CS & IT ENGINEERING

ARRAYS AND POINTER-1

Lecture No.12

By- Pankaj Sharma SIR

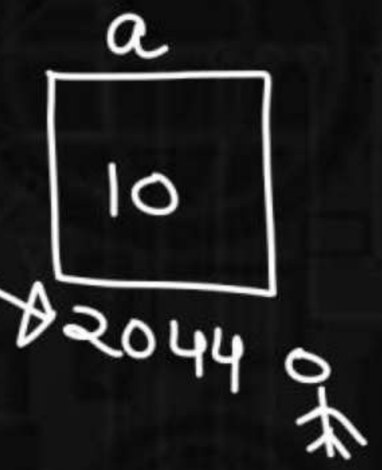
Address :

A-34, Krishna Nagar
Mathura - 281004

& : address of operator

int a = 10;

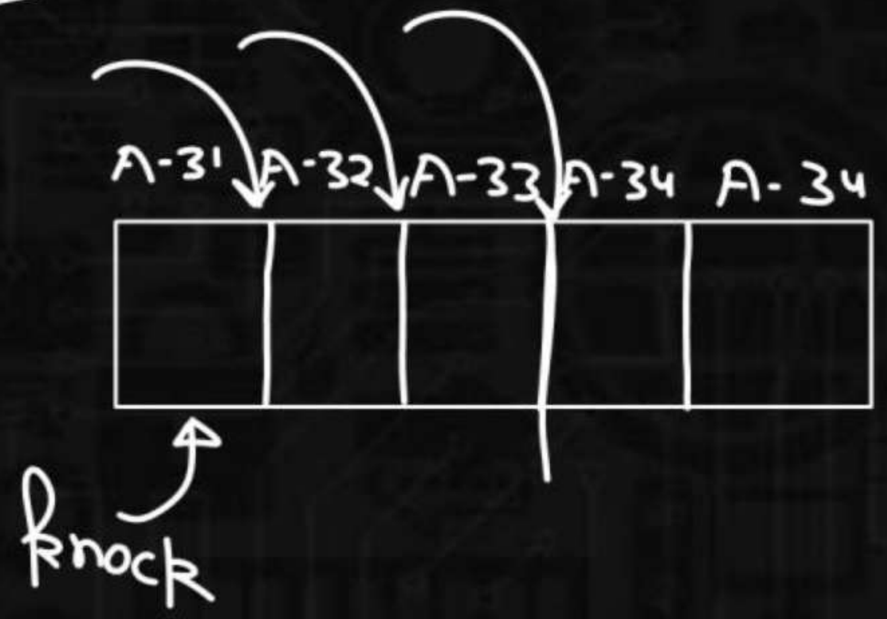
&a



Address → Relative address

Address → Abs. address

Purpose



*: value at operator

*(Memory location / Address)

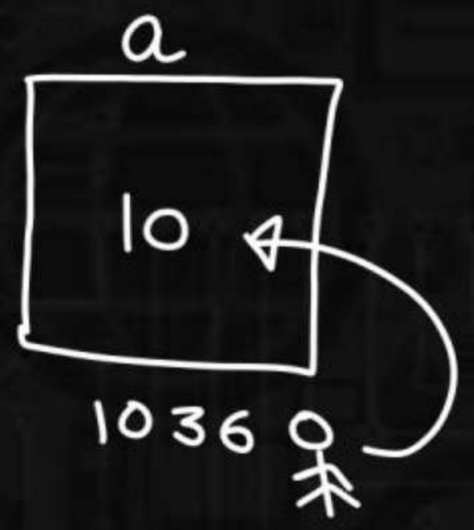
int a = 10;

a → 10

&a → 1036

*(&a) ≡ value at (Memory location 1036)

