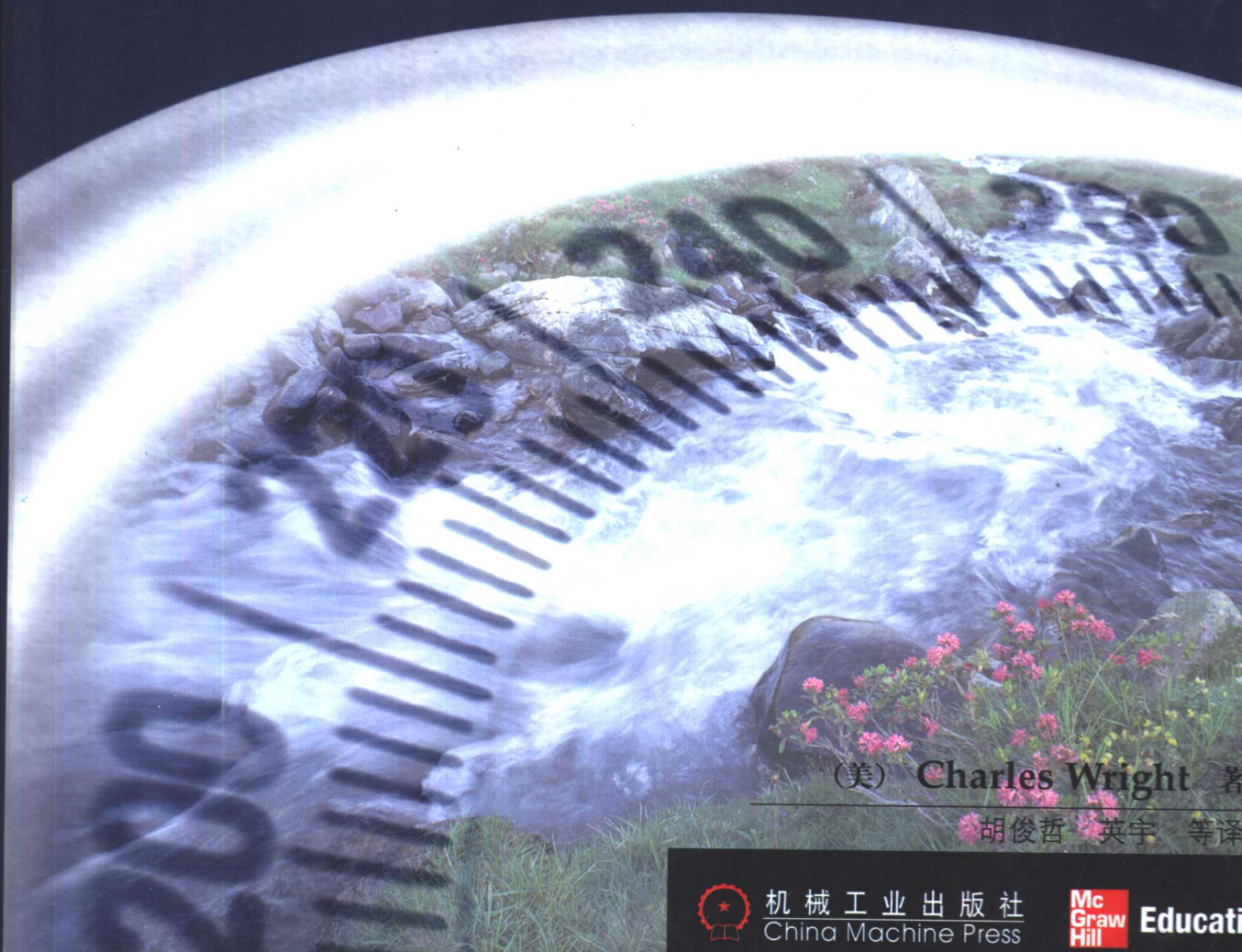
HZ BOOKS

C# Tips & Techniques

开发人员专业技术丛

C#

编程技术与技巧



(美) Charles Wright 著

胡俊哲 英宇 等译

机械工业出版社
China Machine Press

McGraw Hill Educati

开发人员专业技术丛书

C#编程技术与技巧

（美） Charles Wright 著

胡俊哲 英宇 等译

机械工业出版社
China Machine Press

本书全面系统地讲述了C# 编程语言，例如C#的数据类型、结构和类、流操作、异常处理、封装、继承和多态等。介绍了Visual Studio .NET集成开发环境以及如何使用C#语言来编写Windows应用程序，书中还提供了大量的代码实例、编程技巧以及中文屏幕图，更利于快速理解和掌握C#语言。

