CS & IT ENGINEERING

Programming in C

Arrays and Pointers

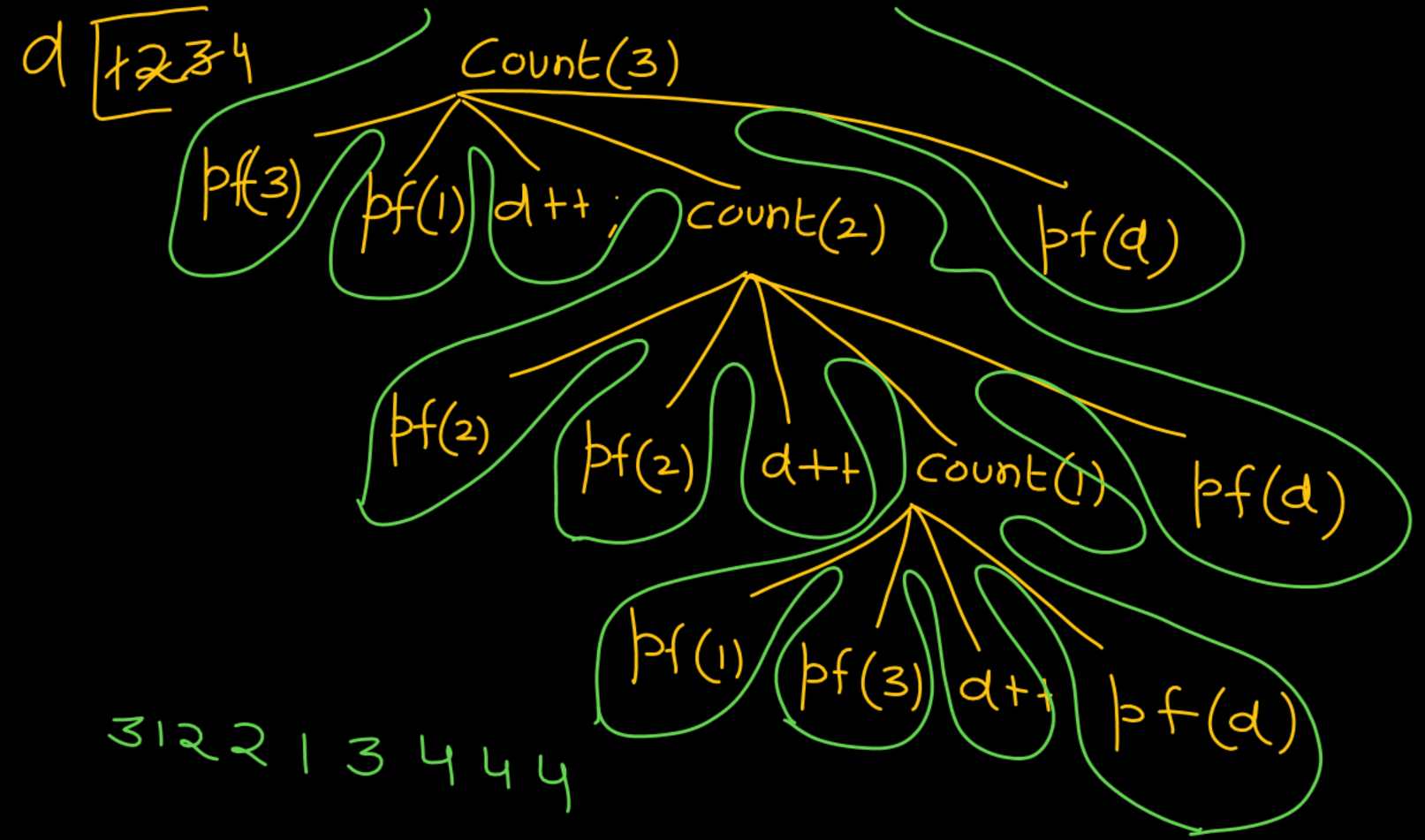
Lec- 01

By- Pankaj Sharma sir

Arrays

Q27.

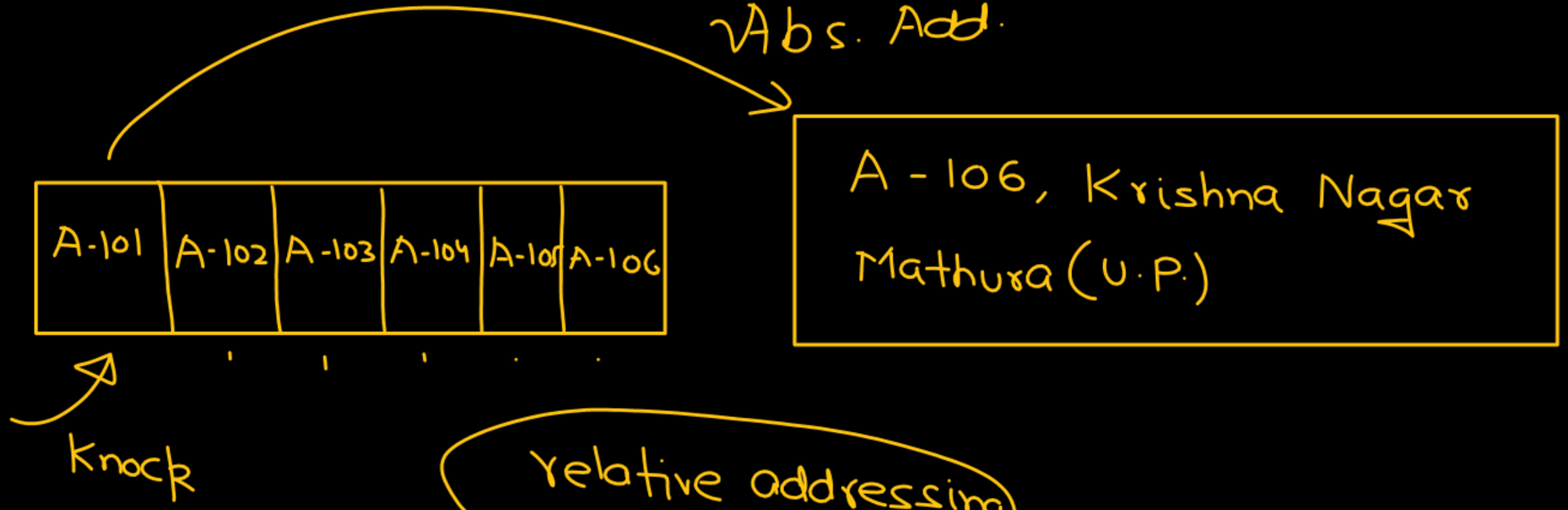
```
void count (int n){
    static int d=1;
1.  printf("%d",n);
2.  printf("%d",d);
3.  d++;
4.  if(n>1) count(n-1).
5.  printf("%d",d);
    }
```



