100 时尚百例丛书

Visual C#
时尚编程百例

网冠科技 编著

光盘包含本书素材、效果文件

机械工业出版社

本书通过100个精选的实例，由浅入深地介绍了Visual C#程序设计的方法和技巧，方便读者学习以及深入地理解Visual C#。本书的主要内容有：C#初级编程、Windows 程序设计、控制语句、网络编程、数据库编程、进程和线程、高级GDI、游戏和工具。在实例中，除了介绍Visual C#主要的知识点外，还重点介绍了Visual C#在网络编程、数据库编程和游戏编程等方面的应用，使读者能够真正掌握C#语言。

本书附赠一张光盘，其主要内容包括：100个实例的源代码、资源文件、可执行文件等。

本书体系结构清晰，解释说明详细，开发实例典型。可作为广大计算机工作者和爱好者进一步学习C#语言的参考书，也可供相关计算机语言爱好者参考使用。

图书在版编目（CIP）数据

Visual C# 时尚编程百例 / 网冠科技编著.

-北京：机械工业出版社，2002.7

（时尚百例丛书）

ISBN 7-111-10584-2

Ⅰ.V … Ⅱ.网… Ⅲ.C语言-程序设计 Ⅳ.TP312

中国版本图书馆CIP数据核字（2002）第049779号

机械工业出版社（北京市百万庄大街22号 邮政编码100037）

策　　划：胡毓坚

责任编辑：王琼先

责任印制：闫　焱

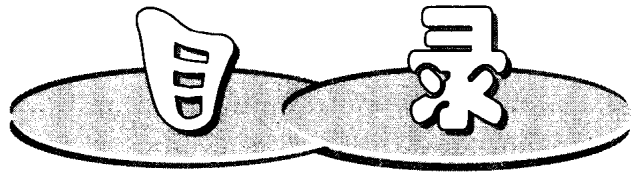
北京交通印务实业公司印刷 • 新华书店北京发行所发行

2002年7月第1版 • 第1次印刷

787mm × 1092mm 1/16 • 18.75印张 • 465千字

0001-5000册

定价：34.00元（含1CD）



第三篇　中级 Windows 程序设计

第四篇　网络程序设计

第五篇 数据库编程

第六篇 线程和进程

第七篇 高级GDI

第八篇 游戏和工具

第一篇

C#编程初步

本篇导览

在企业计算领域，C#将会变成为用于编写“下一代窗口服务”(Next Generation Windows Services,简写为 NGWS)应用程序的主要语言。C#是一种简单的、现代的、面向对象的和类型安全的程序设计语言。

在本篇中主要介绍了C#的一些基本概念、数据类型、控制语句、数组、类、虚方法及文件操作等。

通过本篇的实例编程学习，读者可以初步了解C#的程序设计方法，略窥C#的优点。

实例 1　Hello,the World!

实例说明

本例运行效果如图 1-1 所示。

本例程序运用 C#来实现一个简单而著名的程序：Hello,the World!。

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#程序是如何有效运行的，同时了解怎样编译一个 C#程序。

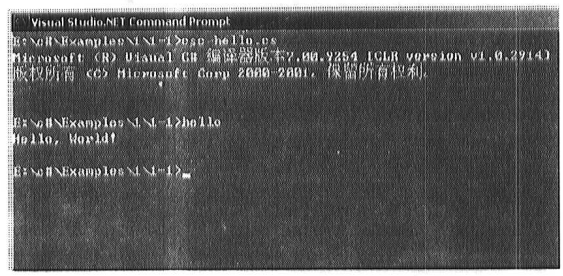


图 1-1　效果图

编程思路

本例是利用Console.WriteLine()方法来实现基本输出的。

创作步骤

在编写第一个程序之前，先对C#做一个简要的介绍。

使用微软正在推行的.NET技术和C#语言可以快速建立Web应用程序，其安全性和可升级性都大大胜过普通的ASP应用程序。C#读作“C sharp”，它是Microsoft公司开发的一种新语言，结合了C/C++的强大功能和Visual Basic的易用性。从最初的语言规范即可看出，C#无论在语法、丰富的Web开发支持还是自动化的内存管理上都和Java非常相似。因此，如果读者曾经用过C++或者Java，再来学习C#应该是相当轻松的。

