C#语言程序设计教程

(第2版) (微课版)

彭玉华　刘　艳　徐文莉　王先水　主编

清華大學出版社

北　京

内 容 简 介

本教材以项目为轴线，秉持成果导向教育理念，以 Windows 窗体应用程序开发为载体，讲解面向对象 C#语言的基础知识和 Windows 程序设计的基本技能，定位应用型人才培养。

本教材共 15 章，内容涵盖 C#语言开发环境概述、C#语言程序设计基础、字符串和数组、类和方法、继承和多态、集合和泛型、调试和异常处理、委托和事件、Windows 窗体应用程序、文件和流、进程和线程、ADO.NET 技术、数据绑定技术、三层架构学生信息管理系统实现及上机实验。

本教材理论与实践相结合，注重基础、突出应用、案例丰富、步骤完整，Windows 窗体应用程序界面设计、事件驱动后台代码设计详细具体，三层架构项目搭建、功能模块代码清晰，例题和项目均在 Visual Studio 2019 环境下测试通过。

本教材可作为高等院校计算机及相关专业的教材，也可作为计算机编程爱好者的自学用书。

图书在版编目(CIP)数据

C#语言程序设计教程 : 微课版 / 彭玉华等主编. 2 版. -- 北京 : 清华大学出版社, 2024. 6. -- ISBN 978-7-302-66436-9

Ⅰ. TP312.8

中国国家版本馆 CIP 数据核字第 202441FH15 号

责任编辑：刘金喜
封面设计：高娟妮
版式设计：芃博文化
责任校对：孔祥亮
责任印制：丛怀宇

出版发行：清华大学出版社
网　　址：https://www.tup.com.cn，https://www.wqxuetang.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-83470000　　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

印 装 者：三河市君旺印务有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张：**26.25　　**字　　数：**705 千字
版　　次：2020 年 8 月第 1 版　　2024 年 7 月第 2 版　　**印　　次：**2024 年 7 月第 1 次印刷
定　　价：79.00 元

产品编号：107747-01

前　言

C#是微软公司在 2000 年为 Visual Studio 开发平台推出的一款简洁、类型安全的面向对象编程语言，开发人员通过它可以编写在.NET Framework 上运行的各种安全可靠的应用程序，如窗体程序、Web 程序等。

本书以构建 SPOC 混合教学模式对 C#语言程序设计课程进行总体设计：课程以“准职业人”的身份，以工作过程为导向、以工作任务为基础、以学生能力为落脚点，突出培养学生的软件设计、代码编写和算法设计能力，通过课内课外双线同步实施教学，培养软件开发设计、数据库设计、ADO.NET 技术等方面的高技能与高素质应用型人才；按职业岗位能力设计五大课程模块，包括 C#语言程序设计基础模块、C#语言高级应用模块、ADO.NET 数据库访问技术模块、基于三层架构综合项目训练模块、上机实验模块。

本书的编写目的在于让学生更快、更好地理解和掌握 C#语言的每一个知识要点。本书在整理时参考了目前市面上已有的相关书籍，集各家之所长，结合作者多年的教学手稿笔记进行扩展与整理，将一些原本深奥并难以理解的开发技术思想通过一些简单的案例进行解析，让学生能够轻松掌握 C#语言程序设计思想的精髓。

本书遵循“案例驱动教学”的整体编写原则，秉持成果导向教育理念。每一个知识要点均基于一个或两个案例，通过案例来加深读者对程序设计中语法结构和算法思想的理解，设计的案例来自于作者多年的教学总结与反思，在上机实验部分体现了知识的综合应用及设计开发能力的培养。本书中的所有例题、上机实验内容均在 Visual Studio 2019 以上版本开发平台下通过测试且运行无误。在这种思想指导下，组织本书的内容如下：

第 1 章 C#开发环境概述，重点讲述.NET Framework 体系结构，Visual Studio 2019 的安装及开发 Windows 窗体程序的具体步骤；

第 2 章 C#语言程序设计基础，重点讲述 C#语言基本数据类型、运算符、常量与变量、选择语句、循环语句；

第 3 章 字符串和数组，重点讲述 C#的常用字符串、数据类型的转换、正则表达式、一维数组、枚举和结构体；

第 4 章 类和方法，重点讲述类的设计、方法的设计、构造方法及重载、属性的作用、几个常用类的属性及方法；

第 5 章 继承和多态，重点讲述继承的应用、多态的实现、抽象类和抽象方法的实现、接口的实现；

第 6 章 集合和泛型，重点讲述 ArrayList 类的属性和方法的应用、Queue 类与 Stack 类的属性和方法的应用、Hashtable 类与 SortedList 类的属性和方法的应用、泛型类、泛型方法、泛型集合的高

级应用；

第 7 章 调试和异常处理，重点讲述 try…catch…finally 形式语句的应用；

第 8 章 委托和事件，重点讲述命名方法委托、多播委托、事件；

第 9 章 Windows 窗体应用程序，重点讲述窗体属性、事件和方法、窗体中的基本控件、窗体中的对话框控件、窗体间的数据交互；

第 10 章 文件和流，重点讲述文件基本操作、流的基本应用；

第 11 章 进程和线程，重点讲述进程的基本操作、线程的基本操作；

第 12 章 ADO.NET 技术，重点讲述 ADO.NET 五大对象、使用 ADO.NET 技术操作数据库实现增删改查；

第 13 章 数据绑定技术，重点讲述数据视图控件使用代码法绑定数据的基本方法和基本应用。

第 14 章 三层架构学生信息管理系统实现，重点讲解项目的需求分析、项目总体功能结构分析、数据库设计、项目目录结构搭建、三层构架基本原理，管理员模块中用户信息添加、浏览、查询、修改、删除功能的界面设计和后台功能逻辑设计、测试。

第 15 章 上机实验，重点讲解 C#语言在今后项目开发中常用的和重要的综合知识的应用。

为便于教学，本书提供了大量的教学资源，如教学大纲、教学课件、源代码、微视频等，这些资源可通过扫描下方二维码下载。微课视频可通过扫描书中二维码观看。

教学资源下载

本书在武汉工程科技学院计算机与人工智能学院和武昌理工学院人工智能学院的大力支持下，由武汉工程科技学院计算机与人工智能学院计算机系的王先水和刘艳、武昌理工学院人工智能学院计算机科学与技术系的彭玉华、软件工程系的徐文莉四位老师共同编写完成。书中的案例全部来自于教师多年上课的手稿笔记和讲稿，同时引用了参考文献中列举的 C#语言相关书籍中的部分内容，吸取了同行的宝贵经验，在此谨表谢意。因编者水平有限，书中难免会出现欠妥之处，欢迎广大读者批评指正。

编 者

2024 年 1 月于武汉

目　录

第1章

C#开发环境概述

1.1 C#简介

C#(英文名为 C Sharp)是微软公司为配合.NET 开发平台推出的一种面向对象的、运行于.NET Framework 的编程语言，具有现代的、类型安全的、面向对象语言的基本特征，即封装性、继承性、多态性、抽象及接口，并增加了事件和委托，增强了编程的灵活性。C#凭借通用的语法和便捷的使用方法受到众多企业和开发人员的青睐。Visual Studio 是专门的一套基于组件的开发工具，主要用于.NET 平台的软件开发。

1998 年，Anders Hejlsberg(安德斯·海尔斯伯格，Delphi 和 Turbo Pascal 的设计者)与他的软件开发团队开发设计了 C#的第一个版本。2000 年 9 月，欧洲计算机制造联合会(European Computer Manufacturers Association，ECMA)成立任务小组，着力为 C#定义一个开发标准。标准是制定“一个简单、现代、通用、面向对象的编程语言”，于是发布了 ECMA-334 标准，这是一种令人满意的简洁的语言，它既借鉴了 C 语言和 C++的风格，又有类似 Java 语言的语法特征。

设计 C#是为了增强软件的健壮性，它提供了数组越界检查和“强类型”检查，并且禁止使用未初始化的变量。C#正式发布于 2002 年，伴随着 Visual Studio 开发环境一起，受到众多程序员的青睐。

C#继承 C 和 C++特征的同时又派生为一种面向对象和类型安全的编程语言，并且与.NET 开发平台完美结合，C#具有以下突出的特点。

- 语法简洁，不允许直接操作内存，去掉了指针操作。
- 完全面向对象设计，具有面向对象编程语言的基本特征，即封装性、继承性、多态性、抽象和接口等。
- 与 Web 紧密结合，C#支持绝大多数的 Web 标准，如 HTML、HTML5、XML 等。
- 强大的安全机制，可以消除软件开发中常见的语法错误，.NET 开发平台提供的垃圾回收器能够帮助开发者有效地管理内存资源。
- 兼容性。C#遵循.NET 开发平台的公共语言规范(CLS)，从而保证能够与其他语言开发的组件兼容。
- 完善的错误、异常处理机制。C#提供完善的错误和异常处理机制，更好保证了程序交付应用的健壮性。

1.2 .NET 开发平台

1.2.1 .NET Framework

.NET 平台是 Microsoft 公司在 2000 年推出的全新的应用程序开发平台，可用来构建和运行新一代 Microsoft Windows 和 Web 应用程序。它建立在开放体系结构之上，集 Microsoft 在软件领域的主要技术成就于一身。.NET 框架也是 Windows 7、Windows 8、Windows 10 操作系统的核心部件。

