CS & IT ENGINEERING

Programming in C

Arrays and Pointers

Lec- 05

By- Pankaj Sharma sir

- Arrays and Pointers

Complex declaration

Identifier

1) ( ) functions
2) [ ] array

] 1 L to R

3.) Identifier
4.) *

] 2 R to L

5.) Data type] 3

1.)

③ ① ②

int (*p)[4];

P is a pointer to array of 4 integer.

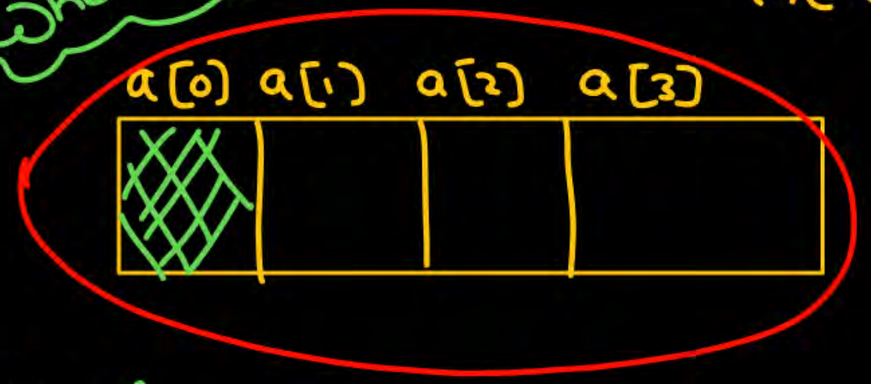
int a[4];

P = &a;

① &a → add. of whole array

② a → add of 1st element. i.e &a[0]

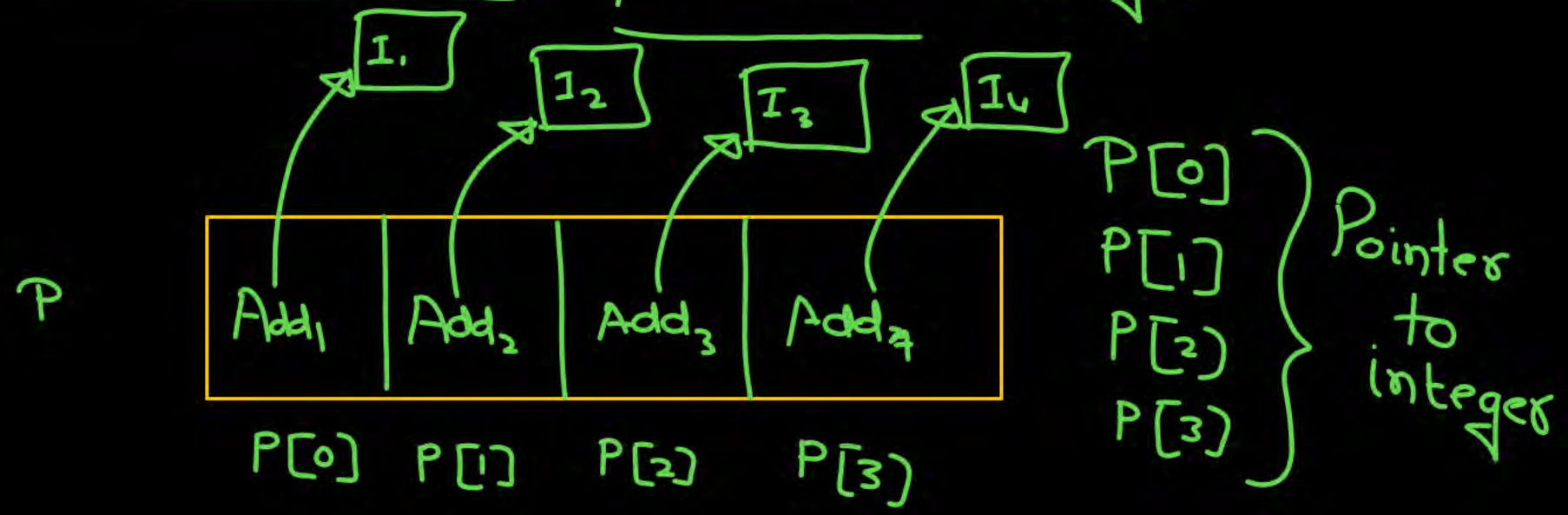
int a[4];



(2) ③ ② ①

int *(P[4])

P is an array of 4 pointer to integer



3) ③ int ① (*P) ② (int);

(int); → function

लेता देता है

Arguments return value

P is a pointer to function that takes one integer argument and returns a integer value.

```
void main(){
    int a[4] = {10,20,30,40};
    ++a;
```

✗ Invalid

```
void main(){
    int a[4] = {10,20,30,40};
    fun(a); //a → array name : &a[0]
    printf("%d %d",a[0],a[1]);
}
```

fun(&a[0])

address of int

```
void fun( int *p )
{

}
```

++ * R to L

```
void fun(int*);
void main(){

 int a[4]= {10,20,30,40};

 fun(a);

 printf("%d %d",a[0],a[1]);
}
```

