WinForm 和 WPF 的完美统一

基于 C# 的桌面端开发技术

主编　张永财

JIYU C# DE
ZHUOMIANDUAN
KAIFA JISHU

沈阳出版发行集团
沈阳出版社

基于C#的桌面端开发技术

主编　张永财

沈阳出版发行集团
沈阳出版社

图书在版编目（CIP）数据

基于 C# 的桌面端开发技术 / 张永财主编 . -- 沈阳 : 沈阳出版社 , 2018.7

ISBN 978-7-5441-9588-1

Ⅰ . ①基… Ⅱ . ①张… Ⅲ . ① C 语言 - 程序设计 Ⅳ . ① TP312.8

中国版本图书馆 CIP 数据核字 (2018) 第 153508 号

出版发行：沈阳出版发行集团 | 沈阳出版社
（地址：沈阳市沈河区南翰林路 10 号　邮编：110011）
网　　址：http://www.sycbs.com
印　　刷：定州启航印刷有限公司
幅面尺寸：185mm × 260mm
印　　张：21.25
字　　数：480 千字
出版时间：2019 年 1 月第 1 版
印刷时间：2019 年 1 月第 1 次印刷
责任编辑：丁　昊
封面设计：优盛文化
版式设计：优盛文化
责任校对：王冬梅
责任监印：杨　旭

书　　号：ISBN 978-7-5441-9588-1
定　　价：79.00 元

联系电话：024-24112447
E - mail：sy24112447@163.com

前言

随着信息技术的广泛应用和互联网的迅猛发展，以信息产业发展水平为主要特征的综合国力竞争日趋激烈，软件产业作为信息产业的核心和国民经济信息化的基础，越来越受到世界各国的高度重视。中国加入世贸组织后，必须以积极的姿态，在更大范围和更深程度上参与国际合作和竞争。

C# 是微软公司发布的一种面向对象的、运行在 .NET Framework 上的高级程序设计语言。C# 几乎集中了所有关于软件开发和软件工程研究的最新成果：面向对象、类型安全、组件技术、自动内存管理、跨平台异常处理、版本控制、代码安全管理……是一种安全、稳定、简单、优雅、由 C 和 C++ 衍生而来的面向对象的编程语言，综合了 VB 简单的可视化操作和 C++ 的高运行效率优点，以其强大的操作能力、优雅的语法风格、创新的语言特性和便捷的对面向组件编程的支持成为 .NET 开发的首选语言。Visual Studio（简称 VS）是美国微软公司的开发工具包系列产品，是目前最流行的 Windows 平台应用程序的集成开发环境，它包括了整个软件生命周期中所需的大部分工具，如 UML 工具、代码管控工具、集成开发环境（IDE）等。

本书共分 15 章，全面地介绍了基于 C# 的桌面端开发技术，从该语言本身一直到桌面编程和云编程，再到数据源的使用等。主要内容包括：C# 的基础知识、.NET 平台与框架、C# 与 .NET 的关系、Windows 编程基础、WinForm 实训与高级控件、WPF 基础与控件、资源及样式控制、文件系统的功能结构及其操作实现、GDI+ 应用编程、三维图像、多媒体音频、流媒体技术、数据库与 LINQ 技术与应用。

本书文笔优美流畅，阐述清晰，内容全面详细，采用理论与实际相结合的方式，既包括传统和现代理论，又紧跟发展形势，研究最新的开发应用技术。

由于编者水平有限，时间又很紧，书中难免存在不妥甚至错误之处，恳请广大读者批评指正。

编者

2018.3

目录

第 1 章　C# 概述

1.1　.NET Framework 的含义

.NET Framework 是 Microsoft 为开发应用程序而创建的一个具有革命意义的平台。这句话最有趣的地方在于它的广义性，但这是有原因的。首先，注意这句话没有说“在 Windows 操作系统上开发应用程序”，尽管 .Net Framework 的 Microsoft 版本运行在 Windows 操作系统和 Windows Phone 操作系统上，但它也有运行在其他操作系统上的版本，例如 Mono，它是 .NET Framework 的开源版本（包含 C# 编译器），该版本可以运行在几个操作系统上，包括各种 Linux 版本和 Mac OS 。另外，Mono 还有一些版本可以运行在 iPhone（MonoTouch）和 Android（Mono for Android，也称为 MonoDroid）智能手机上。我们使用 .NET Framework 的一个重要原因是它可以作为集成各种操作系统的方式。

