第 1 部分

C#面向对象程序设计语言基础知识

扫一扫

视频讲解

第1章 C#程序设计导论

C#语言源于C语言家族，是一种简洁、类型安全的面向对象的程序设计语言，主要用来构建在.NET上运行的各种安全、可靠的应用程序。

本章要点：

- 程序设计语言概述;
- C#语言及其特点;
- C#语言的编译和开发运行环境;
- 使用记事本创建简单的C#程序;
- 基于集成开发环境创建简单的C#程序;
- 基于“C#交互”窗口测试C#代码片段;
- C#程序的结构和书写规则;
- 类型的声明和使用;
- 命名空间;
- C#注释与XML文档注释;
- 控制台输入和输出。

1.1 程序设计语言

1.1.1 计算机和程序

计算机是可以连续快速执行一系列指令的机器。通用计算机系统由硬件和软件组成。物理计算机和外围设备统称为硬件；计算机执行的程序称为软件。软件一般分为系统软件和应用软件两大类。系统软件为计算机的使用提供最基本的功能，但是并不针对某一特定的应用领域。而应用软件则恰好相反，不同的应用软件根据用户和所服务的领域提供不同的功能。

按照程序设计语言规则组织起来的一组计算机指令称为计算机程序，计算机程序指定计算机完成任务所需执行的一系列操作步骤。

1.1.2 程序设计和编程语言

设计和实现计算机程序的行为称为程序设计（Program Design），又称为编程（Programming）。

编程语言又称为程序设计语言，是一组用来定义计算机程序的语法规则。每种语言都有一套独特的关键字和程序指令语法。编程语言随硬件的发展而发展。

编程语言分为低级语言和高级语言两类。

- 低级语言与特定的机器有关。每种低级语言都运行在特定的计算机上，与CPU的机器语言或者指令直接对应，很难移植到其他类型的计算机上。低级语言由于无须大量的编译即可被CPU运行，因此以该类编程语言编写的源代码一般比高级语言

编写的源代码编译和运行效率高。

- 高级语言独立于机器，一种高级语言可以在多种计算机和操作系统上运行。高级语言是以人类的日常语言为基础的程序设计语言，使用一般人易于接受的文字表示，使程序编写更容易，也具有较高的可读性。

机器语言和汇编语言属于低级语言。机器语言是第一代程序设计语言，使用二进制代码编写程序，可读性差，但能够直接被计算机识别和执行。不同类型的 CPU 都有自己独特的机器语言。汇编语言是第二代程序设计语言，使用简单的助记符表示指令。汇编语言与特定的物理（或者虚拟）计算机体系结构相关，由汇编器将源代码转换为机器语言。

高级语言是独立于计算机体系结构的语言，其最大特点是使用类似自然语言的形式描述对问题的处理过程。通过编译器或者解释器将其翻译成机器语言。当前存在许多高级程序设计语言，包括 C、C++、C#、Java 和 Python 等。

计算机语言可以根据其解决问题的方法进行分类，按照程序如何处理数据的模型或者框架（即范式，paradigm），编程语言通常分为以下几类。

（1）面向过程的编程语言：在面向过程（procedural，也称为命令 imperative）的编程语言范式中，程序是一组命令。每个命令的执行都会更改与该问题相关的内存状态。FORTRAN、COBOL、Basic、Ada、PASCAL、C 等编程语言属于该范式。

（2）面向对象的编程语言：在面向对象（objected-oriented）的编程语言范式中，特定类型的数据与操作封装在一起成为一个对象（object）。C#、C++、Java、Smalltalk、Visual Basic 等编程语言属于该范式。

（3）函数式编程语言：在函数式（functional）编程语言范式中，程序是一个数学函数，将输入列表映射到输出列表。Lisp、Scheme、Haskell、F#等编程语言属于该范式。

