“十五”国家重点电子出版物规划项目 计算机基础知识普及和软件开发系列
2002精通热门软件工具丛书（3）

Inside C#

C#编程从入门到精通

北京希望电子出版社 总策划
丁鹏 编 写

中国科学出版集团
北京希望电子出版社

“十五”国家重点电子出版物规划项目 计算机基础知识普及和软件开发系列
2002 精通热门软件工具丛书（3）

Inside C#

C# 编程从入门到精通

北京希望电子出版社 总策划
丁鹏 编 写

中国科学出版集团
北京希望电子出版社

内 容 简 介

本书既为入门者提供了简单教程，同时也为高级编程人员提供了大量的例程。由3部分组成：

第一部分是C#入门教程，用12章阐述了C#语言的最基本特征和主要功能，包括：C#介绍；一个简单的欢迎程序；表达式、类型和变量；控制语句——选择；控制语句——循环；方法；名称空间；类的入门；类的继承；多态性；属性；索引指示器。

第二部分是本书的重点，精选了大量C#实用例程来让读者更深入地理解C#，同时这些C#例程都是经过专业程序员测试，可以直接实用，从而大大缩短编程时间，提高编程效率，这部分一共有文件访问；API；客户/服务器；数据库；线程处理及电子邮件；图像处理；空间；杂项等8个方面**34**个典型例程，是高级程序员不可多得的素材。

最后一部分提供了为第一次接触C#的C/C++程序员准备的C# FAQ，通过9个方面**48**个问题来回答C/C++程序员一些有关C#的基本问题，使得他们可以尽快了解并能迅速使用C#。

本书用来帮助现在的C/C++开发者迅速跟进至C#，此外，有Java和Delphi经验的开发人员或对强大的C#有兴趣的其他程序员也会发现这本书很有意义。

本书内容极为丰富，有章可循、编排精细、可操作性强；并提供了针对具体的数据库管理问题的、极具参考价值的解决方案。

本版CD为内容中提到的源代码。

系 列 盘 书：“十五”国家重点电子出版物规划项目　计算机知识普及和软件开发系列
2002精通热门软件工具丛书（3）
盘 书 名：Inside C#——C#编程从入门到精通
总 策 划：北京希望电子出版社
文本著作者：丁鹏
CD制作者：希望多媒体开发中心
CD测试者：希望多媒体测试部
责任编辑：栾大成
出版、发行者：北京希望电子出版社
地 址：北京中关村大街26号，100080
网址：www.bhp.com.cn　E-mail: lwm@hope.com.cn
电话：010-62562329, 62541992, 62637101, 62637102, 62633308, 62633309
（图书发行和技术支持）
010-62613322-215（门市）　010-62547735（编辑部）
经 销：各地新华书店、软件连锁店

排 版：希望图书输出中心　周宇
CD生产者：北京中新联光盘有限责任公司
文本印刷者：北京媛明印刷厂
开本/规格：787毫米×1092毫米　16开本　26.85印张　625.5千字
版次/印次：2002年3月第1版　2002年3月第1次印刷
本 版 号：ISBN 7-900088-71-7
印 数：1~3000册
定 价：42.00元（本版CD）

前言

在过去二十年中，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，这种灵活性是以生产的成本作为代价的。就拿 VB 来与之比较，C 和 C++应用程序相对来说需要较长的开发时间。由于复杂性和长周期，许多 C 和 C++程序员已经在寻找一种能在功能和生产力之间提供更好均衡的编程语言。

有几种编程语言是通过牺牲 C 和 C++程序员常常需要的灵活性来提高生产力的。这样的解决方案对开发者的约束太多（例如：通过省略低级代码控制）并且通用性很差。它们与先前存在的系统很难相互操作，并且它们与当前的 Web 设计方法不能很好的吻合。