3122134444

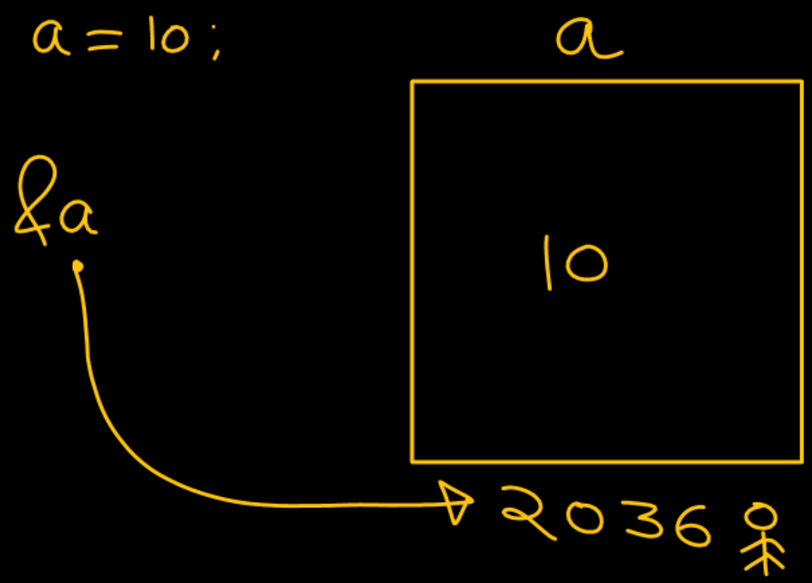
Array and address

① Address → Abs. Address
→ Relative Address



② How to find address : & (address of operator)

int a = 10;



③ * (value at operator): value at (Memory location)

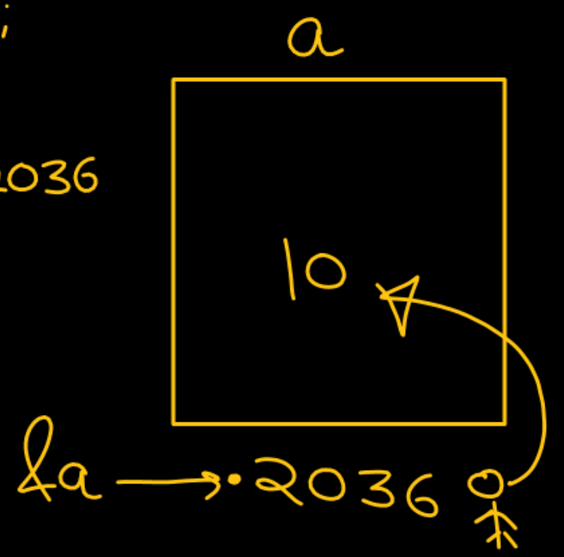
int a = 10;

&a : Memory location 2036

*(&a)

≡ value at (Memory location 2036)

⇒ 10



(i) $a \equiv 10$

$\&a \equiv$ Memory location 2036

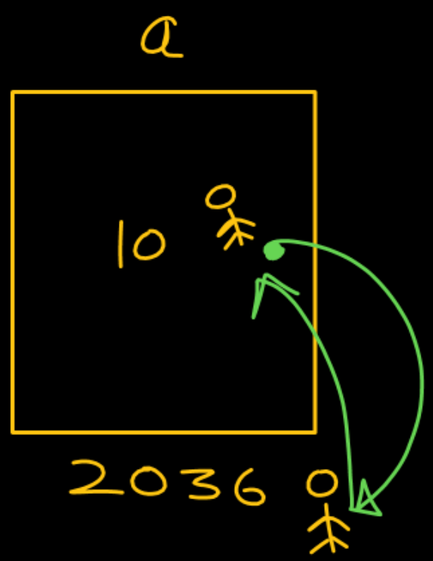
$*(\&a) \equiv$ value at (Memory location 2036)

