F# - OPERATORS

An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations. F# is rich in built-in operators and provides the following types of operators −

- Arithmetic Operators
- Comparison Operators
- Boolean Operators
- Bitwise Operators

Arithmetic Operators

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

Show Example

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
**	Exponentiation Operator, raises an operand to the power of another	B**A will give 20^{10}

Comparison Operators

The following table shows all the comparison operators supported by F# language. These binary comparison operators are available for integral and floating-point types. These operators return values of type bool.

Assume variable A holds 10 and variable B holds 20, then −

Show Example

Operator	Description	Example
=	Checks if the values of two operands are equal or not, if yes then condition becomes true.	$A == B$ is not true.
<>	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	$A <> B$ is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	$A > B$ is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	$A < B$ is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes	$A >= B$ is not true.

	true.	
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	$A <= B$ is true.

Boolean Operators

The following table shows all the Boolean operators supported by F# language. Assume variable A holds **true** and variable B holds **false,** then −

Show Example

Operator	Description	Example
&&	Called Boolean AND operator. If both the operands are non-zero, then condition becomes true.	A && B is false.
||	Called Boolean OR Operator. If any of the two operands is non-zero, then condition becomes true.	$A \| \| B$ is true.
not	Called Boolean NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	not A && B is true.

Bitwise Operators

Bitwise operators work on bits and perform bit-by-bit operation. The truth tables for &&& $bitwiseAND$, ||| $bitwiseOR$, and ^^^ $bitwiseexclusiveOR$ are as follows −

Show Example

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 if A = 60; and B = 13; now 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

~~~A = 1100 0011

The Bitwise operators supported by F# language are listed in the following table. Assume variable A holds 60 and variable B holds 13, then −
~~~

Operator	Description	Example
&&&	Binary AND Operator copies a bit to the result if it exists in both operands.	A &&& B will give 12, which is 0000 1100
|||	Binary OR Operator copies a bit if it exists in either operand.	$A\|\|\|B$ will give 61, which is 0011 1101
^^^	Binary XOR Operator copies the bit if it is set in one operand but not both.	A ^^^ B will give 49, which is 0011 0001
~~	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	A will give -61, which is 1100 0011 in 2's complement form.
<<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A <<< 2 will give 240 which is 1111 0000
>>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >>> 2 will give 15 which is 0000 1111

Operators Precedence

The following table shows the order of precedence of operators and other expression keywords in the F# language, from lowest precedence to the highest precedence.

Show Example

Operator	Associativity
as	Right
when	Right
| *pipe*	Left
;	Right
let	Non associative
function, fun, match, try	Non associative
if	Non associative
→	Right
:=	Right
,	Non associative
or, ||	Left
&, &&	Left
< op, >op, =, |op, &op	Left
&&& , |||, ^^^, ~~~, <<<, >>>	Left
^ op	Right
::	Right

:?>, :?	Non associative
- op, +op, *binary*	Left
* op, /op, %op	Left
** op	Right
f x *functionapplication*	Left
| *patternmatch*	Right
prefix operators &plus;op, -op, %, %%, &, &&, !op, ~op	Left
.	Left
f*x*	Left
f<types>	Left

Loading [MathJax]/jax/output/HTML-CSS/jax.js