微软对.NET 的定义是：.NET 拥有以新方式融合计算与通信的工具和服务，它是建立于开放互联网协议标准的革命性的新平台。是一种新的跨语言、跨平台、面向组件的操作系统环境，适用于 Web 服务(Web Services)和因特网(Internet)分布式应用程序的生成、部署和运行。.NET 应用是一个运行于.NET Framework 之上的应用程序。

.NET 平台主要由.NET Framework(.NET 框架)、基于.NET 的编程语言、开发工具 Visual Studio 构成。其中，.NET Framework(.NET 框架)是基础和核心。.NET Framework 的体系结构如图 1.1 所示。

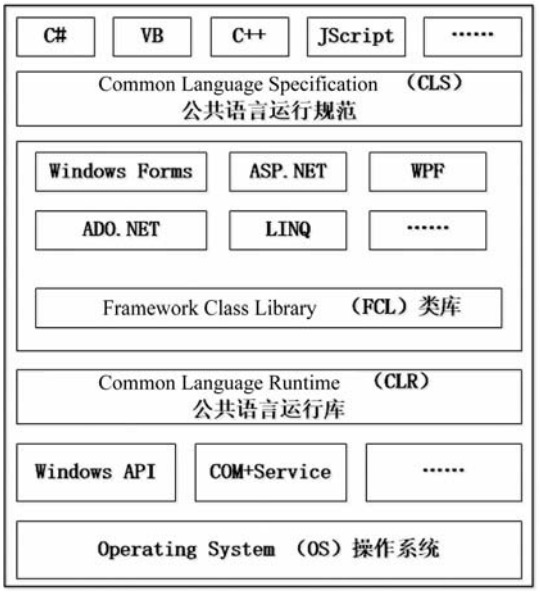

图 1.1 .NET Framework 的体系结构

.NET Framework(.NET 框架)由编程语言(如 C#、VB 等)、公共语言运行规范(CLS)、类库(开发 Windows Forms 应用程序、ASP.NET 应用程序等所用到的类库文件 FCL)、公共语言运行库(用户可以将 CLR 看作一个在执行管理代码的代码，以公共语言运行库为目标的代码称为托管代码，不以公共语言运行库为目标的代码称为非托管代码)、操作系统构成。

类库(Framework Class Library，FCL)和公共语言运行库(Common Language Runtime，CLR)两部分为.NET 框架的核心。

(1) 框架类库

框架类库为开发人员提供统一的、面向对象的、分层的可扩展的类库集，其主要部分是 BCL(Base Class Library，基类库)。通过创建跨所有编程语言的公共 API 集，公共语言运行库使得跨语言继承、错误处理和调试成为可能。框架类库采用命名空间来组织和使用。

(2) 公共语言运行库

公共语言运行库负责内存分配和垃圾回收，保证应用和底层系统的分离。公共语言运行库所负责的应用程序在运行时是托管的，具有跨语言调用、内存管理、安全性处理等优点。

1.2.2　Visual Studio 2019 集成开发环境

Visual Studio 2019(也可简称 VS 2019)是软件公司为了配合.NET 开发平台推出的开发环境，同时也是开发 C#程序的工具。

Visual Studio 2019 集成开发环境有三个版本，分别是 Community 社区版、Professional 专业版、Enterprise 企业版。其中 Visual Studio Community 社区版是完全免费的，其他两个版本是收费的。本教材采用的开发环境是 Visual Studio Community 社区版，可以到微软官方网站下载并安装。

1.2.3　Visual Studio 2019 安装的步骤

用户在安装 Visual Studio 2019 集成开发环境之前，可以在微软的官网上了解 Visual Studio 2019 中各版本的具体功能和特点、安装的操作系统要求。

1. 启动安装程序

从微软的官网上下载 Visual Studio 2019 的在线安装程序 VS_Community。在计算机联网状态下双击该程序，程序运行 Visual Studio 2019 安装界面，如图 1.2 所示。

2. 下载安装程序

在图 1.2 所示的 Visual Studio 2019 安装界面中单击“继续”按钮，则从软件官网上下载安装程序，如图 1.3 所示。

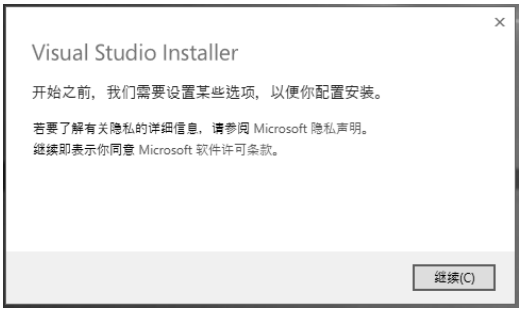

图 1.2　Visual Studio 2019 安装界面

图 1.3　下载安装程序

3. VS 开发环境选择

程序下载完毕，则出现图 1.4 所示的 VS 安装程序开发环境选择界面，根据开发需要勾选相应的开发环境，如开发 C#程序，则勾选“.NET 桌面开发”。一般选默认的安装路径，选择安装方式后，单击“安装”按钮，进入安装状态。

4. VS 安装过程

在图 1.4 中单击“安装”按钮，则程序的安装过程如图 1.5 所示。

图 1.4　VS 安装程序环境选择界面

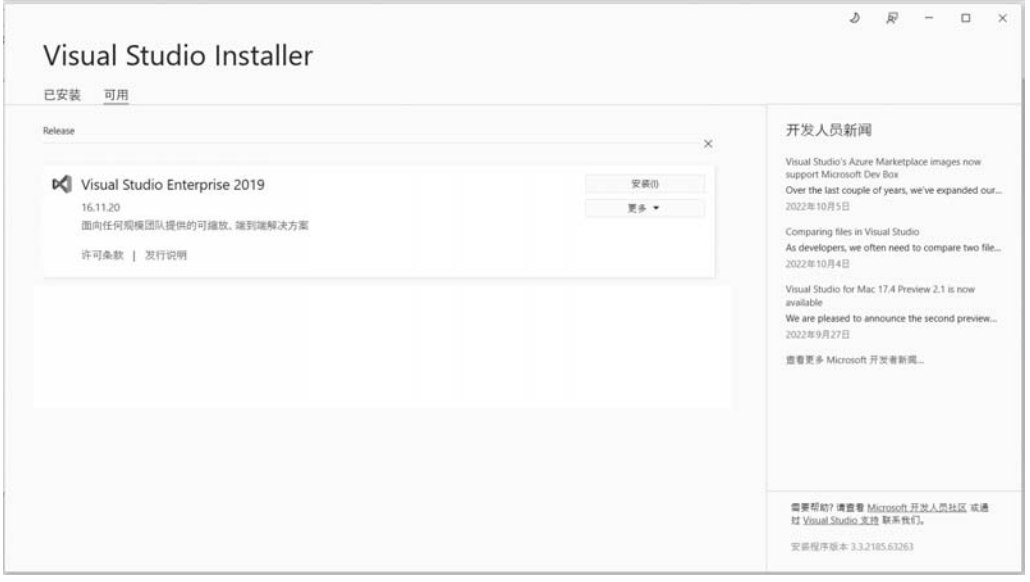

图 1.5　VS 安装程序的安装过程

1.3 Visual Studio Community 2019 的开发环境

Visual Studio Community 2019 社区版是一款非常好用的开发工具，软件除了大多数 IDE 提供的标准编辑器和调试器之外，还包括编译器、代码完成工具、图形设计器和许多其他功能，可以简化软件开发过程，适用于 Android、iOS、Windows、Web 和云开发。

1.3.1　Visual Studio Community 2019 创建项目

安装完成后，启动 Visual Studio Community 2019，出现创建或打开项目界面，如图 1.6 所示。

图 1.6　创建或打开项目界面

在图 1.6 中，直接单击“继续但无需代码(W)”，则打开 Visual Studio Community 2019 开发环境，开发环境中显示菜单栏、工具栏、工具箱面板、解决方案资源管理器面板等信息。通过执行“新建｜项目”命令，打开“创建新项目”界面。

在图 1.6 的“开始使用”中选择“创建新项目”，打开“创建新项目”界面；在“创建新项目”界面右边提供的模板中选择“空解决方案”，单击“下一步”按钮，打开“配置新项目”界面；在“配置新项目”界面中输入解决方案名称并选择解决方案的保存位置，单击“创建”按钮。Visual Studio Community 2019 创建空解决方案界面如图 1.7 所示。

