Lecture 5: Operators

5.1 C++ Operators

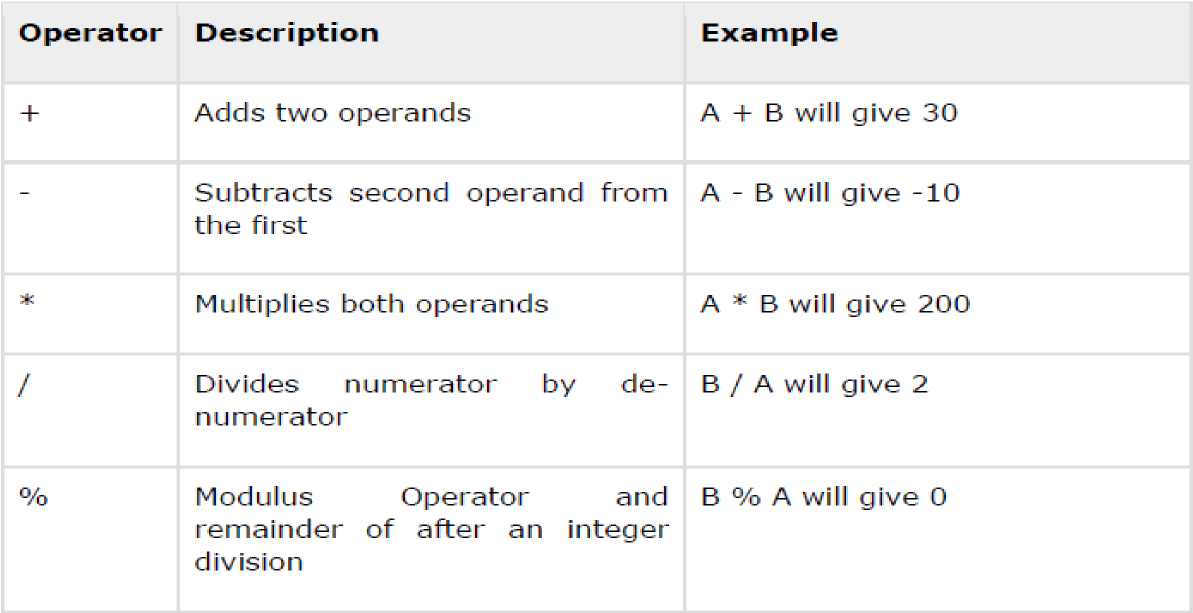
An operator is a symbol that tells the compiler to perform specific mathematical or logical calculations on operands(variables).

5.2 Types of operators available in C++

- Arithmetic / Mathematical operator
- Assignment operator
- Increment Decrement operator
- Relational operator
- Logical operator
- Unary operator

Arithmetic Operator:

There are following arithmetic operators supported by C++ language:
Assume variable A holds 10 and variable B holds 20, then:

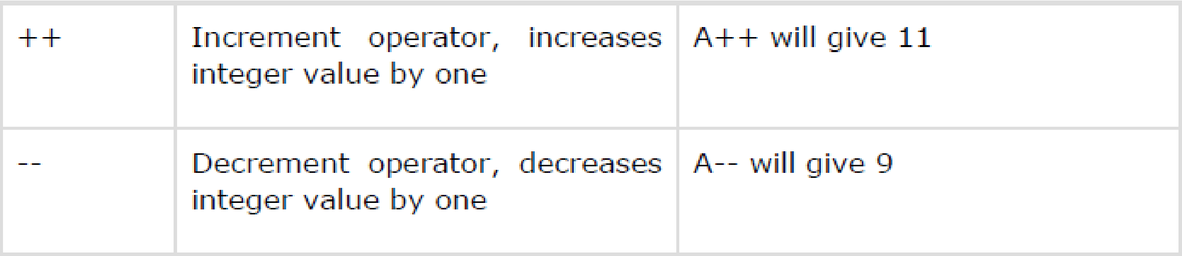


Operator	Description	Example
+	Adds two operands	A + B will give 30
-	Subtracts second operand from the first	A - B will give -10
*	Multiplies both operands	A * B will give 200
/	Divides numerator by de-numerator	B / A will give 2
%	Modulus Operator and remainder of after an integer division	B % A will give 0

Increment Decrement operator

Increment Decrement operators increase or decrease the operand by one value .

Example: Assume A=10, find the output result for the following expressiones:



++	Increment operator, increases integer value by one	A++ will give 11
--	Decrement operator, decreases integer value by one	A-- will give 9

Assignment operator

Assignment operator is used to copy value from right to left variable.

