# Bitwise Operators

Miscellaneous Concepts :-

→ They perform operations on the binary form of the numbers.

① → Bitwise & (AND)

a = 4 b = 8

100 1000

0100

&1000

0000 = $(0)_{10}$

0 & 1 = 0

0 & 0 = 0

1 & 0 = 0

1 & 1 = 1

Can be verified in the code as well

② Bitwise | (OR)

a = 4 b = 8

100 1000

0100

|1000

$(1100)_2 = (12)_{10}$

0 | 1 = 1

0 | 0 = 0

1 | 0 = 1

1 | 1 = 1

8 + 4 = 12

③ Bitwise ^ (xOR)

(Exclusive OR)

if bits same = 0

If bits different = 1

$0 \wedge 0 = 0 \quad 0 \wedge 1 = 1$

$1 \wedge 1 = 0 \quad 1 \wedge 0 = 1$

$a = 4 \quad b = 8$

$100 \quad 1000$

$0100$

$1000$

$(1100)_2 = (12)_{10}$

④ Bitwise $<<$

$n = 4 \ (100)_2$

Leftshift

Actually it shifts the binary form by i times

| a<<b | n<<i |
|---|---|
| ans = $a * 2^b$ | n<<1 |
| 8<<1 | ~~1000~~ 100 |
| 1000 | (1000) = $(8)_{10}$ |
| 10000 = $(16)_{10}$ | New zero added |
| $= 8.2^1$ | |
| $= 8.2$ | |
| $= 16$ | |

⑤ $>>$ Bitwise

Rightshift

| a >>b |
|---|
| ans = $a / 2^b$ |
| ans = $\frac{a}{2^b}$ |

$10 << 1$

$\frac{10}{2^1} = \frac{10}{2} = (5)$

$10 << 1$

$1010$

$1010$

$(0101) = (5)_{10}$

$8 >> 2$

$1000$

$0010 \ (2)_{10}$

Hence proved.

## Operater Precedence

| operaters | Precedence | |
|---|---|---|
| !, +, - | Unary operaters first | R to L |
| *, /, % | second | L-R |
| +, - | third | L-R |
| <, <=, >=, > | fourth | L-R |
| ==, != | fifth | L-R |
| && | sixth | L-R |
| \|\| | seventh | L-R |
| = | Last | R-L |

Note: Bitwise operaters exists

we can overwrite all these operations using the bracket.

If operaters of same precedence comes the the associative property applies

4 * 5 / 2

* /

first second

L → R

## Scope

Area of accessibility/usability of the variables.

① Local

② Scope(Global)

① Local → if-else
→ functions
→ block of code { }
→ loops

② Global → accessible everywhere

## Data type modifiers

Change meaning of datatypes

long
short
long long
signed
unsigned

Suppose

int x=2 (4 bytes)


Combinations = $2^{31}$

0 → +ve → 0 to $2^{31}-1$

1 → -ve → $-2^{31}$

If $2^{32}$ comes int does not have capacity to store the value because it have the capacity of

$-2^{31}$ to $2^{31}-1$

So, we can change the capacity

→ long int
long double
long gives extra 4bytes
So,
long int 64 bits $-2^{63}$ to $2^{63}-1$

→ short 2 bytes
will down the capacity to 2bytes.

→ long long 8 bytes

→ signed
int is signed by default
signed means can save +ve values
-ve values

→ usigned
only positive values

(MS) _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ to 32 bits

No need of
Most Significant
So size increases upto
$-2^{32}$ to $2^{32}-1$ which is
double of
$-2^{31}$ to $2^{31}-1$