对 C 和 C++程序员来说，理想的解决方法应该是快速的开发与访问所有潜在平台的能力相结合。开发环境应该完全与新的 Web 标准同步并容易与现存的应用系统集成，同时,C 和 C++程序员还希望能够在必要时在底层编写代码。

微软为了解决上述问题，提出了一种叫 C#（读作：C sharp）的语言。C#是一种现代的、面向对象的语言，它使开发人员能够在微软新的 .NET 平台上快速建立广泛的应用，其提供的工具和服务能充分发掘系统的计算和通讯能力。

因为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。通过这种继承，C# 高度保持了与 C 和 C++的一致。开发者只要熟悉 C 和 C++语言就可以快速地掌握 C#，并写出 C# 应用程序。

可以预料到的是：C#将很快流行起来并成为.NET 平台上的最主要开发语言。对于 C/C++程序设计者来说，目前急迫地需要一本能详细介绍 C#，并且提供大量例程的工具型、实用型书籍。对于入门者来讲，一本通俗易懂，简单易学的教材类书籍也是当务之急。本书就是在这样一种思路下撰写的：它既为入门者提供了简单的教程，同时也为高级编程人员提供了大量的例程。

本书内容

第一部分是 C#入门教程。这个教程一共用 12 章阐述了 C# 语言的最基本特征和主要功能，包括：C#介绍；一个简单的欢迎程序；表达式、类型和变量；控制语句——选择；控制语句——循环；方法；名称空间；类的入门；类的继承；多态性；属性；索引指示器。

第二部分是本书的重点，它精选了大量 C#实用例程来让读者更深入地理解 C#，同时这些 C#例程都是经过专业程序员测试，可以直接使用，从而大大缩短的编程时间，提高了编程效率。这部分一共有文件访问；API；客户/服务器；数据库；线程处理及电子邮件；图像处理；空间；杂项等 8 个方面 34 个典型例程，是高级编程者不可多得的素材。

最后一部分提供了专门为第一次接触 C#的 C/C++编程者准备的 C#FAQ。它通过 9 个方面 48 个问题来回答了 C/C++编程者一些有关 C#的基本问题，使得他们可以尽快地了解

C#，并能迅速使用 C#。

建议阅读方式

如果您是 C#的初学者，并且对于 C/C++不是很熟悉的话，可以先从第一部分看起，了解 C#的一些基本知识。如果您已经有很多 C#的编程经验，或者现在正在编写 C#程序，可以直接看第二部分的例程，其中有些程序段是可以直接使用的。如果您是 C/C++编程者，现在正准备使用 C#，可以先看一下第三部分，了解一些 C#与 C/C++的区别，以便能尽快上手。

联系作者

作者希望能以自己的努力，为读者提供尽可能多、尽可能好的帮助，使得不同档次的读者都能从本书中得到所需要的知识。限于作者本身能力的原因，书中在所难免的会出现错误和脱漏之处，敬请读者谅解。如果您在阅读过程中发现了问题、或者建议和意见，请和作者（hw3588_cn@sina.com）联系，作者将不胜感激。

本书光盘使用方法

下面是在《C#教程及精选例程》一书中出现的源代码文件的主要情况和运行调试环境。

包含的内容

本书光盘包含《C#教程及精选例程》中出现的绝大部分例程源代码。包括：**API** 的 3 个例程：C#自动编译器、XML 目录列表、一个类似 DOS 的应用；**客户_服务器**的 4 个例程：SNTP 客户端、聊天_客户服务器、文件共享客户_服务器、C#浏览器；**控件**的 7 个例程：C#布局管理器、ListView、编写定制的 AboutBox、创建.NET SDK 控件、向 Windows Form 中添加菜单支持、一个 C#服务器的下拉菜单、增加/删除用户控件；**数据库**部分的 7 个例程：.NET SQL 程序、网上书店、数据库浏览器（1~4）、一个数据库访问类；图像处理的 3 个例程：C#中的画笔、一个 C#图像组件、一个颜色向导；**文件访问**部分的一个例程：读一个文本文件；**线程处理及电子邮件**部分的 3 个例程：C#线程处理、SmtpMail、编写 Unsafe 代码；**杂项部分**的 4 个例程：计算器、记事本、使用 C#探测操作系统版本、数字时钟。