$\Rightarrow 10$

$a \Rightarrow 10$

$\&a \Rightarrow 2036$

$*\&a \Rightarrow 10$




```
int a = 10;
printf("%d", *&*&a);
```

$a \equiv *\&a$

avg of marks 3 student

```
int m1, m2, m3 ;
float avg;
```

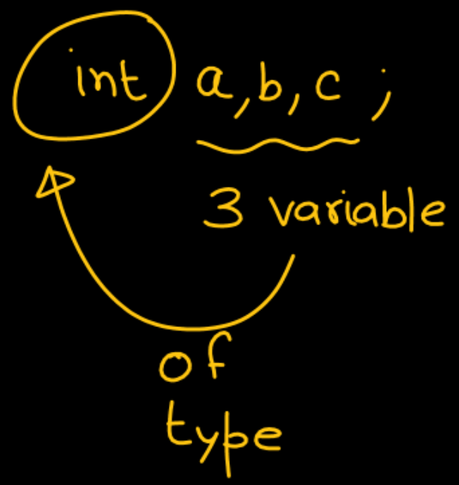
Why array

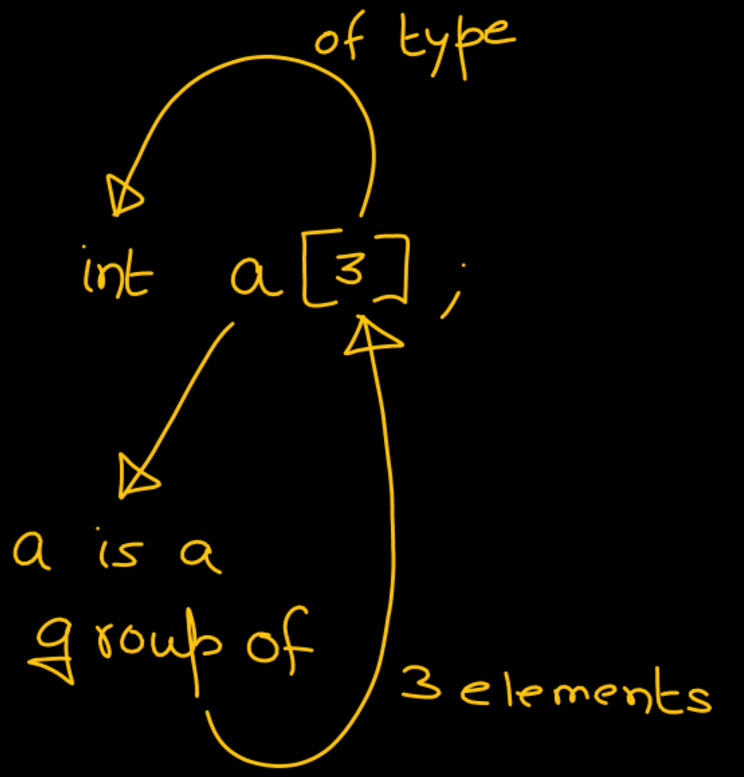
So students

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

```
scanf("%d %d %d . . . %d", &m1, &m2, . . .
```

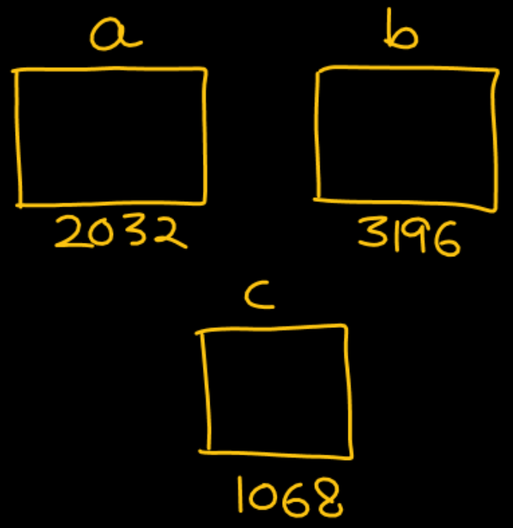
```
for ——
scanf("%d %d %d", &m1, &m2, &m3);
```



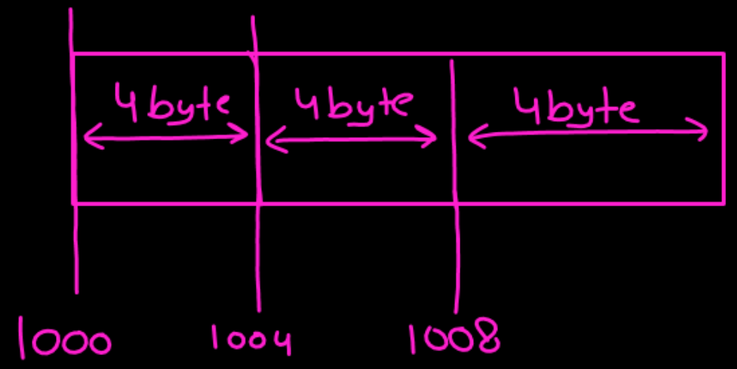


a is a group of 3 elements each of type int.

int a, b, c ;



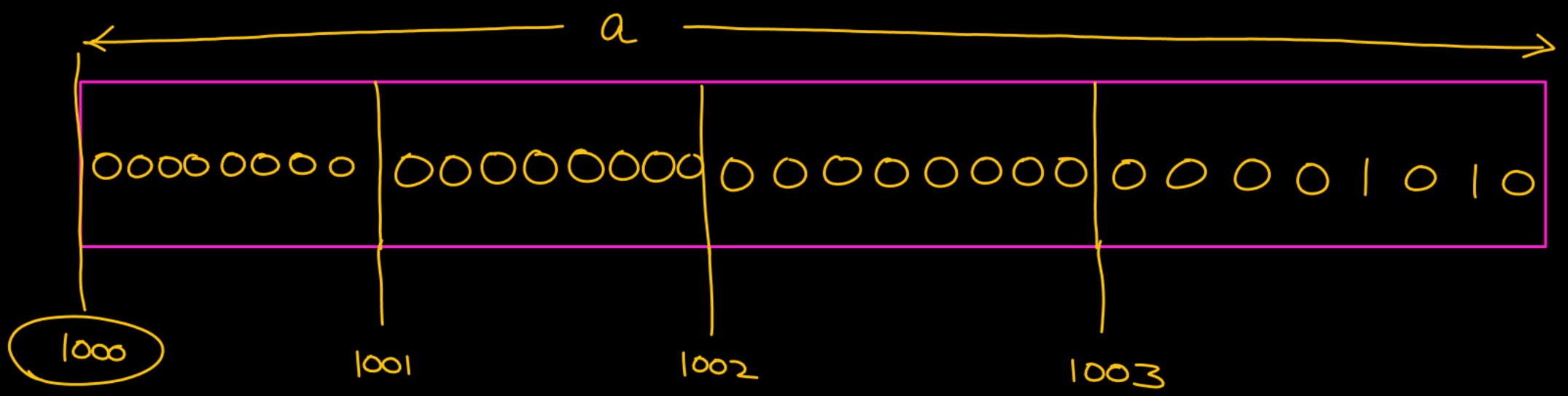
int a[3] ;

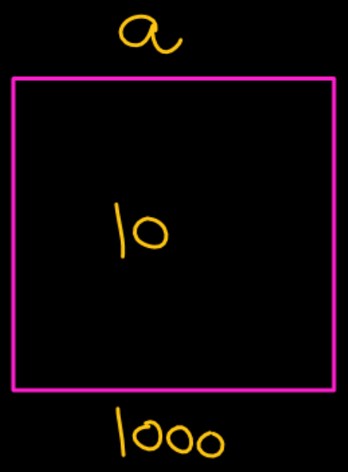


store

(i) One after another (seq.)

int a = 10 ;


a
00000000
00000000
00000000
00001010
1000
1001
1002
1003


a
10
1000

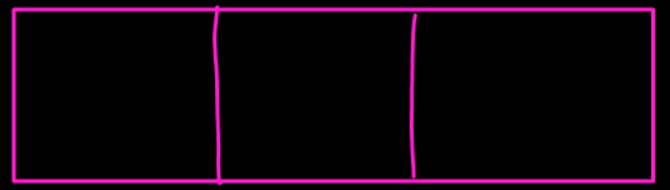
```
int a, b, c;

a = 10;
  =
b = 30;
  =
c = 50;
  =
```