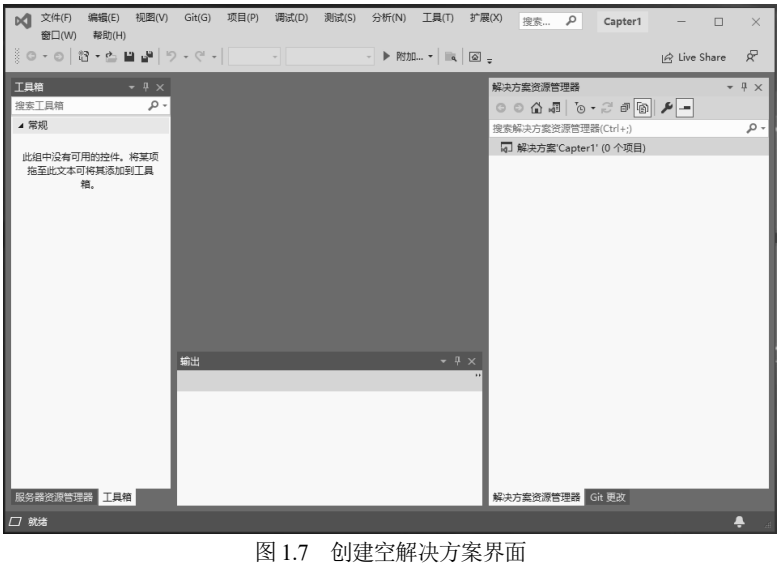

图 1.7　创建空解决方案界面

在图 1.7 中右击“解决方案 Capter1”，执行“添加｜新建项目”命令，打开“添加新项目”界面；在“添加新项目”界面中选择开发项目“控制台应用程序”模板，单击“下一步”按钮，打开“配置新项目”界面；在“配置新项目”界面中输入项目名称，单击“下一步”按钮，打开“其他信

息”界面，在“其他信息”界面上单击“创建”按钮，则创建一个 C#控制台程序，如图 1.8 所示。

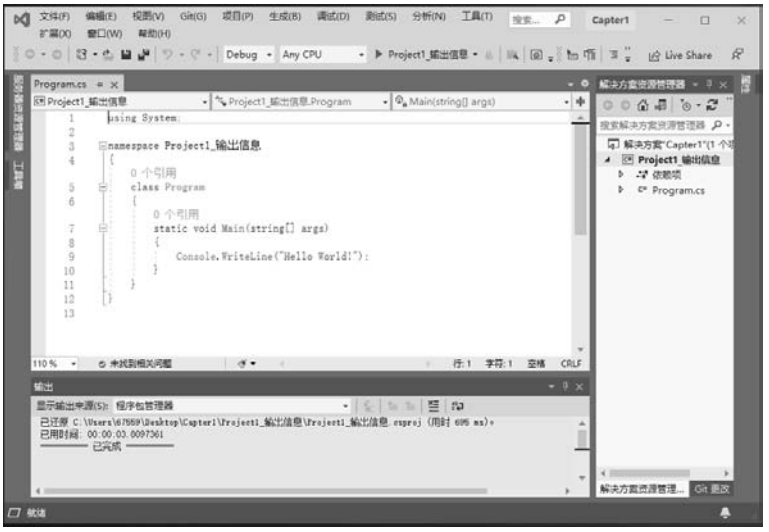

图 1.8　创建一个 C#控制台程序

1.3.2　C#程序

使用 C#语言，可以创建多种应用程序，包括控制台应用程序、Windows 窗体应用程序、WPF 应用程序、Web 应用程序和 Silverlight 应用程序。在 Visual Studio Community 2019 中，创建这些程序的基本步骤相似。本教材先创建空解决方案，然后在已创建的空解决方案中添加开发程序新项目。

【例题 1.1】 设计一个 C#控制台程序，程序功能是在控制台中输出信息：“欢迎来到 C#编程世界！”。

【操作步骤】

(1) 启动 Visual Studio 2019。

(2) 创建空解决方案。

在 Visual Studio Community 2019 的打开项目界面的“开始使用”中选择“创建新项目/继续但无需代码”，打开“创建新项目｜Visual Studio 开发环境”界面；在“创建新项目”界面右边提供的模板中选择“空解决方案”，单击“下一步”按钮，打开“配置新项目”界面；在“配置新项目”界面中输入解决方案名称“Capter1”并选择解决方案的保存位置，单击“创建”按钮，完成解决方案 Capter1 的创建。

(3) 添加新项目。

右击“解决方案 Capter1”，执行“添加｜新建项目”命令，打开“添加新项目”界面；在“添加新项目”界面上选择开发项目“控制台应用程序”模板，单击“下一步”按钮，打开“配置新项目”界面；在“配置新项目”界面上输入项目名称“Project1_输出信息”，单击“下一步”按钮，打开“其他信息”界面；在“其他信息”界面上单击“创建”按钮，则创建一个 C#控制台程序。

系统自动完成项目的配置，其中必不可少的配置是对.NET Framework 类库的引用，控制台应用程序和源代码文件是 Program.cs。新建项目“Project1_输出信息”效果如图 1.9 所示。

从图 1.9 中可以看到，项目创建好后自动生成一段程序代码，其中 Main 所在行的代码表示一个方法，在该方法中编写程序代码，并且 Main()方法是程序的主入口，程序执行时会从 Main()方法开

始执行。

图 1.9 中，第 1 行是 C#程序引用的命名空间。命名空间是一种组织 C#程序中出现的不同类的方式，可使类具有唯一的完全限定名称。一个 C#程序至少包含一个或多个命名空间，命名空间可由程序员定义，或者用先前编程写好的类库的一部分定义，或者由系统生成的项目自动生成。第 3 行是开发者定义的命名空间，也就是项目的名称。第 4 行“{”花括号和第 13 行“}”花括号是项目的组织结构，在该组织结构中可以定义类、方法。第 5 行是系统自动声明的 Program 类，在该类中可以定义字段、属性、构造函数、方法成员。第 7 行是程序的入口或静态 Main()方法，方法的返回值是 void。

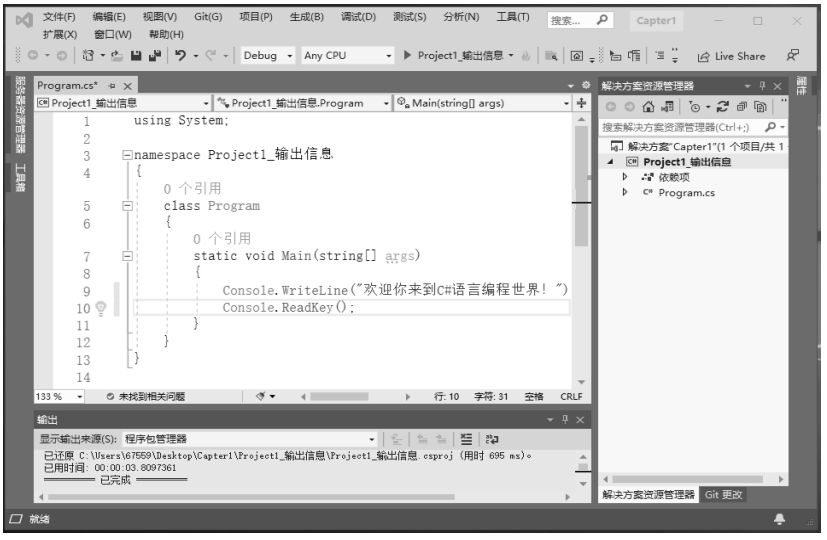

图 1.9　创建解决方案下的新项目效果

程序实现功能语句在 Main()方法的一对大括号内进行书写。语句功能是调用控制台字符读写系统静态 Console 类的 WriteLine()方法、ReadKey()方法实现信息输出。

(4) 功能代码设计。

在编写 C#程序时，程序中出现的空格、括号、逗号、分号等符号必须采用英文半角格式，否则程序会出现编译错误。设计代码参考如下。

```
using System;                          //系统自动导入命名空间

namespace Project1_输出信息        //开发者自命名空间，也称项目名称
{
    class Program                  //系统默认的类
    {
        static void Main(string[] args)//系统默认的主方法，程序运行入口地址
        {
            //开发者编写的程序代码
            Console.WriteLine("欢迎你来到 C#语言编程世界！");
            Console.ReadKey();
        }
    }
}
```

(5) 编译运行。

编译程序：完成代码设计，保存所有代码，右击解决方案，执行“生成解决方案｜重新生成解决方案”；或者右击项目名称，执行“生成｜重新生成”，以检查项目的语法错误。

运行程序：项目调试设置为“设为启动项目”调试，执行“调试｜开始调试”命令；或单击工具栏中的“启动调试”按钮；或按快捷键 F5。项目调试为指定项目，右击调试项目，执行“调试｜启动新实例”命令。

运行程序原理：使用 C#语言进行程序开发时，必须了解 C#语言程序的运行原理。程序运行机制分两步完成：① 编译期，CLR 对 C#代码进行第一次编译，将编写的代码编译成.dll 文件或.exe 文件，此时的代码被编译为中间语言；② 运行期，CLR 针对目前特定的硬件环境使用即时编译(JIT)，将中间语言编译成为本机代码并执行，把编译后的代码放入一个缓冲区中，下次使用相同的代码时，就直接从缓冲区中调用，也就是说相同的代码只编译一次，从而提高了程序运行的效率。C#程序在.NET Framework 中编译和运行的过程如图 1.10 所示。

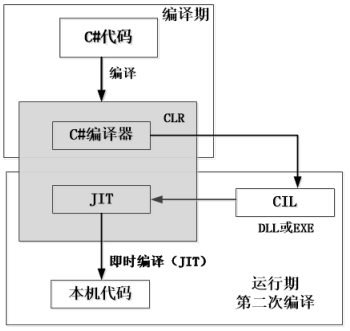

图 1.10　C#程序在.NET Framework 中编译和运行的过程

单击工具栏中的“启动”按钮后，Visual Studio 2019 将自动启动 C#语言编译器编译源程序并执行程序，最后将程序的运行结果显示在命令提示符窗口中。Project1_输出信息程序运行结果如图 1.11 所示。

