

# Pointers

main()

{

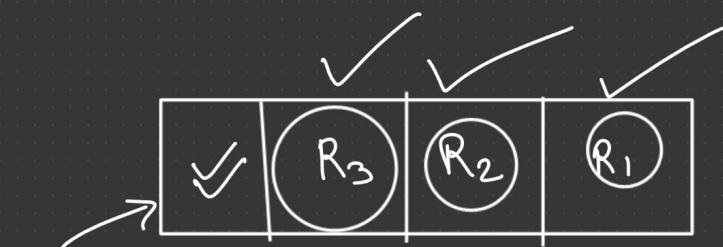


}

int x;  
char y;  
float z;

int \*P;  
char \*p;

float \*p;

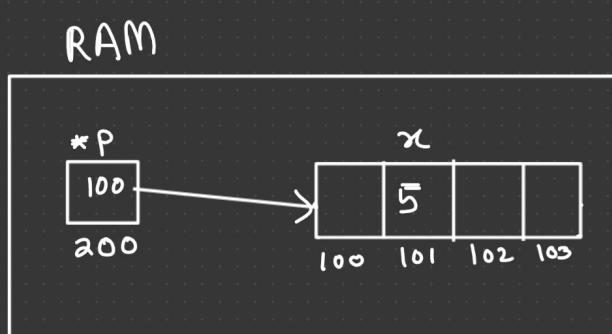


int x = 5;

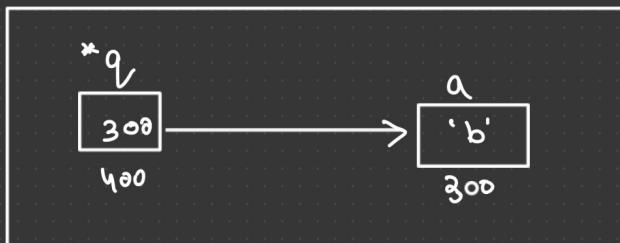
int \*p = &x;

char a = 'b';

char \*q = &a;



value at address of x  
 $\&x = p$



value at address of a  
 $\&a = q$

Address	Value
$\&x$ $\&a$	x a
P q	$*(&x) = *P$ $*(&a) = *q$

$$*P = x \quad P = \&x$$

$$\Rightarrow x = * \&x$$

v.v. Imp  
 $* \&$  → cancel each other

```
int main()
{
```

```
    int x = 5;
```

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

```
    x = 6;
```

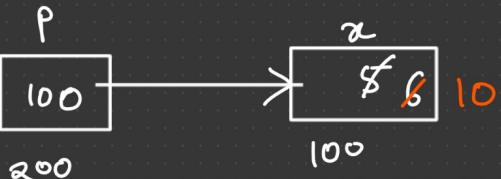
```
    printf("%d %d", x, *p);
```

```
*p = 10;
```

```
    printf("%d %d", x, *p);
```

```
}
```

6 6



$*p = *(100) = \text{value at } 100$

$*(100) = 10;$

```
int main()
```

```
{
```

```
    int x = 5, y = 10;
```

```
    printf("Before swapping x=%d, y=%d", x, y);
```

```
    swap(&x, &y);
```

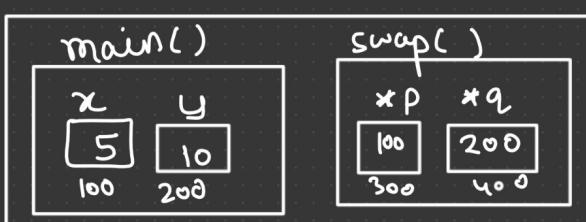
```
    printf("After swapping x=%d, y=%d", x, y);
```

```
return 0;
```

```
}
```

① swap(x, y);

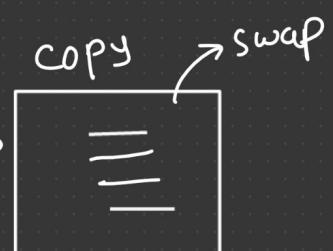
② swap(&x, &y);



original



copy



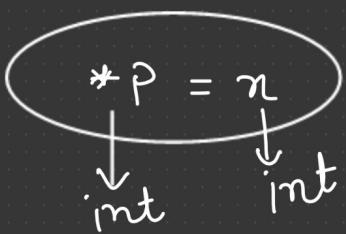
swap(&x, &y)

↓    ↓  
void swap( int \* p, int \* q ) // int \* p = &x; int \* q = &y;

{

int t = \* p;  
\* p = \* q;  
\* q = t;

}

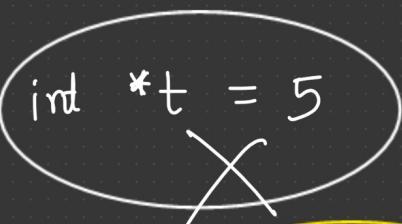


int \* r = &n;

void swap( int a, int b )

