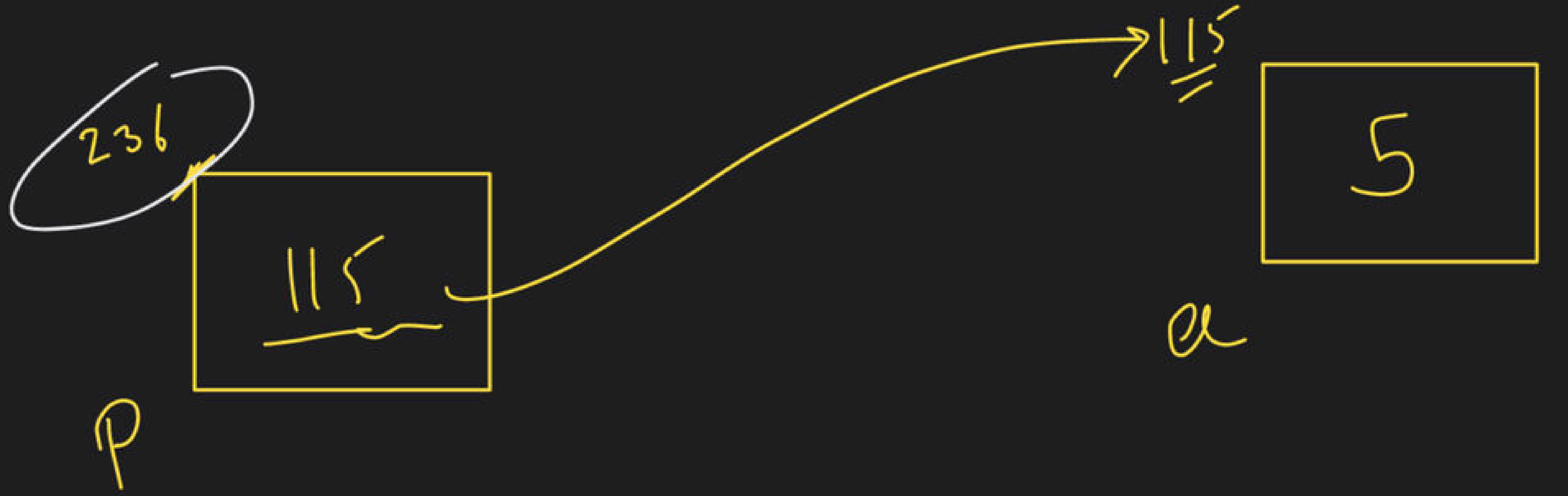
Pointers - Level 2

Special class

Love Babbar • Mar 4, 2023

we will crete a new Context

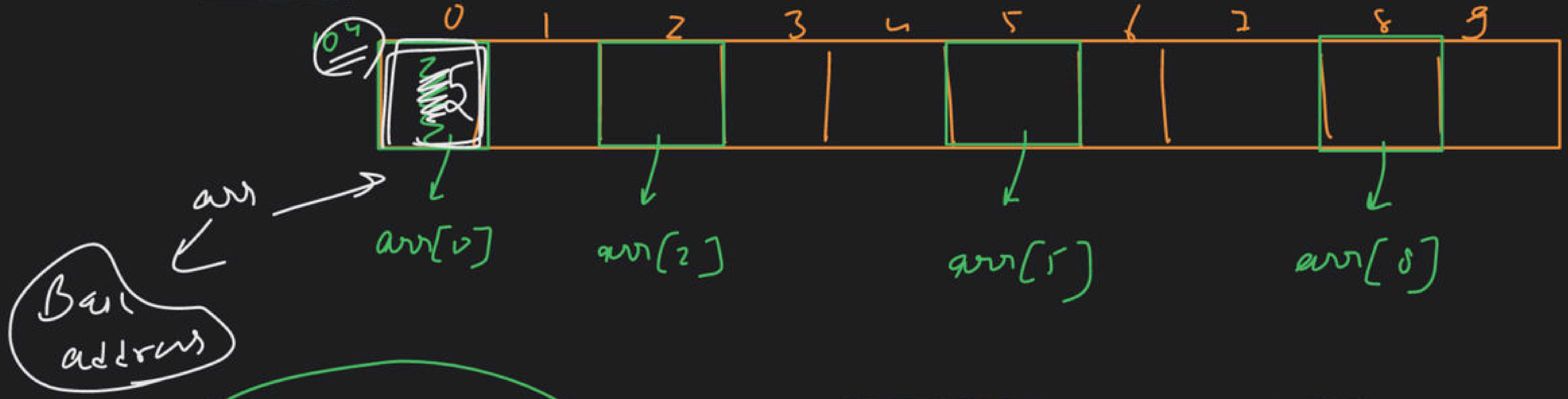
```
int a = 5;
int* p = &a
```



12:00

12:15

int arr[10];

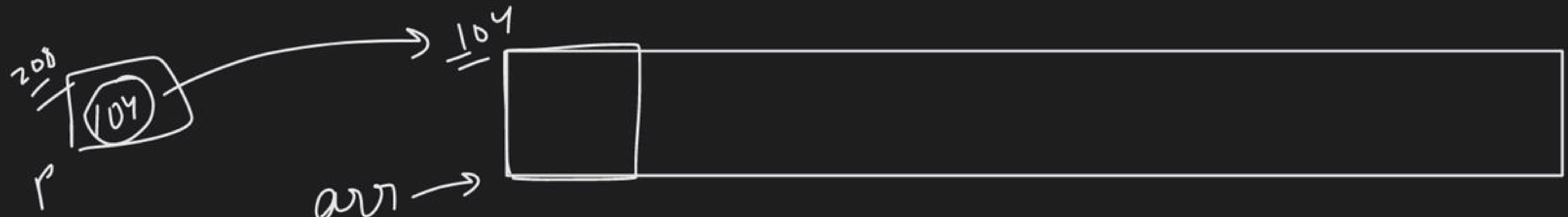


&arr[0]

