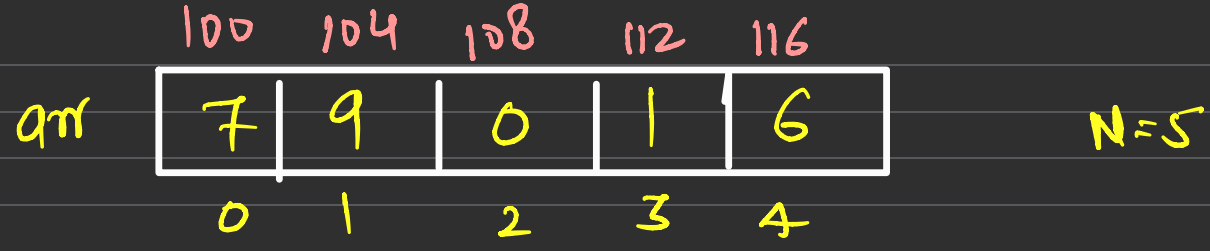
$x_3 = 90$ ——— $x_{100} = 60$

What is Array

Collection of elements of same data type in contigous memory

```
int x = 10  ⟶ 4B
int *p = &x
print (x) ⟶ 10
print (p) ⟶ 1000
print (*p) ⟶ 10
```

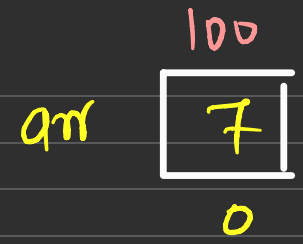




Syntax: to declare

datatype name [size]

int arr [5]



N=5

cout << arr(3) → 1

Compiler

arr(3)

* (arr + 3)

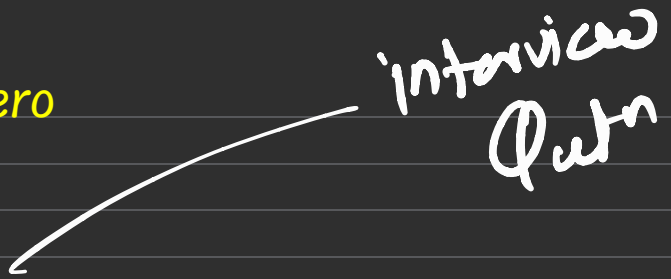
* (100 + 3)

* (100 + 3x4)

* (100 + 12)

* (112)

Why array index start from zero



Operations on Array

declare

$N = 5 \longrightarrow (0 \ldots N-1) \Rightarrow (0 \ldots 4)$

```
int arr [N]
```

```
cin >> arr(0)
cin >> arr(1)
cin >> arr(2)
   ⋮
cin >> arr(4)
```

$i \leq (N-1)$

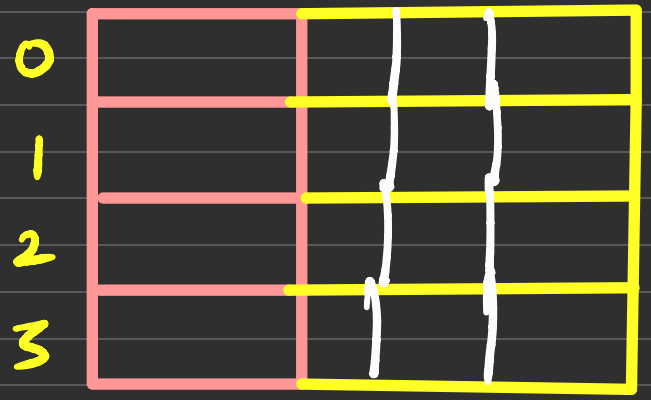
```
for (i=0; i<N; i++)
    cin >> arr(i)

for (i=0; i<N; i++)
    cout << arr(i)
```

