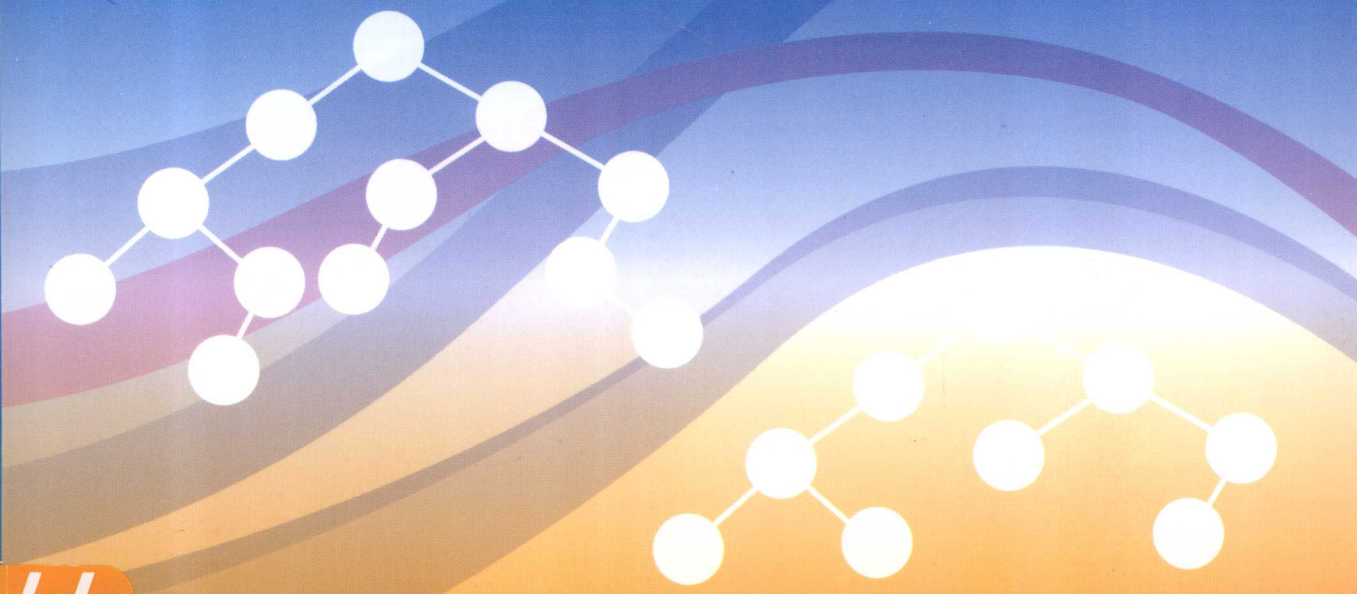
高等学校计算机专业规划教材

Visual C#程序设计

（2012版）



邱仲潘 王帅 孙赫雄 编著

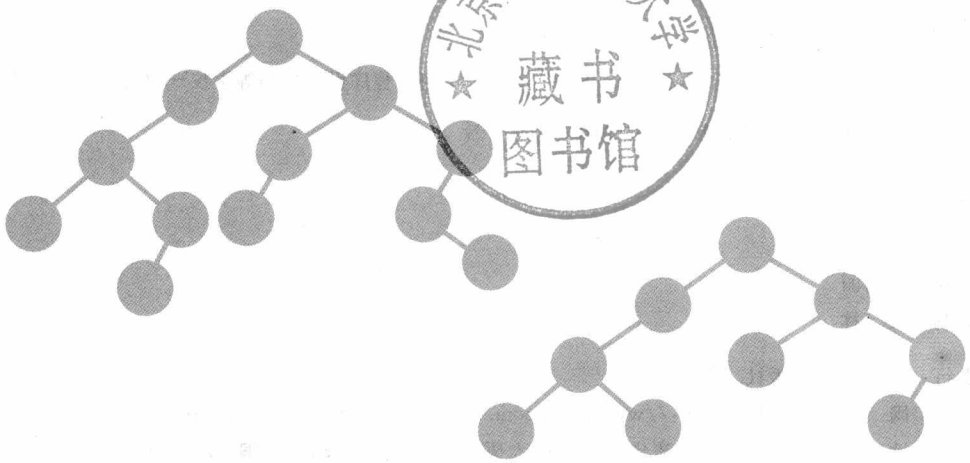
清华大学出版社

· 013034451

TP312C
2168

高等学校计算机专业规划教

Visual C#程序设计
（2012版）



北京航空航天大学
藏书
图书馆

邱仲潘 王帅 孙赫雄 编著

TP312C
2168

清华大学出版社

北航 C1641759

01303451

内 容 简 介

本书以 Visual C# 2012 语言为工具，介绍面向对象程序设计中的基本概念和方法，并通过大量的程序实例和相关练习逐步掌握高级程序设计语言的基本知识和基本技术，在理论和实践上使学生掌握面向对象的思想方法并初步具备软件开发的能力。全书共 17 章，内容包括流程控制语句、类与对象、继承与多态、委托与事件、数组与集合、泛型、反射、字符串操作、文件与流、多线程编程、Windows 窗体与控件设计、数据库编程、网络通信编程、异步编程等。本书的代码可以直接使用在读者开发的应用程序中，非常适合作为 Visual C# 程序设计的教材或参考用书。

本书封面贴有清华大学出版社防伪标签，无标签者不得销售。
版权所有，侵权必究。侵权举报电话：010-62782989　13701121933

图书在版编目 (CIP) 数据

Visual C# 程序设计：2012 版 / 邱仲潘，王帅，孙赫雄编著．—北京：清华大学出版社，2013.4
高等学校计算机专业规划教材
ISBN 978-7-302-30987-1

Ⅰ. ①V…　Ⅱ. ①邱…　②王…　③孙…　Ⅲ. ①BASIC 语言－程序设计－高等学校－教材　Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字(2012)第 301993 号

责任编辑：龙启铭
封面设计：何凤霞
责任校对：焦丽丽
责任印制：宋　林

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课 件 下 载：http://www.tup.com.cn，010-62795954

印 刷 者：清华大学印刷厂
装 订 者：三河市新茂装订有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：34　　**字　　数**：783 千字
版　　次：2013 年 4 月第 1 版　　**印　　次**：2013 年 4 月第 1 次印刷
印　　数：1～3000
定　　价：69.00 元

产品编号：050226-01