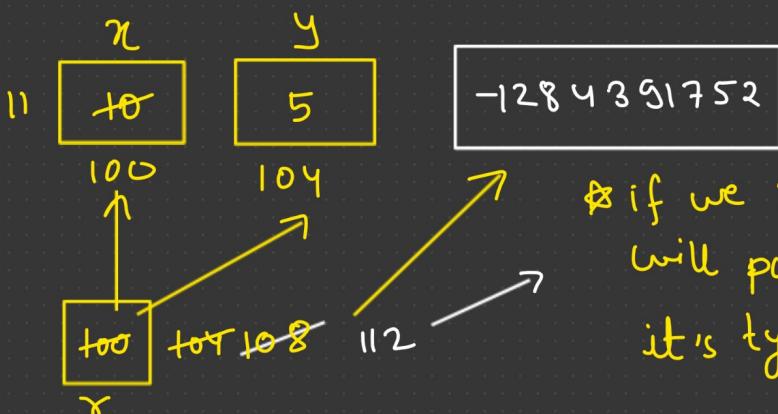
{  
int t = a;  
a = b;  
b = t;

}

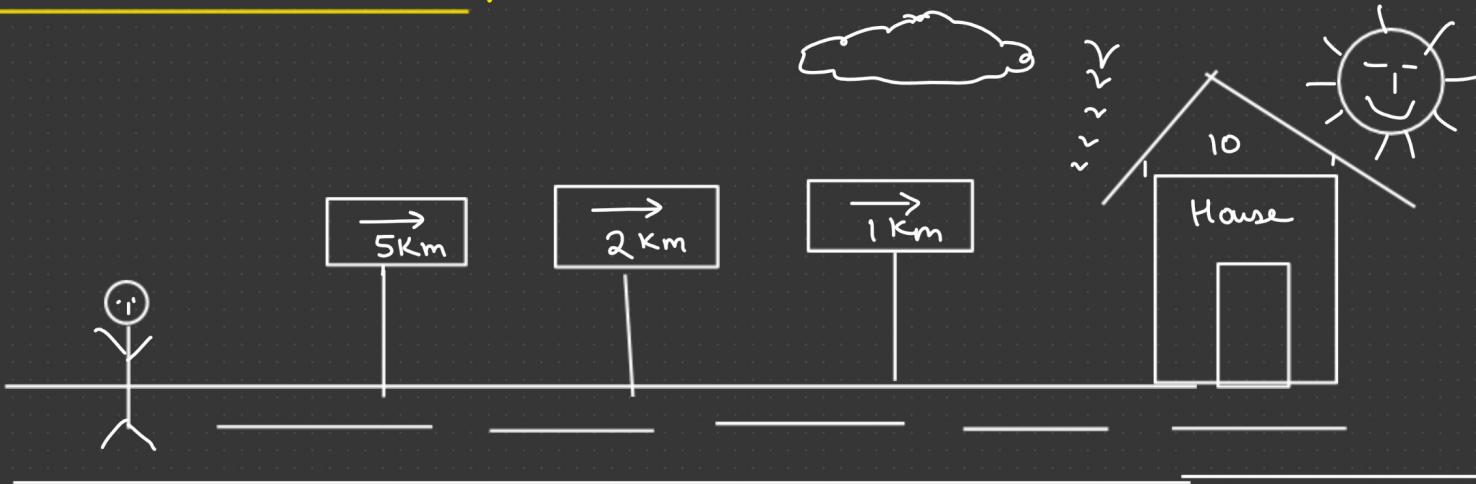


R to L

\*  
++\*r = 11    →    ++(\*r) = ++x = 11  
\*++r = 5    →    \*(++r) = 5 (garbage)  
\*r++ = 5    →    (\*r) = 5 (garbage) =  
(\*r)++ = -1284391752

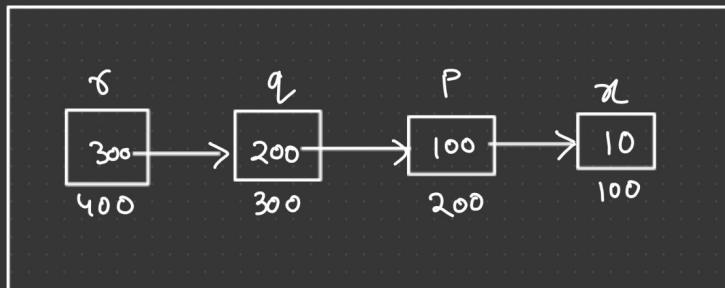


# Pointers to Pointer:-



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

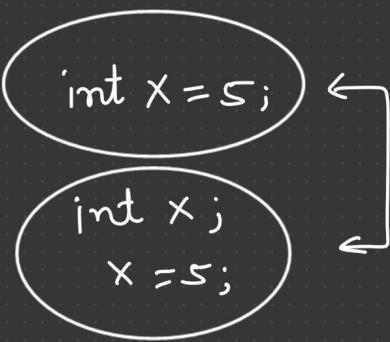
$p = \&x$   
 $q = \&p$   
 $r = \&q$

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

$$***r = **\underline{\&q} = **q = *\underline{\&p} = *p = *\underline{\&x} = x = 10$$

$$***r = *\underline{\&}q = *q = *\underline{\&}p = p = \&x = 100$$

1) `int x = 5;`  
`int *p = &x;` ✓



2) `int x = 5;`  
`int *p;` ✓  
`p = &x;`

3) `int *p;` X  
`*p = &x; // wrong`

1)  $\text{++} * \text{*q} = \text{++} * \underline{\&p} = \text{++} * p = \text{++} \underline{\&x} = \text{++} x = 11$  ✓

