Solving Mathematical Equations and BitManipulations

Organised & Supported by **RuggedBOARD**

Agenda

- Operators
- Types of Operators
- Properties of operators
- More on Bitwise operators

Operators

Def:
- Special symbol.
- That can perform a specific operation without changing their meaning.
- All operators are special characters. But, all special characters are not operators.

Operators are classified into 3 categories:

Unary Operators: Single operand with an operator is called "Unary Operators".
Ex. Increment and Decrement operators.

Binary Operators: Two operands with an operator are called "Binary Operators".
Ex. Arithmetic, Logical, Relational etc.

Ternary Operators: Three expressions with an operator are called "Ternary operators".
Ex. Ternary operators.

Types of Operator

1. Arithmetic Operators
   -, +, *, /, %, ++, --
2. Relational Operators
   <, <=, >, >=,==, !=
3. Logical Operators
   !, &&, ||
4. Bitwise Operators
   &, |, ^, ~, <<, >>
5. Assignment Operators
   =, +=, -=, *=, /=, %=, <<=, >>=
6. Misc Operators
   sizeof(), &, *, ?:

Types of operators

Types of operators:

Arithmetic operators :

The special characters +, - , *, / and % can performing their corresponding arithmetic operations are called 'Arithmetic Operators'.

Relational operators :

The special characters >, <, >=, <=, == and != are called Relational operators.

These operators can be used for designing a condition only.

Note:

Any two quantities are combined with a relational operator that is called a condition.

Any condition can returns either TRUE or FALSE only. But, not both at a time.

Types of operators

Logical operators :

- The special characters &&, || and ! are called 'Logical operators'.
- Those can perform their corresponding actions.
- These operators can be used for joining more than one condition.
- It returns either TRUE or FALSE only. But, not both.

Assignment operator :

- The special character '=' is called an Assignment operator.
- That means the right quantity is assigned to left side variable.

Syn: var = constant / var / exprn ;

```
Eg: x = 3              // constant assignment
     y = x             // variable assignment
    z = x + 10 * y     // expression assignment
```

Types of operators

Conditional Operators [Ternary Operators] :

- The symbols ? and : are called Conditional operators. Those are also called as Ternary operators.
- A statement formed with these two operators is called Conditional operator statement.

syn: expr1 ? expr2 : expr3 ;

Here, expr1 is a condition. expr2 is a true statement and expr3 is a false statement.

Note:

The conditional operator statement can executes only one statement
(either TRUE or FALSE statement)

Types of operators

Increment and / or Decrement operators :

The symbols ++ and -- are called Increment and Decrement operators.

These two operators are applied to variables only.
Increment : It means 1 is added to the previous value of that variable.
This operator is placed in two ways.

Pre-increment operator:
Before assigning to a variable its value will be incremented by 1
Syn: ++ var ; ==> var = var + 1;

Eg:x = 5;
y= ++x;

Post-increment operator :
After assigning to a variable, its value will be incremented by 1.
Syn: var ++ ; ==> var = var + 1;

Eg: x = 5
Y= x ++;

Types of operators

Decrement : It means 1 is deducting from the previous value of that var. This operator is placed in two ways.

Pre-decrement operator :

Before assigning to a variable its value will be decremented by 1.

Syn: -- var ; ==> var = var - 1;

Eg: x = 5;

y = -- x;

Post-decrement operator:

After assigning to a variable, its value will be decremented by 1.

Syn: var --; ==> var = var - 1;

Eg: x = 5;

y = x --;

Bitwise operators:

The following operators can be used for performing their actions on bit values only.

Operator	Meaning
&	Bitwise AND
|	Bitwise OR
^	Exclusive OR (XOR)
~	Complement
>>	Right shift
<<	Left shift
>>>	Right shift fill withzero

Properties of operators

1. Arity of Operator

It is the number of operands an operator can take

e.g. a++ => arity = 1

a+b => arity = 2

2. Precedence of Operator

When there are more than one operators involved in an expression then it is the precedence of the operators which decides that which operator should be evaluated first

e.g. 1 > 2 + 3 && 4

/*there are 3 operators involved in the expression >, + and && */

The order of evaluation of this expression will be

1.+ => 2+3=5

2.> => 1>5=0

3.&& => 0&&4=0

It's better to write this expression like

((1>(2+3))&&4)

Note: In C, zero means false, and anything else means true

3. Associativity of Operator

This property is used when two or more operators in an expression have the same precedence

It can be

- left associative => expression on the left should be evaluated first, e.g. + operator is left associative
- right associative => expression on the right should be evaluated first, e.g. ^ operator is right associative

