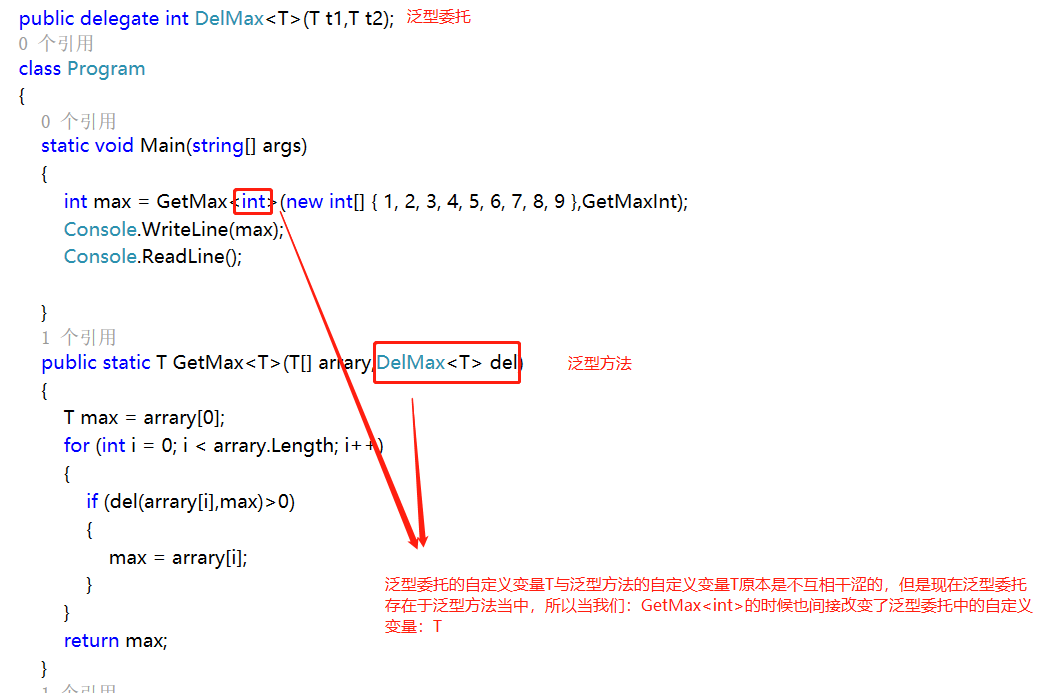
**泛型**

**泛型的详解：**

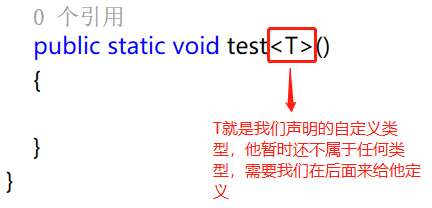
1. 泛型，为自定义类型，我们除了能在泛型集合能使用到自定义类型（泛型），还能在**泛型类**、**泛型方法**、**泛型接口**、**泛型事件**和**泛型委托**上能使用到泛型
2. 泛型自定义类型声明后，他暂时还不属于任何类型，也就是无法在声明泛型的类、方法、接口、事件或委托上对他进行与其他类型，如：int之间的运算、赋值等操作，除非我们在主函数中给我们所声明的泛型自定义类型赋值其他类型，让他变成我们所赋值的的类型，可以理解为一个**延迟声明的类型**
3. 如果我们在泛型方法、接口、类等都申明了一个泛型T，那么当我调用某一个泛型成员，并对该泛型成员的自定义变量：T进行初始化，那么不会影响到其他泛型成员的自定义变量：T，除非该成员存在于我们已经初始化的泛型成员之中，**如下图所示**，该泛型方法与泛型委托都存在自定义变量T，并且泛型委托存在于泛型方法中，所以我们在初始化泛型方法的自定义变量的时候也改变了泛型委托的自定义变量，但是如果我们只改变了泛型委托的自定义成员T，那么泛型方法的自定义成员T不受影响，因为泛型方法并不在泛型委托中：



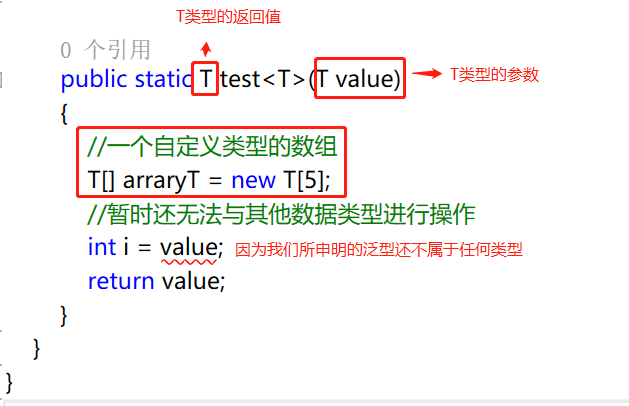
**泛型的语法和注意：**以下用**泛型方法**来说明**语法**和**注意事项**

1. 泛型的声明，需要在类、方法、接口、事件或委托的名字前加上：<**T**>

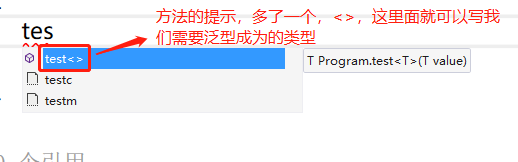
**T**就代表了是我们的自定义类型



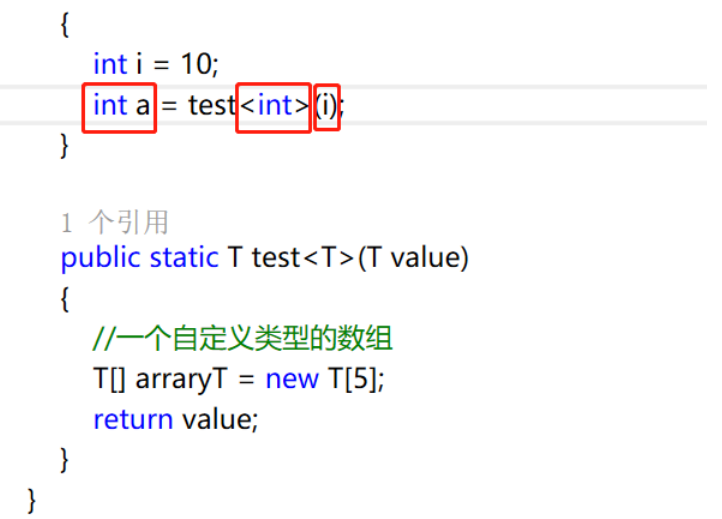
1. 泛型的声明成功后（<T>），我们就能够在该方法使用T这个泛型，但是我们还不能对他进行与其他类型之间的运算操作，因为这是个延迟声明的数据类型，暂时还不属于任何类型



1. 现在我们申明了一个泛型的方法，当我们在主函数中调用该方法的时候会发现：



1. 我们在，<>，上写我们需要泛型T需要变成的类型，如：int、string等，当我们写入后，所有在该方法里面存在的T类型都会变成我们所写的类型，如下图，该方法原本的参数和返回值都是T，现在变成了我们所写的，<int>，类型



**泛型委托配合泛型方法的使用：**

****