C#

程序设计基础

鼎新 查礼 编著

清华大学出版社
http://www.tup.tsinghua.edu.cn

C#程序设计基础

鼎新　查礼　编著

清华大学出版社

(京) 新登字 158 号

内 容 简 介

C#是微软为.NET 平台量身订制的一种语言。使用 C#开发基于.NET 的应用程序，具有良好的安全性和跨平台性。利用 Visual Studio .NET 的所见即所得的功能，可以使整个开发过程更为简洁明快。本书从C#的语法入手介绍了面向对象的程序设计思想，并结合在.NET 平台上 Windows 和 Web 应用程序的开发，详细地阐述了C#的编程方法和编程技巧。

全书内容翔实，结构清晰，语言通俗易懂，示例典型、丰富，既可作为 C#语言的自学、培训教材，也可供高等院校相关专业的师生作为教学参考。

图书在版编目(CIP)数据

C#程序设计基础/鼎新，查礼编著.——北京：清华大学出版社，2002
ISBN 7-302-05942-X

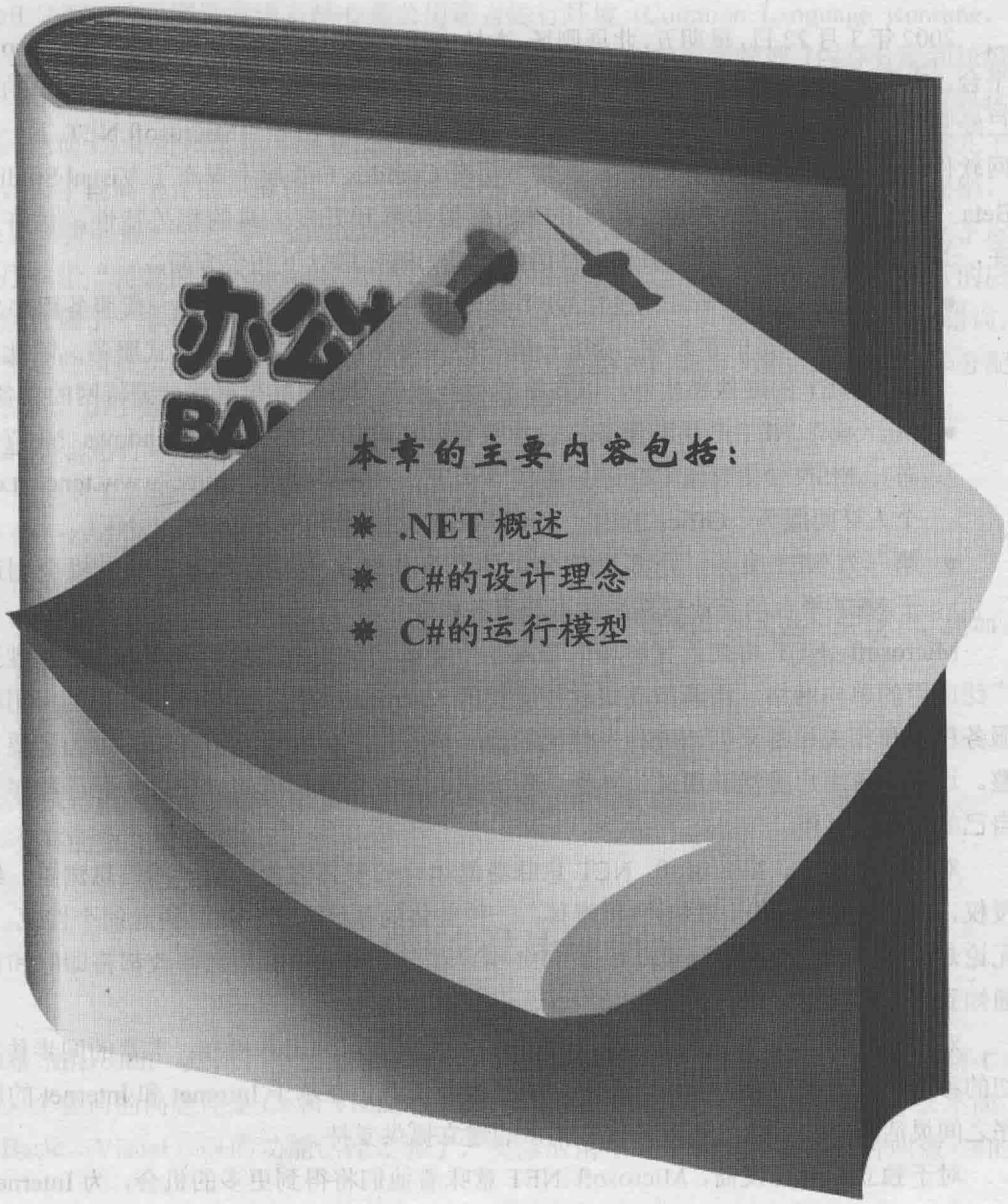
Ⅰ.C… Ⅱ.①鼎…②查… Ⅲ.C 语言—程序设计 Ⅳ. TP312
中国版本图书馆 CIP 数据核字(2002)第 076864 号

