Operators in C++

Operators are nothing but symbols that tell the compiler to perform some specific operations. Operators are of the following types -

1. Arithmetic Operators

Arithmetic operators perform some arithmetic operation on one or two operands. Operators that operate on one operand are called unary arithmetic operators and operators that operate on two operands are called binary arithmetic operators.
+,-,*,/,% are binary operators.
++, -- are unary operators.

Suppose : A=5 and B=10

Operator	Operation	Example
+	Adds two operands	A+B = 15
-	Subtracts right operand from left operand	B-A = 5
*	Multiplies two operands	A*B = 50
/	Divides left operand by right operand	B/A = 2
%	Finds the remainder after integer division	B%A = 0
++	Incrementer	A++ = 6
--	Decrementer	A-- = 4

Pre-incrementer : It increments the value of the operand instantly.
Post-incrementer : It stores the current value of the operand temporarily and only after that statement is completed, the value of the operand is incremented.
Pre-decrementer : It decrements the value of the operand instantly.
Post-decrementer : It stores the current value of the operand temporarily and only after that statement is completed, the value of the operand is decremented.

Example -

```
int a=10;
int b;

b = a++;
cout<<a<<" "<<b<<endl;
```

Output : 11 10

```
int a=10;
int b;

b = ++a;
cout<<a<<" "<<b<<endl;
```

Output : 11 11

2. Relational Operators

Relational operators define the relation between 2 entities.
They give a boolean value as result i.e true or false.

Suppose : A=5 and B=10

Operator	Operation	Example
==	Gives true if two operands are equal	A==B is not true
!=	Gives true if two operands are not equal	A!=B is true
>	Gives true if left operand is more than right operand	A>B is not true
<	Gives true if left operand is less than right operand	A<B is true
>=	Gives true if left operand is more than right operand or equal to it	A>=B is not true
<=	Gives true if left operand is more than right operand or equal to it	A<=B is true

Example -

We need to write a program which prints if a number is more than 10, equal to 10 or less than 10. This could be done using relational operators with if else statements.

```
 int n;
cin>>n;


if(n<10){
   cout<<"Less than 10"<<endl;
}
else if(n==10){
   cout<<"Equal to 10"<<endl;
}
else{
   cout<<"More than 10"<<endl;
}
```

3. Logical Operators

Logical operators are used to connect multiple expressions or conditions together.
We have 3 basic logical operators.

Suppose : A=0 and B=1

Operator	Operation	Example
&&	AND operator. Gives true if both operands are non-zero	(A && B) is false
||	OR operator. Gives true if atleast one of the two operands are non-zero.	(A || B) is true
!	NOT operator. Reverse the logical state of operand	!A is true

Example -

If we need to check whether a number is divisible by both 2 and 3, we will use AND operator

(num%2==0) && num(num%3==0)

If this expression gives true value then that means that num is divisible by both 2 and 3.

(num%2==0) || (num%3==0)

If this expression gives true value then that means that num is divisible by 2 or 3 or both.

4. Bitwise Operators

Bitwise operators are the operators that operate on bits and perform bit-by-bit operations.

Suppose : A=5(0101) and B=6(0110)

Operator	Operation	Example
&	Binary AND. Copies a bit to the result if it exists in both operands.	0101 & 0110 0100
|	Binary OR. Copies a bit if it exists in either operand.	0101 | 0110 0111
^	Binary XOR. Copies the bit if it is set in one operand but not both.	0101 ^ 0110 0011
~	Binary Ones Complement. Flips the bit.	~0101 => 1010
<<	Binary Left Shift. The left operand's bits are moved left by the number of places specified by the right operand.	4 (0100) 4<<1 =1000 = 8

>>	Binary Right Shift Operator. The left operand's bits are moved right by the number of places specified by the right operand.	4>>1 =0010 = 2

If shift operator is applied on a number N then,

- N<<a will give a result N*2^a
- N>>a will give a result N/2^a

5. Assignment Operators

Operator	Operation	Example
=	Assigns value of right operand to left operand	A=B will put value of B in A
+=	Adds right operand to the left operand and assigns the result to left operand.	A+=B means A = A+B
-=	Subtracts right operand from the left operand and assigns the result to left operand.	A-=B means A=A-B
*=	Multiplies right operand with the left operand and assigns the result to left operand.	A*=B means A=A*B
/=	Divides left operand with the right operand and assigns the result to left operand.	A/=B means A=A/B

6. Misc Operators

Operator	Operation	Example
sizeof()	Returns the size of variable	If a is integer then sizeof(a) will return 4

Condition? X:Y	Conditional operator. If condition is true, then returns value of X or else value of Y	A+=B means A = A+B
Cast	Casting operators convert one data type to another.	int(4.350) would return 4.
Comma (,)	Comma operator causes a sequence of operations to be performed. The value of the entire comma expression is the value of the last expression of the comma-separated list.	

Precedence of Operators

Category	Operator	Associativity
Postfix	() [] -> . ++ - -	Left to right
Unary	+ - ! ~ ++ - - (type)* & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right

Conditional	?:	Right to left
Assignment	= += -= *= /= %=>>= <<= &= ^= |=	Right to left
Comma	,	Left to right

Examples –

Ques. Give the output of the following programs.

1.

```
#include<stdio.h>
int main()
{
        int a = 5;
        a = 1, 2, 3;
        printf("%d", a);
        return 0;
}
```

Ans. 1
(Priority for the values assigned to any variable is given from left to right)

2.

```
#include<stdio.h>
int main()
{
        int a;
        a = (1, 2, 3);
        printf("%d", a);
        return 0;
```

```
}
```

Ans. 3
(Priority for the values inside a brackets () assigned to any variable is given from right to left.)

3.

```
#include<stdio.h>

int main()

{

        int x = 2;

        (x & 1) ? printf("true") : printf("false");

        return 0;

}
```

Ans. False
(As & is a unary operator we have to assume all decimal values to binary(0's and 1's)
2_{10} = 00000010_2
Now we go for condition (00000010 & 00000001)
Clearly, condition false as it leads to 0 when multiplied.)

4.

```
#include<stdio.h>

int main()

{

        printf("%d",3 * 2--);

}
```

Ans. 6
(2-- stands meaningless)

5.

```
#include<stdio.h>
```

```
int main()
{
        int i = 10;
        i++;
        i * i;
        printf("%d\n",i);
        return 0;
}
```

Ans. 11
(i++ alone store the result in variable i.)

6.

```
#include<stdio.h>
int main()
{
        int a = 1, b = 3, c;
        c = b <<  a;
        b = c * (b * (++a)--);
        a = a >> b;
        printf("%d",b);
        return 0;
}
```

Ans. 36
(c = 0011 << 1
c = 0110
b = 6* (3*(2)--)
b = 6*6
)