CS & IT ENGINEERING

Programming in C

Arrays and Pointers

Lec- 02

By- Pankaj Sharma sir

Arrays and Pointers-2

① Array-name represent address of its first element.

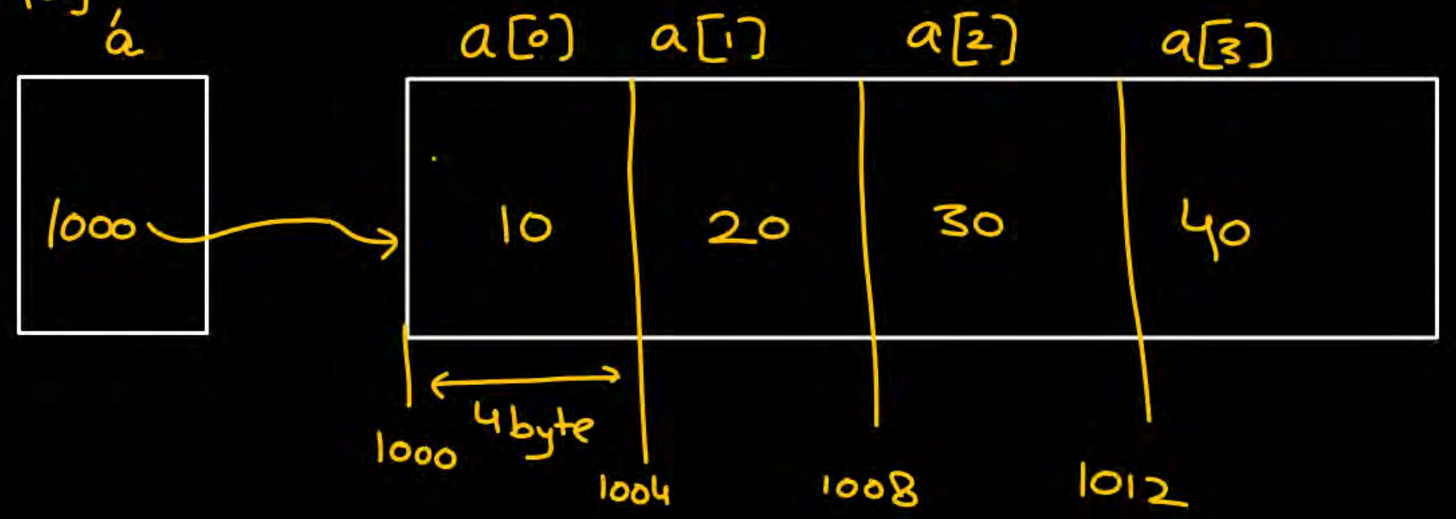
```
int a[4] = {10,20,30,40};
```

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

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