另外，上面给出的 .NET Framework 定义并未限制应用程序的类型，这是因为本来就没有限制。我们可以使用 .NET Framework 创建桌面应用程序、Windows Store 应用程序、云 / Web 应用程序、WebAPI 和其他各种类型的应用程序。另外注意，对于 Web、云和 Web API 应用程序，按照定义，它们是多平台的应用程序，因为任何带有 Web 浏览器的系统都可以访问它们。

.NET Framework 的设计方式确保它可以用于各种语言，包括本书介绍的 C# 语言，以及 C++、Visual Basic、JScript，甚至一些旧语言，如 COBOL。为此，还推出了这些语言的 .NET 版本，目前还在不断推出更多版本。所有这些语言都可以访问 .NET Framework，它们彼此之间还可以通信。C# 开发人员可以使用 Visual Basic 程序员编写的代码，反之亦然。

以上这些提供了意想不到的多样性，这也是 .NET Framework 具有诱人前景的部分原因。

1.1.1　.NET Framework 的内容

.NET Framework 主要包含一个庞大的代码库，可以在客户语言（如 C#）中通过面向对

象编程技术（OOP）来使用这些代码。这个库分为多个不同的模块，这样就可以根据希望得到的结果来选择使用其中的各个部分。例如，一个模块包含 Windows 应用程序的构件，另一个模块包含网络编程的代码块，还有一个模块包含 Web 开发的代码块。一些模块还分为更具体的子模块，例如，在 Web 开发模块中，有用于建立 Web 服务的子模块。

其目的是，不同操作系统可以根据各自的特性，支持其中的部分或全部模块。例如，智能手机支持所有的核心 .NET 功能，但不需要某些更高级的模块。

部分 .NET Framework 库定义了一些基本类型。类型是数据的一种表达方式，指定最基本类型（如 32 位带符号的整数）有助于使用 .NET Framework 的各种语言之间进行交互操作，这称为通用类型系统（Common Type System, CTS）。

除提供这个库外，.NET Framework 还包含 .NET 公共语言运行库（Common Language Runtime,CLR），它负责管理用 .NET 库开发的所有应用程序的执行。

1.1.2 使用 .NET Framework 编写应用程序

使用 .NET Framework 编写应用程序，就是使用 .NET 代码库编写代码（使用支持 Framework 的任何一种语言）。本书用 VS 进行开发，VS 是一种强大的集成开发环境，支持 C#（以及托管和非托管 C++、Visual Basic 和其他一些语言）。这个环境的优点是便于把 NET 功能集成到代码中。我们创建的代码完全是 C# 代码，但使用了 .NET Framework，并在需要时利用了 VS 中的其他工具。

为执行 C 制代码，必须把它们转换为目标操作系统能理解的语言，即本机代码（native code）。这种转换称为编译代码，由编译器执行。但在 .NET Framework 下，此过程包括两个阶段。

（1）CIL 和 JIT

在编译使用 .NET Framework 库的代码时，不是立即创建专用于操作系统的本机代码，而是把代码编译为通用中间语言（Common Intermediate Language,CIL）代码，这些代码并非专门用于任何一种操作系统，也非专门用于 C#。其他 .NET 语言（如 Visual Basic .NET）也会在第一阶段编译为这种语言。开发 C# 应用程序时，这个编译步骤由 VS 完成。

显然，要执行应用程序，必须完成更多工作，这是 Just-In-Time（JIT 编译器）的任务，它把 CIL 编译为专用于 OS 和目标机器结构的本机代码，这样 OS 才能执行应用程序。这里编译器的名称 Just-In-Time 反映了 CIL 代码仅在需要时才编译的事实。这种编译可以在应用程序的运行过程中动态发生，不过开发人员一般不需要关心这个过程。除非要编写性能十分关键的代码，否则只需知道这个编译过程会在后台自动进行，并不需要人工干预就可以了。

