OPERATOR IN C

An operator is a symbol that tells the compiler to perform specific mathematical or logical functions. C language is rich in built-in operators and provides the following types of operators

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Bitwise Operators
- Assignment Operators
- Misc Operators

We will, in this chapter, look into the way each operator works.

Arithmetic Operators

The following table shows all the arithmetic operators supported by the C language. Assume variable **A** holds 10 and variable **B** holds 20 then −

Show Examples

Operator	Description	Example
+	Adds two operands.	A + B = 30
−	Subtracts second operand from the first.	A − B = -10
*	Multiplies both operands.	A * B = 200
/	Divides numerator by de-numerator.	B / A = 2
%	Modulus Operator and remainder of after an integer division.	B % A = 0 5%5=0
++	Increment operator increases the integer value by one. a++=a+1	A++ = 11 A++=A+1=5+1=6

--	Decrement operator decreases the integer value by one.	A-- = 9=10-1

Relational Operators

The following table shows all the relational operators supported by C. Assume variable **A** holds 10 and variable **B** holds 20 then −

Show Examples

Operator	Description	Example
= =	Checks if the values of two operands are equal or not. If yes, then the condition becomes true. if(a= =b) , 10= =10	(A == B) is true.
!=	Checks if the values of two operands are equal or not. If the values are not equal, then the condition becomes true. if(5!=10)	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand. If yes, then the condition becomes true. 5>10	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand. If yes, then the condition becomes true. 5<25	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand. If yes, then the condition becomes true. 10>=9	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand. If yes, then the condition becomes true.	(A <= B) is true.

Logical Operators

Following table shows all the logical operators supported by C language. Assume variable **A** holds 1 and variable **B** holds 0, then – A=1, B=0

Show Examples

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non-zero, then the condition becomes true. if (A>10 && B<20) true 15>10 && 10<20	(A && B) is false.
||	Called Logical OR Operator. If any of the two operands is non-zero, then the condition becomes true. if (A=10 || B= 20) true A=10, B=5	(A || B) is true.
!	Called Logical NOT Operator. It is used to reverse the logical state of its operand. If a condition is true, then Logical NOT operator will make it false. !(A&&B==5) true	!(A && B) is true.

Bitwise Operators

Bitwise operator works on bits and perform bit-by-bit operation. The truth tables for &, |, and ^ is as follows −

p	q	p & q	p | q	p ^ q
0	0	0	0	0
0	1	0	1	1

1	1	1	1	0
1	0	0	1	1

Assume A = 60 and B = 13 in binary format, they will be as follows −

A = 0011 1100

B = 0000 1101

A&B = 0000 1100

A|B = 0011 1101

A^B = 0011 0001

Assignment Operators:

The following table lists the assignment operators supported by the C language −

Show Examples

Operator	Description	Example
=	Simple assignment operator. Assigns values from right side operands to left side operand A=5,B=6, C=A+B=11	C = A + B will assign the value of A + B to C
+=	Add AND assignment operator. It adds the right operand to the left operand and assign the result to the left operand. C+=1; C=C+1	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator. It subtracts the right operand from the	C -= A is equivalent

	left operand and assigns the result to the left operand. C-=1; C=C-1	to C = C - A
*=	Multiply AND assignment operator. It multiplies the right operand with the left operand and assigns the result to the left operand. C*=1; C=C*1 =10*1=10	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator. It divides the left operand with the right operand and assigns the result to the left operand. C/=1; C=C/1	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator. It takes modulus using two operands and assigns the result to the left operand. C%=1; C=C%1	C %= A is equivalent to C = C % A
<<=	Left shift AND assignment operator. C<<=1; C=C<<1	C <<= 2 is same as C = C << 2
>>=	Right shift AND assignment operator. C>>=1; C=C>>1	C >>= 2 is same as C = C >> 2
&=	Bitwise AND assignment operator. C&=2, C=C&2	C &= 2 is same as C = C & 2

^=	Bitwise exclusive OR and assignment operator.	C ^= 2 is same as C = C ^ 2
|=	Bitwise inclusive OR and assignment operator.	C |= 2 is same as C = C | 2