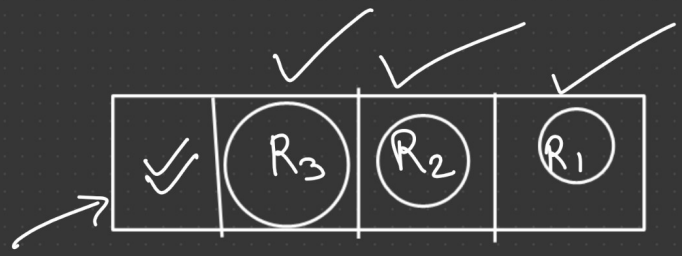
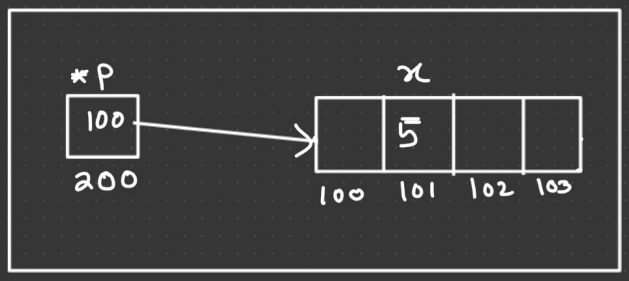
# Pointers

```
main()
{
    int x;
    char y;
    float z;

    int *p;
    char *p;
    float *p;

}
```


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

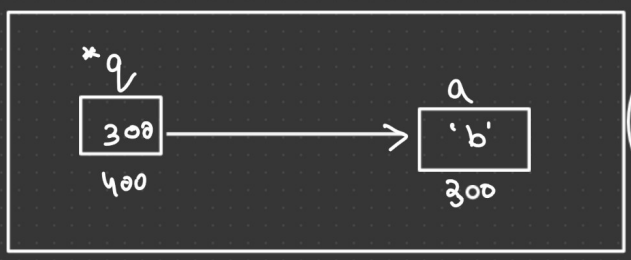
RAM




value at address of x

* &x = p

```
char a = 'b';
char *q = &a;
```




value at address of a

* &a = q

| | Address | Value |
|---|---|---|
| int x = 5; | &x | x |
| char a = 'b'; | &a | a |
| int *p = &x; | p | *(&x) = *p |
| char *q = &a; | q | *(&a) = *q |

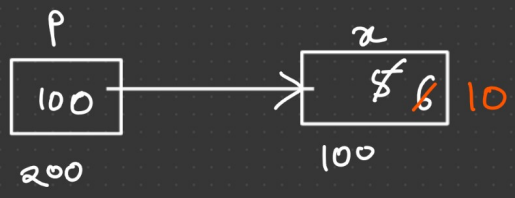
*p = x

p = &x

⇒ x = *&x (v.v.Imp)

*& → cancel each other

```
int main()
{
    int x = 5;
    int *p = &x;
    x = 6;
    printf("%d %d", x, *p);
    *p = 10;
    printf("%d %d", x, *p);
}
```




6 6 (↑ ↑)

10 10 (↓ ↓)

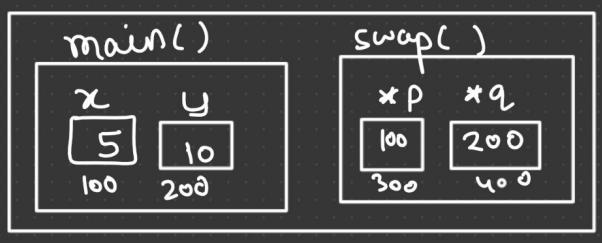
*p = *(100) = value at 100

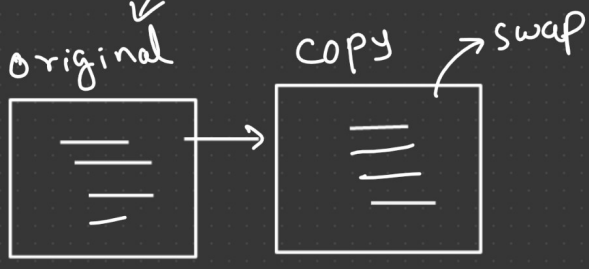
*(100) = 10;

```
int main()
{
    int x=5, y=10;
    printf("Before swapping x=%d, y=%d", x, y);
    swap(&x, &y);
    printf("After swapping x=%d, y=%d", x, y);
    return 0;
}
```