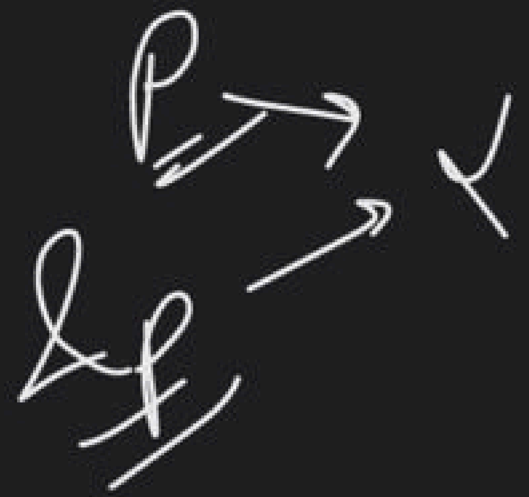
arr[0] → 5

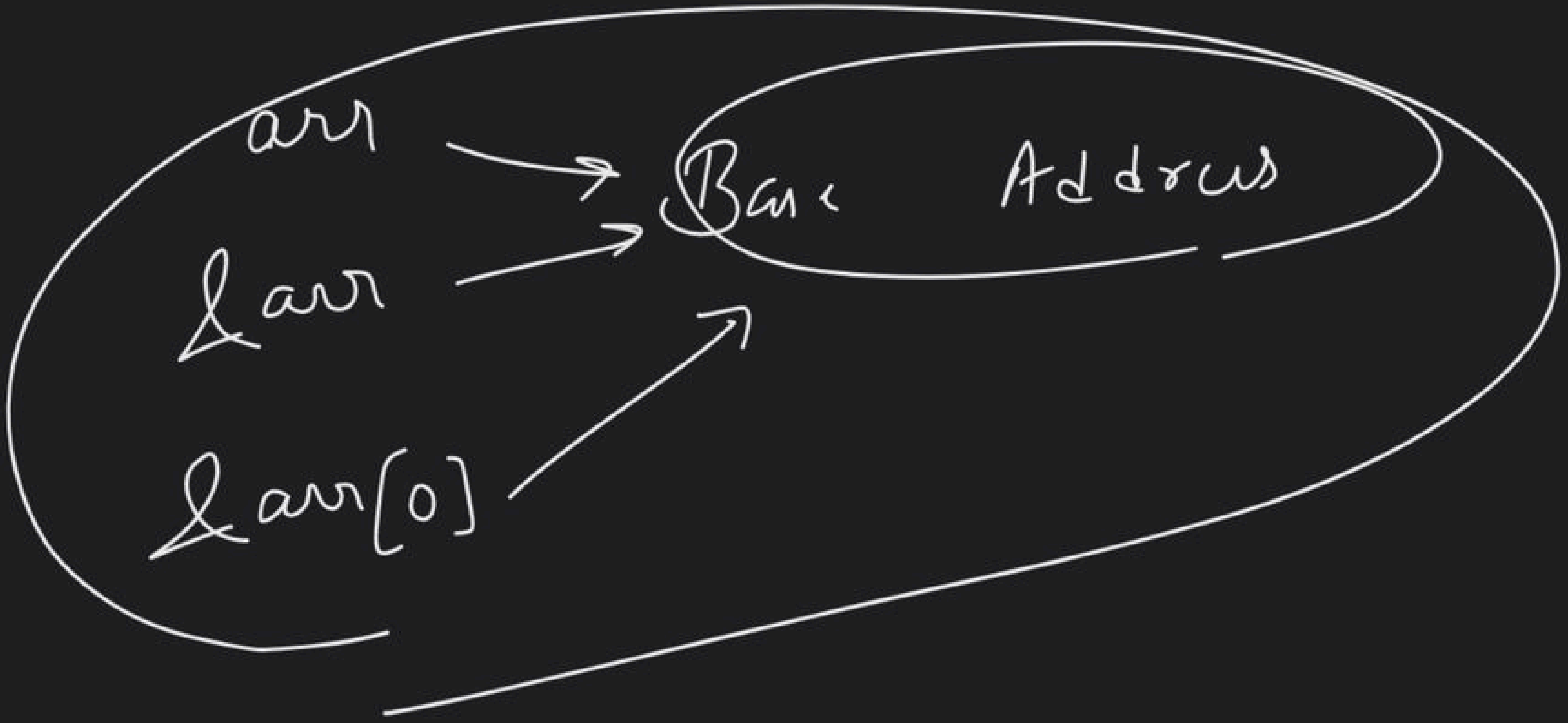
&arr[0] → 104

arr → 104

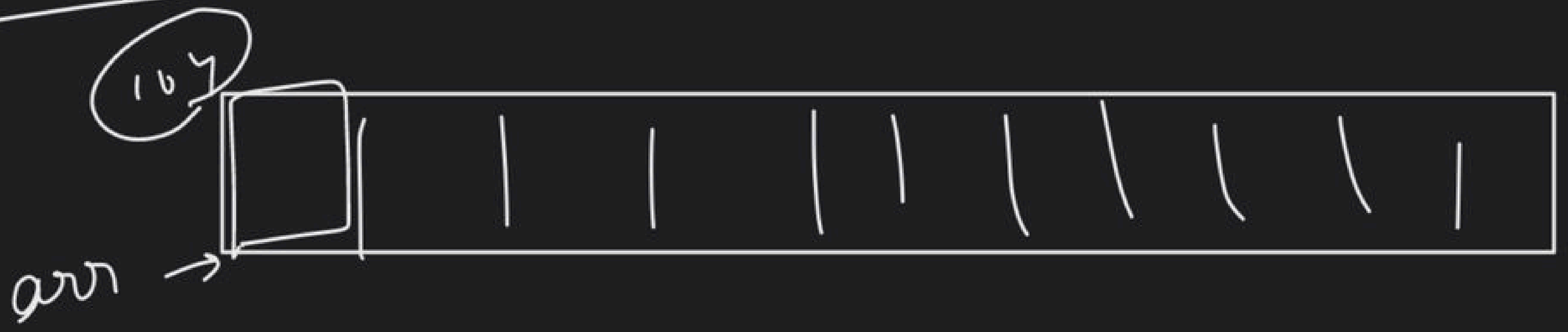


```
int *p = arr;
```





int arr[10]



S.T

arr → 104

arr → B.A

&arr → 104

why?

&arr[0] → 104

arr
&arr → why same

int arr[10]

