高等学校计算机基础教育教材精选

Visual C# 大学程序设计

崔舒宁 杨振平 贾应智 杨琦 编著

清华大学出版社

高等学校计算机基础教育教材精选

Visual C# 大学程序设计

崔舒宁 杨振平 贾应智 杨琦 编著

清华大学出版社
北京

内 容 简 介

本书以 Visual Studio 2013 为平台，讲述了关于 C# 的编程知识。全书共分为 14 章，其中第 1～10 章主要讲述控制台下的 C#，讲述的重点为面向对象的编程思想；第 11～13 章讲述 Windows 窗体程序的设计，介绍常用的控件、GDI+以及文件读写等方面的知识；最后一章简单讲述常用的数据结构，如线性表和栈等。

本书从基础讲起，是 C# 的入门书籍，可作为高等学校程序设计课程的教材，也可供 C# 程序设计爱好者自学使用。

图书在版编目(CIP)数据

Visual C# 大学程序设计/崔舒宁等编著. —北京：清华大学出版社，2016
高等学校计算机基础教育教材精选
ISBN 978-7-302-42304-1

Ⅰ. ①V…　Ⅱ. ①崔…　Ⅲ. ①C 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(2016)第 287007 号

责任编辑：焦　虹
封面设计：何凤霞
责任校对：焦丽丽
责任印制：杨　艳

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：三河市君旺印务有限公司
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**：21.75　　**字　　数**：499 千字
版　　次：2016 年 2 月第 1 版　　**印　　次**：2016 年 2 月第 1 次印刷
印　　数：1～2000
定　　价：39.50 元

产品编号：064923-01

目录

Visual C# 大学程序设计

第1章

C#和.NET

1.1 C#概述

编程工具与消费类电子设备(如移动电话和 PDA)的出现带来了新的问题与需求。因为新版本的共享组件与旧软件不兼容,所以集成不同语言的软件组件很困难,安装问题也很常见。人们需要基于 Web 的程序,以便访问和使用 Internet。随着移动电子设备的普及,客户不再局限于桌面计算机,软件需要让任何人通过各种不同类型的设备访问。

为了满足这些需求,2000 年微软公司推出了 C# 编程语言。C# 是由微软公司 Anders Hejlsberg 和 Scott Wiltamuth 领导的小组开发的,作为.NET 平台上的语言,C# 可以方便地集成到.NET 中。C# 源于 C、C++ 和 Java,采它们之所长并增加了自己的新特性。C# 是面向对象的,包含强大的预建组件类库,从而使程序员可以迅速地开发程序。C# 和 Visual Basic 共享框架类库,Visual C# 是事件驱动的可视化编程语言,程序在集成开发环境(IDE)中创建。编写的程序响应定时器和用户启动的事件(如鼠标单击与键击)。除了编写程序语句建立 C# 程序之外,还可以用 Visual Studio 的图形用户界面方便地把按钮、文本框之类的预定义对象拖放到屏幕上某个位置,然后标注和缩放它们。Visual Studio 会产生大部分 GUI 代码,利用 IDE,程序员可以方便地生成、运行、测试和调试 C# 程序,从而减少生成可工作程序所需的时间,比不用 IDE 要快得多。

1.2 .NET 框架和公共语言运行时

1.2.1 .NET 框架

技术人员一般将微软公司看成平台厂商。微软公司搭建技术平台,而技术人员在这个技术平台上创建应用系统。从这个角度看,.NET 是微软公司的新一代技术平台,可为敏捷商务构建互联互通的应用系统,这些系统是基于标准的、联通的、适应变化的、稳定的和高性能的。从技术的角度看,一个.NET 应用是一个运行于.NET Framework 之上的应用程序。更精确地说,.NET 应用使用.NET Framework 类库编写,并运行于公共语言运行时(Common Language Runtime)之上的应用程序。如果一个应用程序与.NET Framework 无关,就不能叫做.NET 程序。比如,仅使用 XML 就不是.NET 应用,仅使用 SOAP SDK 调用 Web Service 也不是.NET 应用。目前 .NET Framework 的最新版

本为 4.5，和 Visual Studio 2013 一起发布。

1.2.2 CLR 与 CLI

CLI(Common Language Infrastructure，公共语言基层结构)是为虚拟机环境而制定的规范，使得由各种高级语言所编制的程序可以在不同的系统环境中执行而不必更改或重新编译源程序代码。CLI 指定一种标准的虚拟机中介语言，然后将各种高级语言编制的源代码映射为该中间语言。对于.NET Framework 而言，这种中介语言是 Microsoft Intermediate Language(MSIL)。