运行调试环境

本书提供的都是源代码，并不是编译后的可执行程序，所以用户可以针对自己的需要选取相应源代码段，并在实际情况中做适当修改，编写自己的代码。可以直接从光盘上将源代码拷贝到编辑器中，编译或者修改，这极大的提高了这些源代码应用的灵活性。

在光盘的例程目录中，提供了一个 AssemblyInfo.cs 的文件，其中详细说明了与本例程有关的编译情况。

可以采用 C#命令行编译器（在.NET SDK 中）csc.exe 来编译这些源代码，Visual Studio 的下一个版本（Visual Studio 7，或者是 Visual Studio.NET），将会全面支持 C#开发。

安装运行环境

安装.NET SDK 是在机器上运行 C# 的第一步。.NET SDK 可以安装在 Windows ME、Windows NT 或 Windows 2000/XP 上，但是最好的选择是 Windows 2000/XP 上。选择了操作系统后，再执行以下步骤：

1. 安装 IE 5.5 。
2. 安装 Microsoft .NET Framework SDK。它是免费的，可以从 http://msdn.microsoft.com/downloads/default.asp?URL=/code/sample.asp?url=/msdn-files/027/000/976/msdncompositedoc.xml 下载。
3. 完成以上安装后，就能在文本编辑器中编写代码，最后保存为扩展名为.cs 的文件。

C#编辑器

编写 C#程序可以在文本编辑器中进行，或者在集成开发环境 Visual Studio 中进行

注意事项

Microsoft.NET SDK 的不同版本会有些不兼容现象，本书中的个别例程在某些版本下可能无法运行，此时只要将其中的引用文件名称改动即可。

（本文内容同光盘 readme.txt）

目　录

第一部分　C# 入门教程

第三部分 附录 C#FAQ

第一部分　C# 入门教程

对于以前没有接触过 C#语言的读者来说，一本好的入门教材无疑将会使自己的学习事半功倍。本书第一部分提供了一个简单、易学的入门教程，它用浅显、易懂的例子，将 C#语言的基本特征和基本功能展现在你面前，你不必阅读 C#语言规范，就可以轻松进入 C#殿堂。准备好了吗？那我们就开始了。

第 1 章　C#介绍

欢迎来到 C#（发音 c sharp）世界！

这一章将让你对 C#有一个基本了解，并回答一些相关问题，如：你为什么要使用 C#？C++和 C#主要有什么不同？为什么 C#使软件开发更容易而且还更有趣？……

你也许首先会问这样一个问题：当已经使用 C++或 VB 从事企业级开发时，为什么还要学习另一种语言呢?

以商业的观点来看，答案是这样的：

在企业计算领域，C#将会变成为用于编写“新一代 windows 服务（next generation windows services，简写为 ngws）”应用程序的主要语言。

C#语言自 C/C++演变而来，相比之下，它更加现代、简单、完全面向对象和类型安全。如果你是 C/C++程序员，学习起来将简单的多。许多 C#语句直接使用你最喜欢的语句，包括表达式和操作符，乍一看，还以为它就是 C++。