s.t

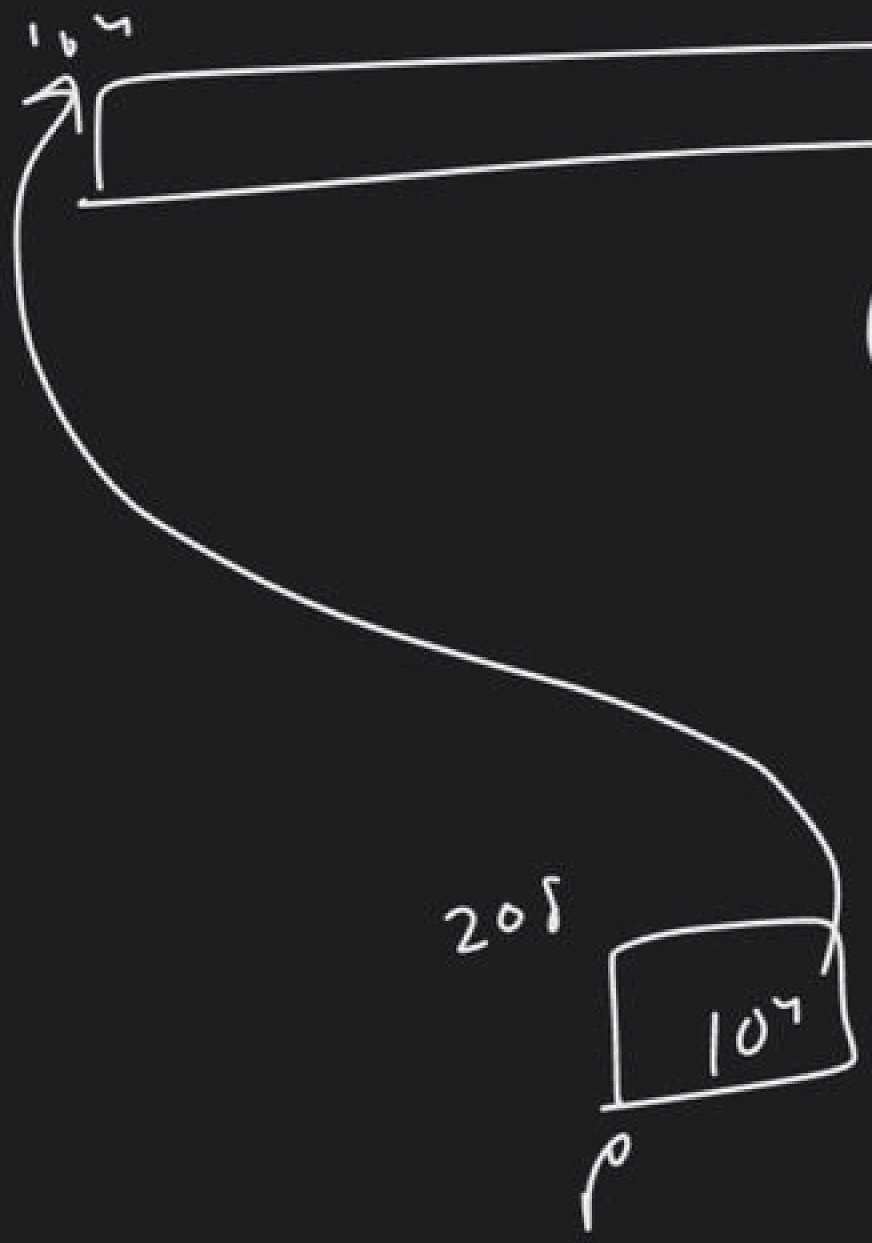
arr → 104

&arr → 104

int *p = arr

p → 208

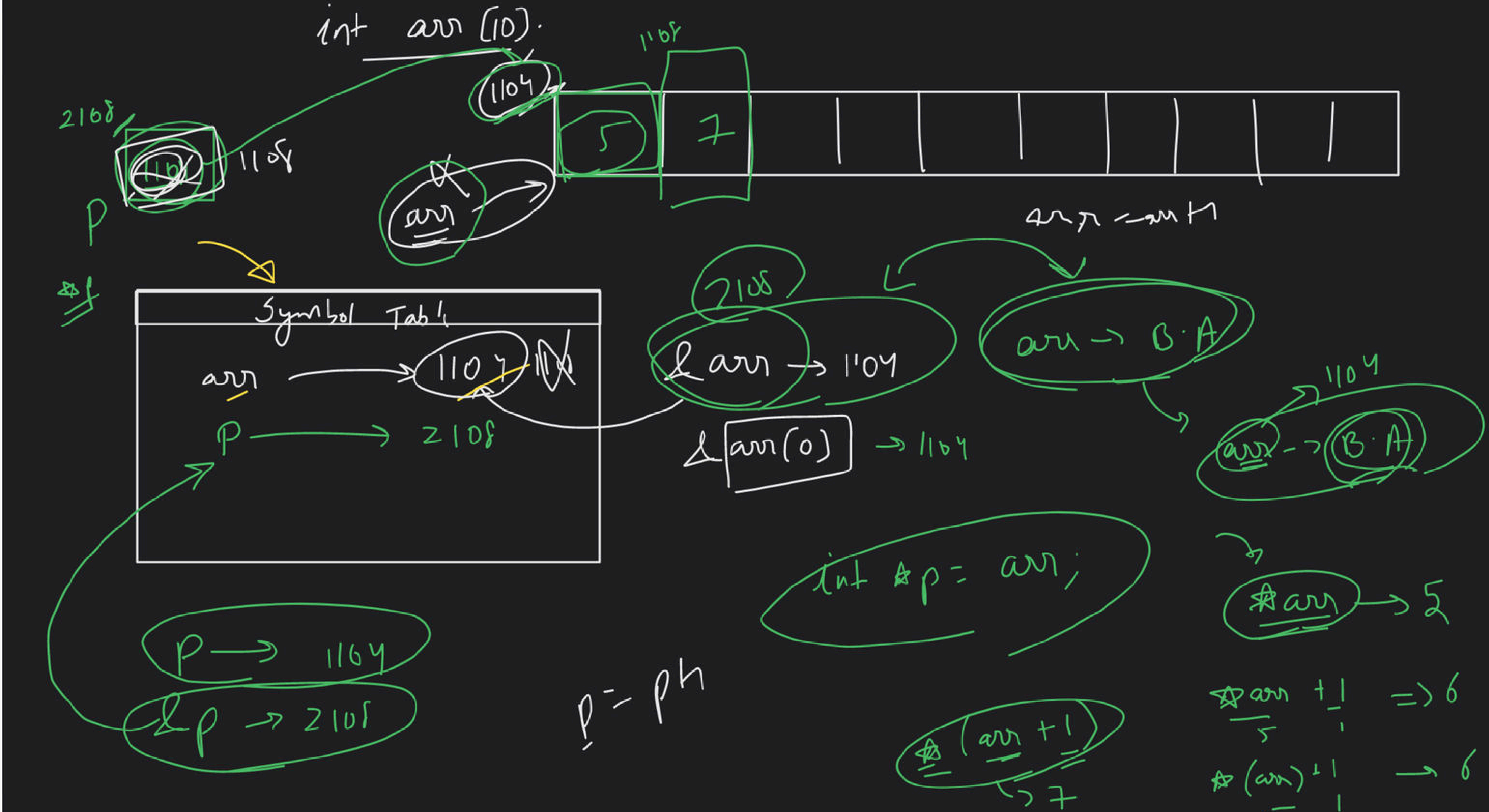
arr → 104

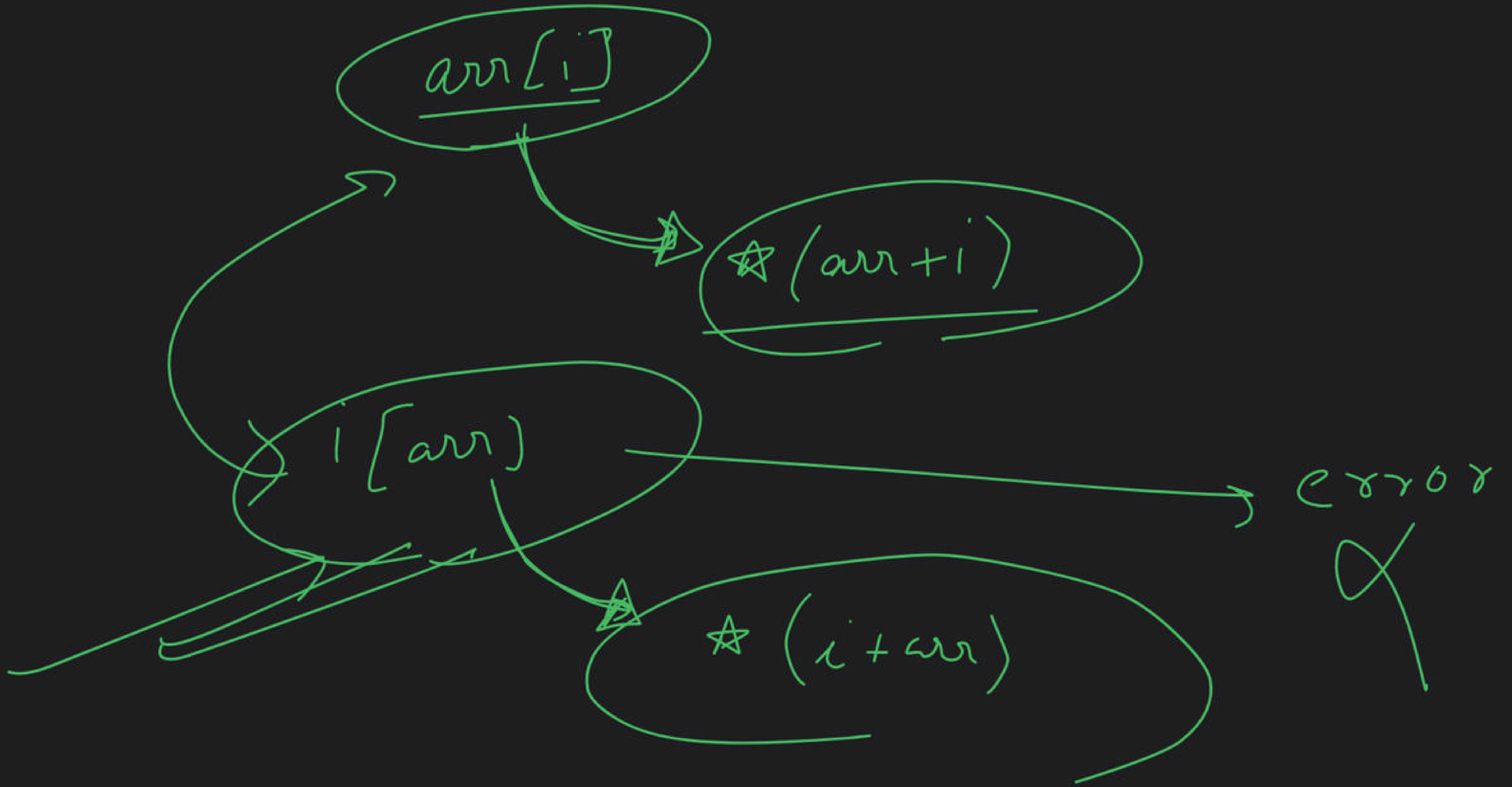


&arr → B.A

arr
&arr
&arr[0] → B.A

&p

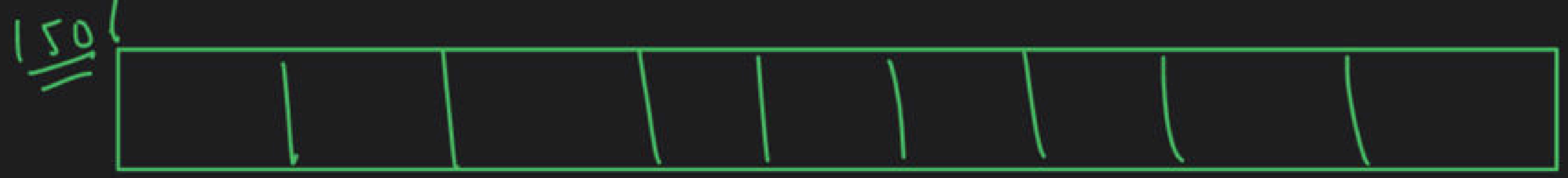

int arr[10].
1108
1104
5
7
2108
1104
1108
P
arr
arr → arr+1
*p
Symbol Table
arr → 1104
P → 2108
2108
&arr → 1104
&arr[0] → 1104
arr → B.A
1104
arr → B.A
int *p = arr;
P → 1104
&p → 2108
p = p+1
*arr → 5
*(arr+1)
→ 7
*arr + 1 => 6
5 1
*(arr)+1 → 6
1


arr[i]
*(arr+i)
i[arr]
*(i+arr)
error

$$\begin{tikzcd}
arr[i] \arrow[r] & *(arr+i) \arrow[d, leftrightarrow] \\
i[arr] \arrow[r] & *(i+arr)
\end{tikzcd}$$

$$\begin{tikzcd}
*arr \arrow[d] \\
*arr \arrow[d] \\
*(arr+0) \arrow[d] \\
arr[0]
\end{tikzcd}$$

int arr[10];



arr = arr+2 → why

arr

&arr

arr[0]