（4）逻辑式编程语言：逻辑式（logic）编程语言范式使用一组事实和一组规则来回答查询，它基于希腊数学家定义的形式逻辑。Prolog 等编程语言属于该范式。

Python 程序设计语言属于多范式编程语言，同时支持面向对象、面向过程以及函数式编程范式。

1.1.3 计算思维和程序设计方法

人类在认识世界和改造世界过程中形成了以下三种基本的思维。

（1）逻辑思维：以推理和演绎为特征，以数学学科为代表。

（2）实验思维：以实验和验证为特征，以物理学科为代表。

（3）计算思维：以设计和构造为特征，以计算机学科为代表。

计算思维的本质是抽象（abstraction）和自动化（automation）。随着计算机的发展，计算思维已经成为求解问题的主要思维。掌握一门程序设计语言，有助于使用计算思维求解日常学习、生活和工作中的各种各样问题。

程序设计方法属于计算思维的范畴，常见的程序设计方法主要包括两种，即结构化程序设计和面向对象的程序设计。

结构化程序设计通常采用自顶向下（top-down）、逐步求精（stepwise refinement）的程序设计方法。首先从主控程序开始，然后把每个功能分解成更小的功能模块。自顶向下是一种有序的问题分解和逐步求精的程序设计方法，其特点是层次清楚、编写方便、调试容易。

自顶向下程序设计的基本思想如下。

（1）问题分解：将求解问题分解为一系列的小问题，将小问题进一步分解，直到得到可以使用算法求解的简单问题。

（2）算法实现：为分解后的可求解的简单问题设计接口和算法，并编写各个模块函数的实现程序。

（3）组合程序：将各个模块函数组合起来，完成求解问题的最终程序设计。

采用自顶向下方法设计的程序，一般可以通过自底向上（bottom-up）的方法来实现。即先实现、运行和测试每一个基本函数，再测试由基本函数组成的整体函数，这样有助于定位错误。

1.1.4 程序的编写和执行

一般使用文本编辑器编写和编辑程序。文本编辑器包括通用的文本编辑器，例如Notepad、Vim、Emacs、Sublime等，以及专用的集成开发环境，例如Visual Studio、Visual Studio Code等。专用的程序编辑器提供代码提示和编译调试功能。

使用文本编辑器编写一个程序后，将文件保存到磁盘上，包含程序代码的文件被称为源文件（source file）。

不管使用什么程序设计语言，最终都需要将源文件转换成机器语言，计算机才能理解和执行程序。将源文件转换成机器语言有以下两种转换方法。

（1）编译：编译器（compiler）将源代码翻译成目标语言。源代码一般为高级程序设计语言，而目标语言则是汇编语言或者目标机器的目标代码（机器代码）。编译器一般执行语法分析、预处理、语义分析、代码生成、代码优化等操作。编译器的主要工作流程如下：

源代码（source code）$\xrightarrow{\text{编译器（compiler）}}$目标代码（object code）$\xrightarrow{\text{链接器（linker）}}$可执行程序（executables）

（2）解释：解释器（interpreter）直接解释执行高级程序设计语言。解释器不会一次把整个程序翻译出来。它每翻译一行程序语句就立刻执行，然后翻译下一行程序语句并执行，直至完成所有程序的执行。

高级编程语言根据执行机制的不同可以分成静态语言和脚本语言两类。

采用编译方式执行的语言属于静态语言，例如C、C++、C#、Java等。静态语言的优点在于：编译后的目标代码可以直接运行；编译所产生的目标代码执行速度通常更快。

采用解释方式执行的语言属于脚本语言，例如JavaScript、PHP、Python等。脚本语言的优点在于：源代码可以在任何操作系统上的解释器中运行，可移植性好；解释执行需要保留源代码，因此程序纠错和维护十分方便。

1.2 C#语言概述

1.2.1 C#语言简介

C#起源于C语言家族，因此具有C++的功能。C#已经分别由ECMA International和ISO/IEC组织接受并确立为ECMA-334标准和ISO/IEC 23270标准。

C#采用与C、C++或Java一致的大括号（{和}）语法，简单易学。C#语法简化了C++的诸多复杂性，同时又提供了Java所不具备的许多强大功能，例如可为null的值类型、枚举、委托、Lambda表达式和直接内存访问等。

