变量命名是小写词缀与大写词缀的区别：

小写（深蓝色）:关键字,c#独有

大写（浅蓝色）：类，.net平台通用

Decimal 适合用于声明一个财务和货币变量，变量后面加上m；

decimal money = 5000m;

交换两个变量的值（数字类型），不使用第三个变量：

Int a = 1；

Int b = 2；

a = a-b；

b = a+b；

a = b-a；

赋值：

1）camel骆驼命名规范：首个单词首字母小写，其余大写（highSchool），多用于给变量命名；

2）pascal命名规范：要求每个单词首字母大写，其余小写（HighSchool）多用于给类或方法命名；

+号可以连接两个字符串

占位符

使用方法：先挖个坑，在填个坑

Int n1 = 1;

Int n2 = 2；

Int n3 = 3;

Console.writeline(“第一个数字是{0}，第二个数字是{1}，第三个数字是{3}”,n1,n2,n3);

有点像printf里面的操作，挖坑首个是{0}，按照挖坑的顺序输出，如果顺序变成了{1}{0}{2}，则会以n2,n1,n3的顺序输出。

需要注意的是，挖了了几个坑，就得填几个坑，但是多填坑没效果，少填坑会异常；

异常（bug）：语法上并没有错误，只不过程序运行区间，由于某些原因，使得程序不能正常运行的现象

转义符

转义符指的就是一个“\”+一个特殊的字符，组成了一个特殊意义的字符；

\n 表示换行，如果要让window认识换行，得写\r\n,(新建一个文本时使用)

\” 表示一个英文半角的双引号

\t 表示一个tab键的空格，多用于排版

\b 表示退格键back，放在字符串两端没有效果

\\ 表示一个\

@ 取消\在字符串中的转义作用（在路径中常使用）、将字符串按照原格式输出

String path = @“F:\a\b\c\d\e\f.avi”;

Console.writeline(@”今天是个

好日子”)；（这是一个正确的表达式）

算数运算符

+、- 、\*、/、% 、++、--

给项目取名字时，如果取的不爽，想改，不要直接改名，需要删除，再重建，若非如此，面向对象时涉及到项目之间的调用会出问题

异常捕获

try\_catch try和catch之间不能有其他代码

显示转换数据

Convert.toint32();成功便成功，失败则异常

⬄Int.parse();,这两者的区别是，再执行convert时，程序会自己调用int.parse

Int.tryparse（string，out int）；使用这个函数时，失败不会报异常，成功会将前者转换完，放入后面的变量中

三元表达式

表达式1？表达式2：表达式3

产生随机数

1、创建能够产生随机数的对象，2、让产生随机数的这个对象调用方法来调用随机数

Random r = new Random（）；

int rNumber = r.next(1,10);左闭右开区间，表示随机数的范围

常量

声明常量时加上const，不能被重新赋值

枚举（规范开发，规定）

将枚举声明到命名空间的下面，类的上面，使得命名空间内都可以使用到这个枚举

声明枚举的基本语法

[public] enum 枚举名

{

值1，

值2，

…

}

Public：访问修饰符，公开的公共的，哪都可以访问

enum：关键字，声明枚举的关键字

枚举名：要符合pascal命名规范

声明完之后就可以命名变量：

Public enum Gender

{

男 = 2，

女

}

Gender gender = Gender.男；//赋值

枚举类型默认可以跟int类型转换（兼容）

当转换的枚举中没有对应的值时，不会异常，而是直接将数字显示出来

枚举类型默认从0开始，依次递增；

所有的类型都可以转换为string类型

将字符串转换为一个枚举类型，需要用到Enum.parse（typeof(枚举名)，需要转换的字符串）；

String s ;

Gender gender = （Gender）Enum.parse(typeof(Gender),s);如果转换不了（字符类型）则异常

结构（可以帮助我们一次性声明多个不同的变量）

[public] struct 结构名

{

成员；//字段，与变量不同，程序运行时，变量只能存一个值，字段可以多个，字段命名时前面需加上 下划线。

}

public struct Person //此处的public只能保证外部能够访问到这个结构体， 不能保证访问到内部成员

{

public string \_name;

public int \_age;

public Gender \_gender; //如果不加上public，出了这个声明结构体，就无法访问内部的变量

}

Person zsPerson;

zsPerson.\_name = “张三”；

zsPerson.\_age = “21”;

zsPerson.\_gender = Gender.男；

数组

int[] nums = new int[10];

