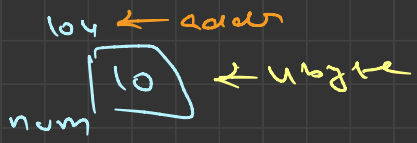
Pointer → variable → Store address of other variable

```
int num = 10;
```



Symbol table

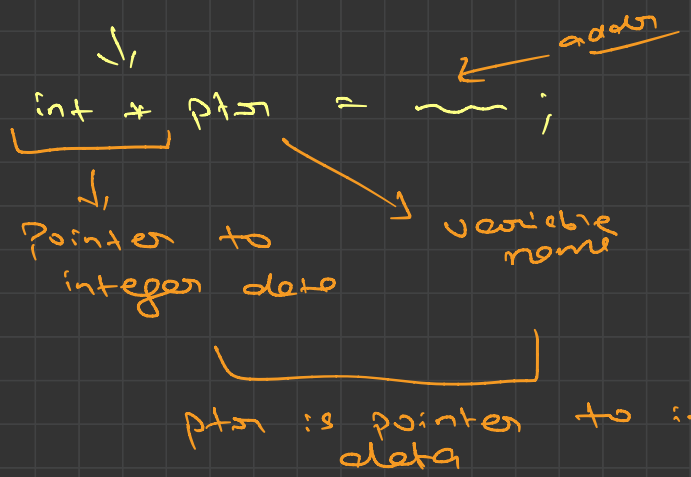
name	address
num →	104

Address of operator

↳ & → &num → address → hex

```
int a = 10;

cout << a      // 10

cout << &a     // 104 ← address
```

Pointer → Creation




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

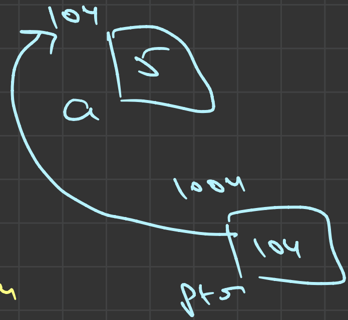
Access → ? → value stored at address stored in ptr

↳ dereference → *

↓

cout → *ptr

10 ↓




```
int a = 5;

cout << a;      → 5

cout << &a;     → 104

int* ptr = &a;

cout << *ptr;   → 5
```

→ pehle addr pe jao then print the value

↓ Accessing value stored at addr stored in ptr

```
cout << ptr;    → 104

cout << &ptr;   → 1004
```

Bad Practise

```
int *ptr;
cout << *ptr
```

} dont assign this

```
int *ptr = 0;
```

↳ null → no data

```
int a = 5;
int *ptr = &a;
```
Size of (ptr) = ? 8

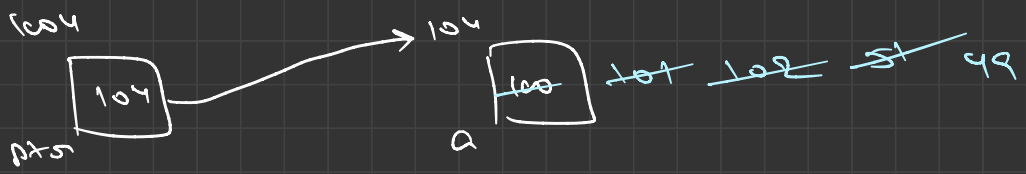
```
char ch = 'A';
char *cptr = &ch;
```
Size of (cptr) = ? 8

```
long la = 10;
long *lptr = &la;
```
Size of (lptr) = ? 8

Because it store address


```
int a = 5;
int *ptr = &a;
a = a + 1;          → a = 6
*ptr = *ptr + 1;    → a = 7
ptr = ptr + 1;      → 1008
```

```
int a = 100;
int *ptr = &a;
```



a	→	100
&a	→	104
*a	→	Error
ptr	→	104
*ptr	→	100
&ptr	→	108

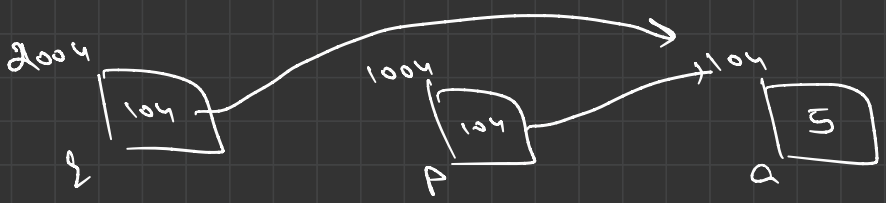
(*ptr)++ → 101

*(ptr)++ → 102

*ptr = *ptr / 2 → 51

*ptr = *ptr - 2 → 49

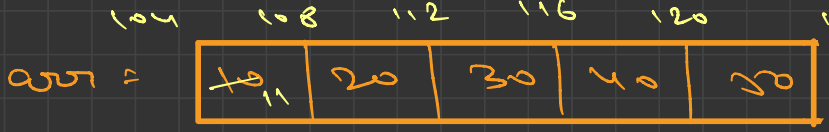
```
int a = 5;
int *p = &a;
int **q = p;
```



