Practice...

C#
编程思想与实践

Programming Series

张 青 郭亚萍 主编

冶金工业出版社

C#编程思想与实践

张　青　郭亚萍　主编

北　京

冶金工业出版社

内容简介

C#是 Microsoft 推出的新一代面向对象的语言，它功能强，易于表现，使用灵活。本书通过具体的实例，介绍了 C#的基础知识和编程方法。主要包括 C#基础、第一个 C#程序——hello C#、基本控件的使用、文本编辑、图形处理，以及多媒体编程、数据库编程、网络编程等内容。

本书内容翔实，实例丰富，注释详细，能使读者尽快掌握 C#的编程方法和技巧。本书不但适合于 C#的初学者，同时对中等水平的用户以及具有一定面向对象编程基础的读者也是一本很好的进阶教材。

图书在版编目（CIP）数据

C#编程思想与实践 / 张青等编著. —北京：冶金工业出版社，2002.8

ISBN 7-5024-3071-7

I. C… II. 张… III. C#语言—程序设计 IV. TP312

中国版本图书馆 CIP 数据核字（2002）第 055787 号

出版人 曹胜利（北京沙滩嵩祝院北巷 39 号，邮编 100009）

责任编辑 戈兰

中山市新华印刷厂有限公司印刷；冶金工业出版社发行；各地新华书店经销

2002 年 10 月第 1 版，2002 年 10 月第 1 次印刷

787mm × 1092mm 1/16； 27.5 印张； 640 千字； 432 页； 1-2600 册

48.00 元

冶金工业出版社发行部 电话：（010）64044283 **传真：**（010）64027893

冶金书店 地址：北京东四西大街 46 号（100711） **电话：**（010）65289081

（本社图书如有印装质量问题，本社发行部负责退换）

前　言

一、关于 C#

2000 年 7 月，Microsoft 向人们展示了它的新一代开发工具—— Visual Studio.NET。Microsoft 推出.NET 平台，可以支持 20 多种语言编写的程序，加强了语言的平台无关性，编写出来的程序的可移植性大大加强。

Microsoft 推出 Visual Studio.NET，目的是推动网络应用软件的开发，通过 Visual Studio.NET，程序员可以快速地开发各种应用程序。从 Windows 编程，到 XML Web 服务和应用程序；从简单的数据库到通过 Internet 进行数据共享，使用 Visual Studio.NET 都可以得心应手的开发出合适的工具。

作为 Visual Studio.NET 的核心语言—— C#，更是体现出 Visual Studio .NET 的特点。利用 C#可以极其方便地开发出强大的分布式应用系统。作为一种新生的语言，C#一出现就备受关注。Microsoft 推出 C#的目的就是综合 Visual Basic 的易用性和 C++的高效性。所以，使用 C#，可以很容易地开发出 Windows 程序、控制程序、C/S 模式程序、图形程序，或者是数据库程序和网络程序，甚至，可以用 C#编写组件和 Web 页面。

正是由于 C#的强大功能和易于使用，Microsoft 可以自豪地说“为未来的 10 年做好了准备”。C#毕竟是一种新生的语言，虽然它是从 C/C++衍生而来，但是仍然有很大的区别。如何快速掌握 C#，并能尽快使用 C#编写出功能强大的程序，这是很多读者拭目以待的事情，这也是我们编写这本书的原因——通过具体实例，介绍如何使用 C#快速、高效地编写出高质量的应用程序。

二、本书特点

本书通过具体的实例，由浅处着手，引导读者一步一步掌握用 C#编写应用程序的原理和方法。在编写程序的过程中，穿插讲解了 C#的类的使用，使读者能够一边看实例，一边学习 C#所提供的强大的类库，进行高效率的编程。为了帮助读者巩固所学的知识，每章的后面还有一定量的练习题，并在书后给出了参考答案，以供读者对照练习。所以，本书十分适合 C#的初学者，当然，如果读者有一定的面向对象的编程基础则更好。

