运算符重载的方法必须为public和static（C#完全手册.pdf说的有点问题）。

运算符参数不能使用ref或out修饰符。

在C#中，重载操作符不应该改变操作数对象，而应该返回一个新的对象。

C#中的运算符重载与C++有比较大的相似之处，而区别有：

1. 不能重载赋值运算符=，这是与C++比较大的区别；
2. 不能重载运算符()。
3. 不能重载运算符[]
4. 不能重载运算符->
5. 重载了==就必须重载!=

在C++中实现类的运算符重载时，二元运算符只有一个参数，另一个是调用它的对象本身，而一元运算符没有参数。

而在C#中，二元运算符有两个参数，其中一个是调用它的对象本身，而一元运算符有一个参数，就调用它的对象本身。

重载自增运算符++

参考：<http://www.cnblogs.com/instance/archive/2011/05/21/2052722.html>

C#自增自减运算符的重载与其他如+，-等运算符的重载不太相同。

例：程序operator\_test1

// Copyright 2016.刘珅珅

// author：刘珅珅

// 运算符重载

using *System*;

using *System*.*Collections*.*Generic*;

using *System*.*Linq*;

using *System*.*Text*;

using *System*.*Threading*.*Tasks*;

namespace operator\_test1

{

class DoubleD

{

int x;

int y;

public DoubleD() { x = y = 0; }

public DoubleD(int i, int j) { x = i; y = j; }

// 在C#中，前缀++和后缀++的重载相同

public static DoubleD operator ++(DoubleD op)

{

DoubleD result = new DoubleD();

result.x = op.x + 1;

result.y = op.y + 1;

return result;

}

public void Show()

{

*Console*.*WriteLine*(x + ", " + y);

}

}

class OperatorTest

{

static void Main(string[] args)

{

DoubleD a = new DoubleD(1, 2);

DoubleD resultA;

// 执行后缀++重载运算符后，仍以原始值调用Show()

a++.Show(); // 1, 2

// 执行后缀++重载运算符后，会将结果本身赋给调用对象

a.Show(); // 2, 3

// 执行前缀++重载运算符后，直接将结果返回并修改对象本身

resultA = ++a;

resultA.Show(); // 3, 4

a.Show(); // 3, 4

// 执行后缀++重载运算符后，resultA接收的是修改之前的结果

resultA = a++;

resultA.Show(); // 3, 4

a.Show(); // 4, 5

}

}

}