**运算符（C# 编程指南）**

在 C# 中，运算符是应用于表达式或语句中的一个或多个操作数的程序元素。 接受一个操作数的运算符称为*一元*运算符，例如递增运算符 (**++**) 或 **new**。 接受两个操作数的运算符称为*二元*运算符，例如算术运算符（**+**、**-**、**\***、**/**）。 条件运算符 **?:** 接受三个操作数，是 C# 中唯一的三元运算符。

下面的 C# 语句包含一个一元运算符和一个操作数。 递增运算符 **++** 修改操作数 y 的值。

y++;

下面的 C# 语句包含两个二元运算符，它们分别有两个操作数。 赋值运算符 **=** 将一个整数变量 y 和一个表达式 2 + 3 作为操作数。 表达式 2 + 3 本身由加法运算符和两个操作数 2 和 3 组成。

y = 2 + 3;

**运算符、计算和运算符优先级**

操作数可以是由任何长度的代码组成的有效表达式，且可包含任意数量的子表达式。 在包含多个运算符的表达式中，运算符的应用顺序由运算符优先级、关联性和括号确定。

每个运算符都具有已定义的优先级。 在包含具有不同优先级级别的多个运算符的表达式中，运算符的优先级确定运算符的计算顺序。 例如，下列语句将 3 赋给 n1。

n1 = 11 - 2 \* 4;

因为乘法的优先级高于减法，所以首先执行乘法。

下表根据运算符执行的操作类型将它们划分到不同的类别中。 类别按优先级顺序列出。

主要运算符

| **Expression** | **描述** |
| --- | --- |
| x[.](https://msdn.microsoft.com/zh-CN/library/6zhxzbds.aspx)yx?.y | 成员访问条件成员访问 |
| f[(x)](https://msdn.microsoft.com/zh-CN/library/0z4503sa.aspx) | 方法和委托调用 |
| a[[x]](https://msdn.microsoft.com/zh-CN/library/a3hd7ste.aspx)a?[x] | 数组和索引器访问条件数组和索引器访问 |
| x[++](https://msdn.microsoft.com/zh-CN/library/36x43w8w.aspx) | 后递增 |
| x[--](https://msdn.microsoft.com/zh-CN/library/wc3z3k8c.aspx) | 后递减 |
| [new](https://msdn.microsoft.com/zh-CN/library/fa0ab757.aspx) T(...) | 对象和委托创建 |
| **new** T(...){...} | 具有初始值设定项的对象创建。 请参阅 [对象和集合初始值设定项（C# 编程指南）](https://msdn.microsoft.com/zh-CN/library/bb384062.aspx)。 |
| **new** {...} | 匿名对象初始值设定项。 请参阅 [匿名类型（C# 编程指南）](https://msdn.microsoft.com/zh-CN/library/bb397696.aspx)。 |
| **new** T[...] | 数组创建。 请参阅 [数组（C# 编程指南）](https://msdn.microsoft.com/zh-CN/library/9b9dty7d.aspx)。 |
| [typeof](https://msdn.microsoft.com/zh-CN/library/58918ffs.aspx)(T) | 获取 T 的 System.Type 对象 |
| [checked](https://msdn.microsoft.com/zh-CN/library/74b4xzyw.aspx)(x) | 在已检查的上下文中计算表达式 |
| [unchecked](https://msdn.microsoft.com/zh-CN/library/a569z7k8.aspx)(x) | 在未检查的上下文中计算表达式 |
| [default](https://msdn.microsoft.com/zh-CN/library/25tdedf5.aspx) (T) | 获取类型 T 的默认值 |
| [delegate](https://msdn.microsoft.com/zh-CN/library/900fyy8e.aspx) {} | 匿名函数（匿名方法） |