图 1.11　Project1_输出信息程序运行结果

【结果分析】

对于一个 C#控制台程序来说，虽然功能单一，但包含了 C#程序的基本结构。C#语言程序基本结构如下。

- 引用命名空间：告诉编译器程序集引用的命名空间。
- 自定义命名空间：C#使用命名空间来控制源程序代码的范围，以加强源程序代码的组织管理，采用 namespace 关键字来声明。Visual Studio 2019 在创建应用程序项目时，自动使用项目名称来设置命名空间的名字，用户编写的代码就放在自定义的命名空间的一对大花括号“{}”中。
- 定义类：C#是面向对象的编程语言，C#语句必须封装在类中，一个程序至少包括一个自定义类。类的定义采用 class 关键字声明，类中成员语句放在类的一对大花括号“{}”中。

- 定义方法：C#控制台程序必须有一个 Main()方法，程序运行时，首先从 Main()方法开始执行，当最后一条语句被执行之后，程序结束运行。方法有规范的格式要求，在 Main()方法中有一个可带若干个字符串参数的数组 args。
- 编写语句：一个 C#程序由若干条语句组成，每条语句必须符合 C#语法规范，每条语句的结束标志为英文字符“；”。
- 添加代码注释：C#语言的注释有单行注释//、块注释/* */、参数化注释///。

【例题 1.2】设计一个 C#的 Windows 程序，实现功能：在姓名文本框中输入姓名“王先水”字符串，单击“显示”按钮，在 Windows 窗体的指定标签位置显示信息为“王先水：你好！欢迎学习 C#语言程序设计！”。程序运行效果如图 1.12 所示。

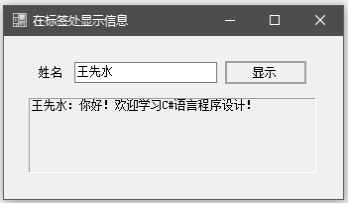

图 1.12　程序运行效果

【操作步骤】

(1) 启动 Visual Studio 2019。

(2) 创建空解决方案。

在启动 Visual Studio Community 2019 打开界面的“开始使用”中选择“创建新项目｜继续但无需代码”命令，打开“创建新项目｜Visual Studio 开发环境”界面；在“创建新项目”界面右边提供的模板中选择“空解决方案”，单击“下一步”按钮，打开“配置新项目”界面；在“配置新项目”界面中输入解决方案名称“Capter1”并选择解决方案的保存位置，单击“创建”按钮，完成解决方案 Capter1 的创建。

(3) 添加新项目。

右击“解决方案 Capter1”，执行“添加｜新建项目”命令，打开“添加新项目”界面；在“添加新项目”界面上选择开发项目 “C# Windows 窗体应用(.NET Framework)” 模板，单击“下一步”按钮，打开“配置新项目”界面；在“配置新项目”界面上输入项目名称“Project2_输出文本框信息窗体程序”，单击“下一步”按钮，打开“其他信息”界面，在“其他信息”界面上单击“创建”按钮。

单击“确定”按钮后，系统自动完成项目的配置，包括对.NET Framework 类库的引用，生成包括 Main()方法的 Program.cs 文件，生成 Windows 窗体文件 Form1.cs。

(4) 窗体界面设计。

在窗体界面上设计两个 Label 标签对象，一个表示静态提示信息标签，一个表示动态显示信息标签；一个输入信息的 TextBox 文本框对象；一个 Button 按钮对象，用于实现程序交互的“单击”事件。各控件对象的属性和属性值设置如表 1.1 所示。

表 1.1　各控件对象的属性和属性值设置

控件对象	属性	属性值	控件对象	属性	属性值
Label2	Name	lblShow	Label1	Text	姓名
	AutoSize	False	TextBox1	Name	txtName
	BorderStyle	Fixed3D	Button1	Name	btnShow
	Text	Null		Text	显示

根据表 1.1，例题 1.2 的窗体界面设计效果如图 1.13 所示。

图 1.13　例题 1.2 的窗体界面设计效果

注意：

各控件对象的 Name 属性的属性值是后台程序访问窗体界面对象的唯一标识符。

(5) 后台代码设计。

C#的Windows 应用程序开发是基于事件驱动编程，例题 1.2 中“显示”按钮事件驱动为单击事件。在窗体设计界面双击“显示”按钮或者在“显示”按钮“属性”面板中切换到“事件”面板，在“事件”面板中双击“Click”，则进入后台“显示”按钮单击事件代码设计界面，在“btnShow_Click”事件中设计实现程序功能代码，设计代码参考如下。

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace Project2_输出文本框信息窗体程序
{
    public partial class Form1 : Form
    {
        public Form1()   //构造函数
        {
            InitializeComponent();
        }
        private void btnShow_Click(object sender, EventArgs e)   //显示按钮单击事件
        {
            lblShow.Text = txtName.Text + "：你好！欢迎学习 C#语言程序设计！";
        }
    }
}
```

(6) 编译运行程序。

编译程序：完成代码设计，保存所有代码，右击解决方案，执行“生成解决方案｜重新生成解决方案”；或者右击项目名称，执行“生成｜重新生成”，以检查项目的语法错误。

运行程序：项目调试设置为“设为启动项目”调试，执行“调试｜开始调试”命令；或单击工具栏中的“启动调试”按钮▶；或按快捷键 F5。项目调试为指定项目，右击调试项目，执行“调

试｜启动新实例”命令。程序运行的结果如图 1.12 所示。

【结果分析】

在自命名空间 Project2 的 From1 类的 btnShow_Click()事件方法中编写程序代码。将文本框中输入的字符串“王先水”与字符串“：你好！欢迎学习 C#语言程序设计!”，通过连接运算符“+”连接的结果在窗体的 lblShow 标签外显示。

C# Windows 程序的 From1 类继承了基类 From 类的基本属性，如窗体的最大化、最小化、还原属性。在 From1 类中，有一个 From1 类的构造函数(在后续章节中介绍)，在构造函数中调用初始化窗体的方法 InitializeComponent()进行窗体界面设计；用“显示”按钮的单击事件方法 btnShow_Click()，用于实现程序功能代码设计。

习题 1

1. 填空题

(1) C#是一种_____类型的编程语言。

(2) .NET Framework 由编程语言、_____、_____、公共语言运行库和操作系统组成。

(3) .NET 开发平台主要由_____、_____和_____构成，其中_____是核心和基础。

(4) 托管代码是_____。

(5) C#语名必须封装在_____中，一个程序至少包括一个自定义_____。

(6) C#语言必须从_____开始执行。

(7) 一个 C#程序由若干条语句组成，每条语句的结束标志必须是_____。

(8) C#程序的注释有_____、_____和_____。

(9) Windows 窗体程序是基于_____的编程。

(10) 在 Windows 窗体程序结构中，public partial class Form1 : Form 的含义是____。

(11) 在 Windows 窗体程序结构中，下列语句的作用是_____。