Types of Arrays

- 1D Array

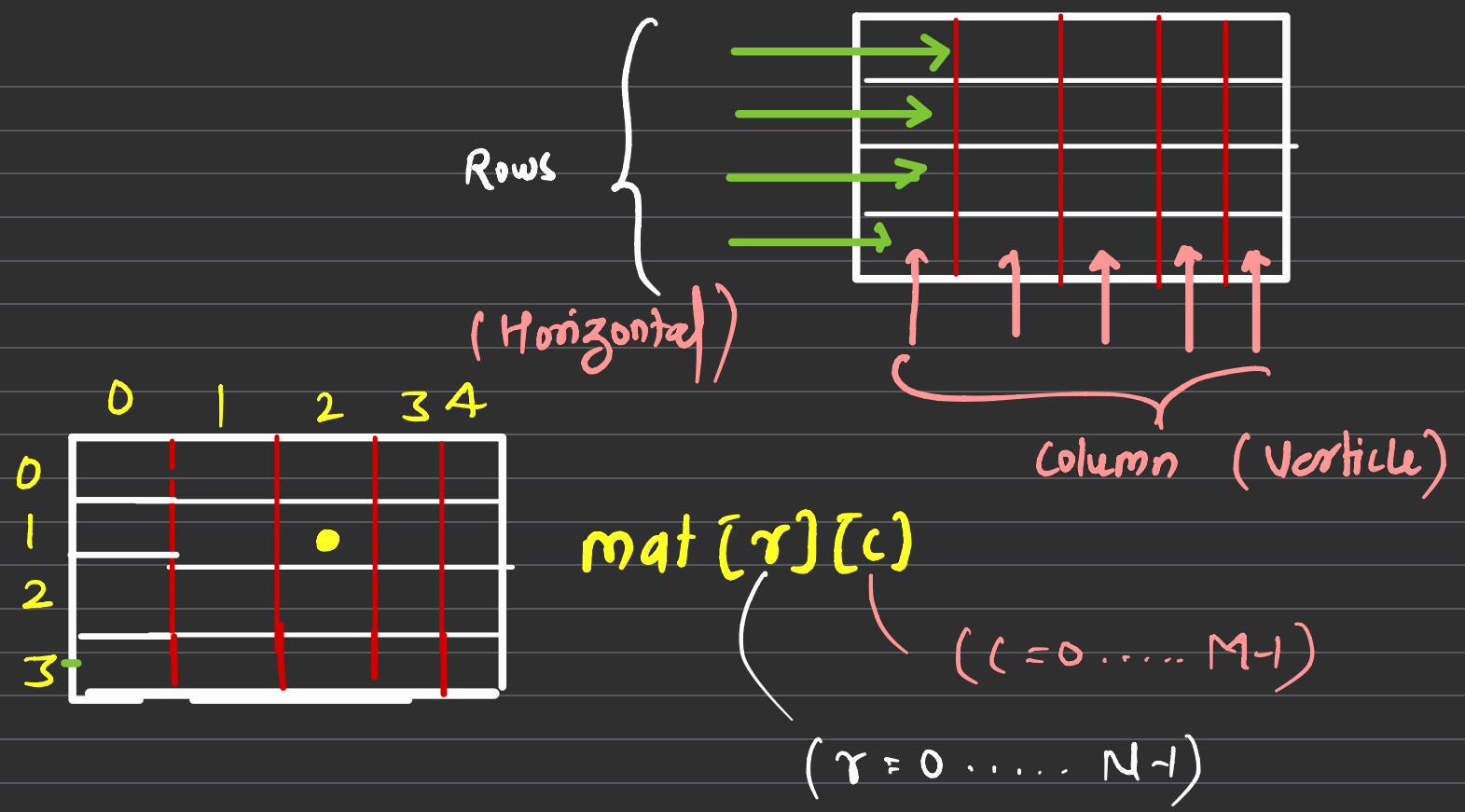
  Syntax : dataType name [size]

- 2D Array : Array of Arrays



Syntax :

dataType name [N] [N]



```
for (r=0; r<N; r++)
{
    for (c=0; c<M; c++)
    {
        cin>> mat[r][c]
    }
}
```

r = 0 1 2

c = 0

N = 3 M = 3

int mat[N][M]

r \ c	0	1	2
0	7	9	10
1	12	13	23
2	10	11	9

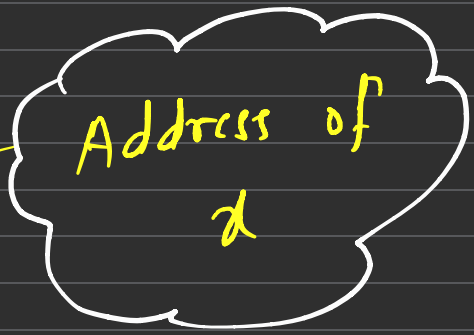
Pointers :

It is a special variable which store address of other variable. of Same data Type

