BIT Notes

By Karna Bahadur Shrestha

Operators

An operator is a symbol which helps the user to command the computer to do a certain mathematical or logical manipulations. Operators are used in C language program to operate on data and variables.

Operators classified on the basis of no of operands

Unary Operators

The operators that require only one operand are known as unary operators. Such as unary minus,plus(-/+), increment decrement operators (++/--) etc.

Binary Operators

The operators that require two operands are known as binary operators. Such as addition, subtraction(-/+),multiplication, division etc.

Ternary Operators

The operators that requires three operands are known as ternary operator. For example. Conditional operator (? :)

Operators on the basis of utility(functions) of an operator

1. Arithmetic operators
2. Relational Operators
3. Logical Operators
4. Assignment Operators
5. Unary Operators
6. Conditional Operators
7. Bitwise Operators
8. Special Operators

1. **Arithmetic Operators**

All the basic arithmetic operations can be carried out in C. All the operators have almost the same meaning as in other languages. Both unary and binary operations are available in C language. Unary operations operate on a single operand, therefore the number 5 when operated by unary – will have the value –5.

Operator	Meaning
+	Addition or Unary Plus
–	Subtraction or Unary Minus
*	Multiplication
/	Division
%	Modulus Operator

Examples of arithmetic operators are:

x + y

x - y

-x + y

a * b + c

-a * b etc.,

Here a, b, c, x, y are known as operands. The modulus operator is a special operator in C language which evaluates the remainder of the operands after division.

Points to be remember

- The operands acted upon by arithmetic operators must represent numeric values.
- The division operator requires that the second operant must be non zero.
- The remainder operator requires that both operands must be integers and second operands be non zero.

Division Rule

When and operand of certain data type is divided by another operand of certain data type the result will depend upon their values as well as data type.

i. int/int=int
ii. float/float=float
iii. int/float=float
iv. float/int=float

2. Relational Operators

These operators are very helpful for making decisions. Depending upon the condition, it returns either 0 or 1. When the condition with these operators is true, 1 is returned. If the condition is false, it returns 0

Operator	Meaning	Example	Output(a=20,b=6)
<	Less than	a<b	0(20<6)
<=	Less than or equal to	a<=b	0(20<=6)
>	Greater than	a>b	1(20>6)
>=	Greater than or equal to	a>=b	1(20>=6)
==	Equal to	a==b	0(20==6)
!=	Not equal to	a!=b	1(20!=6)

Relational expressions are used in decision making statements of C language such as if, while and for statements to decide the course of action of a running program.

3. Logical Operators

These operators are generally used along with relation operators. Like relational operators, output of these operators is either True (1) or False (0).The operands of logical operators must be either Boolean (1 or 0) or expressions that produce Boolean values.

Different logical operators available in C programming language are tabulated with examples:

Operator	Meaning
&&	Logical AND
||	Logical OR
!	Logical NOT

Logical AND (&&)

This operator is used to evaluate 2 conditions or expressions with relational operators simultaneously. If both the expressions to the left and to the right of the logical operator is true then the whole compound expression is true.

Example: a > b && x = = 10

The expression to the left is a > b and that on the right is x == 10 the whole expression is true only if both expressions are true i.e., if "a" is greater than "b" and "x" is equal to 10.

Logical OR (||)

The logical OR is used to combine 2 expressions or the condition evaluates to true if any one of the 2 expressions is true.

Example: a < m || a < n

The expression evaluates to true if any one of them is true or if both of them are true. It evaluates to true if "a" is less than either "m" or "n" and when "a" is less than both "m" and "n".

Logical NOT (!)

The logical not operator takes single expression and evaluates to true if the expression is false and evaluates to false if the expression is true. In other words it just reverses the value of the expression.

Example: ! (x > y)

The NOT expression evaluates to true only if the value of "x" is less "y".

4. Assignment Operators

The Assignment Operator evaluates an expression on the right of the expression and substitutes it to the value or variable on the left of the expression.

Example x = a + b

Here the value of a + b is evaluated and substituted to the variable x. In addition, C has a set of shorthand assignment operators of the form.

var oper = exp;

Here var is a variable, exp is an expression and oper is a C binary arithmetic operator. The operator oper = is known as shorthand assignment operator.