```
public Form1()
{
    InitializeComponent();
}
```

(12) Console 类静态类，调用 Console 静态类的成员属性或方法的方式是_____。

(13) 控件对象中的 Name 属性的作用_____。

2. 编程题

(1) 编写一个控制台程序，求任意两个整型数据的和，以算术等式的形式输出结果。

(2) 编写一个 Windows 程序，在文本框中分别输入自己的“学号”“姓名”“专业”，单击“显示”按钮，在窗体的指定标签处分行显示相关的信息。

∞ 第2章 ∞

C#语言程序设计基础

数据类型、运算符和表达式是编程的基础，C#支持丰富的数据类型和运算符，掌握好C#的基本语法、基本数据类型、运算符、常量和变量是非常必要的。程序由若干条语句构成，任何一个程序的设计都离不开程序的基本结构(顺序、选择、循环)语句。本章将介绍C#程序设计必备的基础知识。

2.1 基本数据类型

数据类型主要是指常量和变量存储值的类型，C#是一门强类型的编程语言，它对变量的数据类型有严格的限定。在定义变量时必须声明变量的数据类型，在为变量赋值时必须赋予和变量同一类型的值，否则程序编译会报错。

C#语言的数据类型有值类型和引用类型两大类。值类型主要包括整型、浮点型、字符型、布尔型、枚举型等；引用类型主要包括字符串、数组、类、接口、委托。

从内存存储空间来看，值类型的值是存储到栈中，每次存取值都是在栈内存中操作；引用类型则会在栈中先创建一个引用变量，然后在堆中创建对象本身，再把这个对象所在内存的首地址赋给引用变量。

本节将介绍C#语言中常用的基本数据类型，包括值类型中的整型、浮点型、字符型、布尔型，引用类型中常用的字符串类型。

2.1.1 整型

整型是存储整数的类型。在C#中，为了给不同取值范围的整数合理分配存储空间，将整数类型分为4种不同的类型：字节型(byte)、短整型(short)、整型(int)、长整型(long)。4种整数类型所占存储空间大小和取值范围如表2.1所示。

在C#中默认的整型是int类型。在为一个长整型变量赋值时需要在所赋值的后面加上一个字母L或l，说明赋值为long类型，如果赋值未超出int类型的取值范围，则可以省略字母L或l。

表2.1 4种整数类型所占存储空间大小和取值范围

类型名	占用存储空间	取值范围
byte	8位(1字节)	$-2^7 \sim 2^7-1$
short	16位(2字节)	$-2^{15} \sim 2^{15}-1$

(续表)

类型名	占用存储空间	取值范围
int	32 位(4 字节)	-2^{31}～$2^{31}-1$
long	64 位(8 字节)	-2^{63}～$2^{63}-1$

2.1.2　浮点型

浮点类型是指小数类型，在 C#语言中浮点型有两种：一种称为单精度(float)浮点型；另一种称为双精度(double)浮点型。两种浮点型所占存储空间大小和取值范围如表 2.2 所示。

表 2.2　两种浮点型所占存储空间大小和取值范围

类型名	占用存储空间	取值范围
float	32 位(4 字节)	−3.4E+38～3.4E+38
double	64 位(8 字节)	±5.0E−324～±1.7E+308

在 C#中，一个小数会被默认为 double 类型的值，在赋值时可以加上字母 D 或 d，也可不加；但在为 float 类型赋值时，所赋值的后面一定要加上字母 F 或 f。在程序中也可以为浮点型变量赋整型类型值，但整型变量不可以赋浮点类型值。

2.1.3　字符型和字符串型

字符型只能存放一个字符，它占用两个字节，能存放一个汉字。字符类型的类型名用 char 关键字表示，存放 char 类型的字符需要使用英文单引号括起来，如'3'、'我'等。

字符串类型能存放多个字符，它是一个引用类型，在字符串类型中存放的字符数可以认为没有限制，因为其使用的内存大小不是固定的而是可变的。字符串类型名用 string 关键字表示，字符串类型的数据必须使用英文双引号括起来，如"欢迎来到 C#编程世界！"。

在 C#语言中还有一些特殊的字符串，代表不同的特殊作用。由于在声明字符串类型数据时需要使用英文双引号将其括起来，那么英文双引号就成了特殊字符，不能直接输出，转义字符的作用就是输出这个有特殊含义的字符。转义字符非常简单，常用的转义字符如表 2.3 所示。

表 2.3　常用的转义字符

转义字符	等价字符
\'	单引号
\\"	双引号
\\	反斜杠
\n	换行
\0	空
\a	警告(产生蜂鸣音)
\b	退格
\f	换页

(续表)

转义字符	等价字符
r	回车
\t	水平制表符
\v	垂直制表符

2.1.4 布尔类型

在 C#语言中，布尔类型使用 bool 关键字来表示，布尔类型的值只有两个：真值(true)和假值(false)。当某个值只有两种状态时可以将其声明为布尔类型。例如，判断某个数是否为奇数时，编写一个向数据库的数据表中插入一条记录的方法，判断方法的返回值若是真，说明插入一条记录成功，否则插入一条记录失败。

2.2 运算符

运算符是每一种编程语言中实现程序计算的符号，运算符主要执行程序代码的运算，如加法、减法、乘法、除法、大于、小于等。本节主要介绍算术运算符、逻辑运算符、比较运算符、赋值运算符、三元运算符及运算符的优先级别。

2.2.1 算术运算符

算术运算符是最常用的一类运算符，包括加法、减法、乘法、除法等。算术运算符的表示符号如表 2.4 所示。

表 2.4 算术运算符

运算符	功能说明
+	实现两个操作数的加法运算
–	实现两个操作数的减法运算
*	实现两个操作数的乘法运算
/	实现两个操作数的除法运算
%	实现两个操作数的模运算

在 C#语言中，如果两个字符串类型的值使用“+”运算符，则这个“+”实现的是两个字符串的连接，而不是做加法运算。在使用“/”运算符时，要注意操作数的类型，如果两个操作数的数据类型均为整数，结果相当于取整运算，不包括余数；如果两个操作数的数据类型中有一个是浮点数，则结果是正常的除法运算。在使用“%”运算符时，如果两个操作数的数据类型均为整数，则结果相当于进行除法运算，结果取其余数，经常用该运算符来判断某个数能否被某个数整除。

【例题 2.1】设计一个控制台程序，求任意两个整数的和，以算术等式的形式输出结果，程序运行效果如图 2.1 所示。

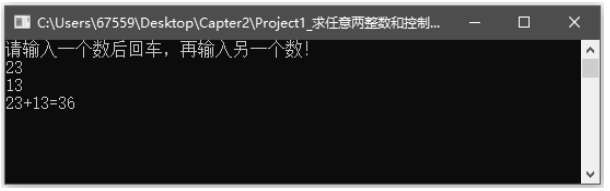

图 2.1　例题 2.1 程序运行效果

【实现步骤】

(1) 启动 Visual Studio 2019。

(2) 创建空解决方案。

在 Visual Studio 2019 开始使用界面，单击“继续但无需代码”选项，打开“Visual Studio 开发环境”界面；执行“文件｜新建｜项目”命令，打开“创建新项目”界面；在“搜索模板”中搜索“空白解决方案”，选定模板中的“空白解决方案”模板，单击“下一步”按钮，打开“配置新项目”界面；在“解决方案名称”框中输入 Capter2(若没有输入解决方案名，则系统默认为 Solution1)，在位置框中选择解决方案保存的磁盘路径，单击“创建”按钮，完成空解决方案 Capter2 的创建。

(3) 添加项目。

右击解决方案 Capter2，执行“添加｜新建项目”命令，打开“添加新项目”界面；选择“C#控制台应用程序”，单击“下一步”按钮，打开“配置新项目”界面；在项目名称框中输入项目名称为“Project1_求任意两整数和控制台程序”，单击“下一步”按钮，打开其他信息界面；单击“创建”按钮，完成项目创建，并打开控制台程序代码编辑窗口。

(4) 代码设计。

在控制台代码编辑窗口的 Main()方法中，定义两个整型变量用于接收键盘输入的数据，调用 Console 静态类的 WriteLine()方法提示输入数据信息；调用 Console 静态类的 ReadLine()方法输入数据，键盘输入的整数是数字字符串，则需调用 Convert 静态类的 ToInt32()方法将字符串转换为整型数据；调用 Console 静态类的 WriteLine()方法以格式化参数形式输出结果。代码设计参考如下。

```
using System;

namespace Project1_求任意两整数和控制台程序
{
    class Program
    {
        static void Main(string[] args)
        {
            //定义两个整型变量
            int num1;
            int num2;
            //提示用户通过键盘输入两个整数
            Console.WriteLine("请输入一个数后回车，再输入另一个数！");
            num1 = Convert.ToInt32(Console.ReadLine());
            num2 = Convert.ToInt32(Console.ReadLine());
            //以算术等式形式输出结果
            Console.WriteLine("{0}+{1}={2}", num1, num2, num1 + num2);
            Console.ReadKey();
```

```
            }
        }
    }
```

(5) 编译运行。

编译程序：完成代码设计，保存所有代码，右击解决方案，执行“生成解决方案｜重新生成解决方案”命令；或者右击项目名称，执行“生成｜重新生成”命令，以检查项目的语法错误。

运行程序：项目调试设置为“设为启动项目”调试，执行菜单“调试｜开始调试”命令；或单击工具栏中的“启动调试”按钮▶；或按快捷键 F5。项目调试为指定项目，右击调试项目，执行“调试｜启动新实例”。程序运行结果如图 2.1 所示。

【例题 2.2】设计 Windows 程序。设计要求：在第一、第二文本框中分别输入 23、12，单击“=”标签，两数运算和在第三个文本框中显示，程序运行效果如图 2.2 所示。

图 2.2　例题 2.2 运行效果

【实现步骤】

(1) 启动 Visual Studio 2019。

(2) 创建空解决方案。

在 Visual Studio 2019 开始使用界面中，单击“继续但无需代码”选项，打开“Visual Studio 开发环境”界面，执行“文件｜新建｜项目”命令，打开“创建新项目”界面；在“搜索模板”中搜索“空白解决方案”，选定模板中的“空白解决方案”模板，单击“下一步”按钮，打开“配置新项目”界面；在“解决方案名称”框中输入 Capter2(若没有输入解决方案名，则系统默认为 Solution1)，在位置框中选择解决方案保存的磁盘路径，单击“创建”按钮，完成空解决方案 Capter2 的创建。

(3) 添加项目。

右击解决方案 Capter2，执行“添加｜新建项目”命令，打开“添加新项目”界面；选择“Windows 窗体应用”，单击“下一步”按钮，打开“配置新项目”界面；在项目名称框中输入项目名称为“Project2_求任意两整数和窗体程序”，单击“下一步”按钮，打开其他信息界面；单击“创建”按钮，完成项目创建，并在开发环境窗口中显示创建的 Form1 窗体。

(4) 窗体界面设计。

在窗体界面上设计 3 个文本框对象，分别实现输入任意的两个整数和存放结果；设计两个标签对象分别表示“+”和“=”。各对象的属性及属性值设置如表 2.5 所示。

表 2.5　例题 2.2 界面各对象属性及属性值

对象名	属性	属性值	对象名	属性	属性值
TextBox1	Name	txtNum1	Label1	Text	+
TextBox2	Name	txtNum2	Label2	Text	=
TextBox3	Name	txtResult	Form1	Text	求任意两整数和

(5) 功能代码设计。

在窗体界面上，双击标签“=”，或选定标签“=”，右击执行“属性”命令，切换到事件面板，

双击“Click”事件，则打开后台代码“lblResult_Click”事件编辑框架。设计代码参考如下。

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Project2_求任意两整数和窗体程序
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void lblResult_Click(object sender, EventArgs e)
        {
            int num1 = Convert.ToInt32(txtNum1.Text);
            int num2 = Convert.ToInt32(txtNum2.Text);
            txtResult.Text = Convert.ToString(num1 + num2);
        }
    }
}
```