本书从全方位多角度出发，从最基本的 Windows 编程入手，介绍了 C#在不同领域的应用，包括文本编辑、C#图形处理、Windows 应用程序编写、数据库编程、网络编程等方面。在编写的时候，尽量使每一部分的内容保持一定的独立，这样读者可以选择自己需要的章节去学习，而不会影响整体阅读。

本书所采用的例子不是很复杂，但是却是学习 C#很好的例子。每个实例都是编者实际经验的总结，较为全面的反映了 C#在各个方面的应用。所有实例都经过精心调试而成，并且有详细的注释，使读者能尽快掌握 C#的编程技术。

三、本书结构

全书共分为 8 章和一个附录，介绍了 C#的基础知识和编程方法。其结构安排如下：

第 1 章介绍了 C#的基础知识，主要包括 C#的数据类型、C#程序结构、C#的命名空间、

类、接口、反射和程序集以及 C#调试技术等内容。

第 2 章第一个 C#程序——hello C#，主要包括 hello C#与图形界面的 hello C#等内容。

第 3 章基本控件，主要包括 Windows Forms 模式、标注类控件、文本编辑类控件、按钮类控件、容器类控件等 6 类控件及其他主要控件的介绍。

第 4 章文本编辑，主要介绍利用 C#编写一个类似记事本的文本编辑器。内容包括加入文本框、加入菜单、加入上下文菜单、实现打开与保存文件，如何实现改变字体、颜色和打印文本等。

第 5 章 C#中的图形处理，主要包括 GDI+、System.Drawing 和 System.Drawing.Drawing 2D，并通过实例介绍了屏幕保护程序的设计。

第 6 章多媒体编程，主要包括图像浏览器，图像格式转换、音频/视频播放等内容。

第 7 章数据库编程，主要包括.NET 中的数据库编程基础和通讯录应用程序实例等内容。

第 8 章网络编程，主要包括网络编程概述、System.Net 和 System.Net.Socket 等内容，最后通过三个实例，介绍了 C#的网络编程。

在附录中介绍了 Visual Studio .NET 的安装方法。

四、适用对象

本书适用于 C#的初学者、中等水平的用户以及具有一定面向对象编程基础的读者。

本书的作者都具有丰富的编程经验，其中，本书第 1 章由朱润酥编写，第 2 章、第 4 章、第 5 章、第 6 章由陈继轩编写，第 3 章、第 8 章由陈鹏翼编写，第 7 章由刘宇恒编写。何耀彬参与了第 5 章的编写工作，陈梁参与了第 8 章的编写工作。陈继轩和陈鹏翼分别对本书进行了校对。

本书中的实例使用 Visual Studio C# 英文版编写，并且全部经过测试。读者可以在网上下载本书中的源代码，**网址：http://www.TianDing.net**。虽然作者在编写本书的过程中，对本书进行过反复的审核，但是由于水平有限，加上时间仓促，疏漏之处在所难免，希望读者不吝赐教。

编　者

2002 年 7 月

目　录

第 1 章　C#基础知识

本章是 C#的基础部分，先介绍 C#的产生背景、重要特性等，让读者对 C#这门语言有初步的认识，然后介绍 C#的基本数据类型，分别从数值类型和引用类型两大部分进行逐个介绍，然后再介绍 C#的装箱和拆箱操作。

接着，将学习 C#的运算符，然后讲解 C#的程序结构，包括循环、选择、转移三部分，使读者对程序设计的结构有一定的了解。

在接下来详细讲解 C#中的类和接口以及 C#中的重载，使读者能够进一步掌握 C#的基本语法。

最后，将会给大家介绍几个高级主题，包括 C#中的异常处理和非安全代码，反射和程序集以及 C#的调试技术。

相信读者阅读完本章之后，会掌握 C#的基本内容。为以后的学习打下坚实的基础。

1.1　C#的产生

1.1.1　C#的产生背景

在过去的 20 年内，C 和 C++已经是开发商用和企业软件时使用最广泛的编程语言之一。这两种语言为开发者提供了大量细致灵活的控制，然而这种灵活性是以生产的成本作为代价的。如果和其他的开发语言相比（比如说 VB），相同功能的 C/C++软件通常会需要更长的开发周期。正是由于 C/C++开发的复杂性和需要较长的开发周期，所以许多 C/C++开发人员都在寻找一种可以在功能和开发效率间提高更多平衡的开发语言。