开始这个程序的编制，需要注意的是：计算机上必须已经满足或超过了如下基本需求：

（1）Windows 2000 Professional、Server 或 Advanced Server；

（2）ASP＋/Microsoft ..NET (预览版可以在 http:// msdn.microsoft.com/net 下载)；

（3）Internet Explorer 5.5。

使用写字板（Notepad）来编写程序，更好的选择是使用Visual C++ 6.0或新的Visual Studio .NET，当然，还可以选择一些第三方的程序编辑器，比如CodeWright。

在满足了基本需求之后，就可以开始编制程序了。

下面我们来看看Hello, the World!是怎么实现的。

规范的“Hello, the World!”程序可以这样写：

```
using System; // 引用了一个叫System的名空间
class HellotheWorld   // 定义一个叫HellotheWorld的类
{
    public static void Main()   // 静态的Main方法是程序的入口
```

```
    {
        Console.WriteLine("Hello, the World!");  // 输出Hello,the World!
    }
}
```

C#程序的源码存储在以扩展名.cs结尾的一个或多个文本文件中，比如HellotheWorld.cs。使用命令行编译器，比如编译上例使用命令行

```
E:\C#\example>csc HellotheWorld.cs
```

这将创建一个叫HellotheWorld.exe的可执行程序。这个程序的输出是：

```
Hello,the World!
```

下面我们来仔细分析这个程序：

- using System; 表明引用了一个叫 System 的名空间，该名空间包含在公共基础代码构件（Common Language Infrastructure（CLI））类库中。这个名空间包含了在 Main 方法中用到的 Console 类。名空间提供组织程序中用到的各类库元素的分层的方法。using 表明使用该名空间没有任何限制。“hello,the world”程序使用了 Console.WriteLine 作为 System.Console.WriteLine 的简写。
- 在 C#中，代码块(语句组)由大括弧（{和}）括住。所以，即使读者以前没有 C++的经验，也可以说出 Main()方法就是 HellotheWorld 类语句的一部分，因为类被括在所定义的大括弧中。
- Main 方法是类 HellotheWorld 的一个成员。它使用 static 作为修饰，因此它是类 HellotheWorld 的一个方法，而不是类 HellotheWorld 的一个实例。
- C#应用程序（可执行）的入口点就是 static Main 方法，它必须包含在一个类中。仅有一个类能使用该标志定义，除非告诉编译器它应使用哪一个 Main 方法（否则，会产生一个编译错误）。
- public static void Main()中 public 的访问标志说明这个方法可以被任何程序访问，这是它被调用的必要条件；static 意味着没有先创建类的实例也可以调用方法；另一重要的方面是返回类型，对于方法 Main，可选择 void（意味着根本就没有返回值），或用 int 为整型结果（应用程序返回的错误级别）。
- 通过使用一个类库来输出“hello,the world”。这个标准不包含一个类库，而是改为使用 CLI 提供的基本类库。

对于C/C++开发人员，一定会很有趣地发现在“hello, the world”程序中没有出现一些东西：

- 和 C++相比，Main 的第一个字母是大写的 M，而不是你曾经使用过的小写字母。在这个方法中，你的程序开始并结束。方法中可以调用其他方法——如这个例子中，用于输出文本——或者创建对象并激活该方法。
- 这个程序不是把 Main 方法作为一个全局方法。C#不支持全局的方法和变量，这样的元素总是包含在类型定义里，例如类和结构的定义。
- 这个程序没有使用“::”和“->”操作符。在 C#中，“::”根本就不是一个操作符，“->”操作符仅在一小部分程序中用到。分隔符“.”被用来连接不同的名字，例如 Console.WriteLine。
- 这个程序不包含向前的声明。向前的声明不再需要了，定义与顺序无关。
- 这个程序没有使用#include 来导入程序正文。程序中的依赖关系被用符号象征性地处理了，而不是引用原文。这种方法消除了使用不同语言来写程序的障碍。例如，类 Console 可以用其他语言来写。

实例 2　Hello, my dear friend.