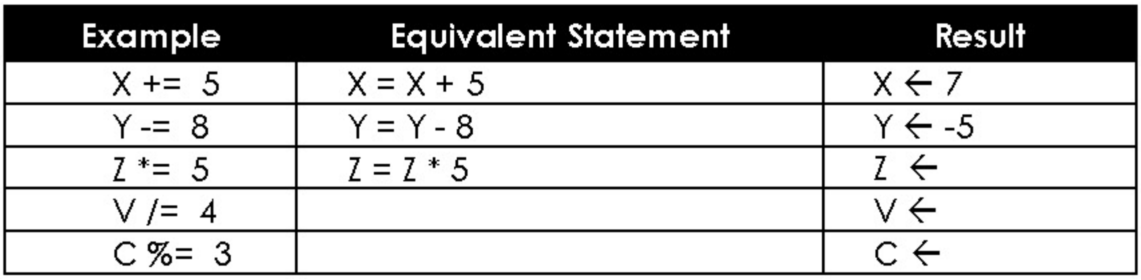
Suppose we have:

float X = 5, Y = 2;

=	Equal sign Copy value from right to left.	X = Y, Now both X and Y have 2
+=	**Plus Equal operator** to increase the left operand by right operand.	X+=5 → X=X+5 will give X= 10
-=	**Minus Equal operator** will return the subtraction of right operand from left operand.	Y-=1 → Y= Y-1 will give Y=1
***=**	**Multiply Equal operator** will return the product of right operand and left operand.	X *= Y → X = X * Y, X = 10
/=	**Division Equal operator** will divide right operand by left operand and return the quotient.	X /= Y → X = X / Y, X = 2.5
%=	Modulus Equal to operator will divide right operand by left operand and return the mod (Remainder).	X %= Y is similar to X = X % Y, now X is 1

Examples:

Rewrite the equevelment statmentes for the following expressions anf find the results, assume X=2, Y=3, Z=4, V= 12, C=8.



Example	Equivalent Statement	Result
X += 5	X = X + 5	X ← 7
Y -= 8	Y = Y - 8	Y ← -5
Z *= 5	Z = Z * 5	Z ←
V /= 4		V ←
C %= 3		C ←

Relational Operator:

Relational operators are used for checking conditions whether the given condition is true or false. If the condition is true, it will return non-zero value, if the condition is false, it will return 0.

Suppose we have,

int X = 5, Y = 2;

Operator	Name	Description	Example
>	Greater than	Check whether the left operand is greater than right operand or not.	(X > Y) will return true
<	Smaller than	Check whether the left operand is smaller than right operand or not.	(X < Y) will return false
>=	Greater than or Equal to	Check whether the left operand is greater or equal to right operand or not.	(X >= Y) will return true
<=	Smaller than or Equal to	Check whether the left operand is smaller or equal to right operand or not.	(X <= Y) will return false
==	Equal to	Check whether the both operands are equal or not.	(X == Y) will return false
!=	Not Equal to	Check whether the both operands are equal or not.	(X != Y) will return true

Operator	Name	Example
==	Equality	5 == 5 // gives 1
!=	Inequality	5 != 5 // gives 0
<	Less Than	5 < 5.5 // gives 1
<=	Less Than or Equal	5 <= 5 // gives 1
>	Greater Than	5 > 5.5 // gives 0
>=	Greater Than or Equal	6.3 >= 5 // gives 1

Logical Operators

Logical operators are used in situation when we have more then one condition in a single if statement.

Suppose we have,

int X = 5, Y = 2;

Operator	Name	Description	Example
&&	AND	Return true if all conditions are true, return false if any of the condition is false.	if(X > Y && Y < X) will return true
||	OR	Return false if all conditions are false, return true if any of the condition is true.	if(X > Y || X < Y) will return true
!	NOT	Return true if condition if false, return false if condition is true.	if(!(X>y)) will return false

Operator	Name	Example
&&	Logical And	5 < 6 && 6 < 6 // gives 0
||	Logical Or	5 < 6 || 6 < 5 // gives 1
!	Logical Negation (Not)	!(5 == 5) // gives 0

AND (&&) Table:

A	B	A && B
T	T	T
T	F	F
F	T	F
F	F	F

OR (| |) Table:

A	B	A | | B
T	T	T
T	F	T
F	T	T
F	F	F

NOT (!) Table:

A	!A
T	F
F	T

AND (&&) Table:

A	B	A && B
1	1	1
1	0	0
0	1	0
0	0	0

OR (| |) Table:

A	B	A | | B
1	1	1
1	0	1
0	1	1
0	0	0

NOT (!) Table:

A	!A
1	0
0	1

Examples: The following example to understand all the arithmetic operators available in C++.

```
 #include <iostream>
  using namespace std;

main()
{
int a = 21;
int b = 10;
int c ;
c = a + b;
cout << "Line 1 - Value of c is :" << c << endl ;

c = a - b;
cout << "Line 2 - Value of c is :" << c << endl ;

c = a * b;
cout << "Line 3 - Value of c is :" << c << endl ;

c = a / b;
cout << "Line 4 - Value of c is :" << c << endl ;

c = a % b;
cout << "Line 5 - Value of c is :" << c << endl ;

c = a++;
cout << "Line 6 - Value of c is :" << c << endl ;

c = a--;
cout << "Line 7 - Value of c is :" << c << endl ;

return 0;
```

