国外经典教材

PEARSON Addison Wesley

C# by Dissection

C#解析教程

THE ESSENTIALS OF C# PROGRAMMING 层层解析，揭示C#语言的精髓

[美] Ira Pohl 著 周靖译

Pearson Education 清华大学出版社

国外经典教材

C# 解析教程

[美] Ira Pohl 著
周 靖 译

清华大学出版社
北 京

内 容 简 介

本书全面剖析了如何在 Microsoft .NET 平台上用 C# 快速生成可运行的应用程序。书中采用作者独创的解析法（类似于代码的结构化走查），着重讲解了 C# 语言中的最新编程元素和习惯用语。通过本书的阅读，程序员可更透彻地理解代码，养成良好的编程习惯，避免常见的编程错误。

本书适合刚入门的程序员、编程爱好者、具有其他语言编程经验但打算换用 C#的中高级程序员阅读和参考。

EISBN: 0-201-87667-1
C# by Dissection
Ira Pohl

北京市版权局著作权合同登记号：图字 01-2003-0852 号

图书在版编目（CIP）数据

C# 解析教程/（美）波尔著；周靖译. —北京：清华大学出版社，2003
（国外经典教材）
书名原文：C# by Dissection
ISBN 7-302-06602-7
Ⅰ.C... Ⅱ.①波...②周... Ⅲ.C 语言—程序设计—教材 Ⅳ.TP312
中国版本图书馆 CIP 数据核字（2003）第 032615 号

出 版 者：清华大学出版社（北京清华大学学研大厦，邮编 100084）
http://www.tup.com.cn
http://www.tup.tsinghua.edu.cn
责任编辑：文开琪
印 刷 者：北京市清华园胶印厂
发 行 者：新华书店总店北京发行所
开　　本：787×1092 1/16　**印张**：24.5　**字数**：588 千字
版　　次：2003 年 6 月第 1 版　2003 年 6 月第 1 次印刷
书　　号：ISBN 7-302-06602-7/TP・4943
印　　数：0001～5000
定　　价：39.00 元

前　言

C#是一种重要的新语言，微软将其设计和定位成.NET 战略的一部分。C#内建了很多有用的库，并由高级集成开发环境提供支持。它有效地支持目前占主导地位的编程方法——面向对象编程（Object-oriented programming，OOP）。

本书提供了我们精心开发并编写的、能实际运行的代码，用以解释 C#语言的关键特性，同时，还对编程过程进行了全面介绍。书中采用浅显易懂的方式，对程序代码进行了详细解释。C#问世于 20 世纪 90 年代，是继 C 语言之后的一种功能强大的现代语言。它在 C 的基础上，增加了“类”的概念。类是提供用户自定义类型的一种机制，这些用户自定义类型也称为**抽象数据类型**（Abstract data type，ADT）。C#通过这些方式和提供继承及运行时类型绑定，来支持面向对象编程。

解析法

本书通过精心开发能实际运行的代码，利用“解析法”来解释 C#程序语言的重要特性，向读者清晰、全面地介绍编程过程。解析法是一种独特的教学手段，它是作者在 1984 年为解释能实际运行的代码的重要特性而开发的。解析法类似于代码的结构化走查（walk-through），目的是向读者解释在代码中出现的最新编程元素和用语。程序和功能的讲解分步骤进行，关键概念则通过在不同背景中的运用进行全面巩固。

无需任何背景知识

本书不要求读者具有编程背景，学生和初级计算机用户均可使用。对于有经验的程序员，如果不熟悉 C#，也可受益于本书对 C#语言的详细讲解。作为教材，本书可供计算机科学或编程专业一年级学生使用。

本书适合计算机系一年级课程或其他学科的编程入门课程。每章提供大量经过精心讲解的程序，可指导学生从整体上提高编程技能。本书一开始，就介绍了完整程序以及如何编写函数（函数是结构化编程中的重要概念）。函数之于程序，犹如段落之于短文。编写函数的能力是衡量专业程序员水平的一项重要指标，因此本书将其列为重点。书中提供了大量难度不一的示例和练习题，以便教师根据学生的具体情况，挑选相应的内容。