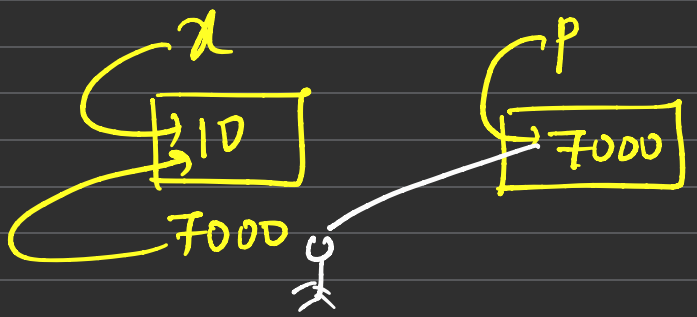
Syntax :

```
dataType *name
```

```
int x = 10
int *p = &x
```



```
int x = 10
int *p = &x
print (x)    10
print (p)    7000
print (*p)   10
```



int x = 4B / 2B
32bit 16 bit

pointer size = fixed = 4B

Types of Pointer

- void pointer
- Dangling pointer
- Null pointer
- Wild pointer

void pointer :

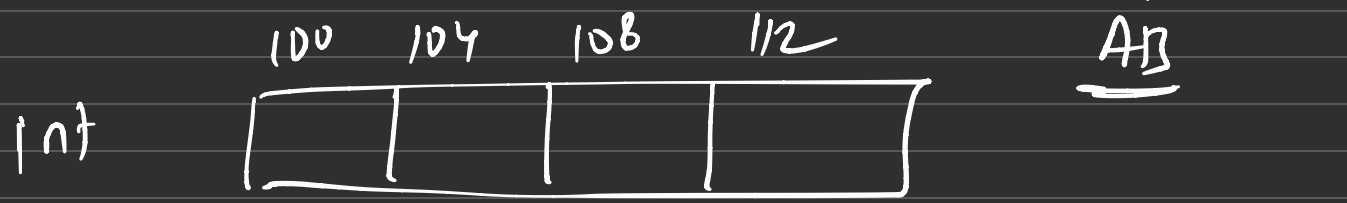
It can store address of any data

Syntax :

```
void *p
```

Eg:

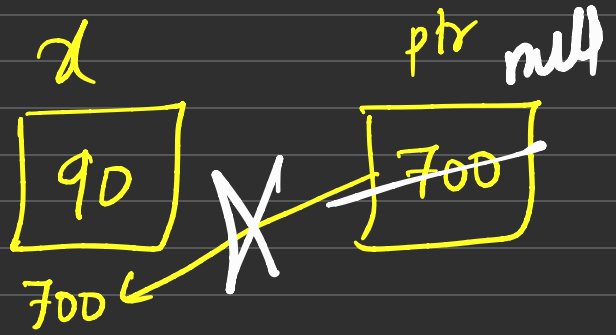
```
int x = 90          float y
void *p = &x        p = &y
```



NULL pointer:

pointer pointing to NULL

```
int x = 90
int *ptr = &x
ptr = null
```

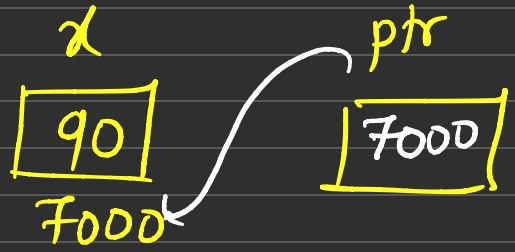


Wild pointer:

pointer pointing to nothing not even a null

```
int x;
print (x) ⟶ Garbage value
```

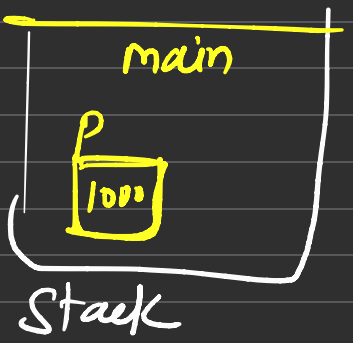
```
int x = 90
int *ptr
```



Dangling Pointer.

pointer pointing to memory location which is deleted

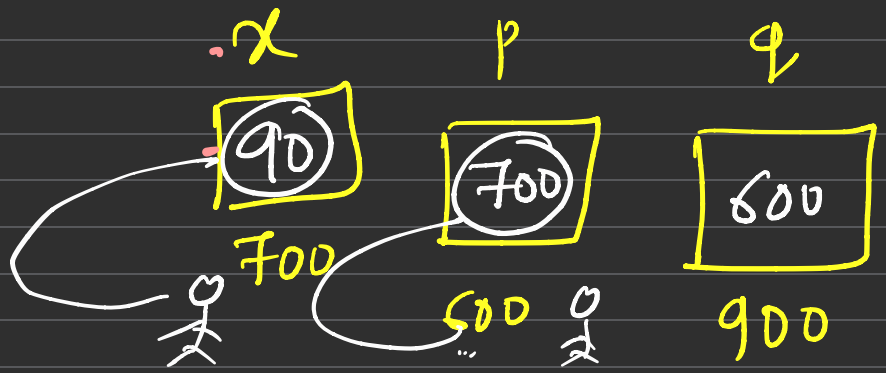
```
int * fun()
{
    int x = 90
    return &x
}

main()
{
    int *p = fun()
}
```