→ 1000

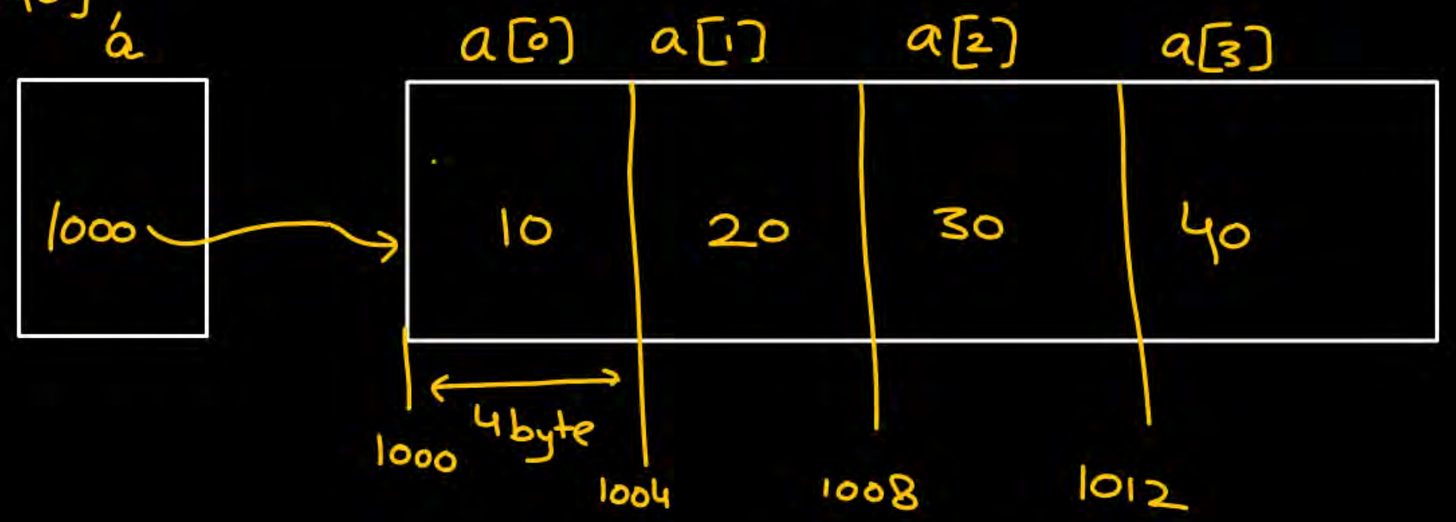
$a = \&a[0]$



← a →

(II) Array-name represent address of its first element.

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

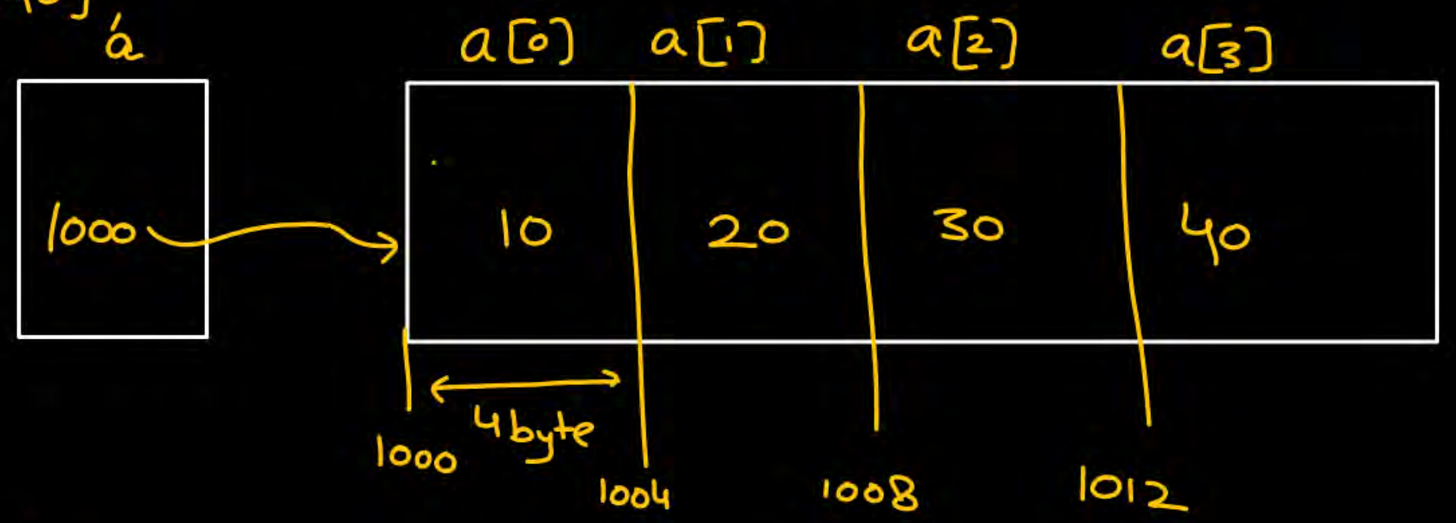


Array-name does not rep. an address with 2 operators

- &
- sizeof

II Array-name represent address of its first element.