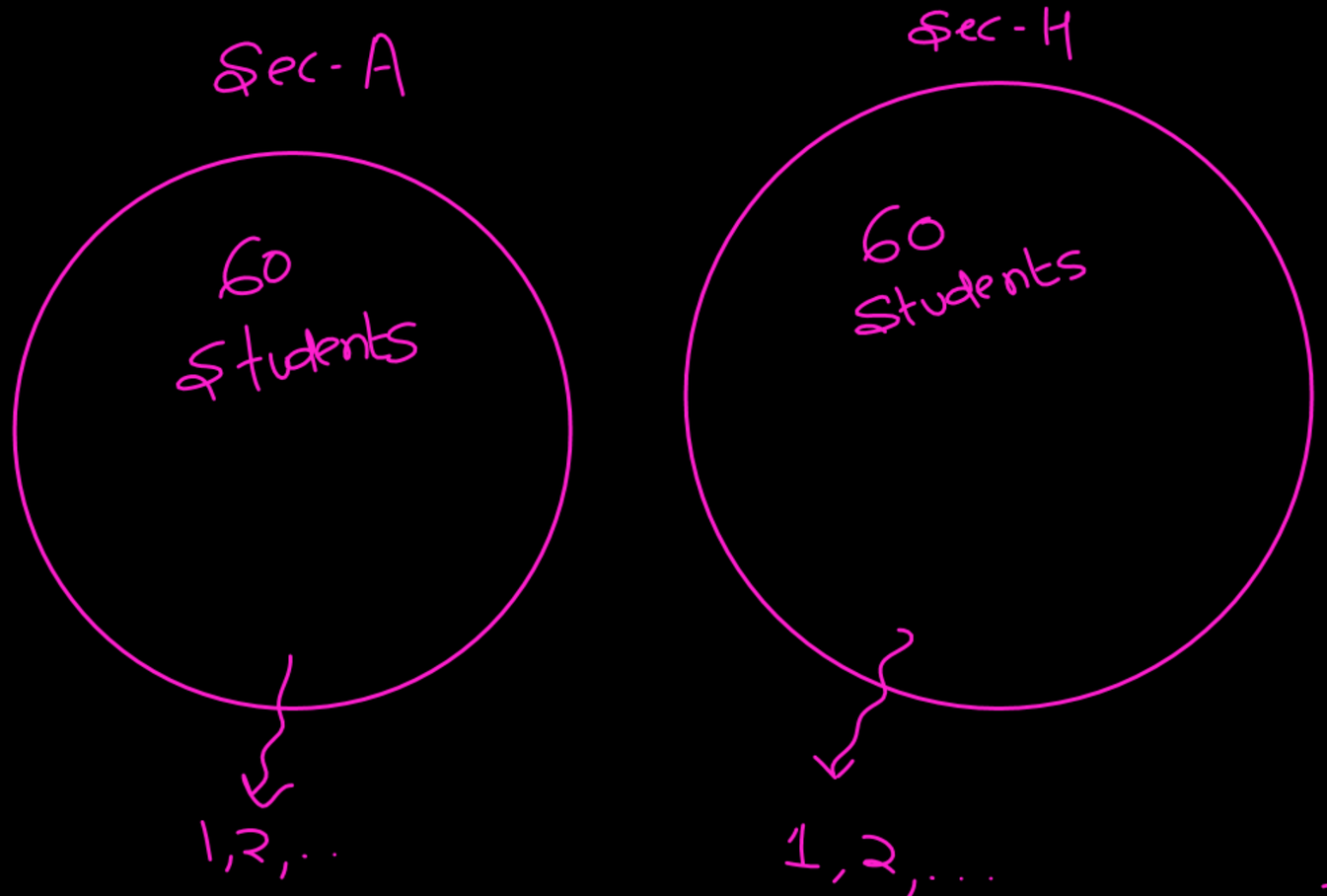
```
int a[3];
```

Group/collection

a



All 3 elements are represented by same entity/name



H

Unique identification number

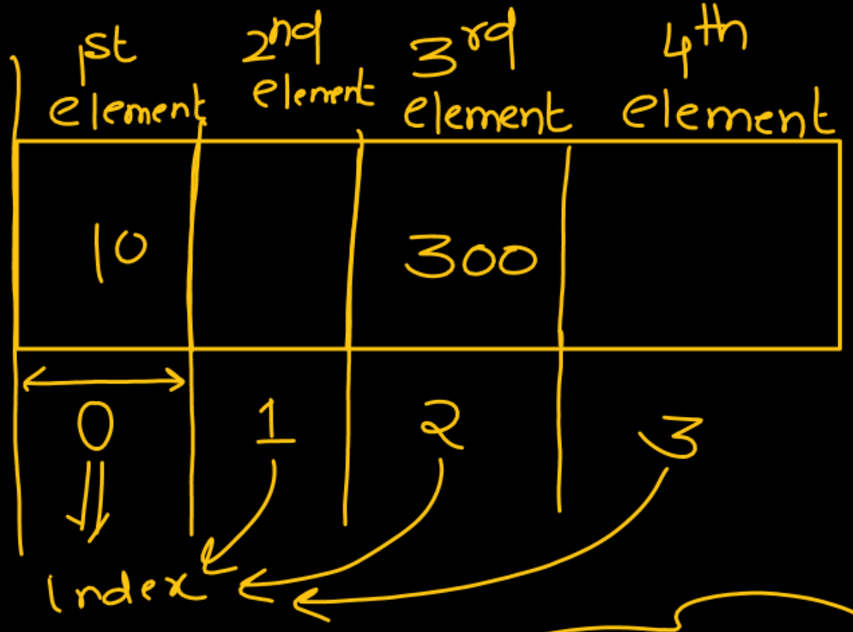
$\Rightarrow$ Roll no.

```
int a[4];
```

```
a[0] = 10;
```