Pointer of Pointer

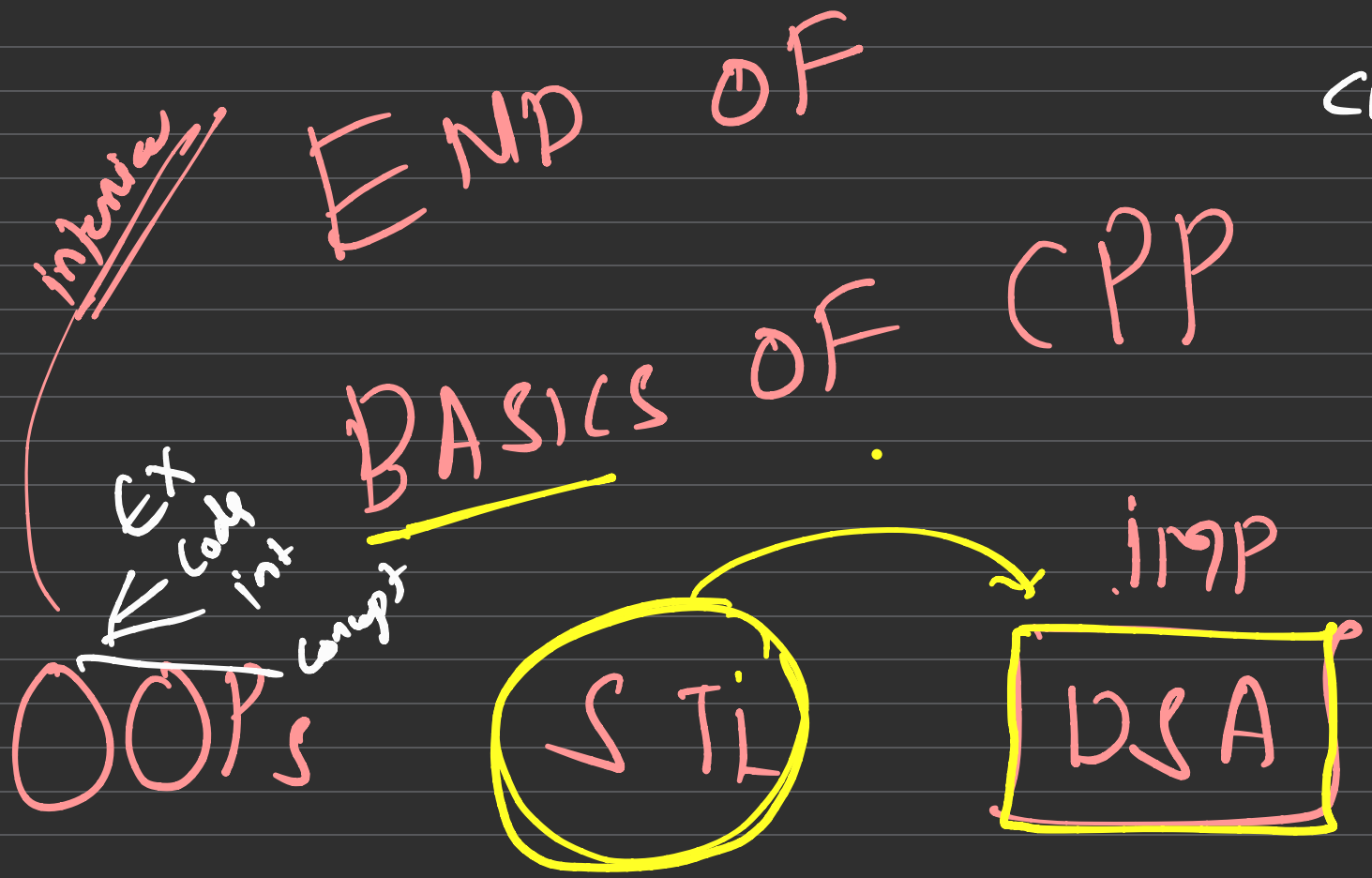
```
int x = 90
int *p = &x
int **q = &p
```

```
int **q = &p
```




```
print (x) → 90
print (p) → 700
print (q) → 600
```

```
print (*p) → 90
print (*q) → 700
print (**q) → 90
```



Class

object

Access Specifier

this pointer

Encapsuln

Poly

Inheritance (5)

Abstraction

Constructor

Real Life Eg

code ⟶ IMP Interview quest^n

concept