****类型说明****

**C#认可的基础数据类型并没有内置于C#语言中，而是内置于.net Framework中。**

如，在c#中声明一个int类型的数据时，声明的实际上是.net结构System.Int32的一个实例。这听起来似乎很深奥，但其意义深远：这表示在语法上，可以把所有的基础数据类型看作是支持某些方法的类。

     类型实际上仍存储为基本类型。基本类型在概念上用.Net结构表示，所以肯定没有性能损失。

# 值类型

## 预定义基本类型

13个

### 1.整形



### 2.浮点类型

float数据类型用于较小的浮点数，因为它要求的精度较低。

double数据类型比float数据类型大，提供的精度也大一倍(15位)。

### 3、decimal类型



### 4.bool(布尔)类型



### 5.char(字符)类型



char类型的字变量是用单引号括起来的。     如'A'

     如果把字符把在"A"(双引号)内，编译器会把它看作是字符串，从而产生错误。

## 复合类型

### 枚举类型

枚举类型（也称为枚举）为定义一组可以赋给变量的命名整数常量提供了一种有效的方法

默认情况下，枚举中每个元素的基础类型是 [int](https://msdn.microsoft.com/zh-cn/library/5kzh1b5w.aspx)。

准许使用的枚举类型有 **byte**、[sbyte](https://msdn.microsoft.com/zh-cn/library/d86he86x.aspx)、[short](https://msdn.microsoft.com/zh-cn/library/ybs77ex4.aspx)、[ushort](https://msdn.microsoft.com/zh-cn/library/cbf1574z.aspx)、[int](https://msdn.microsoft.com/zh-cn/library/5kzh1b5w.aspx)、[uint](https://msdn.microsoft.com/zh-cn/library/x0sksh43.aspx)、[long](https://msdn.microsoft.com/zh-cn/library/ctetwysk.aspx) 或 [ulong](https://msdn.microsoft.com/zh-cn/library/t98873t4.aspx)。

声明 定义属性和赋值操作

enum Days { Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday };

可以将任意值赋给枚举类型的枚举数列表中的元素，也可以使用计算值：

enum MachineState

{

PowerOff = 0,

Running = 5,

Sleeping = 10,

Hibernating = Sleeping + 5

}

Enum.GetName方法

Enum.GetNames方法

### 结构类型

[struct person](http://shop111432812.taobao.com/)

[{](http://shop111432812.taobao.com/)

[string m\_name; //姓名](http://shop111432812.taobao.com/)

[int m\_age; //年龄](http://shop111432812.taobao.com/)

[string m\_sex; //性别](http://shop111432812.taobao.com/)

[}](http://shop111432812.taobao.com/)

# 引用类型

## 预定义引用类型

### object类型和字符串类型



## 委托类型

[delegate](https://msdn.microsoft.com/zh-cn/library/900fyy8e.aspx) 是表示对具有特定参数列表和返回类型的方法的引用的类型

声明委托

这里委托声明的仅仅是一个方法的类型,比如返回值是什么,形参是什么;

public delegate int PerformCalculation(int x, int y);

可将任何可访问类或结构中与委托类型匹配的任何方法分配给委托。

将方法作为参数进行引用的能力使委托成为定义回调方法的理想选择。

传递委托;

这里将另一个实例的方法传递给委托赋值;

public static weituoFunc p1;

委托类型派生自 .NET Framework 中的 [Delegate](https://msdn.microsoft.com/zh-cn/library/system.delegate.aspx) 类。委托类型是[封装的](https://msdn.microsoft.com/zh-cn/library/88c54tsw.aspx)，它们不能派生出其他类，也不能从 [Delegate](https://msdn.microsoft.com/zh-cn/library/system.delegate.aspx) 派生出自定义类。

由于实例化的委托是一个对象，因此可以作为参数传递或分配给一个属性。这允许方法作为参数接受委托并在稍后调用委托。这被称为异步回调，是在长进程完成时通知调用方的常用方法

回调是不用再关心父类是谁,直接是用方法;减少了耦合;对象与对象之间的关联,最终是属性和方法的关联;

合并委托:

按传递顺序一一执行委托;

MethodClass obj = new MethodClass();

Del d1 = obj.Method1;

Del d2 = obj.Method2;

Del d3 = DelegateMethod;

//Both types of assignment are valid.