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



$a = \&a[0]$

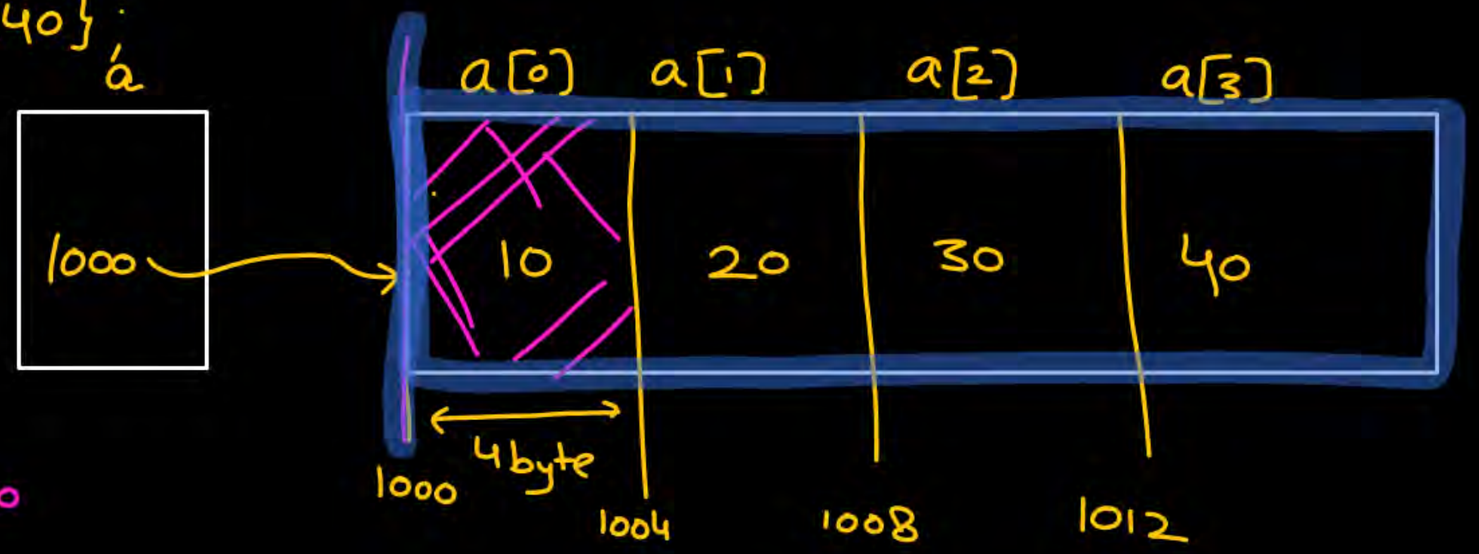
$\&a$?

we are talking about address of whole array (16 bytes)

$\longleftarrow a \longrightarrow$

II Array-name represent address of its first element.

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




```
printf("%u", a);      1000
printf("%u", &a[0]);  1000
printf("%u", &a);     1000
```

add of पूरा array

a] Numerical value same
&a] but first ⇒ address of an ele of 4 byte
second ⇒ " " " " element of 16 byte

a ⟶

Numerical value of a is 100

then what is a+1 ?

simple var. value

```
int a = 100;
printf("%d",a+1);
```

101

value + value = value

address

{ address arithmatic }

address + 1 ⇒ Address

address + val = Add

val + address = address

Address + Add ⇒ Invalid ✗

** If the declaration of an array has n-dimensions

```
int a[4];          1 dim.
int a[2][3];       2 dim.
int a[2][4][5];    3 dim.
```

a) Anywhere other than declaration if we give exactly n-dimension ⟹ element.

b) if we give less than n-dimensions ⟹ address

Ex1:

int a[4];

→ declaration

a → element ✗
→ add ✓

Ex2:

int a[2][3] = {1,2,3,4,5,6}; // 2-dim

a → add ✓
→ element ✗ // 0-dim

a[0] → Address
a[1] → Address

} 1-dim

a[0][0] → element // 2-dim

Ex1: `int a[4];`

dim indec = 1 dim

a → address (&a[0])

Ex2: `int a[2][3] = {1,2,3,4,5,6};` // 2-dim

dim = 2

a[0][0] → element

a[0] → Address

a[1] → Address

a → Address

a+1 ?