Example x + = 1 is same as x = x + 1

Statement with simple assignment operator	**Statement with shorthand operator**
a = a + 1	a += 1
a = a – 1	a -= 1
a = a * (n+1)	a *= (n+1)
a = a / (n+1)	a /= (n+1)
a = a % b	a %= b

5. Increment/D Operators

The increment and decrement operators are one of the unary operators which are very useful in C language. They are extensively used in for and while loops. The syntax of the operators is given below:

++variablename;

variable name++;

– –variable name

variable name– –

The increment operator ++ adds the value 1 to the current value of operand and the decrement operator

– – subtracts the value 1 from the current value of operand. ++variable name and variable name++ mean the same thing when they form statements independently, they behave differently when they are used in expression on the right hand side of an assignment statement.

Consider the following :

m = 5;

y = ++m; (prefix)

In this case the value of y and m would be 6

Suppose if we rewrite the above statement as

m = 5;

y = m++; (post fix)

Then the value of y will be 5 and that of m will be 6. A prefix operator first adds 1 to the operand and then the result is assigned to the variable on the left. On the other hand, a postfix operator first assigns the value to the variable on the left and then increments the operand.

6. Conditional or Ternary Operator

The conditional operator consists of 2 symbols the question mark (?) and the colon (:)

The syntax for a ternary operator is as follows:

exp1 ? exp2 : exp3

The ternary operator works as follows:

exp1 is evaluated first. If the expression is true then exp2 is evaluated & its value becomes the value of the expression. If exp1 is false, exp3 is evaluated and its value becomes the value of the expression.

Note that only one of the expressions is evaluated.

For example

```
a = 10;
b = 15;

x = (a > b)? a: b
```

Here “x” will be assigned to the value of “b”. The condition follows that the expression is false therefore “b” is assigned to “x”.

/* Example : to find the maximum value using conditional operator*/

```
#include<stdio.h>
void main()       //start of the program
{
     int i,j,larger;   //declaration of variables
     printf (“Input 2 integers : ”);
     scanf(“%d %d”,&i, &j);
     larger = i > j ? i : j;
}    printf(“The largest of two numbers is %d \n”, larger);
   }
```

7. Bitwise Operators

C has a distinction of supporting special operators known as bitwise operators for manipulation data at bit level. A bitwise operator operates on each bit of data. Those operators are used for testing, complementing or shifting bits to the right on left. Bitwise operators may not be applied to a float or double.

There are three types of bitwise operators:

- Bitwise logical operators
- Bitwise shift operators
- One's complement operator

Bitwise logical operators

Bitwise logical operators perform logical test between two integer type operands. They work on their operands bit by bit from the least significant(i.e the right most) bit. There are three Bitwise Logical Operators. Bitwise AND (&), Bitwise OR(|), Bitwise Exclusive OR(^).

AND(&) performs logical AND between two operands.
OR(|)performs logical OR between two operands.
XOR(^)performs logical XOR between two operands.

Bitwise shift Operators

Bitwise shift operators are used to move bit patterns either to the left or to the right.

Left Shift(<<)

Operand<<n

The bits in the operands are shifted left by n positions.

Right Shift(>>)

Operand>>n

The bits in the operands are shifted right by n positions.

One's Complement operator(~)

It performs one complements of given integer number.

8. Special Operators

C supports some special operators of interest such as comma operator, size of operator, pointer operators (& and *) and member selection operators (. and ->). The size of and the comma operators are discussed here. The remaining operators are discussed in forth coming chapters.

The Comma Operator

The comma operator can be used to link related expressions together. Comma-linked lists of expressions are evaluated left to right and value of right most expression is the value of the combined expression.

For example the statement: value = (x = 10, y = 5, x + y);

First assigns 10 to x and 5 to y and finally assigns 15 to value. Since comma has the lowest precedence in operators the parenthesis is necessary. Some examples of comma operator are:

In for loops:
for (n=1, m=10; n <=m; n++,m++)

In while loops
While (c=getchar(), c != „10“)

Exchanging values
t = x, x = y, y = t;

The sizeof Operator

The operator size of gives the size of the data type or variable in terms of bytes occupied in the memory. The operand may be a variable, a constant or a data type qualifier.

Example: m = sizeof (sum);

n = sizeof (long int);

k = sizeof (235L);

The size of operator is normally used to determine the lengths of arrays and structures when their sizes are not known to the programmer. It is also used to allocate memory space dynamically to variables during the execution of the program.