前言

编程语言这么多，为什么选择C#？因为C#语言是一门简单、现代、优雅、面向对象、类型安全、平台独立的新型组件编程语言。其语法风格源自大家熟悉的C/C++家族，融合了Visual Basic的高效和C/C++强大，其优雅的语法风格、创新的语言特性深受世界各地程序员的好评和喜爱。众所周知，C#起源于C语言家族，C、C++和Java的程序员能很快熟悉它。C#获得了ECMA和ISO/IEC的国际标准认证，分别是ECMA-334标准和ISO/IEC 23270标准。.NET框架的C#编译器就是根据这两个标准实现的。如果你希望学习微软公司最新推出的Visual C# 2012，请试一试这本书。本书作者从事IT行业二十多年，是最早一批接触C#的人，而且至今仍然在这块土地上辛勤耕耘。本书一定能够给你一段轻松愉快的学习历程。

本书的作者包括邱仲潘、王帅、孙赫雄，宋智军、张星成、黄宇轩也参与了其中，做了大量工作，在此深表感谢。

目录

此为试读，需要完整PDF请访问：www.ertongbook.com

此为试读，需要完整PDF请访问：www.ertongbook.com

第 1 章 Visual C# 2012 简介

微软公司从 20 世纪 90 年代开始就一直主导了桌面计算机的操作系统和开发环境，我们都是从视窗系统的一个个新版本、VS 环境的一个个新版本跟踪过来的，每个新版本总是有许多让老用户惊喜的改进和新特性，本书要介绍的 Visual C# 2012 亦不例外。如果你对某些内容已经熟悉，不妨向下翻页，直到看到让你惊喜的新特性。后面很可能用到前面介绍的某些知识，但既然书在手里，可以随时回头翻阅。

1.1 C# 概 述

C#（读作 C Sharp）是一种简单、现代、面向对象、类型安全的编程语言，开发人员可以使用它来构建在.NET Framework 上运行的各种安全、可靠的应用程序。C# 源于 C、C++、Java，采三家之所长并增加了自己的特点。ECMA 和 ISO/IEC 都对 C# 做了标准化，分别为 ECMA-334 和 ISO/IEC23270，而 Microsoft Visual C# 2012 对这两个标准都提供了很好的支持。

1.1.1 C# 的产生与特点

在过去的二十多年里，C 和 C++ 已经成为商业软件开发领域中使用最为广泛的语言。它们为程序员提供了十分灵活的操作，不过却牺牲了一定的效率。与诸如 Microsoft Visual Basic 等语言相比，同等级别的 C/C++ 应用程序往往需要更长的时间来开发。同时，C/C++ 与其他一些语言相比又有其复杂性，所以程序员都试图寻找，是否有一种新的语言能在功能与效率之间达到一个更为理想的平衡点？C# 正是 Microsoft 对这一问题的解决方案。