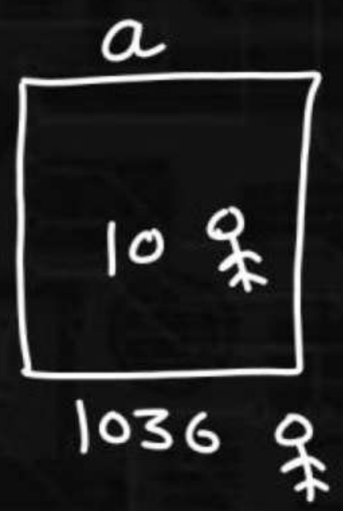
*&a = 10



$a \rightarrow 10$

$\&a \rightarrow 1036$

$*\&a \rightarrow 10$



$a \Rightarrow \cancel{*\&}a$

```
printf("%d", *&*&a);
```

10

```
int m1,m2,m3;
float avg;
avg = (m1+m2+m3)/3
```

50 students

```
int m1,m2,m3,m4,m5,m6,m7,m8,m9,m10,m11,....
```

500 students

```
int m1,m2,m3,m4,m5,.....
```

```
int a,b,c;
```

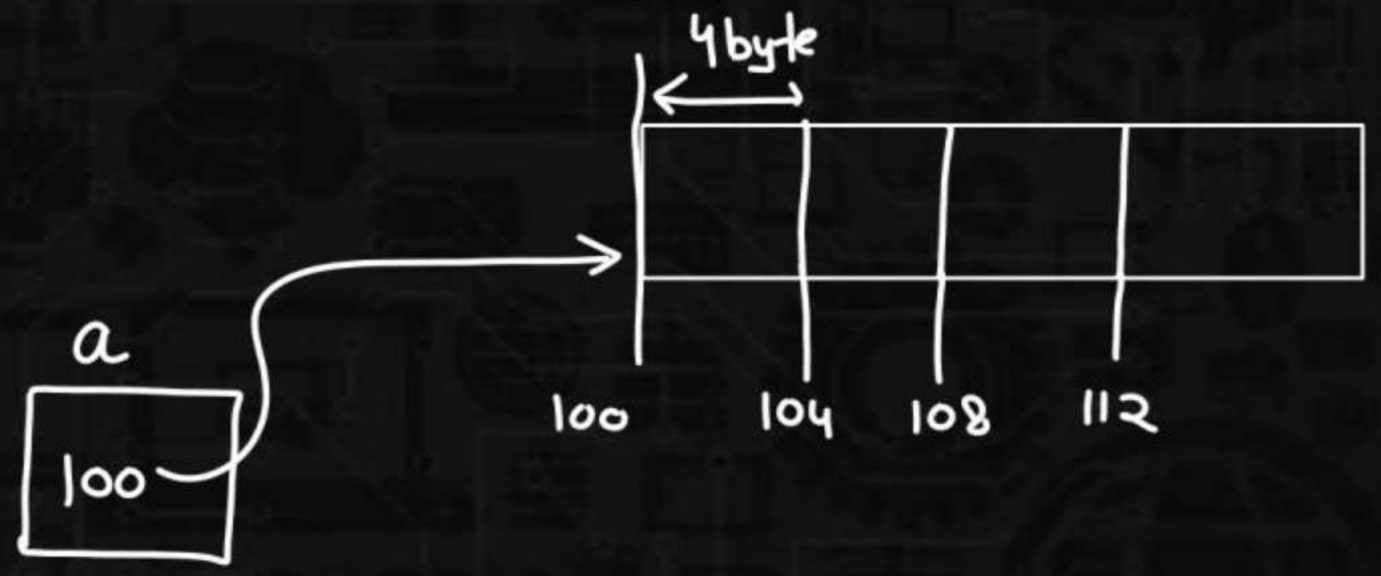
```
int a[3];
```

```
data_type  Name[size]
```

① Homogenous types of elements.

```
int a[50];
```

int a[4];



(2) sequential store

(3) Array name represent address of first element.

④ array_name : constant

Lvalue can not be a constant

$2 = a;$ ✗

$a++$

$a = a+1$

$2++$

$2 = 2+1$ ✗

Array_name = ✗

$2++, 2--, --2, ++2;$

Array_name ++;

Array_name --;

--Array_name;

