CS & IT ENGINEERING

Programming in C

Arrays and Pointers

Lec- 05

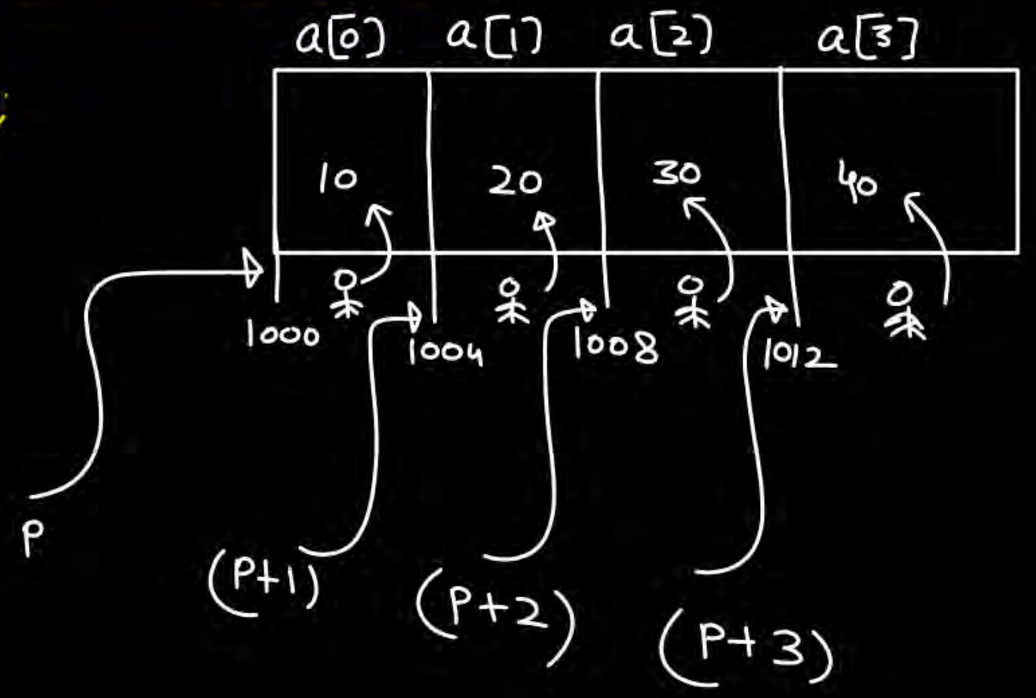
By- Pankaj Sharma sir

- Arrays and Pointers

Arrays and Pointers

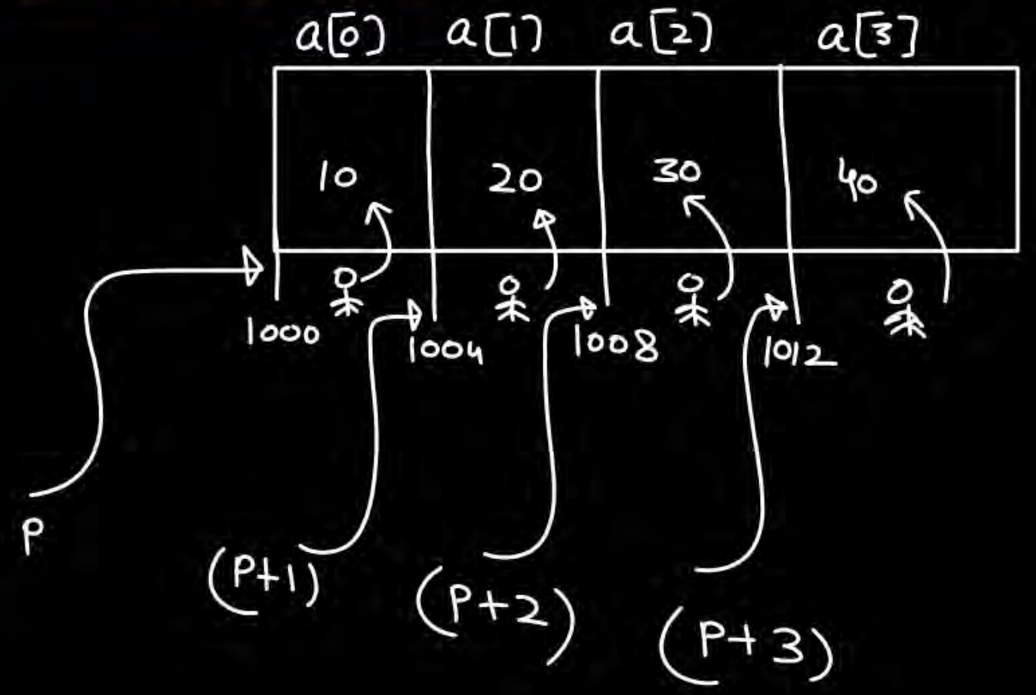
```
int a[4] = {10,20,30,40};
int *p ;
p = &a[0];
```

```
pf("%d", *(p+0));   10
pf("%d", *(p+1));   20
pf("%d", *(p+2));   30
pf("%d", *(p+3));   40
```



