C# 7.2.1 Operator precedence and associativity

When an expression contains multiple operators, the precedence of the operators controls the order in which the individual operators are evaluated. For example, the expression `x + y * z` is evaluated as `x + (y * z)` because the `*` operator has higher precedence than the binary `+` operator. The precedence of an operator is established by the definition of its associated grammar production. For example, an additive-expression consists of a sequence of multiplicative-expressions separated by `+` or `-` operators, thus giving the `+` and `-` operators lower precedence than the `*`, `/`, and `%` operators.

The following table summarizes all operators in order of precedence from highest to lowest:

Section	**Category**	**Operators**	**Associativity**
7.5	Primary	`x.y` `f(x)` `a[x]` `x++` `x--` `new` `typeof` `checked` `unchecked`	Right to left
7.6	Unary	`+` `-` `!` `~` `++x` `--x` `(T)x`	Left to right
7.7	Multiplicative	`*` `/` `%`	Left to right
7.7	Additive	`+` `-`	Left to right
7.8	Shift	`<<` `>>`	Left to right
7.9	Relational and type testing	`<` `>` `<=` `>=` `is` `as`	Left to right
7.9	Equality	`==` `!=`	Left to right
7.10	Logical AND	`&`	Left to right
7.10	Logical XOR	`^`	Left to right
7.10	Logical OR	`|`	Left to right
7.11	Conditional AND	`&&`	Left to right
7.11	Conditional OR	`||`	Left to right
7.12	Conditional	`?:`	Right to left
7.13	Assignment	`=` `*=` `/=` `%=` `+=` `-=` `<<=` `>>=` `&=` `^=` `|=`	Right to left

When an operand occurs between two operators with the same precedence, the associativity of the operators controls the order in which the operations are performed:

Precedence and associativity can be controlled using parentheses. For example, `x + y * z` first multiplies `y` by `z` and then adds the result to `x`, but `(x + y) * z` first adds `x` and `y` and then multiplies the result by `z`.