

时尚百例丛书



Visual C#

时尚编程百例



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 $\times$ $\frac{1}{16}$ ·18.75 印张·465 千字

0001-5000 册

定价：34.00 元 (含 1CD)

凡购本图书，如有缺页、倒页、脱页，由本社发行部调换

本社购书热线电话：(010) 68993821、68326677-2527

封面无防伪标均为盗版

出版说明

随着 21 世纪的到来,人们更深切地感受到了计算机在生活和工作中的作用越来越重要,越来越多的职业需要具有计算机的应用技能。掌握计算机是职业的需要,更是事业发展的需要。

目前计算机技术不但广泛地应用在办公自动化中,它还全面渗透到各行各业。如果要从事平面设计的相关行业,就应该学会平面设计软件,如 Photoshop、CorelDRAW、FreeHand 等;如果要从事三维设计的相关行业,就应该学会三维设计软件,如 3DS MAX、Maya、Poser 等;如果要从事多媒体设计的相关行业,就应该学会多媒体制作软件,如 Authorware、Director、Premiere 等;如果要从事与网络相关的行业,就应该学会 Flash、Dreamweaver、Fireworks、ASP、PHP、JavaScript 等;如果要从事建筑产品、工业产品设计的相关行业,就应该学会 AutoCAD、3DS VIZ、Protel 等;如果要从事软件开发的相关行业,就应该学会 VB、VC、VFP、Delphi、PowerBuilder 等编程。

所有与计算机相关的职业都要求工作者有很强的计算机操作技能,做到运用自如,熟练而且深入地掌握软件的应用。而要做到这一点,必须从软件的各个方面入手,通过实例演练的方式训练自己,而且要反复练习,做到举一反三。

为了让读者能深入而且熟练地掌握相关软件的应用方法,机械工业出版社特别为广大读者推出了这套时尚百例丛书。本丛书对每一个应用软件精心制作了 100 个实例,其宗旨就是让读者全方位掌握软件的应用,为广大读者提供一条快速掌握计算机应用技能的捷径。

本丛书采用新颖的版式,将知识和实例紧密结合,通过对各种实例的详细讲解,使读者不必事先学习各种软件,而从实例的制作过程中体会到每个软件每项功能的使用方法,并自己做出各种实例效果,这样既节省了大量时间,同时也使读者有身临其境的感觉,并可以反复演练,将所学知识运用到职业工作中去。

书山有路勤为径。愿广大读者能通过本丛书的学习掌握计算机技能,并应用到自己的工作和事业中去。

机械工业出版社



前言

Visual C#(读作 C sharp)语言是微软公司最新推出的程序设计语言,是由 C/C++语言演变而来,并结合了 Java 语言的特点。C#是一种具有简单、灵活、面向对象、类型安全和版本可控等多方面优点的程序设计语言。它为网络开发、数据库开发、游戏开发等提供了多个简单易用的类库,同时,它还能方便地调用其他语言组件。

C#语言的初学者可以通过本书系统深入快速地学习 C#语言,使自己在使用 C#语言开发应用程序方面的能力和技巧上一个新的台阶。

C#语言的中级开发者可以通过本书加深对 C#语言的了解,学习到一些实用的技巧,达到快速开发应用程序的目的。

在介绍每个实例时,主要采用以下结构:

(1) 实例说明:对本实例的目的、涉及的知识点做简要的介绍,为学习各知识点提供参考。

(2) 编程思路:介绍一些涉及的知识点内容,以及创作本例的主要思路,将碰到的难点及其解决方法。

(3) 创作步骤:按照实际的步骤一步一步讲解,读者依据这些创作步骤就能完成本例中的编程任务。主要提供了控制台程序的编写和利用 Visual Studio.NET 编程环境。

本书内容丰富,涉及范围广泛,从简单的 Hello World 程序开始,到深入解剖多线程和进程的使用,到游戏和工具的开发,都有详细的讲解。读者可以循序渐进地学习,也可以将本书作为编程的参考书籍。