&arr(0)

int *p = ~~&~~arr;

p

&p

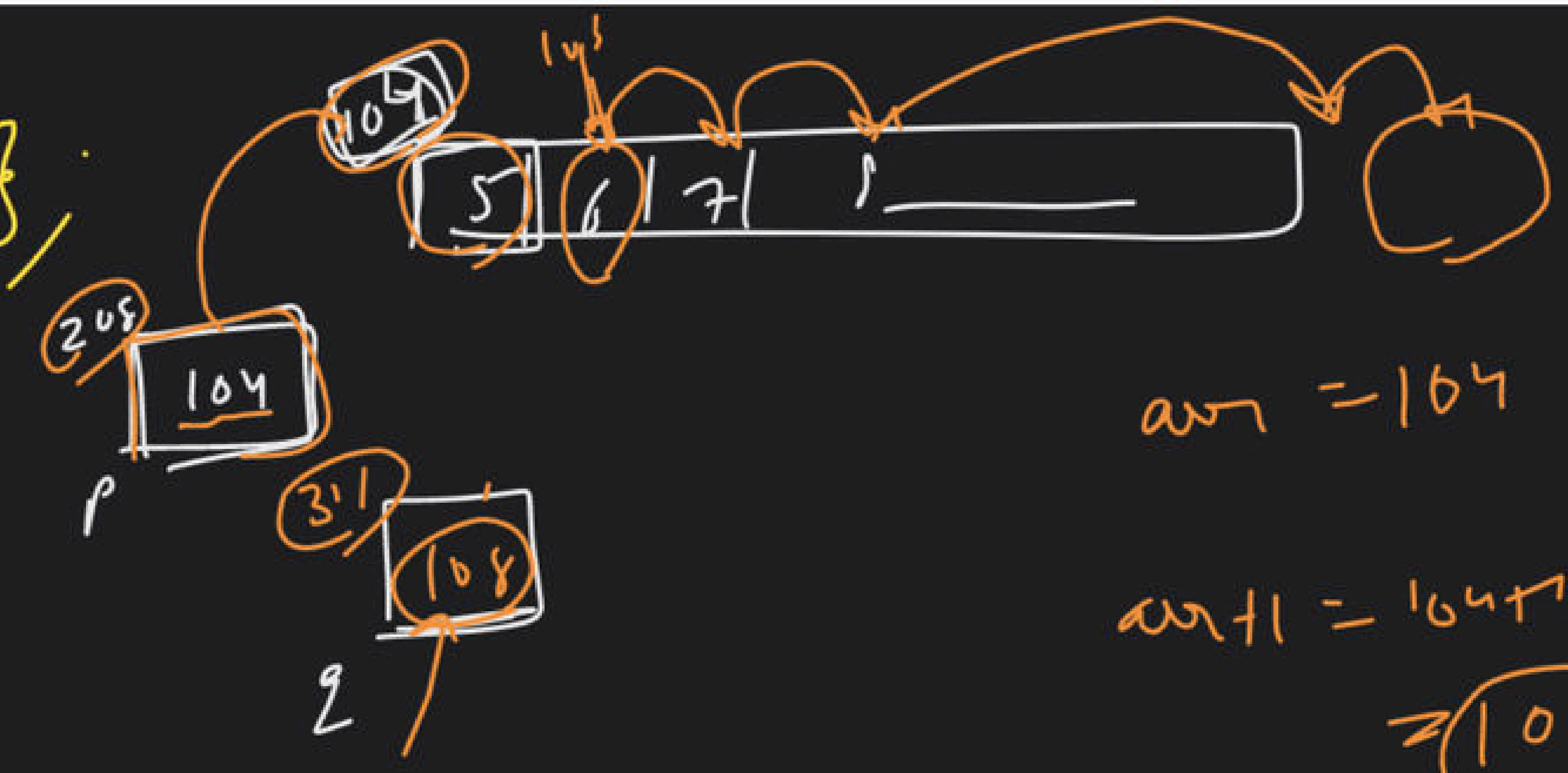
*p

*(p+3) → p[3]

int arr[4] = {5, 6, 7, 8};

int *p = arr;

int *q = arr+1;
104



arr = 104

arr+1 = 104+4
= 108

→ cout → arr → 104

&arr → 104

arr[0] → 5

&arr[0] → 104

p → 104

&p → 208

*p → 5

q → 108

96%.

&q → 316

*q → 6

*(p)+1 → 5+1 = 6

*(p)+2 → 5+2 = 7

*(q)+3 → 9

*(q+4) → Segfault
Out of Bound

Pointer & Array

①

int arr[10]

sizeof (arr) → 40 byte

arr
&arr → B.A

int * p = arr

sizeof (p) → 8 → why

array

→ array k name → sizeof( )

↳ total space taken by array

pointr

→ sizeof(——) → 8 / 4

p = p+1 →

→

108

~~104~~

(2)

arr = arr + 1 → ×

p = p + 1 →

Difference

↳ int arr[10]