实例说明

本例运行效果如图 2-1 所示。

本例在例 1 的基础上引入 Console.ReadLine()方法，实现了简单的交互。通过这个例子的学习，读者能了解怎么通过 Console.ReadLine()方法读取用户的输入，用 Console.WriteLine()实现格式化输出，同时还可以了解怎么声明一个变量。

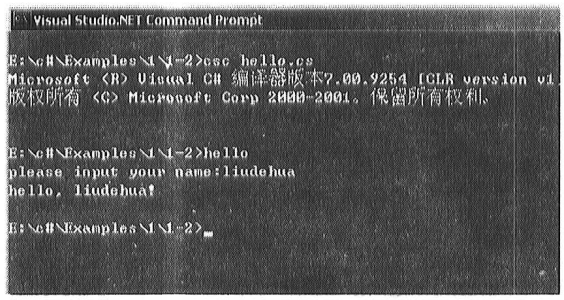


图 2-1　效果图

编程思路

本例是利用Console.ReadLine()方法来读取用户的输入，把用户的输入放在一个string类型的变量中，并用Console.WriteLine()实现了格式化的输出功能。

创作步骤

（1）声明一个string变量，用于存储用户输入的字符串。

（2）读入用户的输入。

（3）格式化输出Hello信息。

下面是规范的程序代码：

```
using System;   // 引用了一个叫System的名空间

class easyInput   // 类的名字与文件名不同也无所谓
{
 public static void Main()
 {
      string strName;   // 声明一个string类型的值变量
      Console.Write("please input your name:");   // 输出一句话，但不换行
      strName = Console.ReadLine();   // 从键盘读入用户的输入，回车表示输入结束
      Console.WriteLine("hello, {0}!", strName);   // 格式化输出hello信息
 }
}
```

实例 3 命令行参数

实例说明

本例运行效果如图 3-1 所示。

本例在例 2 的基础上引入了命令行参数。程序读入命令行参数，把它赋给变量，然后打印出来。通过这个例子的学习，读者可以了解怎么读取命令行参数。

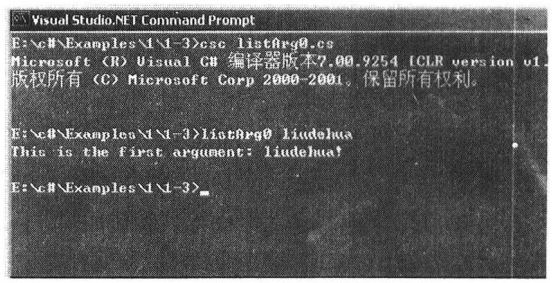


图 3-1 效果图

编程思路

本例在主函数中添加参数String[] args，通过读取参数，把它赋值给字符串变量strName，最后格式化输出字符串。

创作步骤

（1）声明一个 string 变量，用于存储命令行方式的第一个参数。

（2）把第一个参数赋值给 strName。

（3）格式化输出信息。

下面是程序代码：

```
using System;

class listArg0
{
public static void Main(String[] args)
{
        string strName;   // 声明一个 string 类型的值变量
        strName = args[0];   // 把第一个参数赋给变量 strName
        Console.WriteLine("This is the first argument: {0}!", strName);   // 格式化输出第一个参数
}
}
```

实例4 预定义类型变量

实例说明

本例运行效果如图 4-1 所示。

本例通过介绍 string/float/double/bool/object 等几个常用预定义类型，引出了 C#中预定义类型变量的声明、赋值等基本知识。通过这个例子的学习，读者可以了解 C#中都有哪些预定义类型，以及这些预定义类型变量的相关信息。