what is a ? → value → dance

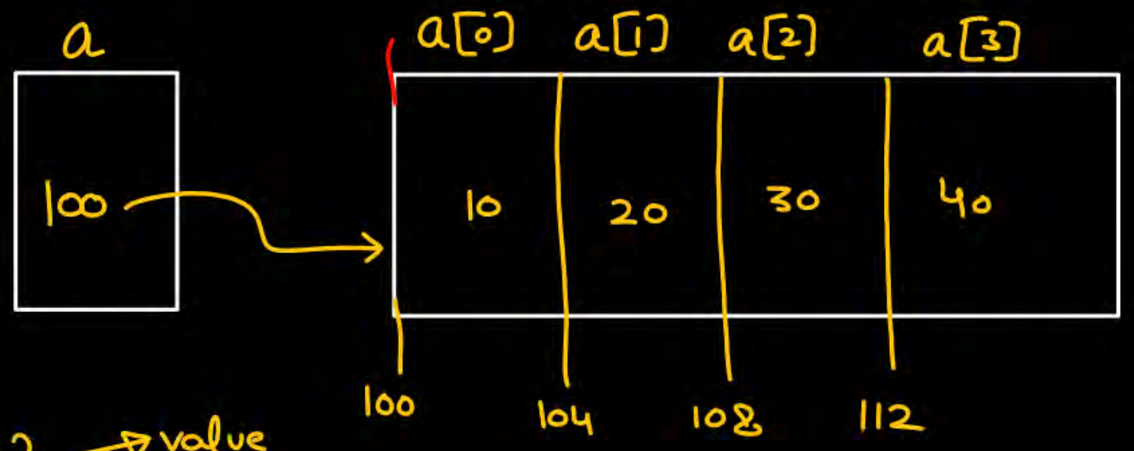
→ Address

(i) किसका address है ?

(ii) उसका size क्या है ?

```
int a[4] = {10,20,30,40}; //dec. 1 dimension
```

```
printf("%u", a+1);
```



a+1 (i) what is a ? → value
→ address ✓

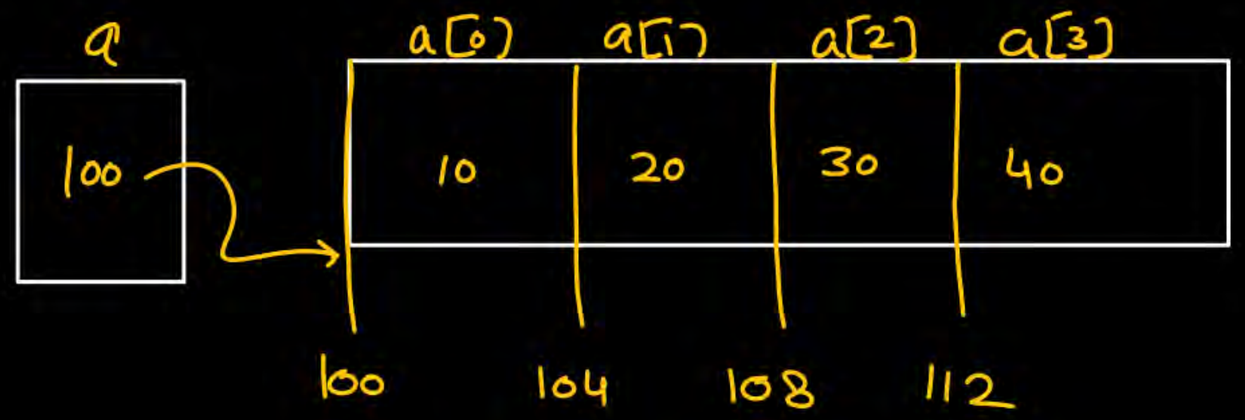
(ii) किसका address है ⇒ &a[0]