↳ sizeof(arr) → 4×10 = 40

→ total space taken by arry

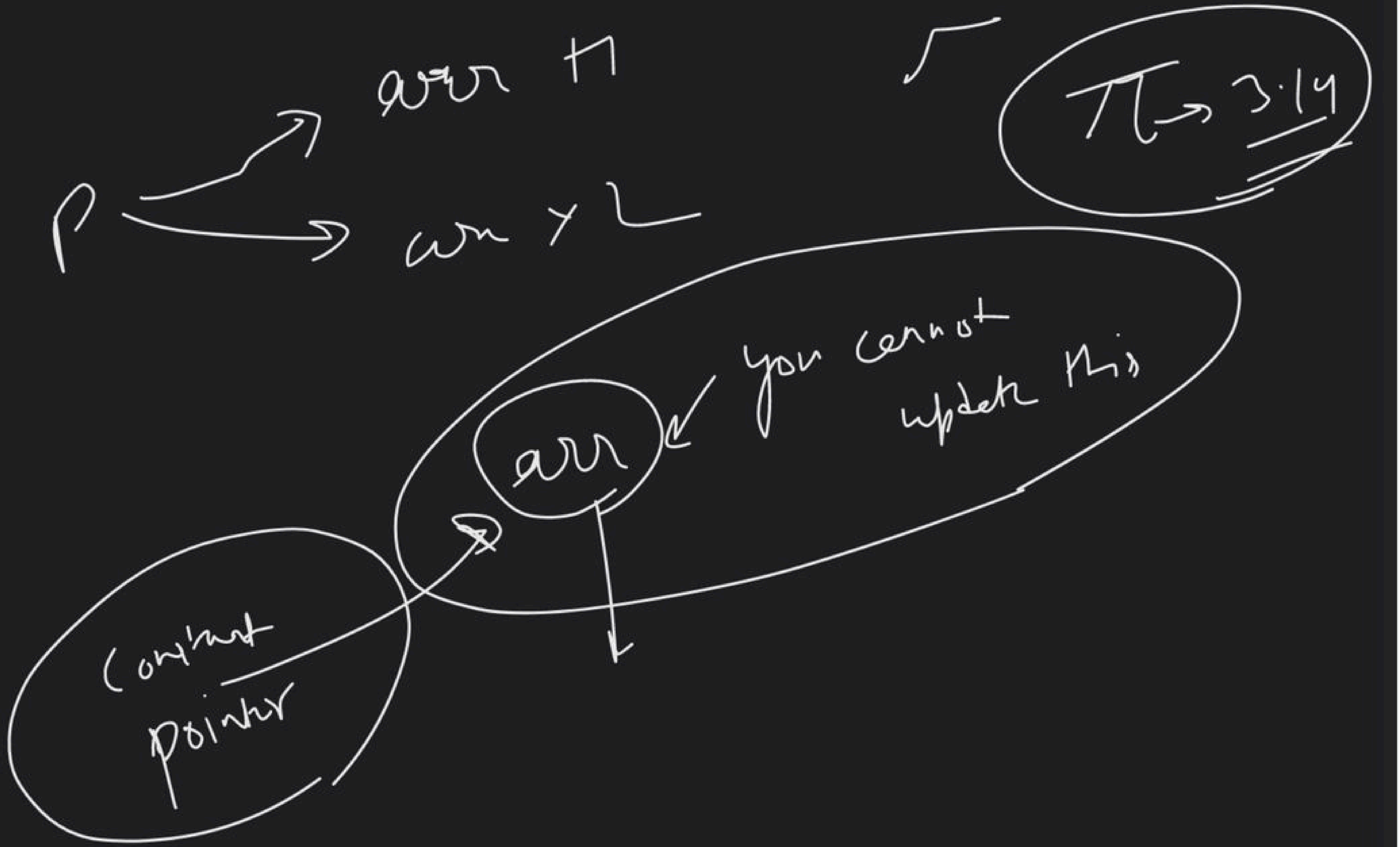
↳ int *p = arr

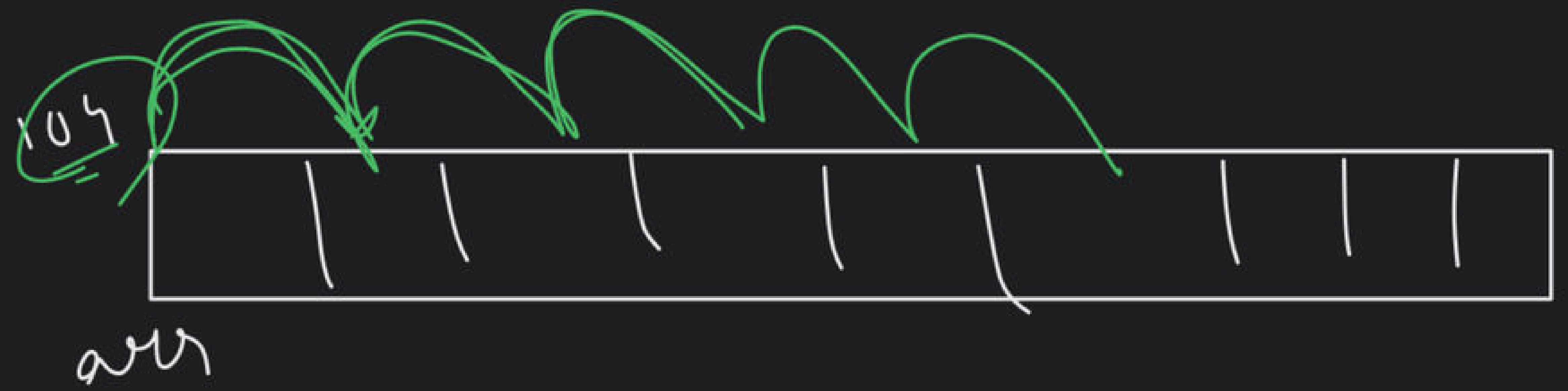
↳ sizeof(p) → 8/4

↳ total space taken by pointer

↳ curr = curr + 1

↳ p = p + 1


p
arr +1
arr + 2
π → 3.14
you cannot update this
arr
Constant pointer



~~arr = arr+1~~

p

point

arr + 1 → 108 → 104 + 1×4 = 108

arr + 3 → 116 → 104 + 3×4 = 116

arr + 5 → 124 → ~~[illegible]~~

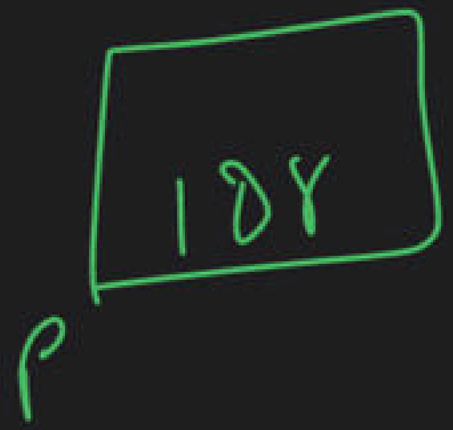
104 + 5×4 = 124

arr = 104

cout << arr+1;

108

int *p = arr+1;



updation

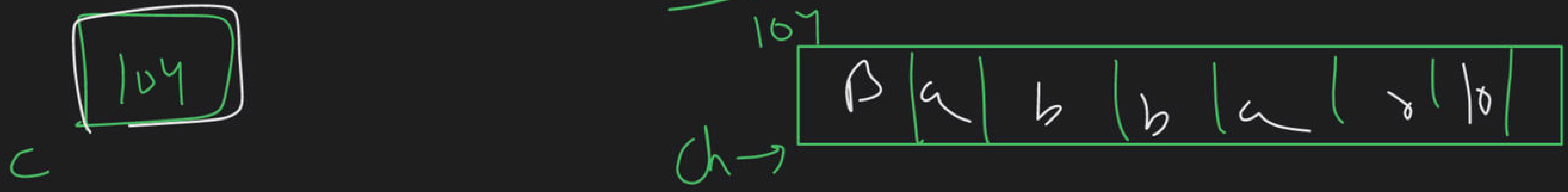
arr = arr+1;

fail
error

constant
pointer

char array

char ch[10]



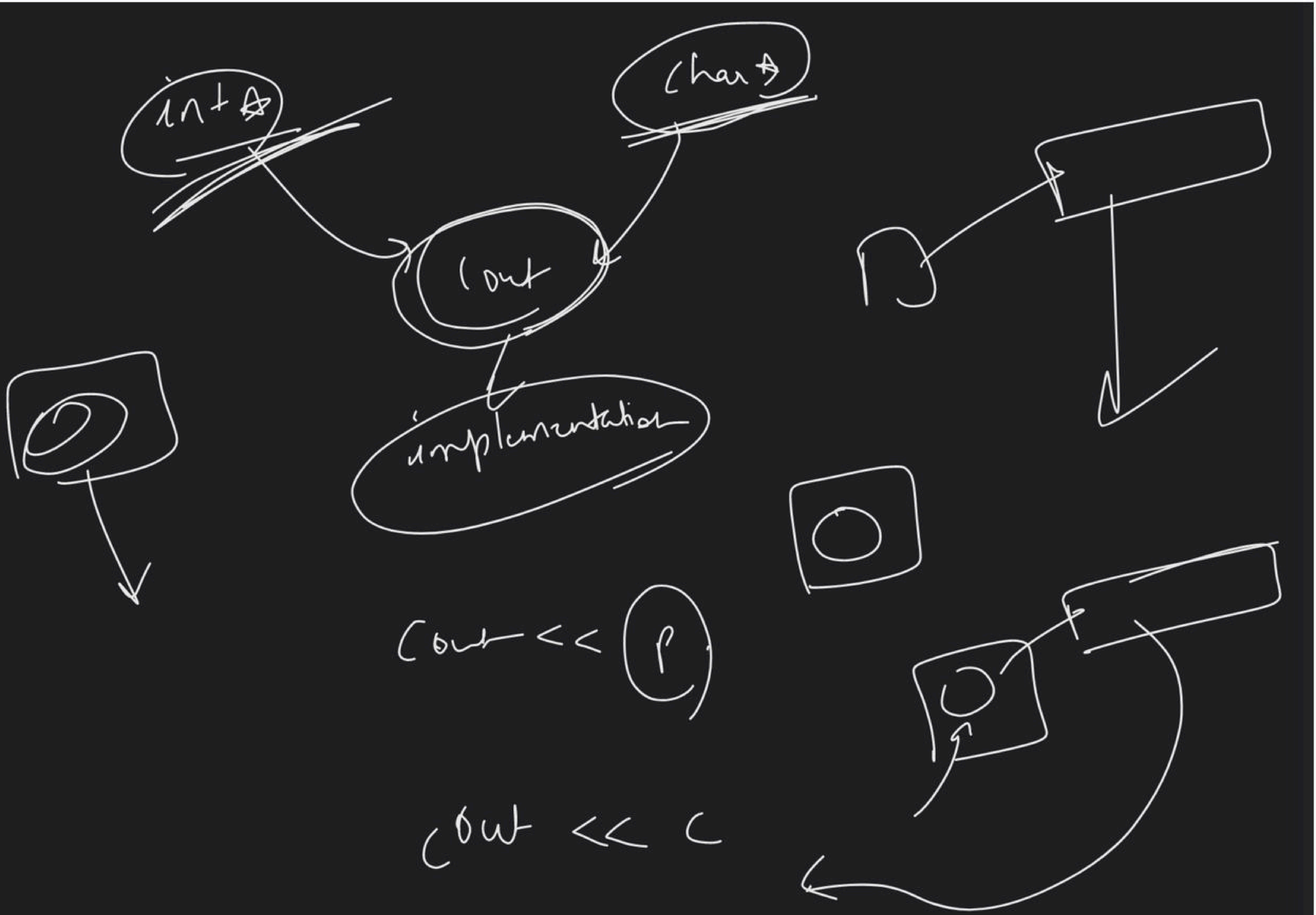
char ~~int~~ * c = ch

&ch

cout << c;

104

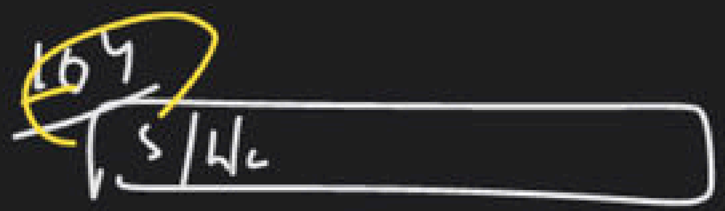
Babbar


int*
char*
cout
implementation
cout << p
cout << c

$$*C = *(C+0)$$

$$\begin{tikzcd}
= c[0] \arrow[r] & 0^{th} \text{ index} \arrow[d] \\
& B
\end{tikzcd}$$