本书特色

- 软件工程知识的描述贯穿全书。
- 每章都提供 Pohl **博士提醒**和简洁的编程提示。
- 提前解释简单的递归，以满足计算机科学课程的需要。

- 全面覆盖程序无错性和类型安全性。
- 深入解释方法和参数传递（这些往往是阻碍初学者迅速掌握编程的绊脚石）。
- 强调面向对象编程的概念。
- 提供 UML 图，帮助读者理解面向对象编程。
- 与 Java 和 C++的比较，也是作者另外两本书 *Java by Dissection* 和 *C++ by Dissection* 的对等参考书。[①]

各章特色

每章都包含以下教学元素：

解析 通过对程序代码的解析，详细讲解了重要示例程序中的主要元素。这种逐步讨论新的编程思想的方式有助于读者透彻理解所碰到新的编程思想。

面向对象的编程 书中引导读者循序渐进地接受面向对象的编程风格。第 6 章“类和抽象数据类型”介绍了类。类是生成模块化程序和实现抽象数据类型的基本机制。类变量是要处理的对象。第 8 章“继承”解释了继承和虚拟函数，这是继承的两大重要元素。第 12 章“使用 C#进行 OOP 编程”讨论了 OOP 编程的基本原理。

编程风格和软件工程方法 编程风格和软件工程方法在全书各处都进行了重点讲解。本书一开始就解释了重要概念，如结构化分支语句、嵌套式控制流程、自上而下设计法和面向对象编程等。此外，本书一开始便提倡并采用统一的、正确的编码风格，同时，还仔细解释了它的重要性及基本原理。本书采用的编码风格是 C#社区的程序专家们最常用的。

能实际运行的代码 书中一开始就介绍了完全能实际运行的程序。利用可执行代码，读者能更好地理解和吸收本书讨论的编程思想。许多程序及其功能都是通过解析法来解释的。编程思想的某些变化形式会重复出现在练习题中。

Pohl 博士提醒 作者根据多年经验而得出的一系列编程提示，概述了基本原理。

与 Java 和 C++的比较 越来越多的入门级程序员将 Java 作为首选的起点语言。C#，C++和 Java 都源于 C。大多数严谨的程序员会学习所有这 4 种语言。这部分内容对已经了解或希望了解 Java 或 C++的读者来说很有参考价值。如果这些内容反而让自己分心，完全可将其忽略。从很多方面看，C#，C++和 Java 都有对等的元素。本书有助于已经熟悉 Java 或 C++，但又打算转移到 C#的读者迅速掌握 C#。同时，还能帮助那些日后打算使用 Java 或 C++来开始职业生涯的学生。此外，由于本书是 *Java by Dissection* 和 *C++ by Dissection* 的配套参考书，所以读者能了解对 Java 或 C++概念的完整解释，从而充分利用本书的教学方法（即解析法）。

小结 各章最后提供简要小结，对新概念加以巩固。

复习题和练习题 复习题和练习题有助于读者自测所学得的编程知识。许多练习题都可以在阅读本书时完成，这样大大鼓励了读者自学。除温习 C#语言的特性，有的练习

① 清华大学出版社即将出版这两本书的中译本，书名分别为《Java 解析教程》和《C++解析教程》。

题还更详细地分析了某一个主题，有的练习题还能扩展读者的知识面，帮助读者了解 C# 语言的高级用法。

作为教材

本书可作为大学一年级编程教材，课时安排为一学期。第 1～6 章通过使用数组和基本的对象编程，概述了 C#程序语言。如果将课时安排为两学期，那么，第 2 学期就可以专门讲解更高级的数据类型、OOP、继承、文件处理和软件工程，这些是第 7～12 章讨论的主题。对于已具备一定编程基础，但不一定了解 C#的学生，教师可简要讲解本书的所有内容。本书还可作为要求学生使用 C#的其他计算机科学教材。在类似的程序语言教材中，本书还可以用作同样采用解析法来讲解 C，Java 和 C++的概念与基本原理、并使用大量相同示例，但处理方式不同的同类书籍的补充读物。