关于 C#最重要的一点：它是现代的编程语言。它简化和革新了 C++中的类、名称空间、方法重载和异常处理等领域。摒弃了 C++的复杂性，使它更易用、更少出错。

C#减少了 C++的一些功能，更易于使用，不再有宏、模板和多重继承。对企业级开发者来说，上述功能只会产生更多的麻烦而不是效益。

使编程更方便的新功能是严格的类型安全、版本控制、垃圾回收（garbage collect）等。所有这些功能的目标，都是在于开发面向组件的软件。

总的来讲，C#具有一下一些特征：

- 简单
- 现代
- 面向对象
- 类型安全
- 版本控制
- 兼容
- 灵活

1.1　简单

C#的一个优势是简单易学。该语言首要的目标就是简单，很多功能（还不如说是缺少了 C++的一些功能）有助于 C#全方位地简化。

在 C#中，没有 C++中流行的指针。在缺省情况下，你工作在受管理的代码中。在那里不允许一些不安全的操作诸如直接存取内存等。

与指针的“悲剧性”密切相关的，就是“愚蠢”的操作。在 c++中，有“::”、“.”、和“->”操作符，它们分别用于名称空间、成员和引用。对于新手来说，操作符至今仍是学习的一道难关。C#弃用其他操作符，仅使用单个操作符“. ”。现在程序员所需要的就是对嵌套名字的理解了。

你不必记住基于不同处理器架构的隐含的类型，甚至各种整型的变化范围。C#使用统一的类型系统，摒弃了 C++多变的类型系统。这种系统允许你把各种类型作为一个对象查看，以确定它是一个原始类型还是一个真正的类。和其它编程语言相比，由于封装（boxing）和反封装（unboxing）机制，把简单类型当作对象处理并不能获得性能的改善。后面章节中将详细解释封装和反封装，但基本上，仅当需要时才能使用简单类型作为对象来访问。

老练的程序员可能不喜欢它，但是整型和布尔型如今终究是两种完全不同的数据类型。这就意味着原来 if 语句中错误的赋值现在会被编译出错，因为 if 语句只接受布尔类型的值，再也不会出现误用赋值符为比较符这样的错误！

C#同时也解决了存在于 C++中已经有些年头的冗余。这种冗余包括对常数和多种不同字符类型的#define 预定义。

1.2　现代

投入学习 C#的努力将是值得的，因为 C#是为编写 ngws 应用程序的主要语言而设计，你将会发现很多自己用 C++无法实现或者很费力实现的功能，在 C#中不过是一部分基本功能而已。

对于企业级编程语言来说，新增的金融数据类型很受欢迎。C#用到了一种新的十进制数据类型，它专用于金融计算方面。如果不喜欢这种现成简单的类型，根据应用程序的特殊需求，你也可以很容易地创建出新的一种数据类型。

前面已经提到，指针不再是 C#编程武器的一部分。不要惊讶，全面的内存管理已经不是你的任务。运行时 ngws 提供了一个垃圾回收站，负责 C#程序中的内存管理。当内存和应用程序都受到管理时，就很有必要加强类型安全，以确保应用程序的稳定性。

对于 C++程序员，异常处理确实不是新的东西，可它还是 C#的主要功能。C#的异常处理与 C++的不同点在于它是跨语言平台的。在没有 C#之前，你必须处理怪异的 hresults，但现在由于使用了基于异常的健壮的出错处理，　这一切都结束了。

对于现代的应用程序，安全是首要的，C#也不会例外。它提供了元数据语法，用于声明 ngws 安全模式的性能和许可。元数据是 ngws runtime 的一个关键的概念。

1.3　面向对象

C#当然支持所有关键的面向对象的概念，如封装、继承和多态性。完整的 C#类模式构建在 ngws runtime 的虚拟对象系统（vos，virtual object system）的上层。对象模式只是基础，而不再是编程语言的一部分。

你一开始必须关注的，就是不再有全局函数、变量或者是常量。所有的东西都被封装在类中，包括实例成员（通过类的实例——对象，可以访问成员）或静态成员（通过数据类型访问）。这样就使 C#代码更加易读，且有助于减少潜在的命名冲突。

定义类中的方法在默认条件下是非虚拟的（它们不能被派生类改写）。主要原因是，这样做会消除由于偶尔改写方法而导致另外一些源码出错。要改写方法，必须具有明确的虚拟标志。这种方式不但缩减了虚拟函数表的大小，而且还确保正确版本的控制。