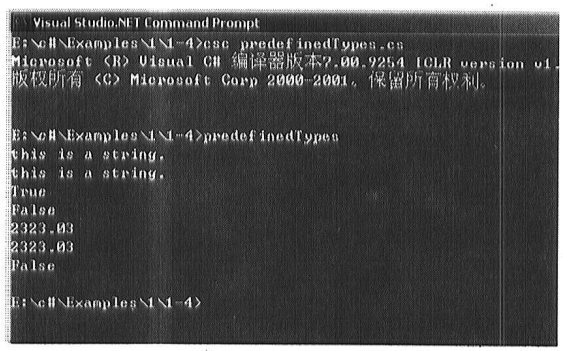


图 4-1 效果图

编程思路

本例声明了几个常用的预定义数据类型，通过赋值、比较，学习 C#中预定义类型的声明、赋值等基本知识。

创作步骤

（1）声明两个 string 变量，进行赋值、比较并打印结果。

（2）用 object 强制转换数据类型，进行比较并打印结果。

（3）声明 float 和 double 数据类型，赋相同的值，比较并打印结果。

下面是程序代码：

```
using System;

class predefinedTypes
{
 public static void Main()
 {
        string str = "this is a string.";  // 声明一个字符串变量
        Console.WriteLine(str);  // 打印出变量 str
        string strCopy = string.Copy(str); // 把 str 的值赋给另一个字符串变量 strCopy
        Console.WriteLine(strCopy);  // 打印出变量 strCopy
        bool testbool = (str == strCopy);  // 判别 str 的值是否和 strCopy 的值相等，并把结果赋给逻辑变量 testbool
        Console.WriteLine(testbool);  // 打印出 str 和 strCopy 是否相等的逻辑结果
```

```
        testbool = ((object)str == (object)strCopy);  // 判别 str 所指的对象是否和 strCopy 所指的对象相同，并把结果赋给逻辑变量 testbool
        Console.WriteLine(testbool); // 打印出 str 所指对象和 strCopy 所指对象是否相同的逻辑结果
        // float testfloat = 2323.03;  // 这样写是错误的，因为 C#中默认的数值数据类型为 double。一定要在数字后加上 F 才行。长类型到短类型的转换需要强制进行。
        float testfloat = 2323.03F;  // 这样写才是正确的。
        Console.WriteLine(testfloat); // 打印出 testfloat 的值
        double testdouble = 2323.03;  // 声明一个 double，并给它赋值
        // double testdouble = 2323.03D;  // 这样写也行，更清晰
        Console.WriteLine(testdouble); // 打印出 testdouble 的值
        testbool = ( testfloat == testdouble );  // 判别 testfloat 和 testdouble 是否相等
        Console.WriteLine(testbool);  // 这里的结果是 False，同是 2323.03，因为数据类型不同，存储的长度也不同，其近似结果也不同，故不相等。
    }
}
```

上例中对程序做了详细的注释，扼要地说明了产生这些比较结果的原因。

C# 提供了一套预定义类型，它们中的大部分是 C/C++程序员所熟悉的。

预定义引用类型包括 object 和 string。object 类型是其他所有类型的最基本的类型。string 类型被用来声明符合统一字符编码标准的字符串值。

预定义数值类型包括带符号的和不带符号的整数类型、浮点数类型、逻辑类型、字符类型和十进制小数类型。带符号的整数类型有：sbyte、short、int 和 long；无符号的整数类型有：byte、ushort、uint 和 ulong；浮点数类型有：float 和 double。