本书的 100 个实例都是针对某一个或一类应用问题而编写的,具有很强的针对性。掌握了这些实例的开发方法和技巧,再开发大型软件就比较容易了。

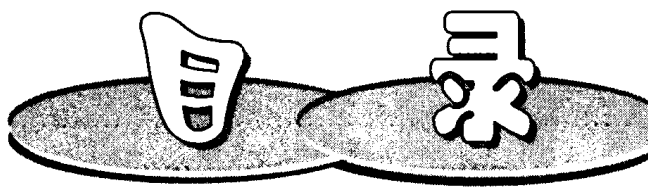
本书由网冠科技策划并组织编写,编者都具备深厚的软件理论基础和丰富的开发经验。

由于时间仓促,水平有限,书中错误和疏漏之处在所难免,恳求专家、同行以及广大读者指正。



网冠科技

本书光盘含配套素材,技术支持请点击网冠科技站点 <http://netking.163.com>。E-mail: netking_@yeah.net。



出版说明 前言

第一篇 C# 编程初步

实例 1	Hello,the World!	2
实例 2	Hello, my dear friend.	4
实例 3	命令行参数	5
实例 4	预定义类型变量	6
实例 5	预定义类型的转换	9
实例 6	if 语句	11
实例 7	switch 语句	15
实例 8	for 循环	17
实例 9	while 循环	19
实例 10	do-while 语句	22
实例 11	foreach 语句	25
实例 12	数组	27
实例 13	锯齿型数组	30
实例 14	枚举类型	33
实例 15	位运算	36
实例 16	读写文件	38
实例 17	简单的类	42
实例 18	操作符重载	47
实例 19	名字空间	52
实例 20	虚方法	56

第二篇 Windows 程序设计初步

实例 21	第一个 Windows 应用程序	60
实例 22	简单的图片浏览器	62
实例 23	点不中的按钮	65



实例 24	使用状态栏	67
实例 25	使用菜单	68
实例 26	使用工具条	70
实例 27	自制数字时钟	73
实例 28	自己制作控件	75
实例 29	使用自己制作的控件	78
实例 30	使用树控件	80
实例 31	浏览我的电脑	82
实例 32	使用进度条	85
实例 33	使用 ToolTips 控件	87
实例 34	电子便条	89
实例 35	可调窗口	92
实例 36	MDI 窗口	95
实例 37	漂浮的窗口	97
实例 38	使用列表选择框	99
实例 39	使用 DateTimePicker	102
实例 40	使用拆分窗口	104

第三篇 中级 Windows 程序设计

实例 41	Do It Yourself	107
实例 42	在程序运行中添加控件	109
实例 43	简易计算器	111
实例 44	透明窗体	115
实例 45	网页生成器	117
实例 46	图像转换器	119
实例 47	文本搜索器	122
实例 48	显示所有字体	128
实例 49	满堂红游戏	130
实例 50	拼图游戏	135

第四篇 网络程序设计

实例 51	用 Socket 建立服务器程序	142
实例 52	用 Socket 建立客户端程序	144
实例 53	时钟服务器	146
实例 54	时钟客户端	149

实例 55	Echo 服务器	152
实例 56	Echo 客户端	154
实例 57	命令行聊天服务器	158
实例 58	命令行聊天客户端	160
实例 59	得到任意主机的 IP	163
实例 60	得到任意 Web 服务器默认页	165
实例 61	IRC 客户端	168
实例 62	使用异步机制获取网页	172
实例 63	发送 SMTP 邮件	175
实例 64	邮件提示器	178
实例 65	网页上/下传器	183
实例 66	自己制作 Web 服务器	188
实例 67	WhoIs 工具	192
实例 68	ping 工具	196
实例 69	Windows 聊天服务器	199
实例 70	Windows 聊天客户端	203

第五篇 数据库编程

实例 71	用 ADO 读数据库	207
实例 72	用 ADO 写数据库	209
实例 73	从 DataSet 中读取数据库数据	211
实例 74	读取列的属性	213
实例 75	用 DataGrid 显示数据	215
实例 76	使用 DataTable	217

第六篇 线程和进程

实例 77	启动线程	222
实例 78	停止线程	224
实例 79	线程休眠	226
实例 80	挂起/恢复线程	228
实例 81	加入线程	230
实例 82	线程应用	232
实例 83	得到所有进程	235
实例 84	读取注册表信息	237
实例 85	使用剪贴板	240