Properties of operators

Precedence	Operator	Description	Associativity
1	++ --	Suffix/postfix increment and decrement	Left-to-right
	()	Function call	
	[]	Array subscripting	
	.	Structure and union member access	
	->	Structure and union member access through pointer	
	(*type*){*list*}	Compound literal(C99)	
2	++ --	Prefix increment and decrement[note 1]	Right-to-left
	+ -	Unary plus and minus	
	! ~	Logical NOT and bitwise NOT	
	(*type*)	Cast	
	*	Indirection (dereference)	
	&	Address-of	
	sizeof	Size-of[note 2]	
	_Alignof	Alignment requirement(C11)	
3	* / %	Multiplication, division, and remainder	Left-to-right
4	+ -	Addition and subtraction	
5	<< >>	Bitwise left shift and right shift	
6	< <=	For relational operators < and ≤ respectively	
	> >=	For relational operators > and ≥ respectively	
7	== !=	For relational = and ≠ respectively	
8	&	Bitwise AND	
9	^	Bitwise XOR (exclusive or)	
10	|	Bitwise OR (inclusive or)	
11	&&	Logical AND	
12	||	Logical OR	
13	?:	Ternary conditional[note 3]	Right-to-left
14[note 4]	=	Simple assignment	
	+= -=	Assignment by sum and difference	
	*= /= %=	Assignment by product, quotient, and remainder	
	<<= >>=	Assignment by bitwise left shift and right shift	
	&= ^= |=	Assignment by bitwise AND, XOR, and OR	
15	,	Comma	Left-to-right

More on Bitwise operators

Bitwise operators are very important in low level programming:

- Applications of Bitwise operators include

1. Checking file permissions
2. Low level device control
3. Error detection & Correction
4. Data Compression Algorithm
5. Encryption Algorithm

More on Bitwise operators

Understanding Bitwise Operators

1. NOT(~)

It simply switches the bits from 0->1 and 1->0

e.g.

```
unsigned char ch=5; //ch=0000 0101
ch = ~ch;  //ch=1111 1010
printf("%c",ch); //will print 250
```

2. AND(&)

It applies AND operation on the bits of the two numbers on which it is applied

e.g.

```
unsigned char ch1=5;          //ch1=0000 0101 = 5
unsigned char ch2=4;          //ch2=0000 0100 = 4
unsigned char ch3=ch1&ch2; //ch3=0000 0100=4
```

3. OR(|)

It applies OR operation on the bits of the two numbers on which it is applied

e.g.

```
unsigned char ch1 = 5;          //ch1=0000 0101 = 5
unsigned char ch2 = 4;          //ch2=0000 0100 = 4
unsigned char ch3=ch1 | ch2;//ch3=0000 0101=5
```

4. XOR(^)

It performs XOR operation on the two numbers on which it is applied

In XOR operation, the result is 1 if odd number of bits are 1, otherwise the result is 0

e.g.

```
unsigned char ch1 = 5;          //ch1=0000 0101 = 5
unsigned char ch2 = 4;          //ch2=0000 0100 = 4
unsigned char ch3=ch1 ^ ch2;//ch3=0000 0001=1
```

5. Left Shift(<<)

It adds n no. of 0-bit(s) on the right side of the bits of the number

Left shift by n-bits is like multiplying the number by 2 n

e.g.

5<<2 //it says that left shift 5 by 2-bits as 5 = 0000 0101

5 << 2 gives 0001 0100 and 0001 0100 = 20 also 5*2 2 =20

6. Right Shift(>>)

It pumps n no. of MSB(Most Significant Bit) on the left side of the bits of the number

i.e.

For Signed number it pumps 1's (called Arithmetic Shift)

For Unsigned numbers it pumps 0's (called Logical Shift)

Right shift by n-bits is like dividing the number by 2 n

e.g.

5>>2 //it says that right shift 5 by 2-bits as 5 = 0000 0101

so 5 >> 2 gives 0000 0001 = 1 and also 5/2 2 =1

More on Bitwise operators

Some Important Concepts of Bitwise Operators

AND (&)	OR (|)	XOR (^)
0&x =0	0|x =x	0^x =x
-1&x =x	-1|x =-1	-1^x =x
x&x =x	x|x =x	x^x =0
x&x =0	x|x =1	x^x =-1

More on Bitwise operators

Applications of Bitwise Operators:

1. Swapping
2. Check Ranges of various Datatypes
3. Checking Status of a bit
4. Setting a bit

Rugged Board

Thank You