一元运算符

| **Expression** | **描述** |
| --- | --- |
| [+](https://msdn.microsoft.com/zh-CN/library/k1a63xkz.aspx)x | 标识 |
| [-](https://msdn.microsoft.com/zh-CN/library/wch5w409.aspx)x | 求反 |
| [!](https://msdn.microsoft.com/zh-CN/library/f2kd6eb2.aspx)x | 逻辑求反 |
| [~](https://msdn.microsoft.com/zh-CN/library/d2bd4x66.aspx)x | 按位求反 |
| [++](https://msdn.microsoft.com/zh-CN/library/36x43w8w.aspx)x | 前递增 |
| [--](https://msdn.microsoft.com/zh-CN/library/wc3z3k8c.aspx)x | 前递减 |
| [(T)](https://msdn.microsoft.com/zh-CN/library/0z4503sa.aspx)x | 将 x 显式转换为类型 T |

乘法运算符

| **Expression** | **描述** |
| --- | --- |
| [\*](https://msdn.microsoft.com/zh-CN/library/z19tbbca.aspx) | 乘法 |
| [/](https://msdn.microsoft.com/zh-CN/library/3b1ff23f.aspx) | 除号 |
| [%](https://msdn.microsoft.com/zh-CN/library/0w4e0fzs.aspx) | 余数 |

相加运算符

| **Expression** | **描述** |
| --- | --- |
| x [+](https://msdn.microsoft.com/zh-CN/library/k1a63xkz.aspx) y | 相加、字符串串联、委托组合 |
| x [-](https://msdn.microsoft.com/zh-CN/library/wch5w409.aspx) y | 相减、委托移除 |

移位运算符

| **Expression** | **描述** |
| --- | --- |
| x [<<](https://msdn.microsoft.com/zh-CN/library/a1sway8w.aspx) y | 左移 |
| x [>>](https://msdn.microsoft.com/zh-CN/library/xt18et0d.aspx) y | 右移 |

关系和类型运算符

| **Expression** | **描述** |
| --- | --- |
| x [<](https://msdn.microsoft.com/zh-CN/library/z5wecxwa.aspx) y | 小于 |
| x [>](https://msdn.microsoft.com/zh-CN/library/yxk8751b.aspx) y | 大于 |
| x [<=](https://msdn.microsoft.com/zh-CN/library/hx063734.aspx) y | 小于或等于 |
| x [>=](https://msdn.microsoft.com/zh-CN/library/a59bsyk4.aspx) y | 大于或等于 |
| x [is](https://msdn.microsoft.com/zh-CN/library/scekt9xw.aspx) T | 如果 x 为 T，则返回 True；否则返回 False。 |
| x [as](https://msdn.microsoft.com/zh-CN/library/cscsdfbt.aspx) T | 返回类型为 T 的 x，如果 x 不是 T，则返回 null |

相等运算符

| **Expression** | **描述** |
| --- | --- |
| x [==](https://msdn.microsoft.com/zh-CN/library/53k8ybth.aspx) y | 等于 |
| x [!=](https://msdn.microsoft.com/zh-CN/library/3tz250sf.aspx) y | 不等于 |

逻辑、条件和 null 运算符

| **类别** | **Expression** | **描述** |
| --- | --- | --- |
| 逻辑“与” | x [&](https://msdn.microsoft.com/zh-CN/library/sbf85k1c.aspx) y | 整型按位“与”，布尔型逻辑“与” |
| 逻辑 XOR | x [^](https://msdn.microsoft.com/zh-CN/library/zkacc7k1.aspx) y | 整型按位 XOR，布尔型逻辑 XOR |
| 逻辑“或” | x [|](https://msdn.microsoft.com/zh-CN/library/kxszd0kx.aspx) y | 整型按位“或”，布尔型逻辑“或” |
| 条件“与” | x [&&](https://msdn.microsoft.com/zh-CN/library/2a723cdk.aspx) y | 仅当 x 为 True 时计算 y |
| 条件“或” | x [||](https://msdn.microsoft.com/zh-CN/library/6373h346.aspx) y | 仅当 x 为 False 时计算 y |
| null 合并 | x [??](https://msdn.microsoft.com/zh-CN/library/ms173224.aspx) y | 如果 x 为 Null，则计算结果为 y，否则计算结果为 x |
| 条件运算 | x [?](https://msdn.microsoft.com/zh-CN/library/ty67wk28.aspx) y : z | 如果 x 为 True，则计算结果为 y；如果 x 为 False 则计算结果为 z |