(iii) size of a[0] ⇒ 4 byte

$$a+1 = \&a[0] + 1$$

$$= \&a[0] + 1 \times 4 = 100 + 4 = 104$$

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



$$a+1 = \&a[0]+1$$
$$= \&a[0]+1\times 4$$
$$= 100+4 = 104$$

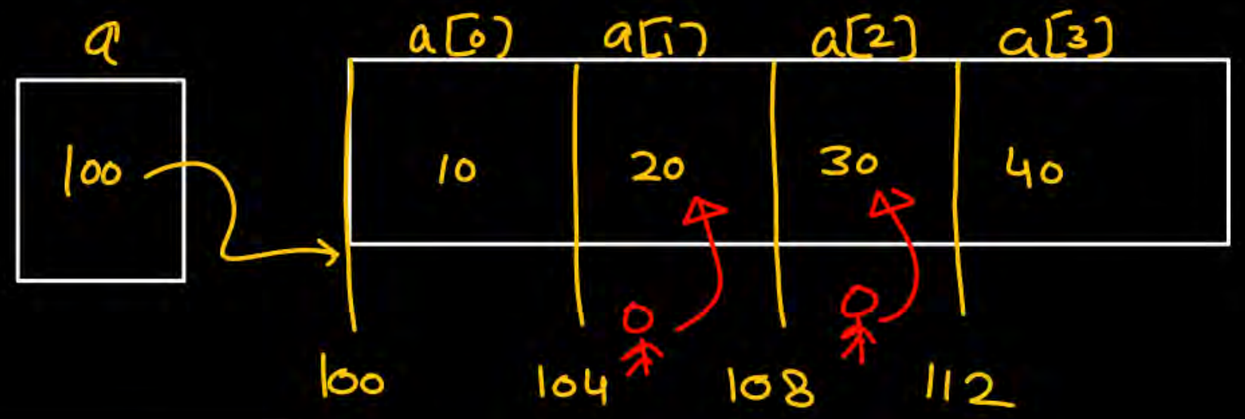
$$a+1 = 104 = \&a[1]$$

$$a+2 = \&a[0]+2$$
$$= \&a[0]+2\times 4$$
$$= 100+8$$
$$a+2 = 108$$

$$\boxed{a+1 = \text{Memory location } 104 = \&a[1]} \quad -(1)$$

$$\boxed{a+2 = \text{Memory location } 108 = \&a[2]} \quad -(2)$$

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



$$\boxed{a+1 = \text{Memory location } 104 = \&a[1]} \quad -(1)$$

$$*(a+1) = \text{value at}\left(\begin{matrix}\text{Memory}\\ \text{location}\end{matrix}\ 104\right) = \cancel{*}\cancel{\&}a[1] \Rightarrow \boxed{*(a+1) = a[1]}$$

$$\boxed{a+2 = \text{Memory location } 108 = \&a[2]} \quad -(2)$$

$$*(a+2) = \text{value at}\left(\text{Memory loc. } 108\right) = \cancel{*}\cancel{\&}a[2] \Rightarrow \boxed{*(a+2) = a[2]}$$

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

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

$2+3 = 3+2$

Commutative

Address + val = Address

val + Address = Address

```
void main(){
int a[4] = {1,2,3,4};
printf("%d", a[1]);
printf("%d", *(a+1));
printf("%d", *(1+a));
printf("%d", 1[a]);
}
```

$$*(a+i) = *(i+a)$$

$$= a[i] = i[a]$$

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

Error

Compiler ud ke laat Maarega

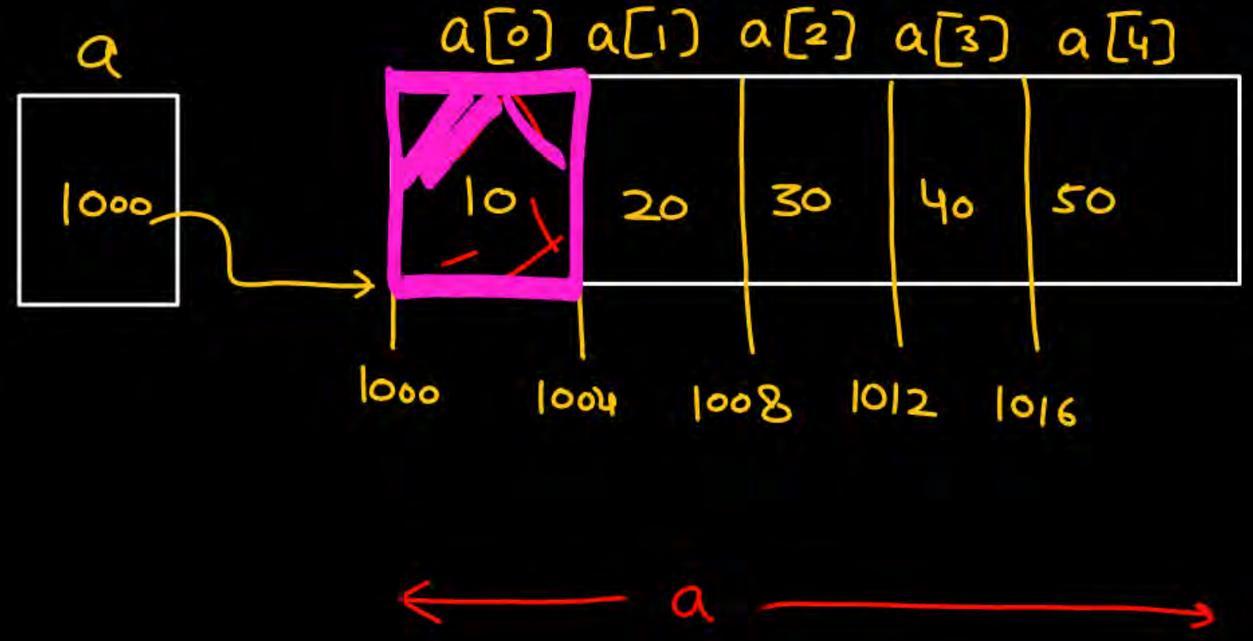
int a = 10; → Compiler info → Save

```
void main(){
int a[5] = {10,20,30,40,50};

printf("%u", a);
printf("%u", &a);
printf("%u", a+1);
printf("%u", &a+1);

}
```