过去，常需要把代码编译为几个应用程序，每个应用程序都用于特定的操作系统和 CPU 结构。这通常是一种优化形式（例如，为了让代码在 AMD 芯片组上运行得更快），有时则是非常重要的（例如，使应用程序可以同时工作在 Win9x 和 WinNT/2000 环境下）。现在就没必要了，因为 JIT 编译器使用 CIL 代码，而 CIL 代码是独立于计算机、操作系统和

CPU 的。目前有几种 JIT 编译器，每种编译器都用于不同的结构，CIL 会使用合适的编译器创建所需的本机代码。

这样，开发人员需要做的工作就比较少了。实际上，可以忽略与系统相关的细节，将注意力集中在代码的功能上就够了。

（2）程序集

编译应用程序时，所创建的 CIL 代码存储在一个程序集中。程序集包括可执行的应用程序文件（这些文件可以直接在 Windows 上运行，不需要其他程序，其扩展名是 .exe）和其他应用程序使用的库（其扩展名是 .dll）。

除包含 CIL 外，程序集还包含元信息（即程序集中包含的数据的信息，也称为元数据）和可选的资源（CIL 使用的其他数据，例如，声音文件和图片）。元信息允许程序集是完全自描述的，不需要其他信息就可以使用程序集，也就是说，我们不会遇到没有把需要的数据添加到系统注册表中这样的问题，而在使用其他平台进行开发时这个问题常常出现。

因此，部署应用程序就非常简单了，只需把文件复制到远程计算机上的目录下即可。因为不需要目标系统上的其他信息，所以只需从该目录中运行可执行文件即可（假定安装了 .NET CLR）。当然，不必把运行应用程序需要的所有信息都安装到一个地方。可以编写一些代码来执行多个应用程序所要求的任务。此时，通常把这些可重用的代码放在所有应用程序都可以访问的地方。

在 .NET Framework 中，这个地方是全局程序集缓存（Global Assembly Cache, GAC），把代码放在这个缓存中是很简单的，只需把包含代码的程序集放在包含该缓存的目录中即可。

（3）托管代码

在将代码编译为 CIL，再用 JIT 编译器将它编译为本机代码后，CLR 的任务尚未全部完成，还需要管理正在执行的用 .NET Framework 编写的代码 [这个执行代码的阶段通常称为运行时（runtime）]。即 CLR 管理着应用程序，其方式是管理内存、处理安全性以及允许进行跨语言调试等。

相反，不受 CLR 控制运行的应用程序属于非托管类型，某些语言（如 C++）可以用于编写此类应用程序，例如，访问操作系统的底层功能。但是在 C# 中，只能编写在托管环境下运行的代码。我们将使用 CLR 的托管功能，让 .NET 处理与操作系统进行任何交互。

（4）垃圾回收

托管代码最重要的一个功能是垃圾回收（garbage collection）。这种 .NET 方法可确保应用程序不再使用某些内存时，就会完全释放这些内存。在 .NET 推出以前，这项工作主要由程序员负责，代码中的几个简单错误会把大块内存分配到错误的地方，使这些内存神秘失踪。这通常意味着计算机的速度逐渐减慢，最终导致系统崩溃。

.NET 垃圾回收会定期检查计算机内存，从中删除不再需要的内容。执行垃圾回收的时间并不固定，可能一秒钟内会进行数千次的检查，也可能几秒钟才检查一次，不过一定会进行检查。

这里要给程序员一些提示。因为在不可预知的时间执行这项工作，所以在设计应用程序时，必须留意这一点。需要许多内存才能运行的代码应自行完成清理工作，而不是坐等垃圾回收，但这不像听起来那样难。