交互式环境

本书显然是为交互式环境编写的，鼓励学生使用键盘和屏幕来实际体验 C#程序。

专家如何使用本书

本书虽然是为初级程序员设计的，但同样适合打算全面了解 C#的高级程序员。计算机专家如果结合本书和 Al Kelley 及 Ira Pohl 合著的 *A Book on C*，能从这两本书中汲取两种语言的精华。这两本书可以配套购买和使用，书中综合讲解了 C/C#程序语言及其可能的用法。本书甚至还可以与 Ira Pohl 和 Charlie McDowell 合著的 *C++ by Dissection* 一起使用，书中向学生或专家解释了如何综合利用 C#，C++和 Java 来实现面向对象编程。

本书教参

本书提供辅导材料，导师可根据具体的课程安排来修改本教程。

- 复习题和练习题答案
- 示范程序

要想了解本书教参的详情，请访问 *www.aw.com/cssupport*，查看本书联机信息。

关于 SDK

我们没有提供 SDK 印刷文档，但安装了 SDK 后，会获得联机帮助。可设置 SDK，以运行.NET Framework Redistributable，后者只运行.NET Framework 应用程序或.NET

Framework Software Development Kit（用于生成和运行.NET Framework 应用程序）。每个选项的安装需求见后。

.NET Framework Redistributable 需求

- 个人计算机采用 Pentium 级 90 MHz 或更高主频的处理器
- 操作系统：Windows 2000，要从微软安全网页（*www.microsoft.com/security*）安装最新版本的 Windows 服务包和关键更新；Windows XP（要运行 ASP.NET 应用程序，需要专业版）；Windows NT 4.0；Windows Millennium Edition（Windows Me）；Windows 98
- Microsoft Internet Explorer 5.01 或更高版本
- CD-ROM 驱动器
- 显示：800×600，256 色
- 微软鼠标或其他兼容指点设备
- RAM：32 MB（推荐 96 MB 或更大）
- 安装所需硬盘空间：160 MB
- 运行所需硬盘空间：70 MB

.NET Framework SDK 需求

- 处理器：Intel Pentium 级 133 MHz 或更高主频
- 操作系统：Windows 2000，已经安装了最新 Windows 服务包和关键更新，后两者可从微软安全网页（*www.microsoft.com/security*）下载；Windows XP（如果要运行 ASP.NET 应用程序，则需要专业版）；Windows NT 4.0
- RAM：128 MB（推荐 256 MB 或更大）
- 安装所需硬盘空间：600 MB
- 运行所需硬盘空间：370 MB
- 显示：800×600，256 色
- Microsoft Internet Explorer 5.01 或更高版本
- CD-ROM 驱动器
- 微软鼠标或其他兼容指点设备

目　录

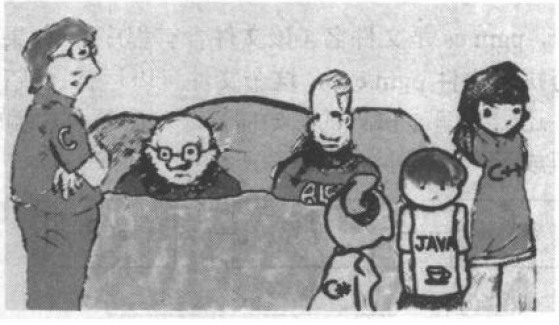
第 1 章　编写 C#程序

本章带您进入 C#编程世界。下面将讨论如何编程，并全面讲解大量基本程序要素。这里展示的基本思路是为后文所做的铺垫，我们将在随后各章详细讲解。这些“第一个”程序（也就是最开始展示的几个 Hello 程序）强调了 C#的基本输入/输出。在任何程序语言中，将信息输入机器以及从机器输出信息是程序员要掌握的首要任务。

C#分别用 Write()和 ReadLine()方法来实现输出和输入。本章要解释这两个方法的用法。此外，还要介绍如何用变量保存值、如何通过表达式和利用赋值来更改变量值。

全书各处穿插并解析了大量例子，可从中详细了解每种编程结构是如何工作的。本章介绍的主题将在以后各章重复出现，并在相应的位置进行详细解释。这种螺旋式的学习方法突出强调了 C#程序员必须掌握的基本思想和技巧。

