Visual C#.NET

编程精粹150例

张怀庆　谢益诚　洪　槲　编著

冶金工业出版社

Visual C#.NET 编程精粹 150 例

张怀庆　谢益诚　洪　槲　编著

北　京

冶 金 工 业 出 版 社

内 容 简 介

微软公司在新世纪推出了.NET 计划，C#语言是.NET 框架的核心语言。本书以实例的形式向读者展示了 C#语言的编程精髓以及 Visual C#编程的方法和技巧。通过 150 个实例全面介绍 C#编程基础、界面编程、图形与图像处理、多线程、多媒体、网络编程、数据库编程等。这些实例典型简洁，所涉及的技术对解决同类问题具有实用性。使用本书的最好方法是掌握实例中提供的技术或技巧，然后使用这些技术尝试实现更为复杂的功能，并应用到更多方面，在实践中掌握和领会 C#编程的基本技巧和思想精髓。

本书内容丰富、结构合理、思路清晰，可以作为广大编程爱好者并具有初步的 C#编程知识的读者提高自己编程水平的自学教材，帮助读者迅速掌握实际应用中的各种经验、技巧。

图书在版编目（CIP）数据

Visual C#.NET 编程精粹 150 例 / 张怀庆等编著.
北京：冶金工业出版社，2005.8
ISBN 7-5024-3795-9

I. V... II. 张... III. C 语言—程序设计
IV. TP312

中国版本图书馆 CIP 数据核字（2005）第 080575 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
佛山市新粤中印刷有限公司印刷；冶金工业出版社发行；各地新华书店经销
2005 年 9 月第 1 版第 1 次印刷
787mm × 1092mm 1/16； 24 印张； 708 千字； 374 页
39.00 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 Visual C#

Microsoft C#（读作 C sharp）是一种新的编程语言，它是为生成运行在.NET Framework 上的、广泛的企业级应用程序而设计的。微软公司对 C#的定义是："C#是一种类型安全的、现代的、简单的，由 C 和 C++衍生出来的面向对象的编程语言。它牢牢根植于 C 和 C++语言之上的，并可立即被 C 和 C++的使用者所熟悉。C#的目的就是综合 Visual Basic 的高生产率和 C++的行动力。" C#代码被作为托管代码编译，这意味着它能够从公共语言运行库的服务中受益。这些服务包括：语言互操作性、垃圾回收、增强的安全性以及改进的版本支持。

C#在 Visual Studio.NET 套件中作为 Visual C#引入，对 Visual C#的支持包括项目模板、设计器、属性页、代码向导、一个对象模型以及开发环境的其他功能。Visual C#编程的支持库是.NET Framework。

.NET Framework 支持生成和运行下一代应用程序和 XML Web Services 的内部 Windows 组件。.NET Framework 旨在实现下列目标：

（1）提供一个一致的面向对象的编程环境，而无论对象代码是在本地存储和执行，还是在本地执行但在 Internet 上部署，或者是在远程执行的。

（2）提供一个将软件部署和版本控制冲突最小化的代码执行环境。

（3）提供一个可提高代码（包括由未知的或不完全受信任的第三方创建的代码）执行安全性的代码执行环境。

（4）提供一个可消除脚本环境或解释环境性能问题的代码执行环境。

（5）使开发人员的经验在开发不同类型的应用程序（如基于 Windows 的应用程序和基于 Web 的应用程序）时保持一致。

（6）按照工业标准生成所有通信，以确保基于.NET Framework 的代码可与任何其他代码集成。

.NET Framework 具有两个主要组件：公共语言运行库和.NET Framework 类库。公共语言运行库是.NET Framework 的基础。.NET Framework 的另一个主要组件是类库，它是一个综合性的面向对象的可重用类型集合，可以使用它开发多种应用程序。

C#的出现给广大程序开发者又多了一种选择，而今在众多的软件设计中使用 C#的原因有：

（1）C#是一种精确、简单、类型安全、面向对象的语言，它使企业程序员得以构建广泛的应用程序。

（2）C#还提供生成持久系统级组件的能力：对集成现有代码提供完全 COM 平台支持；通过提供垃圾回收和类型安全实现可靠性；通过提供内部代码信任机制保证安全性；完全支持可扩展元数据概念。

