#csharp是完全面向对象的语言，所有方法都应该写在class中。其实csharp和java很相似，很多知识可以参考java。如数据类型的转换和数组相关内容。

# cs常用快捷键

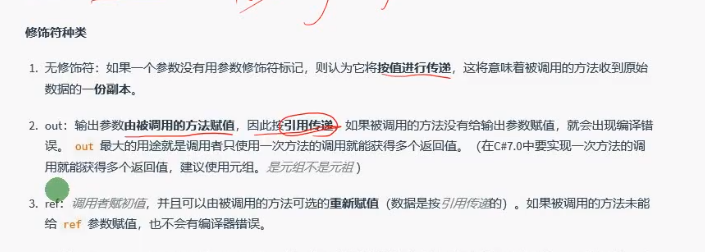


#数据类型：decimal（十进制）

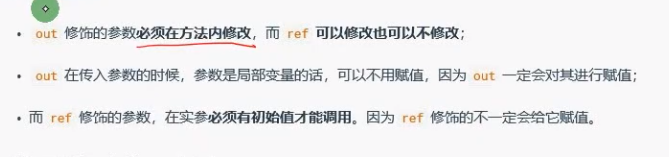
和double一样是浮点数，但是范围比double小、精度比double高。

声明格式：decimal de=1.1M。不加m默认为double，而double和decimal是不能互相转换的，编译会报错

#函数参数的修饰符：out ref



Out：方法中必须给这个参数赋值（否则编译报错），且是引用传递，caller里面的对应参数的值确实改变了。

Ref：方法中可以给参数赋值也可以不赋值，如果赋值了，按引用传递。

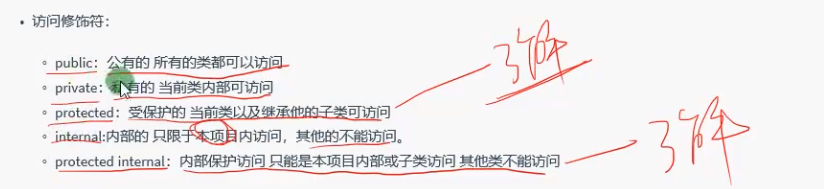
# csharp中声明类

不同的一点是变量需要声明属性。



声明属性快捷键：输入prop 按两下tab，自动补全

# 访问权限（protceted在继承中才会用到；private本类内才可以访问，继承也继承不下去，protected虽然外界无法访问，但可以继承下去）

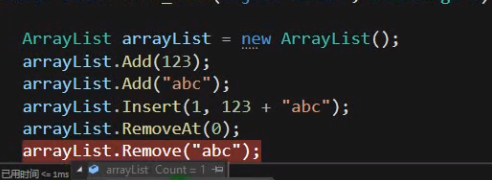


# Arraylist

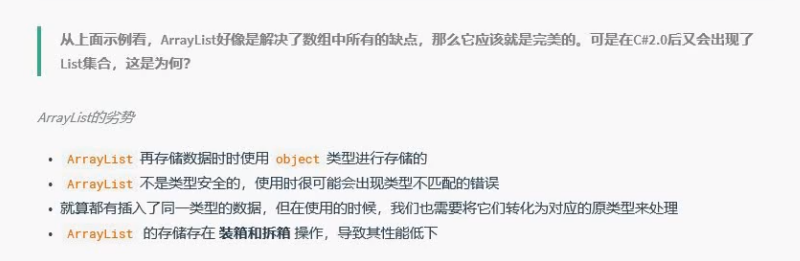
和java中的arraylist基本相同，但不需要规定泛型，一个list中的数据可以不是同一类型



