C PROGRAMING

by Ketan.Kore@ Sunbeam Infotech

Operators Precedence and Associativity

OPERATOR	TYPE	ASSOCIAVITY
( ) [ ] . ->		left-to-right
++ -- + - ! ~ (*type*) * & sizeof	Unary Operator	right-to-left
* / %	Arithmetic Operator	left-to-right
+ -	Arithmetic Operator	left-to-right
<< >>	Shift Operator	left-to-right
< <= > >=	Relational Operator	left-to-right
== !=	Relational Operator	left-to-right
&	Bitwise AND Operator	left-to-right
^	Bitwise EX-OR Operator	left-to-right
|	Bitwise OR Operator	left-to-right
&&	Logical AND Operator	left-to-right
||	Logical OR Operator	left-to-right
? :	Ternary Conditional Operator	right-to-left
= += -= *= /= %= &= ^= |= <<= >>=	Assignment Operator	right-to-left
,	Comma	left-to-right

Short-hand operators

- Short-hand operators will change value in variable.
- +=, -=, …
  - num+=2;
  - num=+2;
  - num-=2;
  - num=-2;

- Pre-increment/decrement
  - x = ++a;
  - y = --b;

- Post-increment/decrement
  - x = a++;
  - y = b--;

Comma, Relational and logical operators

- Comma operator
  - evaluate to right-most value.
  - have lowest precedence.

- Relational and logical operators result in 0 or 1.
  - 0 – indicate false condition
  - 1 – indicate true condition
- Relational operators
  - <, >, <=, >=, ==, !=
- Logical operators
  - &&, ||, !

Logical operators

- Logical operators
  - &&, ||, !

P	Q	P && Q	P || Q	!P
T	T	T	T	F
T	F	F	T	F
F	T	F	T	T
F	F	F	F	T

  - Logical operators operate according to the truth table given above
  - Logical AND and Logical OR operator guarantee left to right evaluation
  - Logical NOT OperatorIt is used to reverse the logical state of its operand. If a condition is true, then Logical NOT operator will make it false.

Bit-wise operators

The C language provides six operators for bit manipulation they operate on the individual bits of the operands . The Bitwise operators available in C are

- Bitwise AND &

A	B	A&B
0	0	0
0	1	0
1	0	0
1	1	1

Bitwise AND operators on the individual bits of the operand according to the truth table shown above

Example : - 10 & 5

0000 1010 -> Binary of 10

0000 0101 -> Binary of 5

0000 0000 → O/P is 0

- Bitwise OR

x	y	x | y
1	1	1
1	0	1
0	1	1
0	0	0

Bitwise OR operators on the individual bits of the operand according to the truth table shown above

Example : - 10 | 5

0000 1010 -> Binary of 10

0000 0101 -> Binary of 5

0000 1111 → O/P is 15

- Bitwise XOR ^

Input		Output
A	B	A xor B
0	0	0
0	1	1
1	0	1
1	1	0

Bitwise XOR operators on the individual bits of the operand according to the truth table shown above

Example : - 10 | 5

0000 1010 -> Binary of 10

0000 0101 -> Binary of 5

0000 1111 → O/P is 15

- Bitwise NOT ~

  Bitwise NOT operator results in one’s compliment of its operand

NOT "~"	
INPUT	OUTPUT
0	1
1	0

- Left shift << and Right shift >>
  - The bitshift operators take two arguments, and looks like:
    - x << n : shifts the value of x left by n bits
    - x >> n : shifts the value of x right by n bits

- Left shift operator : - num << n = num * 2 raise to n
  - **5 << 2 = 5 * 2 to the power 2**

    5 * 4 = 20
- Right Shift operator :- num >> n = num / 2 raise to n
  - **9 >> 1 = 9 / 2 to the power n**

    = 4

Twisters

- If precedence of two operators in an expression is same, their associativity is considered to decide their binding with operands.
- Data type conversions and ranges should be considered while doing arithmetic operations.
- sizeof() is compile time operator. Expressions within sizeof are not executed at runtime.
- Relational and logical operators always result in 0 or 1.
- In logical AND, if first condition is false, second condition is not evaluated. Result is false.
- In logical OR, if first condition is true, second condition is not evaluated. Result is true.
- Increment/Decrement operators in arithmetic expressions are compiler dependent.

Thank you!

Ketan Kore <ketan.kore@sunbeaminfo.com>