（3）C#可以与其他语言交互操作、跨平台互用并与遗留的数据交互操作：通过 COM+ 1.0 和.NET Framework 服务提供具有紧密库访问的完全相互作用支持。

（4）C#还对基于 Web 的组件交互提供 XML 支持，且其版本转换功能使管理和部署变得更加简易。

二、本书的内容结构

本书是介绍面向对象的程序开发语言 C # 语言的专题书籍，书中通过 150 个经典实例详细介绍了用 C # 语言开发应用的方法与技巧。全书的结构如下：

第 1 章：C#编程基础。主要介绍了 C#面向对象编程的基础。
第 2 章：界面编程。主要介绍了利用 C#的各种组件编写用户图形界面。
第 3 章：图形与图像处理。主要介绍了 C#在绘制各种图形和进行各种图像变换的处理。
第 4 章：线程。主要介绍了 C#的多线程的编程。
第 5 章：文件处理和系统操作。主要介绍了 C#的文件操作和与操作系统相关的编程。
第 6 章：数据库编程。主要介绍了利用 ADO.NET 进行数据库编程。
第 7 章：网络编程。介绍了 C#的 Socket、TCP、HTTP、UDP 等编程。
第 8 章：Web 应用程序的开发。主要介绍了 ASP.NET 编程。
第 9 章：多媒体。主要介绍了用 C#编写视频和音频播放器。
第 10 章：安全性。主要介绍了 Visual C#实现程序的安全性。
第 11 章：设计模式。主要介绍了如何利用常见的设计模式进行编程。
第 12 章：杂例。介绍的内容主要是对前面章节的补充。

三、本书的特点

本书内容丰富、结构合理，叙述简洁明了，有很强的实用性，以实例的形式向读者展示了 Visual C#编程的技巧，并且有针对性地对实例中用到的技术知识点进行讲解，对每一个实例都有详细的分析和总结。

四、本书的适用对象

本书适用于已经初步掌握 C#编程概念、方法的读者阅读，可以帮助读者迅速掌握 Visual C#在各种实际应用中的技术和方法。本书针对 Visual C#的各种技术进行了全面的介绍，要求读者初步掌握 C#程序设计的基本语法，可将此书作为 Visual C#入门和提高的参考书，可以帮助读者迅速掌握 Visual C#在各个领域中的应用方法。

本书第一、二、四章由洪榭编写，第五、七、八章由谢益诚编写，剩余章节由张怀庆编写。本书的实例都通过 Visual Studio.NET 2003 企业中文版编译并给出了运行结果。虽然笔者做了最大的努力，但不足和错误之处仍在所难免，敬请读者批评指正。欢迎广大读者和专家对我们的工作提出宝贵建议，联系方法如下：

电子邮件：service@cnbook.net

网址：www.cnbook.net

本书附有的源代码可在本网站免费下载，此外，该网站还有一些其他相关书籍的介绍，可以方便读者选购参考。

编　者

2005 年 7 月

目　录

第 1 章　C#编程基础

本章通过二十一个实例简单介绍了 C#编程的基础知识，使读者对其基本语法规则有大致的了解。其中包括数据类型、类型转换、流程控制等基础知识，类、继承等面向对象编程的基础，委托、属性等 C#特有的特性，递归、排序等简单应用。

实例 1　Hello C#!

【编程要点】

在编写第一个程序之前，先对 C#（读作 C Sharp）作一个简要的介绍。

C#的运行环境：

C#在.NET 平台上运行，其各种特性与.NET 密切联系。它没有自己的运行库，许多强大的功能都来自.NET 平台的支持。它结合了 C/C++的强大功能和 VB（Visual Basic）的易用性，并引入了一些先进的程序语言特性，如内存管理（Memory Management）、垃圾回收（Garbage Collection）和类型安全（Type Safe）等等。

.NET 由虚拟对象系统（Virtual Object System，VOS）、元数据、公用语言规范（Common Language Specification，CLS）和虚拟执行系统（Virtual Execution System，VES）四个部分组成。.NET 跨语言集成的特性来自于虚拟对象系统的支持；元数据是对 VOS 中类型描述代码的一种称呼；公用语言规范是通用语言运行环境（Common Language Runtime，CLR）定义的语言特性的集合，主要用来解决互操作问题；虚拟执行系统是虚拟对象系统的实现，用来驱动运行环境。

