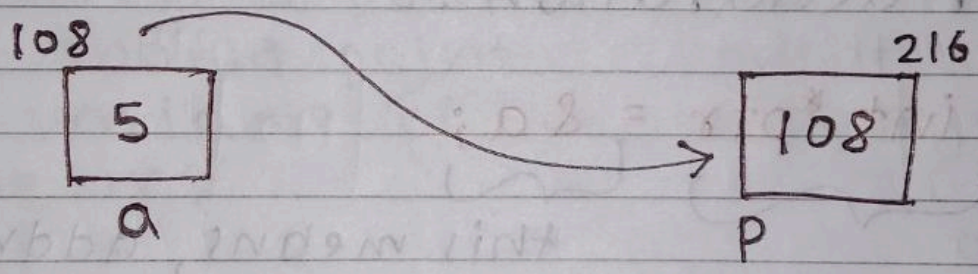
Pointers (class 1)

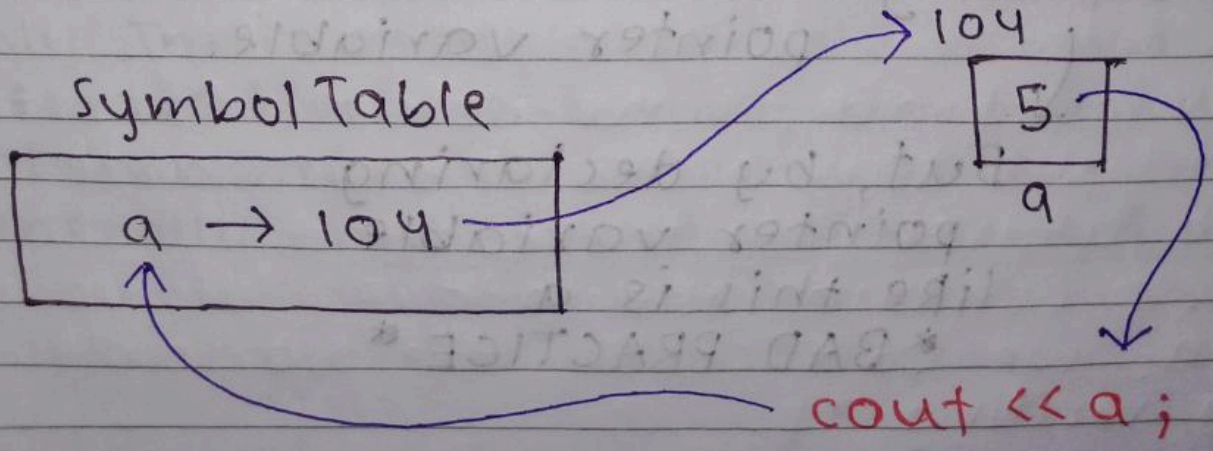
Date 03-03-2023

A pointer is a variable that stores memory address.
Like all other variables, it also has a name, has to be declared & occupy some space in memory.
It is called pointer because it points to a particular location in memory by storing the address of that location.
Example :-

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



* Basically, what happens at the backend is :- the name of the variable & its address is mapped in Symbol table.



Date.....

when user enters a statement :-

```
cout << a;
```

This statement will go to symbol table & the variable 'a' is searched then the mapped address is checked, i.e., 104.
Then compiler will go to that address & corresponding value is displayed to the user.

* Pointer declaration :-

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

this means, ptr is a variable of integer pointer type.

this means, address of variable 'a'.

```
int *ptr ;
```

} this is how we declare pointer variable.

But, by declaring pointer variable like this is a *BAD PRACTICE*

Date.....

Instead of this, we can declare pointer like this,

```
int *ptr = 0;
        or
int *ptr = NULL;
```

```
int *ptr = nullptr;
```

↓

modern way to initialize pointers.