① swap(x, y); — copy

② swap(&x, &y); — permission



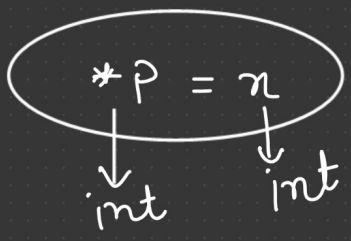



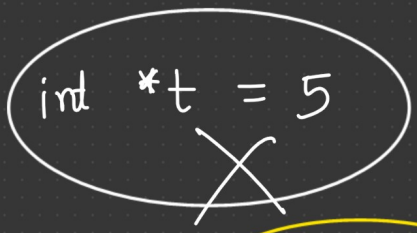

swap(&x, &y)

```
void swap(int *p, int *q)
{
    int t = *p;
    *p = *q;
    *q = t;
}
```

// int *p = &x; int *q = &y;

```
void swap(int a, int b)
{
    int t = a;
    a = b;
    b = t;
}
```




R to L

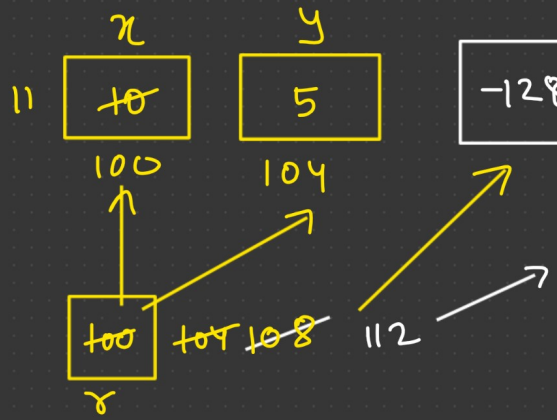
int *r = &x;

++*r = 11 ⟶ ++(*r) = ++x = 11

*++r = 5 ⟶ *(++r) = 5 (Garbage)