如果你想访问到数组中的某一个元素，则需要通过元素的下标或者索引去访问

方法（函数）

函数就是将一堆代码进行重用的一种机制。函数就是一段代码，这段代码可能有输入的值（参数），可能有返回值。

函数的语法：

[public] static 返回值类型 函数(方法)名（[参数列表]）

{

方法体；

}

Public :访问修饰符，公开的

Static：静态的

返回值类型：如果不需要，写void

方法名：符合pascal命名规范，每个单词首字母大写，其余小写

参数列表：完成这个方法所需要提供的条件，如果没有参数，小括号也不能省略

Return：在方法中返回要返回的值，立即结束本次方法，（可以用于直接跳出函数体）

方法写好后，如果想要被执行，必须要在main函数中调用，方法的调用语法：

类名.方法名（[参数]）； 在某些情况下类名是可以省略的，如果你的函数与main函数在同一个类中，就可以省略

方法写好以后，给整个方法进行注释，使用///，这样在使用方法时可以方便看懂

我们在main函数中调用XX函数，那么main函数称为调用者，XX函数称为被调用者。

如果被调用者想要得到调用者的值，有两种方式：

1）传参

2）使用静态字段来模拟全局变量

静态字段：再类的括号中如此声明：public static int \_number = 10; 前面加上下划线称之为字段

如果调用者想要得到被调用者的值，使用返回值

数组排序

Array.Sort([数组名]); 对整个数组进行排序

Array.Reverse([数组名])；反转整个数组中的元素

Out参数

如果你在一个方法中，返回多个相同类型的值，可以使用返回数组，但是，如果返回多个类型的值的时候，返回数组就不行了，这个时候，可以使用out参数，out参数就侧重于在一个方法中可以返回多个不同类型的值。

这样，方法不需要返回值，在参数里面，除了要写上的调用者的参数，还要写上需要返回的类型的参数，如：

Public static 返回类型 函数名（形参，out int 返回值1，out string 返回值2，out char 返回值3）out表示出了函数本身的返回值所多余返回的值

{

方法体；在方法体内，必须要对out参数的值进行赋值

}

在调用这个函数时：函数名（实参，out 返回int类型值1‘，out 返回string类型值2’…）；

Ref参数

相当于c语言中的指针变量\*，有所不同的是，c#中在传入了ref参数后，在方法体中赋值不需要加上ref，但在C语言中要加\*号

要求在方法外必须先赋值，在方法内可以不赋值，作用是将变量值改变

Params可变参数

如果方法的参数是一个数组，但又不想在主函数中定义一个新数组，而是直接将数组的元素当成参数传给方法，那么在方法的数组参数前加上params即可实现这种效果。

将实参列表中跟可变参数数组类型一致的元素都当成数组的元素去处理，被params修饰的参数，必须是形参列表中的的最后一个元素，所以一个参数列表中只能出现一个params

