C-72 → Pointers in C → Part 2

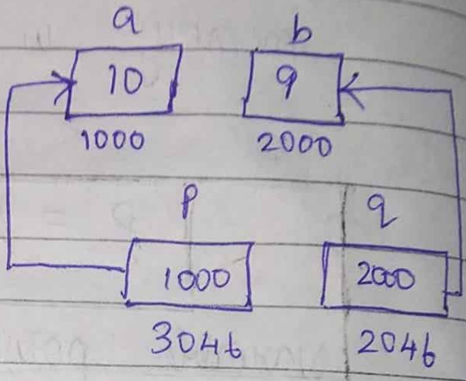
Address of (&) and Indirection (*) operators in pointers

* → Indirection Operator (or) De-referencing
& → Address of operator (or) Referencing

```
int a=10, b=9;
int *p, *q;
```

```
P = &a;
q = &b;
```

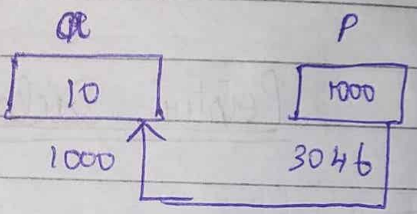
⟹



Equal to has higher precedence.

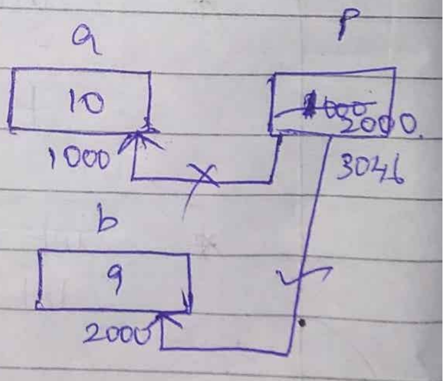
```
P = &a, &b;
```

⟹



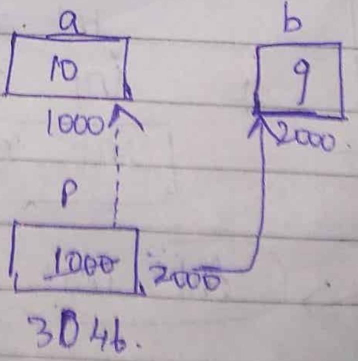
```
P = (&a, &b);
```

⟹



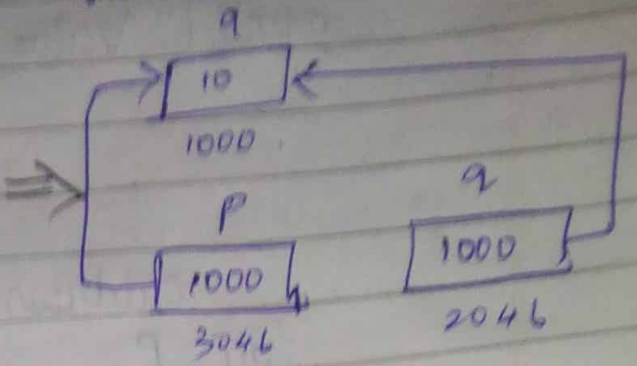
```
P = &a;
P = &b;
```

⟹



* More than one pointer can also point to single variable address.

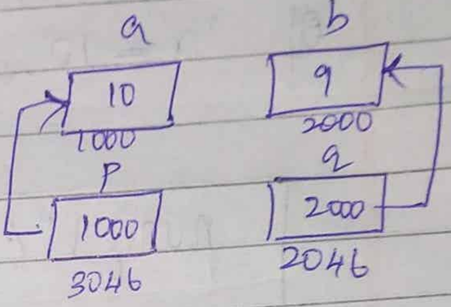
```
p = &a;
q = &a;
```



* Here both P and q pointers are pointing to same Variable address.