通用语言运行环境在.NET 平台中的角色类似于 JAVA 的虚拟机（Java Virtual Machine，JVM）在 JAVA 的应用框架下的作用，让所有遵循通用语言规范开发出来的源程序通过编译器可以编译成相同的中间语言，在.NET Runtime 上可以相互调用。

通用语言运行环境是整个.NET 平台的核心。通用语言运行环境还提供了以下好处：跨语言的整合，自动化内存管理，跨语言的异常处理（Exception Handling），增加 .NET 平台的稳定性，提高效率（两次以上的执行只需要编译一次）。

.NET 程序在执行架构上和以往的程序执行架构有很大的不同。当程序的源代码被编译器编译后，产生的是一种叫做 Managed Code 的中间语言（Intermediate Language）。由于不是二进制的机器代码（Native Code），不能直接被载入 CPU 执行，而是载入.NET 的通用运行环境，再由 JIT（Just-In-Time）编译器编译为机器码，通用语言环境会将这个机器码放入主高速缓冲区中。从执行效率方面讲，若程序第一次执行这样的过程确实要比从源程序直接编译成二进制代码要慢。但若第二次以后的程序执行速度则完全相反，因为程序在执行时可以先到高速缓存区中读取前次的执行结果，不需要额外的载入和重新编译。

Microsoft Visual Studio .NET 是一个强大的集成开发环境，它提供漂亮的图形用户界面和各种其他支持机制。不过本章主要介绍 C#的初步编程，基本上利用.NET 提供的命令行方式运行编译，而暂时先不使用集成环境提供的机制。

C#程序的源代码存储在扩展名为.cs 结尾的文本文件中。安装 Microsoft Visual Studio .NET 之后，可以通过.NET 自带的编译器 csc 进行编译。比如说 Windows XP 下在 C 盘安装.NET，其编译器位置为“C:\WINDOWS\Microsoft.NET\Framework\v1.1.4322”目录下，进入系统的 command 后，将目录改到以上位置，就可以直接通过命令行进行编译，无需设置环境变量。Microsoft .NET 集成环境的安装和使用将在后边的章节中介绍。

【程序实现】

Hello world 程序的实现：

先建立一个文本文件 HelloCsharp.cs（仔细检查大小写，C#对大小写是敏感的），然后开始编写这个简单的程序：

```
//文件 HelloCsharp.cs：输出 Hello C#
using System;              //引用一个叫 System 的命名空间
class HelloCsharp          //定义类 HelloCsharp
{
   static void Main()      //静态的 Main 方法是程序的入口
   {
      System.Console.WriteLine("Hello C#");//输出 Hello C#
   }
}
```

【运行结果】

通过操作系统的 command 命令进入到编译器所在位置后，输入“csc+空格+程序所在完整目录和文件名”可对文件进行编译。机器可能会暂停一会，然后退回到提示符下。这时，编译工作已经完成，并在编译器目录下生成一个与文件名相同的可执行文件，直接输入文件名，就可以看到运行结果，HelloCsharp 的编译运行过程如图 1-1 所示。

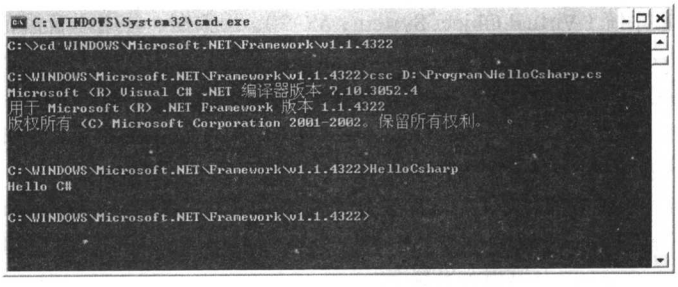

图 1-1

【分析与总结】

通过本例的学习，可以初步了解 C#程序的组织架构，明白一个 C#程序是如何运行和编译的。程序第一行引用了 System 的命名空间，该空间包含在公共基础代码构件（Common Language Infrastructure，CLI）库中，这个命名空间包含了程序体中用到的 Console 类。using 表示没有限制的使用该命名空间。