(6) 编译运行。

编译程序：完成代码设计，保存所有代码，右击解决方案，执行“生成解决方案｜重新生成解决方案”；或者右击项目名称，执行“生成｜重新生成”，以检查项目的语法错误。

运行程序：项目调试设置为“设为启动项目”调试，执行菜单“调试｜开始调试”命令；或执行工具栏“启动调试”按钮；或按快捷键 F5。项目调试为指定项目，右击调试项目，执行“调试｜启动新实例”。在运行窗体的第 1 个文本框中输入 23，第 2 个文本框中输入 12，单击“=”，则运行结果显示在第 3 个文本框中，程序运行效果如图 2.2 所示。

【例题 2.3】设计一个控制台程序，通过键盘输入一个任意的 4 位正整数，要求输出这个 4 位正整数的各位数字，程序实现效果如图 2.3 所示。

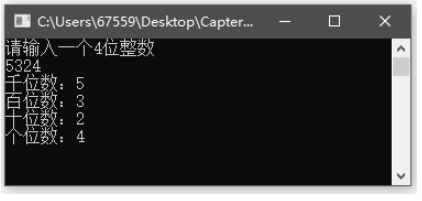

图 2.3　例题 2.3 程序运行效果

【实现步骤】

(1) 启动 Visual Studio 2019。

(2) 创建空解决方案。

在 Visual Studio 2019 开始使用界面中，单击“继续但无需代码”选项，打开“Visual Studio 开发环境”界面；执行“文件｜新建｜项目”命令，打开“创建新项目”界面；在“搜索模板”中搜索“空白解决方案”，选定模板中的“空白解决方案”模板，单击“下一步”按钮，打开“配置新项目”界面；在“解决方案名称”框中输入 Capter2(若没有输入解决方案名，则系统默认为 Solution1)，在位置框中选择解决方案保存的磁盘路径，单击“创建”按钮，完成空解决方案 Capter2 的创建。

(3) 添加项目。

右击解决方案 Capter2，执行“添加｜新建项目”命令，打开“添加新项目”界面；选择“控制台应用程序”，单击“下一步”按钮，打开“配置新项目”界面；在项目名称框中输入项目名称为“Project3_显示一个 4 位数的各位数字”，单击“下一步”按钮，打开其他信息界面，单击“创建”按钮，完成项目创建，并在开发环境窗口中显示 Program 程序类的 Main()主方法，同时系统自动完成项目的配置，其中必不可少的配置是对.NET Framework 类库的引用。

(4) 功能代码设计。

调用 Console 静态类的读写方法完成操作提示信息、写入一个 4 位整数赋给定义的整型变量 num，通过键盘写入的整数是一个字符串，因此需调用 Convert 静态类的转换方法，将字符串转换为整型数据。运用整除或求余运算分别得到千位数、百位数、十位数、个位数。设置各位数的读出格式以换行格式显示，格式中的“{}”为占位符，占位符中输出信息为格式中对应的参数，“\n”为转义字符回车换行，格式中的文本信息原样输出。设计代码参考如下。

```
using System;

namespace Project3_显示一个 4 位数的各位数字
{
    class Program
    {
        static void Main(string[] args)
        {
            int num;
            Console.WriteLine("请输入一个 4 位整数");
            num = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("千位数：{0}\n 百位数：{1}\n 十位数：{2}\n 个位数：{3}\n", num / 1000,
                num / 100 % 10, num / 10 % 10, num % 10);
            Console.ReadKey();
        }
    }
}
```

(5) 编译运行。

编译程序：完成代码设计，保存所有代码，右击解决方案，执行“生成解决方案｜重新生成解决方案”；或者右击项目名称，执行“生成｜重新生成”，以检查项目的语法错误。

运行程序：项目调试设置为启动项调试，执行“调试｜开始调试”命令；或单击工具栏中的“启动调试”按钮；或按快捷键 F5。项目调试为指定项目，右击调试项目，执行“调试｜启动新实例”命令。在程序运行字符界面中输入 5324，按 Enter 键，程序运行效果如图 2.3 所示。

【例题 2.4】设计一个 Windows 程序，实现在窗体加载时在窗体界面的指定位置显示 4321 这个

数的千位数是 4、百位数是 3、十位数是 2、个位数是 1 的信息，程序运行效果如图 2.4 所示。

图 2.4　例题 2.4 程序运行效果

【实现步骤】

(1) 启动 Visual Studio 2019。

(2) 创建空解决方案。

在 Visual Studio 2019 开始使用界面，单击“继续但无需代码”选项，打开“Visual Studio 开发环境”界面；执行“文件｜新建｜项目”命令，打开“创建新项目”界面；在“搜索模板”中搜索“空白解决方案”，选定模板中的“空白解决方案”模板，单击“下一步”按钮，打开“配置新项目”界面；在“解决方案名称”框中输入 Capter2(若没有输入解决方案名，则系统默认为 Solution1)，在位置框中选择解决方案保存的磁盘路径，单击“创建”按钮，完成空解决方案 Capter2 的创建。

(3) 添加项目。

右击解决方案 Capter2，执行“添加｜新建项目”命令，打开“添加新项目”界面；选择“Windows 窗体应用”，单击“下一步”按钮，打开“配置新项目”界面；在项目名称框中输入项目名称为“Project4_显示 4 位数各位数字窗体程序”，单击“下一步”按钮，打开其他信息界面；单击“创建”按钮，完成项目创建，并在开发环境窗口中显示创建的 Form1 窗体。系统自动完成项目的配置，其中必不可少的配置是对.NET Framework 类库的引用。

(4) 窗体界面设计。

在 Windows 窗体对象 Form1 上设计一个 Label 标签对象，同时设置 Label 标签对象的名字、自动大小、边框和显示属性的属性值。Label 标签对象属性和属性值设置如表 2.6 所示。

表 2.6　Label 标签对象属性和属性值设置

控件对象	属性	属性值
Form1	Text	显示 4 位数中千、百、十、个各位数
Label1	Name	lblShow
	AutoSize	False
	BorderStyle	Fixed3D
	Text	空

(5) 功能代码设计。

按功能要求，我们只需要设计窗体的加载事件，也就是说在程序运行时实现将 4321 这个数进行分离拆分，将其结果加载到窗体的指定标签处显示。

执行窗体加载事件的方法：第一种是直接双击窗体 Form1 的空白处；第二种是右击窗体 Form1，在弹出的下拉菜单中执行“属性”命令，在“属性”面板中单击“事件”标签，则显示窗体的所有事件，找到 Load 按钮单击即可。设计后台代码如下。

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace Project4_显示 4 位数各位数字窗体程序
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            int num = 4321;
            lblShow.Text += "\n" + "千位数是：" + num / 1000;
            lblShow.Text += "\n" + "百位数是：" + num / 100 % 10;
            lblShow.Text += "\n" + "十位数是：" + num / 10 % 10;
            lblShow.Text += "\n" + "个位数是：" + num % 10;
        }
    }
}
```

(6) 编译运行。

编译程序：完成代码设计，保存所有代码，右击解决方案，执行“生成解决方案｜重新生成解决方案”；或者右击项目名称，执行“生成｜重新生成”，以检查项目的语法错误。

运行程序：项目调试设置为“设为启动项目”调试，执行菜单“调试｜开始调试”命令；或执行工具栏中的“启动调试按钮▶”；或按快捷键 F5。项目调试为指定项目，右击调试项目，执行“调试｜启动新实例”。程序运行效果如图 2.4 所示。

【结果分析】

从程序运行结果中可以看出，定义一个整型变量 num 并对其赋值 4321，运用算术运算符的“/”除法运算和“%”求余运算，设计相应的表达式完成各个数的分离。分离结果与“\n”转义字符串、“千位数是：”字符串通过“+”连接运算符连接的结果在标签 lblShow 处显示。

若要分离任意的一个多位数，并将各位数字在窗体的指定标签处显示出来，如何设计？请读者自行完成。

2.2.2 逻辑运算符

逻辑运算符用于布尔类型数据的操作，其结果仍然是一个布尔类型数据。逻辑运算有：逻辑与、逻辑或、逻辑非，逻辑运算符如表 2.7 所示。

表 2.7　逻辑运算符

运算符	含义	说明
&&	逻辑与	运算符两边值为 true，表达式结果为 true
‖	逻辑或	运算符两边值一个为 true，表达式值为 true
!	逻辑非	表示和原来逻辑相反的逻辑

在使用逻辑运算符时需要注意逻辑运算符两边的表达式返回的值都必须是逻辑型的。判断某年是闰年的条件有两个：一个是年份能被 4 整除但不能被 100 整除，另一个是能被 400 整除，要求设计成逻辑表达式。

假设年变量是 year，判断 year 是闰年的逻辑表达式设计如下：