目前，有一些开发语言通过牺牲 C/C++语言一些必要的灵活性来换取开发效率。有些语言对开发人员产生了过多的限制（比如说限制使用底层控制代码）并且提供更少的通用命名能力（比如说对变量，函数的引用能力）。这些语言不能够轻易的与现存的系统相结合，并且不能够和当前的 Web 开发相结合。

一种合理的 C/C++替代语言应该是能够提供对现存和潜在的平台上的高效开发提供有效和有力的支持。并可以使 Web 开发非常方便的与现存的应用开发相结合。而且 C/C++开发人员都倾向于在必要的时候使用底层代码。

1.1.2　C#的推出

由于以上背景上 Microsoft 的解决方案是推出一种命名为 C#（发音为 C Sharp）的开发语言。C#是一种先进的面向对象的语言，它功能强，易于表现，使用灵活。通过 C#可以让开发人员快速的建立大范围的基于微软新的 .NET 平台的应用，并且提供大量的开发工具和服务帮助开发人员开发基于计算和通信的各种应用。

由于其优良的面向对象设计，在构建从高级业务对象到系统级应用的各种不同组件时，

C#是一个首要的选择。使用简易的 C# 语言构造，组件可以被转换为 Web 服务，从而允许从运行在任何操作系统上的任何语言中跨越 Internet 调用它们。

不仅仅如此，C# 的设计为 C++程序员带来了快速的开发能力，而不用牺牲 C++已有的功能和控制能力。C# 高度地保持了与 C 和 C++ 的一致性。从继承角度来看，C#在更高层次上重新实现了 C/C++，熟悉 C/C++开发的人员可以很快的转变为 C#开发人员。

1.1.3 C#的重要特性

1. 开发效率与安全性

目前的各种基于 Web 应用的软件开发向传统的商业应用软件开发提出了挑战，开发者被组织起来开发具有更短开发周期的各种应用，并且需要能够提供更好的可修正性，而不是建立一个可以长久使用的软件系统。

C#的设计正是充分考虑了这些因素。C#会帮助开发者通过更少的代码完成相同的功能，并且能够更好的避免错误发生。

2. 与 Web 开发相结合

新的开发模式意味着需要更好的利用现有的各种 Web 标准，例如 HTML、XML、SOAP（简单对象存取协议）。现存的开发工具是在 Internet 出现前或是未得到充分应用前出现的，所以都不能很好的适应目前 Web 技术的开发需要。

C#开发者可以方便的在微软.NET 平台上扩展自己的应用。C#可以将任何组件转变为 Web 服务，并且可以被运行于 Internet 上的任何平台的任何应用调用，重要的是 C#对这一特性提供了内置的支持。

更重要的一点，Web 服务框架可以让任何 Web 服务都看起来类似于 C#的内置对象，所以可以让开发人员在开发过程中继续使用它们已经具备的面向对象的开发方法和技巧。

此外，C#还拥有许多其他特性使自己成为最出色的 Internet 开发工具。例如，XML 目前已经成为网络中数据结构传送的标准，为了提高效率 C#将允许直接将 XML 数据映射成为结构。这样的话可以有效的处理各种数据。

3. 减少开发中的错误

即使是优秀的 C/C++开发人员都难于避免在编码过程出现一些常见错误，比如错误的初始化一个变量，而这种错误将有可能导致各种不可预知的错误，并且难以被发现。如果一旦错误在发现前被投入生产环境，排除这些错误将会付出昂贵的代价。而 C#的先进设计思想可以消除 C/C++开发中的许多常见错误，比如：