C#是纯面向对象的语言，程序执行体 Main 方法是类中的一个成员，用 static 修饰意味着没有先创建类实例时也可以调用。同时，Main()方法是 C#应用程序的入口，它必须包含在一个类中，而且仅有一个类能使用该标志定义，否则会产生编译错误。

值得注意的是，C#中 Main 的第一个字母是大写，这是与旧版本的 C 语言的小写不同的。C#语言中没有指针，程序一般用分隔符“.”来连接不同的名字，“->”操作符仅在一小部分程序中用到。但是为了保持 C#语言的弹性，程序还是可以使用关键字 unsafe 来获得指针功能。

实例 2　HeyGuy

本实例演示命令行参数和标准输入流的用法。

【编程要点】

上个实例演示了如何实现简单的输出，这个实例主要通过介绍命令行参数和用户输入的方法，实现简单的交互。

程序可以读入命令行参数，把它复制给变量，然后打印出来。标准输入流使用 Console.ReadLine()

方法同样可以实现这个功能。

【程序实现】

```
//HeyGuy.cs 程序内容：演示命令行参数和标准输入流的使用
using System;
class input
{
   public static void Main(String[] args)
   {
      string strName1;                    //声明一个 string 类型的变量
      strName1 = args[0];                 //把第一个参数赋给变量 strName1
      Console.WriteLine("Hi " + strName1);        //输出第一个参数
      Console.WriteLine("What's your partner's name?");
      string strName2;
      strName2 = Console.ReadLine();              //从控制台读取一行字符串并复制给 strName2
      Console.WriteLine("Nice to meet you, {0}" , strName2);
   }
}
```

【运行结果】

与上一实例一样，利用 csc 对该程序源文件进行编译。在运行结果时，将命令行参数附带在生成的可执行文件之后，中间以空格隔开。程序读入参数（Michael）后复制给 strName1 并输出，然后提示要求输入。标准输入流读入输入内容（Kelly）后复制给 strName2 并输出。HeyGuy 运行结果如图 1-2 所示。

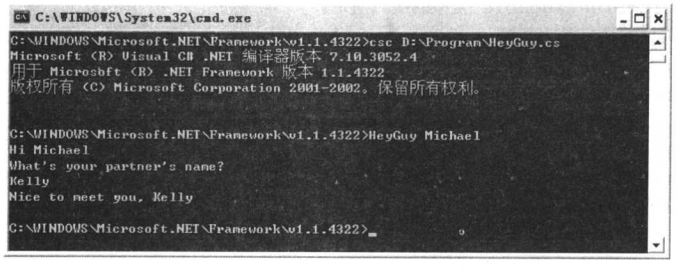

图 1-2

【分析与总结】

本例演示了命令行参数的用法，利用系统标准输入/输出流实现程序与用户的交互。Console 类表示控制台应用程序的标准输入流、输出流和错误流，此类无法继承。来自控制台的数据从标准输入流读取；传给控制台的数据会写入标准输出流；而传给控制台的错误数据会写入标准错误输出流。Console 类对从控制台读取字符并向控制台写入字符的应用程序提供基本支持。

程序中用到了 WriteLine 输出的两种用法。在输出的字符串常量和变量之间的连接，可以用加号，也可以用括号“{”和“}”加参数标号。除 ReadLine 和 WriteLine 两种处理整行字符串的方法外，Console 还支持 Read 和 Write 方法。Read 方法表示从标准输入流读取下一个字符；而 Write 方法表示将定制的信息写入标准输出流。

细心的读者可能还会发现，这个实例中类的名字和文件名并不相同。这是 C#和 Java 不同的地方：Java 要求类名和文件名必须一致，而 C#中没有这样的规定。

实例 3　预定义类型

本实例演示了部分简单预定义数据类型的声明、赋值、复制和比较等操作。

【编程要点】

本例通过对 type，short，int，long，float，double，bool，string 等几个常用预定义数据类型的介

绍，引出了这些类型声明、赋值等基本知识和在这些数据类型上提供的一些操作。

利用各种类型的 MaxValue 函数获取该类型的最大值；用操作符“=”比较两个变量是否相等；并用 System 提供的 ToString 函数将数字转换成字符。