1.1.1.1 C# 产生

C# 是在 2000 年由 Microsoft 公司 Anders Hejlsberg 和 Scott Wiltamuth 领导的小组开发的，作为.NET 平台上的语言，使程序员可以方便地集成到.NET。C# 主要沿袭了 C/C++ 语言，而语法又与 Java 相近。它摒弃了 C++ 复杂性，不再支持宏、模板和多重继承，使它更易用、更少出错。并且 C# 对 C++ 也提供了相应的扩充，如严格的类型安全、版本控制、垃圾收集（garbage collect）等。这些功能不仅提高了 C# 语言的安全性、可靠性，同时使 C# 更适合于开发组件级应用。

1.1.1.2 C# 特点

C#作为一种针对.NET平台的开发语言，可以说是一个各种语言优点的集大成者：完全支持类和面向对象编程，包括接口和继承、虚函数和运算符重载的处理；定义完整、一致的基本类型集；对自动生成XML文档说明的内置支持；自动清理动态分配的内存；可以使用用户定义的特性来标记类和方法；对.NET基类库的完全访问权，并易于访问Windows API；以VB的风格支持属性和事件；改变编译器选项，可以把程序编译为可执行文件或.NET组件库；可以用于编写ASP.NET动态Web页面和XML Web服务。

1.1.2 C# 4.0新特性

回顾C#发展的历史，C# 1.0模仿Java，并保留了C/C++的一些特性如struct，新学者很容易上手；C# 2.0加入了泛型，也与Java 1.5的泛型如出一辙；C# 3.0加入了一些语法，并在没有修改CLR的情况下引入了LINQ，简直是神来之笔，虽然很多项目出于各种各样如性能之类的原因没有采用，但非常适合小型程序的快速开发，减轻了程序员的工作量，也提高了代码的可读性；C# 4.0增加了动态语言的特性，从里面可以看到很多JavaScript、Python这些动态语言的影子。虽然越来越偏离静态语言的道路，但从另一个角度来说，这些特性也都是为了提高程序员的生产力。

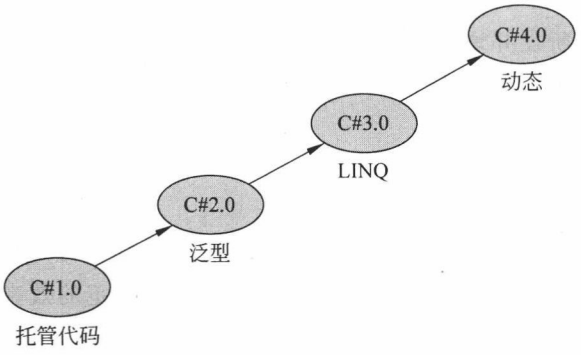


图1.1 C#的演化

C# 4.0的主题就是动态编程(Dynamic Programming)。虽然C#仍然是一种静态语言，但是对象的意义开始变得越来越“动态”。它们的结构和行为无法通过静态类型来捕获，或者至少编译器在编译程序时无法得知对象的结构和行为。C# 4.0包括大量的增强和新增特性，其主要新特性如下所示。

1. 支持动态查找

动态查找允许在编写方法、运算符和索引器调用、属性和字段访问甚至对象调用时，绕过C#静态类型检查，而在运行时进行解析。

2. 同时支持参数命名和可选参数

现在C#中的参数可以通过在成员声明中为其提供默认值来指名它是可选的。在调用该成员时，可选参数可以忽略。另外，在传入任何参数时都可以按照参数名而不是位置

进行传递。

3. 增强的 COM 互操作特性

动态查找以及命名参数和可选参数都有助于使针对 COM 的编程不再像现在这样痛苦。在这些特性之上，增加了大量其他小特性，进一步改善了互操作体验。

4. 变性

C# 4.0 中，把过去的 IEnumerable<string>变为现在的 IEnumerable<object>，并包含了类型安全的“协变性和逆变性”(co-and contravariance)，而且通用的 BCL 也将利用这一特性进行更新。