Arrays and pointers

```
int a[4] = {10,20,30,40};
int *p ;
P = &a[0]; OR P = a ;
pf("%d", *(P+0));   10   P[0]
pf("%d", *(P+1));   20   P[1]
pf("%d", *(P+2));   30   P[2]
pf("%d", *(P+3));   40   P[3]
```



Arrays and pointers

```
int a[4] = {10,20,30,40};
int *p;
p = &a[0]; OR p = a;
```

{whole array}

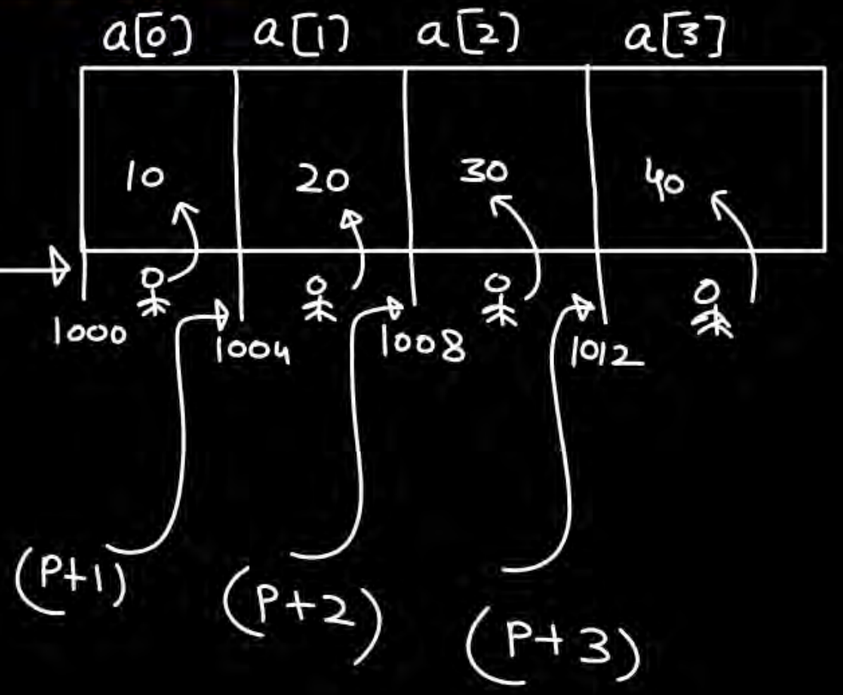
P = &a

P = (int*)&a;

```
pf("%d", p[0]);
pf("%d", p[1]);
pf("%d", p[2]);
pf("%d", p[3]);
```

p

```
for(i=0; i<4; i++)
    pf("%d", p[i]);
```



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

a++;
++a;
--a;
a--;
```

invalid

```
int a[4] = {10, 20, 30, 40};
int *p;
p = &a[0];

p++;
++p;
--p;
p--;
```

$\rightarrow$ $p = p + 1$

```
int a[4] = {10,20,30,40};
int *p;
p = &a[0];
```

p+1;

we are not updating content of p

```
void main()
{
17.38;
20;
}
```

P = P+1

we are updating P

1.