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.tsinghua.edu.cn
责任编辑：张秋香
封面设计：王伟
版式设计：康博
印 刷 者：北京昌平环球印刷厂
发 行 者：新华书店总店北京发行所
开　　本：787×1092 1/16　**印张**：18.25　**字数**：433 千字
版　　次：2002 年 11 月第 1 版　2002 年 11 月第 1 次印刷
书　　号：ISBN 7-302-05942-X/TP・3533
印　　数：0001～5000
定　　价：26.00 元

目　录

第 1 章

C# 简介



本章的主要内容包括：

- .NET 概述
- C#的设计理念
- C#的运行模型

本章要点：

本章主要介绍了C#语言的3个特性，它们分别是：面向组件(C#是第一个面向组件的语言)、面向对象、稳定安全。这3个特性都是未来.NET时代对于程序开发语言最基本的要求。C#的产生和.NET平台是分不开的，它是一种为.NET平台量身定制的编程语言。因此，介绍C#，需要从认识.NET平台开始。

1.1 .NET概述

2002年3月22日，星期五，北展剧场。微软中国有限公司发布了中文版的Microsoft .NET平台。实际上，在微软中国的办公地点——希格玛大厦的楼顶，一块蓝色的.NET的广告牌已经竖立了将近2年的时间。2000年6月，微软公司推出了“Microsoft.NET下一代互联网软件和服务战略”，同年11月，微软公司在Comdex Fall展上发布了Visual Studio .NET Beta 1版，并展示了其.NET发展战略的框架体系和开发工具的相关特性。也许到了现在，.NET的含义还在不断地加深，目前的含义大概包括以下几个方面。

- Microsoft .NET平台：包含.NET基础结构和工具，以运行新一代服务程序，.NET支持更加丰富的客户端，.NET构造模块支持新一代高度分布式服务。除此之外，还有.NET的硬件和软件，以支持新型智能化因特网甚至是移动因特网的设备。
- Microsoft .NET产品和服务：包含带有核心构造模块服务的Windows .NET(尚未发布)、MSN .NET(比QQ还好用的聊天工具，下载地址：http://www.tencent.com)、个人订阅服务、Office .NET、Visual Studio .NET和bCenteral for .NET。
- 第三方.NET服务：众多微软公司的商务伙伴和第三方开发商将有机会制造出基于.NET平台的企业软件和垂直型服务程序。

Microsoft .NET将把计算和通信带入一个丰富、合作和交互的环境中，远远胜过目前广泛应用的单向网络。由新型高级软件支持的Microsoft .NET将利用一系列的应用程序、服务程序和相关设备来创造一种个性化的数字体验，它能够不断地按照用户的需要自动调整。这意味着用户将能够通过一整套由新型软件开发技术构成的一个整体服务程序来管理自己的生活和工作。

对于用户来说，Microsoft .NET意味着简单化的整体服务；统一的信息浏览、编辑和授权，查看资料，工作，联机/脱机媒体；一种整体的系统方案；随时随地的个性化。例如，无论是通过个人电脑、便携设备还是SIM卡，对于个人信息的任何修改都将即时和自动地通知到所有需要这些信息的地方。

对于企业，Microsoft.NET意味着统一的信息浏览、编辑和授权；丰富的同步传播；密切的移动通信联系；得力的信息管理和电子商务工具，在基于Intranet和Internet的服务程序之间灵活地切换，为动态商务伙伴关系的建立提供支持。