C#在很大程度上是 C 的一个超集，是 C++和 Java 的代替语言。通过学习 C#，读者还可以学到核心语言 C。本书是“解析”丛书中的第 4 本。这套丛书中的另外 3 本分别是：Al Kelley 和 Ira Pohl 合著的 *C by Dissection*（第 4 版）、Ira Pohl 的 *C++ by Dissection* 以及 Ira Pohl 和 Charlie McDowell 合著的 *Java by Dissection*。程序员要想紧跟技术发展潮流，必须能驾轻就熟地使用这 4 种基于 C 的语言。



“Algol 爷爷像你们这么大的时候，只能在 4K 的内存中运行程序，用纸带来解决问题！”

1.1　编程前的准备

编程是为了指导机器执行特定任务或解决特定问题。按步骤完成既定任务的方式称为算法。因此，编程是一项与算法和计算机打交道的活动。我们都习惯于用英文向某人发出指令，让他来执行指令。编程过程也如此，不同的是机器不能容忍歧义，而且必须用简练的语言和繁复的细节来指定所有步骤。

编程过程

1. 指定任务。
2. 找出解决任务的算法。

3. 用 C#为算法编写代码。
4. 测试代码。

计算机是数字化的电子设备，由 3 大组件构成：处理器、内存和输入/输出设备。处理器也叫做**中央处理器**，简称 CPU。处理器负责执行保存在内存中的指令。数据随同指令一起，也保存在内存中。处理器一般遵从指令按既定的形式处理数据。输入/输出设备从外部代理处将信息输入到机器，并向这些代理提供信息。输入设备一般是终端键盘、硬盘、CD 光驱和 DVD 光驱。输出设备一般是终端屏幕、打印机、硬盘和可写光驱等。一台机器的物理构成可能非常复杂。

操作系统由一系列特殊程序构成，具有两大用途。其一，整体上检查和协调机器资源。例如，在磁盘上创建文件时，操作系统会负责把这个文件放入相应的位置，并跟踪名称、大小和创建日期。其二，操作系统提供一些工具，这些工具对 C#程序员而言，是非常有用的。其中两个工具相当重要：文本编辑器和 C#编译器（或称生成器）。

我们假设读者会用文本编辑器来创建和修改 C#代码文件。C#代码也称为**源代码**，含有源代码的文件称为**源文件**。创建含有源代码的一个文件（一个程序）之后，要调用 C#编译器。例如，使用 Microsoft Windows 终端环境，可用命令：

```
csc pgm.cs
```

来调用 C#编译器。

csc 是 C#编译命令，pgm.cs 是文件名（该文件含有程序）。如果 pgm.cs 中没有错误，该命令就会生成一个**可执行文件** pgm.exe，这个文件可以运行（或称“执行”）。

写好一个程序后，还需要编译和测试。如需修改，必须再次编辑源代码。因此，编程过程的各个部分便组成了一个周期，如下图所示：

程序员对程序表现感到满意后，整个编程周期就算结束了。

还可使用大量集成开发环境（Integrated Development Environment，IDE）工具，比如 Microsoft Visual Studio。IDE 提供了一个编辑器来创建程序，并提供了编译、调试和运行这些程序所需的全部工具。附录 D“Visual Studio .NET 中的 C# ”介绍了如何使用 Microsoft Visual Studio IDE。

1.2 第一个程序

任何人学习编程的第一个任务都是屏幕打印。首先介绍如何编写传统的第一个 C#程序（在屏幕上打印“Hello, World!”）。完整程序如下：

程序 Hello1.cs

```
// Hello world in C#
// by Olivia Programmer

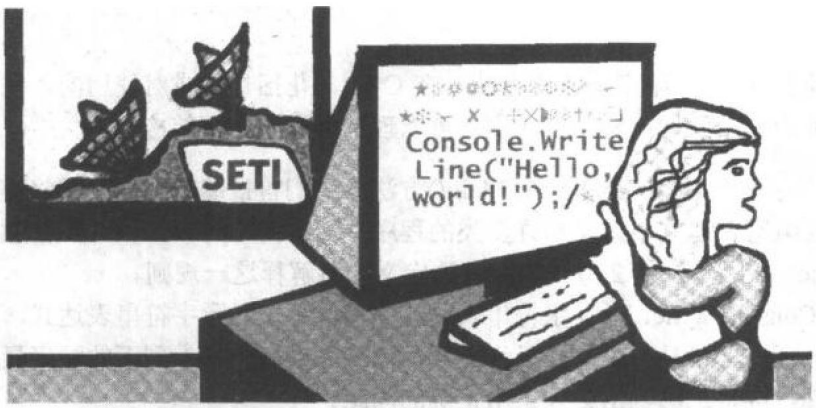
using System;

class Hello {
  public static void Main()
  {
    Console.WriteLine("Hello, world!");
  }
}
```