C#作为微软.NET 的主要语言，其主要发展历史如表 1-1 所示。

表 1-1　C#主要发展历史

C#版本	.NET 版本	Visual Studio 版本	发布时间
1.0	.NET Framework 1.0	Visual Studio .NET	2002/02
1.2	.NET Framework 1.1	Visual Studio .NET 2003	2003/04
2.0	.NET Framework 2.0 .NET Framework 3.0	Visual Studio 2005 Visual Studio 2005+Extension	2005/11 2006/11
3.0	.NET Framework 3.5	Visual Studio 2008	2007/11
4.0	.NET Framework 4.0	Visual Studio 2010	2010/04
5.0	.NET Framework 4.5 .NET Framework 4.5.1	Visual Studio 2012 Visual Studio 2013	2012/08 2013/10
6.0	.NET Framework 4.6 .NET Core 1.0	Visual Studio 2015	2015/07 2016/06
7.0	.NET Framework 4.6.2 .NET Core 2.0	Visual Studio 2017	2016/08 2016/08
7.1	.NET Framework 4.7	Visual Studio 2017 v15.3	2017/04
7.2	.NET Framework 4.7.1 .NET Framework 4.7.2 .NET Core 2.1 .NET Core 2.1	Visual Studio 2017 v15.5 Visual Studio 2017 v15.7	2017/10 2018/04 2018/05 2018/12
8.0	.NET Framework 4.8 .NET Core 3.0 .NET Core 3.1	Visual Studio 2019 v16.3	2019/04 2019/09 2019/12
9.0	.NET 5.0		2020/11
10.0	.NET 6.0	Visual Studio 2022	2021/11
11.0	.NET 7.0	待定	待定

本书主要基于 Visual Studio 2022/.NET 6，讲述 C# 10 语言的基础知识，以及使用 C# 10 语言的实际开发应用实例。

1.2.2　C#语言各版本的演变历史

C#语言各版本的主要演变历史及新增功能如下。

1. C# 1.0（Visual Studio.NET）：新语言诞生

C# 1.0 是为.NET Framework 设计的全新的计算机编程语言，是一种面向对象的编程语言。C# 1.0 吸收了其他编程语言的优点，改善了其他编程语言的不足。C# 1.2 主要是对 C# 1.0 的错误改善，没有大的功能增加。

2. C# 2.0（Visual Studio 2005）：泛型

C# 2.0 增加的主要特性是泛型编程能力。基于泛型，.NET Framework 增加了许多新的类库，例如 System.Collections.Generic 等。C# 2.0 的另一个突出的特性就是匿名方法。

.NET Framework 3.0 中，C#版本没有升级。.NET Framework 3.0 主要引入了 WPF 类库、WF 类库和 WCF 类库。

3. C# 3.0（Visual Studio 2008）：LINQ

.NET Framework 3.5 中，C#版本升级为 3.0，其增加的主要特性就是 LINQ。

4. C# 4.0（Visual Studio 2010）：动态编程

C# 4.0 新增 dynamic 关键字，提供动态编程功能。另外，Visual Studio 2010 提供了新的 Web 编程框架 ASP.NET MVC 2.0。

5. C# 5.0（Visual Studio 2012）：异步编程

C# 5.0 新增了 async 和 await 两个关键字，从而实现了更为便捷有效的异步编程方法。

6. C# 6.0（Visual Studio 2015）：提高编程效率

C# 6.0 增加了一些语法糖，主要包括自动属性初始化、字符串插值等，可以减少代码编写量。

7. C# 7.0（Visual Studio 2017）：提高编程效率

C# 7.0 增加了不少新特性和语法糖，主要包括元组、局部函数等，可以提升编程效率并降低出错率。

8. C# 8.0（Visual Studio 2017）：.NET Core

C# 8.0 版是专门面向.NET C# Core 的第一个主要 C#版本。一些功能依赖于新的公共语言运行时（Common Language Runtime，CLR）功能，而其他功能依赖于仅在.NET Core 中添加的库类型。