```
year%4==0&&year/100!=0||year%400==0
```

2.2.3　比较运算符

比较运算符是对两个数值或变量进行比较，其结果是一个布尔类型值。比较运算符包括大于、小于、等于、不等于、大于等于、小于等于，具体的比较运算符如表 2.8 所示。

表 2.8　比较运算符

运算符	含义	说明
>	大于	表示左边表达式值大于右边表达式值
<	小于	表示左边表达式值小于右边表达式值
==	等于	表示左边表达式值等于右边表达式值
!=	不等于	表示左边表达式值不等于右边表达式值
>=	大于等于	表示左边表达式值大于等于右边表达式值
<=	小于等于	表示左边表达式值小于等于右边表达式值

使用比较运算符得到的结果是布尔型的值，因此常常将使用比较运算符的表达式用到逻辑运算中。例如，判断一个数是否是偶数，需要设计一个比较运算符的表达式，该数求余 2 的结果是否等于 0.

2.2.4　赋值运算符

赋值运算符中最常用的是等号，除了等号外还有很多赋值运算符，它们通常与其他运算符连用达到简化操作的效果。赋值运算符如表 2.9 所示。

表 2.9　赋值运算符

运算符	说明
=	等号右边的值赋给等号左边的变量
+=	例如，x+=y，等同于 x=x+y
−=	例如，x−=y，等同于 x=x−y
*=	例如，x*=y，等同于 x=x*y
/=	例如，x/=y，等同于 x=x/y

2.2.5 三元运算符

三元运算符也称为条件运算符，与本章第 4 小节的选择语句中的 if 语句功能完全相似，但条件运算符能使语句更简洁。C#语言中的三元运算符只有一个，其语法形式如下：

```
布尔表达式？表达式 1:表达式 2
```

说明：

布尔表达式：判断条件，其结果是一个布尔型值的表达式。

表达式 1：如果布尔表达式的值为 true，则该三元运算符得到的结果就是表达式 1 的运算结果。

表达式 2：如果布尔表达式的值为 false，则该三元运算符得到的结果就是表达式 2 的运算结果。

例如，判断一个数 23 是偶数还是奇数，则三元运算符表示为 23%2==0?"偶数": "奇数"。

同学们自己设计一个控制台应用程序或 Windows 程序进行测试。

2.2.6 运算符的优先级

在 C#语言的表达式中使用多个运算符进行计算时，运算符的运算有先后顺序。如果改变运算符的运算顺序，则必须依靠括号。运算符的优先级如表 2.10 所示，表中显示的内容从高到低顺序排列。

表 2.10 运算符的优先级从高到低排列

运算符	结合性
.(点)、()(小括号)、[](中括号)	从左到右
++(自增)、--(自减)、！(逻辑非)	从右到左
*(乘)、/(除)、%(取余)	从左到右
+(加)、－(减)	从左到右
>、>=、<、<= 比较运算符中的大于、大于等于、小于、小于等于	从左到右
==、!= 比较运算符中的等于、不等于	从左到右
&&逻辑运算符与	从左到右
|| 逻辑运算符或	从右到左
=、+=、−=、*=、/= 赋值运算符	从右到左

尽管运算符本身有优先级，但在实际设计表达式时尽可能地使用括号来控制优先级，以增强代码的可读性。

2.3 常量和变量

常量和变量是程序语句的重要组成部分，正确定义及使用常量和变量，会使软件开发人员在编程中少犯错误，提高编程效率。

2.3.1　命名规范

命名规范是为了让整个程序代码统一，以增强可读性。每个软件公司在开发软件前都会编写一份编码规范的文档。常用的命名规范有两种：一种是 Pascal 命名法，另一种是 Camel 命名法。Pascal 命名法是指每个单词的首字母大写；Camel 命名法是指第一个单词小写，从第二个单词开始每个单词的首字母大写。

1. 变量的命名规范

变量的命名规范遵循 Camel 命名法，尽量使用能描述变量作用的英文单词，例如，描述学生信息的学号、姓名的变量可定义成 studentNo、studentName。变量的定义可用数据类型的标识符来声明。

2. 常量的命名规范

为了与变量有所区分，通常定义常量单词的所有字母大写。例如，定义求圆面积 π 的值，可以定义成一个常量以保证在整个程序中使用的值是统一的，直接定义成 PI 即可。

3. 类的命名规范

类的命名规范遵循 Pascal 命名法，即每个单词的首字母大写。例如，定义一个存放人信息的类，可以定义成 Person。

4. 接口的命名规范

接口的命名规范遵循 Pascal 命名法，接口通常以 I 开头，并将其后的每个单词的首字母大写。例如，定义一个 USB 的操作接口，可将其命名为 IUsb。

5. 方法的命名规范

方法的命名规范遵循 Pascal 命名法，一般采用动词来命名。例如，修改用户信息的操作方法，可将其命名为 UpdataUser。

6. 属性的命名规范

属性的命名规范遵循 Pascal 命名法，单词的首字母大写，如果由多个单词构成，则每个单词的首字母大写。例如，学生类中属性成员姓名 name，可将姓名属性命名为 Name。

2.3.2　声明常量

常量是指在程序执行过程中其值不发生改变的量。在 C#语言中，声明常量需要使用关键字 const，其语法形式如下：

```
const 数据类型 常量名=值;
```

说明：在定义常量时必须对其赋值，可同时声明多个常量，多个常量间用逗号隔开。程序中使用常量的好处：增强程序的可读性、便于程序的修改。

【例题 2.5】设计一个控制台程序，通过键盘输入圆的半径，在控制台以换行格式输出圆的周长和圆的面积，程序运行效果如图 2.5 所示。

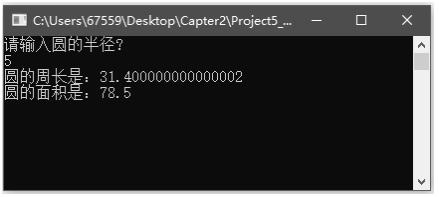

图 2.5　例题 2.5 程序运行效果

【实现步骤】

(1) 启动 Visual Studio 2019。

(2) 创建空解决方案。

在 Visual Studio 2019 开始使用界面，单击“继续但无需代码”选项，打开“Visual Studio 开发环境”界面，执行“文件｜新建｜项目”命令，打开“创建新项目”界面，在“搜索模板”中搜索“空白解决方案”，选定模板中“空白解决方案”模板，单击“下一步”，打开“配置新项目”界面，在“解决方案名称”框中输入 Capter2(若没有输入解决方案名，则系统自动为 Solution1)，在位置框中选择解决方案保存的磁盘路径，单击“创建”，完成空解决方案 Capter2 的创建。

(3) 添加项目。

右击解决方案 Capter2，执行“添加｜新建项目”命令，打开“添加新项目”界面，选择“控制台应用程序”，单击“下一步”，打开“配置新项目”界面，在项目名称框中输入项目名称为“Project5_求圆的周长和面积控制台程序”，单击“下一步”，打开其他信息界面，单击“创建”，完成项目创建，并在开发环境窗口中显示 Program 程序类的 Main()主方法，同时系统自动完成项目的配置，其中必不可少的配置是对.NET Framework 类库的引用。

(4) 功能代码设计。

在 Program 程序类的 Main()主方法中，定义一个圆半径的符号常量 PI，调用 Consoles 静态类的输入输出方法实现半径的输入和圆面积及周长的输出，同时还需调用 Convert 静态类转换方法将输入的数字字符串转换为半径的数据类型，程序代码设计参考如下。

```
using System;