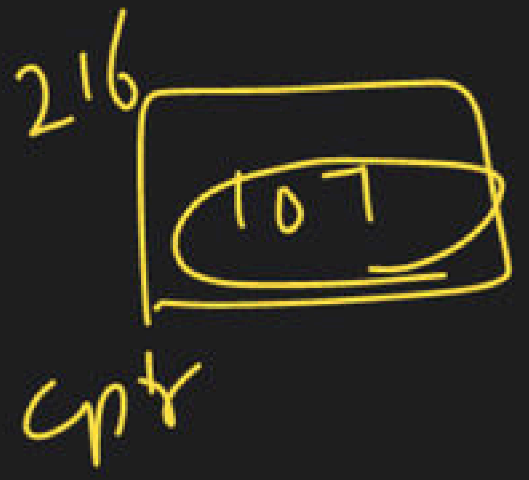
$$\begin{tikzcd}
C \arrow[r] \arrow[dr, bend right] & \text{addr} \\
& \text{entire string}
\end{tikzcd}$$



char name[10] = "SherBano";

char * cptr = &name[0];

2min



cout → name → SherBano

→ &name → 104

→ *(name+3) → name[3]

↳ r

→ cptr → ~~Babbar~~ SherBano

→ &cptr → 216

→ *(cptr+3) → cptr[3]

↳ r

cptr+2 → Rukte

*cptr → *(cptr+0)

→ cptr[0] → S

cptr+8

→ Rukte

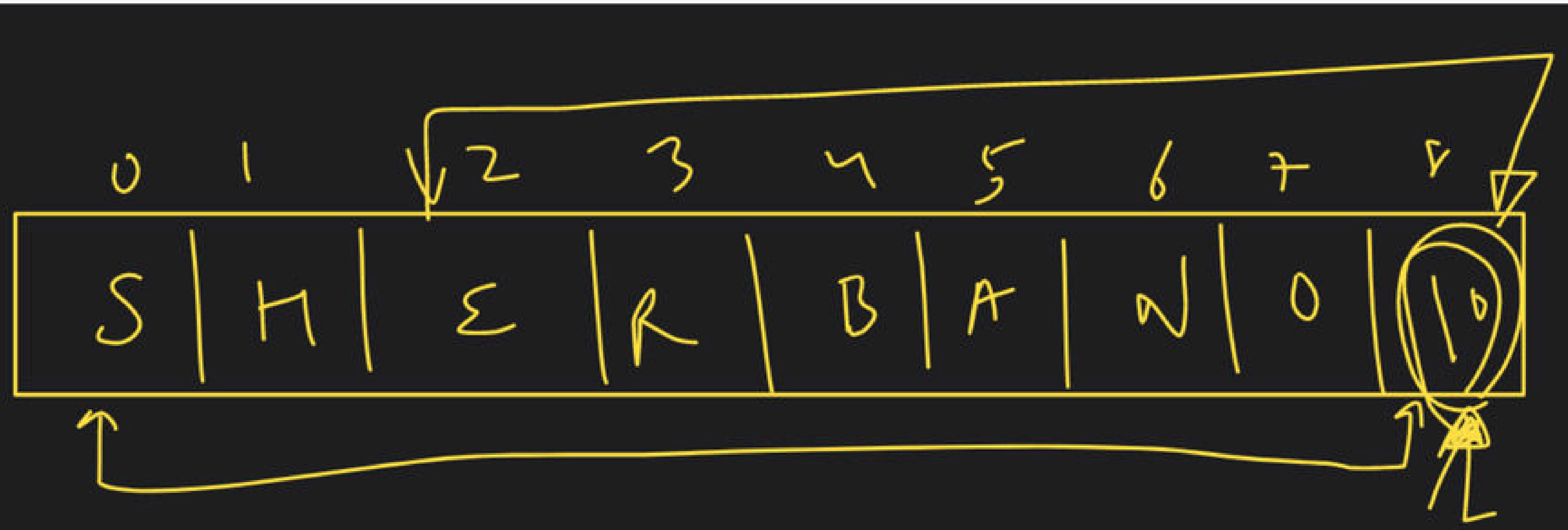
cout

*cptr

↳ *(cptr+0)

↳ cptr[0]

↳ S



cout << c << endl; → SHERBANO

cout << c+2 ; → ERBANO

c+8 →

~~it~~ ~~char~~ =

char ch = 'z'

~~char~~ =

" "

string

char ch = 'k';0OOun

char* cptr = &ch;

cout << cptr;

```
char name[10] = "Babbar";
```

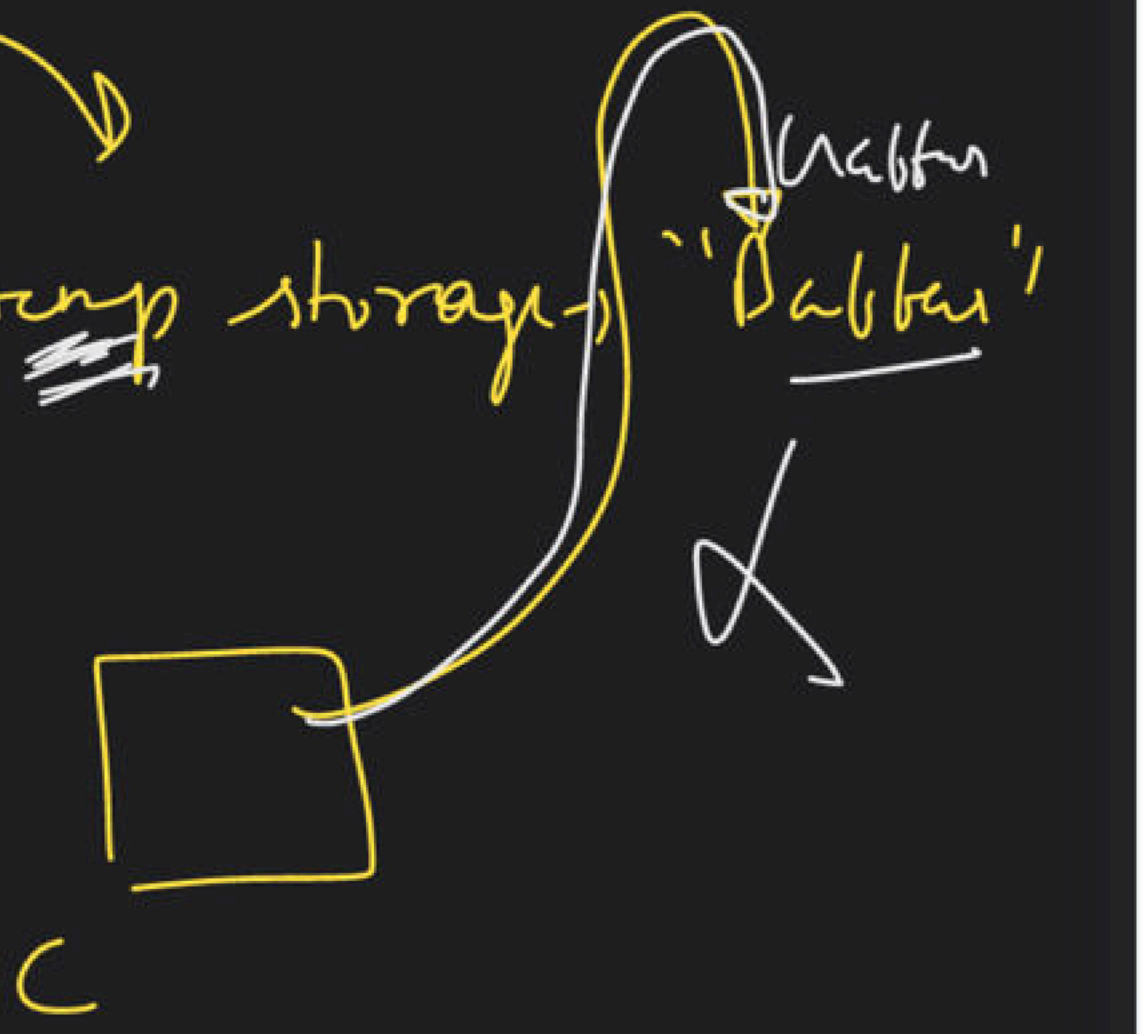
```
char* c = "Babbar";
```

BAD Practice

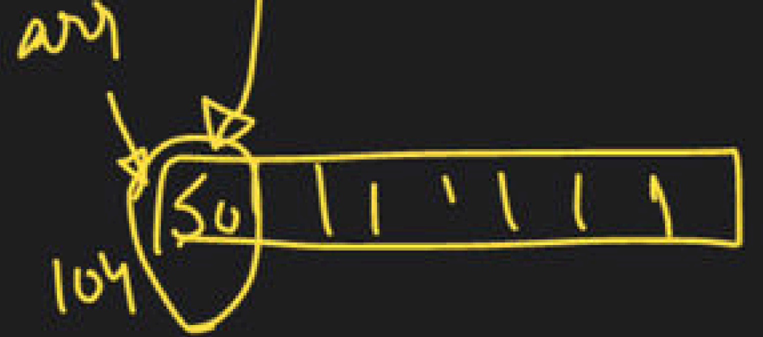
Good

1. temp storage → "Babbar"
2. memory change → copy to name array ki storage

1. temp storage → "Babbar"
2.