.NET Core 是开源的.NET 运行时，基于模块化的 NuGet 包，支持跨平台（各种 Windows 设备、Linux、OS X）。

9. C# 9.0（Visual Studio 2017）：.NET 5

C# 9.0 随.NET 5 一起发布。它是面向.NET 5 版本的任何程序集的默认语言版本。

C# 9.0 继续以前版本中的三大主题，即删除不必要的模式、将数据与算法分离，以及在更多位置提供更多模式。

10. C# 10.0（Visual Studio 2022）：.NET 6

C# 10.0 继续致力于删除不必要的模式、将数据与算法分离以及提高.NET 运行时的性能等主题。

1.2.3 C#特点和开发应用范围

1. C#语言特点

C#是一种现代的、面向对象的、类型安全的编程语言。C#具有下列特点：

（1）简单。C#简化了 C/C++中许多复杂的特性。例如，采用“==”比较操作，从而避免 C 语言中与赋值操作“=”的混淆错误。

（2）面向对象。C#支持数据封装、继承、多态和接口。所有的变量和方法，包括 Main 方法（应用程序的入口点），都封装在类定义中。

（3）现代。C#语言包括许多现代先进语言的特性。例如，支持属性、泛型、Lambda 表达式、垃圾回收和异常处理等。

（4）相互兼容性。C#提供对 COM 和基于 Windows 的应用程序的支持。

（5）可伸缩性和可升级性。C#的设计中充分考虑到版本控制（versioning）的需要，以确保 C#程序和库能够以兼容的方式逐步演进。

2. C#语言开发应用范围

C#语言主要用来构建在.NET Framework 上运行的各种安全、可靠的应用程序。使用 C#可以创建下列类型的应用程序和服务：

（1）控制台应用程序。基于命令行窗口的控制台（console）应用程序。

（2）桌面应用。包括 Windows 窗体应用程序、Windows Presentation Foundation（WPF）应用程序等。

（3）Web 应用。包括 ASP.NET 应用程序、ASP.NET MVC、ASP.NET Core、Web 服务等。

（4）云应用开发。基于 Microsoft Azure 的云应用程序。

（5）游戏开发。使用 Unity 的游戏开发。

（6）UWP 应用。通用 Windows 平台应用，可以运行于所有以 Windows 10 为内核的系统和设备上，包括桌面设备、移动设备、XBox、HoloLens，甚至物联网设备的应用程序。

（7）Office 平台应用程序。

（8）Windows 服务。

（9）跨平台程序。开放跨平台（Windows、Mac OSX、Linux、Android）的应用程序，支持移动设备应用和 IoT（Internet of Things，物联网）应用程序开发。

1.3 C#语言的编译和运行环境

1.3.1 C#语言与.NET Framework/.NET SDK

C#程序在.NET Framework 上运行。.NET Framework 是 Windows 的一个组件，包括一个称为公共语言运行时（Common Language Runtime，CLR）的虚拟运行环境和一组统一的类库（Framework Class Library，FCL）。

.NET Core 则一个模块化的、可跨平台的而且更加精简的运行时，它是.NET Framework 的一个跨平台子集。从.NET 5 开始，.NET Core 统一为.NET。

用 C#编写的源代码被编译为中间语言（Intermediate Language，IL）。IL 代码与资源（例如位图和字符串）一起作为一种称为程序集的可执行文件存储在磁盘上，通常具有的扩展名为.exe（应用程序）或.dll（库）。

执行 C#程序时，程序集将加载到 CLR 中，然后根据程序集清单中的信息执行不同的操作。如果符合安全要求，CLR 执行实时编译（Just-In-Time (JIT) Compilation），将 IL 代码转换为本机机器指令，并执行。CLR 还提供与自动垃圾回收、异常处理和资源管理有关的其他服务。