namespace Project5_求圆的周长和面积控制台程序
{
    class Program
    {
        static void Main(string[] args)
        {
            const double PI = 3.14;       //定义符号常量 PI 为 3.14
            double radius;
            Console.WriteLine("请输入圆的半径?");
            radius = Convert.ToDouble(Console.ReadLine());
            Console.WriteLine("圆的周长是：{0}\n 圆的面积是：{1}", 2 * PI * radius, PI * radius * radius);
            Console.ReadKey();
        }
    }
}
```

(6) 编译运行。

编译程序：完成代码设计，保存所有代码，右击解决方案，执行“生成解决方案｜重新生成解决方案”；或者右击项目名称，执行“生成｜重新生成”，以检查项目的语法错误。

运行程序：项目调试设置为“设为启动项目”调试，执行菜单“调试｜开始调试”命令；或执行工具栏中的“启动调试”按钮；或按快捷键 F5。项目调试为指定项目，右击调试项目，执行“调试｜启动新实例”。在运行的字符界面中，通过键盘输入圆的半径 5 回车，程序运行的效果如图 2.5 所示。

【例题 2.6】设计一个 Windows 程序，实现功能：在窗体界面的文本框中输入圆半径值，单击“计算”按钮，在窗体指定的标签处显示圆的周长和面积，程序运行效果如图 2.6 所示。

图 2.6　例题 2.6 程序运行效果

(1) 启动 Visual Studio 2019。

(2) 创建空解决方案。

在 Visual Studio 2019 开始使用界面，单击“继续但无需代码”选项，打开“Visual Studio 开发环境”界面，执行“文件｜新建｜项目”命令，打开“创建新项目”界面，在“搜索模板”中搜索“空白解决方案”，选定模板中“空白解决方案”模板，单击“下一步”，打开“配置新项目”界面，在“解决方案名称”框中输入 Capter2(若没有输入解决方案名，则系统默认为 Solution1)，在位置框中选择解决方案保存的磁盘路径，单击“创建”，完成空解决方案 Capter2 的创建。

(3) 添加项目

右击解决方案 Capter2，执行“添加｜新建项目”命令，打开“添加新项目”界面，选择“Windows 窗体应用”，单击“下一步”，打开“配置新项目”界面，在项目名称框中输入项目名称为“Project6_求圆的周长和面积窗体程序”，单击“下一步”，打开其他信息界面，单击“创建”，完成项目创建，并在开发环境窗口中显示创建的 Form1 窗体。系统自动完成项目的配置，其中必不可少的配置是对.NET Framework 类库的引用。

(4) 窗体界面设计。

在 Form1 窗体界面上设计 2 个 Label 标签对象，一个是静态标签“半径”，另一个是动态标签显示圆的周长和面积信息；1 个 TextBox 文本框对象，用于输入圆的半径值；1 个 Button 按钮对象，用于单击“计算”按钮实现圆的周长和面积的计算并在动态标签处显示结果。各个控件对象的属性和属性值设置如表 2.11 所示。

表 2.11　各控件对象的属性和属性值设置

控件对象	属性	属性值	控件对象	属性	属性值
Form1	Text	计算圆的周长面积	Label2	Name	lblShow
Label1	Text	半径		AutoSize	False
Button1	Name	btnCalculate		BorderStyle	Fixed3D
	Text	计算		Text	NULL
TextBox1	Name	txtRadius	—		

(5) 功能代码设计。

在窗体界面上，右击“计算”按钮，执行“属性”命令，打开“属性”设置面板，单击“事件”标签，在“事件”列表中双击 Click 按钮，进入“计算”按钮的单击事件代码编辑区。或者直接双击“计算”按钮，也能进入“计算”按钮的单击事件代码编写辑区。在“btnCalculate_Click”事件下编写计算圆的周长和面积的功能代码，设计代码参考如下。

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace Project6_求圆的周长和面积窗体程序
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void btnCalculate_Click(object sender, EventArgs e)
        {
            const double PI = 3.14;       //定义符号常量 π 的值 3.14
            //定义半径 radius 变量获取窗体界面文本框 txtRaiuse.Text 的值，并将该值转换为 double 类型
            double radius =Convert .ToDouble ( txtRadius.Text);
            //定义周长、面积变量并计算周长、面积
            double perimeter = 2 * PI * radius;
            double acreage = PI * radius * radius;
            //采用字符串格式化语句输入周长和面积
            lblShow.Text=string.Format("输入半径：{0}\n 计算圆的周长：{1}\n 计算圆的面积：{2}",radius,
              perimeter ,acreage );
        }
    }
}
```

(6) 编译运行。

编译程序：完成代码设计，保存所有代码，右击解决方案，执行“生成解决方案｜重新生成解决方案”；或者右击项目名称，执行“生成｜重新生成”，以检查项目的语法错误。

运行程序：项目调试设置为“设为启动项目”调试，执行菜单“调试｜开始调试”命令；或执行工具栏“启动调试”按钮▶；或按快捷键 F5。项目调试为指定项目，右击调试项目，执行“调试｜启动新实例”。在运行程序窗体界面的文本框中输入 5，单击窗体界面的“计算”按钮，程序运行结果如图 2.6 所示。

【结果分析】

在程序代码设计中，定义 3 个 double 类型变量，分别是半径 radius、周长 perimeter、面积 acreage。运用数学公式计算圆的周长、面积。采用字符串格式化输出周长、面积的结果在 lblshow 处显示。

字符串格式化输出语句：

```
string.Format("输入半径：{0}\n 计算圆的周长:{1}\n:计算圆的面积: {2}",radius ,perimeter ,acreage );
```

其中，string 是 C#语言的字符串类。Format 是字符串类中的字符串格式化方法，该方法有两部分：一部分是字符串用英文双引号括起来，这部分中有说明性的字符串、转义字符串和输出信息的占位符，如"输入半径：{0}\n"；另一部分是在占位符中显示信息的变量，如 radius。

在程序运行后的窗体界面文本框中输入 3 是字符串类型数据，是引用数据类型，而定义半径变量是 double 双精度类型，是值类型。将一个字符串类型的值赋给一个双精度型变量时，C#语言中不能隐式转换，需要调用 Convert 类中的字符串转换为双精度型的方法 ToDouble()。

2.3.3　声明变量

变量是指在程序运行过程中存放数据的存储单元标识符。变量的值在程序执行过程中是可以改变的。在声明变量时，首先要明确在变量中存放的值的数据类型，再确定变量的内容，根据命名规范定义好变量名。声明变量的具体方法如下：

```
数据类型  变量名;
```

例如，定义一个存放整型数据的变量，可定义成：int number;。

定义变量后对变量赋值有两种方式：一种是在声明变量的同时直接赋值，另一种是在声明变量后再对变量赋值，其定义如下。

```
数据类型   变量名=值;
数据类型   变量名;
变量名=值;
```

在声明变量后，对变量赋值要注意所赋的值与定义变量的数据类型相兼容。定义变量时，变量赋值时也可一次为多个变量赋值，各个变量间用“,”隔开，例如，double a=3.14, b=4.12;。

【例题 2.7】设计一个 Windows 程序，在 Windows 窗体上指定标签外显示两个整型数据交换前、交换后的数据信息，程序运行效果如图 2.7 所示。

图 2.7　例题 2.7 程序运行效果

【实现步骤】

(1) 启动 Visual Studio 2019。

(2) 创建空解决方案。

在 Visual Studio 2019 开始使用界面，单击“继续但无需代码”选项，打开“Visual Studio 开发环境”界面，执行“文件｜新建｜项目”命令，打开“创建新项目”界面，在“搜索模板”中搜索“空白解决方案”，选定模板中“空白解决方案”模板，单击“下一步”，打开“配置新项目”界面，在“解决方案名称”框中输入 Capter2(若没有输入解决方案名，则系统默认为 Solution1)，在位置框

中选择解决方案保存的磁盘路径，单击“创建”，完成空解决方案 Capter2 的创建。

(3) 添加项目。

右击解决方案 Capter2，执行“添加｜新建项目”命令，打开“添加新项目”界面；选择“Windows 窗体应用”，单击“下一步”，打开“配置新项目”界面，在项目名称框中输入项目名称为“Project7_两数交换窗体程序”，单击“下一步”，打开其他信息界面，单击“创建”，完成项目创建，并在开发环境窗口中显示创建的 Form1 窗体。系统自动完成项目的配置，其中必不可少的配置是对.NET Framework 类库的引用。

(4) 窗体界面设计。

在 Form1 窗体界面上设计 1 个 Label 对象，并设置 Label 对象的相关属性。Label 对象的属性及属性值如表 2.12 所示。

表 2.12 Label 对象的属性及属性值

控件对象	属性	属性值
Form1	Text	显示两个整型数据交换前后的信息
Label1	Name	lblShow
	AutoSize	False
	BorderStyle	Fixed3D
	Text	空

(5) 功能代码设计。

在 Form1 窗体界面上，右击 Form1 窗体，执行“属性”命令，打开“属性”设置面板，单击“事件”标签，在“事件”面板列表中双击 Load 按钮，进入事件代码编辑界面，在“Form1_Load”窗体加载事件中编辑代码，代码设计参考如下。

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Project7_两数交换
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            int num1 = 23;
            int num2 = 36;
```

```
                lblShow.Text = "两数交换前：";
                lblShow.Text += "\n" + string.Format("num1={0};num2={1}", num1, num2);
                lblShow.Text += "\n" + "两数交换后：";
                //设计两数交换的算法
                int temp;
                temp = num1;
                num1 = num2;
                num2 = temp;
                lblShow.Text += "\n" + string.Format("num1={0};num2={1}", num1, num2);
            }
        }
    }
```

(6) 编译运行。

编译程序：完成代码设计，保存所有代码，右击解决方案，执行“生成解决方案｜重新生成解决方案”；或者右击项目名称，执行“生成｜重新生成”，以检查项目的语法错误。

运行程序：项目调试设置为“设为启动项目”调试，执行菜单“调试｜开始调试”命令；或执行工具栏“启动调试”按钮；或按快捷键 F5。项目调试为指定项目，右击调试项目，执行“调试｜启动新实例”。程序运行结果如图 2.7 所示。

【结果分析】

在 Form1 窗体的加载事件中设计两数交换的代码。在程序中定义两个整型变量 num1、num2，在声明变量时对其赋值，同时还声明一个中间变量 temp，实现两数交换过渡。采用字符串格式法将信息在 Form1 窗体的指定标签位置上输出。

同学们思考两数交换不引入第三个数的算法如何实现？请自己编程测试；如果要实现任意的两个数交换，如何实现，请读者自己编程测试。

2.4 选择语句

选择语句是用于根据某些条件来选择执行不同操作的语句。例如，学生登录学校教务系统时，需要输入用户名和密码，若用户名存在和密码正确，则能进入教务系统查看自己的成绩，否则，学生就无法进入学校教务系统。这里的用户名和密码的存在就是条件。本节将介绍 C#语言中的条件语句：if 语句和 switch 语句。

2.4.1 if 语句

if 语句是最常用的条件语句，并且 if 语句形式有多种，包括单分支条件 if 语句、双分支条件 if 语句、多分支条件 if 语句。

1. 单分支条件 if 语句

单分支条件 if 语句是最简单的 if 语句，只有满足语句中的条件才能执行相应的语句。具体语法格式如下：