（5）把它们组合在一起

在继续学习之前，先总结一下上述创建 .NET 应用程序所需的步骤：

① 使用某种 .NET 兼容语言（如 C#）编写应用程序代码，如图 1-1 所示。

② 把代码编译为 CIL，存储在程序集中，如图 1-2 所示。

图 1-1

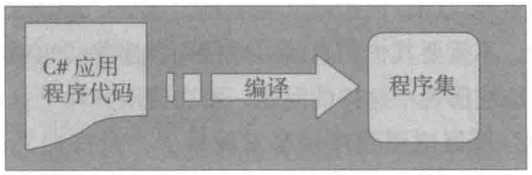

图 1-2

③ 在执行代码时（如果这是一个可执行文件，就自动运行，或者在其他代码使用它时运行），首先必须使用 JIT 编译器将代码编译为本机代码，如图 1-3 所示。

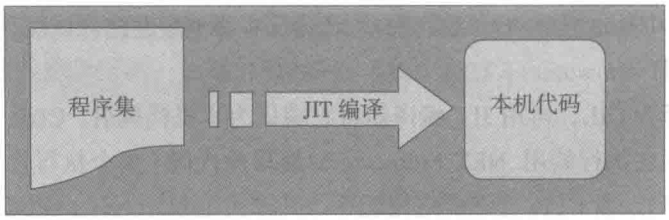

图 1-3

④ 在托管的 CLR 环境下运行本机代码，以及其他应用程序或进程，如图 1-4 所示。

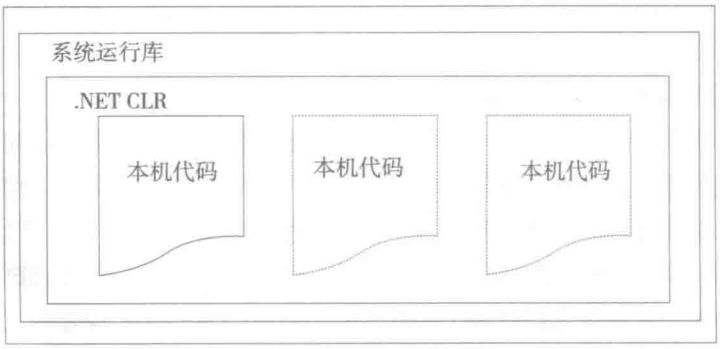

图 1-4

（6）链接

在上述过程中还有一点要注意。在第（2）步中编译为 CIL 的 C# 代码未必包含在单独文件中，可以把应用程序代码放在多个源代码文件中，再把它们编译到一个程序集中。这个过程称为链接（linking），是非常有用的。原因是处理几个较小的文件比处理一个大文件要简单得多。可以把逻辑上相关的代码分解到一个文件中，以便单独进行处理，这也更便于在需要时找到特定的代码块，让开发小组把编程工作分解为一些可管理的块，让每个人编写一小块代码，而不会破坏已编写好的代码部分或其他人正在处理的部分。

1.2 C# 的含义

如上所述，C# 是可用于创建要运行在 .NET CLR 上的应用程序的语言之一，它从 C 和 C++ 语言演化而来，是 Microsoft 专门为使用 .NET 平台而创建的。C# 吸取了以往语言失败的教训，考虑了其他语言的许多优点，并解决了它们存在的问题。

使用 C# 开发应用程序比使用 C++ 简单，因为其语法更简单。但是，C# 是一种强大的语言，在 C++ 中能完成的任务几乎都能利用 C# 完成。虽然如此，C# 中与 C++ 高级功能等价的功能（例如，直接访问和处理系统内码），只能在标记为“unsafe”的代码中使用。顾名思义，这个高级编程技术存在潜在威胁，因为它可能覆盖系统中重要的内存块，导致严重后果。因此，本书不讨论这个问题。C# 代码常比 C++ 略长一些，这是因为 C# 是一种安全类型的语言（与 C++ 不同）。在外行人看来，这表示一旦为某个数据指定了类型，就不能转换为另一个不相关的类型。所以，在类型之间转换时，必须遵守严格的规则。执行相同的任务时，用 C# 编写的代码通常比用 C++ 编写的代码长。但 C# 代码更健壮，调试起来也比较简单，.NET 始终可以随时跟踪数据的类型。在 C# 中，不能完成诸如把 4 字节的内存分配给这个数据后，我们使其有 10 个字节长，并把它解释为“X”等任务，但这并不是一件坏事。