在中介语言中，执行程序时，代码通过实时编译器(Just-In-Time(JIT)Compiler)最终被映射为机器码。当然，在 CLI 中介语言中，代码可在其他任何具有 CLI 功能的环境中执行。一个具体的环境就是 CLR。

CLR(Common Language Runtime，公共语言运行时)是一个执行程序的标准化环境。不管程序是用 Visual Basic、C#，还是 C++ 等高级语言来编制，都可以在此环境中执行。因此，按照 CLR 语言标准编写的 C++ 程序属于 C++ /CLI，也就是为 CLI 而编制的 C++ 。

CLI 也被定义为一种数据类型的公共集，叫做公共类型系统(Common Type System，CTS)。它可用于任意语言编写的程序，以实现 CLI 为目标。CTS 规定了在 CLR 环境中如何使用数据类型，并包括一些预定义类型。用户可以定义自己的数据类型，但必须用一种特殊方法来定义以便与 CLR 兼容。有一种标准的表达数据类型的系统，允许不同语言编制的组件用统一的方法来处理数据，从而使得不同语言编制的组件可以结合到单个程序中。CLR 不过是 CLI 规范在个人计算机、Windows 操作系统中的执行而已。

毫无疑问，在其他操作系统环境和硬件平台上，CLI 也同样可行。有时会发现术语 CLI 和 CLR 可交换使用，尽管很明显它们不是一回事。CLI 是一种标准规范，而 CLR 是微软公司对 CLI 的实现。

1.2.3 C#的执行过程

执行 C#代码的过程包括下列步骤。

1. 选择编译器

为利用 CLR 环境提供的优点，必须使用一个或多个针对运行时的语言编译器，如 Visual Basic、C#、Visual C++、F#或第三方编译器(如 Eiffel、Perl 或 COBOL 编译器)。

2. 将代码编译为 MSIL

当编译为托管(CLR)代码时，编译器将源代码翻译为 Microsoft 中间语言(MSIL)，这是一组可以有效地转换为本机代码且独立于 CPU 的指令。MSIL 包括用于加载、存储和初始化对象以及对对象调用方法的指令，还包括用于算术和逻辑运算、控制流、直接内存访问、异常处理和其他操作的指令。要使代码可运行，必须先将 MSIL 转换为特定于 CPU 的代码，这通常是通过实时(JIT)编译器来完成的。由于公共语言运行时为它支持

的每种计算机结构都提供了一种或多种 JIT 编译器，因此同一组 MSIL 可以在所支持的任何结构上编译和运行。

当编译器产生 MSIL 时，它也产生元数据。元数据描述代码中的类型，包括每种类型的定义、每种类型的成员的签名、代码引用的成员和运行时在执行时使用的其他数据。MSIL 和元数据包含在一个可移植、可执行的文件中，文件格式包含 MSIL 或本机代码以及元数据，使得操作系统能够识别公共语言运行时映像。

3. 将 MSIL 编译为本机代码

运行 MSIL 之前，必须先根据公共语言运行时将其编译为适合目标计算机体系结构的本机代码。.NET Framework 提供了两种方式来执行此类转换：.NET Framework 实时(JIT)编译器和.NET Framework Ngen.exe(本机映像生成器)。

应用程序运行时，JIT 编译器可以在加载和执行程序集内容的过程中根据需要将 MSIL 转换为本机代码。由于公共语言运行时为所支持的每种 CPU 体系结构都提供了一个 JIT 编译器，因此开发人员可以生成一组在具有不同的计算机体系结构的不同计算机中进行 JIT 编译和运行的 MSIL 程序集。但是，如果托管代码调用特定于平台的本机 API 或特定于平台的类库，则将只能在该操作系统上运行。

JIT 编译器考虑了在执行过程中某些代码永远不会被调用的可能性。它不是耗费时间和内存将文件中的所有 MSIL 都转换为本机代码，而是在执行期间根据需要转换 MSIL 并将生成的本机代码存储在内存中，供该进程上下文中的后续调用访问。在加载并初始化类型时，加载程序将创建存根(stub)并将其附加到该类型的每个方法中。首次调用某个方法时，存根会将控制权交给 JIT 编译器，后者会将该方法的 MSIL 转换为本机代码，并修改存根以使其直接指向生成的本机代码。这样，对 JIT 编译的方法的后续调用将直接转到该本机代码。