【程序实现】

```
//SimpleTypes.cs 程序内容：演示简单预定义数据类型的部分基本操作
using System;
class SimpleTypes
{
   public static void Main()
   {
      //声明各种简单类型变量
      byte a = byte.MaxValue;
      short b = short.MaxValue;
      int c = int.MaxValue;
      long d = long.MaxValue;
      float e = float.MaxValue;
      double f = double.MaxValue;
      //输出各种类型的最大值
      Console.WriteLine("byte 类型的最大值是： " + a);
      Console.WriteLine("short 类型的最大值是： " + b);
      Console.WriteLine("int 类型的最大值是： " + c);
      Console.WriteLine("long 类型的最大值是： " + d);
      Console.WriteLine("float 类型的最大值是： " + e);
      Console.WriteLine("double 类型的最大值是： " + f);
      //比较不同类型的变量是否相等
      e = 3.14F;     //加 f 或 F 表示为 float 类型
      f = 3.14D;     //加 d 或 D 表示为 double 类型
      bool test = ( e == f );
      Console.WriteLine("e 为 " + e + " ; f 为 " + f);
      Console.WriteLine("e 和 f 的比较结果是： " + test );
      //从数字到字符串的转换并比较字符串
      string s = b.ToString();
      string t = string.Copy( s );
      bool testStr = ( s == t );
      Console.WriteLine("s 为 " + s + " ; t 为 " + t);
      Console.WriteLine("s 和 t 比较结果是： " + testStr );
   }
}
```

【运行结果】

```
byte 类型的最大值是：255
short 类型的最大值是：32767
int 类型的最大值是：2147483647
long 类型的最大值是：9223372036854775807
float 类型的最大值是：3.402823E+38
double 类型的最大值是：1.79769313486232E+308
e 为 3.14 ; f 为 3.14
e 和 f 的比较结果是：False
s 为 32767 ; t 为 32767
s 和 t 比较结果是：True
```

【分析与总结】

e 和 f 的值虽然都是 3.14，但是其比较结果为 False，这是因为两个数的数据类型不同，存储长度也不同，所以它们近似的结果也不相同。s 和 t 形式上是数字，但其存储的格式为字符串类型。值类型的变量赋值会创建所赋的值的副本，这区别于引用类型的变量赋值。引用类型的变量赋值复制的是引

用，而不是由引用标识的对象。在这个程序中，修改 s 的值后，并不影响 t 的值。

C#中预定义的数据类型不只以上这几种。比如整型中还包括段字节型 sbyte、无符号短整型 ushort、无符号整型 uint、无符号长整型 ulong 等。

由于一般的浮点类型通常都只是一个近似结果而不是精确的值，这些不可预期的输入错误可能在某些场合导致严重的后果。C#中提供 128 位的 decimal 类型（十进制类型，也叫小数类型）就适合财务和货币等计算。decimal 表示为以 10 为幂的 96 位整数，对绝对值小于 1.0 的 decimal 最多精确到 28 个小数位。与浮点型相比，decimal 类型的精度更大但范围更小。如小数 0.1 在浮点型中是二进制数的近似值，在 decimal 类型中却可以精确地表示。浮点型与小数类型之间不能随意的转换。从浮点型到小数类型可能导致溢出异常，从小数类型到浮点型则可能导致精度损失。C#中不支持两者间的隐式转换。

实例 4　类型转换

本实例演示 C#中不同类型如何进行转换。

【编程要点】

C#中对不同类型之间的转换提供了显式转换和隐式转换两种方式，但由于不同的数据类型表示的数值范围不同，这样的转换有时会导致精度丢失或者溢出等问题。在编程过程显示转换类型必须注意这个问题。