Example:
```
int a=10, b=9;
 int *p, *q,

P = &a;
q = &b;
```

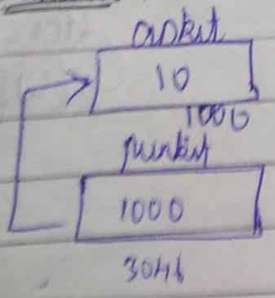


Same
```
printf ("Value of a = %d", a);  => 10
printf ("Value of a = %d", *P); => 10 .
```

Both same ; to print value of 'a'

(* → Indirection operation ; which indirects the value to be printed through means of address of 'a')

Scenario:



sanjay (3rd person) dont know ankit address ; but he knows purkit who holds address of ankit; so by means of purkit ; we access ankit.

* If we simply give;

printf (" Value of a = %d ", P); ⇒ 1000

↓

* It prints the address 1000 which is stored in P.

* But when we give *P; it is indirection to address of the 1000 variable (ie) 'a' and prints value of 'a' = 10.

printf ("%d", P) ⇒	Value of P ⇒ (1000)
printf ("%d", *P) ⇒	Value of address Stored in P ⇒ (10)

$\boxed{\begin{array}{l} *P \Rightarrow *(\&a) \\ \quad \Rightarrow *(1000) \Rightarrow 10 \end{array}}$

* If we want to print address of 'a'

printf ("%~~d~~x", &a); ⇒ hexadecimal form

(1000) → address of a

↳ (01feca)

↓

address will be in (hexa decimal) so "%x" is format specifier.

Same

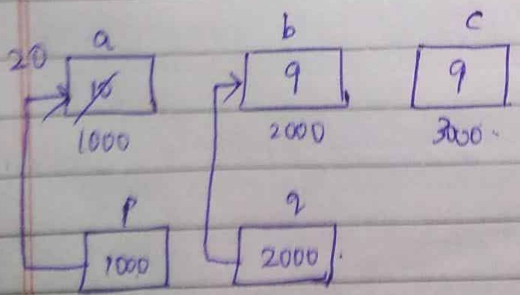
```
printf("address of a : %x", P);     => 1000
printf("address of a : %x", &a);    => 1000
```

* Both same to print address of 'a'.