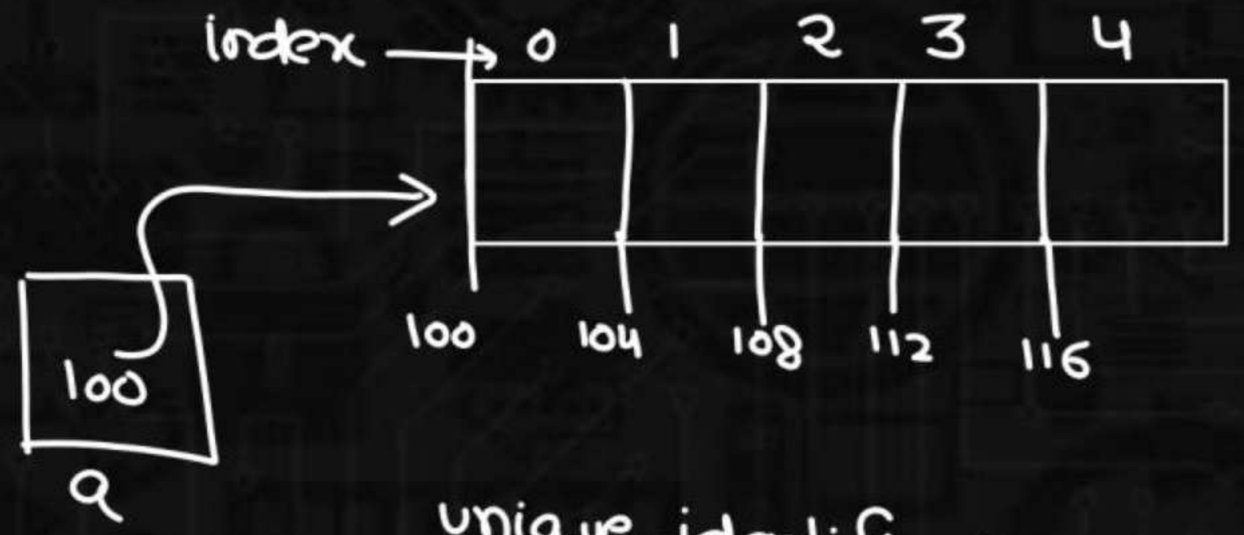
++Array_name;

all are invalid

5. int a[5];

0 to size-1

a is a group of 5 similar type (int) of element.



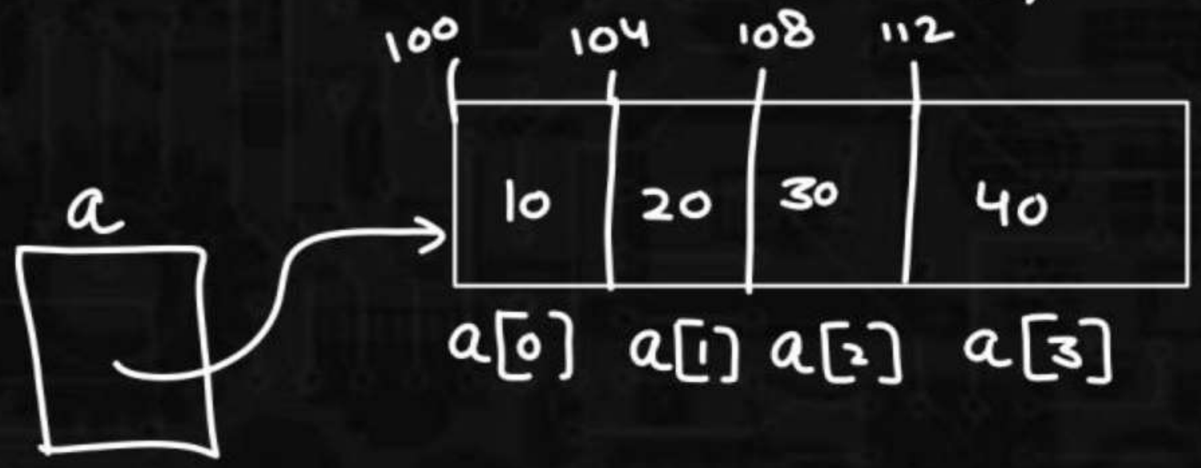
Sec-B

60 elements (student)

unique-identification no: index

6.

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




```
printf("%d",a[1]);   20
```

1st element → 0
2nd element → 1
3rd element → 2

```
int a;
printf("%d",a);   Garbage
```

```
int b[4];
```