array-name
= First ele
add
= &a[0]
= 1000



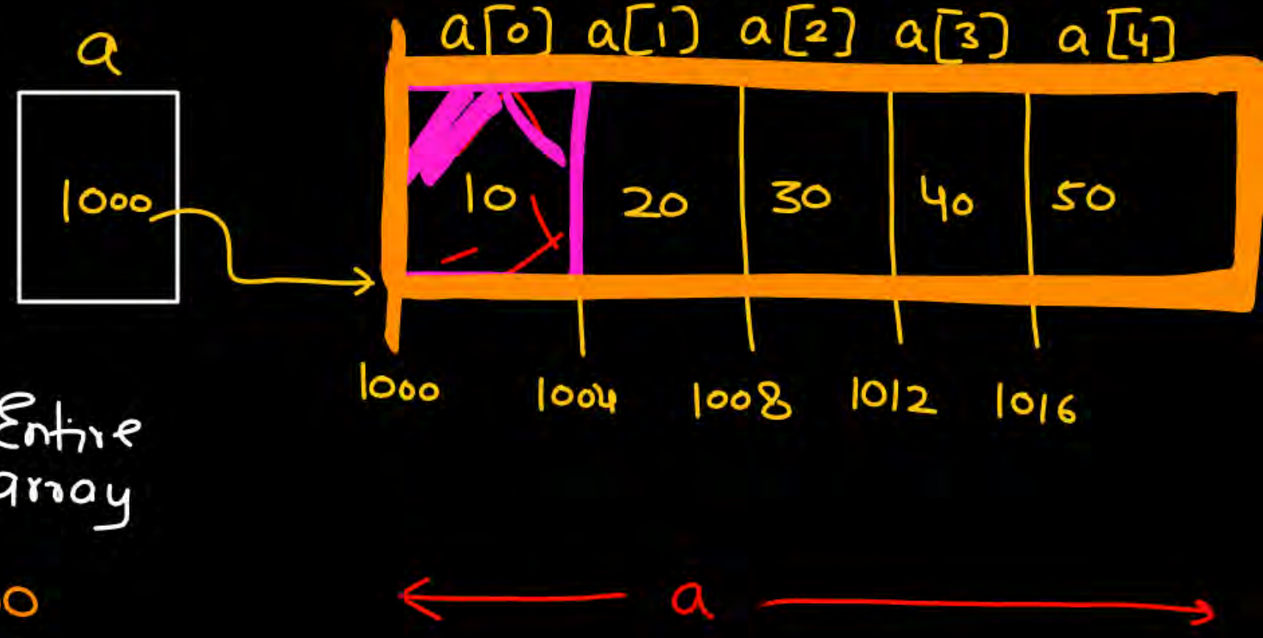
```
void main(){
int a[5] = {10,20,30,40,50};

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

→ Address of Entire array

```
printf("%u", a+1);
printf("%u", &a+1);

}
```



1000

```
void main(){
int a[5] = {10,20,30,40,50};

printf("%u", a+1);