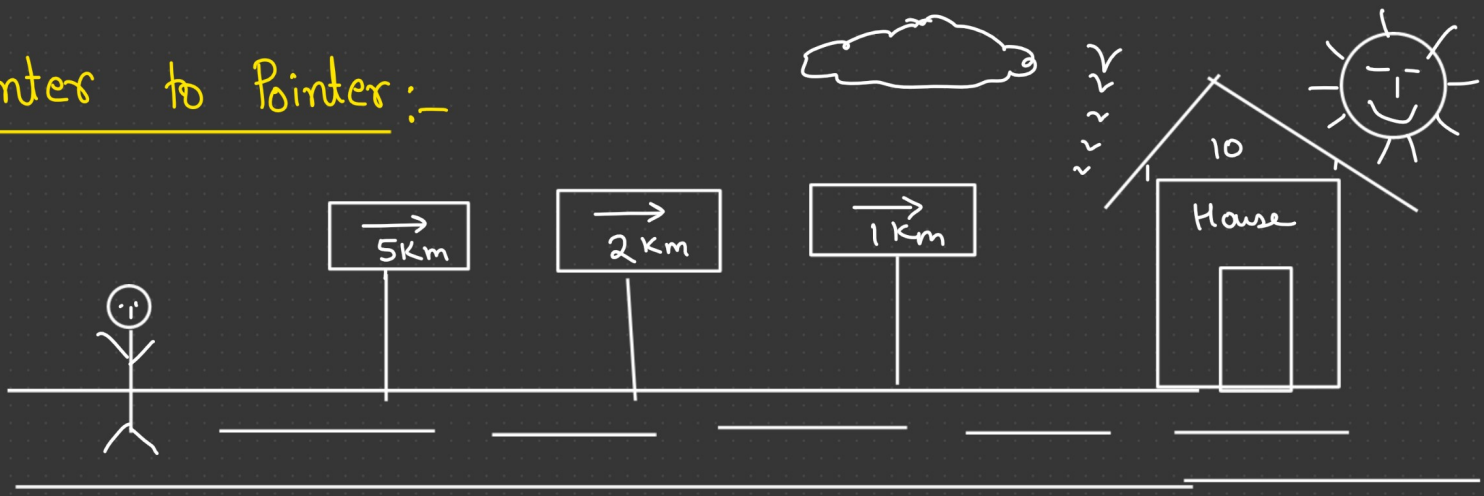
*r++ = 5 ⟶ (*r) = 5 (Garbage) =

(*r)++ = -1284391752



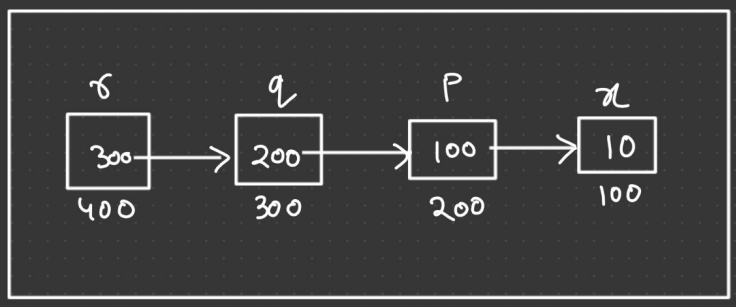

* if we increment a pointer then it will point to next memory location of it's type.

# Pointer to Pointer :-




int *** r → int ** q → int* p → int x;

```
int x = 10;
int *p= &x;
int **q = &p;
int ***r = &q;
```




P = 100 , *P = 10 ; &P = 200

| | 400 | 300 | 200 | 100 | 10 |
|---|---|---|---|---|---|
| → r | &r | r | *r | **r | ***r |
| → q | - | &q | q | *q | **q |
| → p | - | - | &p | p | *p |
| x | - | - | - | &x | x |

* value at ①
& address ②
~~(*&)~~ ③

P = &x
q = &p
r = &q

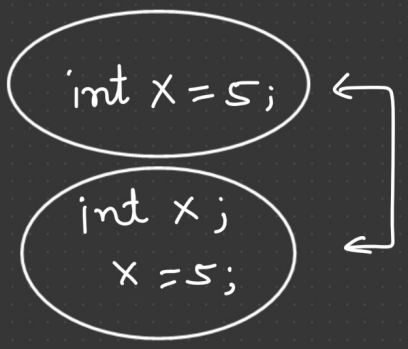
***r = **q = *p = x = 10

**q = **&p = *p = *&x = x = 10

***r = ***&q = **q = ***&p = *p = **&x = x = 10

***r = ***&q = *q = **&p = p = &x = 100

1) int x = 5;
int *p = &x; ✓




2) int x = 5;
int *p; ✓
p = &x;

3) int *p; ✗
*p = &x; //wrong

1) ++***q = ++***&p = ++*p = ++*&x = ++x = 11 ✓

2) ***r + **q = 10+10 = 20

# Returning pointer from a function:-

```
int main()
{
   int x = 10;
   int *p = fun(&x);
}
```

```
int * fun(int *q)
{
   return q;
}
```

```
int *p;
p = fun(&x);
```

```
int main()
{
   int *p;
   p = fun();
}
```

```
int* fun()
{
   int x = 10;
   return &x;
}
```

logically incorrect

Dangling Pointer

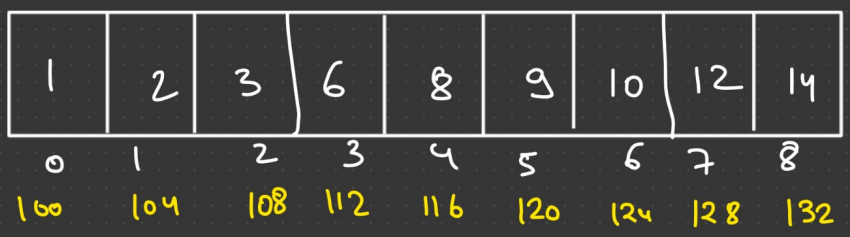
# Array & Pointers :-

int a[9] = {1,2,3,6,8,9,10,12,14};

1) int *p = a ; ✓

2) int *p = &a[0]; ✓

3) int *p = &a; ✗




P = &a[0]

*P = *&a[0] = a[0] = 1

*(P+1) = a[1] = 2

*(P+2) = 3

(P+6) = 124 = &a[6]

a[2] = *(P+2)

a[i] = *(P+i)

*(P+i) = *(i+P) = *(a+i) = *(i+a) = a[i] = i[a] = P[i] = i[P]

# 2D Array & Pointer :—

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

int *p = a;
```


p = 0th row = 100

p+1 = 1st row = 116

p+2 = 2nd row = 132

p+3 = 3rd row = 148

100 = *(p+0) = 0th location of 0th row

116 = *(p+1) = 0th location (address) of 1st row

132 = *(p+2) = 0th address of 2nd row

Program →

1) Create 2D array

2) pointer that points to 2D array.

3) Print [p, *p], [(p+1), *(p+1)]