index of 1st element

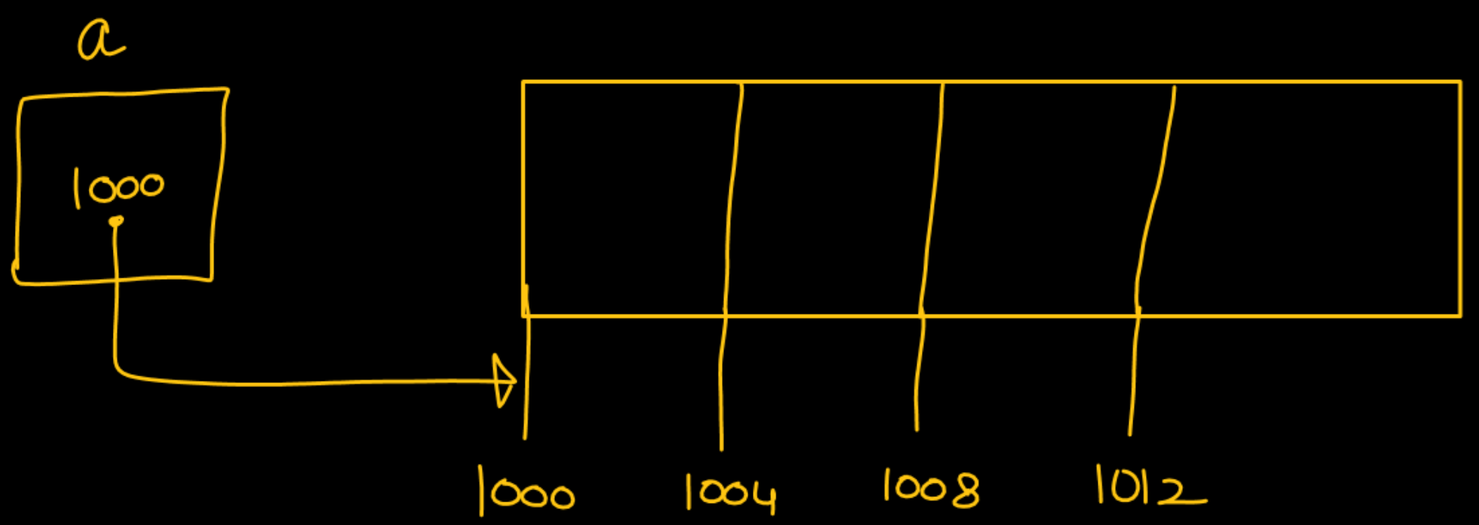
```
a[2] = 300;
```



(unique id. no.)

In C prog. index always starts from 0

int a[4];

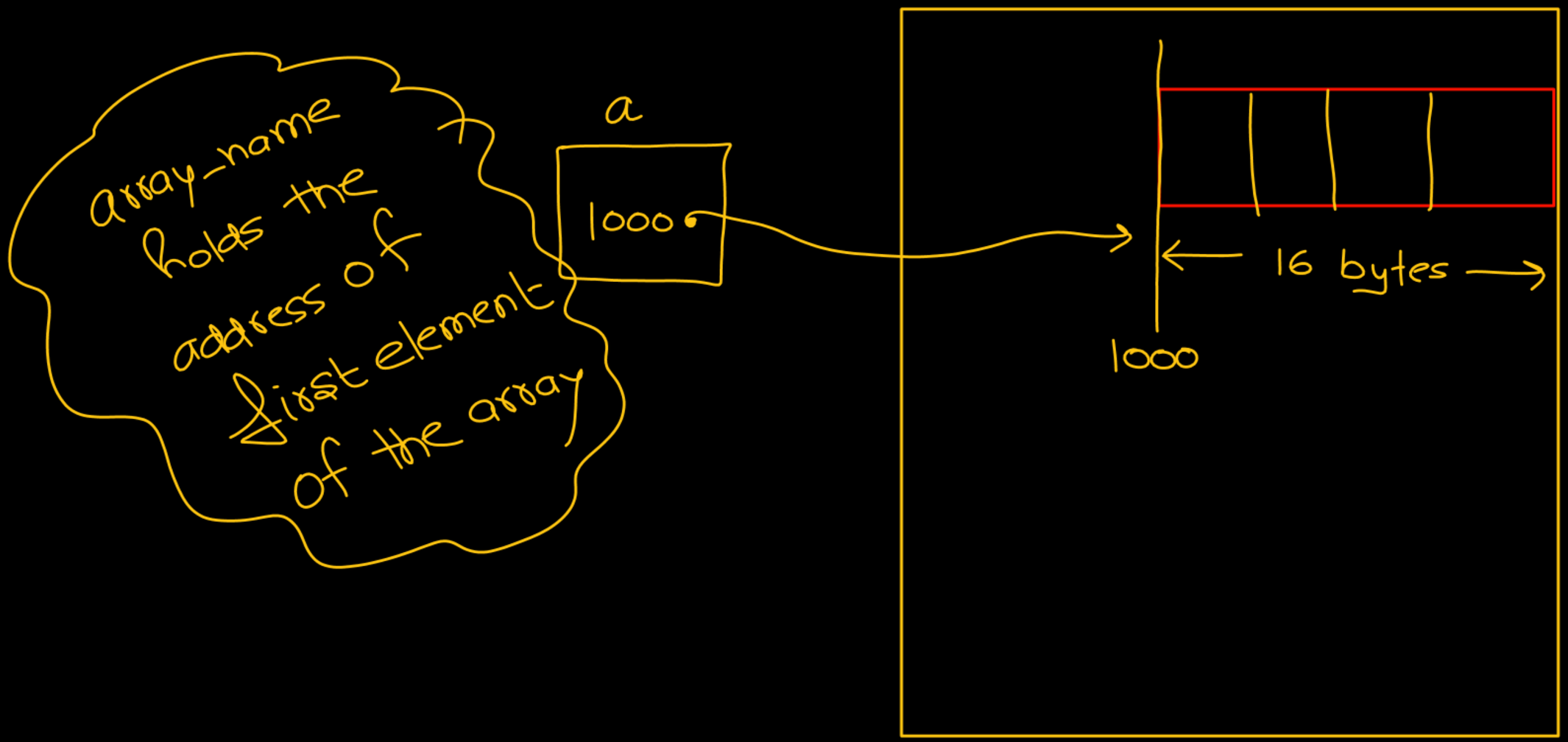


Array name ⇒ address of first element of the array

Constant Add.

int a[4];

4 × 4 ⇒ 16 bytes

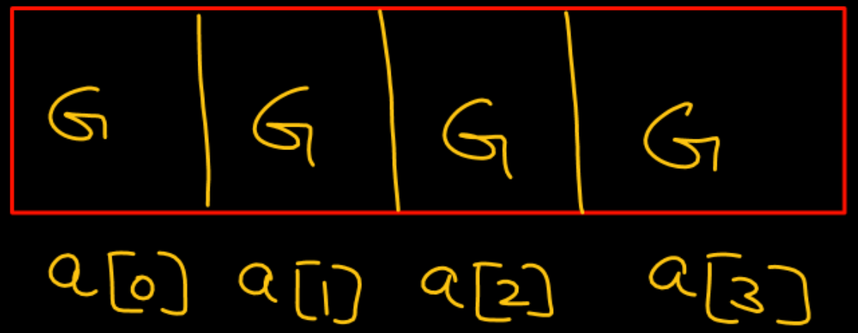


```
void main(){
   int a ;
 printf("%d",a);
}
```