赋值和匿名运算符

| **Expression** | **描述** |
| --- | --- |
| [=](https://msdn.microsoft.com/zh-CN/library/sbkb459w.aspx) | 赋值 |
| x op= y | 复合赋值。 支持以下运算符：[+=](https://msdn.microsoft.com/zh-CN/library/sa7629ew.aspx)、[-=](https://msdn.microsoft.com/zh-CN/library/2y9zhhx1.aspx)、[\*=](https://msdn.microsoft.com/zh-CN/library/s2bkaksf.aspx)、[/=](https://msdn.microsoft.com/zh-CN/library/d31sybc9.aspx)、[%=](https://msdn.microsoft.com/zh-CN/library/ydwa9zh0.aspx)、[&=](https://msdn.microsoft.com/zh-CN/library/e669ax02.aspx)、[|=](https://msdn.microsoft.com/zh-CN/library/h5f1zzaw.aspx)、[!=](https://msdn.microsoft.com/zh-CN/library/3tz250sf.aspx)、[<<=](https://msdn.microsoft.com/zh-CN/library/ayt2kcfb.aspx), [>>=](https://msdn.microsoft.com/zh-CN/library/23as4533.aspx) |
| (T x) [=>](https://msdn.microsoft.com/zh-CN/library/bb311046.aspx) y | 匿名函数（lambda 表达式） |

**结合性**

当表达式中出现两个或两个以上具有相同优先级的运算符时，将根据结合性计算它们。 左结合运算符按从左到右的顺序计算。 例如，x *y / z 将计算为 (x*y) / z。 右结合运算符按从右到左的顺序计算。 例如，赋值运算符是右关联的。 如果不是，下面的代码将导致错误。

int a, b, c; c = 1; // The following two lines are equivalent. a = b = c; a = (b = c); // The following line, which forces left associativity, causes an error. //(a = b) = c;

再举一个例子，三元运算符 ([?:](https://msdn.microsoft.com/zh-CN/library/ty67wk28.aspx)) 是右结合运算符。 大多数的二元运算符是左结合运算符。

无论表达式中的运算符是左结合运算符还是右结合运算符，都将先从左至右评估各表达式的操作数。 以下示例显示运算符和操作数的计算顺序。

| **语句** | **计算顺序** |
| --- | --- |
| a = b | a、b、= |
| a = b + c | a、b、c、+、= |
| a = b + c \* d | a、b、c、d、\*、+、= |
| a = b \* c + d | a、b、c、\*、d、+、= |
| a = b - c + d | a、b、c、-、d、+、= |
| a += b -= c | a、b、c、-=、+= |

**添加括号**

可通过使用圆括号更改运算符优先级和相关性。 例如，2 + 3 *2 通常计算结果为 8，因为乘法运算符的优先级高于加法运算符。 但是，如果你将表达式编写为 (2 + 3)*2，则先计算加法，再计算乘法，且结果为 10。 以下示例显示括号表达式中的计算顺序。 如前面的示例中所示，计算操作数之前会应用运算符。

| **语句** | **计算顺序** |
| --- | --- |
| a = (b + c) \* d | a、b、c、+、d、\*、= |
| a = b - (c + d) | a、b、c、d、+、-、= |
| a = (b + c) \* (d - e) | a、b、c、+、d、e、-、\*、= |

**运算符重载**

对于自定义类和结构，你可以更改运算符的行为。 此过程称为“运算符重载”。 有关更多信息，请参见[可重载运算符（C# 编程指南）](https://msdn.microsoft.com/zh-CN/library/8edha89s.aspx)。