Del allMethodsDelegate = d1 + d2;

allMethodsDelegate += d3;

## 数组类型

## 接口类型

## 类类型

类类型是由字段数据（成员变量）和操作字段数据的成员（属性、方法、[构造函数](https://www.baidu.com/s?wd=%E6%9E%84%E9%80%A0%E5%87%BD%E6%95%B0&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBPWfYPycYnAD1uj7-PH6s0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En10LrHDzPHDz)、事件等）所构成的自定义类型。其中字段数据表示类实例（对象）的状态。

# 匿名类型

匿名类型是直接从[对象](https://msdn.microsoft.com/zh-cn/library/9kkx3h3c.aspx)派生的[类](https://msdn.microsoft.com/zh-cn/library/0b0thckt.aspx)类型，并且其无法强制转换为除[对象](https://msdn.microsoft.com/zh-cn/library/9kkx3h3c.aspx)外的任意类型。

匿名类型包含一个或多个公共只读属性。包含其他种类的类成员（如方法或事件）为无效。

# ?可以为null的类型

可以为 null 的类型是 [System.Nullable<T>](https://msdn.microsoft.com/zh-cn/library/b3h38hb0.aspx) 结构的实例。

可以为 null 的类型可以表示其基础值类型正常范围内的值，再加上一个 null 值。

如，Nullable<Int32> 读作“可以为 null 的 Int32”，可以将其赋值为 -2147483648 到 2147483647 之间的任意值，也可以将其赋值为 null 值。

## null关键字

**null** 关键字是表示不引用任何对象的 null 引用的文字值。 **null** 是引用类型变量的默认值。

## Nullable<T>语法表述

int? num = null;

Nullable<T> num = null;

无法创建基于引用类型的可以为 null 的类型。（引用类型已支持 null 值。）

 T 可以是包括 **struct** 在内的任何值类型；但不能是引用类型。

语法 T? 是 [Nullable<T>](https://msdn.microsoft.com/zh-cn/library/b3h38hb0.aspx) 的简写，此处的 T 为值类型。这两种形式可以互换。

## 可以为null的类型成员

将 [HasValue](https://msdn.microsoft.com/zh-cn/library/sksw8094.aspx) 和 [Value](https://msdn.microsoft.com/zh-cn/library/ydkbatt6.aspx) 只读属性用于测试是否为空和检索值，如下面的示例所示：if(x.HasValue) j = x.Value;

如果此变量包含值，则 HasValue 属性返回 true；或者如果是 null 则返回 false。

如果已赋值，则 Value 属性返回该值。否则，将引发 [System.InvalidOperationException](https://msdn.microsoft.com/zh-cn/library/system.invalidoperationexception.aspx)。

HasValue 的默认值为 false。 Value 属性没有默认值。

还可以将 == 和 != 操作数用于可为 null 的类型，如下面的示例所示：if (x != null) y = x;

使用 ?? 算符分配默认值，在将当前值为 null 的可以为 null 的类型赋值给不可以为 null 的类型时，将应用该默认值，如 int? x = null; int y = x ?? -1;

不允许使用嵌套的可以为 null 的类型。将不编译下面一行：Nullable<Nullable<int>> n;

# Void结构

system.Void

为不返回值的方法指定返回值类型。

继承与Object;所以默认值为null

# undefined

在运行时出现对未定义对象的引用;

在编辑和编译阶段 编辑工具和编译器不允许时候未定义的对象;

# 默认值



所有引用类型,默认值null;在声明了而没有初始化的时候出现;在运行时出现;

方法中的变量编辑器不允许不初始化而使用;