C#源代码文件、.NET Framework 类库、程序集和 CLR 的编译时与运行时的关系如图 1-1 所示。

注意：由 CLR 执行的代码称为“托管代码”，而直接编译为面向特定系统的本机机器语言的代码则称为“非托管代码”。

有关.NET Framework 的详细信息，请参见附录 A。

1.3.2 C#的运行环境

C#的运行环境也即.NET Framework 的运行环境。

1. .NET Framework

Windows 7 中包含了.NET Framework 3.5；Windows 10 中包含了.NET Framework 4.6；

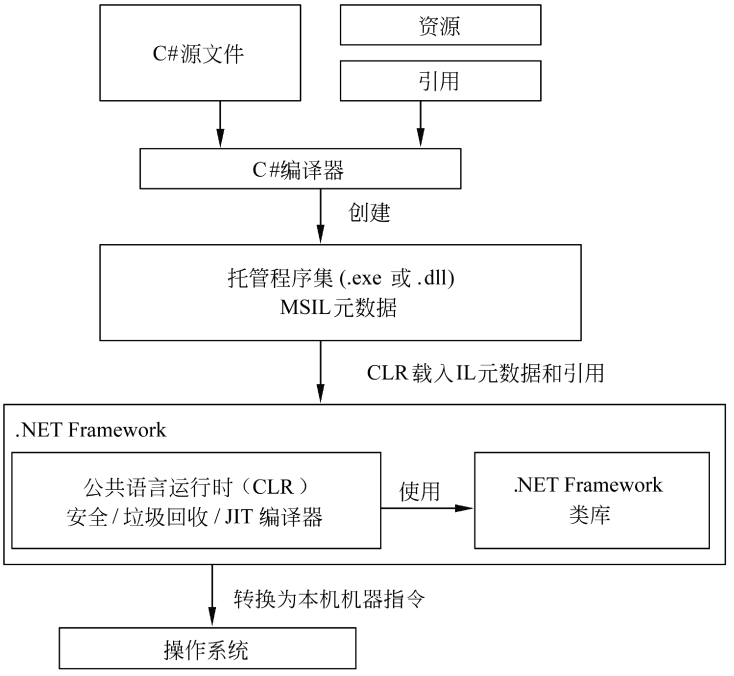

图 1-1　C#源代码的编译运行环境

Windows 10 v1703 中包含了.NET Framework 4.7。安装 Visual Studio 时，也会安装相应版本对应的.NET Framework。可以从 Microsoft 官网下载并安装最新版本的.NET Framework。

各版本的.NET Framework 包括相应的语言包，语言包支持本地化信息文本（如错误信息）的显示，每种语言对应一个语言包，可以同时安装多个语言包。

2. .NET Core/.NET

.NET Core 是新一代的开源.NET Framework 开发和运行环境，是具有跨平台（Windows、Mac OS X、Linux）能力的应用程序开发框架。

3. Mono

Mono 是由 Xamarin 公司开发的跨平台开源.NET 开发和运行环境，同样是具有跨平台（Windows、Mac OS X、Linux、Android）能力的应用程序开发框架。具体请参见官网 http://www.mono-project.com/。

1.3.3　C#的开发环境

要开发 C#应用程序，可以使用文本编辑器（如 Notepad）编写代码，并使用.NET Framework 中的编译器进行编译、运行；也可以使用微软公司集成开发工具（如 Microsoft Visual Studio）；还可以使用第三方提供的工具（如 Turbo C#）。

1. .NET Framework SDK/.NET SDK

.NET Framework/.NET SDK 软件开发工具包（SDK）包括开发人员编写、生成、测试和部署.NET Framework/.NET 应用程序时所需要的一切，如文档、示例以及命令行工具和编译器等。

2. Microsoft Visual Studio

Microsoft Visual Studio 是基于.NET Framework 开发应用程序的专业平台，提供了高级

开发工具、调试功能、数据库功能和创新功能，帮助在各种平台上快速创建当前最先进的应用程序。

本书使用下列软件组成一个完整、基于.NET 的应用系统开发运行环境。

（1）Windows 10。

（2）Microsoft Visual Studio Community 2022（社区版）。

3. Xamarin Studio

Xamarin Studio 是一个免费的.NET IDE，可以运行于 Windows、Mac OS X 和 Linux 操作系统。Xamarin Studio 提供类似于 Visual Studio Community 的功能，适合在其他操作系统平台开发构建.NET 应用程序。Xamarin Studio 和 Visual Studio 支持相互读入对方创建的项目。具体请参见官网 http://xamarin.com/。