Garbage

```
void main()
   {
    int a[4];
 printf("%d",a[0]);
}
```

Garbage



```
int a[4];
```

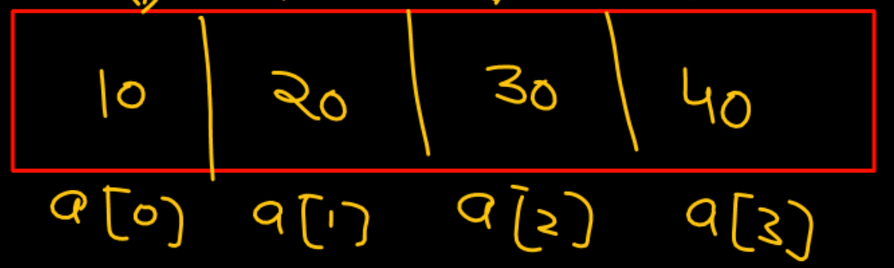
a[0]
a[1]
a[2]
a[3]

collection of 4

```
int a,b,c,d
```

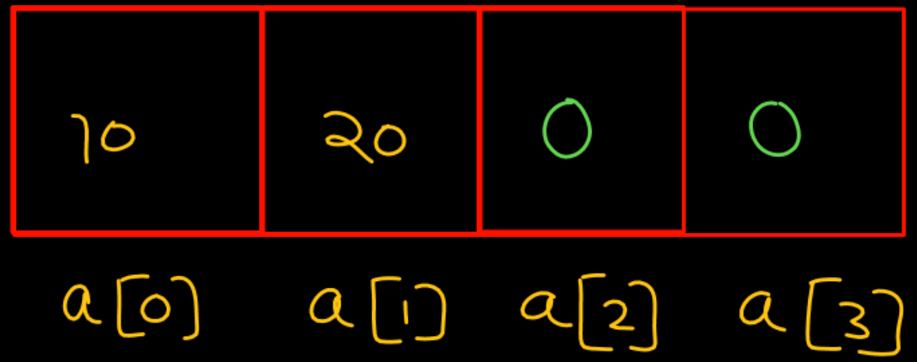
```
int a = 1;
```

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

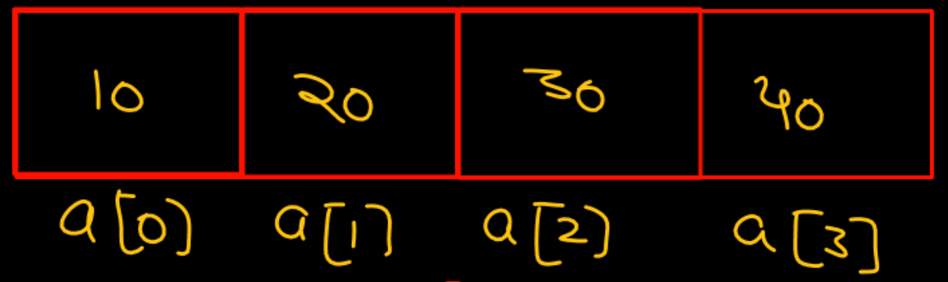


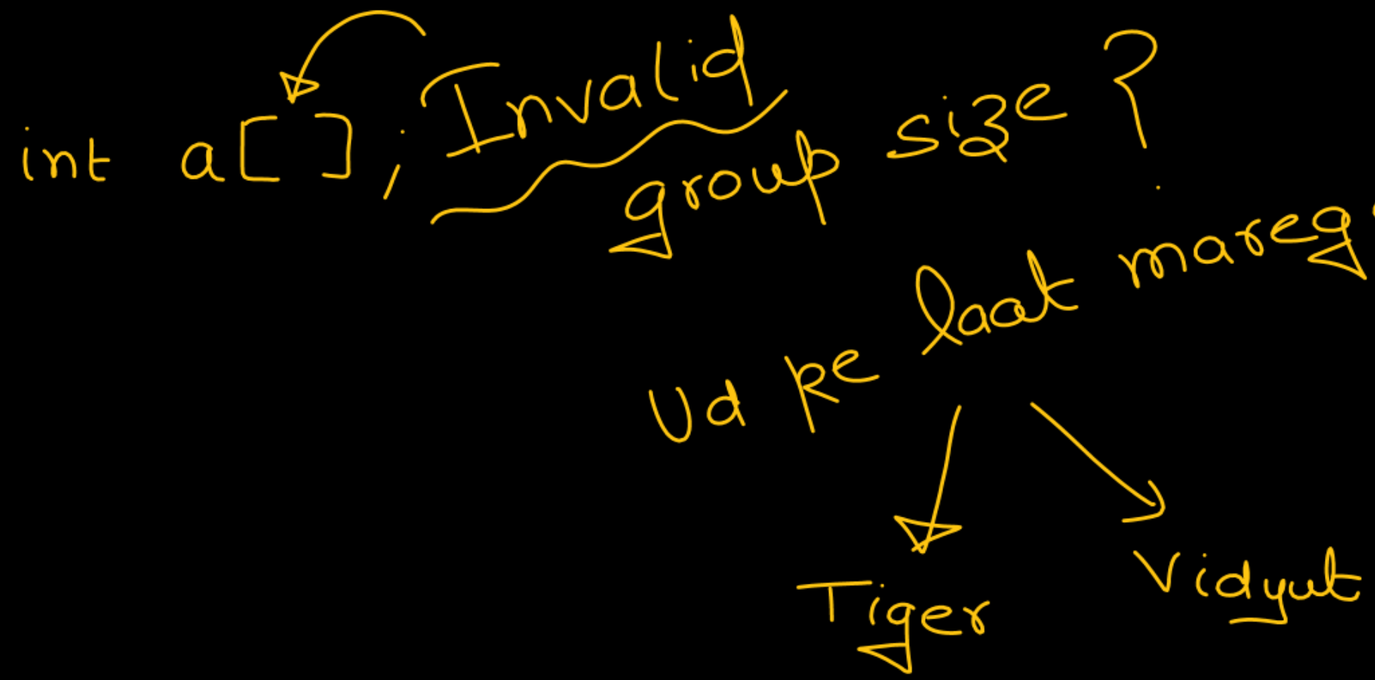
1. `int a[4] = {10,20,30,40};` ✓

2. `int a[4] = {10,20};` ✓



3. `int a[4] = {10,20,30,40,50,60};` ignored



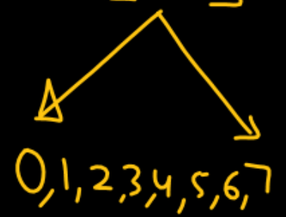

int a[];
Invalid group size?
Ud Re laak marega
Tiger
Vidyut

1.) `int a[] = {10, 20, 30};`

array_size ⇒ 3

`int a[3] = {10, 20, 30};`

2.) `int a[8]`



`int a[SIZE]`

```
int a[2] = {100, 90, 10, 20, 50};
```

ignored

```
int a[2+2];
```

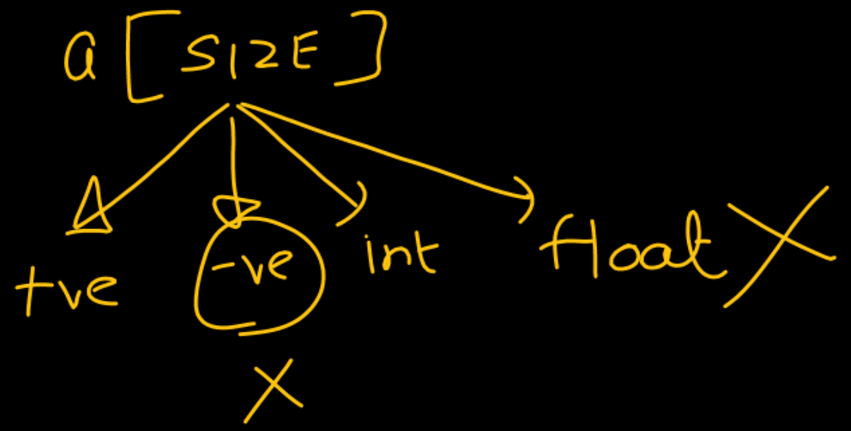
SIZE

```
int a[2x3];
int a[4xsizeof(int)];
```