由于 JIT 编译器会在调用程序集中定义单个方法时将该程序集的 MSIL 转换为本机代码，因而必定会对运行时的性能产生不利影响。在大多数情况下，这种性能降低是可以接受的。更为重要的是，由 JIT 编译器生成的代码会绑定到触发编译的进程上，它无法在多个进程之间共享。为了能在多个应用程序调用或共享一组程序集的多个进程之间共享生成的代码，公共语言运行时支持一种提前编译模式。此提前编译模式使用 Ngen.exe(本机映像生成器)将 MSIL 程序集转换为本机代码，其作用与 JIT 编译器极为相似。但是，Ngen.exe 的操作与 JIT 编译器的操作有 3 点不同：

(1) 它在应用程序运行之前而不是在应用程序运行过程中执行从 MSIL 到本机代码的转换。

(2) 它一次编译一个程序集，而不是一次编译一个方法。

(3) 它将本机映像缓存中生成的代码以文件的形式持久保存在磁盘上。

在编译为本机代码的过程中，MSIL 代码必须通过验证过程，除非管理员已经建立了允许代码跳过验证的安全策略。验证过程检查 MSIL 和元数据以确定代码是否是类型安全的，这意味着它仅访问已授权访问的内存位置。类型安全帮助将对象彼此隔离，因而可以保护它们免遭无意或恶意的破坏。它还提供了对代码可靠地强制进行安全限制的

保证。

验证过程中检查 MSIL 代码，确认该代码只能通过正确定义的类型访问内存位置和调用方法。例如，代码不允许以超出内存范围的方式来访问对象。另外，验证过程检查代码以确定 MSIL 是否已正确生成，这是因为不正确的 MSIL 会导致违反类型安全规则。验证过程通过正确定义的类型安全代码集，并且它只通过类型安全的代码。然而，由于验证过程存在一些限制，某些类型安全代码可能无法通过验证，而某些语言在设计上并不产生可验证的类型安全代码。如果安全策略要求提供类型安全代码，而该代码不能通过验证，则在运行该代码时将引发异常。

4. 运行代码

公共语言运行时提供使托管代码执行能够发生以及可在执行期间使用的各种服务的基础结构。在运行方法之前，必须先将其编译为特定于处理器的代码，调用已经为其生成 MSIL 的每个方法；运行该方法时，该方法将是 JIT 编译的。下次运行该方法时，将运行现有的 JIT 编译的本机代码。这种进行 JIT 编译然后运行代码的过程一直重复到执行完成时为止。在执行过程中，托管代码接收若干服务，这些服务涉及垃圾回收、安全性、与非托管代码的互操作性、跨语言调试支持、增强的部署以及版本控制支持等。

1.2.4 垃圾回收

在 C++ 编程中，需要自己来管理申请内存和释放内存，于是有了 new、delete 关键字，还有一些内存申请和释放函数。C++ 程序必须很好地管理自己的内存，不然就会造成内存泄漏(Memory Leak)。在.NET 时代，微软公司为开发人员提供了强有力的机制——垃圾回收(Garbage Collection，GC)。垃圾回收机制是 CLR 的一部分。我们不用操心内存何时释放，从而可以花更多精力关注应用程序的业务逻辑。CLR 里的垃圾回收机制可用一定的算法判断某些内存程序不再使用，回收这些内存并交给系统使用。

1.3 面向对象的编程技术

面向对象的程序设计(Object Oriented Programming，OOP)是软件开发人员多年来从真实世界的建模中受到启发所创造的一种软件开发的方法。利用面向对象的编程方法可以创建功能异常复杂的软件，利用其中的代码复用可以加速软件开发的过程，并增强软件的后期功能的可扩展性与可维护性。

在面向对象的编程方法诞生之前，软件开发人员采用面向过程的程序设计语言编程。面向过程的程序设计语言最主要的特征是采用子过程(子模块)进行软件开发，但随着现代软件的功能越来越强大，面向过程的编程方法明显无法适应大型软件的开发及维护，因此诞生了面向对象的编程方法。

1.3.1 类和对象

在面向对象的编程方法中，类与对象是其核心。对象代表一个存在的实体(实际物

体)或者实际的概念,对象由类建模而成。例如,对雇员、窗口、汽车等都可以建立相应的模型。

面向对象的程序设计思想是现代软件开发的基础。面向对象的程序由各种各样的类、对象和方法组成,它们有机地组织在一起从而实现了复杂的软件功能设计。在面向对象的程序设计中,类与对象是核心,而对象需要由类建模得到。我们在程序中将使用对象来完成不同的功能。对象是消息、数据和行为的组合,对象可以接收和发送消息并使用消息进行交互。消息包含要传递给接收对象的信息。