}
```

→ Address ✓
→ element ✗

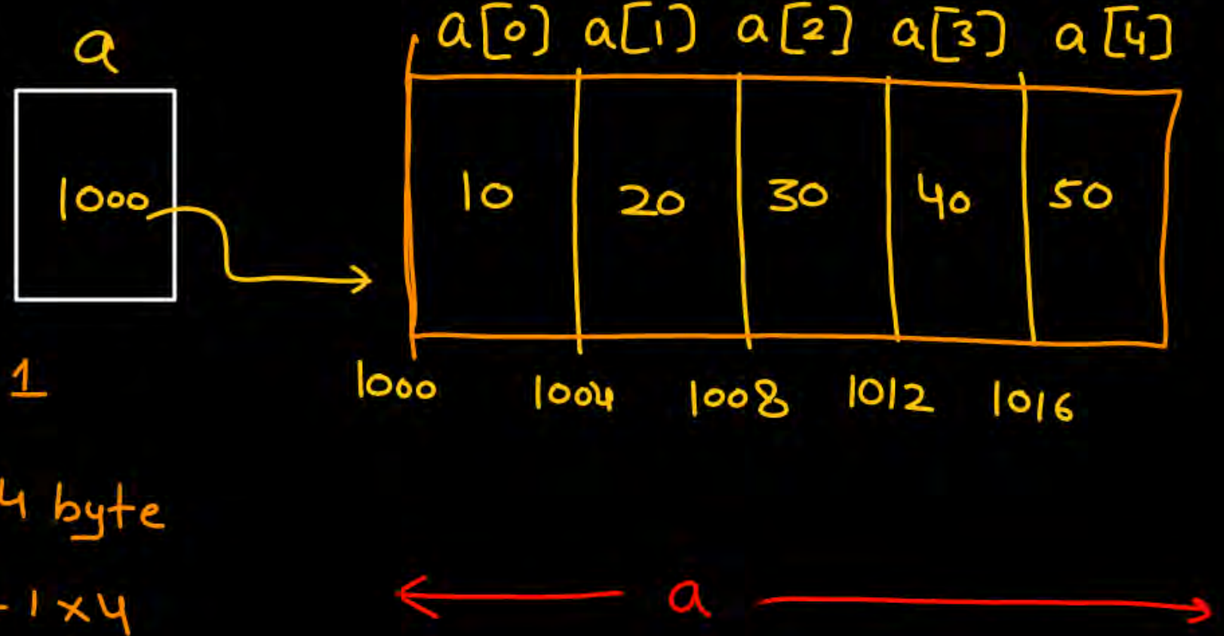
(i) &a[0] + 1

(ii) a[0] → 4 byte

(iii) &a[0] + 1×4

1000 + 4 = 1004

```
printf("%u", &a+1);
```



```
void main(){
int a[5] = {10,20,30,40,50};