对于独立软件开发商，Microsoft .NET意味着他们将得到更多的机会，为Internet时代

创造更多的新型高级服务。这些服务可以自动从本地或远程取得和利用所需信息，无须为不同的工作环境重新编写程序。Internet 上的一切都变成这种新一代服务程序的潜在构造模块，而每一种应用程序都可以在网上使用。

正如微软公司首席执行官鲍尔默所说："Microsoft .NET 代表了一个集合、一个环境、一个可以作为平台支持下一代 Internet 的可编程结构。"Microsoft .NET 的策略是把 Internet 本身作为构建新一代操作系统的基础，将 Internet 和操作系统的设计思想合理延伸。这样，开发人员必将创建出摆脱设备硬件束缚的应用程序，以便轻松地连接 Internet。

虽然 C#是.NET 力推的语言，但是开发.NET 应用的语言不一定是 C#。事实上，Microsoft .NET 的程序开发语言核心是公用语言运行环境 (Common Language Runtime，CLR)。CLR 和 Java 虚拟机一样，也是一个运行环境，它负责资源管理 (内存分配和垃圾收集)，并保证应用和底层操作系统之间必要的分离；不一样的是 Java 虚拟机仅仅可以运行 Java，然而 CLR 可以运行 Visual Basic .NET、Visual C++ .NET、Visual C# .NET 等几十种语言。为了提高平台的可靠性，并达到面向事务的电子商务应用所要求的稳定性级别，CLR 还要负责其他一些任务，比如监视程序的运行。按.NET 的说法，在 CLR 监视之下运行的程序属于"托管的"(Managed)代码，而不在 CLR 监视之下且直接在裸机上运行的应用或者组件属于"非托管的"(Unmanaged)的代码。CLR 将监视形形色色的常见编程错误，许多年来这些错误一直是软件故障的主要根源，其中包括访问数组元素越界，访问未分配的内存空间，由于数据体积过大而导致的内存溢出等。

CLR 语言运行环境由以下几个部分组成：

- 面向对象的编程模式(继承、多态和异常处理等)。
- 统一标准的类型系统。
- 所有类型运行时刻的元数据支持。
- 全面的.NET Framework 类，覆盖了绝大多数的 Win32 API 和其他一些技术，比如：数据库访问、XML 解析等。
- 高级的调试、发布支持。
- 托管的代码执行环境和高级内存管理机制。
- 系统范围内的垃圾回收器负责所有托管内存的生命期控制。
- 公开的安全模型。

1.2 C#的设计理念

C#是 Microsoft 为.NET 平台量身订做的新编程语言。大部分人第一次听到或看到 C#时，第一个会问的问题便是 C#和 Visual Basic、Visual C++等传统的开发语言有什么不同；Visual Basic、Visual C++的功能已经足够了，实际应用中是不是真的需要一种叫做 C#的语言。

说到 Visual Basic 和 Visual C++的差别，也许很多人都可以说得头头是道：Visual Basic 简单，但是功能不强；Visual C++功能强大，但是太复杂。所以，C#的出现是具有一定的历史意义的。C#就是一种具有像 Visual C++一样强大的功能，并且使用起来像 Visual Basic 一样简单的语言。

C#语言是微软公司为它的 Microsoft .NET 计划推出的核心编程语言。该语言不仅继承了 Visual Basic、Visual C++语言的优点，还几乎综合了目前所有编程语言的优点，并结合 Internet 发展的需要，增加了丰富的新特性并增强了功能。C#语言面向实体的设计，可以用来构建服务于高水平的商务目标的组件。C#语言的组成部分使用简单的 C#语言结构体，并且能被转换成 Web 服务，允许人们通过 Internet 调用在任何操作系统上运行的任何语言。C#作为一种编程语言，具有以下几个重要的特点：

- 兼容性
- 灵活性
- 简单性
- 现代性
- 面向组件
- 类型安全
- 可以进行版本控制

注意：

C#是从 C 语言/C++语言发展来的，但是为了符合 1.1 节提到的.NET 安全特性，在 C#中不存在类似于 C 和 C++中的指针的功能。也就是说，C#不能直接访问物理内存。

