11/5/21

Conditional or Ternary Operators →

→ [condition ? code 1 : code 2]

we use when the code is of one or two lines.

similar to,

```
if (condition)
    code1;
else
    code2; ]
```

→ If condition is true, code 1 will be executed and if the condition is false code 2 will be executed.

* String Operator

```
var myString = "Rutvi";
myString += " Kumawat";
myString = "Rutvi" + "Kumawat";
```

* Increment & Decrement Operator →

1. Postfix increment → a++ (a = a+1)
2. Prefix increment → ++a
3. Postfix decrement → a-- (a = a-1)
4. Prefix decrement → --a

- ++ (increment operator)
- -- (decrement operator)

* The diff. b/w Postfix & Prefix is that →

In Post-fix →

if suppose value of 'a' is been used somewhere then firstly it will be used and then it would be incremented / decremented.

In Pre-fix →

firstly the value is been incremented / decremented and then it is been used.

* Bitwise Operators - I ...

1. Bitwise And → a & b
2. Bitwise OR → a | b
3. Bitwise XOR → a ^ b
4. Bitwise NOT → ~a
5. Left shift → a << b
6. Sign propogation Right Shift → a >> b
7. Zero fill Right Shift → a >>> b.

→ In this for any no. to work it will firstly be converted to bit

Ex →

1	00000001
2	00000010
3	00000011
4	00000100
5	00000101
6	00000110
7	00000111
8	00001000

(1) a & b = 8

$2^3\ 2^2\ 2^1\ 2^0$

i.e, 0 0 0 0 1 0 0 1

$2^3 + 2^0$

8 + 1

= 9.

a = 8 → 0 0 0 0 1 0 0 0

b = 9 → 0 0 0 0 1 0 0 1

0 0 0 0 1 0 0 0

$2^3 = 8$

Ans

(2) a | b = 9

0 0 0 0 1 0 0 0

+ 0 0 0 0 1 0 0 1

0 0 0 0 1 0 0 1

$2^3 + 2^0 = 8 + 1$

$= 9$ //

(3) a ^ b = 1

(either bit should be 1)

0 0 0 0 1 0 0 0

0 0 0 0 1 0 0 1

0 0 0 0 0 0 0 1

(4) ~a

a = 9 → 0 0 0 0 1 0 0 1

→ 1 1 1 1 0 1 1 0

If the left most bit is 0 then the no. is positive and is stored in the memory.

but if the left most bit is 1 then the no. is negative and in this case it's

2's complement will be stored.

* -ve no's stored in 2's complement.

-9 = 10001001 } ① the bits are inverted

01110110 } ② we add 1

$\quad\quad\quad\quad 1$

0111 0111

So, for our no., 9

1111 0110 } reverse the bits

① 0000 1001 } we add 1

$\quad\quad\quad\quad 1$

0000 1010

$2^3 + 2^1 = 8 + 2 = 10$

since, it was a -ve no, it will be stored as

-10

⑤ Left Shift (a << b)

(9 << 2)

←

1001

~~10~~ 0100 ✓ ④

← These will be discarded

now, 0 0 0 0 1 0 0 1

0 0 1 0 0 1 0 0 → 8.

(6) Zero fill Right - Shift → Same as left shift but the direction changes.

Ex → 9 >>>2

1001 → 0 0 1 0 discarded

(7) Sign Propogating Right - Shift →

Case (1) Negative no is →

if the left most digit is 1 → 1 0 0 1 0 1 0 1 discarded. >>2

then the empty spaces are filled with 1

1 1 1 0 0 1 0 1

if it was 0 then it would be filled with 0.

0 0 0 0 1 0 1 1.

5 = 00000000000000000000000000000101

-5 = 11111111111111111111111111111011

c = -5

c >> 2

a = 5 << 2

00000101

1111 ---- 1110

10100 = 20

since it is -ve ..

2's Comp —

0000 0001

\+ 1

0000 0010 → -2