1）垃圾收集机制将减轻开发人员对内存的管理负担。

2）C#中的变量将自动根据环境被初始化。

3）变量是类型安全的。

4）使用 C#将会使开发人员更加轻易的开发和维护各种商业应用。

4. 提供内置的版本支持来减少开发费用

更新软件系统中的组件（模块）将会是一种容易产生错误的工作，在代码修改过程中可能对现存的软件产生影响。为了帮助开发人员处理这些问题。与 C++和 Java 不同，C#在语言中内置了版本控制功能，函数重载必须显式进行，这可以防止代码级错误和保留版

本化的特性。C#另一个相关的特性是接口和接口继承的支持。这些特性可以保证复杂的软件可以被方便的开发和升级。这些特性可以帮助开发更强壮的软件后继版本和减轻开发费用。

5. 更好的结合商业应用中的流程与软件实现

为了更好实现公司的各种商业计划，在软件系统中必须在商业流程和软件实现间有紧密的联系。但是大多数的开发语言都不能轻易的将各种应用逻辑与代码相联系。例如，开发人员会使用各种注释来标明各种类所代表抽象商业对象。

C#允许使用在任何对象上使用预定义数据或是经过扩展的元数据。在系统结构中可以使用区域属性（类似 NT 的网络域结构），并且将这些属性添加到类，接口或者其他元素上。开发者可以独立的测试各种元素上的属性。这将会使得一些如同收集区域中对象属性，或是编写自动工具来保证的区域中的类，接口是否被正确定义的类似工作变得简单。

6. 可扩展的协作能力

虽然管理性强，透明性好，类型安全的开发环境对大多的商业应用都适合，但现实的经验告诉我们一些应用出于执行效率或是与现存的应用接口 API 相结合的原因需要使用原有的开发方式来进行编码。也正是如此，许多 C/C++开发人员宁愿放弃使用一些可以提高开发效率的开发工具。

C#通过下面的方法来解决应用接口的问题：

1）内置支持 COM 模型和 Windows 平台 API。在 C#中任何对象都会自动成为 COM 对象，开发者不再需要显式的实现 IUnknown 和其他一些 COM 接口，同时也可以方便而自然的使用现存的 COM 对象，而不需要关心这些 COM 对象是否使用 C#开发。

2）允许有限制的使用指针。对于使用 C#的开发人员来讲，C#允许开发人员调用 OS 所提供的 API。在经过标记的代码区域内使用指针并手工管理内存分配。这可以让 C/C++开发人员更快的熟悉和转向 C#并且不需要放弃在以前开发中所形成的开发习惯，而且以前的 C/C++代码依然可以被重用。无论是对于 COM 的支持还是对于 API 调用的支持都是为了为开发人员提供足够的开发控制能力。

1.1.4 C#、C++和 Java 的比较

C#的语言规范由 Microsoft 的 Anders Hejlsberg 与 Scott Wiltamuth 编写。对于程序员而言，对 C#和 C++、Java 作一番比较总是很有必要的。如果你早就知道 C#更接近 Java 而不是 C++，事情也不值得大惊小怪。

如果你是刚刚加入 C#阵营的读者，表 1-1 让你自己作出判断。显然，结论应该是：Java 和 C#虽然不是孪生兄弟，但 C#最主要的特色却更接近 Java。

表 1-1 比较 C#、C++和 Java 最重要的功能

功能	C#	C++	Java
继承	允许继承单个类，允许实现多个接口	允许从多个类继承	允许继承单个类，允许实现多个接口
接口实现	通过“interface”关键词	通过抽象类	通过“interface”关键词
内存管理	由运行时环境管理，使用垃圾收集器	需要手工管理	由运行时环境管理，使用垃圾收集器