10 21

a[0]	a[1]	a[2]	a[3]
10	~~20~~ 21	30	40
100	104	108	112

&a[1]

~~&a[0]~~

p

```
void fun(int *p)
{
 ① ++p;
 ② ++*p;
}
```

P = P+1
= &a[0]+1
= &a[1]

++(*p)

*p = *p+1;

20+1

*p = 21

++, * same

$\boxed{\text{R to L}}$

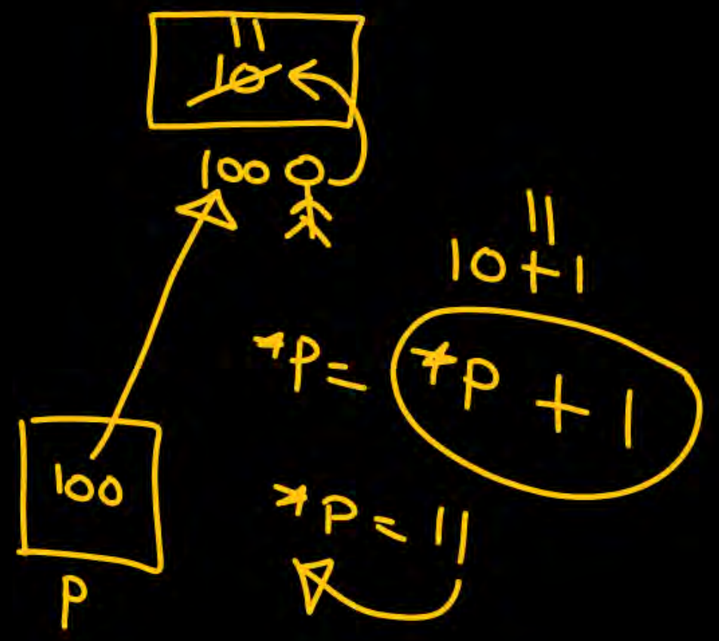
(i) ++*p;

(ii) ++(*p);

$\boxed{*p = *p + 1}$

++x

$\Rightarrow \boxed{x = x + 1}$



```
void fun(int*)
void main(){
    int a[4]= {10,20,30,40};
    fun(a);
    printf("%d %d",a[0],a[1]);
    }
```

a[0]	a[1]	a[2]	a[3]
10	20	30	40
100	104	108	112

p
100

```
void fun(int *p)
{
    *p++;
}
```

(i) *(p++)

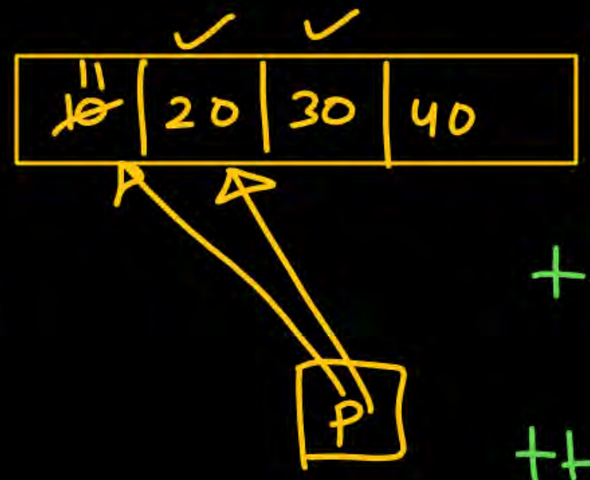
Post inc

(i) *p

(ii) p=p+1

~~*p;~~

p++;




```
void fun(int*);

void main(){

    int a[4] = {10,20,30,40};

    fun(a);

    printf("%d %d",a[1],a[2]);
    }
```

++ *p++

++(*(p++))

Post inc.

(i) ++(*p)

(ii) p=p+1

*p=*p+1

```
void fun(int *p)
{
    ++*p++;
}
```

Result of inc./dec $\Rightarrow$ is treated as constant

++(++p);

Invalid

++ constant

Invalid

++5

++(Invalid)

Whenever u pass an array

⇓

Call by ref.

call by value

```
void main(){
    int a[4] = {10,20,30,40};
    fun(a);
    ==
}
```

$\rightarrow$ &a[0]

```
void fun(int *p)
{

}
```

```
void fun(int a[])
{

}
```

int *a

Pointer

```
void fun(int a[4])
{

}
```

```
void main(){

    int a[4] = {10,20,30,40};

    sum(a, 4);

}
```

a[0]	a[1]	a[2]	a[30]
10	20	30	40

100

100
p

p+1

nothing
Empty

```
void sum(int *p, int n)
{
    int s = 0;
    for(i=0;i<n;i++)
        s=s+p[i];
    printf("%d",s);
}
```

*(p+0) ⇒ 10 → p[0]

*(p+1) ⇒ 20 → p[1]

*(p+2) ⇒ 30 → p[2]

*(p+3) ⇒ 40 → p[3]

10	20	~~30~~ 100	40	50	60
100	104	108	112	116	120

P

```
void main(){
    int a[2][3]={10,20,30,40,50,60};
    fun(a[0]);
    printf("%d %d %d",a[0][0],a[0][1],
                      a[0][2]);
}
```