1-D array

① int a[4] = {10,20,30,40}; ✓

② int a[] = {10,20,30,40}; ✓

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

③ int a[4]; ✓

Gar	Gar	Gar	Gar

④ int a[ ]; ✗

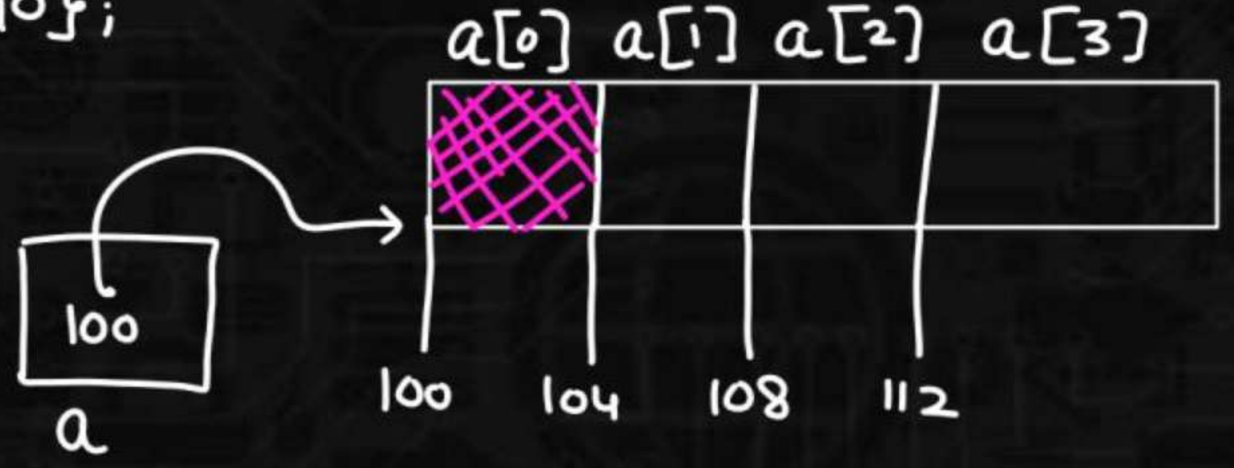
⑤ int a[4] = {10,20}; ✓

10	20	0	0

⑥ int a[4] = {10,20,30,40,50,60}; ✓ → ignored

Array and address

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



① Name of an array is address of first element

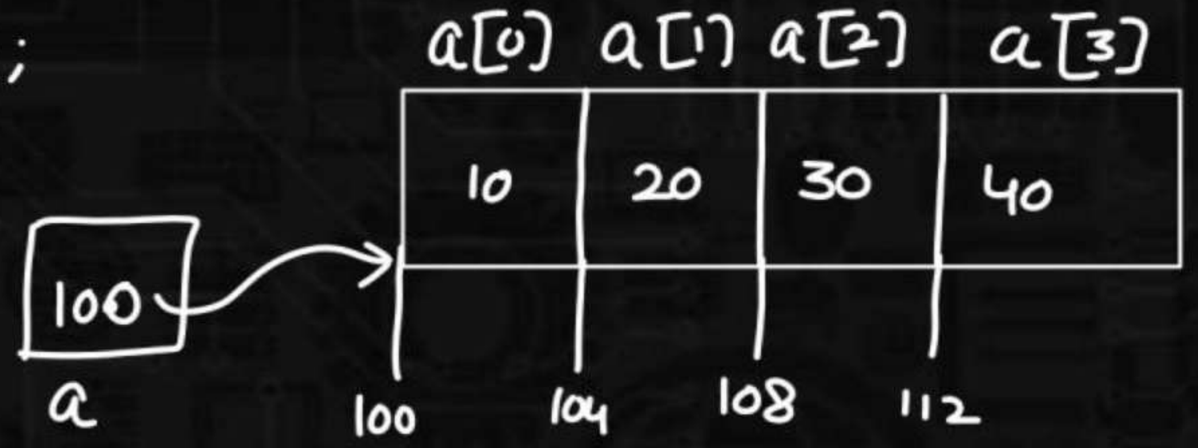
$a \xleftrightarrow{\text{same}} \&a[0]$

```
printf("%u", a);
printf("%u", &a[0]);
```