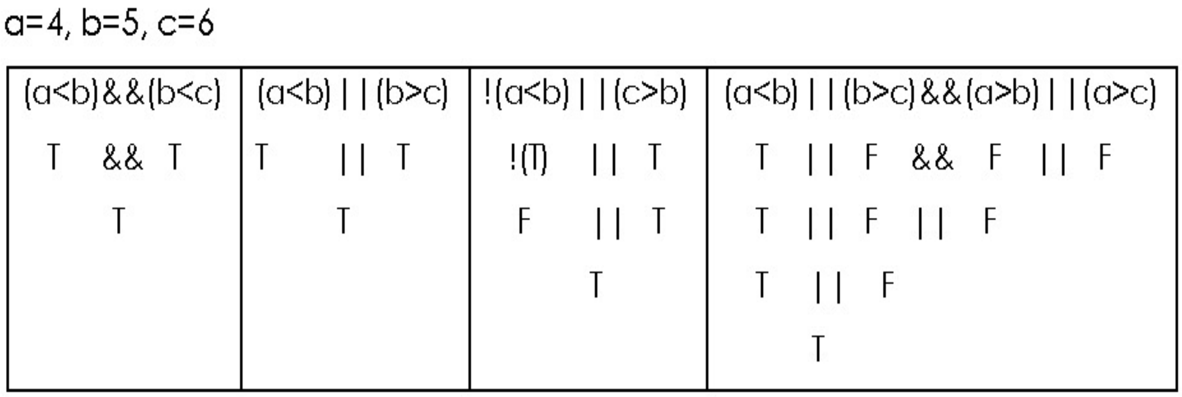
The output for the above program is:

```
Line 1 - Value of c is :31
Line 2 - Value of c is :11
Line 3 - Value of c is :210
Line 4 - Value of c is :2
Line 5 - Value of c is :1
Line 6 - Value of c is :21
Line 7 - Value of c is :22
```

Q/ What's Output:

```
#include<iostream>
using namespace std;
int main()
{ int x,y,z;
x=y=z=0;
x=++y + ++z;
cout<<x<<y<<z<<endl;
x=++y - --z;
cout<<x<<y<<z<<endl;
return 0;
}
```

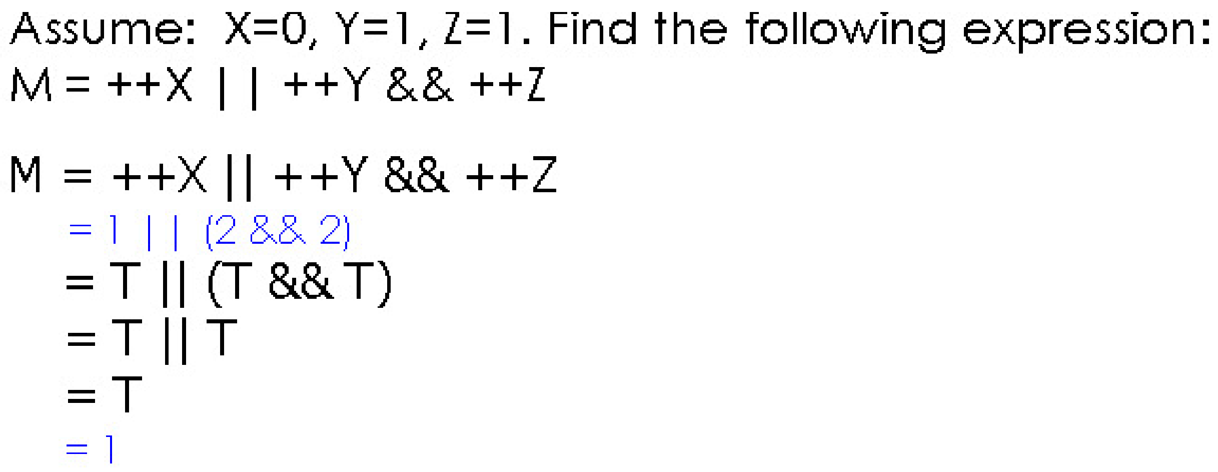
Example: find the output result for the following logical operationes:
Assume a=4, b=5, c=6



a=4, b=5, c=6

(a<b)&&(b<c)	(a<b)||(b>c)	!(a<b)||(c>b)	(a<b)||(b>c)&&(a>b)||(a>c)
T && T	T || T	!(T) || T	T || F && F || F
T	T	F || T	T || F || F
		T	T || F
			T

Example: find the output result for the following logical operationes:



Assume: X=0, Y=1, Z=1. Find the following expression:
M = ++X | | ++Y && ++Z

M = ++X || ++Y && ++Z
= 1 | | (2 && 2)
= T || (T && T)
= T || T
= T
= 1