ref与out之间的区别整理 - - 博客频道 - CSDN.NET

ref和out都是C#中的关键字，所实现的功能也差不多，都是指定一个参数按照引用传递。

对于编译后的程序而言，它们之间没有任何区别，也就是说它们只有语法区别。

总结起来，他们有如下语法区别：

1、ref传进去的参数必须在调用前初始化，out不必，即：int i;SomeMethod(ref i);//语法错误 SomeMethod(out i);//通过

2、ref传进去的参数在函数内部可以直接使用，而out不可：public void SomeMethod(ref int i){　int j=i;//通过　//...}public void SomeMethod(out int i){　int j=i;//语法错误

}

3、ref传进去的参数在函数内部可以不被修改，但out必须在离开函数体前进行赋值。

ref在参数传递之前必须初始化；而out则在传递前不必初始化，且在...值类型与引用类型之间的转换过程称为装箱与拆箱。

总结：应该说，系统对ref的限制是更少一些的。out虽然不要求在调用前一定要初始化，但是其值在函数内部是不可见的，也就是不能使用通过out传进来的值，并且一定要在函数内赋一个值。或者说函数承担初始化这个变量的责任。

下面谈谈ref和out到底有什么区别：

1 关于重载

原则：有out|ref关键字的方法可以与无out和ref关键字的方法构成重载；但如想在out和ref间重载，编译器将提示：不能定义仅在ref和out的上的方法重载

2 关于调用前初始值

原则：ref作为参数的函数在调用前，实参必须赋初始值。否则编译器将提示：使用了未赋值的局部变量；

out作为参数的函数在调用前，实参可以不赋初始值。

3 关于在函数内，引入的参数初始值问题

原则：在被调用函数内，out引入的参数在返回前至少赋值一次，否则编译器将提示：使用了未赋值的out参数;

在被调用函数内，ref引入的参数在返回前不必为其赋初值。

总结：C#中的ref和out提供了值类型按引用进行传递的解决方案，当然引用类型也可以用ref和out修饰,但这样已经失去了意义。因为引用数据类型本来就是传递的引用本身而非值的拷贝。ref和out关键字将告诉编译器，现在传递的是参数的地址而不是参数本身，这和引用类型默认的传递方式是一样的。同时，编译器不允许out和ref之间构成重载，又充分说明out和ref的区别仅是编译器角度的，他们生成的IL代码是一样的。有人或许疑问，和我刚开始学习的时候一样的疑惑：值类型在托管堆中不会分配内存，为什么可以按地址进行传递呢？值类型虽然活在线程的堆栈中，它本身代表的就是数据本身(而区别于引用数据类型本身不代表数据而是指向一个内存引用),但是值类型也有它自己的地址，即指针，现在用ref和out修饰后，传递的就是这个指针，所以可以实现修改后a，b的值真正的交换。这就是ref和out给我们带来的好处。

首先：两者都是按地址传递的，使用后都将改变原来参数的数值。

其次：rel可以把参数的数值传递进函数，但是out是要把参数清空，就是说你无法把一个数值从out传递进去的，out进去后，参数的数值为空，所以你必须初始化一次。这个就是两个的区别，或者说就像有的网友说的，rel是有进有出，out是只出不进。

ref（C# 参考）

ref 关键字使参数按引用传递。其效果是，当控制权传递回调用方法时，在方法中对参数的任何更改都将反映在该变量中。若要使用 ref参数，则方法定义和调用方法都必须显式使用 ref 关键字。

例如：

```
class RefExample
```

```
{
    static void Method(ref int i)
    {
        i = 44;
    }
    static void Main()
    {
        int val = 0;
        Method(ref val);
        // val is now 44
    }
}
```

传递到 ref 参数的参数必须最先初始化。这与 out不同，后者的参数在传递之前不需要显式初始化。

尽管 ref 和 out在运行时的处理方式不同，但在编译时的处理方式相同。因此，如果一个方法采用 ref 参数，而另一个方法采用 out 参数，则无法重载这两个方法。例如，从编译的角度来看，以下代码中的两个方法是完全相同的，因此将不会编译以下代码：

```
class CS0663_Example
{
    // Compiler error CS0663: "cannot define overloaded
    // methods that differ only on ref and out".
    public void SampleMethod(ref int i) { }
    public void SampleMethod(out int i) { }
}
```

但是，如果一个方法采用 ref 或 out 参数，而另一个方法不采用这两个参数，则可以进行重载，如下例所示：

```
class RefOutOverloadExample
{
    public void SampleMethod(int i) { }
    public void SampleMethod(ref int i) { }
}
```

out（C# 参考）

out 关键字会导致参数通过引用来传递。这与 ref 关键字类似，不同之处在于 ref 要求变量必须在传递之前进行初始化。若要使用 out参数，方法定义和调用方法都必须显式使用 out 关键字。

例如：

```
class OutExample
{
    static void Method(out int i)
    {
        i = 44;
    }
    static void Main()
    {
        int value;
```

```
        Method(out value);
        // value is now 44
    }
}
```

尽管作为 out 参数传递的变量不必在传递之前进行初始化，但需要调用方法以便在方法返回之前赋值。

ref 和 out 关键字在运行时的处理方式不同，但在编译时的处理方式相同。因此，如果一个方法采用 ref参数，而另一个方法采用 out 参数，则无法重载这两个方法。例如，从编译的角度来看，以下代码中的两个方法是完全相同的，因此将不会编译以下代码：

```
class CS0663_Example
{
    // Compiler error CS0663: "Cannot define overloaded
    // methods that differ only on ref and out".
    public void SampleMethod(out int i) { }
    public void SampleMethod(ref int i) { }
}
```

但是，如果一个方法采用 ref 或 out 参数，而另一个方法不采用这两类参数，则可以进行重载，如下所示：

```
class RefOutOverloadExample
{
    public void SampleMethod(int i) { }
    public void SampleMethod(out int i) { }
}
```