本书通俗易懂，叙述深入浅出，概念清晰、准确，实例丰富。非常适合有一定C/C++或者Visual Basic编程基础的初学者自学和参考，也可以作为专门的C#程序员培训教材。

本书版权登记号：图字：01-2002-3608

图书在版编目（CIP）数据

C#编程技术与技巧 /（美）莱特（Wright, C.）著；胡俊哲等译. -北京：机械工业出版社，2002.9

（开发人员专业技术丛书）

书名原文：C# Tips & Techniques

ISBN 7-111-10826-4

Ⅰ. C… Ⅱ. ①莱… ②胡… Ⅲ. C语言－程序设计 Ⅳ. TP312

中国版本图书馆CIP数据核字（2002）第062688号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑：梁 莉 张鸿斌

北京忠信诚胶印厂印刷·新华书店北京发行所发行

2002年9月第1版第1次印刷

787mm × 1092mm 1/16 · 32.75印张

印数：0 001-4 000册

定价：49.00元

译 者 序

无论你是一位经验丰富的编程高手，还是刚入门的初学者，阅读完本书，相信C#都会在每位读者的脑海里留下深刻的印象。

本书从最基本的概念开始，系统地介绍了C#语言和.NET编程环境，以及如何使用C#语言来快速开发Windows应用程序。书中各章节的内容按照由浅入深的顺序进行组织，而且每一章节都涉及了C#语言的某个方面，读者能够从基础的编程知识开始学习，直到最终领会C#语言的高级特性。

除了讲述C#语言本身的内容之外，书中还详细介绍了如何在Visual Studio .NET集成开发环境中使用C#语言编写Windows应用程序。在每一章节中还给出了许多非常实用的编程技巧和在编程过程中需要注意的地方。

另外，书中还提供了大量C#程序范例，并对程序代码有详细的解释，以便让读者很轻松地了解到C#语言的语法、技术特性等内容，并且能够把这些知识轻松地应用到自己的程序中。

从本书中，读者可以学到以下内容：

- C#语言和.NET开发环境。
- C#语言的语法。
- C#语言的数据类型。
- 面向对象编程思想如何在C#语言中体现。
- C#语言如何处理程序异常。
- C#语言的高级特性：数组、I/O操作、反射、配件等。
- 如何调试C#程序。
- 如何在C#程序中创建窗口界面。
- 在C#程序中响应用户事件。

学完本书的所有内容之后，读者就可以从对C#一无所知而成为一个能够灵活使用C#语言完成编程任务的程序员。在此基础上，甚至可以参与大型应用项目的编程工作了。如果你正需要了解C#语言——微软公司在21世纪力推的主流编程语言，那么本书将可以让你充分领略到C#的神奇魅力。

本书的翻译主要由胡俊哲、英宇、贾楠完成。鉴于译者水平，如有错误和疏漏之处，请读者不吝指教。我们接收读者意见的邮箱为：yingyu@263.net。

2002年7月

前　言

几年来，Microsoft一直致力于一种称做.NET的新开发平台。设计这种新的开发平台是使开发人员能够利用网络环境编写基于组件的程序。Microsoft这种新的编程语言——C#，是由C、C++及Visual Basic语言派生出来的，运行于.NET环境中。

尽管在Internet上可以看到许多声明，但C#并不是一种变革性的语言。Microsoft 将最新的技术（如Java）和以往的技术（如Smalltalk）糅合在一起形成了一种混合语言，它的语法非常接近C++。