布尔（bool）类型被用来代表逻辑值：真（true）或假（false）。

布尔类型的值使得写判断语句更安全，排除了在用 C++编程时普遍出现的在该用“==”时误用了“=”的现象。像 if (i = 0)这样的语句是无效的，因为表达式 i=0 是 int 类型的，而 if 语句需要布尔类型的表达式。

char 被用来声明统一编码字符。一个 char 类型的变量代表一个 16 位的统一编码字符。

小数类型适合用在由浮点运算带来不可预期的舍入错误的计算场合中。常见的例子是金融计算、税收计算和货币转换。小数类型提供 28 位阿拉伯数字。

表 4-1 列出了所有的预定义类型，并且指明了怎么给它们赋值。

表4-1 预定义类型

类 型	描 述	例 子
object	其他所有类型的基本类型	object o = null;
string	字符串类型；一个字符串是一列统一编码字符	string s = "hello";
sbyte	8 位带符号整型	sbyte val = 12;
short	16 位带符号整型	short val = 12;
int	32 位带符号整型	int val = 12;

（续）

类　型	描　述	例　子
long	64 位带符号整型	long val1 = 12; long val2 = 34L;
byte	8 位无符号整型	byte val1 = 12; byte val2 = 34U;
ushort	16 位无符号整型	ushort val1 = 12; ushort val2 = 34U;
uint	32 位无符号整型	uint val1 = 12; uint val2 = 34U;

ulong	64 位无符号整型	ulong val1 = 12;
		ulong val2 = 34U;
		ulong val3 = 56L;
		ulong val4 = 78UL;

float	单精度浮点类型	float val = 1.23F;
double	双精度浮点类型	double val1 = 1.23; double val2 = 4.56D;
bool	布尔类型，值非真即假	bool val1 = true; bool val2 = false;
char	字符类型，其值为一个统一编码字符	char val = 'h';
decimal	精确的小数类型，有 28 位阿拉伯数字	decimal val = 1.23M;

每个预定义类型都是系统提供的类型的缩写。例如，关键词 int 指向结构 System.Int32。从编码风格来说，使用这些关键字比使用完整的系统类型名更有益。

像 int 等预定义值类型被特殊处理了，但大部分还是严格地和其他结构一样来对待。操作符重载使得开发人员可以自己定义新的和预定义值类型很像的结构类型。例如，Digit 结构能够支持预定义整型的数字操作，也能定义 Digit 和预定义类型的转换。

预定义类型使用操作符重载它们自身。例如，比较操作符 == 和!= 对于不同的预定义类型有不同的含义：

- 如果两个 int 类型的表达式表示同一个整型值，就认为这两个表达式相等了。
- 如果两个 object 类型的表达式指向同一个对象或同时为 null，则认为这两个 object 类型的表达式相等。
- 如果两个 string 类型的表达式长度相同并且在任意相同位置的字符相同，或者同时为 null，则认为这两个 string 类型的表达式相等。

例如上例中的 bool testbool = (str == strCopy) 比较的是两个 string 类型的表达式，而 testbool = ((object)str == (object)strCopy) 比较的是两个 object 类型的表达式。

