**C# Operators**

Updated: July 2008

This page is specific to

**Microsoft Visual Studio 2008/.NET Framework 3.5**

Other versions are also available for the following:

[Microsoft Visual Studio 2003/.NET Framework 1.1](http://msdn.microsoft.com/en-us/library/6a71f45d(VS.71).aspx)

[Microsoft Visual Studio 2005/.NET Framework 2.0](http://msdn.microsoft.com/en-us/library/6a71f45d(VS.80).aspx)

[.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/6a71f45d(VS.85).aspx)

[Microsoft Visual Studio 2010/.NET Framework 4.0](http://msdn.microsoft.com/en-us/library/6a71f45d(VS.100).aspx)

C# provides a large set of operators, which are symbols that specify which operations to perform in an expression. Operations on integral types such as **==**, **!=**, **<**, **>**, **<=**, **>=**, **binary +**, **binary -**, **^**, **&**, **|**, **~**, **++**, **--**, and **sizeof()** are generally allowed on enumerations. In addition, many operators can be [overloaded](http://msdn.microsoft.com/en-us/library/8edha89s.aspx) by the user, thus changing their meaning when applied to a user-defined type.

The following table lists the C# operators grouped in order of precedence. Operators within each group have equal precedence.

|  |  |
| --- | --- |
| **Operator category** | **Operators** |
| Primary | [x.y](http://msdn.microsoft.com/en-us/library/6zhxzbds.aspx)  [f(x)](http://msdn.microsoft.com/en-us/library/0z4503sa.aspx)  [a[x]](http://msdn.microsoft.com/en-us/library/a3hd7ste.aspx)  [x++](http://msdn.microsoft.com/en-us/library/36x43w8w.aspx)  [x--](http://msdn.microsoft.com/en-us/library/wc3z3k8c.aspx)  [new](http://msdn.microsoft.com/en-us/library/fa0ab757.aspx)  [typeof](http://msdn.microsoft.com/en-us/library/58918ffs.aspx)  [checked](http://msdn.microsoft.com/en-us/library/74b4xzyw.aspx)  [unchecked](http://msdn.microsoft.com/en-us/library/a569z7k8.aspx)  [->](http://msdn.microsoft.com/en-us/library/s8bz4d5h.aspx) |
| Unary | [+](http://msdn.microsoft.com/en-us/library/k1a63xkz.aspx)  [-](http://msdn.microsoft.com/en-us/library/wch5w409.aspx)  [!](http://msdn.microsoft.com/en-us/library/f2kd6eb2.aspx)  [~](http://msdn.microsoft.com/en-us/library/d2bd4x66.aspx)  [++x](http://msdn.microsoft.com/en-us/library/36x43w8w.aspx)  [--x](http://msdn.microsoft.com/en-us/library/wc3z3k8c.aspx)  [(T)x](http://msdn.microsoft.com/en-us/library/0z4503sa.aspx)  [true](http://msdn.microsoft.com/en-us/library/eahhcxk2.aspx)  [false](http://msdn.microsoft.com/en-us/library/67bxt5ee.aspx)  [&](http://msdn.microsoft.com/en-us/library/sbf85k1c.aspx)  [sizeof](http://msdn.microsoft.com/en-us/library/eahchzkf.aspx) |
| Multiplicative | [\*](http://msdn.microsoft.com/en-us/library/z19tbbca.aspx)  [/](http://msdn.microsoft.com/en-us/library/3b1ff23f.aspx)  [%](http://msdn.microsoft.com/en-us/library/0w4e0fzs.aspx) |
| Additive | [+](http://msdn.microsoft.com/en-us/library/k1a63xkz.aspx)  [-](http://msdn.microsoft.com/en-us/library/wch5w409.aspx) |
| Shift | [<<](http://msdn.microsoft.com/en-us/library/a1sway8w.aspx)  [>>](http://msdn.microsoft.com/en-us/library/xt18et0d.aspx) |
| Relational and type testing | [<](http://msdn.microsoft.com/en-us/library/z5wecxwa.aspx)  [>](http://msdn.microsoft.com/en-us/library/yxk8751b.aspx)  [<=](http://msdn.microsoft.com/en-us/library/hx063734.aspx)  [>=](http://msdn.microsoft.com/en-us/library/a59bsyk4.aspx)  [is](http://msdn.microsoft.com/en-us/library/scekt9xw.aspx)  [as](http://msdn.microsoft.com/en-us/library/cscsdfbt.aspx) |
| Equality | [==](http://msdn.microsoft.com/en-us/library/53k8ybth.aspx)  [!=](http://msdn.microsoft.com/en-us/library/3tz250sf.aspx) |
| Logical AND | [&](http://msdn.microsoft.com/en-us/library/sbf85k1c.aspx) |
| Logical XOR | [^](http://msdn.microsoft.com/en-us/library/zkacc7k1.aspx) |
| Logical OR | [|](http://msdn.microsoft.com/en-us/library/kxszd0kx.aspx) |
| Conditional AND | [&&](http://msdn.microsoft.com/en-us/library/2a723cdk.aspx) |
| Conditional OR | [||](http://msdn.microsoft.com/en-us/library/6373h346.aspx) |
| Conditional | [?:](http://msdn.microsoft.com/en-us/library/ty67wk28.aspx) |
| Assignment | [=](http://msdn.microsoft.com/en-us/library/sbkb459w.aspx)  [+=](http://msdn.microsoft.com/en-us/library/sa7629ew.aspx)  [-=](http://msdn.microsoft.com/en-us/library/2y9zhhx1.aspx)  [\*=](http://msdn.microsoft.com/en-us/library/s2bkaksf.aspx)  [/=](http://msdn.microsoft.com/en-us/library/d31sybc9.aspx)  [%=](http://msdn.microsoft.com/en-us/library/ydwa9zh0.aspx)  [&=](http://msdn.microsoft.com/en-us/library/e669ax02.aspx)  [|=](http://msdn.microsoft.com/en-us/library/h5f1zzaw.aspx)  [^=](http://msdn.microsoft.com/en-us/library/0zbsw2z6.aspx)  [<<=](http://msdn.microsoft.com/en-us/library/ayt2kcfb.aspx)  [>>=](http://msdn.microsoft.com/en-us/library/23as4533.aspx) |
| Null-coalescing | [??](http://msdn.microsoft.com/en-us/library/ms173224.aspx) |
| Lambda | [=> Operator (C# Reference)](http://msdn.microsoft.com/en-us/library/bb311046.aspx) |