```
int a[4] = {10,20,30,40};
int *p;
p = &a[0];
++*p;
++p;
printf("%d", *p);
```

↓

20

a[0]	a[1]	a[2]	a[3]
~~10~~ 11	20	30	40
100	104	108	112

p

p

← ++*p

++(*p);

(i) *p = *p+1 ⇒ a[0] = a[0]+1

×(ii) *p

meaningless

p = &a[0]

*p = *&a[0] = a[0]

1.

```
int a[4] = {10,20,30,40};
int *p;
p = &a[0];
++*p;
++p;
printf("%d", *p++)
```

$++(*p)$;

use

$*p$ 20

$p = p+1$

$*(p++)$

$*p$

$p = p+1$

a[0]	a[1]	a[2]	a[3]
~~10~~ 11	20	30	40
100	104	108	112

p

p

$\leftarrow$ $++*p$

$p = \&a[0]$

$*p = *\&a[0] = a[0]$

(i) $*p = *p+1 \Rightarrow a[0] = a[0]+1$

x(ii) $*p$ meaningless

```
int a[4] = {10, 20, 30, 40};
int *p;
p = &a[0];
pf("%d", *++p);
    ++*p;
    ++p;
pf("%d", *p++);
```

20 ✓

a[0]	a[1]	a[2]	a[3]
10	21 ~~20~~	30	40

p

(i) *(++p) → Pre-in

a) p = p+1

b) *p

(ii) ++(*p)

Pre-inc ⇒ *p

*p = *p+1

Use *p

(iii) ++p

```
int a[4] = {10, 20, 30, 40};
int *p;
p = &a[0];
pf("%d", *++p);   20
    ++*p;   ✓
    ++p;
pf("%d", *p++);   30
```

a[0]	a[1]	a[2]	a[3]
10	21 ~~20~~	30	40

p

(iv) *(p++)

Post-inc

a) *p ✓ ⇒ 30

After printing b) p=p+1

Complex Declarations

1.) int (*P)[3]

2.) int *(P[3])

Declaration

1) ( ) functions

2) [ ] Array

3) Identifier

4) *

5) Data type

① L to R

② R to L

③

int *p;

(i) int is a pointer p ✗

(ii) Pointer is a int p ✗

(iii) p is a pointer to integer ✓

Identifier

[4]

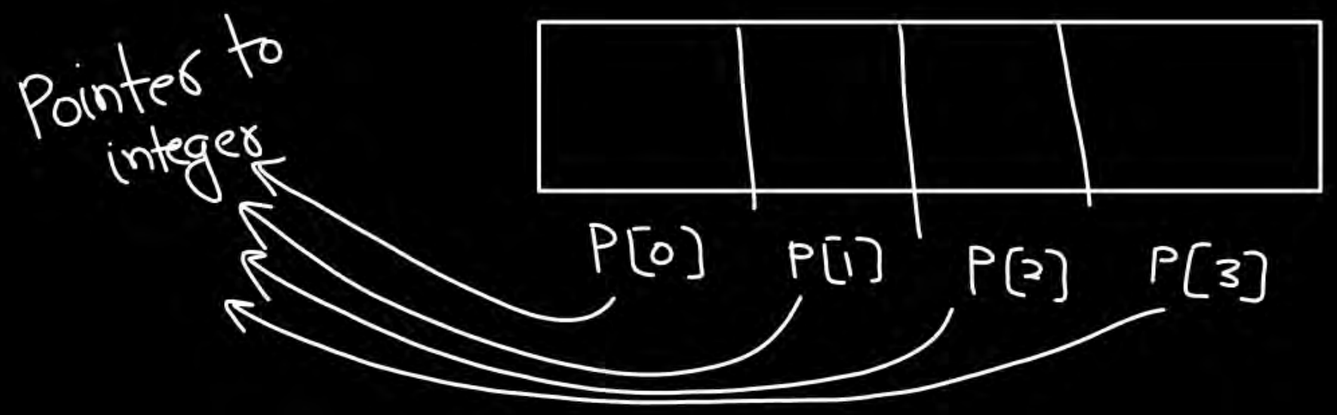
1. int *(P[4]);

P is an array of 4

Pointer to

integer

P is an array of 4 Pointer to integer.



2.

$$\text{int } (\overset{\text{I}}{*}p)\overset{\text{II}}{[4]};$$

p is a pointer to array of 4 integer