Example:-

```
int *ptr = NULL;
```

↓

ptr is a pointer that should point to variable of type int.

```
char *ptr = NULL;
```

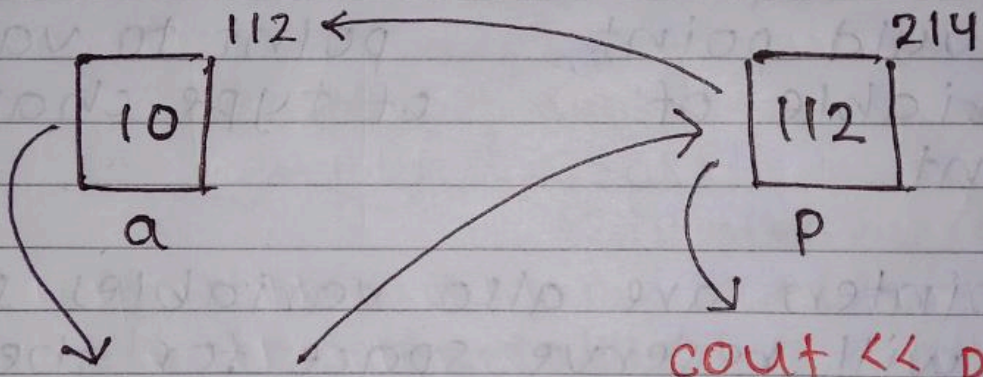
↓

ptr is a pointer that should point to variable of type char.

Note :- Pointers are also variables so compiler will reserve space for them & they will also have some address. All pointers irrespective of their base type will occupy same space in memory since all of them contain addresses only. The size of a pointer depends on the architecture & may vary on differe-nt machines.
Generally, the size of pointer is 8 Bytes.

* Dereferencing a pointer variable :-
we can access a variable indirectly using pointers. For this, we can use indirection operator (*). This is also called dereferencing operator.
By placing the indirection operator (*) before a pointer variable, we can access the variable whose address is stored in the pointer.
Example :-

```
int a = 10;        int *p = &a;
```




```
cout << *p;
```

↓

o/p :- 10

```
cout << p;
```

↓

o/p :- 112

In our program, if we place '*' before p then we can access the variable whose address is stored in p. Since, p contains the address of varia-ble a, we can access the variable a by writing *p.

Teacher's Sign

Date.....

*pl = 9; is equivalent to, a = 9;

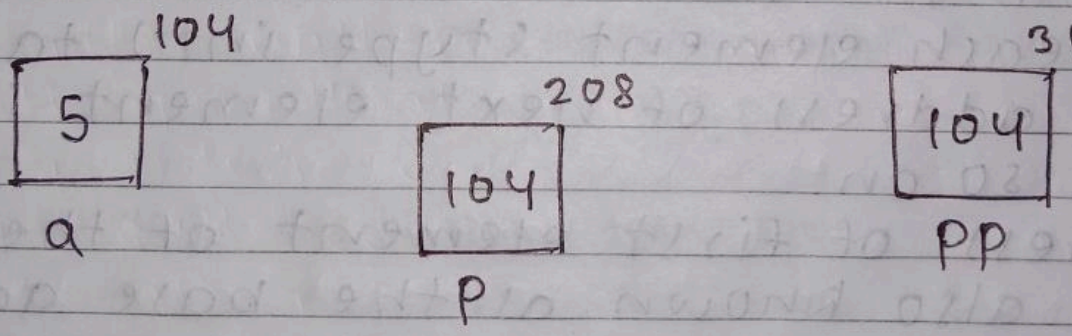
(*pl)++; is equivalent to, a++;

x = *pl + 10; is equivalent to, x = a + 10;

cout << *pl; is equivalent to, cout << a;

The indirection operator (*) can be read as "value at"..
For example:- *p can be read as "value at p".

```
int a = 5; // variable a
int *p = &a; // pointer to variable a
int *pp = p; // copy pointer.
```



a → 5
&a → 104
p → 104
&p → 208
*p → 5
pp → 104
&pp → 312

*pp → 5
(*p)/2 → 5/2
(*pp)/2 → 5/2