对象由类得到,类转换为对象的过程称之为类的实例化。因此对象就是类所表现出来的一个实实在在的例子,一个"具体的东西"或者"具体的事情"。对象可以是有形的实体(实际物体),也可以是无形的概念。对象往往有边界,有属性,有方法(特定用途)。例如,张三这个人:

(1) 是有形的实体,有边界。

(2) 有姓名、身高、年龄、性别等,有属性。

(3) 可以走、跑、跳、叫、唱等,有方法。

一般而言,对象应具有以下特性:

- 对象有状态,对象的状态由对象的各种属性和相应的值构成。
- 对象可以显示行为(即方法),对象的行为(方法)使对象可以完成相应的功能。
- 对象有唯一的身份,对象的身份可以把它与其他对象区别开来。

例如,一辆汽车,它有状态,即它可能正在行驶或者已经静止。它有方法,可以左转、右转、减速、加速等。它有一个车牌号(身份),唯一地标识了这辆汽车。两个对象可能什么都相同,但它们不可能有相同的身份,例如相同的两辆轿车,它们肯定会有不同的车牌号。

在当今的现实生活中到处都是形状、颜色各异,大小、功能不同的对象,所以现实生活就是由各种各样的对象构成的。为了对世界上各种各样的对象进行研究,了解和掌握这些对象的习性、功能和使用方法,科学家将世界上的各种对象进行了逻辑上的分类,例如矿物类、植物类、动物类等;并继续将这些类进行更细的逻辑上的分类,例如动物类分成了鸟类、哺乳动物类、鱼类等。每一个正在运动的哺乳动物、正在飞行的鸟、正在水里游动的鱼都是实实在在的对象,科学家怎样将它们划分到哺乳动物类、鸟类或者鱼类中呢?毫无疑问,科学家根据它们所表现出来的共同特征将这些对象进行了分类。

例如,孔雀、麻雀和翠鸟都属于鸟类,因为它们具有鸟类家族的公共特性,即它们都产蛋,都覆盖羽毛,都有空的骨架结构且都能飞翔。所以,我们可以将类理解成对对象进行分类的一个模型,而该类中的对象都是由这个模型所产生的,它们具有这个类模型的共同特征。同样,现实生活中的《C#编程技术》和《ASP.NET 编程技术》都是"书"这个类的实际对象,都是由"书"这个类模型所得到的对象。它们拥有"书"这个类的共同特征。

1.3.2 字段

属性在类的声明中(参见例 1-1)用变量表示,这样的变量称为字段(field),它是在类

声明的内部声明的，但是位于类的方法声明体之外。当类的每个对象维护自己的属性副本时，代表属性的字段又称为实例变量。类的每个对象（实例）在内存中有该变量的一个单独实例。

字段和方法（见1.3.3节）都具有访问权限。通常有3种访问权限：private、public和protected。以private修饰的字段或者方法只允许在类的内部使用，也就是说只允许类中的其他方法调用或者访问。以public修饰的字段或者方法则没有什么限制，从类的外部可以直接使用。protected修饰的字段和方法可以在派生类（见第8章）中使用。

1.3.3 方法

方法（Method）描述了类实际执行任务的机制，从形式上看，相当于C语言的函数。类中的方法，可以直接存取类中的字段的值。C#语言被称为纯面向对象的语言。也就是说，所有的方法都在某个类中，不存在独立于类的方法。

例1-1 一个简单类的示例。

```
1: public class Hello
2: {
3:     private int x=1;                    //定义x为整数,其值为1
4:     public void DisplayMessage()
5:     {
6:         Console.WriteLine("Hello");
7:     }
8: }
```

程序第1行，声明了一个类Hello。在第3行，声明了一个private的字段，含有一个整型变量x。程序的第4行到第8行，声明了一个public的方法DisplayMessage，该方法在控制台输出Hello字样。