```
int (*p)[4];
int a[4] = {1,2,3,4};
```

&a

&a[0]

p = &a;

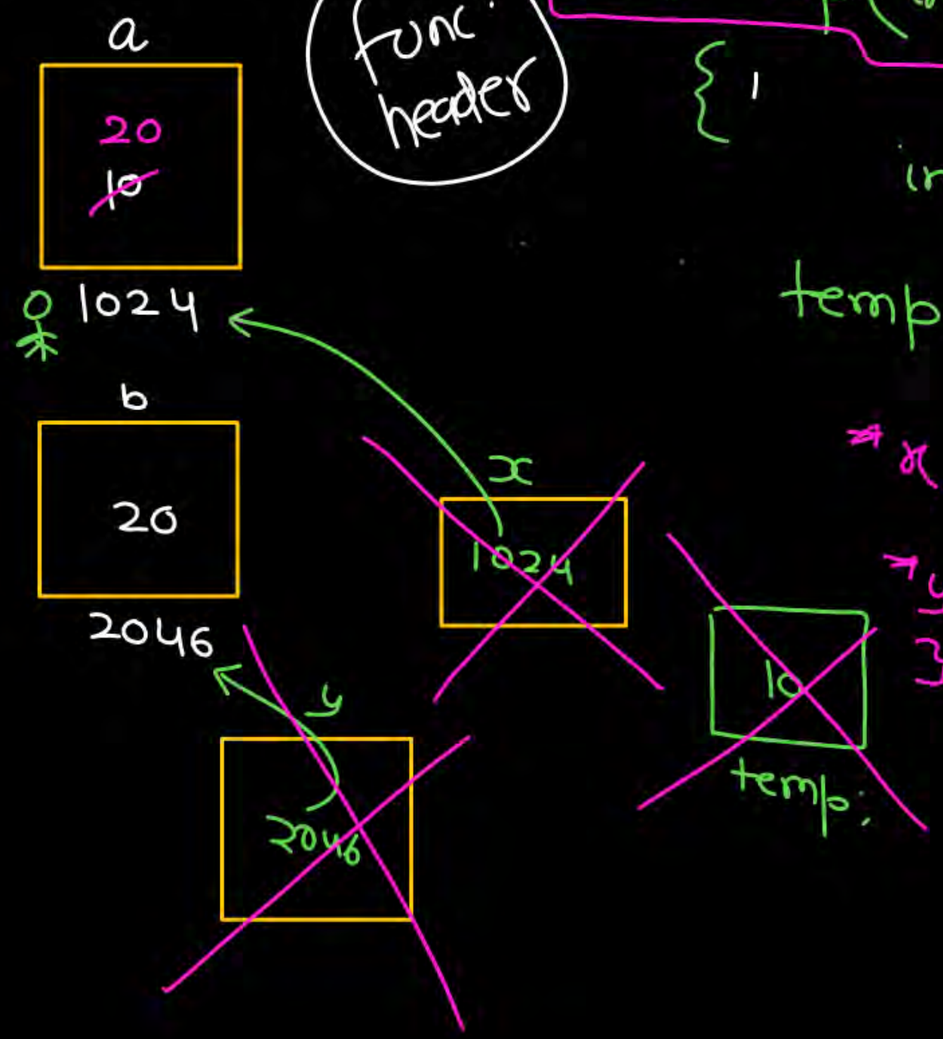
```
#include<stdio.h>
void swap(int*, int*);
void main(){
    int a=10,b=20;
    printf("%d %d",a,b);
    swap(&a,&b);
    printf("%d %d",a,b);
}
```

1024 2046

Forward declaration is as same as function header

Func. header

