Operators and Expressions

Definition: (Operator)

An *Operator* is a symbol that tells the computer to perform certain mathematical or logical calculations. Operators are used in programs to handle data and variables. Operators usually form a part of mathematical or logical expressions.

C operators can be classified into the following types

1. Arithmetic operators
2. Relational operators
3. Logical operators
4. Assignment operators
5. Increment and decrement operators
6. Conditional operators
7. Bitwise operators
8. Special operators

Definition:

An *expression* is a sequence of operands and operators. The value of an expression will be a single value.

For example **10*5+7** is an arithmetic expression. The value of the above expression is **57.**

ARITHMETIC OPERATORS:

Operator	Meaning
+	Addition or unary plus
-	Subtraction or unary minus
*	Multiplication
/	Division
%	Modulo division

The modulo division operator produces the remainder of an integer division.

Example: **15%7=1**

The modulo division operator **%** cannot be used on floating point data.

Integer arithmetic:

If in an arithmetic expression, the operands are integers, the expression is called an integer expression and the operation is called integer arithmetic.

Integer arithmetic always results in an integer value.

Example: Let a and b be two variables of integer data type with a=14 and b=4.

The usage of the operands a and b in arithmetic expressions and the resulting values are given below

a+b=18

a-b=10

a*b=56

a/b=3

a%b=2

Note:

- During integer division, if both the operands are of the same sign, the result is truncated towards zero. (Ex: 6/7=0 and -6/-7=0)
- If one of the operands is negative, then the result is either 0 or -1.
- During modulo division, the sign of the result is always the sign of the first operand.

Example: **-14%3=-2** **-14%-3=-2** **14%-3=2**

Real arithmetic:

An arithmetic operation involving only real operands is called real arithmetic. A real operand may assume values either in decimal or exponential notation.

Mixed mode Arihtmetic:

When one of the operands is real and the other is integer, the expression is called a mixed – mode arithmetic expression.

If either operand is of the real type, then only the real operation is performed and the result is always a real number.

Example: 15/ 10.0 = 1.5

But 15/ 10 = 1

RELATIONAL OPERATORS:

Comparisons can be done with the help of relational operators. An expression containing a relational operator is termed as a relational expression. The value of a relational expression is either one (or) zero. It is one, if the relation is true and zero, if the relation is false.

Example: **10 < 20 is true.**

But **20 < 10 is false.**

C supports six relational operators. The operators and their meanings are given below.

Operator	Meaning
<	is less than
<=	is less than or equal to
>	is greater than
>=	is greater than or equal to
==	is equal to
!=	is not equal to

Note: When arithmetic expressions are used on either side of a relational operator, the arithmetic expression will be evaluated first and then the results are compared.

- Relational expressions are used in decision statements such as if and while.

LOGICAL OPERATORS

C has the following three logical operators.

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

The logical operators && and || are used when we want to test more than one condition and make decisions.

An expression which combines two or more relational expressions is called as a logical expression (or) a compound relational expression.

Example: a>b && x==10

The value of a logical expression may be either one or zero according to the following truth table.

Example: Usage of logical expressions

- If(age>55 &&salary<1000)
- If(number<0||number>100)

ASSIGNMENT OPERATORS

Assignment operators are used to assign the result of an expression to a variable.

'=' is called the assignment operator.

C has a set of ' shorthand' assignment operators.

The general form is

v op=exp;

where *v* is a variable

op is a C binary arithmetic operator

op= is called the shorthand assignment operator

exp is any expression

The assignment statement **v op = exp;** is equivalent to **v = v op(exp);**

Example: x+=y+1; is equivalent to x=x+(y+1);

In the above example, the shorthand operator += means 'add y+1 to x'

Some of the commonly used shorthand assignment operators are given below:

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

Advantages of using shorthand assignment operators:

1. What appears on the left hand side need not be repeated and therefore it becomes easier to write.
2. The statement is more concise and easier to read.
3. The statement is more efficient.

Example: Consider the statement

```
5*j-2= 5*j-2 + delta;
```

The above statement can be simply written with the help of shorthand assignment operator '+=" as

```
5*j-2+=delta;
```

INCREMENT AND DECREMENT OPERATORS

The operator '++' is called the increment operator and the operator '--' is called the decrement operator.