续表 1-1

功能	C#	C++	Java
指针	支持，但只在很少使用的非安全模式下才支持。通常以引用取代指针	支持，一种很常用的功能	完全不支持。代之以引用
源代码编译后的形式	.NET 中间语言（IL）	可执行代码	字节码
单一的公共基类	是	否	是
异常处理	是	返回错误	是

C#是一种先进的，面向对象的开发语言，并且能够方便快捷的在微软.NET 平台建立各种应用和建立能够在网络间相互调用的 Web 服务，它所提供的框架允许 C# 组件变成 Internet 上运行在任何平台上的任何应用可获得的 Web 服务。

从开发语言的角度来讲，C#可以更好帮助开发人员避免错误，提高工作效率。

C# 语言为 C/C++开发者提供了快速的 Web 开发环境，同时也保留了 C/C++程序员想要的功能和灵活性。很多语言都是实验的产物（比如 C 和 Java），而 C#是一门经过深层思考后所提出的开发语言，这一点上要比其他的开发语言更具优势。

1.2 C#的数据类型

要使用一种编程语言，自然少不了和数据结构与算法打交道，而数据结构在编程语言中如何表示？这就要了解编程语言的数据类型了。

C#语言的数据类型主要分为两类：数值类型（Value Type）和引用类型（Referencing Type）。除此之外，指针类型（Pointer Type）在非安全代码中也是非常有效的（不过由于涉及的比较深层，本书不作详细说明）。

数值类型可以分为结构类型和枚举类型两大类，其中，结构类型包括用户自定义的结构类型和简单类型，简单类型包括数字类型和布尔类型，数字类型由整数类型、浮点数类型和小数类型组成。

引用类型包括类类型（class）、接口类型（interface）、代表类型（delegate）、数组类型、object 类型和 string 类型 6 种。可以通过图 1-1 来分类。

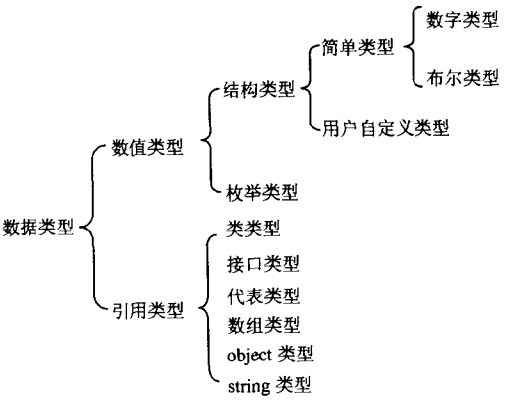

图 1-1 C# 中数据类型

两种基本数据类型最根本的区别是：数值类型的变量本身包含它们的数据，而引用类型的变量包含的是指向包含数据的内存块的引用或者叫句柄。对于数值类型，每个变量有一份自己的数据拷贝，因而也就不能通过操作其中一个来影响另一个。而对于引用类型，两个变量有可能引用同一个对象，因而也就可以通过操作一个变量来影响也被另一个变量引用的对象。

下面给出一个例子来说明这种差别：

【程序清单 1-1】

```
// 演示分别对数值类型和引用类型的变量进行操作的不同
using System;

class Class1
{
        // 声明一个变量 value,并赋初值为 0
        public int value = 0;
}

class Demo
{
        // Main 方法
        static void Main ( )
        {
            // 赋初值为 x = 0,y = 80
            int x = 0;
            int y = x;
            y = 80;

            //分别声明两个对象
            Class1 ref1 = new Class1 ( ) ;
            Class1 ref2 = ref1;
            ref2.value = 80;

            // 输出结果
            Console.WriteLine ( "Value: {0}, {1}", x, y ) ;
            Console.WriteLine ( "Refs: {0}, {1}", ref1.value, ref2.value ) ;
        }

}
```

屏幕上输出结果为：

```
Value: 0, 80
Refs: 80, 80
```

来看最后两句输出：

Console.WriteLine (“Value: {0}, {1}”, x, y) ;

Console.WriteLine (“Refs: {0}, {1}”, ref1.value, ref2.value) ;