1.1.2.1　动态查找

动态查找可以用统一的方式来动态调用成员。有了动态查找，当你拿到一个对象时，不用管它是来自于 COM 还是 IronPython、HTML DOM 或是反射，只需要对其进行操作即可。运行时会帮你指出针对特定的对象，以及这些操作的具体意义。这种方法带来了巨大的灵活性，并能极大程度地精简代码，但它伴随着一个巨大的缺点——不会为这些操作维护静态类型。在编译时，会假设动态对象支持任何操作，而如果它不支持某个操作，则只有到运行时才会出现错误。有的时候这不会有任何损失，因为对象根本不具有静态类型，而且其他情况下必须在简洁和安全之间进行权衡。为了帮助进行权衡，C# 的一个设计目标就是允许在每个单独的调用中选择是否使用动态行为。

1. 动态类型

C# 4.0 引入了一个新的静态类型，称为 dynamic。当你拥有了一个 dynamic 类型的对象后，你“对它做的事情”只会在运行时进行解析。

```
dynamic d=GetDynamicObject(…);
d.M(7);
```

C# 编译器允许你使用任何参数在 d 上调用一个方法，因为它的类型是 dynamic。运行时会检查 d 的实际类型。可以认为 dynamic 类型是 object 类型的一个特殊版本，指出了对象可以动态地使用。选择是否使用动态行为很简单——任何对象都可以隐式转换为 dynamic，“挂起信任”直到运行时。反之，从 dynamic 到任何其他类型都存在“赋值转换”，可以类似于赋值的结构中进行隐式转换。

```
dynamic d=7;                                    //隐式变换
int i=d;                                        //赋值转换
```

2. 动态操作

不仅是方法调用，字段和属性访问、索引器和运算符调用甚至委托调用都可以动态地分派。

```
dynamic d=GetDynamicObject(…);
d.M(7);                                         //调用方法
d.f=d.P;                                        //获取并设置字段属性
```

```
d["one"]=d["two"];                    //通过索引进行获取和设置
int i=d+3;                            //调用运算符
string s=d(5,7);                      //调用委托
```

C#编译器在这里的角色就是打包有关“在 d 上做什么”的必要信息，使得运行时可以获取这些信息，并检测对于实际对象 d 这些操作的确切含义。可以认为这是将编译器的部分工作延迟到了运行时。任何动态操作的结果本身也是 dynamic 类型的。

3. 运行时查找

如果 d 是一个 COM 对象，则操作通过 COM IDispatch 进行动态分派。这允许调用没有主互操作程序集(Primary Interop Assembly，PIA)的 COM 类型，并依赖 C# 中没有对应概念的 COM 特性，如索引属性和默认属性。

如果 d 实现了 IDynamicObject 接口，则请求 d 自身来执行该操作。因此通过实现 IDynamicObject 接口，类型可以完全重新定义动态操作的意义。这在动态语言，如 IronPython 和 IronRuby 中有大量使用，用于实现它们的动态对象模型。API 也会使用这类对象，例如 HTML DOM 允许直接使用属性语法来访问对象的属性。

除此之外，如果 d 是一个标准的.NET 对象，操作是通过在其类型上进行反射来分派的，C# 的“运行绑定器”(runtime binder)实现了运行时的 C# 查找和重载解析。其背后的本质是将 C# 编译器作为运行时组件运行，来“完成”被静态编译器延迟的动态操作。

1.1.2.2 命名参数和可选参数

命名参数和可选参数是两个截然不同的功能，但通常一起使用。在进行成员调用时，可以忽略可选参数；而命名参数的方式可以通过名称来提供一个参数，而无须依赖它在参数列表中出现的位置。有些 API，尤其是 COM 接口(如 Office API)本身就是通过命名参数和可选参数编写的。即便是编写.NET 中的 API，你也会发现很多时候你在被迫为不同的参数组合方式编写一个方法的大量重载形式，以便给调用者提供最高的可用性。在这种情况下，可选参数就会成为一种非常有用的替代方式。

1. 可选参数

为一个参数提供默认值就可以将其声明为可选的。

```
public void M(int x, int y=5, int z=7);
```

这里的 y 和 z 就是可选参数，在调用时可以忽略。

```
M(1, 2, 3);                           //正常调用 M
M(1, 2);                              //省略 z,相当于 M(1,2,7)
M(1);                                 //省略 y 和 z,相当于 M(1,5,7)
```

2. 命名的和可选的实参

C# 4.0 不允许忽略逗号之间的实参，比如 M(1,,3)。否则会导致大量不可读的、需要“数逗号”的代码。替代方式是任何参数都可以通过名字传递。因此，如果在调用 M 时

此为试读，需要完整PDF请访问：www.ertongbook.com