使用 C++编写类，你可以使用存取修饰符为类成员设置不同的访问等级。C#同样支持 **private**、**protected** 和 **public** 三种存取权限，而且还增加了第 4 种：**internal**。

你创建了多少个类是从多重基类派生出来的。在大多数情况下，只要从一个类派生便可。多重基类惹出的麻烦通常比它们解决的问题还多，这就是为什么 C#只允许一个基类的原因。如果你觉得需要多重继承，可以运用接口。

一个可能出现的问题：在 C#中不存在指针，如何模仿它？这个问题的答案很有代表性，它提供了对 ngws runtime 事件模式的支持。

1.4　类型安全

在 C++中拥有一个指针，能自由地把它强制转换成为任何类型，包括做出诸如把一个 int*（整型指针）强制转换成一个 double *（双精度指针）这样的傻事。只要内存支持这种操作，它就能工作。这并不是你所想象的企业级编程语言的类型安全。

原则性的问题，C#实施最严格的类型安全，以保护自己及垃圾回收站（garbage collector）。因此必须遵守 C#中一些有关变量的规则：

- **不能使用没有初始化的变量**　对于对象的成员变量，编译器负责清零。而局部变量，则由你负责清零。当你使用一个没有初始化的变量时，编译器会教你怎么做。这样做的优点是能够避免由于使用不经初始化的变量计算结果而导致的错误，而你还不知道这些奇怪的结果是如何产生的。
- **C#取消了不安全的类型转换**　不能把一个整型强制转换成一个引用类型（如对象），而当向下转换时，C#验证这种转换是正确的。也就是说，派生类真的是从向下转换的那个类派生出来的。
- **边界检查是 C#的一部分**　再也不会出现这种情况：当数组实际只定义了 n-1 个元素，却超额地使用了 n 个元素。
- **算术运算有可能溢出结果数据类型的范围**　C#允许在语句级或应用程序级检测这些运算。在允许检查溢出的情况下，当溢出发生时将会引发一个异常。

➘ **被传递的引用参数是类型安全的**

1.5　版本可控

在过去的几年中，几乎所有的程序员都至少有一次不得不涉及到众所周知的“dll 地狱”。该问题起因于多个应用程序都安装了相同 dll 名字的不同版本。有时，老版本的应用程序可以很好地和新版本 dll 一起工作，但是更多时候它们会中断运行。现在的版本问题真是令人头痛。

ngws runtime 将对应用程序提供版本支持。C#可以最好地支持版本控制。尽管 C#不能确保正确的版本控制，但是它可以为程序员保证版本控制成为可能。有了这种支持，一个开发人员就可以确保当他的类库升级时，仍保留着对已存在的客户应用程序的二进制兼容。

1.6　兼容

C#并没有存在于一个封闭的世界中。它允许使用最先进的 ngws 通用语言规范（Common Language Specification，简写为 CLS）访问不同的 API。CLS 规定了一个标准，使符合这种标准的语言能够协同工作。为了加强 CLS 的编译，C#编译器检测所有的公共出口编译，并在通不过时列出错误。

当然，如果你想访问旧一点的 com 对象。ngws runtime 提供对 com 透明的访问。

OLE 自动化是一种特殊的产物。任何使用 C++创建 OLE 自动化项目的程序员已经喜欢上各种各样的自动化数据类型。有个好消息就是：C#支持它们，而且没有烦琐的细节。

最后，C#允许你用 C 原型的 API 进行协作。可以从你的应用程序访问任何 dll 中的入口点（有 C 的原型）。用于访问原始 API 的功能称作平台调用服务（platform invocation services，缩写 pinvoke）。

1.7　灵活

你可能会问：“难道就没有我要传递指针的 API 吗?” 很正确，不是只有少数的这种 API，而是有很多（有点保守的估计）。这种对原始 win32 代码的访问有时导致对非安全类特定指针的使用（尽管它们中的一些由于受 com 和 pinvoke 的支持可以解决）。