方法的重载

方法的名称相同，但是参数不同（个数不同或者类型不同）

方法的重载跟返回值没有关系

方法的递归

方法自己调用自己，当想使用return退出函数时，不会立刻退出，而是调用了几次方法，就会return多少次

面向对象

找个对象帮你做事！

意在写出一个通用的代码，屏蔽差异

我们把这些具有相同属性和方法的对象进行进一步封装，抽象出类这个概念，类就是一个模子，确定了对象应该具有的属性和方法

对象是根据类创建出来的

（类就是一个盖大楼的图纸，而对象就是盖出来的楼）

类

语法：

[public] class 类名

{

字段；//field

属性；properties

方法；method

}

写好一个类之后，我们需要创建这个类的对象，那么，我们管创建这个类的对象过程称之为类的实例化，使用关键字new。

类是不占内存的，但是对象占内存

属性的作用就是保护字段，对字段的赋值和取值进行限定

属性的本质就是两个方法，一个叫get，一个叫set

**访问修饰符**

Public：公开的，公共的，在哪都能访问

Private：（不加其他修饰符，默认为private）私有的，只能在当前类的内部进行访问，除了这个类就不能访问

**快捷键**

Crtl+R+E：封装字段

Ctrl+K+D：自动排列代码

Ctrl+K+C：注释所选代码

**静态与非静态的区别**

1. 在非静态类中，既可以有实例成员，也可以有静态成员
2. 在调用实例成员时，需要使用 对象名.实例成员；

在调用静态成员的时候，需要使用 类名.静态成员名；

总结：

静态成员必须使用类名去调用，而实例成员使用对象名调用

静态函数中，只能访问静态成员，不允许访问实例成员

实例函数中，既可以使用静态成员，也可以使用实例成员

静态类当中，只允许出现静态成员

静态类无法实例化 实例对象，因为他的所有对象必须使用类名调用

使用：

1. 如果你的类之被当作“工具类”使用，可以考虑将类写成静态类
2. 静态类在整个项目中资源共享

只有在程序结束的时候，静态类才会释放资源

堆 栈 静态存储区域

释放资源：GC（garbage collection）垃圾回收

**构造函数**

作用：帮助我们初始化对象（给对象的每个属性依次的赋值）

构造函数是一个特殊的方法

1. 构造函数没有返回值，也不能写void
2. 构造函数的名称必须和类名一样 修饰符必须为public
3. 构造函数也可以重载
4. 类当中会有一个默认的无参数的构造函数，当你写了一个新的构造函数之后，原来类当中的默认构造函数就会被代替

如果修饰符是private，new就无法进入构造函数

创建对象时首先执行的是构造函数

**New关键字**

New帮我们做了三件事：

1. 在内存中开辟一块空间
2. 在开辟的空间中创建对象
3. 调用对象的构造函数进行初始化对象

**this 关键字**

1. 表示当前这个类的对象
2. 在类当中显式的调用本类的构造函数 “：this（）”

Public student（string name，int age , char gender）

{

This.Name = name;

This.Age= age;

This.Gender = gender;