这是否意味着使用C#语言的时代到来了呢？那还要拭目以待。当然，C#或者说.NET环境在10年前甚至5年前是不可能有的。Microsoft认为，过些年后，技术上的进步（比如更快速的计算机和更便宜的存储器）将会使那些用于开发中间语言产品的语言比过去更具生命力。

C#的最大受益人将是开发者。使用这种语言将减少编程错误，加速开发周期。如果你熟悉C++，那么学习C#就会很容易。使用C#会缩短从编程思想到出产品的过程。

本书内容

任何单独的一本关于使用C语言在Windows环境下编程的书似乎都不完整。在一本书中要涵盖所有内容是不可能的。本书并非试图包含C#编程领域中的所有内容，而是要向你展示一些使用C#的技巧。同时，本书包含了C#的语法和原理，为使用C#编程打下基础。

本书共有18章。你将首先了解.NET的结构和C#语言，然后再学习面向对象的编程以及怎样使用C#先进的编程概念。

第1章：C#和.NET简介

开始的这一章概述了.NET框架和C#语言，还会提到许多在开发C#程序时会用到的命令行工具。

第2章：在Visual Studio.NET中开发C#应用程序

在这一章里，将会看到一个全新的Visual Studio .NET集成环境。将学习使用Visual Studio创建C#程序，并会发现开发环境中的向导和工具窗口非常有用。

第3章：C#的预备知识

学习一种新语言的惟一途径是使用这种语言进行编程。该章阐述了C#的各部分，并将它们与不同语言进行了比较，教你用C#窗体和命令行程序创建一个显示日期和时间的Windows窗体项目。

第4章：C#基础

该章介绍C#的语法、操作符和符号。你会看到“安全”和“不安全”代码，学习怎样使用语言来测试一个变量取值范围。你还将学习控制和循环语句，了解其概貌。

第5章：C#中的数据类型

C#是一种安全类语言，理解C#怎样处理数据是很重要的。该章讨论对象、引用、变量的数值类型。你会用到只读变量、常量，并了解C#处理字符串的方法。

第6章：C#中的结构

结构是一种最老的对象，它在面向对象的编程概念出现之前就已为人所用了，在C#中它仍占有一席之地。在该章中，你会学到怎样声明、使用结构，以及在结构中使用方法和属性。

第7章：理解C#类

和结构一样，类是C#的基本编程单元。你将学习怎样定义类，怎样使用实例和静态成员。你还会看到构造函数和析构函数，以及怎样在类中包含成员方法。

第8章：C#语言和面向对象编程

一般编程者选择一种基于对象的语言主要考虑它的三个特性：封装性、继承性和多态性。C#用各种各样的对象类型实现这些特性。该章还将讲到抽象类和对象浏览器。

第9章：异常处理

在程序中发生不期望的事件和错误时就会产生异常。如果在代码中不进行处理，程序就会异常中断。C#在异常处理上非常迅捷，它通常是抛出异常，而不是像其他语言那样用一个方法简单地返回错误代码。

第10章：C#高级特性

C#是一种现代语言，它实现了一些先进的、现代的编程技巧。该章中讲到了名称空间、配件、引用和接口，还有反射——一种使程序具有探测对象、发现对象相关信息能力的概念。

第11章：使用C#数组

C#将Visual Basic的数组技术带入了C系列编程语言。该章讲述C#如何管理数组，如何声明、创建一维和多维数组，以及元素参差不齐的不规则数组（其元素是数组）。

第12章：文件操作

编写任何实用的程序最终是要读写文件的。C#使用“流”来实现文件的操作。你会看到C#的流如何在文件、存储器、网络中处理数据，你还会遇到一些内置类并学会用它们操纵文件。

第13章：编写Windows应用程序

使用Visual C#编写Windows程序涉及到窗体。窗体大体上等同于对话框。Visual C#赋予了窗体可扩展能力。因此，在Visual C++中编程利用的View类，在C#中是找不到的。

第14章：调试C#程序

无论如何，用C#写的大多数实用程序都会有问题。任何程序开发中，有部分阶段就是反复调试。该章教你程序调试工具的用法、内置类的检查方法。这些类是专门用来帮你查找、修改程序中的错误的。