关于操作符重载的内容我们留到以后的章节再讨论。

实例 5 预定义类型的转换

实例说明

本例运行效果如图 5-1 所示。

本例介绍了预定义类型之间的转换方法。

通过这个例子的学习，读者可以了解C#中的转换分类、如何进行转换以及转换的基本原则。

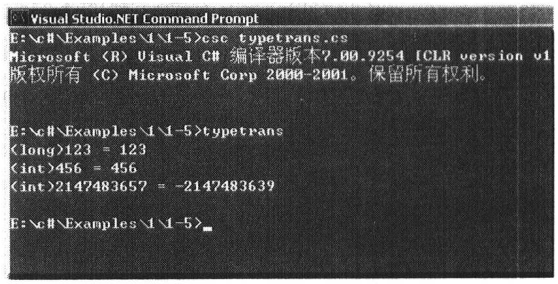


图 5-1 效果图

编程思路

在程序设计过程中，经常需要在各种数据类型之间进行转换。本例中包含两种类型转换方式：隐式转换和显式转换。

创作步骤

（1）声明 int 类型变量 intValue1、intValue2; 声明 long 类型变量 longValue1、longValue2。

（2）分别赋值 intValue1 = 123, longValue1 = 456。

（3）把 intValue1 隐式转换为 longValue2 的值。

（4）把 longValue1 显示转换为 intValue2 的值。

（5）打印出转换结果，转换成功。

（6）声明 long 类型变量 longValue3，并给其赋值 2147483657L。

（7）把 longValue3 显示转换为 int 类型的变量 intValue3。

（8）打印转换结果，转换失败。

下面是程序代码：

```
using System;
class typeTrans
{
public static void Main()
{
        // 转换成功的例子
        int intValue1, intValue2;
        long longValue1, longValue2;
            intValue1 = 123;
```

```
        longValue1 = 456;
        longValue2 = intValue1;  // 隐式转换
        intValue2 = (int)longValue1;  // 显示转换
        // longValue1的值为456，在int类型能存储的范围内，转换成功
        Console.WriteLine("(long){0} = {1}", intValue1, longValue2);
        Console.WriteLine("(int){0} = {1}", longValue1, intValue2);

        // 转换失败的例子
        long longValue3 = 2147483657L;
        int intValue3 = (int) longValue3;
        // int存放的最大的数为2147483647，在这里把2147483657赋值给它，所以溢出了，转换失
败。
        Console.WriteLine("(int){0} = {1}", longValue3, intValue3);
    }
    }
```

int类型是32位带符号的整形，其存储的范围是

–2147483647<= intValue < 2147483647

上例中longValue3的值2147483657，大于2147483647，超越了int类型可以存储的范围，所以出错了。所以这种转换一定要小心，尽量避免溢出错误。

预定义类型都有预定义转换。C# 可区别两种类型的转换：转换显式和隐式转换。

隐式转换通常在安全的不需要仔细审查的场合使用，例如，把 int 类型转换为一个 long 类型就是一个隐式转换。这些转换总是成功，不会引起数据的丢失。隐式转换可以由系统悄悄地执行。上例中，int 类型被悄悄地转换成了 long 类型。

相反，显式转换需要有明了的转换表达式才能执行，例如，上例使用了显式转换把一个 long 类型转换成一个 int 类型，结果第一个转换正确，而第二个转换出错了：

(int)2147483657 = –2147483639

因为发生了溢出。

转换表达式禁止同时使用显式的和隐式的转换。

实例6 if 语 句

实例说明

本例运行效果如图 6-1 所示。

本例介绍了 if 语句的使用，并且对 if 语句中表达式的值做了分析。

通过这个例子的学习，读者可以了解 if 语句的语法要求，以及 bool 值。

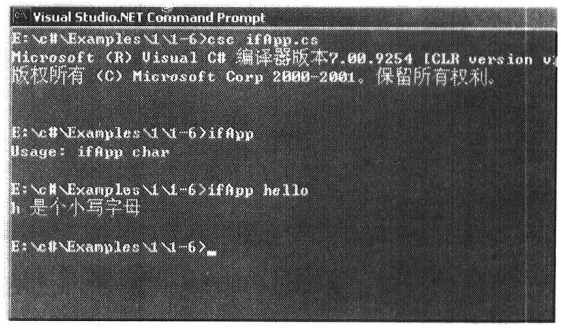


图 6-1 效果图

编程思路

任何一种编程语言，都必须使用程序控制流，这样才能使程序按照一定方式执行。if()语句是一种重要的程序控制语句。

C#中提供if()条件语句，其语法为：

```
if(表达式)
{
    语句1;
}
else
{
    语句2;
}
```

表达式的值必须为bool类型，即true或false，而不再像C语言中一样，int类型的值也可以作为判断的标准。这样，就减少了出错的可能性。

如果表达式的值为true，则执行语句1，否则，执行语句2。

if()语句可以嵌套。如：

```
if(表达式)
{
语句1;
}
else
{
if(表达式2)
```