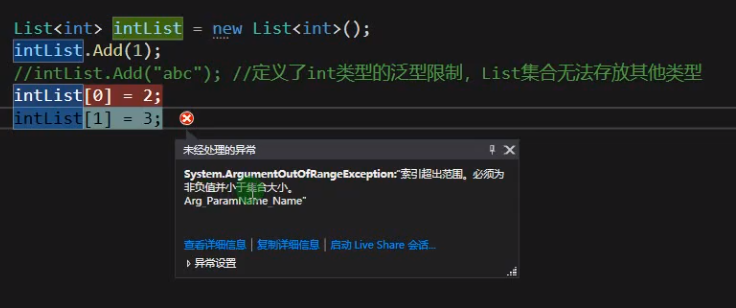
常用方法：



# List集合

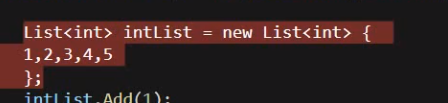


Csharp List等同于java的ArrayList，需要规定泛型



已经add进去的，可以像数组一样直接访问；常用方法和ArrayList相同

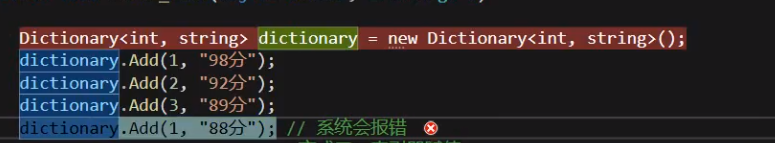
也可以像数组一样初始化：



再add就加到list末尾

# Dictionary字典类

类似c++中的map，是一组键值对。

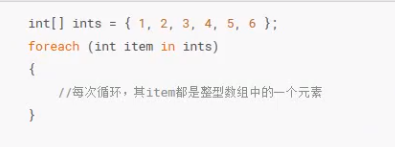


获取值：

删除：.remove

# foreach循环

Foreach循环数组：



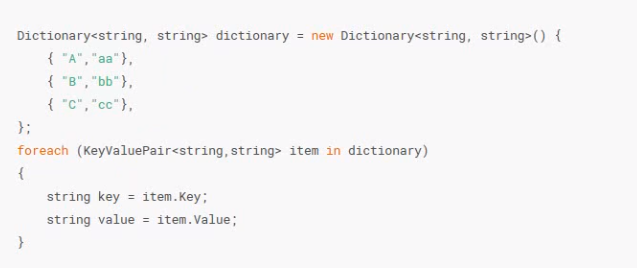
Item类型由数组类型决定

Foreach循环集合：



Item类型由泛型决定

Foreach循环字典：



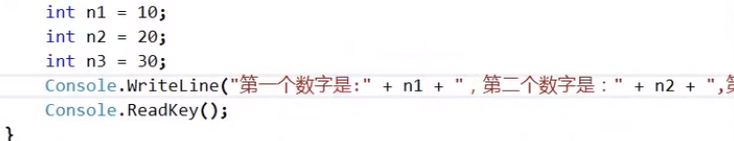
Item类型是KeyValuePair

# console.readkey()

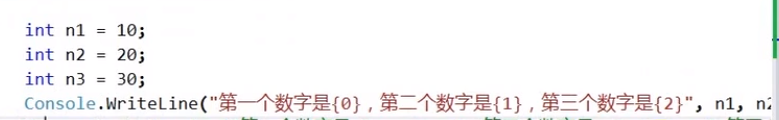
Cmd程序加上这一句，实现程序暂停（也就是按任意键返回），否则cmd一闪而过就没了。

#输入输出、占位符

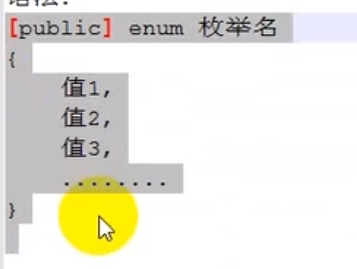
C#在输出时可用占位符：

常规的输出方法：

使用占位符：

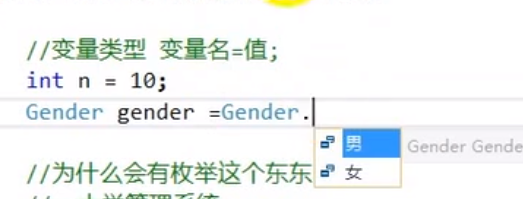
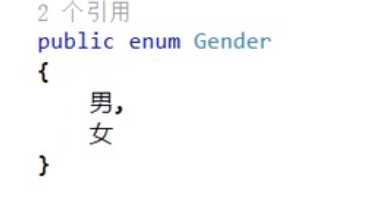


