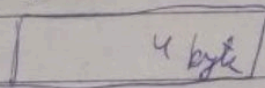


Introduction to Pointers

Suppose, I create a variable and ~~not~~ ~~not~~ initialise it a value

int i = 10

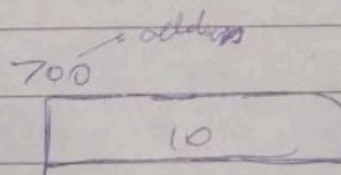


And then perform an operation on it :- $i = i + 5$

Now, the question arises, how will the system know where the value of i is stored?

It will check its symbol table first, the address of 'i'.

i	→	700
j	→	800
k	→	790



cout << &i << endl;

↓
will give address of i.

Pointers are variables which store address of other variable.

Syntax of storing address in pointer:-

()

int *p = &i;

↓
"p" is a pointer to an integer

cout << p << endl;

↓
address of i

Date

If I have the address of any variable, then using ~~at~~ dereference operator, its value can be accessed.

cout << *p << endl;

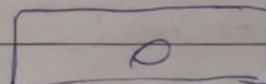
↳ value of 'i'

* - Give me the value pointed by p.

int * p = 0 // null pointer, pointing to nowhere.

(*p)++

↳ it will mean that



p

increase the value of no. of by 1 whose address is 0.

Pointer Arithmetic:

- * Generally, the size of all the pointers is same (8^{byte} ~~bits~~) irrespective of address of which datatype they are storing.

Pointer Arithmetic

int i = 10;
int * p = &i;

cout << p; // Will give address of i
p++;
cout << p; // Will give address stored next to i

This kind of operation is used in array.

Arrays and Pointers:

```
int a[10]; // declaration of array.  
cout << a;  
cout << &a[0];
```

(Both will give the address of first element of array)

Hence "a" can be treated as a pointer.

$a[0] = 5;$

$a[1] = 10$

```
cout << *a // 5  
cout << *a *(a+1) // 10
```

Hence, i th element of an array can be accessed by $*(a+i)$.

Now, looking at some differences b/w Pointer & array pointer.

```
cout << a  
cout << &a
```

(Both will give address of a)

```
cout << &ptr (Will give address where pointer is storing address)
```

Now, separate memory is being created to store "a".

```
int a[10];      sizeof(a) => 40  
int *ptr = 0;   sizeof(ptr) => 8
```

Separate memory is initialised to pointer.

Characters and pointers

```
int a[] = {1, 2, 3};
```

```
char b[] = "abc";
```

```
cout << a << endl;
```

// Gives address of first element
cout << b << endl; // will go to first element of array
+ start printing all the values till
it encounter '\0'.

↳ "abc",

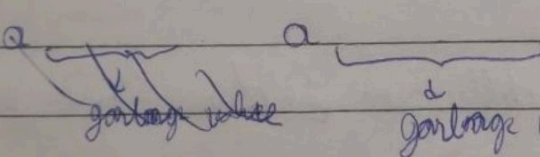
```
char *c = &b[0];
```

```
cout << c; // will give "abc" as output
```

```
char c1 = 'a';
```

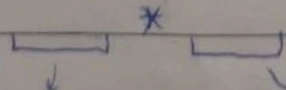
```
char *pc = &c1; // it will go to address of c1, start printing  
the values till it encounter '\0'.
```

Output :-



Double Pointer

Point to remember while declaring a pointer:-



Variable name of
pointer.

Datatype of variable
whose address is
being stored

Now, if a pointer is storing an address of an integer pointer then it will be declared as:-

int ** p;

Jiska address

store kar

rha hai