尽管 C#代码的缺省状态是类型安全的，但是你可以声明一些类或者仅声明类的方法是非安全类型的。这样的声明允许你使用指针、结构，静态地分配数组。安全码和非安全码都运行在同一个管理空间，这就暗示着当从安全码调用非安全码时不会陷入列集（marshaling）。

第 2 章　简单 C#程序

本章将通过介绍一些非常简单的 C#程序，让你了解 C#的基本特点，通过本章学习，可以达到以下目的：

- 理解的 C#语言的基本结构
- 开始了解“名字空间”的概念
- 开始了解“类”的概念
- 知道 Main 方法的作用
- 知道如何获得命令行输入
- 知道 I/O 控制台

例程 2.1　一个简单的欢迎程序: Welcome.cs

```
// 声明名称空间
using System;
// 程序开始类
class WelcomeCSS {

// 主程序
      public static void Main() {
// 向控制台输出
      Console.WriteLine("Welcome to the C# Station Tutorial!");
        }
}
```

在例程 2.1 中有 4 个主要元素：

- 一个名称空间的声明
- 一个类
- 一个 Main 方法
- 一条程序语句

该程序可以用下面的命令编译：

```
csc Welcome.cs
```

这个命令将产生一个名为 Welcome.exe 的可执行文件。其他程序也可以采用类似方法进行编译，只要用它们的文件名替换 Welcome.cs 即可。如果需要了解更多有关该命令的帮助信息，请输入：

```
csc-help
```

名称空间的声明表示该程序引用 System 名称空间。名称空间包含一组可以被 C#程序调用的代码。使用 using System 的声明，程序就可以引用名称空间 System 中的代码。有关名称空间的具体内容将在后面的章节中详细阐述。

类声明 class WelcomeCSS，包含了程序运行所需要的数据和方法的定义。这个特殊的类中没有数据，只有一个方法。方法定义了类的行为，或者说类所能做的事情。

WelcomeCSS 中的这个方法阐述了该类在执行时将会进行哪些动作。Main 方法是程序的起点。在 Main 之前是一个 static。static 说明该方法只为这个类服务，而不是该类的某个对象实例。这是很有必要的，因为当一个程序开始后，是不存在任何对象实例的。有关类、对象和实例的知识将在后面的课程中进一步阐述。

所有方法都必须有一个返回类型。在这个例子中返回类型是 void，意思是 Main 将不返回任何值。每一个方法还有一个参数列表，其中的零个或多个参数用括号刮起来。出于简单考虑，在这个例子中，我们没有给 Main 提供任何参数。在后面的课程中，我们将阐述 Main 方法参数的种类。

Main 中的"Console.WriteLine(...)"语句规定了它的行为，Console 是 System 名称空间中的一个类，"WriteLine(...)"则是 Console 类中的一个方法。二者之间用点“.”分隔。当然，我们完全可以将上述的语句写成 System.Console.WriteLine(...)。这种是 C#程序语句的完全书写规则：namespace.class.method

既然我们已经在程序的开始处声明了 using System，就没有必要在程序中使用 System.Console.WriteLine(...)的书写方式。这条语句的作用是将字符串 Welcome to the C# Station Tutorial!显示在控制台屏幕上。

程序中的注释采用“//”标注起来。这种是单行注释，也就是说注释内容到本行结束为止。如果你需要扩展到多行注释，只要将注释内容用“/*”和“*/”括起来即可。在这两个符号之间的任何内容都被认为是注释语句，程序编译时，编译器将忽略它们。

所有的语句都用分号“;”作为结束符。类和方法包含在左右花括号{ }之间。{ }之间的所有语句构成块。块则定义了程序元素的范围（或者说是寿命和可见性），有关内容将在后面课程中阐述。