same output

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

printf("%u", *a);
```



$$a \equiv \&a[0]$$

$$*a \equiv \cancel{*}\cancel{\&}a[0]$$

$$*a = a[0]$$

② int a[4] = {10, 20, 30, 40};

1-dimension

a[0] → 10

a[1] → 20

1-dim

a → Address

0-dim < 1-dim

int a[3][4]; 2-dimension in declaration

① a → Address/value 0-dimension

② a[0] → Address/value 1-dimension

③ a[0][0] → Address/element or value 2-dimension

③ Numerical value of a is 100

$a+1$?

int a = 100;

$a+1 \Rightarrow 101$

value + value ⇒ value ✓

simple variable/value

⇒ 101

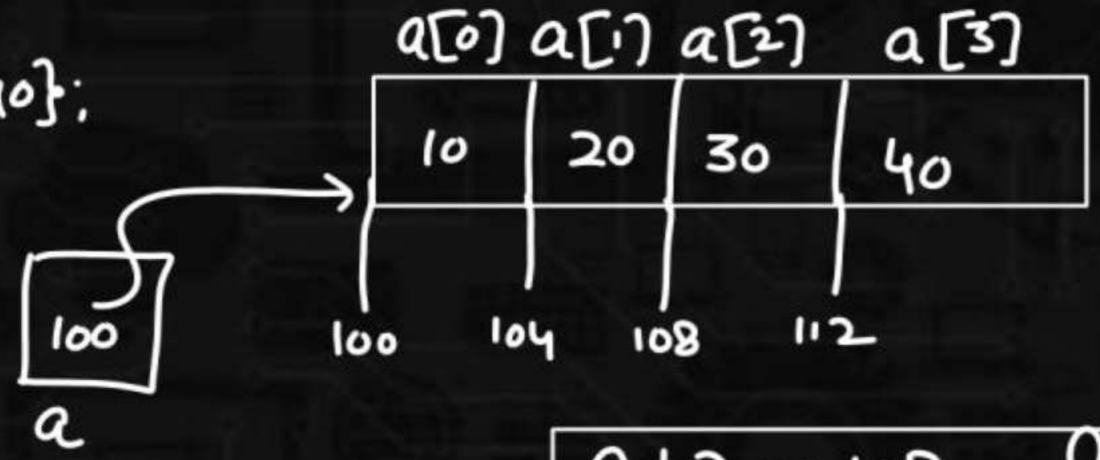
address + value ⇒ ? Address

(i) what is a → Address/value

(ii) Find out whose address is represented by a.

(iii) size

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



$a+1$

(i) a → Address ✓
→ element

(ii) $a \equiv \&a[0]$

(iii) size of a[0] ⇒ 4 bytes

$a+2 = 108 = \&a[2]$

$a+1 \equiv \&a[0] + 1 \times 4$

$100 + 4 \Rightarrow 104$

$a+1 \Rightarrow 104 \Rightarrow \&a[1]$

$a+2 \Rightarrow \&a[0] + 2 \times 4$

$\Rightarrow 100 + 2 \times 4 \Rightarrow 108$

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

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

$a+2$ = Memory location 108 $\equiv \&a[2]$

$*(a+2)$ = value at (Memory location 108)

$= \cancel{*}\cancel{\&}a[2]$

$\boxed{*(a+2) = 30 = a[2]}$

$a+1$ = Memory location 104 $= \&a[1]$

$*(a+1) = *($Memory location 104$) = \cancel{*}\cancel{\&}a[1]$

$\boxed{*(a+1) = 20 = a[1]}$

$a[2] = *(a+2)$

$a[1] = *(a+1)$

$\boxed{a[i] = *(a+i)}$

$*(a+2)$ $\qquad$ $*(2+a)$

$\downarrow$ $\qquad$ $\downarrow$

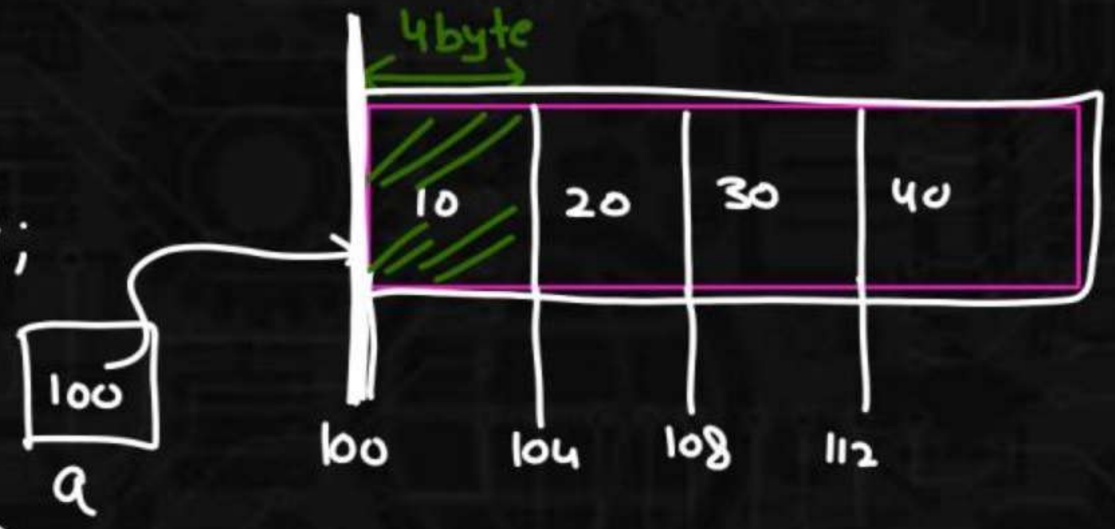
$a[2]$ $\qquad$ $2[a]$

$$\boxed{a[i] \equiv *(a+i) \equiv *(i+a) \equiv i[a]}$$