The operator ++ adds 1 to the operand and the operator -- subtracts 1.

Both the operators have the following forms:

++m; or m++;

--m; or m--;

++m; is equivalent to m= m+1;

--m; is equivalent to m= m-1;

The increment and decrement statements are used in for and while loops.

Note:

Consider the following statements

m = 5;

y = ++m;

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

Whereas the following statements

m =5;

y = m++;

woud result in m =6 and y =5

Thus even though ++m and m++ means the same thing, they behave differently when used in expressions as indicated above.

Hence a prefix operator first adds 1 to the operand and then the result is assigned to the variable on left. On the other hand, a postfix operator first assigns the value to the variable on left and then increments the operand.

CONDITIONAL OPERATOR:

The operator pair " ?:" is called the conditional operator . This operator is used in conditional expression as below

exp1 ? exp2: exp3

where exp1, exp2 and exp3 are expressions.

The operator ?: works as follows:

exp1 is evaluated first. If it is true (or) non zero, then exp2 is evaluated and it becomes the value of the expression.If exp1 is false, then exp3 is evaluated and its value becomes the value of the expression.

Note: Only one of the expressions (either exp2 or exp3 is evaluated.)

BITWISE OPERATORS

Bitwise operators are used for testing the bits, or shifting them right or left.
Bitwise operators may not be applied to float or double.

Operator	Meaning
&	bitwise AND
|	bitwise OR
^	bitwise exclusive OR
<<	shift left
>>	shift right

SPECIAL OPERATORS

C supports special operators such as comma operator, size of operator, and pointer operators.

The comma operator:

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

Example:

value = (x= 10, y=5, x+y);

Here the value 10 is first assigned to x, then value 5 is assigned to y, and finally 15 is assigned to the variable value.

Usage of comma operator in for loops:

Example: **for(n=1,m=10;n<=m;n++,m++)**

Usage in while loops: Example: **while(c=getchar(), c!='10')**

The sizeof operator:

The sizeof operator, when used with an operand, returns the number of bytes the operand occupies. The operand may be a variable, a constant or a data type qualifier.

The sizeof operator is used to determine the lengths of arrays and structures, when their sizes are not known.

Example: m= sizeof(sum);

n=sizeof(long int);

ARITHMETIC EXPRESSION

An arithmetic expression is a combination of variables, constants and arithmetic operators arranged as per the syntax of the language.. C can handle any complex mathematical expressions.

Examples of C expressions:

Algebraic expression	**C expression**
ab – c	a * b – c
(m + n) (x + y)	(m + n) * (x + y)
ab/c)	(a*b)/c
$3x^2+2x+1$	3*x*x + 2*x +1

Evaluation of expressions:

Expressions are evaluated using an assignment statement of the form

variable =expression;

variable is any valid variable name.

The expression is evaluated first and the result then replaces the previous value of the variable on the left hand side. All variable used in the expression must be assigned value before evaluation is made.

Example:

x= a*b-c;

y=b/c*a;

PRECEDENCE OF ARITHMETIC OPERATORS

There are two distinct priority levels of arithmetic operators

High priority* / %

Low priority + -

An arithmetic expression without parentheses will be evaluated from left to right using the above rule.

The basic evaluation procedure includes two passes through the expression.

During the first pass, the high priority operators(if any) are applied as they are encountered.

During the second pass, the low priority operators(if any) are applied as they are encountered.

Example: Consider the following statement

x= 9-12/3+3*2-1

The above statement is evaluated as follows:

First pass:

Step1: x=9-4+3*2-1

Step2: x= 9-4+6-1

Second pass:

Step3: x=5+6-1

Step4: x=11-1

Step5: x=10

Note : The order of evaluation can be changed by introducing paranthases into an expression.

Example: Consider the statement x= 9-12/(3+3)*(2-1)

The above statement is evaluated as follows:

First pass: Step1: x=9-12/6*(2-1)

Step2: x=9-12/6*1

Second pass: Step3: x= 9-2*1

Step4: x= 9-2

Third pass: Step5: x =7