4. Visual Studio Code

Visual Studio Code（简称 VS Code 或 VSC）是一款运行于 Mac OS X、Windows 和 Linux 之上、主要针对网页开发和云端应用的免费开源现代化轻量级代码编辑器，支持几乎所有主流的开发语言的语法高亮、智能代码补全、自定义热键、括号匹配、代码片段、代码对比 Diff、GIT 等特性，并支持插件扩展。

1.4 使用记事本创建简单的 C#程序

1.4.1 Hello World 程序

使用记事本（Notepad.exe）创建程序文件 Hello.cs（C#源文件的扩展名通常是 cs）。

【例 1.1】 Hello World 程序。

```
01  //C:\C#\Chapter01\Hello.cs    A "Hello World!" program
02  //compile: csc  Hello.cs  ->  Hello.exe
03  using System;
04  namespace CSharpBook.Chapter01
05  {
06      class HelloWorld
07      {
08          static void Main()
09          {
10              Console.WriteLine("Hello World!");
11          }
12      }
13  }
```

1.4.2 代码分析

第 1 行和第 2 行为注释。

第 3 行是一个 using 指令，引用了 System 命名空间。命名空间（namespace），也称为“名称空间”或“名字空间”，提供了一种分层的方式来组织 C#程序和库。命名空间中包含类型声明及子命名空间声明（如 System 命名空间包含 Console 类和其他的类）。这样，第 10 行就可以通过非限定方式直接使用 Console.WriteLine，以代替完全限定方式 System.Console.WriteLine。C#语句以分号（;）结束。

第 4 行定义了命名空间 CSharpBook.Chapter01。命名空间可以按层次有效地组织 C#程序中的类型，以保证其唯一性。

第 5 行和第 13 行的大括号对定义了代码块，其中的内容隶属于命名空间

CSharpBook.Chapter01。

第 6 行定义了用户自定义类 HelloWorld。

第 7 行和第 12 行的大括号对定义了代码块，其中的代码为类 HelloWorld 的实现。

第 8 行定义了类 HelloWorld 的一个成员，即名为 Main 的方法。Main 方法是使用 static 修饰符声明的静态方法，将作为程序的入口点。

第 9 行和第 11 行的大括号对定义了代码块，其中的代码为 Main 方法的实现。

第 10 行调用 Console 类的静态方法 Console.WriteLine("Hello World!")，在控制台上输出字符串"Hello World!"。System.Console 是 C#常用的类，本书范例中将大量使用其静态方法用于从控制台输入输出数据，有关 System.Console 的使用，请参见 1.9 节。

1.4.3 编译和运行结果

通过 Windows 菜单命令“开始”|“所有应用”| Visual Studio 2022 | Developer Command Prompt for VS 2022，可以打开“VS 2022 开发者命令提示”命令行窗口。

1. 使用命令行编译程序

切换到目录 C:\C#\chapter01，输入命令“csc Hello.cs”，调用 Microsoft C#编译器编译程序，如图 1-2 所示。

2. 使用命令行运行程序

编译后将产生一个名为 Hello.exe 的可执行程序集。运行结果如图 1-3 所示。

图 1-2　编译 Hello.cs 程序

图 1-3　运行 Hello.exe 程序

注意：Hello.cs 使用了 System.Console 类，默认情况下，Microsoft C#编译器自动连接 System.dll。C#语言本身不具有单独的运行时库。事实上，.NET Framework 就是 C#的运行时库。

1.5 基于集成开发环境创建简单的 C#程序

使用 Visual Studio 集成开发环境，可以更加快速、高效地开发应用程序。

Windows 应用程序通常为基于图形界面的窗口程序，窗口程序虽然友好，但涉及与图形界面有关的代码，会分散对 C#语言基础知识的理解。本书主要讲解 C#语言的基础知识，示例一般采用控制台应用程序。

使用 Visual Studio 创建控制台应用程序一般包括如下步骤。

（1）使用 Visual C#“控制台应用”模板，创建项目。

（2）使用编辑器，编写源代码程序。

（3）使用生成和调试工具，编译和运行程序。