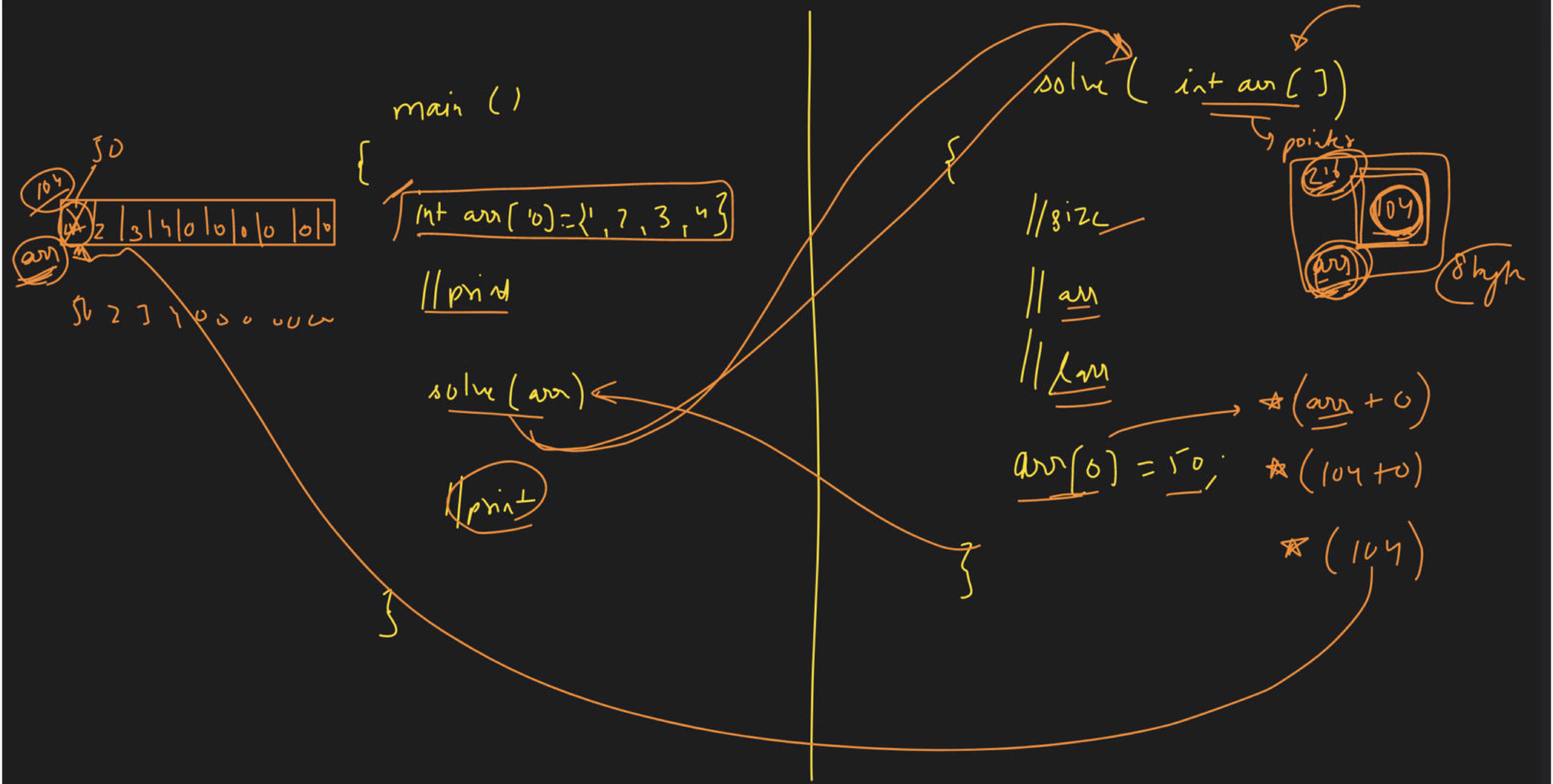


Pointers with function:-



```
main()
{
    int arr[10];

    sizeof(arr);   4x10->40

    solve(arr)

}
```

```
solve(int arr[])
{
    104
    arr

    arr[0] = 50;   -> actual arry

    *(arr+0)

}
```

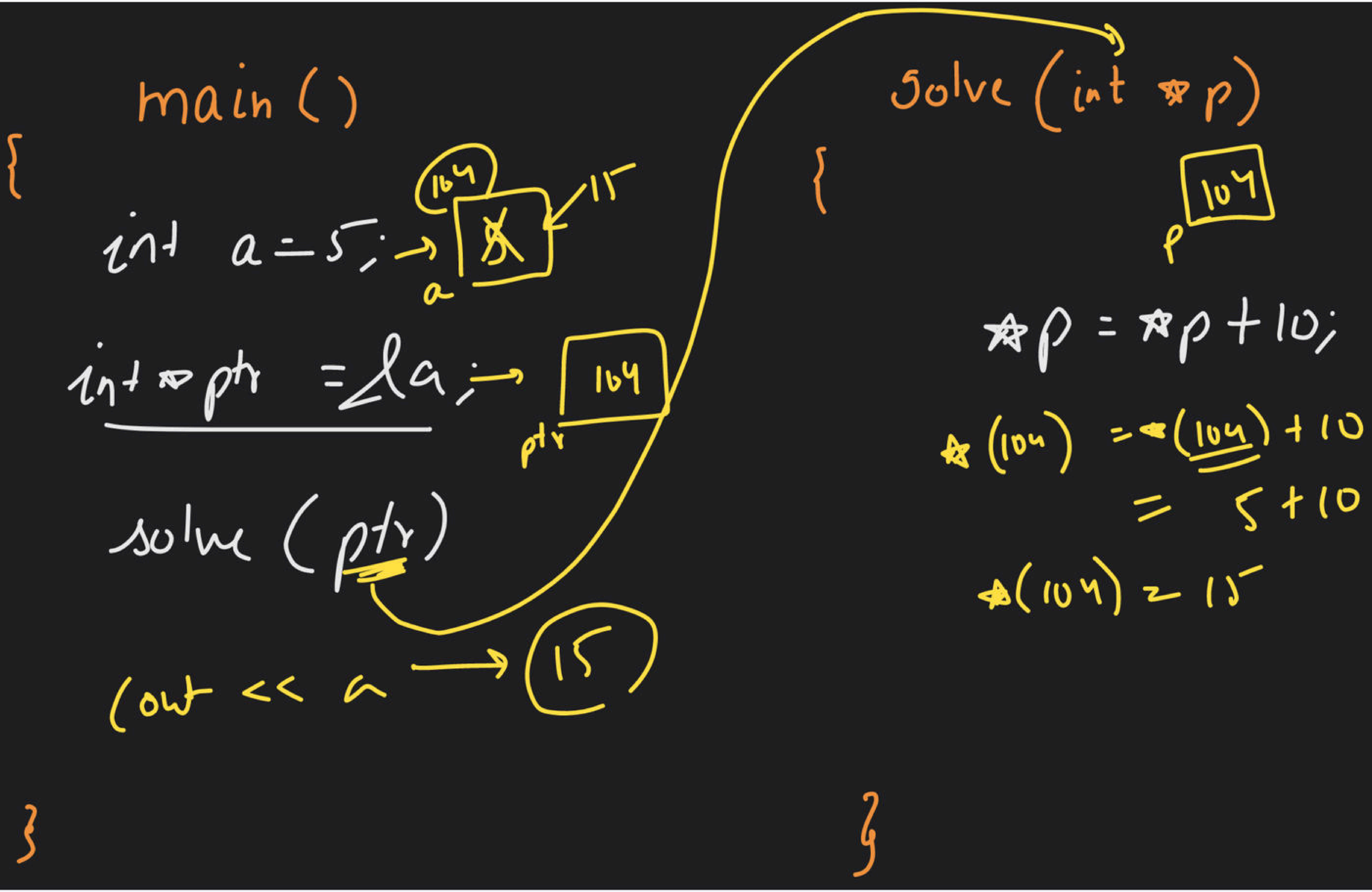


```
main ()
{
    int arr[10] = {1, 2, 3, 4}

    //print

    solve(arr)

    //print
}
```

```
solve( int arr[ ])
{
    //size

    //arr

    //&arr

    arr[0] = 50;
}
```

```
int a=5;
if (          )
{
   int a = 50
}
```