第15章：创建用户界面

如何与用户交互信息，决定了你的程序是否易用——是让用户轻松使用，还是令用户抱怨操作繁琐。该章阐述了怎样在程序中添加菜单（包括提示菜单）和工具条。还讲到如何使用图像列表（控件）来放置工具条上的图标，如何让控件响应窗体大小的改变，如何构造一个能完成记事簿中大多数任务的编辑器项目。你将开始创建一个编辑器工程，该编辑器能够完成Nitepad的大部分功能。

第16章：使用Windows控件

窗体是Windows控件方便的容器。控件是给用户显示信息并接受用户信息和行为的对象。该章讲到两个浏览控件：ListView 和 TreeView，并讲述如何在窗体上使用它们。

第17章：使用公用对话框

Windows公用对话框库中给出了一个通用的“外观和感觉”，这缩短了用户学习使用新程序的时间。该章介绍在C#中可用的通用对话框对象，并介绍如何在第15章中讲到的编辑器项目中添加打印功能。

第18章：使用代理和事件

控件和窗体之间是用“事件”进行相互联系的。这和用Visual C++对窗口消息编程相类似。要处理一个事件，需要对事件赋予代理（delegate）。事件和代理不仅仅用于简单的控件消息，你会看到更多的用法以及怎样运用事件和代理。

如何阅读本书

本书是为帮助一般用Visual C++或Visual Basic的编程者转到用C#编程而写的。为了方便起见，书中有的地方会指出C#和其他语言的相同点和不同点。

本书决不是有关C#的详细信息罗列。没有任何一本有关C语言编程的书是全面完整的，甚至Microsoft Developers Network 上的Visual Studio帮助文件中都有许多空白。编程是个经常变化的领域，大多数语言都是动态变化的。

C#不是一门能够轻松地分门别类进行讲述的语言。要描述一个概念，往往要涉及其他概念。尽管本书各章节的内容都在前述章节的基础上循序渐进，但偶尔有些主题也要借鉴后面的章节来理解，这时会有上下文提示这些主题内容的出处。

尽管本书中有一些代码是节选的，但大部分代码都是完整的程序。有时程序可能很小，仅

仅只是说明一个要点，但仍然可以编译运行，也就是说，该程序是可以用来验证所述要点的起点。我是赞成用完整的程序做范例的。在25年的编程生涯中，我经常遇到这样的情况：一些范例中的代码因为缺少文中提到的个别组件而无法编译。一个完整的程序如果没有测试成功，总能返回到源代码测试其他问题。第18章中的员工排序实例的思想来源于去掉注释后排序部分程序的范例。在这个范例里，将示范如何创建、使用代理方法来排序记录。这个程序是完整的，可以用C#目前的版本进行编译，功能就是排序记录。

本书中的代码着力于引导你如何在C#中实现功能和构造。你应该尽力去理解使用方法而不要误认为那些对象必须以某种特定方式使用。我坚持认为编程与其说是一门科学，不如说是一门艺术，做不同的尝试，会有“柳暗花明”般茅塞顿开的感觉。想像力比任何简单的“如何做”范例都重要。

因此，我从来没有建议什么“合适的代码”。把一个程序从一种理念变成一种现实总是包含周密的设想，过分强调所谓“合适的代码”往往带来的是缺乏想像力的代码。你知道怎样编程，并不意味着你就能写出设计完善、效率高的程序来。写出这样的程序需要经验，具有这种经验就需要尝试新技术且难免犯错误。

使用本书的相关网址

在本书的各个章节中，我们给出了多个C#程序的范例。在许多情况下，你都可以简单地将我们提供的注释和部分代码剪贴到你的程序中去。本书中的代码从McGraw-Hill/Osborne的网址www.osborne.com上可以找到。

本书原书书名：C# Tips & Techniques
本书原书书号：ISBN 0-07-219379-4
本书原书网址：www.osborne.com

目　录

第三部分 用C#进行Windows编程

第一部分 .NET环境和C

第1章 C#和.NET简介

本章内容包括：

- 何时使用C#和C++
- .NET和其他开发环境的区别
- 使用公共语言运行时
- 查看中间语言代码
- 利用实时调试
- 利用.NET与COM的互操作特性
- 通过.NET的版本控制来处理软件更新。
- 使用.NET的反射机制来获取类的相关信息