1.2.1　C#的面向组件特性

C#是第一个面向组件(Component-Oriented)的程序设计语言。近几年来，面向组件式的程序设计方式已被广泛地应用，不论是在三层式应用程序结构中的展示层(Presentation Tier)、商业逻辑层(Business Logic)、数据存取层(Data Tier)的开发上，还是在多层式的开发上，组件均扮演着相当重要的角色。

例如，商业逻辑方面的设计通常是以组件的形式存在，而数据存取方面也可以通过 ADO(ActiveX Data Object)组件来完成。开发组件的能力是 C#设计时相当重视的一环，它支持组件所需的属性(Property)、方法(Method)、事件(Event)、索引(Index)、设计时属性(Design Time Attribute)与运行时属性(Runtime Attribute)，以及 XML 注解(Comment)与 XML 文件(Document)。

而且，令人兴奋的是组件的设计可以在 C#中一次完成，不再像 C++一样需要处理一些头文件(Header File)、IDL 等外部文件。同时，产生的组件也可以在 ASP.NET 的网页中直接使用，这对于广大的 ASP 程序员来讲又是一大福音。

1.2.2 C#的面向对象的特性

C#另一个重要的设计目标是对面向对象(Object-Oriented)能力的支持。在 C#中，每一种东西都是对象，就连原始类型(Primitive Type)也算是一个对象，所以程序员便可以调用对象提供的方法(Method)来操作这个对象。举例来说，可以编写这样的程序代码：

```
String S=789.TOString();
```

这行程序的意义是将整数 789 通过 ToString()函数转换成字符串对象，然后保存在 S 变量中，十分方便！

C#的统一类型系统(United Type System)特性大大简化了系统的开发操作，并提高了扩展性(Extensibility)与重复使用性(Reusability)，例如，集合(Collection)对象可以存放任何类型的对象。

1.2.3 C#的稳定性与安全性

使用 C#可以开发下一代具有很好的稳定性及安全性的应用软件。在它的语法中没有指针，因此 C#程序不会有“乱指”、“弹飞”等困扰。这对于预防黑客攻击是非常有效的，因为黑客软件大多数是利用指针错乱执行有害代码的机会而侵害服务器的。

当然，如果习惯使用指针，或者为了节约系统资源一定要使用指针，可以使用 unsafe 来声明，以告诉 CLR 这个程序使用了 unsafe 的指针。那么，开发人员在开发 C#程序的时候就可以像开发 C/C++程序一样自如地使用指针。但是除非必要，建议慎用。

除了指针之外，C#还有 3 个“绝招”可以使整个系统更加稳定、安全。

1. 自动资源回收

C#利用.NET Framework 自动资源回收 (Automatic Garbage Collection) 功能，自动将被应用程序占用的无效内存回收，并交给操作系统重新配置。因此，使用 C#编写的程序将不会再出现内存漏失(Memory Leak)的现象。

2. 结构化的异常处理与类型安全

另外，C#结构化的异常错误处理提供了一致性的错误处理机制。类型安全检查(Type-Safety Verification)的特性可让 C#程序中不再会有使用到未初始化变量的情况。请引用下面 C#程序代码片段：

```
int a;
a= a + 1;
```

程序中声明一个整数类型的变量 a 在未进行初始化的状态下，便直接进行加 1 的操作，这在编译时会产生编译错误。

在 C#中也不会有不安全的类型强制转换，如下面程序中声明了一个整数类型的变量 a，

并初始化为 777，然后通过转型(Type Casting)变成 byte 类型：

```
int a=777;
byte b=(byte)a ;
```

a 是整数类型，它的范围可以从 -2 147 483 648～2 147 483 647；而 b 是 byte 类型，范围是 0～255。上面的程序代码中，a 的值已被初始化为 1000，超出了 byte 类型能接收的范围。

3. 保留已有的投资

C#拥有极强的交互作用性(interoperability)，它可以通过 XML、SOAP 和 COMDLL 等数据交换形式与任何.NET Framework 中的程序语言链接运行，如 Visual Basic .NET、JScript.NET、COBOL .NET 等。这使得 C#在开发上具备相当大的弹性，也不会让用户浪费已有的投资，如已有的 COMDLL 程序代码和 C++的程序开发经验。实际上，.NET Framework 中有几百万行以上的程序代码都是使用 C#编写而成的。简而言之，使用 C#可以降低学习难度，更能够提高开发的效率。