&a[0][0]

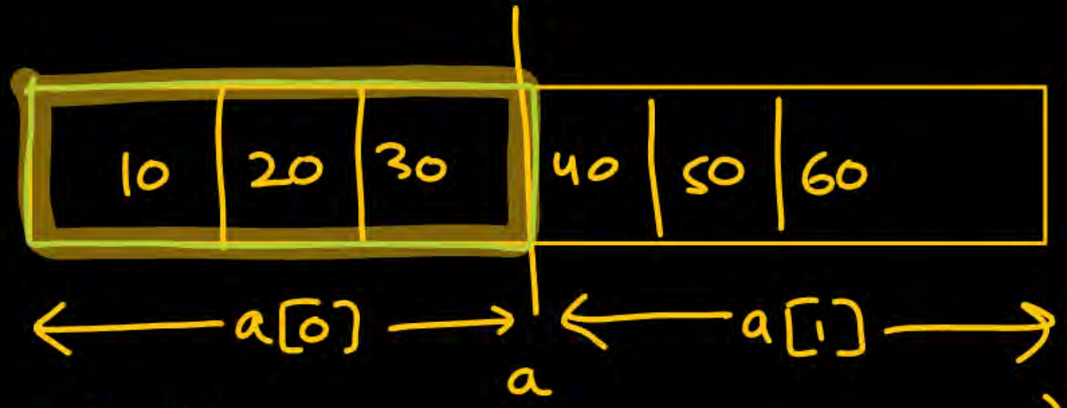
```
void fun(int *p)
{
    ++p ;
    *p++;
    *++p ;
    p-- ;
    *p = 100;
}
```

(i) *p ✗
(ii) ~~*p++~~ p=p+1

(i) p= p+1
(ii) *p ✗

add + value

Pointer +



add of int

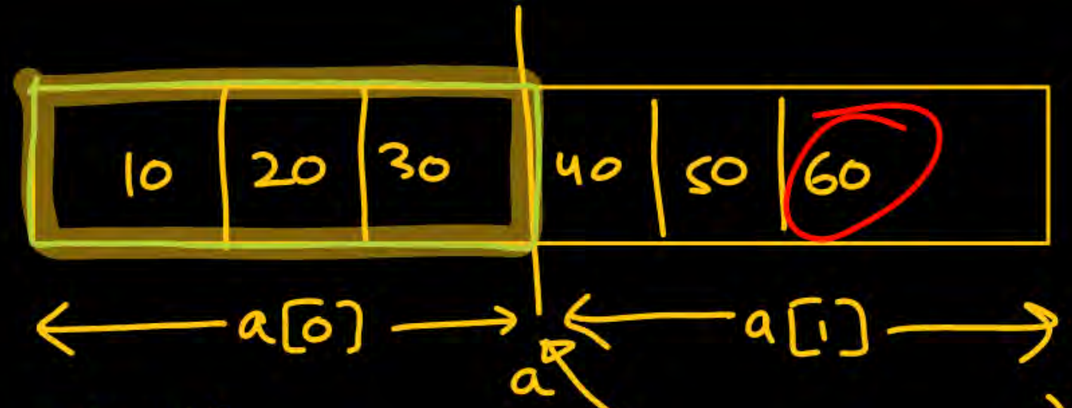
int *p

```
void main(){
    int a[2][3]={10,20,30,40,50,60};
    fun(a);
    printf("%d %d %d",a[0][0],a[0][1],a[0][2]);
}
```

a ? ⇒ &a[0] ⇒ address of an array of 3 integer

```
void fun (int (*p)[3])
{
}
```

Valid




```
void main(){
    int a[2][3]={10,20,30,40,50,60};
    fun(a);
    printf("%d%d%d",a[0][0],a[0][1],a[0][2]);
}
```

p++;

```
void fun(int(*p)[3])
{
    ++p;
    printf("%d",*((*p+1)+1));
}
```

p = &a[1]

a[1][2]

*p ⇒ &a[1]

*p ⇒ &a[1][0]

*p+1 ⇒ &a[1][1]

*p+2 ⇒ &a[1][2]

20byte
a
10
p
int a=10;
a+1;
≡
a++;
a=a+1
p+1
vs
++p;
update
*p
int *p;
p
char *x;
x
float *y;
y
p++ → 4 byte
x++ → 1 byte

$256 + 44$

$256 + 32 + 8 + 4$

```
int a = 300;
char *p;
```

MSB			LSB
00000000	00000000	00000001	00101100
1003	1002	1001	1000

```
P = (char*)&a;
```

typecasting

```
printf("%d", *p);
```

① little Endane: LSB

p

44

Q

```
int a = 160;
char *p;
p = (char*)&a ;
printf("%d", *p);
```

$\rightarrow 128+32$

-ve numbers

00000000	00000000	00000000	10100000

(signed) char

char → integer value

1 0 1 0 0 0 0 0

$2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

$= -(2^6+2^4+2^3+2^2+2^1+2^0)$

$= -(64+16+8+4+2+1)-1$

$= -96$


THANK YOU GW
SOLDIERS !