对于类型转换的溢出问题，C#提供了溢出检查状态（Checked）帮助程序员发现转换中的错误。

【程序实现】

```
//TypeTrans.cs 程序内容：演示 C#中类型转换的方法
using System;
class TypeTrans
{
   public static void Main()
   {
      //隐式转换，int 到 long 和 float 到 double
      Console.WriteLine("隐式转换：");
      int iValue = int.MaxValue;
      long lValue = iValue;        //转换成功
      Console.WriteLine("int {0} -> long {1}", iValue, lValue);
      float fValue = float.MaxValue;
      double dValue = fValue;           //转换成功，但数字精度可能会有影响
      Console.WriteLine("float {0} -> double {1}", fValue, dValue);
      //float fValue2 = dValue;    //不允许隐式转换，会导致编译错误
      //显式转换
      Console.WriteLine("显示转换：");
      int iValue2 = (int) lValue; //long 变量的值在 int 表示范围内，转换成功
      Console.WriteLine("long {0} -> int {1}", lValue, iValue2);
      lValue = lValue + 1;
      int iValue3 = (int) lValue; //long 变量值超出 int 表示范围，转换不成功
      Console.WriteLine("long {0} -> int {1}", lValue, iValue3);
      //C#中提供的显示转换溢出检查机制
      Console.WriteLine("不使用溢出检查的溢出转换：");
      byte bValue = (byte) iValue;
      Console.WriteLine("int {0} -> byte {1}", iValue, bValue);
      Console.WriteLine("使用溢出检查后的溢出转换：");
      byte bValue2 = checked ( (byte) iValue ); //这里会抛出异常
   }
}
```

【运行结果】

程序能正常编译，但在运行时会抛出异常，TypeTrans 的运行结果如图 1-3 所示。

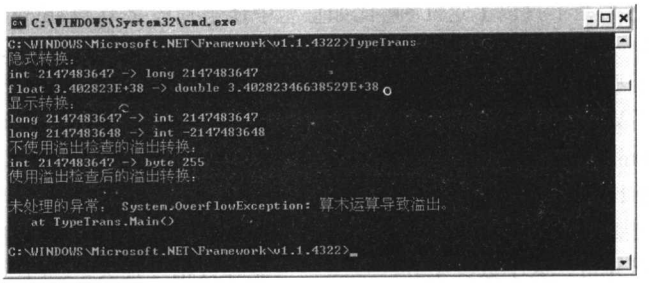

图 1-3

【分析与总结】

int 类型的值域是 long 类型值域的子集，所以从 int 到 long 的隐式转换一般不会有问题。对于浮点型则不尽如此，因为浮点类型表示的是一个小数（二进制数的近似值）从 float 类型隐式转换到 double 类型，转换能成功，但该小数的精度可能会有所增加。数值类型的显式转换必须注意其溢出问题。

从结果可以看到，当 long 类型表示的数值超出 int 表示范围时，程序没有抛出异常，但却得不到预期的结果。

C#提供一整型到另一整型转换的溢出检查状态。若处于溢出检查状态，如果源操作数的值在目标类型值域的范围内，转换就会成功；但如果源操作数的值在目标类型值域的范围外，则会引发异常。

除程序用到的几种外，C#支持的隐式数值转换包含如下转换：

（1）从 sbyte 到 short、int、long、float、double 或 decimal。

（2）从 byte 到 short、ushort、int、unit、long、ulong、float、double 或 decimal。

（3）从 short 到 int、long、float、double 或 decimal。

（4）从 ushort 到 int、uint、long、ulong、float、double 或 decimal。

（5）从 int 到 long、float、double 或 decimal。

（6）从 uint 到 long、ulong、float、double 或 decimal。

（7）从 long 到 float、double 或 decimal。

（8）从 ulong 到 float、double 或 decimal。

（9）从 char 到 ushort、int、uint、long、ulong、float、double 或 decimal。

（10）从 float 到 double。

显式转换是指当不存在相应的隐式转换时，从一种数据类型到另一种数据类型的转换。转换必须显式的指出，否则会产生编译错误。包括：

（1）从 sbyte 到 byte、ushort、uint、ulong 或 char。

（2）从 byte 到 sbyte 和 char。

（3）从 short 到 sbyte、byte、ushort、uint、ulong 或 char。

（4）从 ushort 到 sbyte、byte、short 或 char。

（5）从 int 到 sbyte、byte、short、ushort、uint、ulong 或 char。

（6）从 uint 到 sbyte、bye、short、ushort、int 或 char。

（7）从 long 到 sbyte、byte、short、ushort、int、uint、ulong 或 char。

（8）从 ulong 到 sbyte、byte、short、ushort、int、uint、long 或 char。

（9）从 char 到 sbyte、byte 或 short。

（10）从 float 到 sbyte、byte、short、ushort、int、uint、long、ulong、char 或 decimal。

（11）从 double 到 sbyte、byte、short、ushort、int、uint、long、ulong、char、float 或 decimal。

（12）从 decimal 到 sbyte、byte、short、ushort、int、uint、long、ulong、char、float 或 double。