a[4x4]

```
#define SIZE 10

void main(){
    int a[SIZE];
    ===
}
```



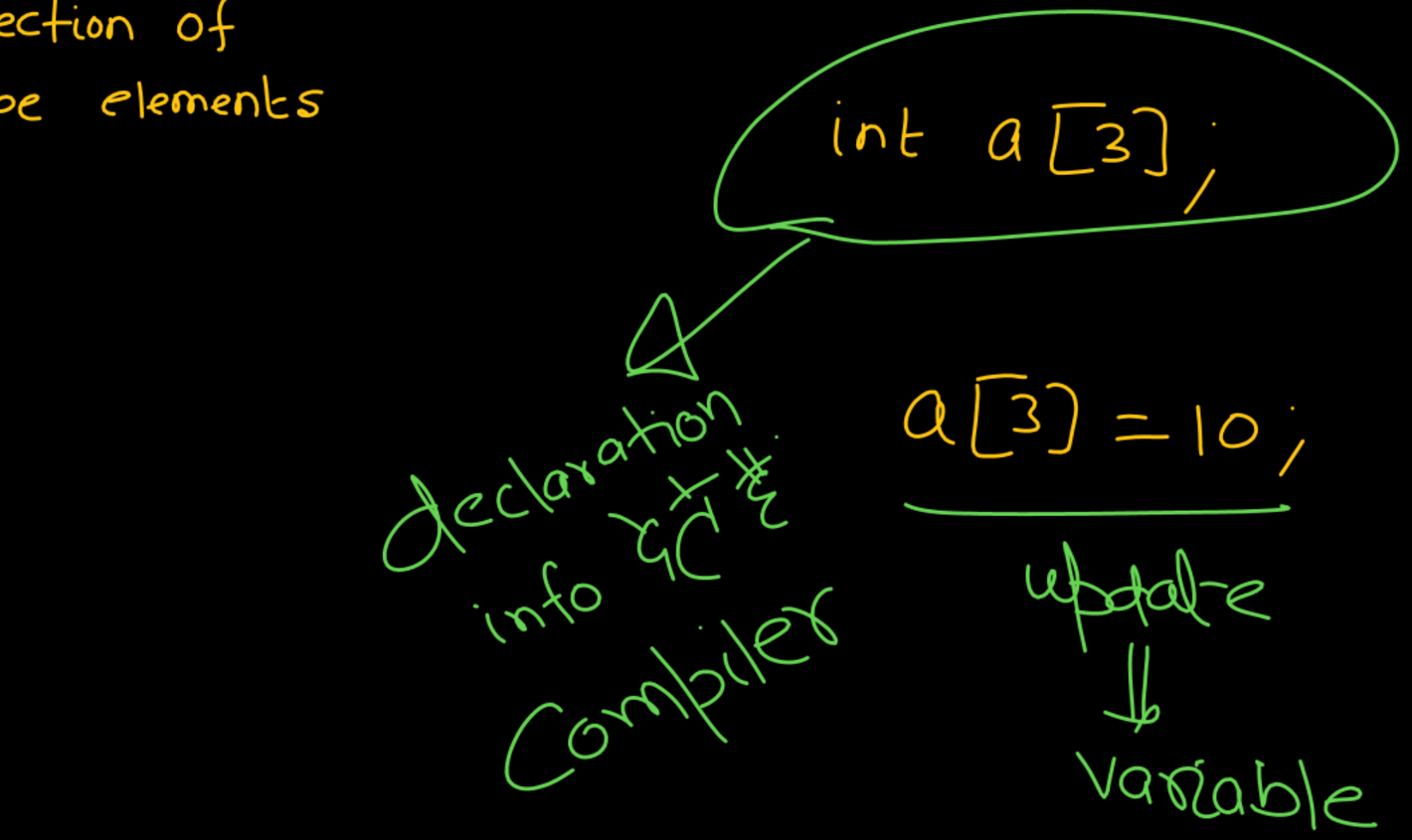
unsigned int > 0

int a[0]; ⇒ undefined

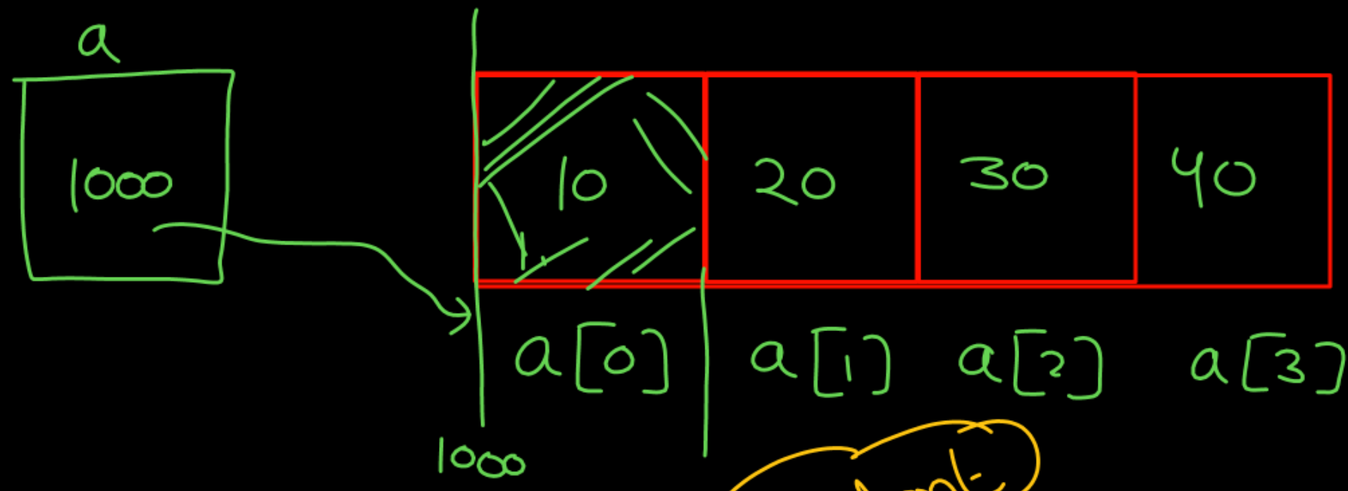
-ve ⇒ online compiler ✓

Collection of

① Similar type elements



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



① Lvalue = Rvalue ;

Lvalue → Constant ✗

Constant

Invalid

array_name = ;

Lvalue

① Array name can not be lvalue of any assignment Statement.

```
Void main(){
    int a[4] = {10,20,30};
    printf("%d",a[0]);
}
```

✓

```
Void main(){
    int a[4];
    a = {10,20,30};
    printf("%d",a[0]);
}
```

Cannot be Lvalue

(2) Array-name ⇒ constant

2++ ; Invalid