http://i.msdn.microsoft.com/Global/Images/clear.gif Arithmetic Overflow

The arithmetic operators ([+](http://msdn.microsoft.com/en-us/library/k1a63xkz.aspx), [-](http://msdn.microsoft.com/en-us/library/wch5w409.aspx), [\*](http://msdn.microsoft.com/en-us/library/z19tbbca.aspx), [/](http://msdn.microsoft.com/en-us/library/3b1ff23f.aspx)) can produce results that are outside the range of possible values for the numeric type involved. You should refer to the section on a particular operator for details, but in general:

* Integer arithmetic overflow either throws an [OverflowException](http://msdn.microsoft.com/en-us/library/system.overflowexception.aspx) or discards the most significant bits of the result. Integer division by zero always throws a **DivideByZeroException**.
* Floating-point arithmetic overflow or division by zero never throws an exception, because floating-point types are based on IEEE 754 and so have provisions for representing infinity and NaN (Not a Number).
* [Decimal](http://msdn.microsoft.com/en-us/library/364x0z75.aspx) arithmetic overflow always throws an [OverflowException](http://msdn.microsoft.com/en-us/library/system.overflowexception.aspx). Decimal division by zero always throws a [DivideByZeroException](http://msdn.microsoft.com/en-us/library/system.dividebyzeroexception.aspx).

When integer overflow occurs, what happens depends on the execution context, which can be [checked or unchecked](http://msdn.microsoft.com/en-us/library/khy08726.aspx). In a checked context, an [OverflowException](http://msdn.microsoft.com/en-us/library/system.overflowexception.aspx) is thrown. In an unchecked context, the most significant bits of the result are discarded and execution continues. Thus, C# gives you the choice of handling or ignoring overflow.

In addition to the arithmetic operators, integral-type to integral-type casts can cause overflow, for example, casting a [long](http://msdn.microsoft.com/en-us/library/ctetwysk.aspx) to an [int](http://msdn.microsoft.com/en-us/library/5kzh1b5w.aspx), and are subject to checked or unchecked execution. However, bitwise operators and shift operators never cause overflow.