20多年以来，C语言一直是为小型计算机系统开发应用程序的主要计算机语言。它是20世纪70年代Dennis Ritchie为UNIX操作系统开发的一种程序设计语言。C语言不仅向程序员提供了很多类似汇编语言的控制计算机硬件的能力，而且也具有FORTRAN等高级语言编程容易的优点。

在过去的10多年里，很多开发工作是使用Bjarne Stroustrup开发的C++语言进行的。C++语言在保留C语言的基本和低级特点的基础上，把C语言变成了一种现代的、面向对象的语言。现在很多人提及C编程的时候，他们实际上是指C++编程。

C语言和C++语言有很多相似性。其中一点就是两种语言都有一个很长的学习曲线。即使不是很深入地学习，依然需要若干年才能真正明白C++语言的特点和微妙之处，才能利用其强大的编程能力来开发大型的应用程序。

在过去的几年里，Microsoft引入并发展了Visual Basic。很快，Visual Basic成为需要为特定目的而快速开发程序的程序员的新宠。Visual Basic大致建立在BASIC语言的语法和表达的基础之上，并提供了一种比C++语言学习曲线短的编译语言。Visual Basic for Applications（VBA）往往被用来和其他应用程序一起为用户提供一种编写宏的途径。

从一开始，C#（发音为“C Sharp”）的开发者就试图开发一种结合了“Visual Basic的快速开发能力和C++的高性能”的语言。这听起来有些自相矛盾。实际上两者往往互为折中，“快速开发能力”的出现往往以牺牲“高效率”为代价。C#也不例外。你可以体会到“快速开发”，但是必须付出“性能”的代价。为了获得“C++的高性能”，程序员将不得不放弃一些C#的特色，比如：允许在C#内嵌入C++的代码，并且调用C++编写的库模块。如果程序员大量放弃使用C#的特色，那么他的应用程序就更适合用C++书写，而不是C#。

很多开发者用“现代”这个词来描述C#语言，但是在很多方面C#语言回到了为小型计算机系

统开发程序的早期阶段。C#松散地从C++语言继承而来，是一种面向对象的、“类型安全”的语言。虽然C#的很多语法和关键词为C++的程序员所熟悉，但C#并不是C++语言的优化（refinement）。

C#是Microsoft的Visual Studio 7.0开发环境支持的一种语言。Visual Studio 7.0开发环境还支持Visual Basic、Visual C++、VBScript 和JScript。C#，以及Visual Basic和“托管”C++的当前版本，是基于下一代Windows服务平台的（NGWS），NGWS被称为.NET软件开发环境。

NGWS的“服务”部分为你提供了解这种语言核心的线索。C#是“面向组件”的。.NET环境本身是面向组件的，所以C#的根本设计目的就是使它更加容易编写组件。在过去的几年里，我们已经接触了不少面向组件的工具，比如组件对象模型COM和Active X。C#和.NET环境简化了组件的编写。你不必用IDL（接口定义语言）文件来产生组件，也不必产生类库就可以在程序中使用组件。当你用C#创建一个组件的时候，组件本身包含了需要用来描述自己的所有信息——元数据。

沿着这个思路，你将会看到很多经常和组件一起使用的术语被用来描述C#的概念。你会发现，我们经常使用方法，而不是函数；经常使用属性、域，而不是变量。

.NET框架定义了一种公共语言规范（你也许有时候看到另外一种表达：公共语言子集），用它来提供公共语言运行时（CLR）。支持.NET框架的程序被编译成中间代码模块，并且由CLR为计算机翻译成原始语言。

和伪代码（就是我们都讨厌的P代码）不一样，.NET环境中的中间语言（IL）不仅仅充当翻译员的角色。当你运行一个被编译成IL的程序的时候，.NET框架就会把中间代码重新编译成原始代码。