```
int a[4]={10,20,30,40};
printf("%d",a[2]);
printf("%d", *(a+2));
printf("%d", *(2+a));
printf("%d",2[a]);
```

Exception : with & operator, array-name is treated as object not address.

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



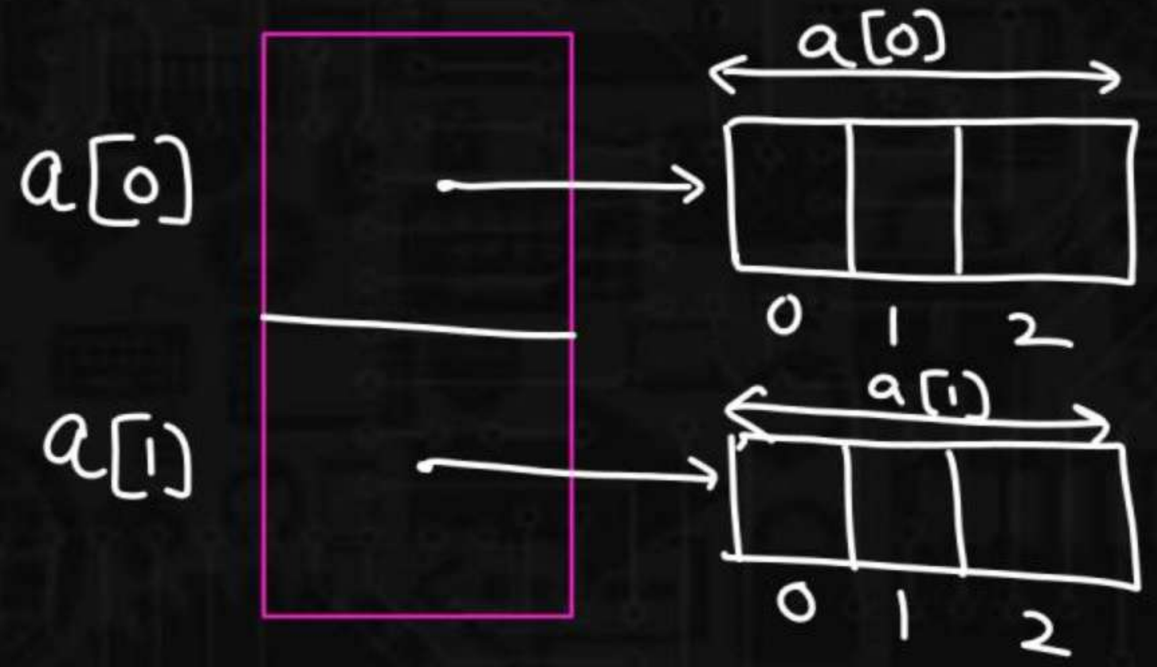
4 byte