第七篇 高级 GDI

实例 86	GDI+的基本用法	243
实例 87	使用刷子(Brush)	246
实例 88	运动的球	248
实例 89	旋转和转换	252
实例 90	运动的秒针	255
实例 91	模拟播放动画	257
实例 92	不规则窗体	260
实例 93	仿制 MediaPlayer	262

第八篇 游戏和工具

实例 94	翻牌游戏(一)	266
实例 95	翻牌游戏(二)	269
实例 96	三点一线游戏	273
实例 97	吃豆子游戏	277
实例 98	决策游戏	280
实例 99	给图像添加版权信息	284
实例 100	撞砖块游戏	288

第一篇

C#编程初步

本篇总览

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, 简称为 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。

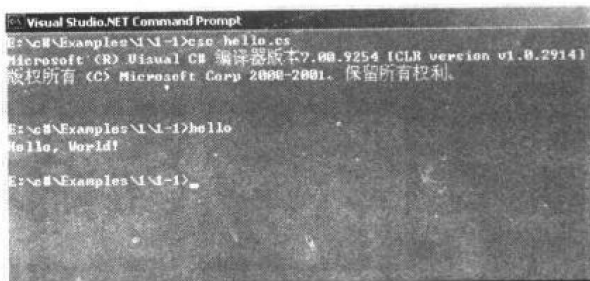


图 1-1 效果图

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。

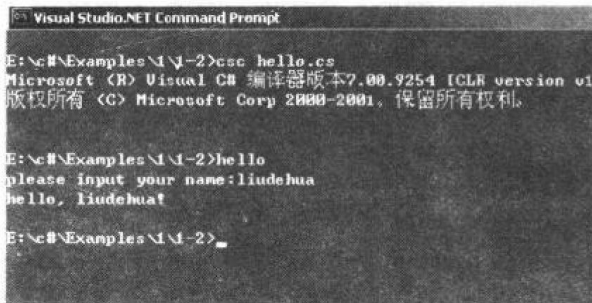


图 2-1 效果图

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。

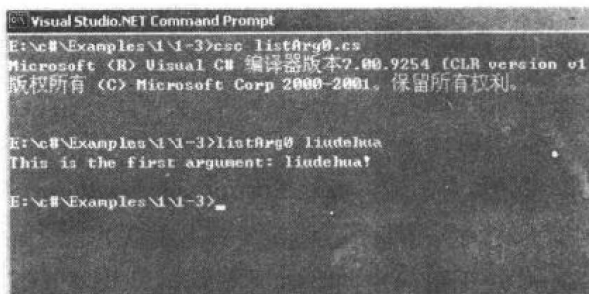


图 3-1 效果图

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
```

```
}
```

```
}
```

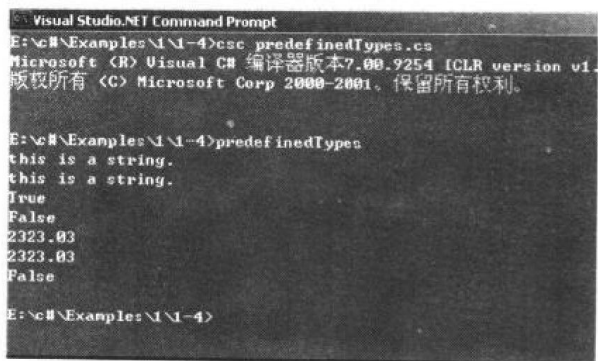
实例4 预定义类型变量



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。



```
Visual Studio .NET Command Prompt
E:\c#\Examples\1\1-4>csc predefinedTypes.cs
Microsoft (R) Visual C# 编译器版本 7.00.9254 [CLR version v1.1.438.0]
版权所有 (C) Microsoft Corp 2000-2001. 保留所有权利。

E:\c#\Examples\1\1-4>predefinedTypes
this is a string.
this is a string.
True
False
2323.03
2323.03
False
E:\c#\Examples\1\1-4>
```

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。