

All C# Operators by precedence



16



Primary operators

- $x.y$ Member access operator
- $x?.y$ Null-conditional member access operator
- $a[i]$ Indexer operator
- $x?[y]$ Null-conditional indexer operator
- $f(x)$ Invocation operator
- $x++$ Increment operator
- $x--$ Decrement operator



Primary operators

- **new** creates a new instance of a type
- **typeof** obtains the System.Type instance for a type
- **checked** explicitly enable overflow checking
- **unchecked** suppress overflow-checking
- **default** produce default value of a type
- **nameof** obtains the name of a variable, type, member
- **delegate** creates an anonymous method
- **sizeof** returns the number of bytes occupied
by a variable of a given type
- **stackalloc** allocates a block of memory on the stack
- **x->y** pointer member access operator



Unary operators

- $+x$ Unary plus operator
- $-x$ Unary minus operator
- $!x$ Logical negation operator
- $\sim x$ Bitwise complement operator
- $++x$ Prefix increment operator
- $--x$ Prefix decrement operator



Unary operators

- $\wedge 1$ index from end operator
- $(T)x$ cast operator - explicit conversion x to type T
- **await** suspend evaluation until the asynchronous operation completes
- $\&x$ address-of operator returns the address of its operand
- $*x$ pointer indirection operator obtains the variable to which its operand points
- **true** returns true to indicate that its operand is definitely true
- **false** returns true to indicate that its operand is definitely false



Range operator

- `x..y` specifies the start and end of a range of indices as its operands (available in C# 8.0 and later)



Multiplicative operators

- $x * y$ multiplication operator computes the product of its operands
- x / y division operator divides its left-hand operand by its right-hand operand
- $x \% y$ remainder operator computes the remainder after dividing its left-hand operand by its right-hand operand



Additive operators

- $x + y$ addition operator computes the sum of its operands
- $x - y$ subtraction operator subtracts its right-hand operand from its left-hand operand



Shift operators

- $x \ll y$ **left-shift operator**
 - shifts its left-hand operand left by the number of bits defined by its right-hand operand
- $x \gg y$ **right-shift operator**
 - shifts its left-hand operand right by the number of bits defined by its right-hand operand



Relational operators

- $x < y$ Less than operator
- $x > y$ Greater than operator
- $x \leq y$ Less than or equal operator
- $x \geq y$ Greater than or equal operator



Type-testing operators

- **is**

the `is` operator checks if the runtime type of an expression result is compatible with a given type

- **as**

the `as` operator explicitly converts the result of an expression to a given reference or nullable value type



Equality operators

- $x == y$

the equality operator `==` returns true if its operands are equal, false otherwise

- $x != y$

the inequality operator `!=` returns true if its operands are not equal, false otherwise

Boolean logical / bitwise operators



- $x \& y$ **Logical/bitwise AND**
 - computes the logical AND of its operands.
 - computes the bitwise logical AND of its operands



- $x \wedge y$ **Logical/bitwise exclusive OR**
 - computes the logical exclusive OR of its operands.
 - computes the bitwise exclusive OR of its operands



- $x | y$ **Logical/bitwise OR**
 - computes the logical OR of its operands.
 - computes the bitwise logical OR of its operands

Conditional operators



$x \ \&\& \ y$ **Conditional logical AND operator**

the conditional logical AND operator $\&\&$, also known as the "short-circuiting" logical AND operator, computes the logical AND of its operands. The result of $x \ \&\& \ y$ is true if both x and y evaluate to true. Otherwise, the result is false. If x evaluates to false, y is not evaluated



$x \ || \ y$ **Conditional logical OR operator**

the conditional logical OR operator $||$, also known as the "short-circuiting" logical OR operator, computes the logical OR of its operands. The result of $x \ || \ y$ is true if either x or y evaluates to true. Otherwise, the result is false. If x evaluates to true, y is not evaluated.

14

Null-coalescing operator

- $x ?? y$

the null-coalescing operator $??$ returns the value of its left-hand operand if it isn't null; otherwise, it evaluates the right-hand operand and returns its result.



Conditional operator

- $c ? t : f$

ternary conditional operator, evaluates a boolean expression and returns the result of one of the two expressions, depending on whether the Boolean expression evaluates to true or false

16

Assignment operators

- $x = y$
- $x += y$
- $x -= y$
- $x *= y$
- $x /= y$
- $x \% = y$
- $x \& = y$
- $x |= y$
- $x \wedge = y$
- $x << = y$
- $x >> = y$
- $x ?? = y$

„ $x \Delta = y$ “ is equivalent to „ $x = x \Delta p$ “

16

Lambda declaration operator

=>

the lambda operator => separates
the input parameters on the left side
from the lambda body on the right side