}

Public student（string name，int age）:this(name,age,’a’) //调用时，参数要写全，如果不需要随便写一个默认值

{

}

**命名空间**

可以认为类是属于命名空间

如果没有当前项目中没有类的命名空间时

使用alt+shift+F10可以快速弹出需要的命名空间

如何在一个项目中引用另一个项目的类

1. 添加引用
2. 引用命名空间（项目名字）

一个解决方案中有多个项目，需要将某个项目设为启动项目，此时调试的就是当前的项目

值类型和引用类型

区别：

1. 在内存上存储的地方不一样
2. 在传递两种类型的时候，传递的方式不一样。

值类型称为值传递，引用类型称为引用传递

值类型：int double char bool decimal struct enum

引用类型：string 自定义类 数组

存储：

值类型的值是存储在内存的栈当中

引用类型的值是存储在内存的堆中

**字符串**

1. 字符串的不可变性，当你给一个字符串重新赋值后，老值并没有被销毁，而是重新在堆上开辟一块新空间，但是变量名在栈中没有改变，变量名存储的堆的地址发生改变

当程序结束时，GC扫描整个内存，如果发现有的空间没有被指向，则立即把它销毁

1. 我们可以将字符串看作是char类型的一个只读数组

ToCharArray（）；将字符串转化为char数组

new string（char[] chs）;能够将char数组转换为字符串

使用StringBuilder创建字符串并重新赋值时不会新开辟一块堆的空间，效率大大提高

任何类型都可以调用tostring（）转化为字符串类型

字符串的方法

Length：获取字符串的长度

ToUpper：将字符串转换成大写形式

ToLower：转换为小写

lessonOne.Equals（lessonTwo，StringComparison.OridinalIgnoreCase）:比较字符串并无视大小写

split（）；分割字符串，返回字符串类型数组

继承（可以解决类之间代码冗余的问题）

Public class Person{}

Public class Student : Person{}

Public class Teacher : Person{}

Public class Driver : Person{}

Person类称为其他类的父类（基类），它具备它的子类（派生类）的共有属性

子类能够访问到父类的东西就成为继承过来的东西（属性和方法）

子类不能继承私有字段、构造函数（但是子类会默认调用父类的无参的构造函数，用于创建父类对象，让子类能够使用父类的成员）

所以，如果在父类中创建了一个有参的构造函数之后，默认的无参构造函数就被干掉了，子类创建对象时就会报错，

解决办法

1. 在父类中再写一个无参的构造函数
2. 在子类中显式的调用父类的构造函数，使用关键字 :base（）,使子类对象能够调用父类的有参构造函数

继承的特性

1. 继承的单根性：一个子类只能有一个父类
2. 继承的传递性

查看类图

Object是所有类的基类

new关键字

1. 创建对象
2. 隐藏从父类那里继承过来的同名成员（后果是子类调用不到父类的成员）

里氏转换

1. 子类对象可以赋值给父类对象
2. 如果父类中装的是子类对象，那么可以将这个父类强转为子类对象

is：表示类型转换，如果能够转换成功，返回true，否则返回false

as：表示类型转换，如果能够成功则返回对应的对象，否则返回一个null

protected访问修饰符

子类的成员能够访问到父类被protected修饰的成员，但其他类不行

ArrayList集合

详见工程

每次集合中实际包含的元素个数（count）超过了可以包含的元素个数（capacity），集合就会向内存中申请多开辟一倍的空间。来保证集合的长度一直够用。

Hash table 键值对集合

在键值对集合当中，我们是根据键去找值的

键值对 对象[键] = 值；

键不能重复，值可以重复

编码方式：

ASC：125

ASCII：256

GB2312简体字

GBK既有简体也有繁体

Big5繁体字

Unicode包括大部分内容的编码，但解析起来慢

UTF-8 web

编码：将字符串以怎样的形式保存为二进制

乱码：产生乱码的原因：保存这个文件所采用的编码和打开这个文件所采用的编码格式不一样。

绝对路径与相对路径

绝对路径：通过给定的这个路径直接能在我的电脑中找到这个文件

相对路径：文件相对于应用程序的路径（将所要操作的文件放到和应用程序同一个目录下，操作该文件时直接使用文件名即可，）。

开发当中尽量使用相对路径

装箱和拆箱

装箱：将值类型转换为引用类型

拆箱：将引用类型转化为值类型

装箱和拆箱会占用时间

判断两种类型是否有装箱或者拆箱：看两种类型之间是否存在继承关系。

将创建文件流(FileStream)对象的过程写在using当中，会自动帮助我们释放流所占用的资源

实现多态的手段

1. 虚方法

步骤：

将父类的方法标记为虚方法，使用关键字virtual，这个函数可以被子类重新写一遍（子类方法加上关键字override）

1. 抽象类

当父类中的方法不知道如何去实现的时候，可以考虑将父类写成抽象类，将方法写成抽象方法

抽象成员必须在抽象类当中；

抽象类中可以存在非抽象成员，虽然无法创建抽象类的对象，但是可以通过继承给子类调用

当非抽象子类继承了一个抽象类之后，该子类必须将父类当中的抽象类重写

抽象类可以继承抽象类，此时子类的抽象类可以不重写父类当中的抽象成员

抽象类当中可以有虚方法。但没必要

1. 接口

C#中的访问修饰符

Public：公共的

Private：私有的，只能在当前类的内部访问

Protected：受保护的，只能在当前类和该类的子类中访问

Internal：只能在当前程序集（项目）中访问

Protected internal：protected+internal

1）、能够修饰类的修饰符只有两个：public和internal

2）、可访问性不一致。

子类的访问权限不能高于父类，或暴露父类的成员

设计模式

设计者给项目的一种方式。

简单工厂设计模式

值传递和引用传递

值类型在赋值时，传递的是这个值本身

引用类型在赋值的时候，传递的是这个对象的引用

序列化和反序列化

序列化：将对象转化为二进制

反序列化：将二进制转换为对象

作用：传输数据

序列化：

1）、将这个类标记为可以被序列化

Partial部分类

用于声明多个相同的类名，实际上是将一个类分为不同的部分

Sealed密封类

不能被继承，能继承其他类

接口

[public] interface 接口名（I----able）

{

成员；

}

自定义属性

[Browsable（true）]//是否将属性添加到用户属性列表中

//Description 指定属性或事件的说明

//Category 给自定义属性进行分组的名称（不添加则为杂项）

//DefaultValue 不加粗显示的值

[Description（“说明”），Category（“分组名称”），DefaultValue（“”）]

Public int PropertyName{get；set；}