初识C#

C#的一个最大的优点就是，花了很多时间和工作学会了C++语言的程序员不必丢弃以前的知识，就可以开始使用这种新的语言开发程序。虽然有一些新的概念、技术和一些新的函数名需要学习，但一般而言它的语法是和C++类似的。

而且，程序员不必丢弃他们已经编写好的代码。C#具有调用已有代码和系统库中的库函数的机制。你可以调用C++中使用的相同Windows函数，并且在很多情况下，你可以在Windows函数和.NET框架之间进行选择。比如，下面的这个小程序MsgBox.cs显示了两个消息框，其中一个在另一个的后面。第一个消息框是通过调用Windows的API函数MessageBox( )生成的，第二个是通过调用CLR的MessageBox类的Show( )方法生成的。你很难区分这两个消息框。（在最终的分析中，这两个消息框都是通过调用Windows的函数产生的。）

```
//
// MsgBox.cs -- demonstrate using the Windows API message box and the
//              .NET framework message box.
//
//               Compile this program with the following command line:
//                   C:>csc msgbox.cs
//
namespace nsMsgBox
```

```
{
    using System.Windows.Forms;
    using System.Runtime.InteropServices;
    class clsMain
    {
// Import the dll and prototype the MessageBox function.
        [DllImport("User32.dll")]
        public static extern int MessageBox (int hwnd, string Message,
                                             string Caption, int Flags);
        static public void Main ()
        {
// Call the Windows API function
        MessageBox (0, "Hello C# World", "Howdy",
                    (int) MessageBoxButtons.OK
                    |(int) MessageBoxIcon.Exclamation);
// Call the CLR method
        System.Windows.Forms.MessageBox.Show ("Hello, C# World!",
                    "Howdy",
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        }
    }
}
```

虽然C#是一种新的语言，但是很难在这种语言中找到新的概念。它在风格上很大程度上受了Sun Microsystems开发的Java的影响，并且基于Visual Basic的编程模型，大量地借用了C和C++的语法和关键词。你也能在C#中发现很多Smalltalk程序设计语言的概念。然而这并不奇怪，因为C++本身就和SmallTalk具有很多相似性。

Java语言试图提供给开发者一个与平台无关的语言。通过Java虚拟机（JVM）的中间语言解释器，Java的目标是提供一种“编写一次，随处运行”的语言。这将避免开发者为不同计算机平台、不同操作系统编写不同的程序。开发者只需要使用相同的代码，JVM将提供平台和操作系统之间的接口。

Microsoft试图通过C#和.NET框架达到同一目标。C#之类的语言使用.NET环境把代码编译成IL，而不是编译成在Intel处理器平台上运行Windows操作系统的计算机的原始代码。然后CLR可以动态地把这些IL代码翻译成适合某种计算机平台的原始代码。至少从理论上讲，一个操作系统只需要执行CLR，就能够让程序员运行他们的程序。

这一点和Java的概念惊人地相似，这也使很多程序开发界的人士认为“C#只不过是Java的另一个名字而已”。C#是不是披着Microsoft伪装的Sun并不重要，重要的是.NET平台。在.NET编程中提供的技术超过了JVM。

虽然C#乍看起来是基于Visual Basic的编程模型，但是它并没有完全遵从这一模型。事实上，C#从C语言家族和Visual Basic中都提取了不少概念，从而达到了某种折中。Visual Basic下的开发很可能比C语言下的开发要好一些。为了和.NET 协调工作，Visual Basic放弃了一些表达和关键字，比如：Option Base这个关键字和Variant这个数据类型。