该程序的屏幕输出如下：

```
Hello, world!
```

使用文本编辑器将以上代码输入文件，文件扩展名为.cs。要选择易记的文件名，而且应与文件包含的类名对应。假定含有 Hello 类（class Hello）的程序文件名为 Hello1.cs。



“嘿！海蒂，Alpha Centauri 在说‘Hello’，是用 C#说的！”

Hello1 程序解析

- ```
  // Hello world in C#
  // by Olivia Programmer
  ```

  双斜杠（//）表示当前行其余部分是注释。程序文本可以放在页面内任何位置，记号之间的空白会被忽略。空白、注释和文本缩进均用于创建一个结构清晰、便于编档和易于阅读的程序，同时不会影响程序的语义。

- ```
  using System;
  ```

  C#程序 hello.cs 导入任何需要的库定义。在这种情况下，可在 System 库中找到用于典型 C#编译器系统的 I/O 库。编译器知道在哪里找到它和其他系统文件。在 C#系统上，标准 C# I/O 定义封装在 System 命名空间。使用 using 定义后，即使没有为每个名称附加 System 前缀，也能使用这些名称。目前，请将 using *Namespace* 视为一种必要的约定。6.5.1 节“圆点运算符、命名空间和类名”将详细解释命名空间。

- ```
  class Hello {
    public static void Main()
    {
  ```

  C#程序是一系列类的集合。这里，程序是一个单独的 Hello 类。它包含 Main()方法，全称是 Hello.Main()，以区别于其他类中的 Main 方法。Main()方法的主体是以左花括号（{）开始的。程序的运行从执行 Main()方法开始。C#中的方法有一个返回类型，那就是 void，表示不返回任何值。如声明为 int Main()，Main()将返回整数值。

- ```
  Console.Write("hello, world!");
  ```

  WriteLine()方法位于 System 库中。它是显示输出的主要方法。它向屏幕打印一个字符串表达式，本例中，则是圆括号内的字符串。该方法还可以将屏幕光标移动到新行。注意，如果没有“using System”语句，我们也可以这样写：

  ```
  System.Console.WriteLine("Hello, world!");
  ```

- ```
  }
  ```

  右花括号（}）结束 Main()方法。在 C#中，花括号是成对使用的，左花括号可理解为一个开始结构，右花括号可以理解为一个结束结构。

这个程序有一个 Hello 类和一个 Main()方法。我们首批编写的程序都将遵循这个结构。以后还要编写含有许多方法和许多类的程序。这些例子的 Main()方法都放在特殊单词 public static void 之后。4.2 节“静态方法定义”将解释这一规则。

表达式 Console.WriteLine（字符串表达式）在屏幕上显示字符串表达式，再将屏幕光标移到下一行。屏幕是一个二维显示屏幕，按从左到右、从上到下的顺序显示输出。为保证可读性，屏幕上的输出必须采用正确的间距。

我们要重新编写刚才的程序，以便使用两条输出语句。尽管这个程序不同于刚才的第一个程序，但两者的输出是一样的。

程序 Hello2.cs

```
// Hello world in C#
// by Olivia Programmer
// Version 2

using System;

class Hello {
  public static void Main()
  {
    Console.Write("Hello, ");
    Console.WriteLine("world!");
  }
}
```

注意第一条语句中的字符串，它的末尾添加了空白字符。如果这里没有空白字符，输出中的“Hello, world!”这两个词之间就不会有空距。

最后一次修改这个程序时，我们要添加词汇“Hello, universe!”，并以两行的形式打
```