C# 只是用于 .NET 开发的一种语言，但它是最好的一种语言。C# 的优点是，它是唯一彻头彻尾为 .NET Framework 设计的语言，是在移植到其他操作系统上的 .NET 版本中使用的主要语言。要使诸如 VB.NET 的语言尽可能类似于其以前的语言，且仍遵循 CLR，就不能完全支持 .NET 代码库的某些功能，至少需要不常见的语法。

但 C# 能使用 .NET Framework 代码库提供的每种功能。而且，.NET 的每个新版本都在 C# 语言中添加了新功能，满足了开发人员的要求，使之更强大。

如前所述，.NET Framework 没有限制应用程序的类型。C# 使用的是 .NET Framework，所以也没有限制应用程序的类型。这里仅讨论几种常见的应用程序类型。

• 桌面应用程序　这些应用程序（如 Microsoft Office）具有我们很熟悉的 Windows 外观和操作方式，使用 .NET Framework 的 Windows Presentation Foundation（WPF）模块就

可以简便地生成这种应用程序。WPF 模块是一个控件库，其中的控件（例如按钮、工具栏和菜单等）可用于建立 Windows 用户界面（UI）。

• Windows Store 应用程序　这是 Windows 8 引入的一类新的应用程序。此类应用程序主要针对触摸设备设计，通常全屏运行，侧重点在于简洁清晰。创建这类应用程序的方式有多种，包括使用 WPF 。

• 云 /Web 应用程序　.NET Framework 包括一个动态生成 Web 内容的强大系统——ASP#.NET，允许进行个性化和实现安全性等。另外，这些应用程序可以在云中驻留和访问，例如 Microsoft Azure 平台。

• Web API　这是建立 REST 风格的 HTTP 服务的理想框架，支持许多客户端，包括移动设备和浏览器。

• WCF 服务　这是一种灵活创建各种分布式应用程序的方式。使用 WCF 服务可以通过局域网或 Internet 交换几乎各种数据。无论使用什么语言创建 WCF 服务，也无论 WCF 服务驻留在什么系统上，都使用一样简单的语法。

这些类型的应用程序也可能需要某种形式的数据库访问，这可以通过 .NET Framework 的 Active Data Objects .NET（ADO#.NET） 部 分、ADO#.NET Entity Framework 或 C# 的 LINQ（Language Integrated Query）功能来实现。也可以使用许多其他资源，例如，创建联网组件、输出图形、执行复杂数学任务的工具。

1.3　编写 C# 程序

在了解了 C# 与 .NET Framework 后，接下来就是编写代码了。本书编写代码所使用的工具是 Visual Studio 2015（VS 2015），所以首先介绍这个开发环境的一些基础知识。VS 是一个庞大的复杂产品，可能会使初学者望而生畏，但使用它创建简单的应用程序是非常容易的。在本章开始使用 VS 时，不需要了解许多知识，就可以编写 C# 代码。

1.3.1　Visual Studio 2015 开发环境

在首次加载 VS 时，会立即显示选项 Sign in to Visual Studio using your Microsoft Account（用 Microsoft 账户注册 Visual Studio）。注册后，Visual Studio 设置就会在设备上同步，在多个工作站上使用 IDE 时，就不必配置它。如果没有 Microsoft 账户，可以创建一个，再使用它注册。如果不希望注册，就单击 Not now，maybe later 链接，继续 Visual Studio 的初始配置。有时建议注册，再得一个开发人员许可证。

如果是首次运行 VS ，则屏幕上会显示一个首选项列表。如果用户使用过这个开发环境的旧版本，则可以在这里做出选择，这些选择会影响到很多方面，例如窗口的布局、控制台窗口运行的方式等。所以应选择 Visual C# Development Settings ，否则会发现一些地方和

