

Bitwise operators

Miscellaneous Concepts :-

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

① → Bitwise & (AND)

$$a = 4 \quad b = 8$$

$$100 \quad 1000$$

$$0100$$

$$\underline{01000}$$

$$0000 = (0)_{10}$$

$$0 \ 8 \ 1 = 0$$

$$0 \ 8 \ 0 = 0$$

$$1 \ 8 \ 0 = 0$$

$$1 \ 8 \ 1 = 1$$

Can be verified in the code as well

② → Bitwise | (OR)

$$a = 4 \quad b = 8$$

$$100 \quad 1000$$

$$0100$$

$$\underline{1000}$$

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

$$0 \ 1 \ 1 = 1$$

$$0 \ 1 \ 0 = 0$$

$$1 \ 1 \ 0 = 1$$

$$1 \ 1 \ 1 = 1$$

$$8 + 4 = 12$$

③ Bitwise ^ (XOR)
[Exclusive OR]

if bits same = 0
if bits different = 1

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

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

$$a = 4 \quad b = 8$$

$$100 \quad 1000$$

$$\begin{array}{r} 0100 \\ 1000 \\ \hline (1100)_2 = (12)_{10} \end{array}$$

④ Bitwise \ll

leftshift
Actually it shifts the
binary form by i
times

$$\begin{array}{|l} a \ll b \\ \text{ans} = a \times 2^b \\ 8 \ll 1 \\ 1000 \\ 10000 = (16)_{10} \\ = 8 \cdot 2^1 \\ = 8 \cdot 2 \\ = 16 \end{array}$$

$$n = 4 \quad (100)_2$$

$$\begin{array}{|l} n \ll i \\ n \ll 1 \end{array}$$

$$\begin{array}{r} 1000 \quad 100 \\ \hline (1000) = (8)_{10} \\ \text{New zero added} \end{array}$$

⑤ \gg Bitwise

Rightshift

$$\begin{array}{|l} a \gg b \\ \text{ans} = a / 2^b \\ \text{ans} = \frac{a}{2^b} \\ 10 \ll 1 \end{array}$$

$$10 \ll 1$$

$$1010$$

$$1010$$

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

$$8 \gg 2$$

$$1000$$

$$0010 \quad (2)_{10}$$

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

Hence proved.

Operator	Precedence	Precedence
operators	Unary operators	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
=		

Now Bitwise operators exists

we can overwrite all these operations using the bracket.

If operators of same precedence comes then the associative property applies

$$4 * 5 / 2$$

$\xrightarrow{\quad}$
 $\quad \quad \quad \rightarrow$
 first second
 $L \rightarrow R$

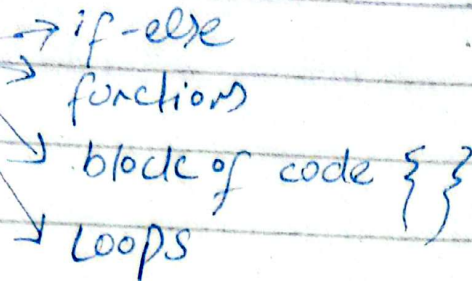
Scope

Area of accessibility/usability of the variables.

① Local

② Scope (Global)

① Local



② Global → accessible everywhere

Data Type modifiers

Change meaning of datatypes

long

short

long long

signed

unsigned

Suppose

int x = 2 (4 bytes)

011011011011

upto 32

MS

0

1

pos

neg

31 bits

Combination = 2^{31}

0 to $2^{31}-1$

-2^{31}

If 2^{32} comes int does not have capacity to store the value because it has the capacity of -2^{31} to $2^{31}-1$

So, we can change the capacity

→ long int

long double

long gives extra 4 bytes

So,

long int 64 bits

-2^{63} to $2^{63}-1$

→ short 2 bytes

will down the capacity to 2 bytes.

→ long long 8 bytes

→ signed

int is signed by default

signed means can save $\left\{ \begin{array}{l} +ve \text{ values} \\ -ve \text{ values} \end{array} \right.$

→ unsigned

only positive values

(MS)

No need of
most significant

to 32 bits

So size increases upto

-2^{32} to $2^{32}-1$

double of

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

which is