1.3　C#的运行模型

第 2 章，我们将会开始介绍第一个 C#的程序。但是，在开始之前，有必要对于 C#的运行模型，也就是 C#程序是怎么运行的做一下了解。因为，过去用一种语言编写一段程序，再编译成二进制的机器代码就可以执行的时代已经一去不复返了。当今的程序开发需要有自己的算法，但是，也要更多地调用系统资源，这个是大趋势。为了更好地管理、调用系统资源，程序就必须运行在一个平台或者一个环境里。例如，ASP 需要在 IIS 上运行；EJB 的运行也需要自己的容器(Container)，因此，一种语言是如何运行的，以及如何在它所属的平台上运行的，就变得很重要。明白了原理，就可以更好地进行开发，出了问题，也好解决。这里主要介绍的是 C#程序是如何在.NET Framework 上运行的。

1.3.1　C#的运行方式

当编写完一个 C#程序，并通过 C#编译器—— CSC.exe(Visual Studio .NET 自带的编译器，可以通过命令行执行)编译后，便会产生 EXE 或 DLL 等可执行文件。然而，这个 EXE 文件不能直接被送进 CPU 运行，它仅仅是一种中间语言(Intermediate Language)。中间语言是一种与 CPU 无关的指令集(Instruction Set)，这使得由 C#所编译的程序可以在任何支持.NET Framework 的操作系统下执行，实现了微软“编写一份程序，到处可以执行”的目标。事实上，不仅是 C#，任何.NET 程序语言(如 Visual Basic .NET、JScript .NET)，甚至是 COBOL .NET 都是以相同的方式运行的。从这个角度来看，支持多种语言也许是.NET Framework 比 Java 虚拟机技高一筹的地方。

除了产生 IL 之外，CSC.exe 会将元数据加到编译过的程序文件中。所谓的元数据就是“描述数据的数据(The data describe data)”。在.NET Framework 中，元数据用来描述程序的类型函数库(Type Library)，程序代码中包括哪些类(Class)、属性(Property)、方法(Method)等，同时也记录参数，也就是一些外部类函数库、版本和安全性信息。

C#程序被编译时，CSC 会加入.NET 通用语言运行引擎(Common Language Runtime Execution Engine)。当用户启动托管执行之后，操作系统载入器(OS Loader)就会载入这个托管执行和 MSCorEE.DLL。程序是由 Unmanaged 进入点开始执行的(就是指 CorExeMain)，但此进入点不过是一个跳转指令(Jump Instruction)，它会跳至 CorExeMain 开始执行。而 CorExeMain 中所要做的事便是初始化.NET Framework 通用语言运行时，如类加载器、资源收集器等，然后由类加载器载入 IL 程序代码。通过即时编译器(Just-In-Time Compiler)将 IL 程序代码转换成 CPU 可以执行的程序代码(Native Code)。

在真正执行之前，安全检查器将会先检查程序代码的安全性。另外，在.NET Framework 中，类型安全性(Type Safety)也会在程序代码真正执行前先进行检验的操作。如果程序中引用到其他的外部类函数库(Class Library)，CLR 类加载器也会动态地将引用到的类函数库载入。C#程序的整个运行过程如图 1-1 所示。