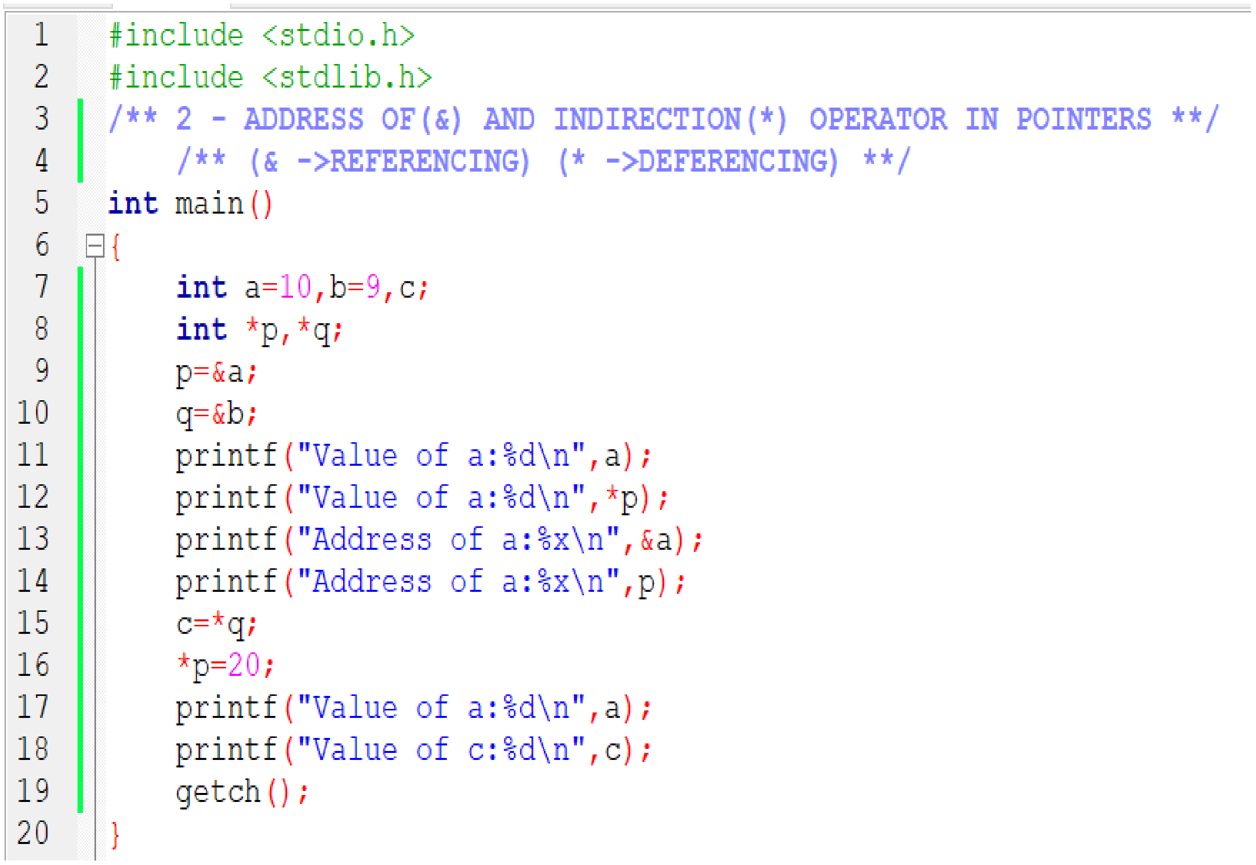
* printf("address of P : %x", &P); => 3046.

Assignment 1 :-

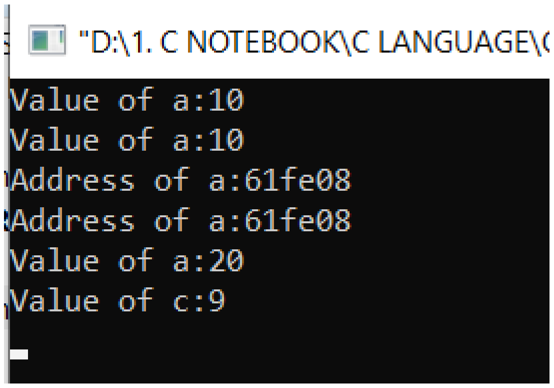
```
int a=10, b=9, c;
  int *P, *q;
  P = &a;
  q = &b;   c = *q;     *P = 20;
printf("value of a : %d", a);          => 10
printf("value of a : %d", *P);         => 10
printf("address of a : %x", &a);       => 61fe08
printf("address of a : %x", P);        => 61fe08
printf("value of c : %d", c);          => 9
printf("value of a : %d", a);          => 20
printf("~~addr~~ address of P : %x", P);    => 61fe08
```



CODE 1:



```c
#include <stdio.h>
#include <stdlib.h>
/** 2 - ADDRESS OF(&) AND INDIRECTION(*) OPERATOR IN POINTERS **/
    /** (& ->REFERENCING) (* ->DEFERENCING) **/
int main()
{
    int a=10,b=9,c;
    int *p,*q;
    p=&a;
    q=&b;
    printf("Value of a:%d\n",a);
    printf("Value of a:%d\n",*p);
    printf("Address of a:%x\n",&a);
    printf("Address of a:%x\n",p);
    c=*q;
    *p=20;
    printf("Value of a:%d\n",a);
    printf("Value of c:%d\n",c);
    getch();
}
```



"D:\1. C NOTEBOOK\C LANGUAGE\(

```
Value of a:10
Value of a:10
Address of a:61fe08
Address of a:61fe08
Value of a:20
Value of c:9
```