本书的描述不一样。注意，可用选项会随着安装 VS 时选择的选项而变化，但只要选择安装 C#，这个选项就是可用的。

如果不是第一次运行 VS ，但以前选择了另一个选项，也不必惊慌。为把设置重置为 Visual C# Development Settings ，只需导入它们即可。为此，单击 Tools 菜单上的 Import and Export Settings 选项，再选中 Reset all settings 选项。

单击 Next 按钮，选择是否要在继续之前保存已有的设置。如果对设置进行了定制，就保存设置，否则选择 No 按钮，再次单击 Next 按钮。在下个对话框中，选择 Visual C# 选项，用的选项可能会变化。最后单击 Finish 按钮，应用设置。VS 环境布局是完全可定制的，但默认设置更适合我们。

所有代码都显示在主窗口中。在 VS 启动时，主窗口会默认显示一个提供帮助信息的 Start Page 。主窗口可以包含许多文档，每个文档都有一个选项卡，单击文件名，就可以在文件之间切换。这个窗口也具有其他功能：它可以显示为项目设计的 GUI、纯文本文件、HTML 以及各种内置于 VS 的工具。

在主窗口的上面，有工具栏和 VS 菜单。这里有几个不同的工具栏，其功能包括：保存和加载文件、生成和运行项目，以及调试控件等。在需要使用这些工具栏时我将会讨论它们。

下面简要描述 VS 的最常用功能：

• 单击 Toolbox 选项卡时，就会显示 Toolbox 工具栏，它提供了桌面应用程序的用户界面构件等条目。另一个选项卡 Server Explorer 也可以在这里显示（通过 View I Server Explorer 菜单项选择），它包含其他许多功能，例如 Azure 订阅细节、访问数据源、服务器设置和服务等。

• Solution Explorer 窗口显示当前加载的解决方案的信息。如前所述，解决方案是一个 VS 术语，表示一个或多个项目及其配置。Solution Explorer 窗口显示了解决方案中项目的各种视图，例如项目中包含了哪些文件，这些文件中又包含了什么内容。

• Team Explorer 窗口显示了关于当前的 Team Foundation Server 或 Team Foundation Service 连接的信息，可用于使用源代码管理、bug 跟踪、自动生成等功能。但是，这是一个高级主题，本书不予介绍。

• Solution Explorer 窗口之下可以显示 Properties 窗口。稍后会看到这个窗口，因为它只在处理项目时才出现（也可以使用 View I Properties Window 菜单项切换它）。这个窗口提供了更详细的项目内容视图，允许另外配置单独元素。例如，使用这个窗口可以改变桌面应用程序中按钮的外观。

• Error List 窗口。可以使用 View I Error List 菜单项打开这个窗口，它显示了错误、警告和其他与项目有关的信息。这个窗口会持续不断地更新，但其中一些信息只有在编译项目时才出现。

这似乎需要理解很多东西，但不必担心，过不了多久就习惯了。下面首先建立第一个示例项目，它将使用上面介绍的许多 VS 元素。

1.3.2 控制台应用程序

下面分步演示如何创建一个简单的控制台应用程序。

① 选择 File|New|Project 菜单项，创建一个新的控制台应用程序项目。

② 在显示窗口的左侧选择 Visual C# 节点，在中间窗格中选择 Console Application 项目类型。把 Location 文本框改为 C:\BegVCSharp\ Chapter02（如果该目录不存在，会自动创建）。Name 文本框中的默认文本（Console Application 1）和其他设置不变。

③ 单击 OK 按钮。

④ 初始化项目后，在主窗口显示的文件中添加如下代码行：

```
Namespace Console Application1
{
    Class program
    {
      Static void Main(string[] args)
      {
          //Output text to the screen.
          Console.WriteLine（{ “The first app in Beginning Visual C# 2015!” }）;
          Console.Readkey();
      }
    }
}
```

⑤ 选择 Debug|Start Debugging 菜单项。

⑥ 按下任意键，退出应用程序（可能需要首先单击控制台窗口，以激活它）。例如，若应用了 Visual Basic Developer Settings，就会显示一个空的控制台窗口，应用程序的输出结果显示在 Immediate 窗口中。这种情况下，Console.ReadKey() 代码也会失败，显示一个错误。