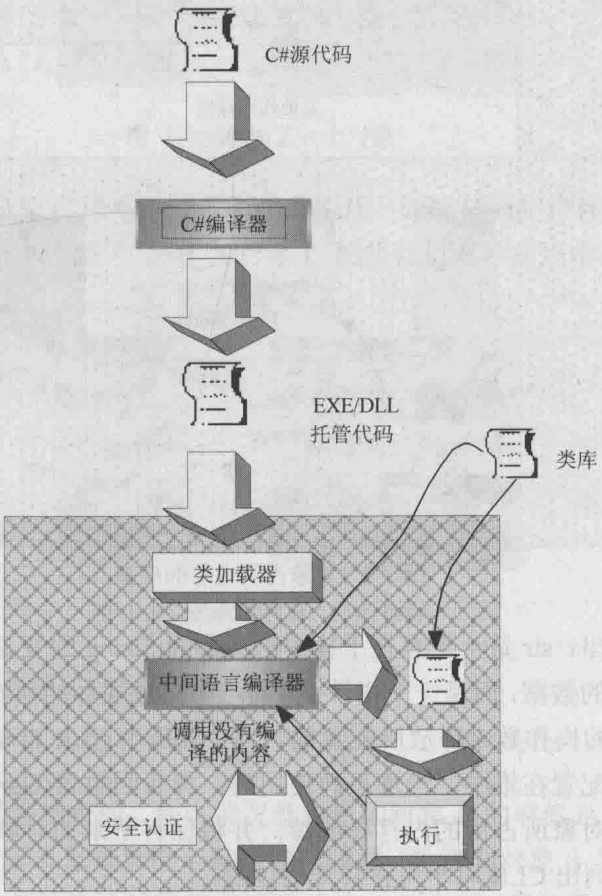


图 1-1　C#的运行方式

图中阴影部分是在.NET Framework 上的 CLR 内进行的工作。从图中还可以看出：在C#真正被执行的时候，还有可能根据程序运行需要不断地提取中间代码进行编译；而不是把这一次运行涉及到的所有的代码进行编译。毫无疑问，这样节约了很多资源。

1.3.2 自动资源回收的过程

本小节将详细讲解资源自动回收的过程。既然C#是面向对象的，那么C#的程序应是一些对象的集合。这些对象以阶层式的形式组织在一起，在 C#中的阶层以“名称空间.类(Namespace Class)”方式组成。对象在物理上就是某一类型在内存中的实体(Instance)，占用一定内存空间，具有一定格式的数据。在 C#中可以使用 new 保留字来建立对象。下面的例子将调用 new 保留字建立一个 String 字符串对象，并将它的值初始化为“Hello World”：

```
string str = new string ("Hello World");
```

一旦执行了以上的代码，Hello World 就被保存在内存中了，格式如图 1-2 所示。

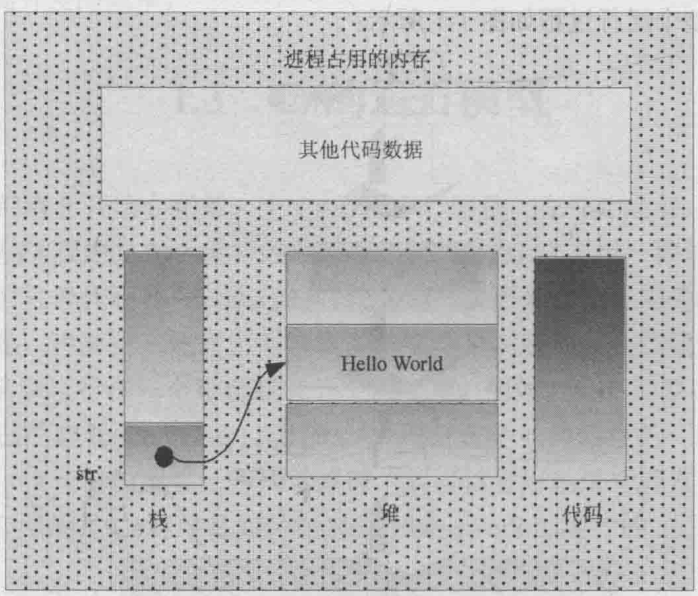


图 1-2　对象占用内存的模型

从图中可以看出，str 是一个存在于栈中的类型为 String 的变量，其实际的内存位置所存放的并不是真正的数据，而是一个指针，指向堆中对象实体的所在位址。C#中并无如 free 或 delete 等相对应的操作数来释放所配置的内存，堆是由 CLR 自动配置与管理的。新的对象实体将会不断地配置在堆中，当堆不敷使用时，资源回收器(Garbage Collector)便自动地将已不再被引用的对象所占用的内存释放掉，并将保存下来的对象重新排列，并修正引用的变量指针。下边列出 CLR 对于内存管理的过程。

(1) 内存的栈中有 3 个对象的指针，分别对应 3 个堆中的对象实体，如图 1-3 所示。