printf("%u", &a+1);
```

Address of

Entire array (20 bytes)

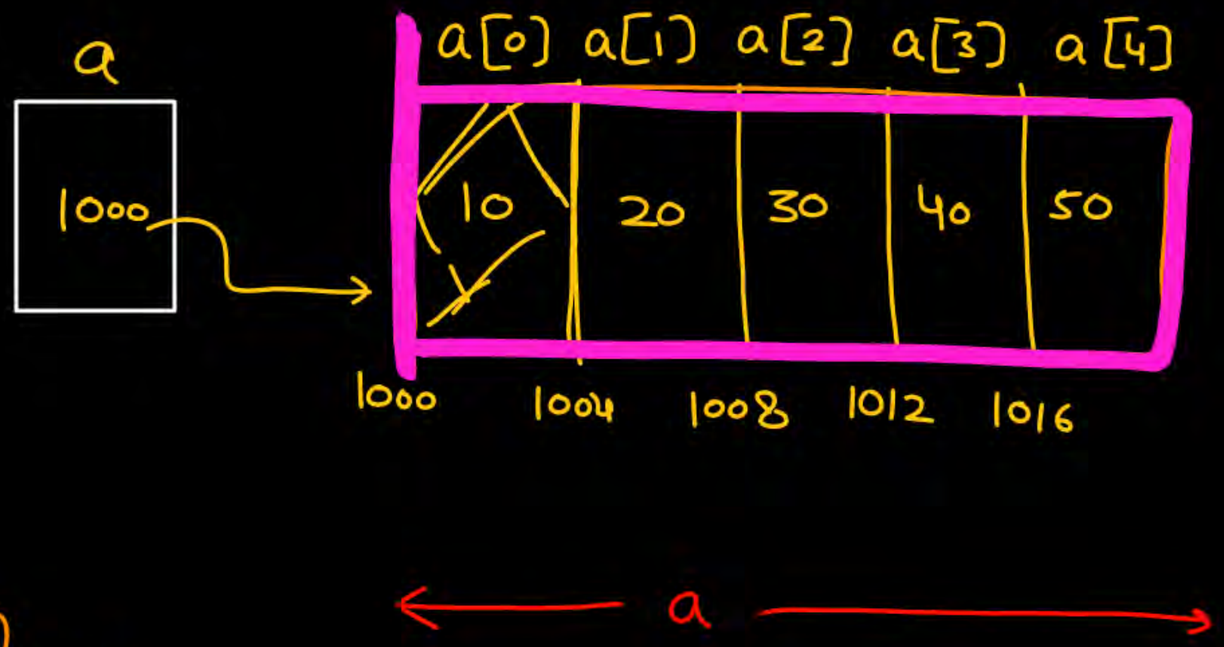
$\Rightarrow \&a+1$

$= \&a + 1 \times 20$

$= 1000 + 20$

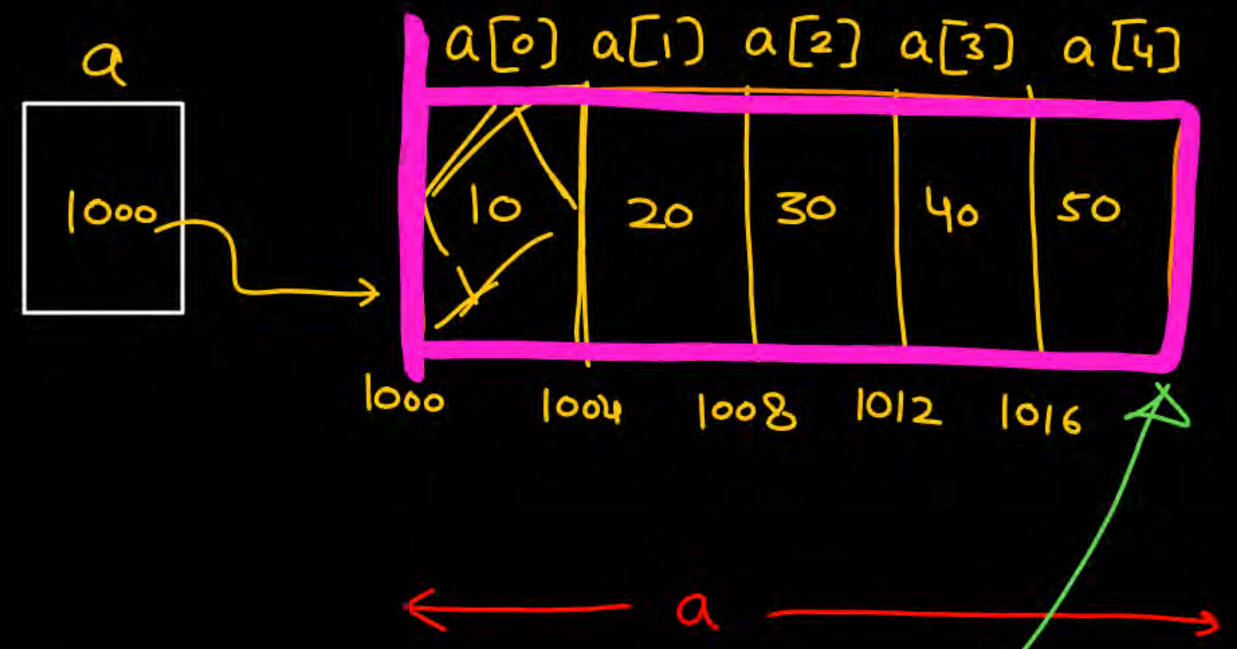
$= 1020$

```
}
```



```
void main(){
int a[5] = {10,20,30,40,50};

printf("%u", &a+1);
```



Student

Slow learner

(i) 50%

Recording (50%) ✗

Fast learner ✓

logic building

$$100x + 300y = 600$$

Scale : 1 small rectangle = 100 units


THANK YOU GW
SOLDIERS !