2)  $\text{* * *} 10 + \text{* * *} 10 = 10 + 10 = 20$

## Returning pointer from a function:-

```
int main()
```

```
{
```

```
    int x = 10;
```

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

```
    {  
        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;` X

1	2	3	6	8	9	10	12	14
100	104	108	112	116	120	124	128	132

$$P = \&a[0]$$

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

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

$$*(P+2) = 3$$

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

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

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

~~$*[P+i] = [i+P] = *[a+i] = *[i+a] = a[i] = i[a] = P[i] = i[P]$~~

## 2D Array & Pointer :-

int  $a[4][4]$ ;

int \*p = a;

$p = 0^{\text{th}}$  row = 100

$p+1 = 1^{\text{st}}$  row = 104

$p+2 = 2^{\text{nd}}$  row = 108

$p+3 = 3^{\text{rd}}$  row = 112

	100	104	108	112
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12
3	13	14	15	16

$\rightarrow a+0$   
 $\rightarrow a+1$   
 $\rightarrow a+2$   
 $\rightarrow a+3$

$$*(p+0) = 1 \quad *(p+9) = 10,$$

$$*(p+1) = 2$$

$$*(p+2) = 3$$

program

int [\*p][4] = a;

1) Create 2D array

2) pointer that points to 2D array.

3) print  $[p, *p], [(p+1), *[(p+1)]]$



Q:- WAP to reverse elements of array by passing array to function using pointers ?

```
int main()
{
    int a[] = {4, 6, 8, 2, 1, 7, 3, 15, 12, 18};
    reverse(a, 10);

    for(i=0; i<10; i++)
        printf("%d", a[i]);
    return 0;
}
```

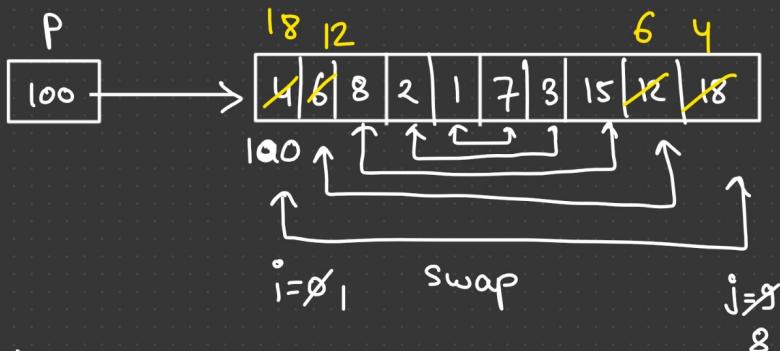
```
Void reverse( int *p, int n)
{
    int i, j, temp;

    for(i=0, j=n-1; i<=j; i++, j--)
    {
        temp = p[i]; // *(p+i)
        p[i] = p[j];
        p[j] = temp;
    }
}
```

int \* p = &a;

wrong

∴ name of array itself  
points to base address.



temp  
12    6

Sorting:-

0	1	2	3	4
1	2	5	4	3

for( $i=0$ ;  $i < n-1$ ;  $i++$ )       $i = 1$        $j = 2 \cancel{3} 4$

{    for( $j=i+1$ ;  $j < n$ ;  $j++$ )

{     if( $a[i] > a[j]$ )

      temp =  $a[i]$ ;  
       $a[i] = a[j]$ ;  
       $a[j] = temp$ ;

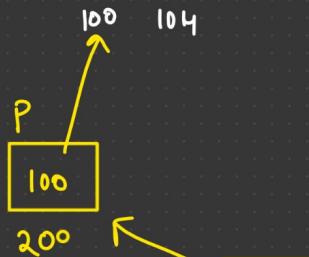
}

}

int \*p = a;  
 int \*\*q = &p;

$\rightarrow$  reverse       $\rightarrow$  main  
 $\rightarrow$  print

0	1	2	3	4	5	6	7	8
6	4	5	3	1	8	9	14	2



$$**q = **\&p \\ = *p = *a = a[0]$$

$$*(\ast q + 1) = 4$$

$$*(\ast q + 5) = 8$$

$$*(\ast q + i) = a[i]$$

## Pointer to a Array :-

```
int a[50];
```

```
int (*p)[5] = a;
```