表示的是 Console.WriteLine 中格式串的某些行为，其变元的个数是可以变的，头一个变元是一个串，包含顺序占位符{0}，{1}，{0}对应到第二变元，{1}对应到第三变元。在发送到 Console 输出之前，每个占位符都用对应变元的格式替换。

分析上面的例子：对局部变量 ref1 的赋值并不能影响局部变量 y。因为这两个局部变量都属于数值类型（int），而每个局部变量都有自己的存储域。与此不同，赋值 ref2.value = 80，影响了 ref1 和 ref2 都引用的对象。

1.2.1 数值类型

数值类型可分为结构类型和枚举类型。结构类型包括简单类型和用户自定义结构类型。枚举类型和用户自定义结构类型我们将在后面详细阐述。简单类型又可分为布尔类型和数字类型。C#语言中布尔类型严格与数字类型区分，只有 true 和 false 两种取值，不存在像 C/C++里那样和其他类型之间的转换。数字类型包括整数，浮点数和小数三种类型。整数类型有 sbyte、byte、short、ushort、int、uint、long、ulong 和 char 9 种。除了 char 类型外，其他 8 种两个一组分别为有符号和无符号两种。浮点数有 float 和 double 两种。小数用于对精度要求比较高的计算环境。

表 1-2 是对这些简单类型作了详细的描述。

表 1-2 C#中的数字类型

类型	说明	范围	示例
<整数类型>			
sbyte	8 位带符号整数	-128 ~ 127	sbyte val = 10
byte	8 位无符号整数	0 ~ 255	byte val1 = 10 byte val2 = 24U
short	16 位带符号整数	-32768 ~ 32767	short val = 10
ushort	16 位无符号整数	0 ~ 65535	ushort val1 =12 ushort val2 = 24U
int	32 位带符号整数	-2147483648 ~ 2147483647	int val = 10
uint	32 位无符号整数	0 ~ 4294967295	uint val1 = 10 uint val2 = 24U
long	64 位带符号整数	-9223372036854775808 ~ 9223372036854775807	long val1 = 10 long val2 = 24L
ulong	64 位无符号整数	0 ~ 18446744073709551615	ulong val1 = 10 ulong val2 = 24U
char	字符类型；一个 char 值即是一个 Unicode 字符	0 ~ 65535	char val = 'h'
<单精度类型，双精度类型和小数类型>			
float	字符类型；一个 char 值即是一个 Unicode 字符	1.5×10^{-45} ~ 3.4×10^{38}	float val = 2.34F
double	双精度浮点数类型	5.0×10^{-324} ~ 1.7×10^{308}	double val1 = 2.34 double val2 = 3.45D
decimal	有 28 位有效数字的高精度小数类型	1.0×10^{-28} ~ 7.9×10^{28}	decimal val = 3.45

每种数据类型都有对应的缺省值，如表 1-3 所示。

表 1-3　数据类型缺省值

类型	缺省值
数字类型（char 除外）	0 或 0.0
char	\x0000
Bool	False
布尔类型	0
结构类型	将它所有的值类型的域设置成对应值类型的缺省值，将它引用勒年个的域设置为 null
引用类型	Null

不同类型的数据之间可以转换，C#的类型转换主要分为隐式类型转换和显式类型转换数据从“小类型”到“大类型”的转换时为隐式类型转换，从“大类型”到“小类型”的转换为显式类型转换，显式类型转换需要如“(Type) data”一般的括号转换操作符。

1. 布尔（bool）类型

bool 类型是可以说是最简单的数据类型，它只有两个值：true 和 false，与 bool 对应的系统类型为 System.Boolean。

如何对一个 bool 类型的变量赋值呢？方法如下：

```
bool A = true;
bool B =  ( c >= 12 && c < 24 ) ;
```