很多程序都支持命令行输入。接受命令行输入是在 Main 方法中。例程 2.2 中的程序可以从命令行接受姓名并将其显示在控制台屏幕上。

例程 2.2　获取命令行输入：NamedWelcome.cs

```
// 声明名称空间
   using System;
// 程序开始类
   class NamedWelcome {
//主程序
    public static void Main(string[] args) {
// 向控制台输出
        Console.WriteLine("Hello, {0}!", args[0]);
        Console.WriteLine("Welcome to the C# Station Tutorial!");
    }
}
```

运行时别忘了在命令行输入你的姓名，例如：NamedWelcome Joe。否则这个程序将崩溃，后面的课程将说明如何检测和避免这种错误。

在例程 2.2 中的 Main 方法中有一个参数列表入口。参数名为 args，参数类型是 string []。string 类型表示其中的参数必须是字符，比如单个单词或者多个单词。方括号 [] 则定义了

一个列表。也就是说，agrs 参数类型是一个从命令行输入的单词列表。

在例程 2.2 中有一条和 2.1 中不同的语句 Console.WriteLine(...)，其中有一条嵌入了{0}的格式化字符串。格式化字符串的第 1 个参数起始于数字 0，第 2 个是 1，以此类推。{0}表示下一个跟随结束引用的参数决定了下一步的操作。例程 2.2 中下一个跟随结束引用的参数是 arg[0]，表示 args 列表中第 1 个字符串。列表中第 1 个元素用数字 0 表示，第 2 个用数字 1 表示，依此类推。例如，如果我们在命令行输入 NameWelcome Joe，那么，arg[0]的值就是 Joe。我们现在再来看看在格式化字符串中嵌入的{0}参数。既然 agrs[0]是第一个参数，它的值将在 Console.WriteLine()语句之后被替换成格式化字符串的第 1 个嵌入参数。当这个命令执行时，agrs[0]的值：Joe 将替换格式化字符串中的{0}。在命令行 NamedWelcome Joe 执行后，输出结果如下：

```
>Hello, Joe!
>Welcome to the C# Station Tutorial!
```

另外一个给程序提供输入的方法就是通过控制台，例程 2.3 显示了如何从用户那里得到交互式输入。

例程 2.3　获取交互式输入：InteractiveWelcome.cs

```
// 声明名称空间
using System;

// 程序开始类
class NamedWelcome {

// 主程序.
 public static void Main() {

// 向控制台输入/获取输入
        Console.Write("What is your name?: ");
        Console.Write("Hello, {0}! ", Console.ReadLine());
        Console.WriteLine("Welcome to the C# Station Tutorial!");
    }
}
```

这次 Main 方法未带任何参数。不过方法中有 3 个语句，其中前面两条和第 3 条不一样。前面两条是 Console.Write(...)，而最后一条是 Console.WriteLine(...)。它们的区别在于 Console.Write(...)向控制台写完数据后停在同一行，而 Console.WriteLine(...)写完后则停在下一行。

第 1 条语句仅仅是向控制台输出 What is your name?。第 2 条语句在参数没有赋值之前不会输出任何数据。在格式化字符串后的参数是 Console.ReadLine()，它让程序等待用户从控制台的输入，这个输入以 Return 和 Enter 结束。这个方法的返回值将替换格式化字符串中的{0}参数，并写回控制台。

最后一条语句则是将数据输出到控制台上。在命令行敲入 InteractiveWelcome 后，程序的输出结果如下：

```
>What is your Name?  <在这里输入你的名字>
>Hello, <名字>!  Welcome to the C# Station Tutorial!
```

2.4　小结

到目前为止，你已经了解了 C#程序的基本结构，熟悉了名称空间和类的概念，知道了 Main 方法是 C#程序的入口，同时还学习了如何从命令行得到输入以及如何执行交互式 I/O。这个仅仅是个开始，下一章将阐述：表达式、类型和变量的概念。