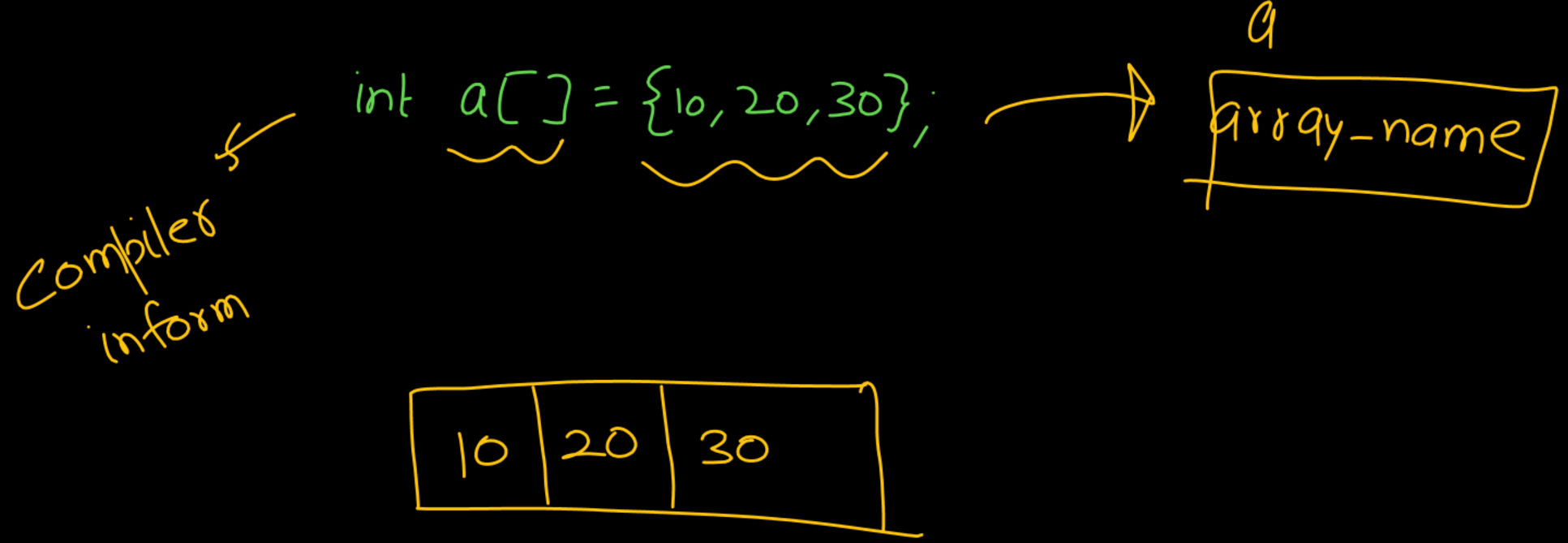
Array-name++ ;

Array-name-- ;

++ Array-name ;

-- Array-name ;

} Invalid


int a[] = {10, 20, 30};
Compiler inform
a
array_name
10
20
30

1. Collection of homo. type of elements.

2. Element are stored seq. one after another.

3. Array_name ⟹ constant

4. Array_name ⟹ Address of first element of array.

5. ++Array_name
   --Array_name
   Array_name++
   Array_name--
   } Invalid

Array_name = 

Invalid

6. int A[] ✗ invalid

7. int A[] = {10,20}; ✓

8) Relative addressing

9) int a[]

lang → rules

↓

Compiler

```
int a = {1};
```

Valid in C

```
int a[3] = { 10, 20, 30 }
int a[3] = { 10 };
int a[3] = { 10, 20, 30, 40, 50 }
```


THANK YOU GW
SOLDIERS !