multiplename

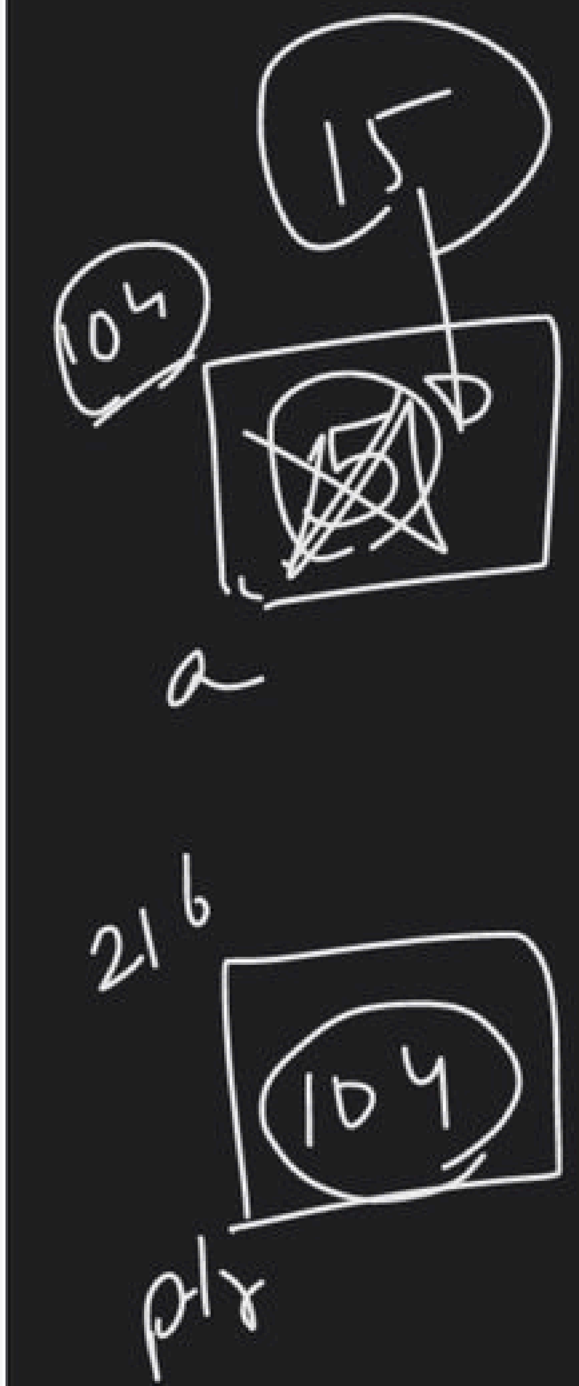


```
main()
{
    int a=5;
    int *ptr =&a;
    solve(ptr)
    cout << a
}

solve(int *p)
{
    *p = *p+10;
}
```

$*(104) = *(104) + 10$

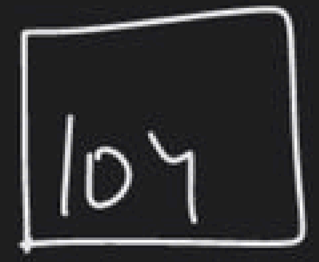
$= 5 + 10$

$*(104) = 15$



```
main( )
{
int a = 5;
int * ptr = &a;
update(ptr);
cout << a;  → 15
}
```

```
update( int * p)
{
*p = *p + 10;
}
```



p

$*(104) = \boxed{*(104)} + 10$

$= 5 + 10 = 15$

$*(104) = 15$

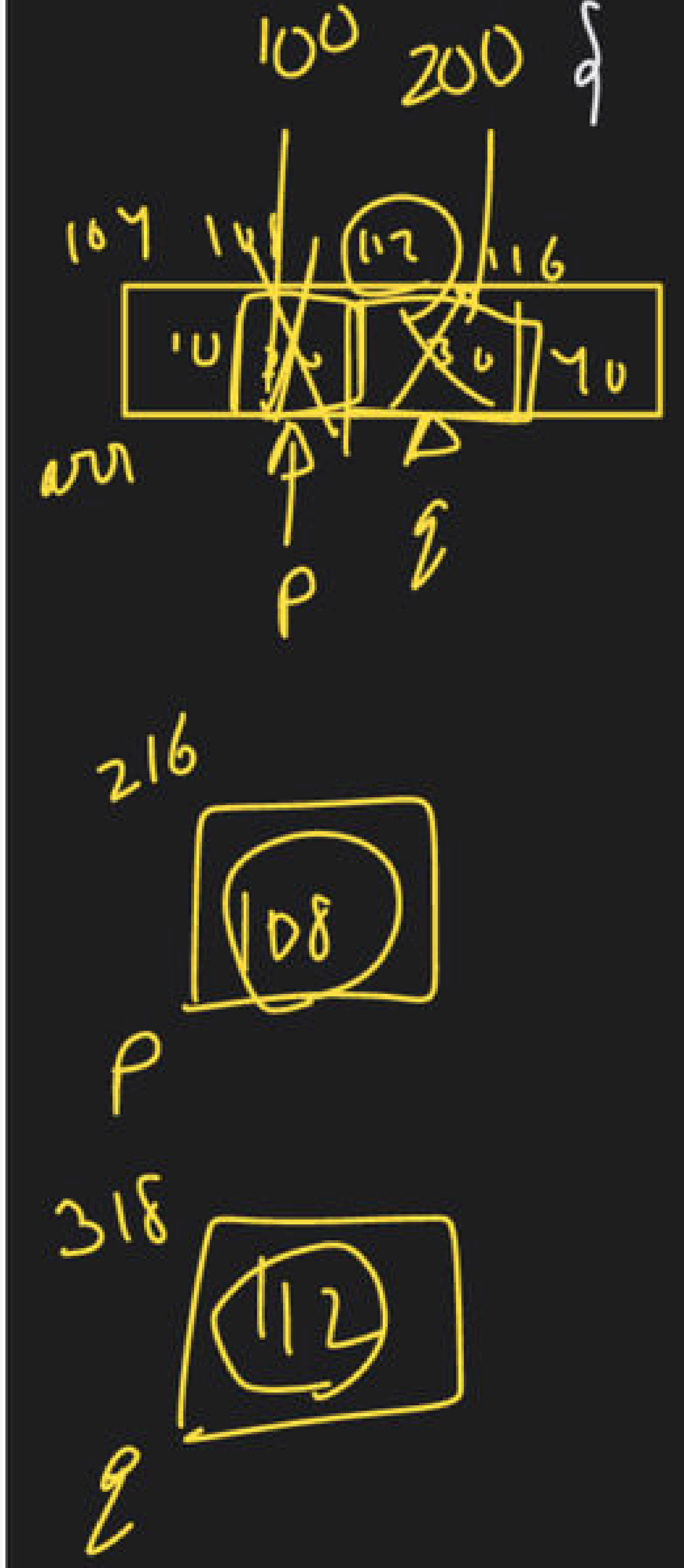
int *p = ptr → copying pointers

10 100 200 40

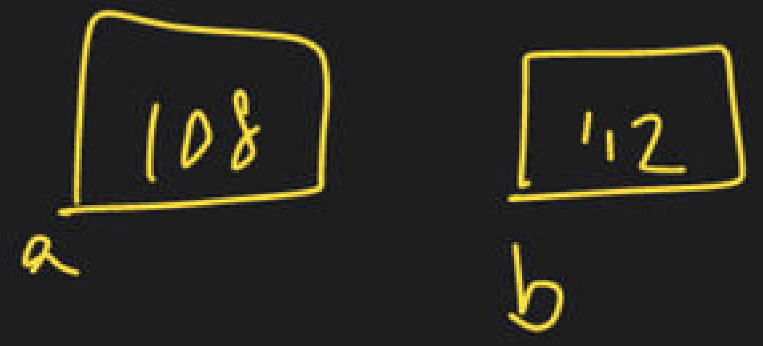
2 min

```
main ( )
{
int arr[4]= {10, 20, 30, 40]
int *p = &arr[1];
int *q = &arr[2];
update (p, q);
//print entire array;
}
```

ans



```
update (int *a, int *b)
{
*a = 100;
*b = 200;
}
```



*(108) = 100.

*(112) = 200

H/W

Read about

① Benefits

② Pointer to functions