a	→	5
&a	→	104
*a	→	Error
p	→	104
&p	→	1004
*p	→	5
q	→	1004
*q	→	5
&q	→	2004

Pointer with array

```
int arr[5] = {10, 20, 30, 40, 50};
```

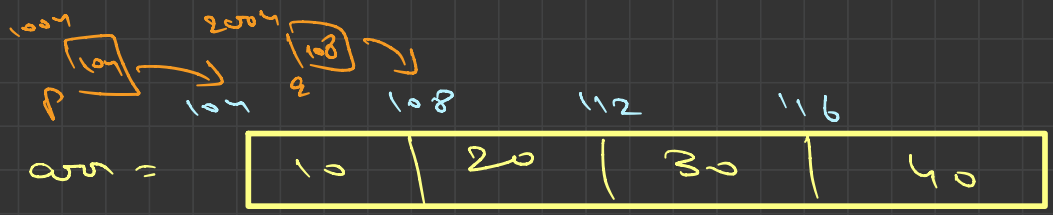



```
arr      = &arr  = 104
arr[0]   = 10
&arr[0]  = 104
&arr     = 104

*arr      => 10
*arr + 1  => 11

*(arr) + 1  => *(104) + 1 => *108 => 20
*(arr + 1)  => 20
*(arr + 2)  => 30
*(arr + 3)  => 40

*(arr + 0) => a[0]
*(arr + 1) => a[1]
*(arr + 2) => a[2]

Some { *(arr + i) => a[i]
       *(i + arr) => i[a] } correct
```

int arr[5] = {1, 2, 3, 4, 5};

arr = arr + 1; → Error ?

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




```
int *p = arr;
int *q = arr+1;
```

arr	→	104
&arr	→	104
arr[0]	→	10
&arr[0]	→	104

p	→	104	q → 108
&p	→	1004	&q → 2004
*p	→	10	*q → 20

*p+1	→	10+1 = 11
*(p)+2	→	11+2 = 12
*(q)+2	→	20+2 = 22
*(q+4)	→	Error / garbage

```
int (*ptr)[4] = &arr;
cout << (*ptr)[0];
```

```
int arr[5] = {10,20,30,40,50};
```

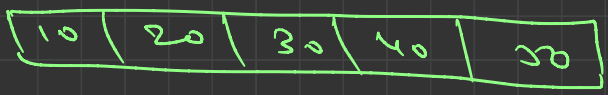
sizeof(arr) = 4 × 4 = 16 bytes

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

sizeof(p) => 8 bytes

---

arr :

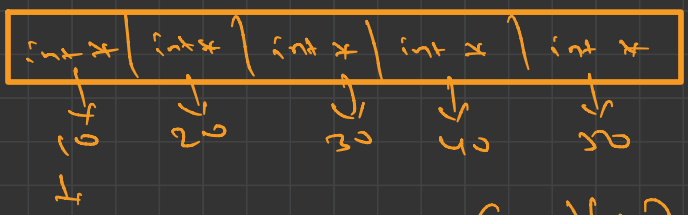


Pointer to an array

```
int *ptr = arr;
```

array of pointer

```
int (*ptr)[5] = &arr;
```

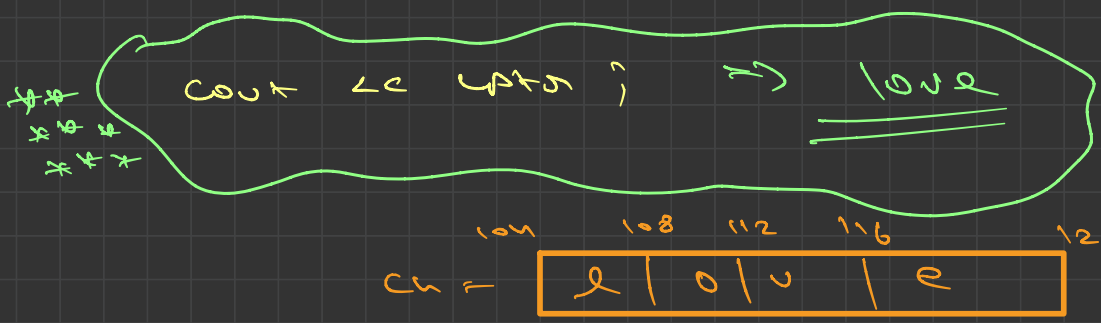


(*ptr)[0] — — — — (*ptr)[4]

Character array

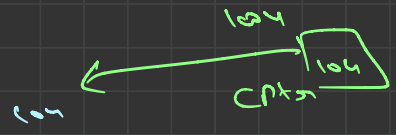
Char array →

```
char ch[50] = "love";
char *ptr = ch;
```



ch → love
&ch → 104
ch[0] → l
&ptr → 1004
*ptr → ⟶ *(ptr+0)
↳ ptr[0]
↳ l
ptr → love

```
char ch[50] = "Statement";
char * cptr = &ch[0];
```



ch :

0	1	2	3	4	5	6	7	8
S	t	a	t	e	m	e	n	t

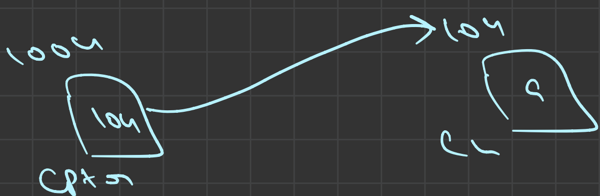
ch	→	Statement
&ch	→	104
*(ch+3)	→	t
cptr	→	Statement
&cptr	→	104
*(cptr+3)	→	t
cptr+2	→	atement
*cptr	→	cptr[0] → S
cptr+0	→	Statement

```
char *cptr = "Babbar";
```

→ Bad practice

→ Stored in temp storage

```
char ch = 'a';

char *cptr = &ch;

cout << cptr;
```



a b-8 * .....

until they find null char