类不能直接使用，如前所述，需要根据类来创建对象，也就是类的实例化。语句如下：

```
Hello myHello=new Hello();
```

上述语句通过new关键字，创建了Hello类的一个对象实例myHello，随后可以通过语句

```
myHello.DisplayMessage();
```

来调用Hello类中的方法。

1.3.4 注释

在编写程序的过程中，明了清晰的注释是至关重要的。一般一段复杂的代码经过一段时间后，即使是自己当时绞尽脑汁想出的，可能也会忘得一干二净。如果当时做了注释，就能帮助我们快速回忆起编程思路和需求背景的内容，可以快速投入修改和功能添加等工作中。此外，注释也可以帮助别人快速理解代码，便于交流。C#注释一般常用的有以下3种。

(1) 单行注释：较为短小的注释，以//开头直到本行结束，用于对一行代码的注释。如例 1-1 中的第 3 行。

(2) 多行注释：标注类、方法、参数、返回值的内容等，以///开头。如果有多行，则每一行都以///开头。在方法或类的前面输入///后，Visual Studio 具有自动完成功能，会补齐其余的部分。

(3) 长段注释：描述需求、版本信息等。多行大段的注释以/* 开头，并且以 */ 结尾。

1.4 使用 Visual Studio

在 C# 编程中，程序可以分为两大类：一类是带有 Windows 窗体的标准 Windows 程序，这类程序将在本书的后半部分介绍；一类是控制台程序，本书前几章都将使用控制台程序，其程序的输入和输出是文本，位于控制台窗口。在 Windows 系统中，控制台窗口被称为命令提示符。

下面通过示例讲解如何使用 Visual Studio 2013 创建一个完整的 C# 控制台程序。

例 1-2 控制台程序示例。在本例中，完善例 1-1，使其可以运行。

(1) 启动 Visual Studio 2013，如图 1-1 所示。选择“新建项目”(也可以在菜单中选择“新建项目”)。

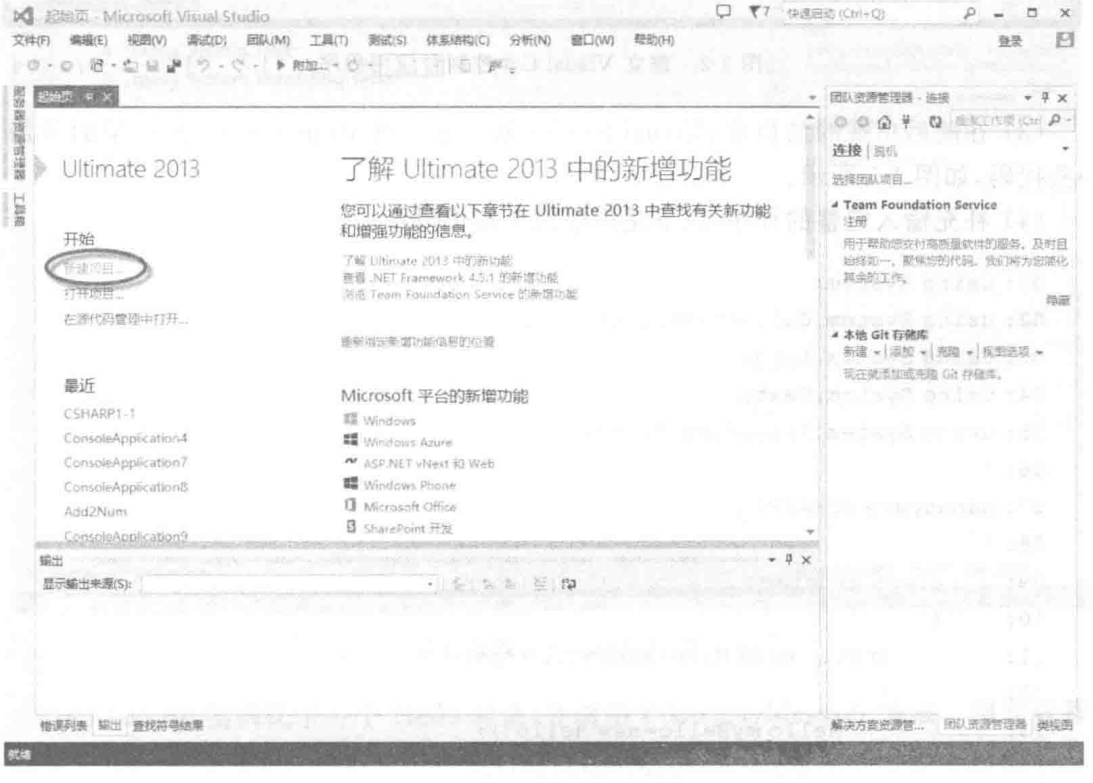

图 1-1 新建一个项目