```
void swap(int *x, int *y)
{
    int temp;
    temp = *x;
    *x = *y;
    *y = temp;
}
```



3. int (*p)(int, int);

function → argument list

function → return

P is a pointer to function that takes two integer arguments and return an integer.

Function Pointer

```
int Add(int, int);
```

↓

```
int (*P) (int, int);
```

```
void fun(float, int);
```

↓

```
void (*P)(float, int);
```

```c
int Add(int, int);
void main(){
    int (*p)(int, int);
    int a=10, b=20, sum;
        p = &Add;
    sum = (*p)(a,b);
    printf("%d", sum);
        }

int Add(int x, int y)
    {
        return x+y;
    }
```

```c
void main()
{
    p = Add;
    sum = p(a,b);
}
```

```c
p = Add;
sum = (*p)(a,b);
```

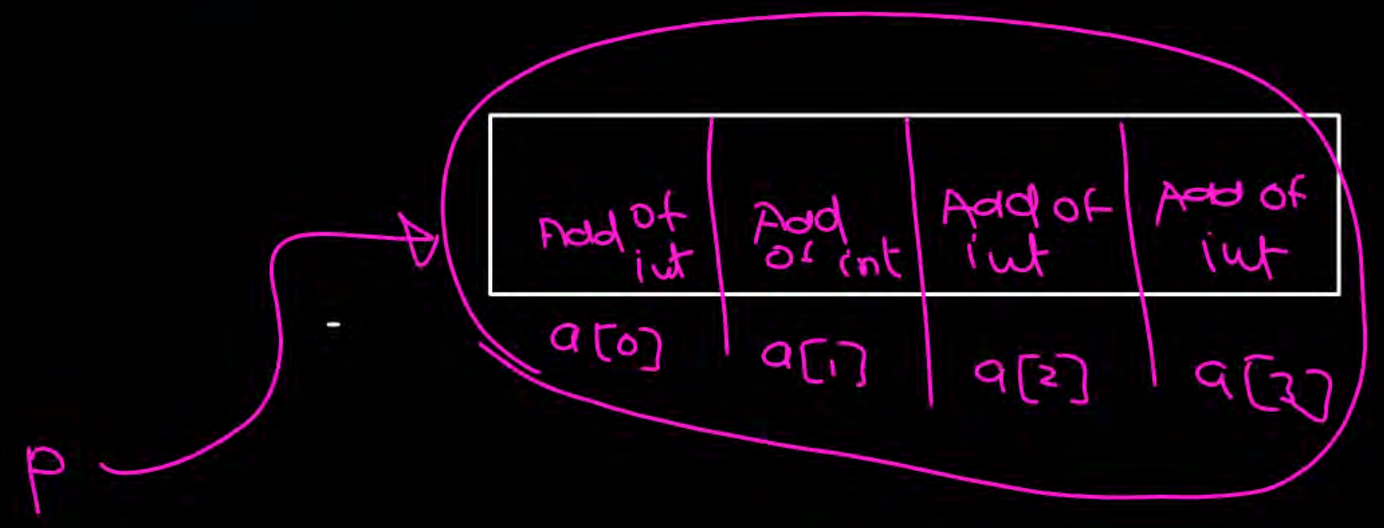
```c
p = &Add;
sum = p(a,b);
```

4. int * (*p)[4];

8:30 PM

P is a pointer to an array of 4 pointer to integer.

int *a[4];
P = &a;



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

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


THANK YOU GW
SOLDIERS !