#枚举与结构



枚举不能在main函数中声明。要写在namespace空间下边、类名外边。

Enum的本质是个变量类型，用来做容器存数据的。



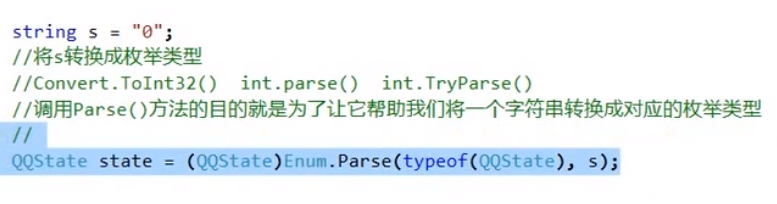
用的时候要枚举名.

枚举与int/string类型的转换：

Int与enum互转：枚举类型可以与int转换，默认从0开始（直接强转就行）

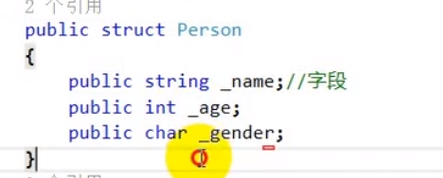
Enum转string：首先明确，任何类型都可以转换成string，点tostring()就可以；

如果直接输出enum中的某一项，输出的是索引值；而你把它点tostring（）再输出才是你起的名字。

String转enum：

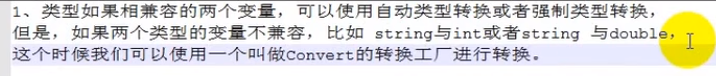
Enum类中的parse（）需要两个参数，第一个是枚举类型，用typeof（）获取，第二个是字符串名字。转换结果是一个通用enum，还要再强转成你写的enum。你把s改成OnLine也可以（QQState中的一个状态）

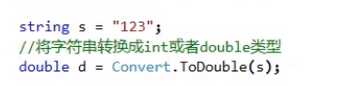
结构：和类差不多，就是没有方法



结构体内的名字，一般要加下划线！规范！

#类型转换



使用convert：

#类的继承与多态

Csharp只能单一继承,一个类只能有一个基类（想要继承多个类需要使用接口）。

Csharp中有sealed关键字，和java中的final一样，被该关键字修饰的类无法再被继承。

两个指针：base指针和this指针（对应java中的super和this，用于访问基类成员/基类构造函数）

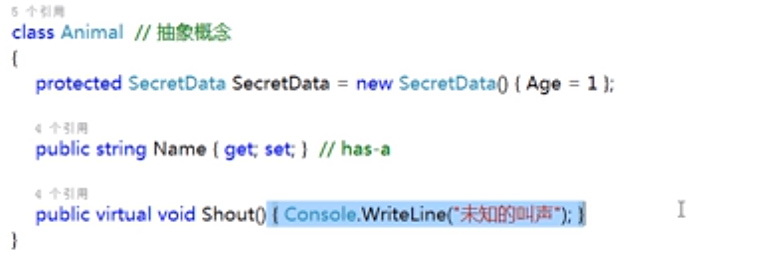
Vitural和override是对称使用的，可以实现多态：



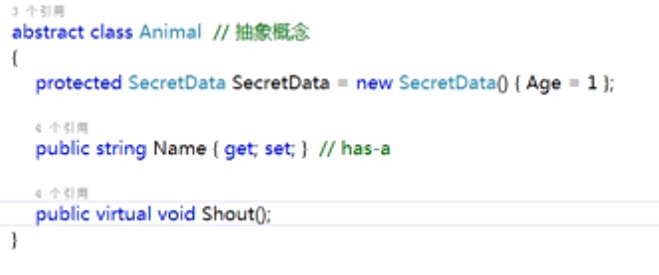
Test（）：调用shout（）

不用vitural的话，只会调用Animal类中的shout()，而无法调用Dog类中的shout()

#抽象类：

比如这就是一个抽象类，它本身无法具象化，而是作为一些具象（dog cat等）的基类而存在。

所以我们直接将其声明为抽象类：



需要注意，抽象类是无法直接实例化的，new不出来。

抽象类中也不需要提供默认方法。（animal只是一个抽象概念，你无法说这个动物会怎么叫，必须要继承到某一个具体的动物上才知道怎么叫）