示例说明

现在不仔细研究这个项目中使用的代码，而是关心如何使用开发工具来启动和运行代码。显然，VS 自动完成了许多工作，简化了编译和执行代码的过程。执行这些简单的步骤还有多种方式。例如，创建一个新项目可以像前面那样使用菜单项，也可以按下 Ctrl+Shift+N 组合键，还可以单击工具栏上的相应图标。

同样，也可以采用多种方式编译和执行代码。上面使用的方法是选择 Debug I Start Debugging 菜单项，也可以按下快捷键（F5），或者使用工具栏中的图标。使用 Debug|Start Without Debugging 菜单项（也可以按下 Ctrl+F5 组合键），还可以采用非调试模式运行代

码，使用 Build | Build Solution 菜单项或快捷键可以编译项目但不运行它（打开或关闭调试功能）。注意，执行项目但不调试，或者使用工具栏中的图标生成项目，只是这些图标在默认情况下没有显示在工具栏中。编译好代码后，在 Windows 资源管理器中运行生成的 .exe 文件，就可以执行代码。也可以在命令提示窗口中执行，为此，应打开一个命令提示窗口，把目录改为 C:\BegVCSharp\Chaptert>2\ConsoleApplicationl\ConsoleApplicationl\bin\Debug\，键入 ConsoleApplication 1，并按下回车键。

控制台应用程序会在执行完毕后立即终止，如果直接通过 IDE 运行它们，就无法看到运行结果。为解决上例中的这个问题，使用 Console.ReadKey(); 告诉代码在结束前等待按键。

（1）Solution Explorer 窗口

Solution Explorer 窗口默认位于屏幕右上角。与其他窗口一样，可把它移到任何位置，或者单击其图钉图标将它设为自动隐藏。Solution Explorer 窗口与另一个有用的窗口 Class View 位于相同的位置，使用 View|Class View 菜单项就可以显示 Class View 窗口。

Solution Explorer 窗口显示了组成 Console Application 1 项目的文件，包括我们在其中添加代码的文件 Program.cs、另一个代码文件 AssemblyInfo.cs 和多个引用。

此时不需要考虑 Assemblylnfo.cs 文件，它包含的项目是目前我们不必关心的其他信息。

使用这个窗口可以改变主窗口中显示的代码，方法是双击 .cs 文件，或右击这些文件并选择 ViewCode ，或选中它们，单击窗口顶部的工具栏按钮。还可以对这些文件执行其他操作，例如，重命名它们，或从项目中删除它们等。在该窗口中还可以显示其他类型的文件，例如，项目资源（资源是项目使用的文件，这些文件可能不是 C# 文件，如位图图像和声音文件等），可以通过同一界面处理它们。

展开代码项（例如 Program.cs）可以查看其中包含的内容。这个代码结构概览是一个很有帮助的工具，可用来直接定位到代码文件中的特定部分，而不必打开该代码文件并滚动到想要处理的部分。

References 项包含项目中使用的一个 .NET 库列表，这个列表在后面介绍，因为标准引用很适于初学者使用。Class View 窗口显示了项目的另一种视图，可以用于查看刚才创建的代码结构。单击这些窗口中的文件或其他图标，Properties 窗口的内容就会发生相应变化。

（2）Properties 窗口

使用 View I Properties Window 菜单项就可以打开 Properties 窗口。这个窗口显示了在其上面的窗口中所选的项的其他信息。例如，选择项目中的 Program.cs 文件，就会显示窗口。这个窗口还显示了其他选中项的信息，例如用户界面组件。

通常在 Properties 窗口中对项目的改变会直接影响代码，添加代码行，或改变文件中的内容。对于一些项目来说，通过这个窗口来操作与手动修改代码所用的时间是相同的。

（3）Error List 窗口

当前 Error List 窗口 (View I Error List）没有显示什么有趣的信息，这是因为应用程序