```
printf("%u", a+1);      104
printf("%u", &a+1);     &a+1 ⇒ &a+1×16 ⇒ 116
printf("%u", &a);       100   whole array (add of)
printf("%u", &a[0]);    100
```

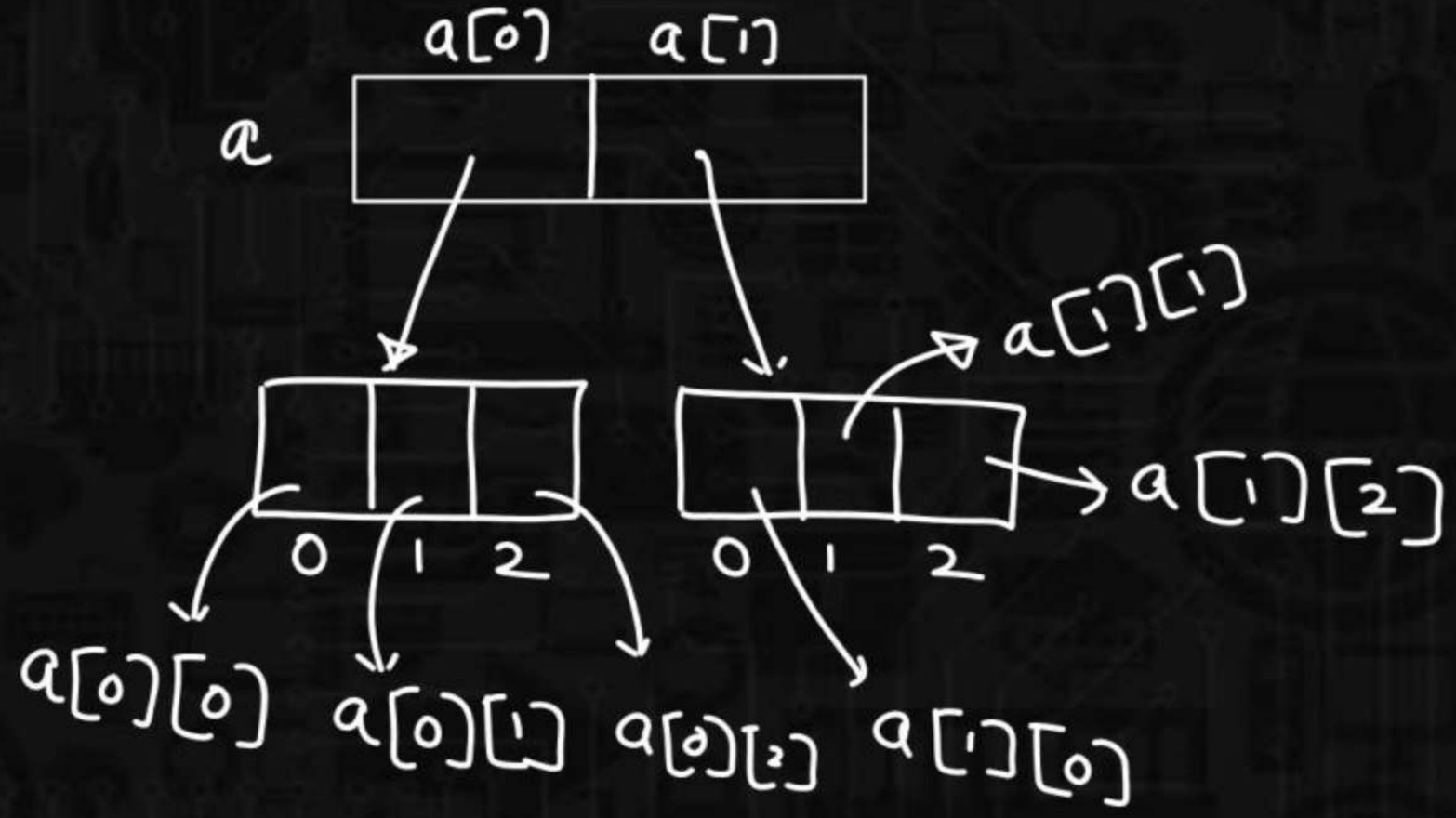
2-D array

0, 1

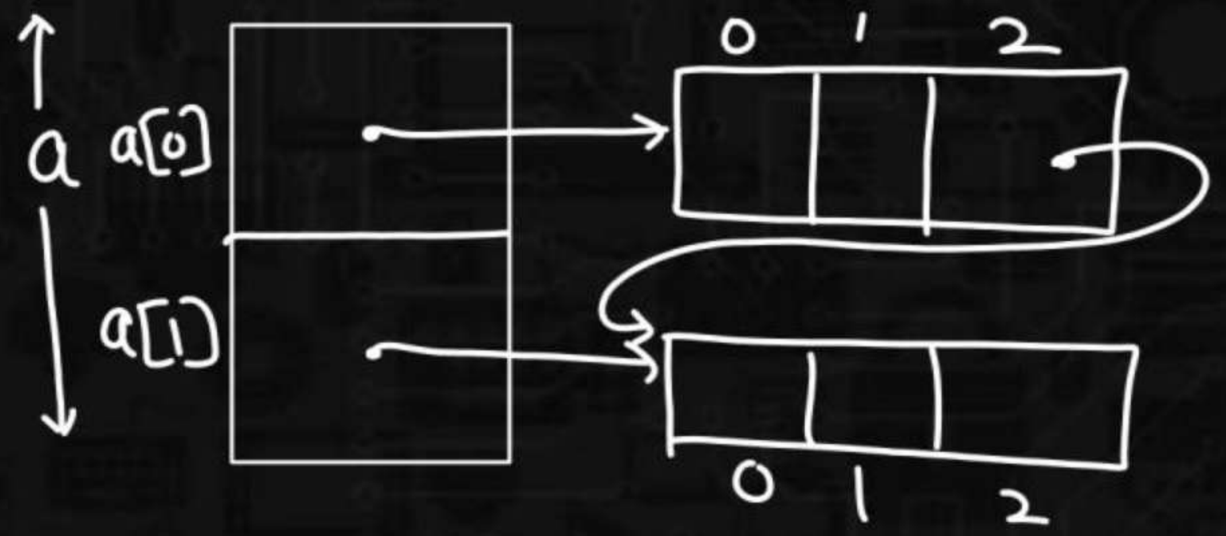
```
int a[2][3] = {1,2,3,4,5,6};
```

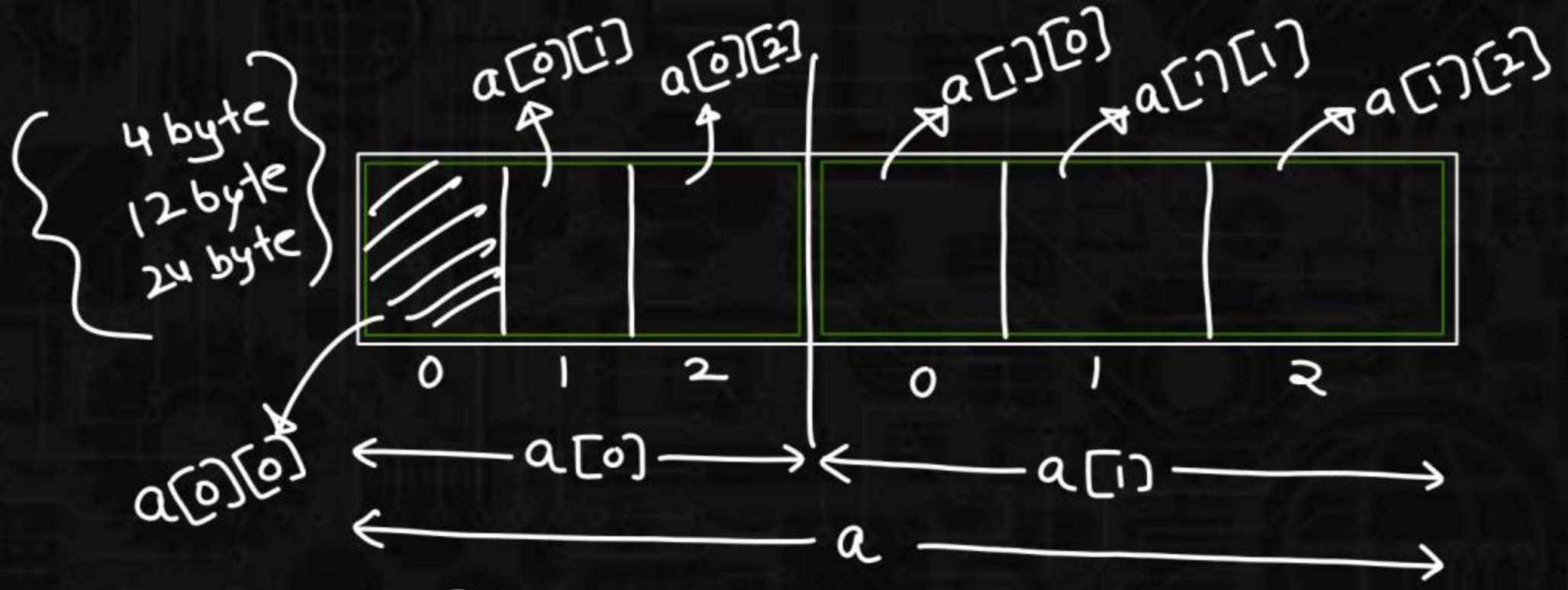


```
int a[2][3] = {1,2,3,4,5,6};
```



```
int a[2][3] = {1,2,3,4,5,6};
```





int a[2][3];

$a \equiv \&a[0]$ (12 byte)

$a[0] = \&a[0][0]$ (4 byte)

$a[1] = \&a[1][0]$ (4 bytes)

$\&a$ = whole

size of $a[0] \Rightarrow$ 12 byte

$a[1] \Rightarrow$ 12 byte

$a[0][0] \Rightarrow$ 4 byte

$\&a \Rightarrow$ 24 byte


THANK YOU GW
SOLDIERS !