图1-1表明了C#的开发者是如何使用不同操作系统、编程语言、组件和分布式环境中可用的关键特征来构筑这个新语言的。

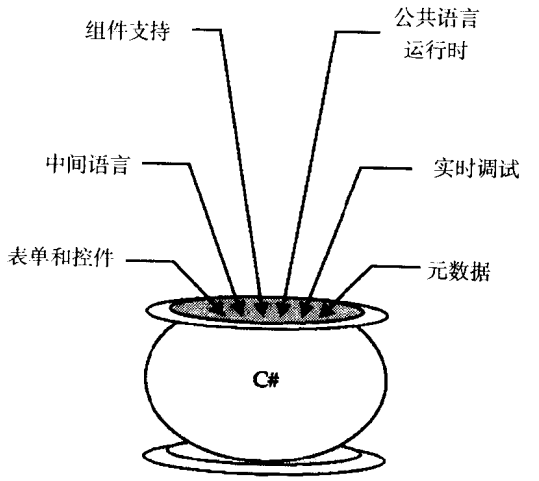


图1-1　C#是融合了很多编程技术而诞生的

使用C#来代替C++

如果你刚开始学习编程的话，那么你一定很乐于得知：作为入门学习的语言，C#比C++更容易学习。虽然C和C++语言是功能强大的语言，但是它们确实让你执行了不应该执行的操作。

C#是一种类型安全的语言，它隐藏了很多初学者在学习C或者C++中会遇到的潜在问题。比如，C#允许使用指针，但是程序员不必直接和指针打交道。C#语言把指针隐藏在名为引用类型的变量中。

C#使用很多与C++相同的语法，如果学习了C#，再学C++就可以节省很多时间。已经熟悉C++语言的程序员就必须克服他们对这种语言的偏见(C++的程序员往往会轻视Visual Basic编程，而相对C++，C#更接近于Visual Basic)。也许很难习惯C#对不同操作符和表达的一些限制，但是这些限制是用来减少错误并促进快速开发的。

在这本书中，我将经常回顾使用C++的编程方式，并将它和C#的编程方法进行对比。一般而言，改变C++方法将会降低一些表达和操作的“效率”，但是也将促进操作的简化。例如，在一整天的C++编程以后，你或许会写出下面的代码：

```
#include <stdio.h>

void Function (double *);

void main (void)
{
    int arr [10];
    Function ((double *) arr);
```

```
}
void Function (double *arr)
{
    for (int x = 0; x < 10, ++x)
        arr[x] = x * 3.14159;
}
```

从句法上讲，这个程序是完全正确的，编译器也不会发出警告。然而，它必定会产生一种破坏。把10个双精度的值写入一个包含10个整型值的数组将会导致堆栈中的一些数据被覆盖。这就是用C++编程的威力，同时也是它的一种潜在危险。

在这样的例子中，C#的编译器就可以避免破坏的发生。它不允许你做类似的事情。C#中类似的代码如下所示：

```
class clsMain
{
    static public void Main ()
    {
        int [] arr = new int [10];
        Function ((double []) arr);
    }
    static public void Function (double [] arr)
    {
        for (int x = 0; x < 10; ++x)
            arr[x] = x * 3.14159;
    }
}
```

不论如何转换变量，你也无法使C#的编译器允许你执行这样的操作。这也体现了C#的类型安全机制。这种语言保证了在你使用某个变量之前，某种类型的变量总是含有一个这种类型的值。

这并不是说编译正确就不会出错。你还是有可能发现你的程序进入了无限循环等类似的情形，但是发生这些事件的可能性大大降低了。同时，C#允许只能通过指针（参照类型变量）来使用大多数对象的事实，也会导致其自身的一系列不足和错误。

所有的这些会缩短调试周期，这也就意味着更短的开发时间。通过使用C#，内部的开发者和在软件公司工作的程序员都可以拥有更高的效率。业余程序员或编程迷也希望更快地看到程序的结果。

C#的代码也和其他支持.NET框架的语言兼容。比如，你将发现与.NET版本的Visual Basic以及其他编程语言编写的程序和控件进行交互将会变得更加容易。

1.1 何时使用C#和C++

早在20世纪70年代的时候，我就在一所大学的数学系学习编程，那时大多数学校还没有设立计算机系。我们经常在计算机（相比现在的PC而言，我们的计算机简直是一个又大又笨的怪物）上做一种有趣的游戏。我们随机地在边长为1的正方形内放入一些点，然后计算每个点与正方形左下角（正方形的原点）之间的距离。我们知道正方形对角线之间的距离是$\sqrt{2}$，那么这些