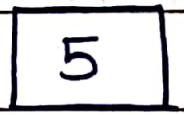
3/03/2023

Pointers

```
int a = 5;
```



But is it really that memory is given name as a. The answer is no. An address is assigned to that memory location & then we access the variable at that particular location. There is a data structure named as symbol table in which there is mapping

stored between the variable name & address.

int x = 12;
There will be a mapping stored in symbol table which will map x to an address & this (memory management) is done via OS. But can we get to know that what is the address to which x is mapped in the symbol table. The answer is yes & we can get to know the address with the & operator.

& operator
This is the ampersand operator & is also known as address of operator.

```
int a = 5;
cout << a << endl;
cout << &a << endl;
```
→ Hexadecimal value will be displayed

Also different variable will have different address.

Concept of pointers
int * ptr ;
The above statement means that it is a pointer to integer data. If we want to store the address, we can store that address with the help of pointers.

```
int a = 5;
int * p = &a;
```
→ address of operator
→ name of variable
→ name of pointer

data type | Pointer syntax / Dereference operator

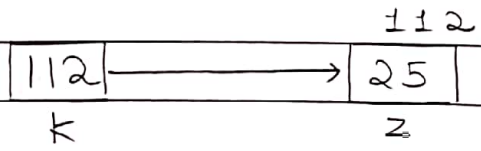
Here we get to know that p is the pointer to integer data.

char * p = &ch;
This means p is a pointer to character data.
bool * p1 = &x;
This means p1 is a pointer to boolean data.

Note → Also note that in pointer, address is stored always.

int b = 5;
int * ptr = &b;
cout << * ptr ; Will print 5
↳ dereference operator
By using * ptr, we can access the value present at the address.

Ex → int z = 25;
int * k = &z;



Pointer is not a data type. It is basically a variable that stores the address of another variable. We can access address & value via pointer.

```
int * ptr = &a;
cout << ptr << endl; // Access address
cout << * ptr; // Access value
```

*ptr means the value stored at address stored in ptr.

cout << &ptr; → Address of ptr block.

Size of pointer

```
int a = 5;              char ch = 'a';
int *ptr = &a;          char *ptr1 = &ch;
```

What will be the size of ptr and size of ptr1. Are they same or dependent on the size of data type of pointer. It is not dependent to which it is pointing but it is storing same thing i.e. address and hence size will be same & i.e. 8 bytes.

```
sizeof(ptr);   } Will be same → 8 byte / however
sizeof(ptr1);  } system dependent.
```

Why we need pointer?
Dyanamic memory allocation is done via pointer. Memory management pointer is also there. To access hardware, we use pointer. There are other applications of pointers also.

Note → `int *ptr;` `cout << *ptr;` } Segmentation fault will occur as we are accessing memory which might not be ours.

This is a bad practice. To rectify it we use the concept of null pointer.

int * ptr = nullptr; //New practice
int * ptr = 0; //Old practice

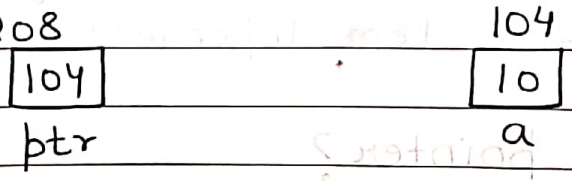
Pointer arithemetic
int a = 5;
int * ptr = & a;

ptr → 104

ptr = ptr + 1; //Changing address
Now ptr will have 108 stored as int takes 4 bytes space.

* ptr = * ptr + 1 ; //Changing value
5 + 1 = 6
Now if we do cout <<a ; , 6 will be printed.

Ex→

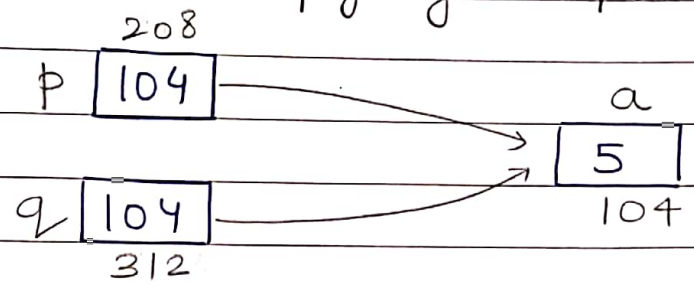


a	10
&a	104
ptr	104
*ptr	10
&ptr	208
*p = *p x 2	30
*p = *p / 2	15

First use
*p *2 20
(*ptr)++ 10
++(*ptr) 12
a = a + 1 13
*p = *p + 2 15

Copying pointer to another pointer
int a = 5;
int * p = &a;

int * q = p ; // Copying the pointer

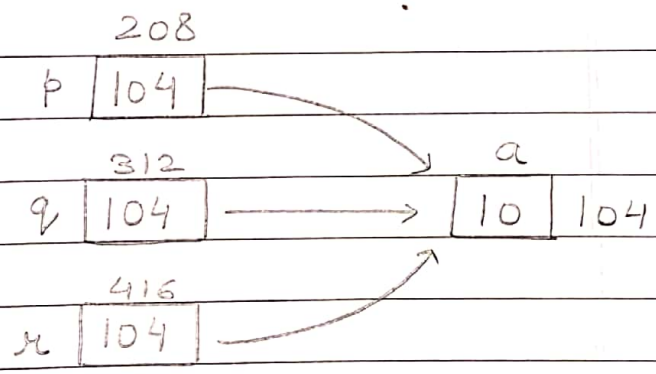


Ex→

a	5	* p	5	cout << (* p /2) ; 2
&a	104	q	104	
p	104	&p	312	cout << (* q /2) ; 2
&p	208	* q	5	

Ex→

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



a	10
&a	104
p	104
&p	208
* p	10
q	104
&q	312
* q	10
r	104
&r	416
* r	10