结果将是一个逻辑结果（即 true / false），看下面的例子：

【程序清单 1-2】

```
using System;
// 声明一个 Booleans 类
class Booleans
{
        public static void Main ( )
        {
            bool content = true;
            bool noContent = false;
            Console.WriteLine ( "It is {0} that C# Station provides C# programming
                                language content.",  content ) ;
            Console.WriteLine ( "The statement above is not {0}.",  noContent ) ;
        }
}
```

在上例中，布尔值作为句子的一部分输出到控制台中。"bool"类型的取值要么为真，要么为假。程序运行之后屏幕显示结果如下：

```
It is True that C# Station provides C# programming language content.
The statement above is not False.
```

在 C++中，bool 类型并不能转换成其他的类型，也不能将其他类型的值相互之间作为条件语句中要判断的条件，这是与 C++不同的(C++中，bool 类型可以与 int 类型互相转换)。例如：

```
int val = 8;
if ( val != 0 )        // 这是正确的;但如果在 C++中，则可以写成 if ( val )
{
```

```
        Console.Write ( "The value of val is nonzero. " ) ;
    }
```

2. sbyte 类型

sbyte 是一个字节（8 位）的存储单元，在计算机中，一个字节为 8 位，每一位用 0 或 1 表示，sbyte 是一个带符号字节的整数，取值范围是 -128 ~ 127 ，与 sbyte 对应的系统类型为 System.Sbyte。

如何声明一个 sbyte 变量呢？方法如下：

```
sbyte A = 64;        // 声明一个值为 64 的 sbyte 类型的变量
```

在这里，64 的缺省类型应该是 int，被隐式地转换成了 sbyte 类型，当时如果值大于 sbyte 的范围的话，则编译出错。什么时候需要强制转换呢？看下面的例子：

```
    public static void A ( int val1 )  {}
public static void A ( sbyte val2 ) {}
```

此时，为了调用参数位 sbyte 类型的重载方法，必须使用强制转换，例如：

```
A ( 8 ) ;                  // 调用参数为 int 类型的方法
A ( ( sbyte ) 8 ) ;        // 调用参数为 sbyte 类型的方法
```

下面具体来看一下 sbyte 类型的转换。sbyte 类型可以隐式地转换成其他多种类型，例如 int，short，long，float，double 或 decimal 等。不过要注意的是，在声明 sbyte 类型的变量时可以使用字面值进行赋值（缺省值是 int 类型），但是不可以使用运算表达式进行赋值，例如：

```
sbyte x = 10，y = 15;          // 正确
sbyte z = x + y;               // 此时编译出错，int 不能隐式地转换成 sbyte
```

正确的方法是进行强制转换，第二行程序如下表示则是正确的：

```
sbyte z = （sbyte）（x + y）;
```

注：float 数是不能隐式转换成 sbyte 类型的，例如：

```
sbyte val = 1.0;  // 错误，因为都不能隐式转换成 sbyte 类型
```

正确的方法是进行强制转换，下面的程序是正确的

```
sbyte val = （sbyte）1.0;
```

当运算结构地类型和别的赋值目标的类型相同，或者赋值目标的类型可以表示的数值范围比运算结果的类型的数值范围更大时，则可以使用运算表达式进行赋值，例如：

```
sbyte x = 10，y =20;
int z = x + y;                 // 正确
long k = x + y;                // 正确
```

3. byte 类型

一个 byte 为一个字节（8 位），取值范围为 0 ~ 255，与 byte 对应地系统类型为 System.Byte。

如何声明一个 byte 类型的变量呢？方法如下：

```
byte val = 129;
```

在这里，129 的缺省数据类型是 int 类型，这里隐式转换成 byte 类型了。不过如果值超出了 byte 类型可以表示的范围，则编译的时候将报错。

关于 byte 类型的转换。由于 byte 类型在整数类型中所能表达的数值范围是最小的，所以它可以隐式地转换成其他所有的数字类型。不过，相反地，不可以把数值范围比 byte 大的类型的非字面值（变量或者表达式）隐式转换成 byte 类型。看下面的例子：